МАТЕРИАЛЫ ЗАДАНИЙ
Всероссийской междисциплинарной олимпиады школьников 8–11 класса «Национальная технологическая олимпиада»

по профилю «Автономные транспортные системы»

2024/25 учебный год

ntcontest.ru

УДК 373.5.016:[656:681.5]
ББК 74.263.0
А22

Авторы:
В. С. Белкин, И. Е. Даниленко, О. В. Зубков, А. В. Коротицкий, М. В. Корячко, Н. Ю. Кузнецов, А. В. Харитонова

А22 Всероссийская междисциплинарная олимпиада школьников 8–11 класса «Национальная технологическая олимпиада». Учебно-методическое пособие
Том 2 **Автономные транспортные системы**
— М.: Ассоциация участников технологических кружков, 2025. — 232 с.

ISBN 978-5-908021-01-2

Данное пособие разработано коллективом авторов на основе опыта проведения всероссийской междисциплинарной олимпиады школьников 8–11 класса «Национальная технологическая олимпиада» в 2024/25 учебном году, а также многолетнего опыта проведения инженерных соревнований для школьников. В пособии собраны основные материалы, необходимые как для подготовки к олимпиаде, так и для углубления знаний и приобретения навыков решения инженерных задач.

В издании приведены варианты заданий по профилю Национальной технологической олимпиады за 2024/25 учебный год с ответами, подробными решениями и комментариями. Пособие адресовано учащимся 8–11 классов, абитуриентам, школьным учителям, наставникам и преподавателям учреждений дополнительного образования, центров молодежного и инновационного творчества и детских технопарков.

Методические материалы также могут быть полезны студентам и преподавателям направлений, относящихся к группам:
01.00.00 Математика и механика
02.00.00 Компьютерные и информационные науки
09.00.00 Информатика и вычислительная техника
11.00.00 Электроника, радиотехника и системы связи
15.00.00 Машиностроение
23.00.00 Техника и технологии наземного транспорта
24.00.00 Авиационная и ракетно космическая техника
25.00.00 Аэронавигация и эксплуатация авиационной и ракетно-космической техники
27.00.00 Управление в технических системах

ISBN 978-5-908021-01-2

9 785908 021012 >

УДК 373.5.016:[656:681.5]
ББК 74.263.0

Оглавление

1. Введение

1.1. Национальная технологическая олимпиада

Всероссийская междисциплинарная олимпиада школьников 8–11 класса «Национальная технологическая олимпиада» (далее — Олимпиада, НТО) проводится в соответствии с распоряжением Правительства Российской Федерации от 10.02.2022 № 211-р при координации Министерства науки и высшего образования Российской Федерации и при содействии Министерства просвещения Российской Федерации, Министерства цифрового развития, связи и массовых коммуникаций Российской Федерации, Министерства промышленности и торговли Российской Федерации, Ассоциации участников технологических кружков, Агентства стратегических инициатив по продвижению новых проектов, АНО «Россия — страна возможностей», АНО «Платформа Национальной технологической инициативы» и Российского движения детей и молодежи «Движение Первых».

Проектное управление Олимпиадой осуществляет структурное подразделение Национального исследовательского университета «Высшая школа экономики» — Центр Национальной технологической олимпиады. Организационный комитет по подготовке и проведению Национальной технологической олимпиады возглавляют первый заместитель Руководителя Администрации Президента Российской Федерации С. В. Кириенко и заместитель Председателя Правительства Российской Федерации Д. Н. Чернышенко.

Национальная технологическая олимпиада — это командная инженерная Олимпиада, позволяющая школьникам работать в самых передовых инженерных направлениях. Она базируется на опыте Олимпиады Кружкового движения НТИ и проводится с 2015 года, а с 2016 года входит в перечень Российского совета олимпиад школьников и дает победителям и призерам льготы при поступлении в университеты.

Всего заявки на участие в десятом юбилейном сезоне (2024–25 гг.) самых масштабных в России командных инженерных соревнованиях подали более 140 тысяч школьников. Общий охват олимпиады с 2015 года превысил 880 тысяч участников.

НТО способствует формированию профессиональной траектории школьников, увлеченных научно-техническим творчеством и помогает им:

- определить свой интерес в мире современных технологий;
- получить опыт решения комплексных инженерных задач;
- осознанно выбрать вуз для продолжения обучения и поступить в него на льготных условиях.

Кроме того, НТО позволяет каждому участнику познакомиться с перспективными направлениями технологического развития, ведущими экспертами и найти единомышленников.

Ценности НТО

Национальная технологическая олимпиада — командные инженерные соревнования для школьников и студентов. Олимпиада создает уникальное пространство, основанное на общих ценностях и смыслах, которыми делятся все участники процесса: школьники, студенты, организаторы, наставники и эксперты. В основе Олимпиады лежит представление о современном технологическом образовании как новом укладе жизни в быстро меняющемся мире. Эта модель предполагает:

- доступность качественного обучения для всех, кто стремится к знаниям;
- возможность непрерывного развития;
- совместное формирование среды, где гуманитарные знания и новые технологии взаимно усиливают друг друга.

Это — образ общества будущего, в котором участники Олимпиады оказываются уже сегодня.

Решать прикладные задачи, нацеленные на умножение общественного блага

В заданиях Олимпиады используются актуальные вызовы науки и технологий, адаптированные под уровень школьников. Они имеют прикладной характер и отражают реальные потребности общества, а системное и профессиональное решение подобных задач способствует развитию общего блага. Олимпиада предоставляет возможность попробовать себя в этом направлении уже сегодня и найти единомышленников.

Создавать, а не только потреблять

Стремление к созданию нового ценится выше потребления готового, а ориентация на общественную пользу — выше личной выгоды. Это не исключает заботу о собственных интересах, но подчеркивает: творчество приносит больше удовлетворения, чем пассивное потребление. Олимпиада — совместный труд организаторов, партнеров и участников, в котором важнее стремление решать общие задачи, чем критика чужих усилий.

Работать в команде

Командная работа рассматривается не только как эффективный способ достижения целей, но и как основа для формирования сообщества, объединенного общими ценностями. Команда помогает раскрыть индивидуальность каждого, при этом сохраняя уважение к другим. Такие горизонтальные связи необходимы для реализации амбициозных технологических проектов. Олимпиада способствует формированию подобного сообщества и приглашает к его созданию всех заинтересованных.

Осваивать и ответственно развивать новые технологии

Сообщество Национальной технологической олимпиады — часть Кружкового движения НТИ, объединенные интересом к современным технологиям, стремлением

к их пониманию и созданию нового. Возможности технологий постоянно расширяются, однако развитие должно сопровождаться ответственностью. Этика инженера и ученого предполагает осознание последствий своих решений. Главное правило — создавая новое, не навредить.

Играть честно и пробовать себя

Ценится честная победа, достигнутая в рамках установленных правил. Это предполагает отказ от списывания, давления и манипуляций. Честная игра означает уважение к себе, команде и соперникам. Олимпиада поддерживается как безопасное пространство, где каждый может пробовать новое, не опасаясь ошибок, и постепенно становиться сильнее и увереннее в себе.

Быть человеком

Соревнования — это сложный и эмоционально насыщенный процесс, в котором особенно важны порядочность, вежливость и чуткость. Эмпатия, уважение и забота делают участие полезным и комфортным. Высоко ценится бережное отношение к людям и их труду, отказ от токсичной критики и готовность нести ответственность за слова и поступки. Участие в общем деле помогает не только окружающим, но и самому человеку.

Организационная структура НТО

НТО — межпредметная олимпиада. Спектр соревновательных направлений (профилей НТО) сформирован на основе актуального технологического пакета и связан с решением современных проблем в различных технологических отраслях. С полным перечнем направлений (профилей) можно ознакомиться на сайте НТО: https://ntcontest.ru/tracks/nto-school/.

Соревнования в рамках НТО проводятся по четырем трекам:

1. НТО Junior для школьников (5–7 классы).
2. НТО школьников (8–11 классы).
3. НТО студентов.
4. Конкурс цифровых портфолио «Талант НТО».

В 2024/25 учебном году 21 профиль НТО включен в Перечень олимпиад школьников, ежегодно утверждаемый Приказом Министерства науки и высшего образования Российской Федерации, а также в Перечень олимпиад и иных интеллектуальных и (или) творческих конкурсов, утверждаемый приказом Министерства просвещения Российской Федерации. Это дает право победителям и призерам профилей НТО поступать в вузы страны без вступительных испытаний (БВИ), получить 100 баллов ЕГЭ или дополнительные 10 баллов за индивидуальные достижения. Преимущества при поступлении победителям и призерам НТО предлагают более 100 российских вузов.

НТО для школьников 8–11 классов проводится в три этапа:

- Первый отборочный этап — заочный индивидуальный. Участникам предлагаются предметный тур, состоящий из задач по двум предметам, связанным

с выбранным профилем, а также инженерный тур, задания которого погружают участников в тематику профиля; образовательный модуль формирует теоретические знания и представления.

- Второй отборочный этап — заочный командный. На этом этапе участники выполняют как индивидуальные задания на проверку компетенций, так и командные задачи, соответствующие выбранному профилю.
- Заключительный этап — очный командный. В течение 5–6 дней команды участников со всей страны, успешно прошедшие оба отборочных этапа, соревнуются в решении комплексных прикладных инженерных задач.

Профили НТО 2024/25 учебного года и соответствующий уровень РСОШ

Профили II уровня РСОШ:

- Автоматизация бизнес-процессов.
- Автономные транспортные системы.
- Беспилотные авиационные системы.
- Водные робототехнические системы.
- Инженерные биологические системы.
- Наносистемы и наноинженерия.
- Нейротехнологии и когнитивные науки.
- Технологии беспроводной связи.
- Цифровые технологии в архитектуре.
- Ядерные технологии.

Профили III уровня РСОШ:

- Анализ космических снимков и геопространственных данных.
- Аэрокосмические системы.
- Большие данные и машинное обучение.
- Геномное редактирование.
- Интеллектуальные робототехнические системы.
- Интеллектуальные энергетические системы.
- Информационная безопасность.
- Искусственный интеллект.
- Летающая робототехника.
- Спутниковые системы.
- Кластер «Виртуальные миры»:
    - ◇ Разработка компьютерных игр.
    - ◇ Технологии виртуальной реальности.
    - ◇ Технологии дополненной реальности.

Профили без уровня РСОШ:

- Инфохимия.
- Квантовый инжиниринг.
- Новые материалы.
- Программная инженерия в финансовых технологиях.

- Современная пищевая инженерия.
- Умный город.
- Урбанистика.
- Цифровые сенсорные системы.
- Разработка мобильных приложений.

Обратите внимание на то, что в олимпиаде 2025/26 учебного года список профилей, в т. ч. входящих в РСОШ, и уровни РСОШ могут поменяться.

Участие в НТО старшеклассников может принять любой школьник, обучающийся в 8–11 классе. Чаще всего Олимпиада привлекает:

- учащихся технологических кружков, интересующихся инженерными и робототехническими соревнованиями;
- школьников, увлеченных олимпиадами и предпочитающих межпредметный подход;
- энтузиастов передовых технологий;
- активных участников хакатонов, проектных конкурсов и профильных школ;
- будущих предпринимателей, ищущих команду для реализации стартап-идей;
- любознательных школьников, стремящихся выйти за рамки школьной программы.

Познакомить школьников с НТО и ее направлениями, а также мотивировать их на участие в Олимпиаде можно с помощью специальных мероприятий — Урока НТО и Дней НТО. Методические рекомендации для педагогов по проведению Урока НТО и организации Дня НТО в образовательной организации размещены на сайте: https://nti-lesson.ru. Здесь можно подобрать и скачать готовые сценарии занятий и подборки материалов по различным направлениям Олимпиады.

Участвуя в НТО, школьники получают возможность работать с практико-ориентированными задачами в области прорывных технологий, собирать команды единомышленников, погружаться в профессиональное сообщество, а также заработать льготы для поступления в вузы.

По всей стране работают площадки подготовки к НТО, которые помогают привлекать участников и проводят мероприятия по подготовке к этапам Олимпиады. Такие площадки могут быть открыты на базе:

- школ и учреждений дополнительного образования;
- частных кружков по программированию, робототехнике и другим технологическим направлениям;
- вузов;
- технопарков и других образовательных и научно-технических организаций.

Любое образовательное учреждение, ученики которого участвуют в НТО или НТО Junior, может стать площадкой подготовки к Олимпиаде и присоединиться к Кружковому движению НТИ. Подробные инструкции о том, как стать площадкой подготовки, размещены на сайте: https://ntcontest.ru. Условия регистрации и требования к ним актуализируются с развитием Олимпиады, а обновленная информация публикуется перед началом каждого нового цикла.

Наставники НТО

В Национальной технологической олимпиаде большое внимание уделяется работе с **наставниками** — людьми, сопровождающими участников на всех этапах подготовки и участия в Олимпиаде. Наставник оказывает поддержку как в решении организационных вопросов, так и в развитии технических и социальных навыков школьников, включая умение работать в команде.

Наставником НТО может стать любой взрослый, готовый помогать школьникам развиваться и готовиться к участию в инженерных соревнованиях. Это может быть:

- учитель школы или преподаватель вуза;
- педагог дополнительного образования;
- руководитель кружка;
- родитель школьника;
- специалист из технологической области или представитель бизнеса.

Даже если наставник сам не обладает достаточными знаниями в определенной области, он может привлекать к подготовке коллег и экспертов, а также оказывать поддержку и организовывать процесс обучения для самостоятельных учеников. Сегодня сообщество наставников НТО насчитывает более **7 000 человек** по всей стране.

Главная цель наставника — **организовать системную подготовку к Олимпиаде в течение всего учебного года**, поддерживать интерес и мотивацию участников, а также помочь им справляться с возникающими трудностями. Также наставник фиксирует цели команды и каждого участника, чтобы в дальнейшем можно было проанализировать развитие профессиональных и личных компетенций.

Основные направления работы наставника

Организационные задачи:

- Информирование и мотивация: наставник рассказывает учащимся об НТО, ее этапах и преимуществах, помогает с выбором подходящего профиля, ориентируясь на интересы и способности школьников.
- Составление программы подготовки: формируется расписание и план занятий, организуется работа по освоению необходимых знаний и навыков.
- Контроль сроков: наставник следит за календарем Олимпиады и напоминает участникам о сроках решения заданий отборочных этапов.

Содержательная подготовка:

- Оценка компетенций участников: наставник помогает определить сильные и слабые стороны учеников и подбирает задания и материалы для устранения пробелов.
- Подготовка к отборочным этапам: помощь в изучении рекомендованных материалов, заданий прошлых лет, онлайн-курсы по профилям.
- Подготовка к заключительному этапу: разбираются задачи заключительных этапов прошлых лет, отслеживаются подготовительные мероприятия (очные и дистанционные), в которых наставник рекомендует ученикам участвовать.

Развитие личных и командных навыков:

- Формирование команд: наставник помогает сформировать сбалансированные команды для второго отборочного и финального этапов, распределить роли, при необходимости ищет участников из других регионов и организует онлайн-коммуникацию.
- Анализ прогресса и опыта: после каждого этапа проводится совместная рефлексия, обсуждаются успехи и трудности, выявляются зоны роста и направления для дальнейшего развития.
- Поддержка и мотивация: наставник поддерживает интерес и энтузиазм участников (особенно в случае неудачных результатов), помогает справиться с разочарованием и сохранить настрой на дальнейшее участие.
- Построение индивидуальной образовательной траектории: наставник помогает школьникам осознанно планировать дальнейшее обучение: выбирать курсы, участвовать в конкурсах, определяться с вузами и направлениями подготовки.

Поддержка наставников НТО

Работе наставников посвящен отдельный раздел на сайте НТО: https://ntcontest.ru/mentors/.

Для систематизации знаний и подходов к работе наставников в рамках инженерных соревнований разработан курс «Дао начинающего наставника: как сопровождать инженерные команды»: https://stepik.org/course/124633/. Курс формирует общие представления об их работе в области подготовки участников к инженерным соревнованиям.

Для совершенствования профессиональных компетенций по направлениям профилей создан курс «Дао начинающего наставника: как развивать технологические компетенции»: https://stepik.org/course/186928/.

Для организации занятий с учениками педагогам предлагаются образовательные программы, разработанные на основе многолетнего опыта организации подготовки к НТО. В настоящий момент они представлены по передовым технологическим направлениям:

- компьютерное зрение;
- геномное редактирование;
- водная, летающая и интеллектуальная робототехника;
- машинное обучение и искусственный интеллект;
- нейротехнологии;
- беспроводная связь, дополненная реальность.

Программы доступны на сайте: https://ntcontest.ru/mentors/education-programs/.

Регистрируясь на платформе НТО, наставники получают доступ к личному кабинету, в котором отображается расписание отборочных соревнований и мероприятий по подготовке, требования к знаниям и компетенциям при решении задач отборочных этапов.

Сообщество наставников НТО существует и развивается. Ежегодно Кружко-

вое движение НТИ проводит Всероссийский конкурс технологических кружков: https://konkurs.kruzhok.org/. Принять участие в конкурсе может каждый наставник.

В 2022 году было выпущено пособие «Технологическая подготовка инженерных команд. Методические рекомендации для наставников». Методические рекомендации предназначены для учителей технологий, а также наставников и педагогов кружков и центров дополнительного образования. Рекомендации направлены на помощь в процессе преподавания технологий в школе или в кружке. Пособие построено на примерах из реального опыта работы со школьниками, состоит из теоретических положений, посвященных популярным взглядам в педагогике на тему подготовки инженерных команд к соревнованиям. Электронное издание доступно по ссылке: https://journal.kruzhok.org/tpost/pggs3bp7y1-tehnologicheskaya-podgotovka-inzhenernih.

В нем рассмотрены особенности подготовки к пяти направлениям:

- Большие данные.
- Машинное обучение.
- Искусственный интеллект.
- Спутниковые системы.
- Летающая робототехника.

Для наставников НТО разработана и постоянно пополняется страница с материалами для профессионального развития: https://nto-forever.notion.site/c9b9cbd21542479b97a3fa562d15e32a.

1.2. Автономные транспортные системы

Организаторы профиля: ООО «Академия Высоких Технологий», Московский политехнический университет и ГК «Геоскан».

Профиль Автономные транспортные системы посвящен применению технологий искусственного интеллекта в задачах беспилотного транспорта. Внедрение беспилотников в информационную инфраструктуру города позволяет тонко управлять ресурсом техники, предсказывать возможные сценарии развития событий и своевременно реагировать на изменения в городской среде.

Финалисты на заключительном этапе решают комплексную инженерную задачу: создать, отладить и внедрить в интеллектуальную городскую инфраструктуру, представленную специальным интерактивным полигоном с моделью города, полностью автономную мультимодальную транспортную систему.

Разрабатываемая система состоит из нескольких программируемых автономных устройств: беспилотный автомобиль и квадрокоптер. Беспилотники собирают информацию, необходимую для доставки грузов, а также полезные данные для города. Например, информацию о состоянии дорожного покрытия, об обрыве линий электропередач, о пожарах и т. д.

Решение задачи требует от участников применения знаний и навыков в области технологий искусственного интеллекта, в частности, компьютерного зрения и нейронных сетей. Алгоритмы компьютерного зрения и искусственного интеллекта, которые разрабатывают участники, можно разделить на две группы:

- алгоритмы локального и глобального позиционирования беспилотников;
- алгоритмы, объединяющие разрозненные устройства в единую слаженно работающую систему.

В рамках первой группы алгоритмов финалисты работают над заданиями, связанными с определением местоположения беспилотных устройств в реальном времени. Это включает в себя использование данных с камер и других сенсоров для создания точной карты окружающей среды, а также применение методов машинного обучения для улучшения точности позиционирования в сложных городских условиях (узкие улицы, плотная застройка и динамическое движение пешеходов и транспортных средств).

Вторая группа алгоритмов обеспечивает интеграцию различных беспилотных устройств в единую транспортную систему. Участники создают программы для обмена данными между устройствами, выстраивают логику распределения задач внутри системы. Это позволяет беспилотникам координировать свои действия, оптимизировать маршруты и реагировать на изменения в окружающей среде.

Сквозная технология профиля — **компьютерное зрение**. В течение всех этапов участники учатся применять ее в различных отраслях:

- локальное и глобальное позиционирование беспилотного автомобиля, детектирование и взаимодействие с объектами городской среды — дорожными знаками, светофорами, пешеходами, дорожной разметкой;
- навигация квадрокоптера по графическим меткам и различным визуальным

маркерам;

- анализ больших данных, собираемых беспилотниками;
- распознавание маркировки для решения задач промышленной робототехники.

На первом отборочном дистанционном этапе участники решают задачи инженерного и предметного туров. Они знакомят школьников с тематикой профиля и вовлекают в изучение образовательной программы профиля.

На предметном туре определяется общий уровень подготовки по школьным предметам: информатика и физика. Задачи по информатике проверяют знания по комбинаторике, теории вероятностей, алгоритмизации, теории игр и программированию. Задания по физике относятся к разделам: механика, оптика и электричество.

В ходе инженерного тура школьники демонстрируют умение применять знания, полученные в первой половине образовательной программы профиля. Здесь задания посвящены:

- представлению изображения в памяти компьютера;
- использованию различных цветовых пространств;
- пороговой и адаптивной бинаризации;
- поиску и анализу контуров;
- детектированию объектов по цветам;
- распознаванию, сравнению с шаблоном;
- обработке видеопотока.

Количество попыток сдачи решения задач не ограничено.

Таким образом, в ходе первого отборочного этапа участники профиля Автономные транспортные системы получают основополагающие знания и умения по технологиям профиля, а также готовятся к решению задач следующего этапа.

Второй отборочный этап посвящен отработке навыков программирования, созданию и обучению нейросетей, работе с алгоритмами компьютерного зрения для детектирования, отслеживания и распознавания объектов. Участники сталкиваются с отдельными элементами комплексной инженерной задачи заключительного этапа и таким образом могут освоить умения, необходимые для ее решения.

Работа на втором этапе помогает лучше понять сильные стороны каждого члена команды и наиболее эффективно распределять обязанности и зоны ответственности.

Организаторы разработали для участников программы подготовки к Национальной технологической олимпиаде по профилю Автономные транспортные системы: https://avt.global/nto_program, выложенную в открытый доступ. Она содержит теоретический материал и практические задания на закрепление полученных знаний, все ее разделы логично выстроены в соответствии с уровнем начальных знаний участников профиля.

Программа включает в себя следующие материалы:

- Курс по основам программирования на Python https://stepik.org/course/67/promo.
- Продвинутый курс по программированию на Python https://stepik.org/course/512/promo.
- Урок НТО «Введение в компьютерное зрение»

https://avt.global/nto-lesson-cv.

- Видеокурс по компьютерному зрению https://avt.global/cv.
- Видеокурс по нейронным сетям https://avt.global/neuralnets.
- Урок по обучению нейросетевого детектора https://colab.research.google.com/drive/1qLyriEpKll-Xa9vEaNXGvvLeopxwn9Pw?usp=sharing.

Организаторы профиля проводят регулярные консультации с участниками в онлайн-формате и отвечают на возникающие вопросы в чате участников в Телеграм и других интернет-сообществах профиля.

Призеры, победители и финалисты профиля поступают в ведущие вузы страны и успешно ведут в них проектную деятельность по инженерным направлениям как в области искусственного интеллекта, так и в смежных направлениях.

Компетенции, полученные за время участия в профиле, востребованы в дальнейшей проектной работе в вузах, а также применяются в отрасли. Так, команда молодых ученых разработала беспилотного робота-курьера (http://vestnik-glonass.ru/news/intro/komanda-molodykh-uchenykh-razrabotala-bespilotnogo-robotakurera/). Примером применения полученных знаний и умений в области, не связанной с беспилотным транспортом, является COVID Challenge (https://avt.global/covid). В рамках мероприятия участники обучили нейронную сеть, определяющую наличие инфекции COVID-19 у сотен пациентов города Москвы по трехмерным снимкам компьютерной томографии легких.

2. Первый отборочный этап

2.1. Работа наставника НТО на этапе

Педагог-наставник играет важную роль в подготовке участника к первому отборочному этапу Национальной технологической олимпиады. На этом этапе школьникам предстоит справиться как с предметными задачами, соответствующими профилю, так и с заданиями инженерного тура, погружающими в выбранную технологическую область.

Наставник может организовать подготовку участника, используя разнообразные форматы и ресурсы:

- Разбор заданий прошлых лет. Совместный анализ задач отборочного этапа предыдущих лет позволяет понять структуру, уровень сложности и типичные подходы к решению. Это формирует у школьника устойчивые стратегии работы с олимпиадными заданиями.
- Мини-соревнования. Проведение тренировочных турниров с заданиями предметных олимпиад муниципального уровня помогает развить соревновательный навык, тренирует скорость и уверенность при решении задач в ограниченное время.
- Углубленные занятия. Наставник может выстроить образовательную траекторию, опираясь на рекомендации разработчиков профиля, и провести занятия по ключевым темам. Это особенно важно для системного понимания предметной области.
- Использование онлайн-курсов. Для самостоятельной подготовки и проверки знаний участник может использовать предметные курсы НТО, размещенные на платформах Степик и Яндекс Контест. Наставник может также организовать занятия с использованием этих материалов в рамках групповой или индивидуальной подготовки.
- Привлечение внешних экспертов. Если у наставника нет достаточной экспертизы в какой-либо предметной области, он может пригласить других педагогов или специалистов для проведения тематических занятий.
- Поддержка в инженерном туре. Инженерный тур включает теоретические материалы и задания, помогающие глубже погрузиться в тематику профиля. Наставник может сопровождать изучение курса, помогать в разборе теоретических вопросов и тренировать участника на практических задачах.

Таким образом, наставник не только помогает систематизировать подготовку, но и мотивирует участника, создавая для него комфортную и продуктивную образовательную среду.

2.2. Предметный тур. Информатика

2.2.1. Первая волна. Задачи 8–11 класса

Задачи первой волны предметного тура по информатике открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63452/enter/.

Задача 2.2.1.1. Ускорение ускорения (10 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Рассмотрим модель движения тела. Будем фиксировать такие параметры, как координата, скорость, ускорение и ускорение ускорения (рывок). Если некоторый параметр равен a и имеет скорость изменения v, то в следующий момент времени этот параметр будет равен $a + v$.

Например, если тело имело координату, равную 10, скорость, равную 20, ускорение, равное 30 и ускорение ускорения, равное 40, то в следующий момент оно будет иметь координату 30, скорость 50 и ускорение 70. Ускорение ускорения будем считать в этой задаче постоянной величиной.

Задача довольно проста: тело в начальный момент времени 0 находится в точке с координатой 0, скоростью 0 и ускорением 0. На это тело действует постоянное ускорение ускорения, равное 6. Требуется определить, в точке с какой координатой окажется это тело в момент времени t.

Формат входных данных

В единственной строке находится одно число t, где $0 \leqslant t \leqslant 10^6$.

Формат выходных данных

Вывести одно число — координату, в которой окажется тело в момент времени t.

Примеры

Пример №1

Стандартный ввод
6
Стандартный вывод
120

Пример №2

Стандартный ввод
2
Стандартный вывод
0

Пример №3

Стандартный ввод
1000000
Стандартный вывод
999997000002000000

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int t;
    cin >> t;
    cout << ((t * (t - 1)) * (t - 2)) << endl;
}
```

Задача 2.2.1.2. Двойное остекление (15 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

У деда Василия есть два прямоугольных куска стекла. Один из них имеет размеры $a \times b$, другой — $c \times d$. Дед собирается из этих кусков сделать окно с двойным остеклением. Он хочет, чтобы окно было обязательно квадратным и как можно большим по размеру. Дед должен вырезать из имеющихся у него прямоугольников два одинаковых квадрата максимально возможного размера. Нужно написать программу, которая по заданным a, b, c, d найдет максимальные размеры квадратного окна. Имейте ввиду, что оба квадрата могут быть вырезаны и из одного прямоугольного куска стекла.

Формат входных данных

На вход подаются две строки. В первой строке находятся размеры первого прямоугольника a, b через пробел, во второй — размеры второго прямоугольника c, d через пробел, где $1 \leqslant a, b, c, d \leqslant 10^9$.

Формат выходных данных

Вывести одно число — максимальную сторону квадратного двойного окна, которое можно вырезать из заданных на входе прямоугольных кусков стекла. Ответ может быть нецелым, требуется вывести его с точностью 1 знак после десятичной точки.

Примеры

Пример №1

Стандартный ввод
5 10 9 6
Стандартный вывод
5

Пример №2

Стандартный ввод
4 10 9 6
Стандартный вывод
4.5

Комментарий

Второй пример показывает, что иногда лучше вырезать оба квадрата из одного и того же куска стекла.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    double a, b, c, d;
    cin >> a >> b >> c >> d;
    double a0 = min({a, b, c, d});
    double a1 = min(max(a, b) / 2.0, min(a, b));
    double a2 = min(max(c, d) / 2.0, min(c, d));
    double ans =  max({a0, a1, a2});
    if( (int)ans == ans ){
        int ians = ans;
        cout << ians << endl;
        return 0;
    }
    cout.precision(1);
    cout << fixed<< ans << endl;
}
```

Задача 2.2.1.3. О золотой рыбке и... досках (20 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

После событий известной сказки А. С. Пушкина старик решил принципиально не пользоваться услугами золотой рыбки. Поэтому для того чтобы изготовить новое корыто, он честно заготовил n одинаковых досок.

Но гостивший в это время у старика со старухой внук решил, что ему нужно научиться пилить. И, не сказав ничего своему деду, внук быстро распилил каждую из досок на две части. В итоге у старика оказались $2n$ кусков досок. Самое интересное, что все эти куски оказались разными по длине, но имели целочисленные размеры. К сожалению, старик забыл, какова была исходная длина целых досок.

Формат входных данных

В первой строке задается целое число n — исходное количество целых досок, где $1 \leqslant n \leqslant 10^5$.

Во второй строке заданы $2n$ целых чисел d_i — длины всех кусков, которые получились после «тренировки» внука, где $1 \leqslant d_i \leqslant 10^9$. Гарантируется, что эти числа попарно различны, и их можно разбить на пары одинаковых по сумме чисел.

Все эти части досок пронумерованы от 1 до $2n$ в том порядке, в котором они заданы на входе.

Формат выходных данных

В первую строку вывести одно число — исходную длину целых досок.

В следующих n строках вывести пары номеров кусков досок, которые составляют по длине целые доски. Номера выводить через один пробел, внутри пары сначала должен идти меньший номер, затем больший. Пары должны быть выведены в порядке возрастания первых номеров в парах.

Примеры

Пример №1

Стандартный ввод
3 4 8 2 3 6 7

Стандартный вывод
10 1 5 2 3 4 6

Комментарий

Отсортируем куски и далее будем брать один из начала и второй к нему из конца.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int n;
    cin >> n;
    vector<pair<int, int> > v(2 * n);
     for(int i = 0; i < 2 * n; i++){
        int d;
        cin >> d;
        v[i] = {d, i + 1};
    }
    sort(v.begin(), v.end());
    vector<pair<int, int> > ans(n);
    for(int i = 0; i < n; i++){
```

```
16          ans[i] = {v[i].second, v[2 * n - i - 1].second};
17          if(ans[i].first > ans[i].second){
18              swap(ans[i].first, ans[i].second);
19          }
20      }
21      sort(ans.begin(), ans.end());
22      cout << v[0].first + v.back().first<< endl;
23      for(int i = 0; i < n; i++){
24          cout << ans[i].first<<' '<< ans[i].second<< endl;
25      }
26  }
```

Задача 2.2.1.4. Бонусы и экономия (25 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Технология производства некоторой металлической детали предполагает вытачивание ее из металлической заготовки. При этом образуются стружки, которые не стоит выкидывать. Ведь из a комплектов стружек (оставшихся после обработки a заготовок) можно бесплатно выплавить еще одну заготовку, которую снова можно использовать для выточки детали и создания еще одного комплекта стружек.

Заготовки можно купить на оптовом складе, при этом в целях привлечения клиентов, проводится акция «купи b заготовок, тогда еще одну получишь бесплатно».

Требуется изготовить c деталей. Нужно определить минимальное число заготовок, которые нужно купить за деньги, чтобы с учетом бонусных заготовок и экономии на стружках можно было изготовить требуемое число деталей.

Формат входных данных

В одной строке через пробел заданы три целых числа a, b, и c такие, что $2 \leqslant$ $\leqslant a \leqslant 10^{18}$, $1 \leqslant b$, $c \leqslant 10^{18}$.

Формат выходных данных

Вывести одно целое число — минимальное количество заготовок, которые нужно купить, чтобы с учетом всех бонусов и экономии выточить c конечных деталей.

Примеры

Пример №1

Стандартный ввод
4 5 41
Стандартный вывод
26

Примечания

В примере из условия нужно закупить 26 заготовок. Тогда за каждые пять купленных заготовок будет предоставлена одна бесплатная, итого по акции добавится еще пять заготовок, то есть получится 31 заготовка. Далее из 31 заготовки выточится 31 деталь, останется 31 комплект стружек. Из каждых четырех комплектов выплавится дополнительная заготовка, получится семь заготовок и три комплекта стружек. Из семи заготовок выточится семь деталей и останется семь комплектов стружек, три комплекта стружек осталось с первого шага, итого 10 комплектов стружек. Из них выплавится еще две заготовки, дающие две детали и два комплекта стружек. Собрав эти два комплекта с двумя, оставшимися от 10, получим еще одну заготовку, из которой выточится еще одна деталь. Останется один комплект стружек, который уже никак не получится использовать. Итого будет произведена $31 + 7 + 2 + 1 = 41$ деталь.

Комментарий

Методом бинарного поиска можно подобрать минимальное необходимое количество исходных заготовок.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
int f1(int M, int a){
    int res = 0, z = 0;
    while(1){
        if(M == 0 && z < a){
            return res;
        }
        res += M;
        M = M + z;
        z = M % a;
        M = M / a;
    }
}
```

```
int f2(int M, int b){
    return M + M / b;
}
signed main(){
    int a, b, c;
    cin >> a >> b >> c;
    int L = 0, R = 1;
    while(f1(R, a) <= c){
        R *= 2;
    }
    while(R - L > 1){
        int M = (R + L) / 2;
        if(f1(M, a) < c){
            L = M;
        }
        else{
            R = M;
        }
    }
    int z = R;
    L = 0, R = 1;
    while(f2(R, b) <= z){
        R *= 2;
    }
    while(R - L > 1){
        int M = (R + L) / 2;
        if(f2(M, b) < z){
            L = M;
        }
        else{
            R = M;
        }
    }
    cout << R << endl;
}
```

Задача 2.2.1.5. Сон таксиста (30 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Одному таксисту приснился красочный сон. Во сне он живет и работает в некотором городе, где абсолютно все улицы с односторонним движением. Эти улицы устроены так, что невозможно проехать с какого-либо перекрестка так, чтобы вернуться обратно на этот же перекресток, то есть в дорожной сети города нет циклов.

Таким образом, если с перекрестка A можно попасть по направлению движения улиц на перекресток B, то люди вызывают такси, иначе их везет специальный муниципальный подземный транспорт бесплатно.

В связи с такими странными правилами, таксистам в этом городе разрешено законом везти пассажира по любому маршруту, не нарушающему направления движения. Все в этом городе привыкли к такой ситуации и абсолютно спокойно относятся к тому, что таксисты везут их самым длинным путем. Разумеется, заработок таксиста за одну поездку прямо пропорционален ее длине. Для упрощения будем считать, что стоимость 1 км поездки составляет ровно 1 руб.

Схема дорог города задана. Перекрестки города пронумерованы числами от 1 до n. Таксист в своем сне находится на перекрестке номер S. Напишите программу, которая подскажет ему, сколько он максимально сможет заработать, когда ему придет заказ от клиента. Так как он не знает, куда попросит его везти клиент, нужно для каждого перекрестка от 1 до n указать максимальную стоимость поездки до этого перекрестка из пункта S на такси. Если по правилам на такси добраться из пункта S до какого-то перекрестка нельзя, вывести -1.

Формат входных данных

Дорожная сеть задана следующим образом: в первой строке находятся два числа через пробел n и m — число перекрестков и число улиц в городе, где $2 \leqslant n, m \leqslant 2 \cdot \cdot 10^5$.

В следующих m строках задана очередная односторонняя улица в виде трех чисел A, B, d через пробел, где A — начало улицы, B — конец улицы и d — ее длина. $1 \leqslant A, B \leqslant n$, $1 \leqslant d \leqslant 10^9$. Гарантируется, что в этой дорожной сети нет циклов. Некоторые пары перекрестков могут быть соединены двумя и более односторонними улицами. Дорожная сеть может быть неплоской за счет мостов и тоннелей.

В последней строке ввода содержится номер стартового перекрестка S, $1 \leqslant S \leqslant \leqslant n$.

Формат выходных данных

Вывести n чисел в одну строку через пробел. i-е число обозначает длину самого длинного пути с перекрестка номер S до перекрестка номер i. Если до перекрестка номер i от S нельзя доехать, не нарушая правила движения, вывести -1.

Примеры

Пример №1

Стандартный ввод
10 20 9 10 15 9 8 3 8 10 7 7 8 4 7 10 10 5 8 2 5 9 10

Стандартный ввод

```
5 6 5
7 6 5
4 6 8
3 6 4
3 4 6
5 3 2
2 5 2
2 3 3
3 1 5
1 4 2
2 1 7
4 7 4
6 8 1
5
```

Стандартный вывод

```
7 -1 2 9 0 18 13 19 10 26
```

Комментарий

Задача решается методом динамического программирования на ориентированном ациклическом графе.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
int n, m;
vector<vector<pair<int, int> > > G;
vector<int> order, used;
void dfs(int a){
    used[a] = 1;
    for(auto to : G[a]){
        if(!used[to.first]){
            dfs(to.first);
        }
    }
    order.push_back(a);
}
signed main(){
    cin >> n >> m;
    G.resize(n + 1);
    used.resize(n + 1, 0);
    for(int i = 0; i < m; i++){
        int a, b, d;
        cin >> a >> b >> d;
        G[a].push_back({b, d});
    }
```

```
    int s;
    cin >> s;
    dfs(s);
    reverse(order.begin(), order.end());
    vector<int> dp(n + 1, -1);
    dp[s] = 0;
    for(auto el : order){
        for(auto to : G[el]){
            dp[to.first] = max(dp[to.first], dp[el] + to.second);
        }
    }
    for(int i = 1; i <= n; i++){
        cout << dp[i] <<' ';
    }
}
```

2.2.2. Вторая волна. Задачи 8–11 класса

Задачи второй волны предметного тура по информатике открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63454/enter/.

Задача 2.2.2.1. Игра на планшете (10 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Маленький Андрей изучает геометрические фигуры при помощи игры на планшете. У него есть прямоугольные треугольники четырех цветов и ориентаций: желтые, зеленые, красные и синие. Для каждой разновидности треугольников есть заданное количество экземпляров этих треугольников. Более точно: у Андрея есть a желтых, b зеленых, c красных и d синих треугольников. Помимо этого у него есть прямоугольная таблица $n \times m$.

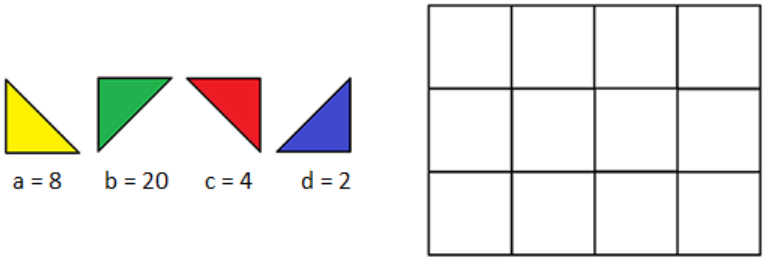

Рис. 2.2.1

Треугольники одного цвета имеют одну и ту же ориентацию, которую нельзя поменять. Андрей может только взять очередной треугольник и переместить его параллельным сдвигом в одну из ячеек этой прямоугольной таблицы. При этом в одну ячейку можно поместить либо вместе желтый и красный треугольники, либо вместе зеленый и синий, либо один любой треугольник из имеющихся.

Андрей хочет расположить в ячейках таблицы как можно больше треугольников из тех, что у него имеются. Нужно подсказать ему максимальное количество треугольников, которые получится разместить в таблице.

Формат входных данных

В первой строке содержатся четыре целых числа a, b, c и d через пробел — количество желтых, зеленых, красных и синих треугольников соответственно.

Во второй строке содержатся два целых числа n и m через пробел — размеры прямоугольной таблицы.

Все числа в пределах от 1 до 10^9.

Формат выходных данных

Вывести одно число — максимальное количество треугольников, которые можно при заданных условиях разместить в таблице.

Примеры

Пример №1

Стандартный ввод
`8 20 4 2` `3 4`

Стандартный вывод
`18`

Примечания

На рис. 2.2.2 представлен один из примеров размещения 18 треугольников из 34 заданных на входе.

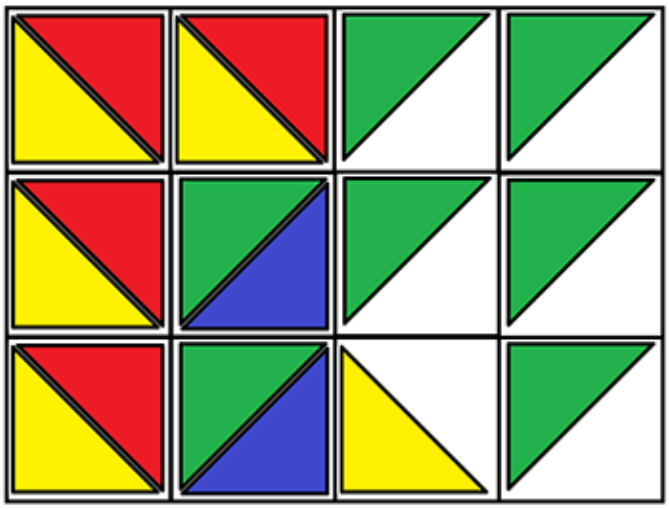

Рис. 2.2.2

Решение

Ниже представлено решение на языке C++.

C++

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int a, b, c, d, n, m;
    cin >> a >> b >> c >> d >> n >> m;
    if(a > c){
        swap(a, c);
    }
    if(b > d){
        swap(b, d);
    }
    int f = a + b;
    int k = n * m;
    if(k <= f){
        cout << k * 2;
        return 0;
    }
    k -= f;
    c -= a;
    d -= b;
    cout << f * 2 + min(k, c + d) << endl;
}
```

Задача 2.2.2.2. Старая задача на новый лад (15 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Одна старая задача имеет следующий вид:

«Разбить число 45 на сумму четырех слагаемых так, что если к первому прибавить 2, из второго вычесть 2, третье умножить на 2, а четвертое разделить на 2, то получится одно и то же число».

Ответ к этой задаче — четыре числа 8, 12, 5 и 20. Можно убедиться, что в сумме они дают число 45, а если с каждым из них проделать соответствующую арифметическую операцию, то получится одно и то же число 10.

Необходимо решить чуть более общую задачу: даны числа n и k. Нужно представить число n в виде суммы четырех целых неотрицательных слагаемых $a + b + c + d$ таких, что $a + k = b - k = c \cdot k = d/k$. Гарантируется, что для заданных n и k такое разбиение существует.

Формат входных данных

В одной строке через пробел два числа n и k, где $1 \leqslant n \cdot k \leqslant 10^{18}$.

Формат выходных данных

Вывести через пробел в одну строку четыре целых неотрицательных числа a, b, c, d таких, что $a + b + c + d = n$ и $a + k = b - k = c \cdot k = d \,/\, k$.

Примеры

Пример №1

Стандартный ввод
`45 2`

Стандартный вывод
`8 12 5 20`

Пример №2

Стандартный ввод
`128 7`

Стандартный вывод
`7 21 2 98`

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int n, k;
    cin >> n >> k;
    int x = (k * n) / (k * k + 2 * k + 1);
    cout << x - k <<' '<< x + k <<' '<< x / k <<' '<< x * k << endl;
}
```

Задача 2.2.2.3. Ладья и обязательная клетка (20 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Шахматная ладья находится в левом верхнем углу прямоугольного поля, разбитого на клетки размером $n \times m$. n обозначает число строк, m — число столбцов. Она хочет попасть в правую нижнюю клетку этого поля кратчайшим путем. Ладья может передвигаться либо вправо, либо вниз на любое количество клеток. Ладья обязана посетить заданную клетку с координатами (x, y), где x — номер строки этой клетки, а y — номер ее столбца.

Требуется найти количество способов построить путь ладьи из левого верхнего угла в правый нижний, которые проходят через обязательную клетку с заданными координатами.

Формат входных данных

В первой строке находятся два числа через пробел: n — число строк и m — число столбцов прямоугольного поля, $2 \leqslant n, m \leqslant 25$. Во второй строке через пробел находятся координаты (x, y) обязательной для посещения клетки, где $1 \leqslant x \leqslant n$, $1 \leqslant y \leqslant m$. Координаты x и y не совпадают с координатами левой верхней и правой нижней клеток.

Формат выходных данных

Вывести одно число — количество кратчайших путей ладьи из верхней левой в правую нижнюю клетку, проходящих через заданную клетку.

Примеры

Стандартный ввод
3 4 2 3

Стандартный вывод
6

Примечания

На рис. 2.2.3 представлены шесть путей, которыми ладья может пройти по полю размером 3×4, обязательно посещая по пути клетку $(2, 3)$.

Комментарий

Задачу можно решить как комбинаторными методами (произведение биномиальных коэффициентов), так и динамическим программированием.

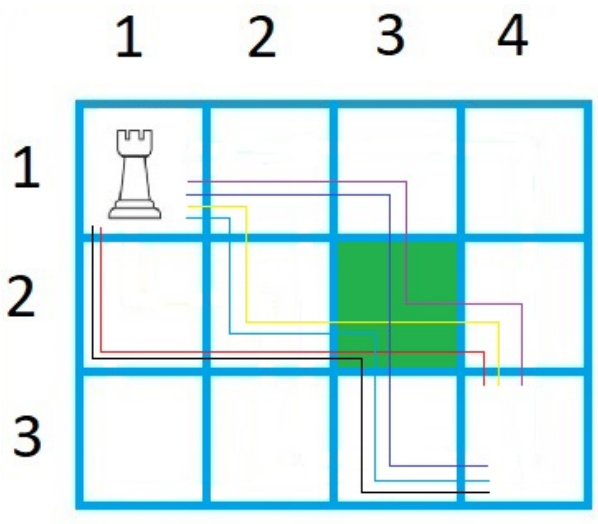

Рис. 2.2.3

Решение

Ниже представлено решение на языке C++.

C++

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    vector<vector<int> > bc(51, vector<int>(51, 0));
    bc[0][0] = 1;
    for(int i = 1; i <= 50; i++){
        for(int j = 0; j < 51; j++){
```

```
            bc[i][j] += bc[i - 1][j];
            if(j - 1 >= 0){
                bc[i][j] += bc[i - 1][j - 1];
            }
        }
    }
    int n, m, x, y;
    cin >> n >> m >> x >> y;
    int d1 = bc[x - 1 + y - 1][x - 1];
    int d2 = bc[n - x + m - y][n - x];
    int ans = d1 * d2;
    cout << ans << endl;
}
```

Задача 2.2.2.4. Танец с цифрами (25 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Десять танцоров репетируют на сцене новый танец. Каждый танцор одет в футболку, на которой написана одна из цифр от 1 до 9, цифры могут повторяться. Изначально они стоят в некотором порядке слева направо, и их цифры образуют некоторое десятизначное число A. Далее во время всего танца участники либо разбиваются на пять пар рядом стоящих танцоров и одновременно меняются местами внутри своих пар, либо самый левый танцор перемещается на самую правую позицию и становится самым правым танцором.

Сын постановщика танца от скуки на бумаге выписывает все получающиеся при каждом перемещении десятизначные числа. Так как танец длинный, то в итоге на бумаге окажутся все возможные числа, которые в принципе могут появится при этих условиях. Нужно найти разницу между самым большим и самым маленьким из этих чисел.

Формат входных данных

На вход подается одно десятизначное число A, обозначающее начальное расположение танцоров. В числе могут встречаться цифры от 1 до 9, некоторые из них могут повторяться.

Формат выходных данных

Вывести одно число, равное разности самого большого и самого маленького из чисел, которые могут быть получены во время танца.

Примеры

Пример №1

Стандартный ввод
1456531355
Стандартный вывод
5182160085

Примечания

Самое маленькое число, которое можно получить в примере, равно 1353155456, самое большое равно 6535315541.

Покажем, как получить эти числа из исходного числа 1456531355. Сначала получим самое большое следующим образом: две левых цифры, 1 и 4, переместим вправо, получим 5653135514, потом поменяем в парах цифры местами и получим самое большое — 6535315541. Далее опять поменяем порядок в парах и в числе 5653135514 переместим три левых цифры 5, 6 и 5 вправо, получим 3135514565 и здесь снова поменяем порядок в парах, получим самое маленькое — 1353155456. Таким образом, искомая разница равна 5182160085.

Решение

Ниже представлено решение на языке C++.

C++

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    string s;
    cin >> s;
    string mx = s, mn = s;

    for(int i = 0; i < 5; i++){
        for(int j = 0; j < 10; j++){
            mx = max(mx, s);
            mn = min(s, mn);
            if(j < 9){
                s = s.substr(1) + s[0];
            }
        }
        for(int j = 0; j < 5; j++){
            swap(s[2 * j], s[2 * j + 1]);
        }
    }
    stringstream ssmn;
    ssmn << mn;
    int imn;
    ssmn >> imn;
    stringstream ssmx;
```

```
26      ssmx << mx;
27      int imx;
28      ssmx >> imx;
29      cout << imx - imn << endl;
30  }
```

Задача 2.2.2.5. Трудная сортировка (30 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 3 с.

Ограничение по памяти: 64 Мбайт.

Условие

Иннокентий работает в отделе сортировки перестановок, подотделе сортировки вставками. Его задача заключается в сортировке перестановок, предоставленных заказчиками. Перестановкой длины n называется такая последовательность чисел, в которой встречаются все числа от 1 до n без повторений в некотором порядке.

Перестановка считается отсортированной, если в ней все числа расположены по возрастанию, то есть она имеет вид $1, \ldots, n$.

Иннокентий начинает рабочий день с пустой последовательности чисел. За день он сортирует вставками перестановку длины n. В начале каждой операции вставки он получает очередное число a_i из перестановки заказчика, после чего обрабатывает его, вставляя в отсортированную последовательность из ранее полученных чисел. После каждого такого добавления последовательность уже обработанных чисел должна быть отсортирована по возрастанию.

Перед тем как вставить число a_i в последовательность, он может выбрать, с какого края последовательности начать вставку. Далее он устанавливает число a_i с этого края и последовательно меняет вставляемое число с рядом стоящим числом b_j до тех пор, пока число a_i не встанет на свое место. На каждую перестановку вставляемого числа a_i с числом b_j Иннокентий тратит b_j единиц энергии.

Дана перестановка длины n из чисел a_i в том порядке, в котором Иннокентий их будет обрабатывать. Подскажите ему, какое минимальное количество энергии ему потребуется потратить, чтобы отсортировать всю перестановку.

Формат входных данных

В первой строке находится одно целое число n — длина перестановки, где $1 \leqslant \leqslant n \leqslant 2 \cdot 10^5$.

Во второй строке содержится n целых чисел a_i через пробел в том порядке, в котором они поступают на обработку Иннокентию. Гарантируется, что эти числа образуют перестановку длины n, то есть каждое число от 1 до n содержится в заданном наборе ровно один раз.

Формат выходных данных

Вывести одно число — минимальные суммарные энергозатраты Иннокентия для сортировки вставками заданной на входе перестановки.

Примеры

Пример №1

Стандартный ввод
9 2 9 1 5 6 4 3 8 7
Стандартный вывод
43

Примечания

Первым устанавливается число 2. Оно ни с чем не меняется местами, поэтому затрат нет.

Далее устанавливается число 9. Выбираем правый край и ставим его туда без потерь энергии.

Затем устанавливаем число 1. Выбираем левый край, ставим его туда и снова потерь нет.

Теперь нужно вставить число 5. Если его вставлять с правого края, придется менять местами с 9, а если с левого, то с 1 и 2, что суммарно явно лучше. Итого затраты на вставку 5 равны 3.

Число 6 снова лучше вставить слева, затраты на его вставку равны 8.

Число 4 вставим слева за 3.

Число 3 так же слева за 3.

А вот число 8 лучше вставить справа за 9.

И осталось число 7. Если вставлять слева, то затратим 21, а если справа, то всего 17.

Итого на сортировку заданной перестановки потратили: $0 + 0 + 0 + 3 + 8 + 3 + 3 + 9 + 17 = 43$.

Комментарий

Построим дерево отрезков на сумму, при обработке числа a будем находить, какая сумма на данный момент меньше: от 1 до $a - 1$ или от $a + 1$ до n. Прибавим ее к ответу и поместим в позицию a это число a.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int LG = 19;
int N = (1 << LG);
vector<int> tr(2 * N, 0);
void upd(int pos, int x){
    pos += N;
    tr[pos] = x;
    pos /= 2;
    while(pos){
        tr[pos] = {tr[2 * pos]+ tr[2 * pos + 1]};
        pos /= 2;
    }
}
int get(int l, int r){
    l += N;
    r += N;
    int res = 0;
    while(l <= r){
        if(l % 2 == 1){
            res += tr[l];
        }
        if(r % 2 == 0){
            res += tr[r];
        }
        l = (l + 1) / 2;
        r = (r - 1) / 2;
    }
    return res;
}
signed main(){
    int n, a;
    cin >> n;
    int ans = 0;
    for(int i = 0; i < n; i++){
        cin >> a;
        int sl = get(0, a - 1);
        int sr = get(a + 1, N - 1);
        ans += min(sl, sr);
        upd(a, a);
    }
    cout << ans << endl;
}
```

2.2.3. Третья волна. Задачи 8–11 класса

Задачи третьей волны предметного тура по информатике открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63456/enter/.

Задача 2.2.3.1. Туннель (10 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Рассмотрим классическую задачу прохождения группы с одним фонариком по туннелю. Есть четыре человека, и у них есть один фонарик. Нужно перевести всю группу на другой конец туннеля. По туннелю можно проходить только с фонариком и только либо вдвоем, либо в одиночку. По этой причине придется сделать пять рейсов по туннелю: три рейса туда и два рейса обратно. Туда идут двое, обратно — один, возвращая фонарик еще не прошедшей части группы. У каждого из четырех человек своя скорость передвижения по туннелю, но некоторые скорости могут совпадать. Двое идут со скоростью самого медленного в этой паре. Нужно найти минимальное время, за которое можно перевести группу по туннелю.

Здесь, в зависимости от скоростей персонажей, есть две стратегии. Проиллюстрируем их на примерах.

Пусть есть люди A, B, C, D. У A — время прохождения туннеля 1 мин, у B — 4 мин, у C — 5 мин, у D — 10 мин. Здесь работает наиболее очевидная стратегия: самый быстрый переводит текущего и возвращается с фонариком обратно за следующим. При этой стратегии нужно проходить так:

- A, B туда, затрачено 4 мин;
- A обратно, затрачена 1 мин;
- A, C туда, затрачено 5 мин;
- A обратно, затрачена 1 мин;
- A, D туда, затрачено 10 мин.

Общее время $4 + 1 + 5 + 1 + 10 = 21$ мин.

Но не всегда эта стратегия оптимальна. Уменьшим время прохождения туннеля персонажем B до 2 мин. По вышеопределенной стратегии будет 19 мин ($2 + 1 + 5 + + 1 + 10 = 19$), но имеется более быстрое решение:

- A, B туда, затрачено 2 мин;
- A обратно, затрачена 1 мин;
- C, D туда, затрачено 10 мин;
- B обратно, затрачено 2 мин;
- A, B туда, затрачено 2 мин.

Общее время $2 + 1 + 10 + 2 + 2 = 17$ мин.

Заметим, что для предыдущего примера такая стратегия не работает: $4+1+10+ + 4 + 4 = 23$ мин.

Если же персонаж B проходит туннель за 3 мин (а все остальные так же, как и в примерах), то независимо от стратегии будет затрачено 20 мин. В этом случае

считаем, что работает первая стратегия.

Поразмыслив, станет понятно, от какого условия зависит выбор стратегии. Далее будем всегда считать, что A движется не медленнее B, B движется не медленнее C, C движется не медленнее D.

Дано время прохождения туннеля персонажами A, C, D. Нужно найти границу `border` для B такую, что если определить для B время прохождения строго меньшее, чем `border`, то выгодна вторая стратегия, иначе — первая.

Формат входных данных

В одной строке задано три целых чисел через пробел — время прохождения туннеля персонажами A, C, D. Времена даны по неубыванию. Все числа на входе в пределах от 1 до 100.

Формат выходных данных

Вывести одно число — границу `border` для B такую, что если определить время прохождения им туннеля строго меньше, чем `border`, нужно использовать вторую стратегию, иначе — первую. Ответ может быть нецелым, поэтому вывести его нужно с одним знаком после десятичной точки.

Примеры

Пример №1

Стандартный ввод
`1 5 10`
Стандартный вывод
`3`

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int A, C, D;
    cin >> A >> C >> D;
    cout.precision(1);
    cout << fixed << (A + C) / 2.0 << endl;
}
```

Задача 2.2.3.2. Математический пазл (15 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

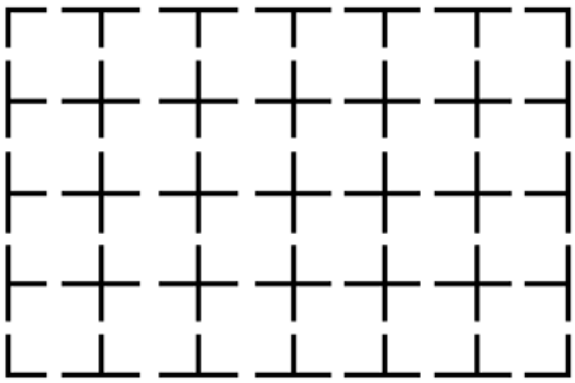

Рис. 2.2.4

Компания по производству пазлов решила освоить принципиально новый тип головоломок. Для этого берется прямоугольная решетка размера $n \times m$, каждый ее столбец и строка разрезаются посередине пополам. После этого образуются фигуры трех типов: четыре уголка, $2 \cdot (n+m-2)$ т-образных фигур и $(n-1) \cdot (m-1)$ крестиков.

Тому, кто решает головоломку, требуется сложить из этих фигур исходную прямоугольную решетку. При этом необходимо использовать абсолютно все имеющиеся в наличии фигуры.

Формат входных данных

В первой строке заданы через пробел два числа a — количество т-образных фигур и b — количество крестиков, которые находятся в одном из пазлов. При этом в наборе всегда есть еще четыре уголка. Известно, что этот комплект позволяет собрать прямоугольную решетку размера $n \times m$, где $1 \leqslant n,\ m \leqslant 10^9$.

Формат выходных данных

Требуется по числам a и b найти размеры исходной решетки n и m. Будем всегда считать, что $n \leqslant m$, то есть нужно вывести в одну строку через пробел два числа, первое из которых не превосходит второго, и вместе они задают размеры загаданной решетки.

Примеры

Пример №1

Стандартный ввод
16 15
Стандартный вывод
4 6

Пример №2

Стандартный ввод
0 0
Стандартный вывод
1 1

Комментарий

Задачу можно решить либо бинарным поиском, либо при помощи квадратного уравнения.

Решение

Ниже представлено решение на языке C++ при помощи бинпоиска.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int a, b;
    cin >> a >> b;
    int L = 0, R = a / 4 + 1;
    while(R - L > 1){
        int M = (R + L) / 2;
        int D = a / 2 - M;
        if(M * D <= b){
            L = M;
        }
        else{
            R = M;
        }
    }
    cout << L + 1 << ' ' << a / 2 - L + 1 << endl;
}
```

Задача 2.2.3.3. Восемь пирогов и одна свечка (20 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Мечта Карлсона наконец-то сбылась! Мама Малыша испекла восемь пирогов прямоугольной формы и в один из них воткнула свечку. После того как Карлсон съел семь пирогов, он решил-таки поделиться кусочком оставшегося восьмого пирога с Малышом. Но, будучи в хорошем настроении, он вынул из пирога свечу и предложил ему решить задачку.

«Так как я самый щедрый Карлсон в мире, то делить оставшийся пирог будешь ты. Но учти, ты должен разрезать пирог одним прямым разрезом так, чтобы линия прошла через один из углов и точку, где стояла свечка. После этого я выберу себе один из двух кусочков, а оставшийся, так и быть, достанется тебе».

Малыш не против этого замысла, однако считает, что разрезать пирог нужно как можно более справедливо, то есть так, чтобы разница между меньшим и большим кусками была как можно меньше. Подскажите Малышу, какой минимальной разницы между площадями кусков он сможет добиться.

Формат входных данных

В первой строке находятся два числа n и m через пробел — размеры прямоугольного пирога. Пирог размещен на координатной плоскости так, что его левый нижний угол находится в точке $(0, 0)$, а правый верхний — в точке (n, m), где $2 \leqslant n,\ m \leqslant 1\,000$.

Во второй строке находятся два числа x и y через пробел — координаты свечки, где $1 \leqslant x \leqslant n - 1$, $1 \leqslant y \leqslant m - 1$, то есть свечка находится строго внутри пирога.

Формат выходных данных

Вывести одно вещественное число с точностью не менее трех знаков после десятичной точки — минимальную разницу между площадями двух получающихся после разрезания кусков, которую сможет получить Малыш.

Примеры

Пример №1

Стандартный ввод
8 5 7 2
Стандартный вывод
12.571

Пример №2

Стандартный ввод
2 2 1 1
Стандартный вывод
0.000

Примечания

На рис. 2.2.5 представлены четыре варианта разделения пирога для первого примера из условия. Можно видеть, что самый близкий к справедливому способ разделения связан с разрезом из левого верхнего угла. Площадь треугольника в этом случае будет равна $96 / 7$, площадь четырехугольника равна $184 / 7$, и разница равна $88 / 7$, что при округлении до трех знаков равно 12,571.

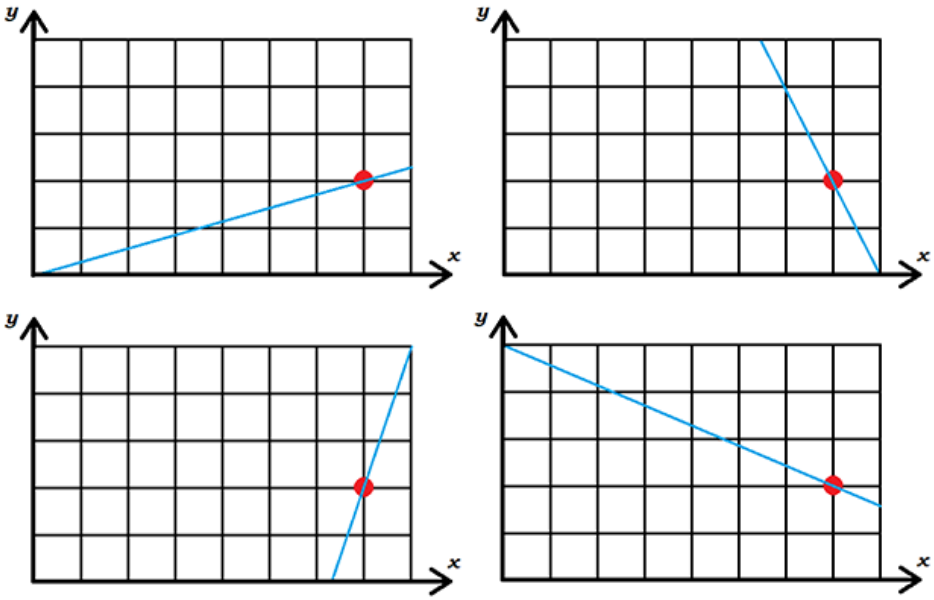

Рис. 2.2.5

Комментарий

Геометрия: для каждого из четырех случаев аккуратно находим катеты прямоугольного треугольника при помощи пропорции, затем находим площадь этого треугольника и, вычитая из всего прямоугольника эту площадь, находим площадь второго куска. Далее выбираем наиболее оптимальное отношение площадей.

Решение

Ниже представлено решение на языке C++.

C++

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int INF = 1e18;
double katy(double x, double y, double n){
    return n * y / x;
}
double n, m, x, y;
double ans = INF;
double k1, k2;
void upd(){
    if(k1 < m){
        double st =k1 * n / 2;
        ans = min(ans, n * m - 2 * st);
    }
    else{
        double st =k2 * m / 2;
        ans = min(ans, n * m - 2 * st);
    }}
signed main(){
    cin >> n >> m >> x >> y;
    k1 = katy(x, y, n);
    k2 = katy(y, x, m);
    upd();
    k1 = katy(n - x, y, n);
    k2 = katy(y, n - x, m);
    upd();
    k1 = katy(x, m - y, n);
    k2 = katy(m - y, x, m);
    upd();
    k1 = katy(n - x, m - y, n);
    k2 = katy(m - y, n - x, m);
    upd();
    cout.precision(3);
    cout << fixed << ans<< endl;
}
```

Задача 2.2.3.4. Плетенка (25 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

У Маши есть n полосок бумаги. i-я полоска имеет ширину 1 и длину a_i. Маша разделит эти полоски на две части и покрасит некоторые в желтый, а оставшиеся — в зеленый цвет. Она сама выберет, какие полоски как покрасить. Далее она хочет из этих полосок сплести максимально большую плетенку. Она расположит полоски одного цвета в некотором порядке горизонтально, а полоски другого цвета в некотором порядке вертикально. После этого она переплетет горизонтальные и вертикальные полоски так, что они будут чередоваться то сверху, то снизу, образуя в местах пересечения шахматную раскраску. Наконец, она обрежет выступающие края полосок так, что останется прямоугольная плетенка с ровными краями. Каждая клетка полученной плетенки должна иметь два слоя.

Маша хочет сплести максимально большую по площади прямоугольную плетенку. Подскажите ей, плетенку какой площади она сможет сделать. Заметим, что она может при создании плетенки использовать не все имеющиеся у нее полоски.

Формат входных данных

В первой строке на вход подается число n — количество полосок бумаги у Маши, где $2 \leqslant n \leqslant 2 \cdot 10^5$. Во второй строке через пробел заданы n целых чисел a_i через пробел — длины полосок, где $1 \leqslant a_i \leqslant 10^9$.

Формат выходных данных

Вывести одно число — площадь прямоугольника, форму которого может иметь самая большая плетенка Маши.

Примеры

Пример №1

Стандартный ввод
8 3 6 5 4 4 5 5 2

Стандартный вывод
12

Примечания

На рис. 2.2.6 представлен один из вариантов получения самой большой плетенки для полосок из примера. Синим обозначена граница полученной максимальной плетенки. Ее размер 3×4, и ее площадь 12. При ее создании Маша не должна использовать полоску номер 8, по этой причине неважно, как она раскрашена.

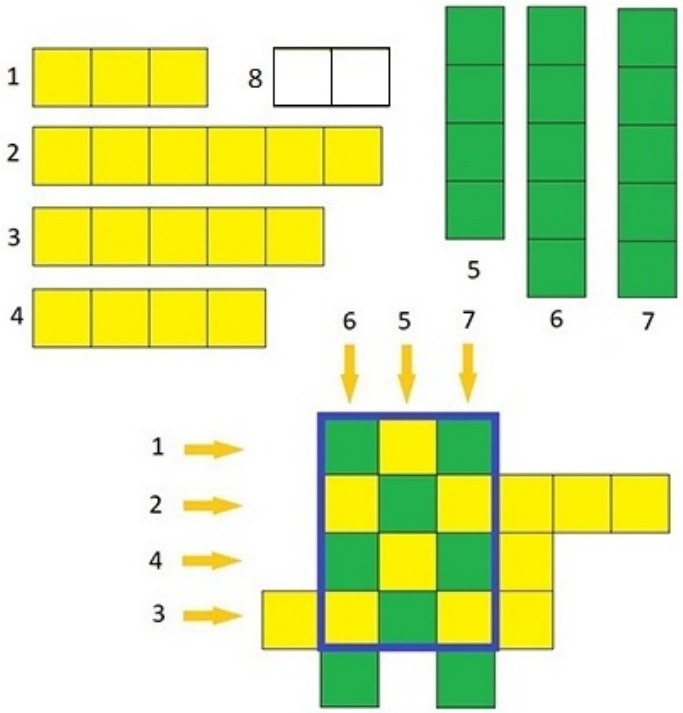

Рис. 2.2.6

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int n;
    cin >> n;
    deque<int> v(n);
    for(int i = 0; i < n; i++){
        cin >> v[i];
    }
    sort(v.begin(), v.end());
    int ans = 0;
    int cnth = 0, minh;
    while(1){
        if(v.size() == 0){
            break;
        }
        cnth++;
        minh = v.back();
        v.pop_back();
        while(v.size() > 0 && v[0] < cnth){
            v.pop_front();
        }
        ans = max(ans, cnth * min(minh, (int)v.size()));
    }
    cout << ans << endl;
}
```

Задача 2.2.3.5. Английский в игровой форме (30 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 3 с.

Ограничение по памяти: 64 Мбайт.

Условие

Маша и Витя запоминают слова английского языка в оригинальной игровой форме. За день им нужно выучить n слов, где $20 \leqslant n \leqslant 100$, каждое из которых имеет длину от 5 до 8 символов. Маша выбирает из этого набора наугад несколько попарно различных слов (также от 5 до 8) и собирает их в одну строку без пробелов. Далее она переставляет буквы в этой строке так, что слова оказываются полностью перепутанными, и дает эту строку Вите. Теперь Витя должен восстановить все слова, которые выбрала Маша.

Но у Вити плохо получается, а Маша уже забыла, какие слова она выбрала. Нужно им помочь — написать программу, которая восстановит слова, выбранные Машей.

Формат входных данных

В первой строке находится строка, которую Маша предложила Вите. Во второй строке содержится число n — количество слов, которые нужно выучить детям, $20 \leqslant \leqslant n \leqslant 100$.

В следующих n строках содержатся эти слова по одному в строке. Все слова в этом наборе различны. Слова отсортированы в лексикографическом (алфавитном) порядке. Все слова состоят из маленьких букв от a до z. Обратите внимание, что в тестах к этой задаче все заданные слова реально существуют в английском языке и случайным образом выбраны из словаря.

Гарантируется, что длина каждого слова из предложенного набора (словаря) в пределах от 5 до 8, строка, которую получила Маша, может быть получена путем перестановки букв некоторых различных слов из предложенного словаря, причем, набор выбранных Машей слов определяется по ней однозначно. Количество слов, из которых составлена Машина строка, находится в пределах от 5 до 8.

Формат выходных данных

Вывести все слова, выбранные Машей, в алфавитном порядке по одному в строке.

Примеры

Пример №1

Стандартный ввод
stirbaexsudueoeidgomttcrnrwlunapntetacwri 24 bridge cranky document drawing farmer fighter figurine gravy havoc minimum reactant reply republic sonata soprano split subset tailor texture tomorrow trout vicinity wrist writer

Стандартный вывод
document drawing republic sonata texture wrist

Комментарий

В случае, выделенном в условии (слова являются случайными, взятыми из английского словаря), задача решается рекурсией с перебором вариантов.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
string frs;
int n;
vector<string> dict;
vector<int> msk(26, 0);
int cnt = 0;
vector<vector<int> > amsk;
vector<string> ans;
bool bigok = 0;
void p(int pos){
   if(!bigok){
        if(cnt == 0){
            sort(ans.begin(), ans.end());
            bigok = 1;
            return;
        }
        for(int i = pos; i < n; i++){
            string ts = dict[i];
            bool ok = 1;
            for(int j = 0; j < 26; j++){
                if(amsk[i][j] > msk[j]){
                    ok = 0;
                }
            }
            if(ok){
                ans.push_back(ts);
                for(int j = 0; j < 26; j++){
                    msk[j] -= amsk[i][j];
                    cnt -= amsk[i][j];
                }
                p(i + 1);
                if(!bigok){
                for(int j = 0; j < 26; j++){
                    msk[j] += amsk[i][j];
                    cnt += amsk[i][j];
                }
                ans.pop_back();
                }
            }
        }
   }
}
signed main(){
    cin >> frs;
    cin >> n;
    amsk.resize(n, vector<int>(26, 0));

    string ts;
    for(int i = 0; i < n; i++){
        cin >> ts;
        dict.push_back(ts);
    }
    for(int i = 0; i < n; i++){
        for(auto el : dict[i]){
            amsk[i][el - 'a']++;
        }
    }
```

```
60      for(auto el : frs){
61          msk[el - 'a']++;
62          cnt++;
63      }
64      p(0);
65      for(auto el : ans){
66          cout << el << endl;
67      }
68  }
```

2.2.4. Четвертая волна. Задачи 8–11 класса

Задачи четвертой волны предметного тура по информатике открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63457/enter/.

Задача 2.2.4.1. Квадратный флаг (10 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Одному портному заказали сделать одноцветный флаг. Особенность этого флага в том, что он должен быть квадратным. У портного есть два прямоугольных куска ткани заданного цвета. Один из них имеет размеры $a \times b$, другой — $c \times d$. Так как клиент будет платить пропорционально площади изготовленного флага, портной хочет сначала сшить имеющиеся у него прямоугольные куски, соединив их двумя какими-то сторонами, а затем из полученного полотна вырезать и сделать флаг с максимально большой стороной. Определить сторону получившегося у него флага.

Формат входных данных

На вход подаются две строки. В первой строке находятся размеры первого прямоугольника — целые числа a, b через пробел, во второй — размеры второго прямоугольника, также целые числа c, d через пробел, где $1 \leqslant a, b, c, d \leqslant 10^9$.

Формат выходных данных

Вывести одно число — сторону самого большого квадрата, который можно получить по условию задачи.

Примеры

Пример №1

Стандартный ввод
2 4 3 6

Стандартный вывод
4

Пример №2

Стандартный ввод
2 2 3 6

Стандартный вывод
3

Примечания

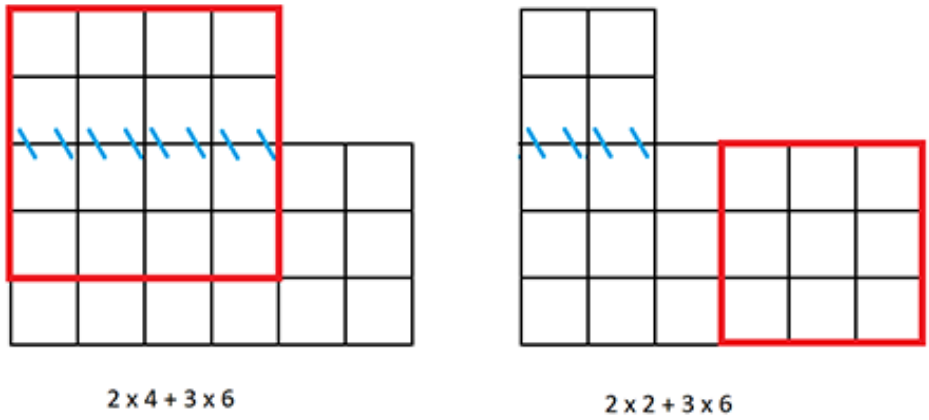

Рис. 2.2.7

На рис. 2.2.7 представлены иллюстрации для тестов из условия. Синими штрихами обозначено место сшивки двух кусков. Красный квадрат выделяет один из вариантов вырезания максимального квадрата.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int a, b, c, d;
    cin >> a >> b >> c >> d;
    int ans = max(min(a, b), min(c, d));
    int p1 = min(a + c, min(b, d));
    int p2 = min(a + d, min(b, c));
    int p3 = min(b + c, min(a, d));
    int p4 = min(b + d, min(a, c));
    ans = max({ans, p1, p2, p3, p4});
    cout << ans << endl;
}
```

Задача 2.2.4.2. Потерянная ДНК (15 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

В данной задаче будем упрощенно считать, что ДНК представляется строкой длины от 10 до 100, состоящей из букв A, C, G, T.

Пусть даны две ДНК D_1 и D_2 одной и той же длины n. Выберем некоторое произвольное число i от 1 до $n-1$ и поменяем местами префиксы (начала) этих ДНК длины i. Будем говорить, что полученные новые две строки образованы путем скрещивания двух исходных по префиксу длины i.

Например, пусть $D_1 =$ **AACGGTAGGT**, а $D_2 =$ TCCCGGAACA. Выберем $i = 4$ и поменяем местами префиксы длины 4. Получим две новые ДНК, одна из которых будет иметь вид **AACG**GGAACA, а вторая — TCCC**GTAGGT**. Для наглядности были выделены части первой из них.

Полученные новые ДНК снова могут быть скрещены по любому префиксу длины от 1 до $n-1$.

Теперь можно рассмотреть популяцию из нескольких ДНК. Выберем из них две, произведем их скрещивание по префиксу какой-либо длины и поместим две новые ДНК в исходную популяцию. В данной задаче будем считать, что количество ДНК не увеличивается, то есть старые две ДНК заменяются на новые две ДНК.

Дана исходная популяция из m ДНК, каждая имеет одну и ту же длину n. После некоторого количества попарных скрещиваний была получена новая популяция. Но при итоговой обработке данных сведения об одной ДНК из новой популяции были потеряны. Задача состоит в отыскании этой потерянной ДНК по оставшимся $m-1$ ДНК из новой популяции.

Формат входных данных

В первой строке через пробел даны два числа n — длина ДНК и m — количество ДНК в исходной популяции, где $10 \leqslant n \leqslant 100$, $2 \leqslant m \leqslant 100$.

В следующих m строках содержится описание исходной популяции ДНК, каждая задается строкой длины n, состоящей из символов A, C, G и T.

Далее следует разделяющая строка, содержащая n символов «$-$».

Далее следует еще $m - 1$ строк, описывающих новую (заключительную) популяцию без одной ДНК.

Гарантируется, что данные верны, то есть $m - 1$ последняя ДНК является некоторой новой популяцией ровно без одной ДНК, полученной из исходной популяции, заданной в m первых строках.

Формат выходных данных

Вывести недостающую утерянную ДНК.

Примеры

Пример №1

Стандартный ввод
10 2 AACGGTAGGT TCCCGGAACA ---------- TCCCGTAGGT

Стандартный вывод
AACGGGAACA

Пример №2

Стандартный ввод
10 4 AACCGGTTAA ACGTACGTAC AAACCCGGGT CATTACTGGA ---------- AAGCGCTTAA CCACACGTGC AACTAGGGGT

Стандартный вывод
AATTCCTGAA

Комментарий

Для каждой позиции нужно найти недостающую букву из первого набора ДНК. Для этого удобнее всего использовать функцию `xor`.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int n, m;
    cin >> n >> m;
    vector<string> v1(m);
    for(int i = 0; i < m; i++){
        cin >> v1[i];
    }
    string d;
    cin >> d;
    vector<string> v2(m - 1);
    for(int i = 0; i < m - 1; i++){
        cin >> v2[i];
    }
    for(int j = 0; j < n; j++){
        int ss = 0;
        for(int i = 0; i < m; i++){
            ss ^= (int)(v1[i][j]);
        }
        for(int i = 0; i < m - 1; i++){
            ss ^= (int)(v2[i][j]);
        }
        cout << (char)(ss);
    }
    cout << endl;
}
```

Задача 2.2.4.3. Утомленные туристы (20 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Рассмотрим следующий вариант известной задачи на перемещение по туннелю группы из четырех человек. В общем виде она выглядит так: четыре туриста хотят пройти по темному туннелю. Имеется один фонарик. По туннелю можно перемещаться либо вдвоем, либо по одному, при этом у тех, кто движется в туннеле,

должен быть фонарик в руках. По этой причине движение должно быть следующим: двое переходят туда, один возвращается обратно и приносит фонарик тем, кто еще не перешел. После этого указанный маневр повторяется снова.

У каждого участника своя скорость движения в туннеле. Пусть участники проходят туннель за A, B, C и D мин. Если идут двое, то они движутся со скоростью того, кто идет медленнее. Требуется по заданным временам прохождения туннеля каждого из участников перевести их максимально быстро через туннель.

Немного усложним данную задачу. Введем фактор усталости. А именно, любой участник, пройдя по туннелю, устает и в следующий раз идет уже медленнее. После каждого прохождения туннеля время прохождения любого участника увеличивается на E мин. Например, если участник до начала движения проходит туннель за 1 мин, а показатель усталости E равен 3 мин, то первый раз участник пройдет туннель за 1 мин, второй раз — за 4 мин, третий раз — за 7 мин и т. д.

По заданным A, B, C, D и E узнать, за какое минимальное время можно провести всю группу через туннель согласно указанным правилам.

Формат входных данных

На вход подаются пять чисел. В первой строке через пробел четыре числа A, B, C и D — время прохождения туннеля каждым из четырех участников до того, как они начали движение. Во второй строке содержится число E — величина, на которую увеличивается время прохождения туннеля каждым участником после каждого перемещения. При этом $1 \leqslant A, B, C, D \leqslant 1\,000$, $0 \leqslant E \leqslant 1\,000$.

Формат выходных данных

Вывести одно число — минимальное время прохождения туннеля всей группой.

Примеры

Пример №1

Стандартный ввод
8 9 10 1 3
Стандартный вывод
44

Пример №2

Стандартный ввод
8 9 10 1 0
Стандартный вывод
29

Примечания

В первом примере при прохождении туннеля каждый турист устает и движется медленнее на 3 мин. Покажем, как перевести группу при этом за 44 мин.

Каждую ситуацию будем обозначать следующим образом: слева от двоеточия находятся туристы, которые стоят в начале туннеля, а справа — те, что стоят в конце туннеля. Туриста будем обозначать при помощи числа, соответствующего его текущему времени прохождения туннеля.

Тогда исходная ситуация имеет вид 1, 8, 9, 10 :.

Сначала идут туристы 1 и 8, каждый после перехода устает на 3 мин, получим ситуацию 9, 10 : 4, 11, затрачено 8 мин.

Обратно возвращается турист 4, он устает еще на 3 мин. Ситуация становится 7, 9, 10 : 11, затрачено $8 + 4 = 12$ мин.

Теперь идут туристы 7 и 9, получится ситуация 10 : 10, 11, 12, затрачено $8 + + 4 + 9 = 21$ мин.

Возвращается турист 10, получится 10, 13 : 11, 12, затрачено $8 + 4 + 9 + 10 = = 31$ мин.

Наконец, оставшиеся двое туристов 10 и 13 за 13 мин переходят туннель, итого затрачено $8 + 4 + 9 + 10 + 13 = 44$ мин.

Комментарий

Задача решается рекурсивным перебором всех вариантов прохождения.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int INF = 1e18;
vector<int> v(4);
int e, ans = INF;
void p(vector<int> &vl, vector<int> &vr, int tv){
    if(vl.size() == 2){
        ans = min(ans, tv + *max_element(vl.begin(), vl.end()));
        return;
    }
    for(int i = 0; i < vl.size() - 1; i++){
        for(int j = i + 1; j < vl.size(); j++){
            vector<int> vl1;
            for(int k = 0; k < vl.size(); k++){
                if(k != i && k != j){
                    vl1.push_back(vl[k]);
                }
            }
            vector<int> vr1 = vr;
```

```
21                vr1.push_back(vl[i] + e);
22                vr1.push_back(vl[j] + e);
23                int tmp = max(vl[i], vl[j]);
24                sort(vr1.rbegin(), vr1.rend());
25                vl1.push_back(vr1.back() + e);
26                vr1.pop_back();
27                p(vl1, vr1, tv + tmp + vl1.back() - e);
28            }
29        }
30    }
31    signed main(){
32        for(int i = 0; i < 4; i++){
33            cin >> v[i];
34        }
35        sort(v.begin(), v.end());
36        cin >> e;
37        vector<int> vl = v, vr;
38        p(vl, vr, 0);
39        cout << ans;
40    }
```

Задача 2.2.4.4. Проектируем мост (25 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

При постройке моста используются два типа пролетов: П-образные (они прочные, но дорогие) и Т-образные (они дешевле, но менее надежные). Мост должен начинаться и заканчиваться П-образными пролетами. Любой Т-образный пролет должен иметь хотя бы один П-образный пролет в качестве соседнего.

Длина проектируемого моста — n пролетов. Муниципалитет выделил средства на постройку a П-образных и b Т-образных пролетов. При этом $a + b = n$. Требуется выяснить, сколькими способами при этих условиях можно скомпоновать мост. Два способа компоновки моста отличаются, если в одной на некоторой позиции стоит П-образный пролет, а в другой на этой же позиции стоит Т-образный пролет.

Формат входных данных

В одной строке через пробел заданы два числа: a — число П-образных пролетов и b — число Т-образных пролетов, на постройку которых выделены средства, где $2 \leqslant a \leqslant 10^6$, $0 \leqslant b \leqslant 10^6$.

Формат выходных данных

Вывести одно число — количество вариантов компоновки моста. Так как ответ может быть очень большим, требуется вывести остаток от его деления на 1 000 000 007 ($10^9 + 7$).

Примеры

Пример №1

Стандартный ввод
4 3

Стандартный вывод
7

Примечания

Для примера из условия имеется 7 вариантов компоновки моста (пробелы добавлены для лучшего восприятия вариантов):

П Т Т П Т П П

П Т Т П П Т П

П Т П Т Т П П

П Т П П Т Т П

П П Т П Т Т П

П П Т Т П Т П

П Т П Т П Т П

Комментарий

При заданных ограничениях задача решается только при помощи комбинаторики с вычислениями по модулю.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int INF = 1e18;
const int MOD = 1e9 + 7;
vector<int> f(2e6 + 1, 1);
```

```
int binpow (int a, int n) {
        int res = 1;
        while (n > 0) {
                if (n % 2 == 1)
                        (res *= a) %= MOD;
                (a *= a) %= MOD;
                n /= 2;
        }
        return res;
}

int bc(int n, int k){
    int res = f[n];
    int p1 = binpow(f[k], MOD - 2);
    int p2 = binpow(f[n - k], MOD - 2);
    (res *= p1) %= MOD;
    (res *= p2) %= MOD;
    return res;
}
signed main(){
    for(int i = 1; i <= 2e6; i++){
        f[i] = (f[i - 1] * i) % MOD;
    }
    int a, b;
    int ans = 0;
    cin >> a >> b;
    a--;
    for(int i = 0; i < a + 1; i++){
        if(2 * i <= b){
            int d = bc(a, i);
            if(b - 2 * i <= a - i){
                (d *= bc(a - i, b - 2 * i) ) %= MOD;
                (ans += d) %= MOD;
            }
        }
    }
    cout << ans << endl;
}
```

Задача 2.2.4.5. Джентльмены на прогулке (30 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 8 с.

Ограничение по памяти: 64 Мбайт.

Условие

По прямому участку улицы, которую будем считать отрезком AB длины d, прогуливаются n джентльменов. i-й джентльмен движется со скоростью v_i. Скорости всех джентльменов попарно различны. Дойдя до любого конца улицы, каждый джентльмен поворачивает и идет в обратную сторону.

При каждой встрече два джентльмена приветствуют друг друга, приподнимая

головной убор. Приветствие происходит и в том случае, когда один джентльмен обгоняет другого. Если два джентльмена встречаются в момент их одновременного поворота, то происходит два приветствия: одно до поворота, другое — после поворота. Если происходит одновременная встреча трех и более джентльменов, то они приветствуют друг друга попарно, то есть каждый каждого. Допустим, если одновременно встретились четыре джентльмена где-то посреди улицы, произойдет шесть попарных приветствий. Если же эти четыре джентльмена встретились в момент их одновременного поворота, произойдет уже двенадцать приветствий.

В этой задаче считаем, что все действия происходят без остановок, то есть и повороты и приветствия происходят мгновенно. Джентльмены одновременно начинают свою прогулку из точки A в момент 0. В этот момент они уже производят свои первые попарные приветствия, то есть в момент 0 уже произведено $n \cdot (n-1)/2$ приветствий. Момент старта не считается моментом поворота, то есть на старте число приветствий не удваивается. Джентльмены гуляют достаточно долго, чтобы произошло любое заданное количество приветствий.

Требуется найти момент, в который было произведено k-е по порядку приветствие.

Формат входных данных

В первой строке ввода через пробел содержится два целых числа: d — длина отрезка AB и n — количество прогуливающихся джентльменов, где $1 \leqslant d \leqslant 200$, $2 \leqslant n \leqslant 2\,000$.

Во второй строке находятся n целых чисел v_i через пробел — скорости каждого джентльмена, где $1 \leqslant v_i \leqslant 2\,000$. Гарантируется, что все скорости попарно различны. Скорости даны в порядке возрастания, то есть $v_1 < v_2 < \ldots < v_n$.

В третьей строке содержится одно целое число k — номер требуемого приветствия, для которого нужно найти момент, когда оно произойдет, где $1 \leqslant k \leqslant 10^9$.

Формат выходных данных

Вывести одно вещественное число — время, когда произойдет k-е по порядку приветствие. Ответ вывести с точностью не менее двух знаков после десятичной точки.

Примеры

Пример №1

Стандартный ввод
5 4 2 5 8 10 6

Стандартный вывод
0.000

Пример №2

Стандартный ввод
5 4 2 5 8 10 7
Стандартный вывод
0.556

Пример №3

Стандартный ввод
5 4 2 5 8 10 11
Стандартный вывод
1.000

Пример №4

Стандартный ввод
5 4 2 5 8 10 15
Стандартный вывод
1.429

Пример №5

Стандартный ввод
5 4 2 5 8 10 17
Стандартный вывод
1.667

Пример №6

Стандартный ввод
5 4 2 5 8 10 19
Стандартный вывод
1.667

Пример №7

Стандартный ввод
5 4 2 5 8 10 21
Стандартный вывод
2.000

Примечания

На рис. 2.2.8 приведено положение джентльменов из примеров в моменты времени 0, 1 и 2. Джентльмены обозначены своими скоростями. Стрелками обозначены направления их движения в соответствующий момент. Перечислим и пронумеруем в порядке возрастания моменты попарных приветствий этих джентльменов до момента времени 2 включительно. Если два и более приветствия происходят одновременно, неважно какое из них конкретно имеет номер k, главное, что они происходят в один и тот же определенный момент времени.

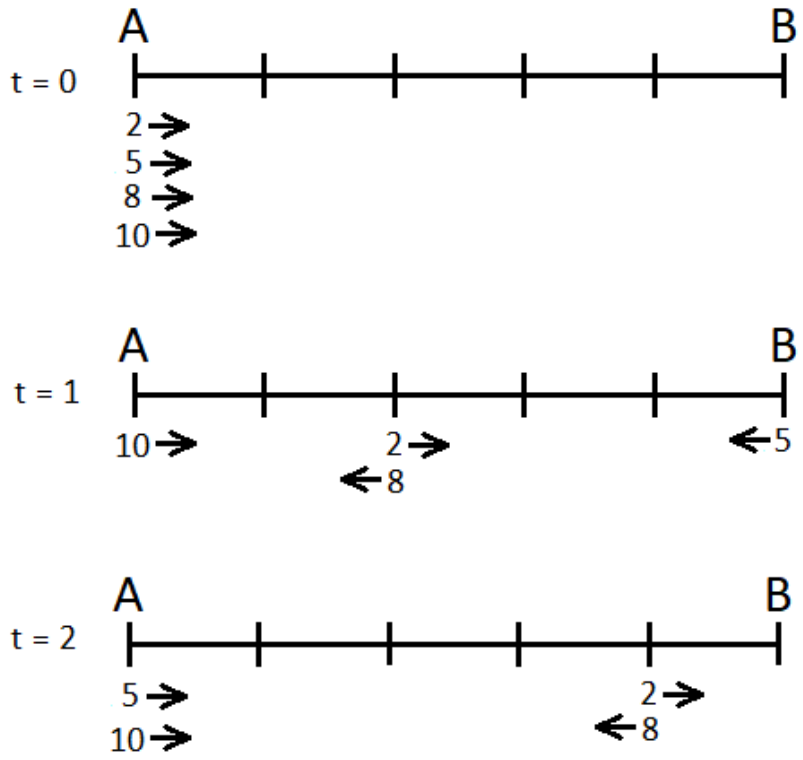

Рис. 2.2.8

1. 2 и 5 приветствуют друг друга в момент 0 (изображено на рис. 2.2.8).
2. 2 и 8 приветствуют друг друга в момент 0 (изображено на рис. 2.2.8).
3. 2 и 10 приветствуют друг друга в момент 0 (изображено на рис. 2.2.8).
4. 5 и 8 приветствуют друг друга в момент 0 (изображено на рис. 2.2.8).
5. 5 и 10 приветствуют друг друга в момент 0 (изображено на рис. 2.2.8).

6. 8 и 10 приветствуют друг друга в момент 0 (изображено на рис. 2.2.8).
7. 8 и 10 приветствуют друг друга в момент 0.556.
8. 5 и 10 приветствуют друг друга в момент 0.667.
9. 5 и 8 приветствуют друг друга в момент 0.769.
10. 2 и 10 приветствуют друг друга в момент 0.833.
11. 2 и 8 приветствуют друг друга в момент 1.000 (изображено на рис. 2.2.8).
12. 8 и 10 приветствуют друг друга в момент 1.111.
13. 2 и 10 приветствуют друг друга в момент 1.250.
14. 5 и 10 приветствуют друг друга в момент 1.333.
15. 2 и 5 приветствуют друг друга в момент 1.429.
16. 5 и 8 приветствуют друг друга в момент 1.538.
17. 2 и 8 приветствуют друг друга в момент 1.667.
18. 2 и 10 приветствуют друг друга в момент 1.667.
19. 8 и 10 приветствуют друг друга в момент 1.667 (в момент 1.667 встретятся одновременно три джентльмена 2, 8 и 10).
20. 2 и 8 приветствуют друг друга в момент 2.000 (изображено на рис. 2.2.8).
21. 5 и 10 приветствуют друг друга в момент 2.000 (до поворота).
22. 5 и 10 приветствуют друг друга в момент 2.000 (после поворота, изображено на рис. 2.2.8).

Комментарий

Задача решается при помощи бинпоиска с квадратичным нахождением ответа в каждой его итерации.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
const double EPS = 1e-7;
double x(double M, int V, int d){
    double dst = V * M;
    int cnt = floor((dst + EPS) / d);
    double pin = dst - cnt * d;
    if(cnt % 2 == 0){
        return pin;
    }
    else{
        return d - pin;
    }
}
int F(double M, vector<int> &v, int d){
    int res = 0;
    for(int i = 0; i < v.size(); i++){
        double dst = v[i] * M;
```

```
            int cnt = floor((dst + EPS) / d);
            res += cnt * i;
            double tx = x(M, v[i], d);
            for(int j = 0; j < i; j++){
                double txj = x(M, v[j], d);
                if(cnt % 2 == 0){
                    res += txj <= tx + EPS;
                }
                else{
                    res += txj >= tx - EPS;
                }
            }
        }
        return res;
}
signed main(){
    int d, n;
    cin >> d >> n;
    vector<int> v(n);
    for(int i = 0; i < n; i++){
        cin >> v[i];
    }
    int k;
    cin >> k;
    double L = 0, R = 1;
    while(F(R, v, d) <= k){
        R *= 2;
    }
    R *= 2;
    while(R - L > 1e-4){
        double M = (R + L) / 2.0;
        if(F(M, v, d) < k){
            L = M;
        }
        else{
            R = M;
        }
    }
    cout.precision(10);
    cout << fixed << L << endl;
}
```

2.3. Предметный тур. Физика

2.3.1. Первая волна. Задачи 8–9 класса

Задачи первой волны предметного тура по по физике за 8–9 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63463/enter/.

Задача 2.3.1.1. Калориметр (10 баллов)

Условие

Внутренний стакан калориметра представляет собой цилиндр с радиусом $R = 8$ см и высотой $3R$. Внешний стакан также имеет форму цилиндра, стенки которого (как боковые, так и торцы) отстоят от стенок внутреннего на расстояние R. Какая масса теплоизоляционного материала с плотностью $\rho = 25$ кг/м^3 необходима, чтобы полностью заполнить пространство между стаканами?

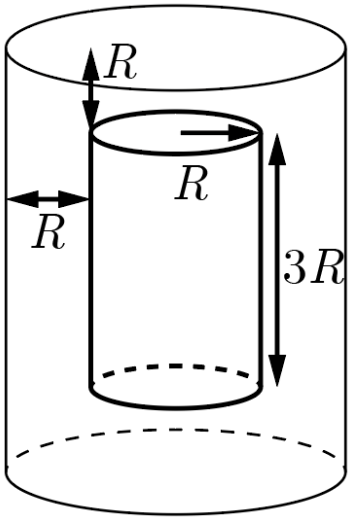

Рис. 2.3.1

Решение

Чтобы найти массу m теплоизоляционного материала, необходимо его плотность умножить на занимаемый им объем V:

$$m = \rho V. \tag{2.3.1}$$

Объем области, заполняемой теплоизоляцией, удобнее всего найти, вычтя из объема V_1 большого внешнего стакана объем V_2 маленького внутреннего. Для любого кругового цилиндра с высотой h и радиусом r объем может быть найден по

формуле $V = \pi r^2 h$. В случае большого цилиндра $r = 2R$ и $h = 5R$, следовательно,

$$V_1 = 5R \cdot \pi(2R)^2 = 20\pi R^3. \tag{2.3.2}$$

Аналогично, для маленького $r = R$, $h = 3R$ и, следовательно,

$$V_2 = 3\pi R^3. \tag{2.3.3}$$

Подставляя (2.3.2), (2.3.3) в (2.3.1), получим, что искомая масса составляет:

$$m = \rho(V_1 - V_2) = 17\pi R^3 \rho \approx 0{,}68\,\text{кг}. \tag{2.3.4}$$

Погрешность 0,01 кг.

Ответ: $m = 17\pi R^3 \rho = (0{,}68 \pm 0{,}01)$ кг.

Задача 2.3.1.2. Нить накала (15 баллов)

Условие

Нити накала ламп изготавливают из вольфрама, удельное сопротивление которого сильно зависит от температуры. По мере прогрева нити оно возрастает от $\rho_0 = 5{,}5 \cdot 10^{-8}$ Ом · м до $\rho_1 = 1{,}1 \cdot 10^{-6}$ Ом · м. Определите электрическую мощность, потребляемую лампой в первый момент после ее включения, если в рабочем режиме (полностью прогревшись) лампа потребляет от той же сети мощность $P_1 = 30$ Вт.

Решение

Электрическая мощность P, потребляемая нитью накала, может быть вычислена по закону Джоуля – Ленца, который для фиксированного напряжения U в сети удобно записать как

$$P = U^2/R. \tag{2.3.5}$$

При этом сопротивление R нити легко выразить через ее удельное сопротивление ρ, длину l и площадь поперечного сечения S

$$R = \frac{\rho l}{S}. \tag{2.3.6}$$

Подставляя (2.3.6) в (2.3.5), получим

$$P = \frac{U^2 S}{\rho l},$$

где только ρ зависит от температуры. В результате придем к выводу, что выделяющаяся в лампе мощность обратно пропорциональна ее удельному сопротивлению, откуда окончательно следует

$$P_0 = P_1 \frac{\rho_1}{\rho_0} \approx 600\,\text{Вт}.$$

Погрешность 1 Вт.

Ответ: $P_0 = (600 \pm 1)$ Вт.

Задача 2.3.1.3. Свая (20 баллов)

Условие

Бетонная свая высотой $h = 1{,}4$ м и массой $m = 160$ кг полностью погружена в грунт так, что ее верхний торец совпадает с уровнем почвы. К сожалению, сваю понадобилось извлечь. Определите, какую работу для этого необходимо совершить, если сила трения со стороны грунта, действующая на сваю, прямо пропорциональна площади соприкосновения ее боковой стороны с землей и в начальный момент ее извлечения равна $F = 4$ кН. Ускорение свободного падения $g \approx 9{,}8$ м/с^2.

Решение

Работа A, необходимая для извлечения сваи, складывается из увеличения потенциальной энергии сваи на величину mgh и работы по преодолению силы трения $A_{\text{тр}}$. Последняя должна быть найдена с учетом постепенного уменьшения силы трения $F_{\text{тр}}$ от максимального значения F до нуля. Поскольку это уменьшение происходит линейно, общая работа оказывается строго вдвое меньше, чем при постоянном значении $F_{\text{тр}} = F$ (аналогично тому, как вычисляется значение работы сил упругости пружины). В результате

$$A = mgh + \frac{Fh}{2} \approx 5\,\text{кДж}. \tag{2.3.7}$$

Погрешность $0{,}1$ кДж.

Ответ: $(5{,}0 \pm 0{,}1)$ кДж.

Задача 2.3.1.4. Двое из ларца (25 баллов)

Условие

Два дрона одновременно вылетают с общей пусковой станции и движутся по прямолинейным траекториям. Первый дрон на начальном этапе движения перемещается с постоянной скоростью $v_1 = 15$ м/с, а через время $\tau = 40$ с быстро переключается на движение с постоянной скоростью $v_2 = 20$ м/с. Второй дрон — наоборот, сначала движется со скоростью v_2, а через время τ переключается на скорость v_1. Наконец, дроны одновременно заканчивают полет. Определите, как долго длился этот полет, если по его итогам средняя путевая скорость первого дрона оказалась на $\Delta v = 1$ м/с выше, чем средняя путевая скорость второго.

Решение

По определению средняя путевая скорость — это отношение общего пройденного пути S к общему времени движения t:

$$v = \frac{S}{t}. \tag{2.3.8}$$

Для первого дрона это уравнение принимает вид

$$v_a = \frac{v_1\tau + v_2(t-\tau)}{t}, \tag{2.3.9}$$

а для второго, соответственно,

$$v_b = \frac{v_2\tau + v_1(t-\tau)}{t}. \tag{2.3.10}$$

Из условий задачи известно, что $v_a - v_b = \Delta v$. Подставляя в это уравнение формулы (2.3.9) и (2.3.10), а также домножая его на t, получим

$$v_1\tau + v_2(t-\tau) - v_2\tau - v_1(t-\tau) = \Delta v t. \tag{2.3.11}$$

Перегруппировав слагаемые, получим

$$v_2 t - 2v_2\tau - v_1 t + 2v_1\tau = \Delta v t, \tag{2.3.12}$$

откуда окончательно

$$t = \frac{2(v_2 - v_1)\tau}{v_2 - v_1 - \Delta v} = 100\,\text{с}. \tag{2.3.13}$$

Погрешность 1 с.

Ответ: (100 ± 1) с.

Задача 2.3.1.5. Призма (30 баллов)

Условие

Для тонкого контроля параметров призмы используется следующая установка: отмечается точка, в которую падает лазерный луч, направленный на экран строго под прямым углом (пунктирный на рисунке). Затем на пути луча устанавливается исследуемая призма так, что задняя (первая по ходу распространения луча) ее грань оказывается строго перпендикулярна лучу, и измеряется расстояние d, на которое в результате этого смещается пятно лазера.

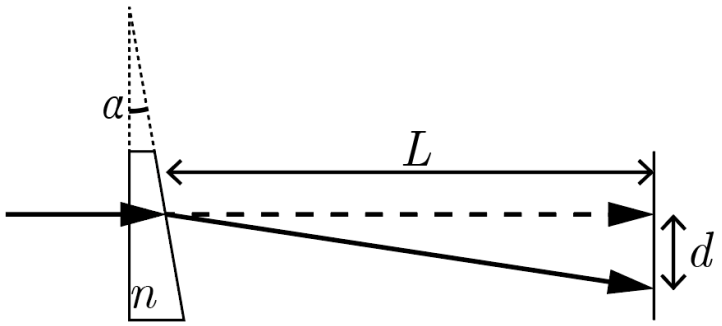

Рис. 2.3.2

Определите показатель преломления стекла, из которого изготовлена призма, если расстояние от передней грани призмы до экрана $L = 3$ м, смещение пятна при установке призмы $d = 12$ см, а угол между передней и задней поверхностями призмы $\alpha = 3°$. Используйте приближение малых углов.

Решение

На первой по ходу распространения луча грани призмы свет не преломляется, поскольку падает на нее под прямым углом. Следовательно, угол падения луча на переднюю грань призмы равен α. Тогда по закону Снеллиуса

$$n = \frac{\sin\beta}{\sin\alpha}, \tag{2.3.14}$$

где β — угол преломления луча, что с учетом приближения малых углов $\sin\alpha \approx \alpha \approx \approx \operatorname{tg}\alpha$ (в радианах) принимает форму

$$n \approx \frac{\beta}{\alpha}. \tag{2.3.15}$$

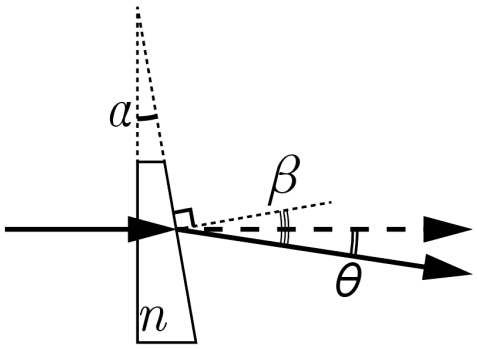

Рис. 2.3.3

Из геометрии рисунка легко видеть, что $\beta = \alpha + \theta$, где θ — угол, который преломленный луч составляет с направлением своего распространения до установки призмы. При этом $\operatorname{tg}\theta = \frac{d}{L}$, откуда

$$n\alpha \approx \beta = \alpha + \operatorname{arctg}\frac{d}{L} \Rightarrow n \approx 1 + \frac{d}{\alpha L} \approx 1{,}76. \tag{2.3.16}$$

Погрешность 0,02.

Ответ: $1{,}76 \pm 0{,}02$.

2.3.2. Первая волна. Задачи 10–11 класса

Задачи первой волны предметного тура по физике за 10–11 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63480/enter/.

Задача 2.3.2.1. Беспилотник (10 баллов)

Условие

Беспилотник, двигаясь равномерно и прямолинейно и обладая при этом импульсом $p_0 = 10^4$ кг $\cdot$ м/с, преодолевает дистанцию $L = 20$ км ровно за 1,5 мин. За какое

время преодолеет ее этот же беспилотник, двигаясь также равномерно и прямолинейно, но обладая на $\Delta p = 10^3$ кг $\cdot$ м/с меньшим импульсом?

Решение

Импульс p тела — это произведение его массы на его скорость, поэтому скорость беспилотника легко может быть вычислена как

$$p_0 = \frac{mL}{t_0}, \tag{2.3.17}$$

где t_0 — время в пути с импульсом p_0. Искомое время t можно аналогично выразить через скорость v беспилотника во второй рассмотренной ситуации как

$$t = \frac{L}{v} = \frac{Lm}{p_0 - \Delta p}. \tag{2.3.18}$$

Выражая массу беспилотника из 2.3.17 и подставляя ее в 2.3.18, получим:

$$t = \frac{\cancel{L} p_0 t_0}{\cancel{L}(p_0 - \Delta p)} = 100\,\text{с}. \tag{2.3.19}$$

Погрешность 1 с.

Ответ: (100 ± 1) с.

Задача 2.3.2.2. Грузовик (15 баллов)

Условие

На плоское горизонтальное дно кузова транспортного грузовика погрузили большой грузовой контейнер и забыли его закрепить. Благодаря силе трения контейнер оставался в покое относительно грузовика до тех пор, пока ускорение последнего не превосходило $a_0 = 2{,}0\,\text{м/с}^2$, и начинал скользить при превышении этого значения. Совершая маневр, грузовик приобрел ускорение $a = 2{,}12\,\text{м/с}^2$, направленное по ходу движения. Какое время длился маневр, если в результате контейнер сдвинулся на $l = 1{,}5\,\text{м}$ относительно грузовика?

Решение

Исходя из того, что контейнер остается на месте при ускорении грузовика до a_0, можно, по второму закону Ньютона, заключить, что сила трения покоя, обеспечивающая это ускорение для контейнера, не превышает значения

$$F = ma_0, \tag{2.3.20}$$

где m — масса контейнера. При любом маневре, при котором контейнер начинает скользить, на него действует сила трения скольжения, равная F и, следовательно, его ускорение относительно дороги оказывается равно a_0.

Во время маневра ускорение контейнера относительно грузовика равно (по модулю) $a - a_0$, а пройденное контейнером относительно грузовика расстояние может быть выражено по законам кинематики как

$$l = \frac{(a - a_0)t^2}{2}, \tag{2.3.21}$$

где t — искомое в задаче время. Преобразуя эту формулу, получим

$$t = \sqrt{\frac{2l}{a - a_0}} = 5\,\text{с}. \tag{2.3.22}$$

Погрешность $0{,}1$ с.

Ответ: $(5{,}0 \pm 0{,}1)$ с.

Задача 2.3.2.3. Методичка (20 баллов)

Условие

На лабораторной работе по физике в распоряжении школьников оказались резисторы двух номиналов: с сопротивлениями x и y кОм, а также конденсаторы двух номиналов: с емкостями x и y мкФ, при этом $x > y$. В старой методичке, посвященной этой лабораторной работе, была изображена схема, приведенная на рисунке, чернила на которой сильно затерлись. В результате Витя решил, что изображенные элементы являются резисторами, и, соединив их согласно схеме, получил элемент с эквивалентным сопротивлением $R = 5$ кОм. Таня же решила, что это конденсаторы и, соединив их, получила элемент с эквивалентной емкостью $C = 2$ мкФ. Определите, чему равнялось число y.

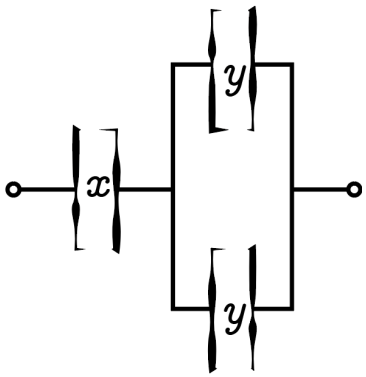

Рис. 2.3.4

Решение

При последовательном соединении резисторов их сопротивления складываются, а при параллельном — складываются обратные сопротивлениям величины. Поэтому сопротивление R схемы Вити через числа x и y в килоомах (кОм) может быть выражено по формуле

$$R = x + \frac{y}{2}. \tag{2.3.23}$$

Для конденсаторов правила поиска эквивалентной емкости при их последовательном и параллельном соединениях в точности обратные, поэтому емкость схемы Тани может быть выражена в микрофарадах (мкФ) по формуле

$$C = \frac{2xy}{x + 2y}. \tag{2.3.24}$$

Выразим x из уравнения (2.3.23) и подставим в (2.3.24):

$$C = \frac{2(R - y/2)y}{R + 3y/2}. \tag{2.3.25}$$

Домножив полученное уравнение на знаменатель дроби и раскрыв скобки, получим

$$RC + \frac{3Cy}{2} = 2Ry - y^2. \tag{2.3.26}$$

Поскольку величины x и y имеют разную размерность в разных частях задачи, имеет смысл сразу перейти к численным значениям

$$10 + 3y = 10y - y^2. \tag{2.3.27}$$

Это квадратное уравнение, которое легко привести к канонической форме

$$y^2 - 7y + 10 = 0 \tag{2.3.28}$$

и решить с помощью дискриминанта

$$y_{1,2} = \frac{7 \pm \sqrt{49 - 40}}{2} = \frac{7 \pm 3}{2}. \tag{2.3.29}$$

Снова воспользовавшись уравнением (2.3.23), легко определить, что при $y = 5$ (корень квадратного уравнения с плюсом) $x = 2{,}5$, что не удовлетворяет условию $x > y$. В то же время, при $y = 2$ (корень с минусом) $x = 4$, что удовлетворяет этому условию. Стало быть, верный корень — $y = 2$.

Погрешность $0{,}1$.

Ответ: $2{,}0 \pm 0{,}1$.

Задача 2.3.2.4. Морозилка (25 баллов)

Условие

Морозильная установка работает по циклу Карно, обходимому в обратном направлении. Какую работу должна потребить такая установка, чтобы заморозить $m = 0{,}4$ кг воды, взятой при ее температуре замерзания, передав полученную от нее теплоту в помещение, температура θ которого равна 30 °C? Удельная теплота плавления и кристаллизации воды $\lambda = 333$ кДж/кг, абсолютный ноль температур $T_0 = -273$ °C.

Решение

Идеальный тепловой двигатель (машина Карно) работает, как известно, по циклу, состоящему из двух изотерм и двух адиабат, и имеет КПД

$$\eta = \frac{T_2 - T_1}{T_2}, \tag{2.3.30}$$

где T_1 — минимальная, а T_2 — максимальная температуры в цикле. По определению КПД тепловой машины он равен отношению работы A, совершаемой газом за цикл, к теплоте Q_2, получаемой им от нагревателя

$$\frac{T_2 - T_1}{T_2} = \eta = \frac{A}{Q_2}. \tag{2.3.31}$$

Отсюда Q_2 может быть выражено как

$$Q_2 = \frac{AT_2}{T_2 - T_1}. \tag{2.3.32}$$

Поскольку за полный цикл внутренняя энергия не изменяется, количество теплоты Q_1, которую газ отдает холодильнику такой машины, равна разнице

$$Q_1 = Q_2 - A = \frac{AT_1}{T_2 - T_1}. \tag{2.3.33}$$

В рассматриваемой задаче тепловой двигатель заменен холодильной машиной, для чего его цикл необходимо обходить в обратном направлении. При этом тепловой резервуар с температурой T_2 начинает получать тепло, а с температурой T_1 — отдавать, но по модулю количества теплоты, которыми газ обменивается с этими тепловыми резервуарами, не изменяются. Остается заметить, что в описанной в условиях холодильной машине $T_1 = 0\,^\circ\mathrm{C} = 273\,\mathrm{K}$, $T_2 = \theta$, $Q_1 = \lambda m$, поскольку забираемая у теплового резервуара с меньшей температурой теплота идет на замораживание в нем воды. Подставив эти данные в (2.3.33), получим

$$\lambda m = \frac{AT_1}{\theta - T_1}, \tag{2.3.34}$$

откуда окончательно выразим ответ

$$A = \frac{\lambda m(\theta - T_1)}{T_1} \approx 14{,}6\,\text{Дж}. \tag{2.3.35}$$

Погрешность: $0{,}5$ Дж.

Ответ: $(14{,}6 \pm 0{,}5)$ Дж.

Задача 2.3.2.5. Электромагнит (30 баллов)

Условие

Для большого промышленного электромагнита критически важной стала проблема охлаждения. Было установлено, что при пропускании через электромагнит

тока $I_1 = 10$ А он нагревается до температуры $t_1 = 70\,^\circ$C, после чего перестает увеличивать свою температуру, а при пропускании через него тока $I_2 = 20$ А — до температуры $t_2 = 205\,^\circ$C.

Определите температуру θ в помещении цеха, в котором используется электромагнит, если известно, что основным механизмом, отвечающим за охлаждение магнита, выступает теплопроводность, мощность которой прямо пропорциональна разнице температур между телами, обменивающимися теплом.

Решение

Как указано в условиях, мощность теплопроводности прямо пропорциональна разнице температур между магнитом и окружающим его воздухом в помещении цеха. Обозначим коэффициент этой пропорциональности κ

$$\begin{cases} P_1 = \kappa(\theta - t_1), \\ P_2 = \kappa(\theta - t_2). \end{cases} \tag{2.3.36}$$

Повышение температуры останавливается, когда мощность производимого катушкой тепла и мощность тепла, уходящего от катушки, благодаря теплообмену, оказываются равны. Первую можно выразить из закона Джоуля – Ленца

$$\begin{cases} P_1 = I_1^2 R, \\ P_2 = I_2^2 R, \end{cases} \tag{2.3.37}$$

где R — сопротивление катушки.

Разделим друг на друга уравнения системы (2.3.36) и уравнения системы (2.3.37), а затем приравняем эти отношения:

$$\frac{P_1}{P_2} = \frac{\theta - t_1}{\theta - t_2} = \frac{I_1^2}{I_2^2}. \tag{2.3.38}$$

Тривиальными алгебраическими преобразованиями выразим θ

$$(\theta - t_1)I_2^2 = (\theta - t_2)I_1^2 \Rightarrow \theta = \frac{t_1 I_2^2 - t_2 I_1^2}{I_2^2 - I_1^2} = 25\,^\circ\text{C}. \tag{2.3.39}$$

Погрешность: $0{,}1\,^\circ$C.

Ответ: $(25{,}0 \pm 0{,}1)$.

2.3.3. Вторая волна. Задачи 8–9 класса

Задачи второй волны предметного тура по физике за 8–9 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63464/enter/.

Задача 2.3.3.1. Аккумулятор тепла (10 баллов)

Условие

Для печи отопления требуется разработать аккумулятор тепла, представляющий собой емкость фиксированного объема, заполненную тем или иным минералом. Используя таблицу 2.3.1 плотностей ρ и удельных теплоемкостей $c_{\text{уд}}$ различных подходящих для этого пород, расположите их в порядке увеличения количества теплоты, которое может быть запасено в таком аккумуляторе при его нагреве до одной и той же температуры θ. Считайте, что θ заведомо меньше температур, при которых любой из этих минералов начнет плавиться или химически разрушаться, а тепловое расширение этих минералов при нагреве до θ пренебрежимо мало.

Таблица 2.3.1. Плотности и удельные теплоемкости

	Минерал	ρ, г/см^3	$c_{\text{уд}}$, кДж/(кг · °C)
A	Кварц	2,6	0,75
B	Базальт	2,8	0,85
C	Талькохлорит	2,75	0,98
D	Нефрит	3	1,1
E	Порфирит	1,45	0,83

Введите в поле ответа последовательность букв, соответствующих выбранным минералам, без пробелов, от наименьшего к наибольшему количеству запасаемой теплоты.

Решение

Количество тепла Q, которое может быть запасено в тепловом аккумуляторе фиксированного объема V, удобно выразить через массу m материала этого аккумулятора

$$Q = c_{\text{уд}} m(\theta - t_0), \tag{2.3.40}$$

где t_0 — начальная температура теплоаккумулятора. В свою очередь, масса m элементарно выражается через плотность вещества и объем V

$$m = \rho V, \tag{2.3.41}$$

откуда

$$Q = c_{\text{уд}} \rho V(\theta - t_0). \tag{2.3.42}$$

Поскольку величины V, θ, t_0 независимы от выбранного вещества, задача сводится к расположению в порядке возрастания произведений $c_{\text{уд}}\rho$. Найдем эти произведения для всей таблицы 2.3.1.

Таблица 2.3.2

	Минерал	ρ, г/см^3	$c_{\text{уд}}$, кДж/(кг · °C)	$c_{\text{уд}}\rho$, кДж/(м^3 · °C)
A	Кварц	2,6	0,75	1 950
B	Базальт	2,8	0,85	2 380
C	Талькохлорит	2,75	0,98	2 695
D	Нефрит	3	1,1	3 300
E	Порфирит	1,45	0,83	1 204

Ответ: EABCD.

Задача 2.3.3.2. Изображения (15 баллов)

Условие

Тонкая собирающая линза имеет фокусное расстояние $F = 20$ см. Вдоль ее оптической оси перед линзой расположено плоское зеркало, на расстоянии $h = 2$ см от которого и $d = 60$ см от линзы размещен светодиод S (рис. 2.3.5). Найдите расстояние между двумя действительными изображениями светодиода, формируемыми этой оптической системой.

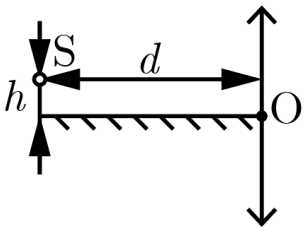

Рис. 2.3.5

Решение

Зеркало формирует мнимое изображение S_1 источника, расположенное в противоположном от него полупространстве на таком же расстоянии от зеркала, как и сам источник. В силу перпендикулярности зеркала и линзы, это мнимое изображение также окажется на расстоянии d от плоскости линзы. Далее линза формирует два действительных изображения: одно непосредственно от источника S (на рис. 2.3.6 оно обозначено S_2) и другое — от его мнимого изображения S_1 (S_3 на рисунке).

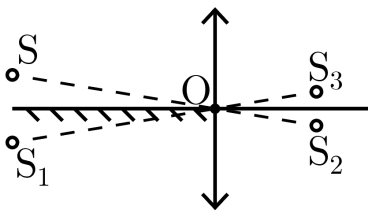

Рис. 2.3.6

Расстояние f, на котором находятся оба действительных изображения от плоскости линзы, легко найти по формуле тонкой линзы

$$\frac{1}{f}+\frac{1}{d}=\frac{1}{F}\Rightarrow f=\frac{Fd}{d-F}. \tag{2.3.43}$$

Искомое расстояние l между изображениями S_2 и S_3, благодаря подобию треугольников $\triangle \mathrm{OSS_1}$ и $\triangle \mathrm{OS_2S_3}$, относится к расстоянию $2h$ между источником и его мнимым изображением S_1 так же, как расстояния от соответствующих изображений и источников до плоскости линзы, являющиеся высотами указанных треугольников

$$\frac{l}{2h}=\frac{f}{d}=\frac{F}{d-F}. \tag{2.3.44}$$

Отсюда окончательно находим

$$l=2h\frac{F}{d-F}=2\,\text{см}. \tag{2.3.45}$$

Погрешность $0{,}1$ см.

Ответ: $l=(2{,}0\pm 0{,}1)$ см.

Задача 2.3.3.3. Пила (30 баллов)

Условие

Циркулярная пила представляет собой пильный диск диаметром $D=19$ см, вращающийся с частотой $4\,500$ об/мин. Определите среднюю силу сопротивления заготовки вращению полотна пилы, если за один пропил, длившийся $t=3$ с, выделилось $Q=5$ кДж тепла, а пила соприкасалась с заготовкой только узкой полоской своей внешней кромки.

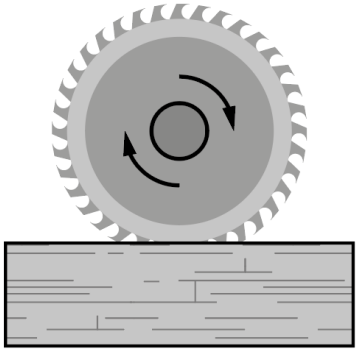

Рис. 2.3.7

Решение

При вращении пилы диссипативные силы (трения различных типов) переводят механическую энергию пильного диска в тепловую. При этом количество выделяющегося тепла равно по модулю работе A этих сил. Последнюю легко найти из ее определения

$$A = FS = Fvt, \tag{2.3.46}$$

где S — путь точек соприкосновения диска с заготовкой, v — скорость этих точек. При вращении диска скорость точек его кромки удобно выразить через период вращения T этого диска

$$v = \frac{\pi D}{T} = \pi D\nu, \tag{2.3.47}$$

где ν — частота вращения диска в оборотах в секунду. Подставляя (2.3.47) в (2.3.46) и учитывая $Q = A$, получим окончательно

$$Q = \pi FD\nu t \Rightarrow F = \frac{Q}{\pi D\nu t} \approx 37\,\text{Н}. \tag{2.3.48}$$

Погрешность 1 Н.

Ответ: $F = (37 \pm 1)\,\text{Н}$.

Задача 2.3.3.4. Питстоп (25 баллов)

Условие

Транспортный робот перемещается из города A в город B, двигаясь практически все время с некоторой постоянной скоростью v. Однако один раз за маршрут ему необходима остановка для заправки и краткого технического обслуживания. Инженеры установили, что при уменьшении длительности этой остановки вдвое скорость движения робота на остальной части маршрута можно будет снизить на $\delta = 10\%$, сохранив при этом его среднюю путевую скорость, что поможет повысить безопасность и экономичность движения. Определите, на сколько процентов (от исходного значения) удалось бы снизить скорость v без изменения средней путевой, если бы от технической остановки удалось полностью отказаться?

Решение

Средняя путевая скорость определяется как отношение всего пути ко всему времени, которое этот путь занимает. Поскольку расстояние между городами неизменно, сохранение средней путевой скорости означает и сохранение общего времени в пути (включая остановку). Следовательно, уменьшение длительности остановки на Δt эквивалентно увеличению времени непосредственного движения на ту же величину. Обозначим общее время робота в пути t, исходную длительность его остановки τ, а исходную скорость движения v_0. Тогда путь робота может быть выражен до и после снижения времени остановки как

$$S = v_0(t - \tau) = v_0(1 - \delta)\left(t - \frac{\tau}{2}\right). \tag{2.3.49}$$

Сократив v_0 и перегруппировав слагаемые, получим

$$\delta t = \tau\left(1 - \frac{1}{2} + \frac{\delta}{2}\right) = \tau\frac{\delta+1}{2}. \tag{2.3.50}$$

Полностью избавившись от остановки, таким образом, робот будет двигаться в течение времени

$$t = \tau\frac{\delta+1}{2\delta}. \tag{2.3.51}$$

Аналогично, время движения $t-\tau$ при наличии остановки удобно записать как

$$t - \tau = \tau\left(\frac{\delta+1}{2\delta} - 1\right) = \tau\frac{1-\delta}{2\delta}. \tag{2.3.52}$$

Обозначив v_1 скорость, которой можно добиться, исключив остановку, запишем через эти выражения путь и приравняем его в случаях с остановкой и без

$$v_1 t = v_0(t-\tau) \Rightarrow v_1\tau\frac{\delta+1}{2\delta} = v_0\tau\frac{1-\delta}{2\delta}. \tag{2.3.53}$$

Отсюда окончательно

$$\frac{v_1}{v_0} = \frac{1-\delta}{1+\delta} \approx 0{,}818. \tag{2.3.54}$$

Итого скорость движения без остановки может составлять приблизительно $81{,}8\%$ от исходной скорости, то есть ниже ее на $18{,}2\%$.

Погрешность $0{,}5\%$.

Ответ: $(18{,}2 \pm 0{,}5)\%$.

Задача 2.3.3.5. Сцепка (30 баллов)

Условие

Два абсолютно одинаковых ползунковых реостата, сопротивления которых могут изменяться в пределах от 0 до $R_0 = 2\,\text{кОм}$, размещены параллельно на печатной плате и соединены как изображено на рис. 2.3.8. Из-за ошибки в процессе пайки изоляция их ползунков слиплась таким образом, что ползунки всегда занимают одно и то же положение на обоих реостатах, но электрический контакт между ними отсутствует (это соединение обозначено на рисунке пунктиром). Найдите разницу между максимальным и минимальным сопротивлениями полученной батареи.

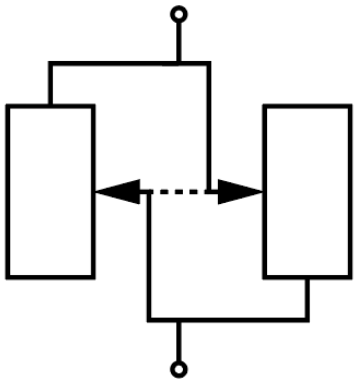

Рис. 2.3.8

Решение

Реостаты на схеме соединены параллельно, поэтому общее сопротивление схемы R может быть выражено через сопротивления $R_{1,2}$ каждого из реостатов по формуле

$$R = \frac{R_1 R_2}{R_1 + R_2}. \tag{2.3.55}$$

Несложно видеть из схемы, что когда ползунок находится в крайнем верхнем положении, левый реостат имеет нулевое сопротивление, а правый — сопротивление R_0 и наоборот. Величина сопротивления находится в линейной зависимости от длины провода. Из этого можно заключить, что при любом положении ползунка

$$R_2 = R_0 - R_1. \tag{2.3.56}$$

Подставляя этот результат в (2.3.55), получим

$$R = \frac{R_1(R_0 - R_1)}{R_0} = R_1 - \frac{R_1^2}{R_0}. \tag{2.3.57}$$

График полученной функции является параболой. Его минимумы и максимумы могут лежать либо на границах диапазона изменения R_1, либо в вершине соответствующей параболы. Поскольку коэффициент перед квадратным слагаемым отрицательный, парабола «повернута» ветвями вниз, то есть на границах диапазона (при $R_1 = 0$ или $R_1 = R_0$) сопротивления батареи минимальны и равны 0, а в ее вершине (которая, как легко видеть из симметрии или непосредственно по формуле $x_{max} = -b/(2a)$, лежит в центре диапазона, при $R_1 = R_2 = \dfrac{R_0}{2}$) сопротивление батареи равно $\dfrac{R_0}{4}$.

Таким образом,

$$R_{max} - R_{min} = \frac{R_0}{4} - 0 = \frac{R_0}{4} = 0{,}5\,\text{кОм}. \tag{2.3.58}$$

Погрешность 0,01 кОм.

Ответ: $l = (0{,}50 \pm 0{,}01)$ кОм.

2.3.4. Вторая волна. Задачи 10–11 класса

Задачи второй волны предметного тура по по физике за 10–11 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63481/enter/.

Задача 2.3.4.1. Зарядка (10 баллов)

Условие

Для увеличения ресурса аккумулятора его зарядка происходит по специальной программе, учитывающей внешние условия, интенсивность использования прибора и другие факторы. Рассчитав оптимальный режим, зарядное устройство в течении $\tau = 10$ мин подавало на аккумулятор ток, линейно возраставший со временем от нуля до максимального значения $I_0 = 3$ А, затем в течение $2{,}5\tau$ поддерживало постоянное значение этого тока и, наконец, на протяжении $\tau/2$, также линейно опускало ток от максимального значения до нуля. Какой общий заряд поступил на положительную клемму аккумулятора за все это время?

Решение

Один из способов решения задачи состоит в обнаружении аналогии между током и движением. Подобно тому, как скорость описывает темп изменения координаты, сила тока описывает темп поступления заряда на аккумулятор. Из кинематики известно, что при равномерном увеличении этого темпа (равноускоренном движении) от нуля, либо при равномерном снижении этого темпа (равнозамедленном движении) до нуля тело проходит вдвое меньшее расстояние, чем при движении с постоянной скоростью, равной максимальной на рассматриваемом участке. Применяя этот результат к току, заметим, что за время $2{,}5\tau$ постоянного тока зарядки на аккумулятор поступил заряд

$$q_1 = 2{,}5\tau I_0, \tag{2.3.59}$$

а за общее время $1{,}5\tau$ увеличения и уменьшения силы тока — заряд

$$q_2 = \frac{1{,}5\tau I_0}{2} = 0{,}75\tau I_0. \tag{2.3.60}$$

Складывая эти заряды, получим окончательно

$$q = 3{,}25\tau I_0 = 5850\,\text{Кл}. \tag{2.3.61}$$

Задача также может быть решена графически, изображением зависимости $I(t)$ и вычислением площади под ней. Фактически такое решение также является применением аналогии, но геометрической, а не кинематической.

Погрешность 50 Кл.

Ответ: (5850 ± 50) Кл.

Задача 2.3.4.2. Принтер (15 баллов)

Условие

Печатающая головка 3D-принтера может перемещаться вдоль направляющей (координата x), которая также может смещаться в перпендикулярном направлении (координата y) под действием двух сервоприводов. Для изготовления детали на принтер была передана программа, согласно которой сервоприводы должны перемещать головку по законам $x(t) = 0{,}2\sin(t/10)$; $y(t) = 0{,}1 + 0{,}2\cos(t/10)$, где t — время, а все величины даны в основных единицах СИ. Найдите величину ускорения печатающей головки при выполнении этой программы.

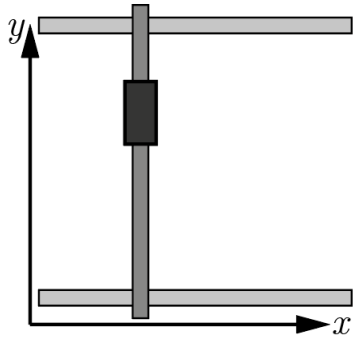

Рис. 2.3.9

Решение

Прежде всего заметим, что уравнения приведенного типа параметрически задают окружность. Это следует из самого определения синуса и косинуса. Радиус R этой окружности равен множителю перед синусом и косинусом (т. к. в математическом определении тригонометрических функций используется единичная окружность), то есть $R = 0{,}2$ м. Здесь было учтено, что основными единицами СИ для измерения длины являются метры.

Поскольку зависимость угла на окружности (аргумента синуса и косинуса) от времени линейна, модуль скорости v печатающей головки постоянен. Чтобы найти его, определим период обращения. Печатающая головка описывает полную окружность, когда аргумент синуса и косинуса меняется на 2π, что происходит при достижении t значения $T = 20\pi$ с. Тогда

$$v = \frac{2\pi R}{T}. \tag{2.3.62}$$

Окончательно заметим, что при неизменной по модулю скорости головки единственное ее ускорение — центростремительное, которое может быть найдено как

$$a = \frac{v^2}{R} = \frac{4\pi^2 R}{T^2} = \frac{4\cancel{\pi^2}\cdot 0{,}2}{400\cancel{\pi^2}} \approx 2\ \text{мм/с}^2. \tag{2.3.63}$$

Погрешность $0{,}1\ \text{мм/с}^2$.

Ответ: $(2{,}0 \pm 0{,}1)\ \text{мм/с}^2$.

Задача 2.3.4.3. Плита (20 баллов)

Условие

Квадратная плита ABCD со стороной $a = 40\,\text{см}$ шарнирно закреплена в одной точке и вращается вокруг оси, перпендикулярной ее плоскости с постоянной угловой скоростью. При этом в некоторый момент времени скорость вершины C этой плиты направлена строго на вершину D, а ускорение вершины A — строго на вершину B (см. рис. 2.3.10). На каком расстоянии от центра пластины находится шарнир?

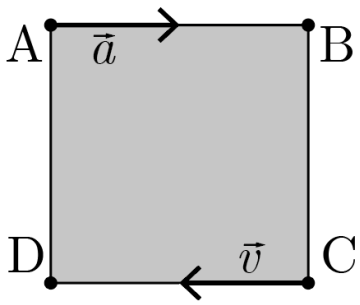

Рис. 2.3.10

Решение

При плоском вращении скорость каждой точки тела направлена перпендикулярно направлению из этой точки на ось вращения, а ускорение (центростремительное) — непосредственно на эту ось. Поэтому, проведя одну прямую через точку C перпендикулярно ее скорости, а другую — через точку A вдоль ее ускорения, можно найти ось вращения как точку пересечения этих прямых. Такой точкой будет, разумеется, вершина B, а расстояние от нее до центра пластины равно

$$l = \frac{\sqrt{2}}{2}a \approx 28{,}3\,\text{см}. \tag{2.3.64}$$

Погрешность $0{,}5\,\text{см}$.

Ответ: $(28{,}3 \pm 0{,}5)\,\text{см}$.

Задача 2.3.4.4. Прямоугольники (25 баллов)

Условие

К проекту модельного теплового двигателя, рабочим телом которого является идеальный одноатомный газ, прилагается pV-диаграмма его рабочего цикла, представляющая собой прямоугольник 1234. К сожалению, автор не указал ни давления, ни объемы характерных точек, а ограничился «площадями» (в энергетических единицах) некоторых прямоугольников на данной диаграмме, которые указаны на рис. 2.3.10. Да к тому же самая важная площадь — площадь внутри цикла 1234, стерлась. Тем не менее определите КПД данного двигателя.

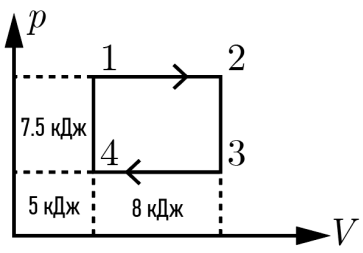

Рис. 2.3.11

Решение

КПД η теплового двигателя определяется как отношение работы A газа за один цикл этого двигателя к количеству теплоты Q, получаемой газом за этот цикл от нагревателя:

$$\eta = \frac{A}{Q}. \tag{2.3.65}$$

Первая — есть площадь прямоугольника 1234. Найти ее просто, если заметить, что поскольку высота (вдоль оси p) у верхних двух прямоугольников одинакова, их площади относятся так же, как и их ширины (вдоль оси V), и то же верно для нижней пары прямоугольников

$$A = 7{,}5\frac{8}{5} = 12\,\text{кДж}. \tag{2.3.66}$$

Чтобы найти теплоту Q, воспользуемся первым началом термодинамики $Q = A + \Delta U$ и заметим, что газ получает теплоту на процессах 41 и 12. Работа за эти два процесса равна полной площади под отрезком 12

$$A_{412} = 12 + 8 = 20\,\text{кДж}, \tag{2.3.67}$$

а внутренняя энергия может быть удобно вычислена по формуле

$$U = \frac{3}{2}PV, \tag{2.3.68}$$

из которой следует, что внутренняя энергия U_4 газа в состоянии 4 равна

$$U_4 = \frac{3}{2}5 = 7{,}5\,\text{кДж}, \tag{2.3.69}$$

поскольку произведение PV для этого состояния есть площадь одного соответствующего прямоугольника. Аналогично, внутренняя энергия U_2 газа в состоянии 2 равна

$$U_2 = \frac{3}{2}(7{,}5 + 12 + 5 + 8) = 48{,}75\,\text{кДж}, \tag{2.3.70}$$

поскольку в этом состоянии соответствующее произведение PV равно общей площади всех прямоугольников на диаграмме.

Подставляя все найденные величины в исходное уравнение (2.3.65), получим окончательно

$$\eta = \frac{A}{A_{412} + U_2 - U_4} = \frac{12}{20 + 48{,}75 - 7{,}5} \approx 19{,}6\%. \tag{2.3.71}$$

Погрешность 1%.

Ответ: $(19{,}6 \pm 1{,}0)\%$

Задача 2.3.4.5. Трюм (30 баллов)

Условие

Трюм грузового судна представляет собой призму с основанием в виде равностороннего треугольника вершиной вниз. Длина внешней стороны треугольника $a = 15\,\text{м}$, толщина его стенок $d = 1{,}5\,\text{м}$, длина киля судна (высота призмы) $l = 40\,\text{м}$. Инженерами было рассчитано, что для сохранения устойчивости судна центр тяжести сыпучего груза, перевозимого в трюме, должен быть хотя бы на $h = 2\,\text{м}$ ниже центра тяжести вытесняемой трюмом воды при его полном погружении. Какой максимальный объем груза можно разместить в трюме, если при насыпании его центр тяжести занимает самое низкое доступное положение?

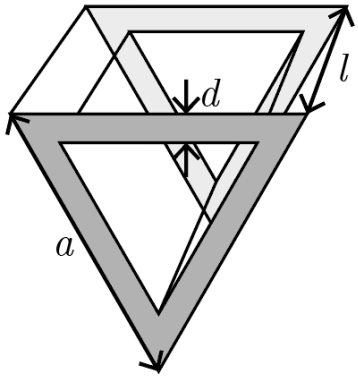

Рис. 2.3.12

Решение

Сыпучий груз занимает объем, форма которого также является призмой с основанием в виде равностороннего треугольника, во всяком случае, при необходимости понизить центр тяжести. В силу симметрии, центр тяжести равностороннего треугольника находится в его геометрическом центре. Поскольку центр тяжести груза должен быть на h ниже, чем центр тяжести вытесненной воды, центр треугольника, формируемого сечением насыпанного груза (темный на рис. 2.3.13), на h ниже центра трюма.

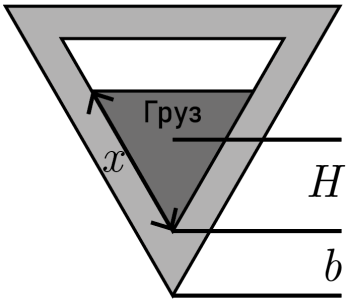

Рис. 2.3.13

Отсчитывая от киля (нижней вершины треугольника) высоту $y_{\text{в}}$, на которой

находится центр тяжести вытесненной воды, легко выразить как

$$y_{\text{в}} = \frac{\sqrt{3}}{3}a, \tag{2.3.72}$$

поскольку радиус описанной окружности для равностороннего треугольника в $\sqrt{3}/3$ раз меньше его стороны.

Высота $y_{\text{г}}$ центра тяжести груза может быть найдена как

$$y_{\text{г}} = b + H, \tag{2.3.73}$$

где b — толщина стенки вдоль соответствующего направления (см. рис. 2.3.13), а H — высота центра тяжести груза от нижней внутренней точки трюма. Обе эти величины вычисляются из тригонометрии

$$b = 2d; \quad H = \frac{\sqrt{3}}{3}x, \tag{2.3.74}$$

где x — сторона треугольника, являющегося поперечным сечением груза.

Учитывая $y_{\text{г}} + h = y_{\text{в}}$, получим

$$\frac{\sqrt{3}}{3}x + 2d + h = \frac{\sqrt{3}}{3}a, \tag{2.3.75}$$

откуда

$$x = a - \sqrt{3}(2d + h) \approx 6{,}34\,\text{м}. \tag{2.3.76}$$

Объем, занимаемый грузом при такой длине его стороны, находится как произведение площади равностороннего треугольника на длину киля

$$V = \frac{\sqrt{3}}{4}x^2 l \approx 696\,\text{м}^3. \tag{2.3.77}$$

Погрешность 10 м^3.

Ответ: (696 ± 10) м^3.

2.3.5. Третья волна. Задачи 8–9 класса

Задачи третьей волны предметного тура по физике за 8–9 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63465/enter/.

Задача 2.3.5.1. Башня (10 баллов)

Условие

В гидравлической системе используется башня, заполненная минеральным маслом с плотностью $\rho = 900\,\text{кг/м}^3$. Верхний уровень масла расположен на $h = 60\,\text{м}$ выше, чем смотровое окно в трубе с маслом, закрепленное на ней при помощи $n = 16$ одинаковых болтов. Определите, какую нагрузку должен выдерживать каждый из этих болтов на разрыв, чтобы обеспечить трехкратный запас прочности? Площадь смотрового окна $S = 0{,}1\,\text{м}^2$. Ускорение свободного падения $g = 9{,}8\,\text{м/с}^2$.

Решение

Давление p масла на уровне окна элементарно вычисляется по формуле гидростатического давления

$$p = \rho g h. \tag{2.3.78}$$

Исходя из определения всякого давления p как отношения силы F к площади S, на которую действует эта сила, найдем силу со стороны жидкости, «пытающуюся выдавить» смотровое окно

$$F = pS = \rho g h S. \tag{2.3.79}$$

Искомая рассчетная нагрузка f каждого из болтов может быть получена домножением этой силы на 3 (требуемый запас прочности) и делением на число болтов n, по которым распределяется нагрузка

$$f = \frac{3F}{16} = \frac{3\rho g h S}{16} \approx 9{,}9\,\text{кН}. \tag{2.3.80}$$

Погрешность $0{,}1$ кН.

Ответ: $(9{,}9 \pm 0{,}1)$ кН.

Задача 2.3.5.2. Реостат (15 баллов)

Условие

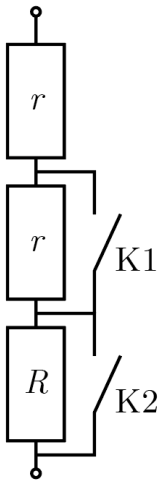

Рис. 2.3.14

Имея в своем распоряжении резисторы только двух различных номиналов, начинающий радиолюбитель изготовил ступенчатый реостат оригинальной конструкции, позволяющий, переключая ключи, получить четыре различных значения сопротивления. Увы, на приложенной к прибору схеме он забыл указать сопротивления отдельных резисторов, указав только, какие из них совпадают. В техническом паспорте устройства остались данные о том, что при замыкании ключа K_1 и размыкании ключа K_2 оно имеет сопротивление $R_1 = 2$ кОм, а напротив, при замыкании ключа K_2 и размыкании ключа K_1 — сопротивление $R_2 = 3$ кОм. Найдите максимальное сопротивление, которое можно получить, используя этот реостат.

Решение

Когда параллельно резистору коротко замыкается цепь, этот резистор фактически перестает работать, поскольку сопротивление провода пренебрежимо мало. Следовательно, замыкание ключа K1 фактически эквивалентно замене среднего резистора на отрезок провода, а ключа K2 — такой же замене нижнего. Учитывая это и применяя формулу эквивалентного сопротивления последовательно соединенных резисторов, легко получим

$$\begin{cases} R_1 = r + R, \\ R_2 = 2r. \end{cases} \tag{2.3.81}$$

Решая эту систему, находим $r = \frac{R_2}{2}$ и $R = R_1 - \frac{R_2}{2}$. Теперь точно так же составим выражения для оставшихся двух конфигураций реостата: R_3 с обоими замкнутыми ключами и R_4 с обоими разомкнутыми

$$\begin{cases} R_3 = r = \dfrac{R_2}{2} = 1{,}5\,\text{кОм}, \\ R_4 = 2r + R = R_1 + \dfrac{R_2}{2} = 3{,}5\,\text{кОм}. \end{cases} \tag{2.3.82}$$

Погрешность $0{,}01$ кОм.

Ответ: $(3{,}50 \pm 0{,}1)$ кОм.

Задача 2.3.5.3. Теплоноситель (20 баллов)

Условие

В некоторых типах ядерных реакторов в качестве теплоносителя используются жидкие металлы. Определите, сколько теплоты за 1 с забирает у реактора жидкий свинец с удельной теплоемкостью $c = 155$ Дж/(кг·°C) и средней плотностью $\rho = 10^4$ кг/м^3, если, двигаясь в трубе диаметром $d = 10$ см со скоростью $v = 20$ м/с, он нагревается от $t_1 = 400$ °C до $t_2 = 900$ °C.

Решение

Обозначим рассматриваемый промежуток времени (1 с) τ. Двигаясь со скоростью v, свинец успевает за это время пройти в трубе дистанцию $l = v\tau$. Учитывая площадь сечения трубы $S = \pi d^2/4$, можно заключить из этого, что за время τ в реактор поступает и из реактора уходит объем

$$V = Sl = \frac{\pi}{4} d^2 v\tau \tag{2.3.83}$$

расплавленного свинца. Количество теплоты Q, которое этот свинец забирает у реактора, задается выражением

$$Q = cm(t_2 - t_1) = c\rho V(t_2 - t_1) = \frac{\pi}{4} c\rho d^2 v\tau (t_2 - t_1) \approx 122\,\text{МДж}, \tag{2.3.84}$$

где m — масса поступившей и ушедшей порции свинца.

Погрешность 2 МДж.

Ответ: (122 ± 2) МДж.

Задача 2.3.5.4. Шкала (25 баллов)

Условие

Шкала вольтметра, используемого в эксперименте, имеет вид, представленный на рис. 2.3.15, и общую длину $l = 15$ см (от минимальной до максимальной отметки). Экспериментатор, глядя на прибор под углом $45°$ к плоскости шкалы, считал показания вольтметра как $U_1 = 1{,}2$ В ровно, однако на самом деле стрелка прибора находилась напротив отметки $U_2 = 0{,}8$ В. Определите, на какое расстояние отстоит стрелка от шкалы, если известно, что глаза экспериментатора находились со шкалой строго на одном уровне высоты, а деления расположены на шкале равномерно.

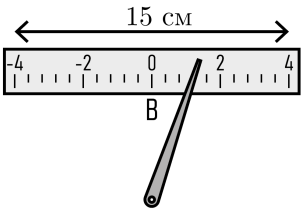

Рис. 2.3.15

Решение

Согласно условиям, глаза экспериментатора находятся на одной высоте со шкалой, поэтому удобно изобразить систему в горизонтальной плоскости (вид сверху), см. рис. 2.3.16. Поскольку угол α, под которым наблюдатель смотрит на стрелку, равен $45°$, искомое расстояние x в точности равно расстоянию y между точкой A действительных показаний прибора и точкой B считанных экспериментатором показаний (треугольник ABC, где C — стрелка — прямоугольный и равнобедренный).

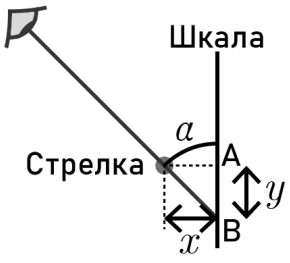

Рис. 2.3.16

Тогда остается вычислить расстояние y между отметками шкалы, соответствующими значениям U_1 и U_2. Поскольку шкала равномерная, это расстояние относится к полной длине шкалы так же, как величина абсолютной ошибки к ее разнице между ее верхним U_{max} и нижним U_{min} пределами измерений

$$\frac{y}{l} = \frac{U_2 - U_1}{U_{max} - U_{min}}. \tag{2.3.85}$$

Отсюда окончательно

$$x = y = l\frac{U_2 - U_1}{U_{max} - U_{min}} = 7{,}5 \text{ мм}. \tag{2.3.86}$$

Погрешность 0,1 мм.

Ответ: $(7{,}5 \pm 0{,}1)$ мм.

Задача 2.3.5.5. Площадка (30 баллов)

Условие

Три робота одновременно стартуют в углу A прямоугольной площадки ABCD. Все они движутся с постоянными по модулю скоростями и все заканчивают движение в точке D одновременно. Но первый робот движется по прямой вдоль стороны AD, второй — по трехзвенной ломаной ABCD, а третий — по двузвенной: сначала вдоль диагонали AC, а затем — по стороне CD. Во сколько раз средняя путевая скорость третьего робота выше, чем первого, если средняя путевая скорость второго робота выше, чем первого, в 2,5 раза? Временем на разгон, остановку и развороты роботов можно пренебречь.

Решение

Обозначив длину стороны AB (и, соответственно, CD) прямоугольника a, а длину стороны BC (и, соответственно, DA) — b, можно легко выразить через эти стороны пути $S_{1,2,3}$ всех трех роботов

$$\begin{cases} S_1 = b, \\ S_2 = 2a + b, \\ S_3 = \sqrt{a^2 + b^2} + a. \end{cases} \tag{2.3.87}$$

Поскольку время движения всех роботов совпадало, отношения их путей точно такие же, как и средних путевых скоростей:

$$\frac{v_2}{v_1} = \frac{S_2}{S_1} = \frac{2a}{b} + 1 = 2{,}5, \tag{2.3.88}$$

откуда легко найти

$$\frac{2a}{b} = 1{,}5 \Rightarrow b = \frac{4}{3}a. \tag{2.3.89}$$

Теперь, пользуясь той же логикой, найдем ответ на вопрос задачи как отношение путей третьего и первого роботов

$$\frac{v_3}{v_1} = \frac{S_3}{S_1} = \frac{\sqrt{a^2 + b^2} + a}{b} = \frac{\sqrt{\dfrac{25a^2}{9}} + a}{\dfrac{4a}{3}} = \frac{\dfrac{8a}{3}}{\dfrac{4a}{3}} = 2. \tag{2.3.90}$$

Погрешность 0,01.

Ответ: $2{,}00 \pm 0{,}01$.

2.3.6. Третья волна. Задачи 10–11 класса

Задачи третьей волны предметного тура по по физике за 10–11 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63482/enter/.

Задача 2.3.6.1. На коленке (10 баллов)

Условие

На конференции, посвященной освоению труднодоступных северных регионов, был представлен проект теплового двигателя, для работы которого используются два тепловых резервуара. Их можно собрать «на коленке»: в качестве холодильника выступает емкость с мокрым снегом, а в качестве нагревателя — котелок с кипящей водой. При этом, по заверениям авторов проекта, КПД двигателя только в $\alpha = 1{,}6$ раз уступает КПД идеальной тепловой машины с такими же холодильником и нагревателем. Найдите этот КПД. Абсолютный ноль температур $T_0 = -273\,^\circ\text{C}$. Двигатель используется при нормальном атмосферном давлении на уровне моря.

Решение

Мокрый снег представляет собой смесь льда и воды, поэтому может существовать только при температуре плавления льда, $t_\text{х} = 0\,^\circ\text{C}$. Аналогично, при атмосферном давлении температура кипящей воды может быть равна только $t_\text{н} = 100\,^\circ\text{C}$. КПД идеальной тепловой машины (машины Карно) с известными абсолютными термодинамическими температурами $T_\text{н}$ и $T_\text{х}$ холодильника задается выражением

$$\eta_0 = \frac{T_\text{н} - T_\text{х}}{T_\text{н}}. \tag{2.3.91}$$

Чтобы дать ответ на вопрос задачи, таким образом, остается разделить этот КПД на α и перевести температуры в шкалу Кельвина

$$\eta = \frac{\eta_0}{\alpha} = \frac{t_\text{н} - t_\text{х}}{\alpha(t_\text{н} - T_0)} \approx 16{,}8\%. \tag{2.3.92}$$

Погрешность: $0{,}2\%$.

Ответ: $(16{,}8 \pm 0{,}2)\%$.

Задача 2.3.6.2. Патруль (15 баллов)

Условие

Корабль береговой охраны движется с постоянной скоростью $v = 12\,\text{м/с}$ относительно поверхности воды. Наблюдательный дрон запрограммирован летать на постоянной высоте по траектории, в системе отсчета корабля представляющей собой окружность с радиусом $R = 2\,\text{км}$ и центром на этом корабле, двигаясь в этой

системе отсчета равномерно и совершая полный оборот за время $T = 10$ мин. Во сколько раз максимальная скорость дрона относительно поверхности воды выше его минимальной скорости относительно нее же?

Решение

Согласно правилу сложения скоростей, скорость $\vec{v}_{\text{дв}}$ дрона относительно воды равна (векторной) сумме его скорости $\vec{v}_{\text{дк}}$ относительно корабля и скорости $\vec{v}_{\text{кв}}$ корабля относительно воды. Поскольку направление вектора $\vec{v}_{\text{кв}}$ неизменно, а вектор $\vec{v}_{\text{дк}}$ в ходе движения дрона принимает все возможные в горизонтальной плоскости направления, обязательно найдутся такие моменты времени, когда эти два вектора сонаправлены и такие, когда они противонаправлены. Эти два случая и будут соответствовать максимальному и минимальному значениям модуля суммы этих векторов, равным $v_{max} = |\vec{v}_{\text{дк}}| + |\vec{v}_{\text{кв}}|$ и $v_{min} = ||\vec{v}_{\text{дк}}| - |\vec{v}_{\text{кв}}||$ соответственно.

Модуль вектора $\vec{v}_{\text{кв}}$ дан напрямую: он равен v. Модуль вектора $\vec{v}_{\text{дк}}$ легко получить, разделив путь дрона в СО корабля на период его обращения в этой СО

$$v_{\text{дк}} = \frac{2\pi R}{T}. \tag{2.3.93}$$

Отсюда получим окончательно

$$\frac{v_{max}}{v_{min}} = \frac{vT + 2\pi R}{|vT - 2\pi R|} \approx 3{,}68. \tag{2.3.94}$$

Погрешность 0,02.

Ответ: $3{,}68 \pm 0{,}02$.

Задача 2.3.6.3. Конденсатор (20 баллов)

Условие

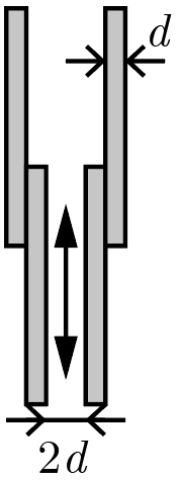

Рис. 2.3.17

Переменный конденсатор состоит из двух пар металлических пластинок толщиной d и площадью $S \gg d^2$, разделенных воздушными зазорами. При этом одна из пар (внутренняя) может частично или полностью входить в зазор другой (внешней), плотно прилегая к ней так, что электрический контакт между соответствующими пластинами никогда не нарушается, но при полном выдвижении площадь этого контакта пренебрежимо мала в сравнении с S.

Определите, во сколько раз максимальная емкость такого конденсатора превосходит минимальную, если зазор между пластинами внутренней пары имеет ширину $2d$.

Решение

Рассматриваемый конденсатор может быть представлен как батарея из двух параллельно соединенных конденсаторов, один из которых (внутренняя пара пластин) всегда имеет зазор $2d$ и площадь обкладок S, а другой (внешняя пара) имеет зазор $4d$ и площадь обкладок, которая может изменяться в пределах от 0 до S. Поскольку эти конденсаторы соединены параллельно, эквивалентная емкость батареи равна сумме их емкостей, а значит, максимальна, когда пары пластин максимально раздвинуты и минимальна, когда они полностью вдвинуты одна в другую. Используя формулу емкости плоского конденсатора, найдем, что емкость внутренней пары пластин (она же минимальная емкость всей батареи) равна

$$C_1 = \frac{S\varepsilon_0}{2d}, \tag{2.3.95}$$

где ε_0 — диэлектрическая постоянная.

Аналогично, емкость полностью выдвинутой внешней пары равна

$$C_2 = \frac{S\varepsilon_0}{4d}, \tag{2.3.96}$$

а максимальная емкость батареи составляет, соответственно, $C_1 + C_2$.

Тогда для искомого отношения максимальной и минимальной емкостей получим:

$$\frac{C_{max}}{C_{min}} = \frac{C_1 + C_2}{C_1} = \frac{\cancel{S\varepsilon_0}/(4\cancel{d}) + \cancel{S\varepsilon_0}/(2\cancel{d})}{\cancel{S\varepsilon_0}/(2\cancel{d})} = \frac{1/4 + 1/2}{1/2} = 1{,}5. \tag{2.3.97}$$

Погрешность 0,01.

Ответ: $1{,}50 \pm 0{,}01$.

Задача 2.3.6.4. Аэростат (25 баллов)

Условие

Горелка теплового аэростата способна поддерживать среднюю температуру воздуха в его оболочке не более, чем на $\Delta t = 70\,^\circ\text{C}$ выше, чем температура окружающего шар воздуха. Аэростат имеет объем $V = 645$ м^3, а общая масса его оболочки, корзины и полезной нагрузки $M = 150$ кг. Определите, при какой максимальной температуре окружающей среды аэростат сможет взлететь?

Абсолютный ноль температур $T_0 = -273\,^\circ\text{C}$, атмосферное давление $p_0 = 100\,\text{кПа}$, молярная масса воздуха $\mu = 29\,\text{г/моль}$, универсальная газовая постоянная $R = 8{,}31\,\text{Дж/(моль}\cdot\text{К)}$.

Решение

Оболочки монгольфьеров (тепловых аэростатов) представляют собой открытые сосуды, поэтому давление внутри и снаружи оболочки должно совпадать (и равняться p_0). Тогда из уравнения Менделеева – Клапейрона

$$p_0 V = \frac{m}{\mu} RT, \tag{2.3.98}$$

где T — абсолютная термодинамическая температура газа, m — его масса.

Легко выразить массу m_1 воздуха внутри оболочки и массу m_2 вытесненного атмосферного воздуха

$$\begin{aligned} m_1 &= \frac{\mu p_0 V}{R(T_0 + \Delta t)}, \\ m_2 &= \frac{\mu p_0 V}{R T_0}, \end{aligned} \tag{2.3.99}$$

где T_0 — искомая температура окружающего воздуха.

Для того чтобы аэростат мог подняться в воздух, необходимо, чтобы вес вытесняемого им воздуха превысил его общий вес (включая вес газа в оболочке)

$$m_2 \cancel{g} = M\cancel{g} + m_1\cancel{g} \Rightarrow \frac{\mu p_0 V}{R T_0} = M + \frac{\mu p_0 V}{R(T_0 + \Delta t)}, \tag{2.3.100}$$

где g — ускорение свободного падения (сразу сокращающееся во всех слагаемых).

Домножим это выражение на $RT_0(T_0 + \Delta T)$ и получим квадратное уравнение относительно T_0

$$\mu p_0 V(\cancel{T_0} + \Delta t) = MRT_0(T_0 + \Delta t) + \cancel{\mu p_0 V T_0}. \tag{2.3.101}$$

В канонической форме:

$$T_0^2 + T_0 \Delta t - \frac{\mu p_0 V \Delta t}{MR} = 0. \tag{2.3.102}$$

Остается найти (методом дискриминанта) единственный положительный корень этого уравнения

$$T_0 = -\frac{\Delta t}{2} + \frac{1}{2}\sqrt{\Delta t^2 + \frac{4\mu p_0 V}{MR}\Delta t} \approx 291\,\text{К} \approx 18\,^\circ\text{C}. \tag{2.3.103}$$

Погрешность $0{,}1\,^\circ\text{C}$.

Ответ: $(18{,}0 \pm 0{,}1)\,^\circ\text{C}$.

Задача 2.3.6.5. Спутник (30 баллов)

Условие

Спутник, движущийся вокруг Земли по высокой круговой орбите, перевели на другую круговую орбиту, в результате чего его кинетическая энергия увеличилась на 5%. На сколько процентов увеличился модуль потенциальной энергии взаимодействия спутника с планетой, если она считается равной нулю на бесконечном удалении от планеты?

Решение

Обозначим радиус орбиты спутника R, его орбитальную скорость v, его массу m, а массу планеты M. На спутник действует сила всемирного тяготения

$$F = G\frac{mM}{R^2}, \tag{2.3.104}$$

где G — гравитационная постоянная, связанная с центростремительным ускорением v^2/R спутника вторым законом Ньютона

$$\cancel{m}\frac{v^2}{\cancel{R}} = G\frac{\cancel{m}M}{R^{\cancel{2}}}. \tag{2.3.105}$$

Из этого выражения легко видеть, что квадрат орбитальной скорости спутника v^2 обратно пропорционален радиусу орбиты. Разумеется, кинетическая энергия спутника $mv^2/2$ прямо пропорциональна этому квадрату скорости и, следовательно, тоже обратно пропорциональна R.

Чтобы понять, как потенциальная энергия спутника зависит от радиуса его орбиты, проще всего обратить внимание на аналогию между гравитацией и электростатическими силами. Сила Кулона взаимодействия двух точечных зарядов зависит от расстояния между ними и обратно пропорциональна квадрату разделяющего их расстояния, точно так же, как сила всемирного тяготения. Одновременно потенциальная энергия взаимодействия этих зарядов обратно пропорциональна первой степени расстояния между ними, следовательно, то же справедливо и для гравитации. В результате видно, что модуль потенциальной энергии обратно пропорционален R, как и кинетическая энергия. Следовательно, при изменении орбиты спутника он изменится ровно во столько же раз.

Погрешность $0{,}01\%$.

Ответ: $(5{,}00 \pm 0{,}01)\%$.

2.3.7. Четвертая волна. Задачи 8–9 класса

Задачи четвертой волны предметного тура по физике за 8–9 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63466/enter/.

Задача 2.3.7.1. Расплав (10 баллов)

Условие

Расплавленная соль предлагается как эффективный аккумулятор тепла для некоторых типов теплоцентралей. Удельная теплота плавления и кристаллизации соли $\lambda = 28{,}7$ кДж/кг, ее теплоемкость в твердой форме $c_1 = 0{,}92$ кДж/(кг°C), а в жидкой — $c_2 = 1{,}5$ кДж/(кг°C), температура ее плавления $\theta = 800$ °C. Определите, какую массу соли необходимо взять, чтобы при ее нагреве от $t_0 = 20$ °C до $t_1 = 1200$ °C запасти $Q = 1$ МДж тепла.

Решение

Количество теплоты, требуемое на нагрев твердой соли до температуры плавления, задается выражением

$$Q_1 = c_1 m(\theta - t_0), \tag{2.3.106}$$

где m — масса нагреваемой соли.

Количество теплоты, уходящее непосредственно на плавление

$$Q_2 = \lambda m. \tag{2.3.107}$$

Наконец, количество теплоты, уходящее на нагрев расплава соли,

$$Q_3 = c_2 m(t_1 - \theta). \tag{2.3.108}$$

Складывая все эти порции тепла, получим, что общая запасаемая теплота

$$Q = Q_1 + Q_2 + Q_3 = m(c_1(\theta - t_0) + \lambda + c_2(t_1 - \theta)), \tag{2.3.109}$$

откуда окончательно

$$m = \frac{Q}{(c_1(\theta - t_0) + \lambda + c_2(t_1 - \theta))} \approx 743\,\text{г}. \tag{2.3.110}$$

Погрешность 1 г.

Ответ: (743 ± 1) г.

Задача 2.3.7.2. Катафот (15 баллов)

Условие

Катафот представляет собой два одинаковых квадратных зеркала, соединенных общей гранью под прямым углом друг к другу. При падении на него видимого света каждое зеркало поглощает $\eta = 20\%$ достигающей его световой энергии, а остальную — отражает. Параллельно биссектрисе образованного зеркалами угла в плоскости, перпендикулярной к их общему ребру, на середину одного из зеркал падает узкий лазерный луч, переносящий мощность $P = 5$ мВт. Какое количество световой энергии поглотит второе зеркало за $\tau = 10$ с?

Решение

Прежде всего отметим, что любой луч, падающий на катафот параллельно его биссектрисе, отразится последовательно от обоих зеркал катафота, как изображено на рис 2.3.18. При этом после первого отражения мощность луча снизится в $(1 - \eta)$ раз, и доля η от этой оставшейся мощности будет поглощена вторым зеркалом. В результате связь между изначальной P и поглощаемой P_1 мощностями имеет следующий вид:

$$P_1 = (1 - \eta)\eta P. \tag{2.3.111}$$

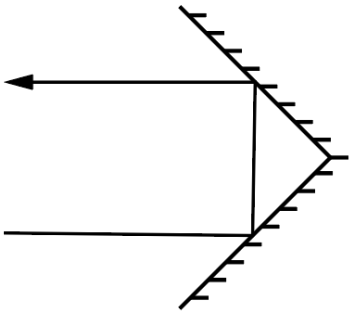

Рис. 2.3.18

Теперь остается лишь вспомнить определение мощности, как отношения энергии (в данном случае переносимой лазерным лучом или поглощаемой зеркалом) ко времени, чтобы получить окончательный ответ

$$Q = P_1\tau = (1 - \eta)\eta P\tau \approx 8\,\text{мДж}. \tag{2.3.112}$$

Погрешность $0{,}1$ мДж.

Ответ: $(8{,}0 \pm 0{,}1)$ мДж.

Задача 2.3.7.3. Соты (20 баллов)

Условие

Композитный материал изготавливают, вырезая из алюминия (плотность $\rho_1 = 2{,}7\,\text{г/см}^3$) строго периодическую вдоль двух взаимно перпендикулярных осей квадратную сетку с толщиной стенки d и длиной внутренней стороны ячейки $4d$, фрагмент которой изображен на рис. 2.3.19. Затем полости заполняют смолой, после затвердевания имеющей плотность $\rho_2 = 1{,}2\,\text{г/см}^3$. Найдите среднюю плотность большого листа из такого материала.

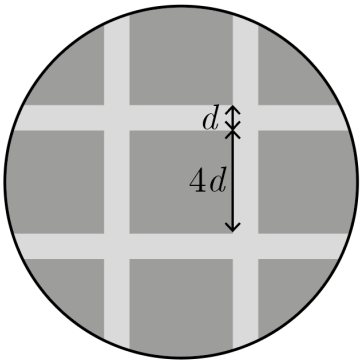

Рис. 2.3.19

Решение

По мере увеличения размеров листа материала роль его краев в общей плотности постепенно снижается, поэтому среднюю плотность большого листа следует вычислять как среднюю плотность одного элемента периодичности, границы которого изображены пунктиром на рис. 2.3.20. Объем V этого элемента равен $25dh$, где h — толщина листа материала.

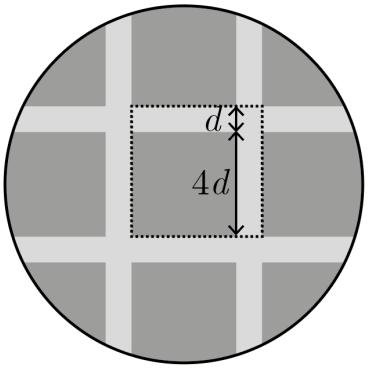

Рис. 2.3.20

Массу элемента найдем, сложив массы алюминия (индексы 1) и смолы (индексы 2)

$$m = m_1 + m_2 = \rho_1 V_1 + \rho_2 V_2 = \rho_1 9dh + \rho_2 16dh. \tag{2.3.113}$$

Тогда искомая плотность окончательно равна

$$\rho = \frac{m}{V} = \frac{\rho_1 9\cancel{dh} + \rho_2 16\cancel{dh}}{25\cancel{dh}} \approx 1{,}74\,\text{г/см}^3. \tag{2.3.114}$$

Погрешность $0{,}01\,\text{г/см}^3$.

Ответ: $(1{,}74 \pm 0{,}01)\,\text{г/см}^3$.

Задача 2.3.7.4. Домкрат (25 баллов)

Условие

На рис. 2.3.21 приведена схема устройства гидравлического домкрата. Его поршни представляют собой цилиндры с радиусами $R = 21\,\text{см}$ и $r = 3\,\text{см}$. Чтобы поднимать при помощи этого домкрата груз $m = 1{,}4\,\text{т}$, установленный на платформе большого цилиндра, к точке A рычага необходимо приложить силу не менее $F = 87{,}5\text{Н}$. Определите отношение $AB : BC$. Ускорение свободного падения $g = 9{,}8\,\text{м/с}^2$. На рисунке точный масштаб не сохранен.

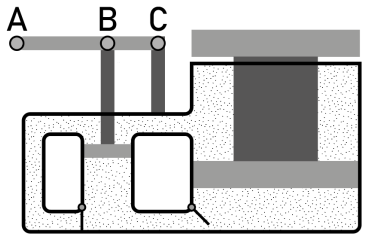

Рис. 2.3.21

Решение

Изображенный домкрат дает выигрыш в силе благодаря двум механизмам: рычагу и гидравлическому прессу. Выигрыш в силе, обеспечиваемый прессом, равен отношению площадей его цилиндров

$$\frac{mg}{F_1} = \frac{\pi R^2}{\pi r^2} \Rightarrow F_1 = mg\frac{r^2}{R^2}, \tag{2.3.115}$$

где F_1 — сила давления малого поршня.

В свою очередь, выигрыш в силе, обеспечиваемый рычагом, равен отношению его плеч, но так как рычаг закреплен в точке C, нужно сравнивать плечи AC и BC

$$\frac{F_1}{F} = \frac{AC}{BC} = \frac{AB + BC}{BC} = 1 + \frac{AB}{BC}. \tag{2.3.116}$$

Совместив эти уравнения, получим окончательно

$$\frac{AB}{BC} = \frac{F_1}{F} - 1 = \frac{mgr^2}{FR^2} - 1 = 2{,}2. \tag{2.3.117}$$

Погрешность $0{,}1$.

Ответ: $2{,}2 \pm 0{,}1$.

Задача 2.3.7.5. Номинальная мощность (30 баллов)

Условие

Изучая электронагреватель прямого действия, ученик заметил, что увеличение подаваемой на него силы тока на $\Delta I = 0{,}1\,\text{А}$ над номинальным значением приводит к увеличению тепловой мощности, выделяемой прибором, на $\Delta P_1 = 44\,\text{Вт}$,

а уменьшение силы тока на ту же величину от номинальной, приводит к уменьшению мощности на $\Delta P_2 = 36\,\text{Вт}$. Найдите номинальную мощность прибора, считая его сопротивление независимым от температуры.

Решение

Согласно закону Джоуля – Ленца, тепловая мощность электронагревателя равна

$$P = I^2 R, \tag{2.3.118}$$

где I — сила пропускаемого через него тока, а R — сопротивление прибора. Последнее по условиям задачи можно считать неизменным, поэтому, введя обозначения I_0 для номинальной силы тока и P_0 для номинальной мощности прибора, можно составить пропорции

$$\begin{cases} \dfrac{P_0 + \Delta P_1}{P_0} = \left(\dfrac{I_0 + \Delta I}{I_0}\right)^2, \\ \dfrac{P_0 - \Delta P_2}{P_0} = \left(\dfrac{I_0 - \Delta I}{I_0}\right)^2. \end{cases} \tag{2.3.119}$$

Обозначив $\frac{\Delta I}{I_0}$ буквой x и частично сократив дроби, приведем их к виду

$$\begin{cases} 1 + \dfrac{\Delta P_1}{P_0} = (1+x)^2, \\ 1 - \dfrac{\Delta P_2}{P_0} = (1-x)^2. \end{cases} \tag{2.3.120}$$

Эту систему можно решить, вычитая второе уравнение из первого

$$\frac{\Delta P_1 + \Delta P_2}{P_0} = 4x \Rightarrow x = \frac{\Delta P_1 + \Delta P_2}{4P_0} \tag{2.3.121}$$

и подставляя результат в любое уравнение системы (2.3.120)

$$\cancel{1} + \frac{\Delta P_1}{P_0} = \cancel{1} + \frac{\Delta P_1 + \Delta P_2}{2P_0} + \frac{(\Delta P_1 + \Delta P_2)^2}{16P_0^2}. \tag{2.3.122}$$

Домножим на $16P_0^2$

$$16\Delta P_1 P_0 = 8P_0(\Delta P_1 + \Delta P_2) + (\Delta P_1 + \Delta P_2)^2. \tag{2.3.123}$$

и выразим окончательно

$$P_0 = \frac{(\Delta P_1 + \Delta P_2)^2}{8(\Delta P_1 - \Delta P_2)} = 100\,\text{Вт}. \tag{2.3.124}$$

Погрешность 1 Вт.

Ответ: 100 ± 1 Вт.

2.3.8. Четвертая волна. Задачи 10–11 класса

Задачи четвертой волны предметного тура по по физике за 10–11 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63483/enter/.

Задача 2.3.8.1. Шестеренки (10 баллов)

Условие

В сложной трансмиссии две шестеренки А и Б вращаются в различных частях механизма так, что угловая скорость вращения шестеренки А в 3 раза выше, чем шестеренки Б, но линейная скорость зубцов шестеренки Б в 2 раза выше, чем шестеренки А. Найдите отношение центростремительного ускорения зубцов шестеренки А к центростремительному ускорению зубцов шестеренки Б.

Решение

Два хорошо известных выражения для центростремительного ускорения a

$$a = \frac{v^2}{R} = \omega^2 R, \tag{2.3.125}$$

где R — радиус траектории, v — линейная скорость, ω — угловая скорость. Перемножив эти выражения, получим

$$a^2 = \frac{v^2}{\cancel{R}}\omega^2\cancel{R} \Rightarrow a = v\omega. \tag{2.3.126}$$

Таким образом, центростремительное ускорение равно произведению линейной скорости на угловую, из чего следует

$$\frac{a_\text{А}}{a_\text{Б}} = \frac{v_\text{А}}{v_\text{Б}} \cdot \frac{\omega_\text{А}}{\omega_\text{Б}} = \frac{1}{2} \cdot \frac{3}{1} = 1{,}5. \tag{2.3.127}$$

Погрешность 0,01.

Ответ: $1{,}50 \pm 0{,}01$.

Задача 2.3.8.2. Маятник (15 баллов)

Условие

Заряженный металлический шарик закреплен на конце тонкой шелковой нити и несет заряд $q = 20$ мкКл. Другой конец нити закреплен к потолку. Определите массу шарика, если при помещении такого маятника в однородное электрическое поле, вектор напряженности которого направлен строго горизонтально и равен по модулю $E = 2{,}5$ кВ/м, сила натяжения нити после установления равновесия оказывается равна $T = 130$ мН. Ускорение свободного падения $g = 9{,}8$ м/с^2.

Решение

На шарик действуют три силы: горизонтально направленная сила электростатического взаимодействия $q\vec{E}$, вертикально вниз направленная сила тяжести $m\vec{g}$ и направленная вдоль нити сила ее натяжения $\vec{T}$. Разумеется, маятник может быть в равновесии, только если векторная сумма этих сил равна нулю, то есть эти три вектора образуют замкнутый треугольник

$$q\vec{E} + m\vec{g} + \vec{T} = \vec{0}. \tag{2.3.128}$$

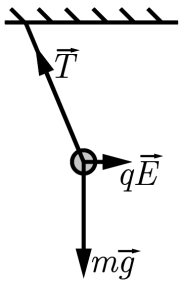

Рис. 2.3.22

Поскольку направления $\vec{E}$ и $\vec{g}$ известны, этот треугольник прямоугольный, а $\vec{T}$ — его гипотенуза. Тогда из теоремы Пифагора

$$q^2E^2 + m^2g^2 = T^2 \Rightarrow m = \frac{\sqrt{T^2 - q^2E^2}}{g} \approx 12{,}2\,\text{г}. \tag{2.3.129}$$

Погрешность $0{,}2$ г.

Ответ: $(12{,}2 \pm 0{,}2)$ г.

Задача 2.3.8.3. Автопилот (20 баллов)

Условие

Автомобиль с автопилотом запрограммирован таким образом, что при движении по прямой на трассе он всегда старается поддерживать дистанцию между собой и движущимся непосредственно перед ним автомобилем ровно такой же, как между собой и движущимся непосредственно за ним автомобилем. В некоторый момент движения скорости всех трех этих автомобилей были равны. Определите ускорение автомобиля, движущегося непосредственно за автопилотируемым, если модули ускорения самого автопилотируемого автомобиля и движущегося непосредственно перед ним в этот момент оба оказались равны $a = 0{,}2\,\text{м/с}^2$, но дистанция между ними при этом начала сокращаться. Колеса автомобилей движутся без проскальзывания, и автопилоту удается соблюдать требования своей программы.

Решение

Программа автомобиля означает, что (до тех пор, пока это позволяет мощность двигателя и сцепление колес) координата x вдоль оси, совпадающей с дорогой, ав-

томобиля с автопилотом равна среднему арифметическому координат x_1 идущего впереди и x_2 идущего позади

$$x = \frac{x_1 + x_2}{2}. \tag{2.3.130}$$

Поскольку такая кинематическая связь справедлива в любые два момента времени t_1 и t_2, для средней скорости v автомобиля на автопилоте в проекции на $0x$ на любом промежутке времени можно записать

$$v_x = \frac{x(t_2) - x(t_1)}{t_2 - t_1} = \frac{x_1(t_2) + x_2(t_2) - x_1(t_1) - x_2(t_1)}{2(t_2 - t_1)} = \frac{v_{x1} + v_{x2}}{2}, \tag{2.3.131}$$

где использована та же система индексов. Таким образом, средняя скорость на любом промежутке времени и, следовательно, мгновенная скорость в любой момент времени автомобиля на автопилоте в проекции на $0x$ равна среднему арифметическому мгновенной скорости в этой же проекции впереди и позади идущих автомобилей. Повторение этих рассуждений приводит к аналогичному результату для ускорений

$$a_x = \frac{a_{x1} + a_{x2}}{2}. \tag{2.3.132}$$

Выразим из этого уравнения проекцию на $0x$ искомого ускорения замыкающего автомобиля a_{x2}

$$a_{x2} = 2a_x - a_{x1}. \tag{2.3.133}$$

По условиям задачи в рассматриваемый момент скорости всех трех автомобилей равны, а ускорения a_1 и a совпадают по модулю, но дистанция начинает сокращаться. Это возможно только если передний автомобиль тормозит — имеет отрицательную проекцию ускорения на направление движения, а автопилотируемый автомобиль, напротив, ускоряется (имеет положительную проекцию). Тогда напрямую из (2.3.133) получим

$$a_{x2} = 2a_x - a_{x1} = 2a - (-a) = 3a = 0{,}6\,\text{м/с}^2. \tag{2.3.134}$$

Погрешность $0{,}01\,\text{м/с}^2$.

Ответ: $(0{,}60 \pm 0{,}01)\,\text{м/с}^2$.

Задача 2.3.8.4. Лифт (25 баллов)

Условие

В лифте, движущемся вверх с некоторым ускорением a, сонаправленным его скорости, уронили без начальной скорости относительно лифта мячик с высоты $h_1 = 0{,}6\,\text{м}$ над уровнем пола лифта. Мячик абсолютно упруго ударился о пол, но своим ударом спровоцировал срабатывание системы аварийной остановки, в результате чего в момент удара ускорение лифта резко поменяло направление на противоположное, а его модуль возрос втрое. После отскока мячик поднялся до высоты $h_2 = 1{,}8\,\text{м}$. Определите a. Ускорение свободного падения $g = 9{,}8\,\text{м/с}^2$.

Решение

В системе отсчета, связанной с лифтом, начальное ускорение мяча равно $g + a$. Проходя с этим ускорением расстояние h_1, мяч приобретает скорость (относительно лифта) v, которую легко вычислить из соотношения

$$\frac{v^2}{2} = (g + a)h_1. \tag{2.3.135}$$

В момент удара резко меняется ускорение, но не скорость лифта, поэтому значение v при абсолютно упругом ударе по модулю остается неизменным.

В процессе подъема мяч уже имеет относительно лифта ускорение $g - 3a$, что позволяет записать аналогично

$$\frac{v^2}{2} = (g - 3a)h_2. \tag{2.3.136}$$

Совмещая эти равенства, получим окончательно

$$(g + a)h_1 = (g - 3a)h_2 \Rightarrow a(h_1 + 3h_2) = g(h_2 - h_1) \Rightarrow a = g\frac{h_2 - h_1}{h_1 + 3h_2} = 1{,}96\,\text{м/с}^2. \tag{2.3.137}$$

Погрешность $0{,}02$ м/с^2.

Ответ: $(1{,}96 \pm 0{,}02)$ м/с^2.

Задача 2.3.8.5. Эффект Лейденфроста (30 баллов)

Условие

Капля воды при температуре немного ниже температуры кипения упала на раскаленную поверхность, в результате чего $\alpha = 10^{-5}$ ее массы практически мгновенно испарилось. Известно, что $\eta = 3\%$ полученной каплей энергии пошло на работу расширяющегося пара над оставшейся частью капли.

На какую высоту «подпрыгнет» капля вертикально вверх в результате такого испарения, если сопротивлением воздуха ее движению, а также потерями тепла в окружающую среду и работой пара против воздуха можно пренебречь?

Удельная теплота парообразования воды равна $L = 2{,}26$ МДж/кг, ускорение свободного падения $g = 9{,}8\,\text{м/с}^2$.

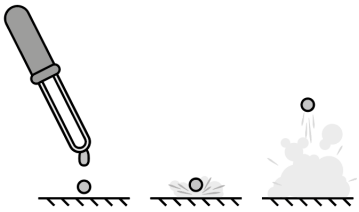

Рис. 2.3.23

Решение

Согласно первому началу термодинамики та теплота, полученная каплей, которая не пошла на работу расширяющегося пара над жидкой частью капли, ушла на изменение ее внутренней энергии; в данном случае — на испарение. Таким образом, можно записать для для этой части энергии

$$\Delta U = (1-\eta)Q = Lm\alpha, \tag{2.3.138}$$

где m — масса капли до испарения, а Q — общая полученная каплей теплота. Отсюда легко найти Q

$$Q = \frac{Lm\alpha}{1-\eta}. \tag{2.3.139}$$

В то же время работа пара над каплей полностью идет на увеличение ее механической энергии, которая в верхней точке траектории чисто потенциальна

$$A = \eta Q = (1-\alpha)mgh. \tag{2.3.140}$$

Выражая из этого уравнения h и подставляя в него Q, получаем

$$h = \frac{\eta Q}{(1-\alpha)mg} = \frac{L\alpha\eta}{(1-\alpha)(1-\eta)g} \approx 7{,}1\,\text{см}. \tag{2.3.141}$$

Разумеется, если величину $1-\alpha$ в этом выражении считать просто единицей, ответ не изменится в пределах любой разумной погрешности.

Погрешность $0{,}5$ см.

Ответ: $(7{,}1 \pm 0{,}5)$ см.

2.4. Инженерный тур

Задача 2.4.1. Ослепленная видеокамера (10 баллов)

Темы: компьютерное зрение, цветовые пространства, насыщенность и яркость пикселей.

Условие

Беспилотник ориентируется по изображению с камеры. На каждом кадре есть блики. На некоторых кадрах бликов так много, что камера «слепнет». В таких случаях беспилотник должен остановиться и дождаться исчезновения бликов.

Необходимо написать функцию, которая анализирует блики на изображении и выносит вердикт: можно ориентироваться по изображению или нет.

Чтобы понять, какой уровень свечения является допустимым, проанализируйте подготовленный набор данных.

Выполнение

1. Скачайте материалы задания: https://disk.yandex.ru/d/0yh18sD7t3TPLg.
2. Ознакомьтесь с материалами задания.
3. Ознакомьтесь с подготовленными файлами `.py`, набором изображений и аннотациями к нему. Среди подготовленных файлов:
   - `eval.py` — файл с функцией, определяющей засвечено изображение или нет. Именно эту функцию необходимо дописать!
   - `main.py` — файл, проверяющий точность работы вашего алгоритма. Не редактируйте его — `main.py` использует написанные вами функции из `eval.py` и сверяет верные ответы с предсказанием вашего алгоритма.
   - `annotations.csv` — файл, устанавливающий соответствие между изображениями и верными ответами. В каждой строке файла содержится путь к файлу с изображением и соответствующим ответом.
4. Прочитайте файл `eval.py`. В этом файле содержится функция `predict_illumination`. Допишите в файле `eval.py` функцию `track_movement`. Функция получает на вход `BGR`-изображение и должна вернуть `True` или `False`.
5. Запустите файл `main.py` и проверьте свой алгоритм. Если программа выдала ошибку, то найдите и исправьте ее в файле `eval`, вновь запустите файл `main.py`.
6. Пришлите решение на онлайн-платформу для проверки.
   В качестве решения необходимо сдать отредактированный файл `eval.py`, либо архив `*.zip` с файлом `eval.py` и остальными файлами, требующимися для его работы. В архиве должны находиться файлы, а не одноименная архиву папка.

Технические ограничения

Размер решения ограничен: не более 0,1 МБ. Если созданный алгоритм успешно проверен платформой, то следующее решение можно прислать только через 10 мин. Если написанный алгоритм в ходе проверки выдал сообщение об ошибке, то следующее решение можно прислать сразу.

Пакеты, ориентированные на работу с изображениями и данными, использующиеся на платформе проверки: `Python 3.8.10; dlib 19.24.0; keras 2.8.0; Keras-Preprocessing 1.1.2; imutils 0.5.4; numpy 1.22.4; opencv-python 4.6.0.66; pandas 1.4.3; scikit-image 0.19.3; scikit-learn 1.1.1; scipy 1.8.1; tensorflow-cpu 2.8.2.`

Решение

Перевести изображение в цветовое пространство `HLS`, просуммировать `L` составляющую цвета на всем изображении. Определить по представленным изображениям, какая сумма `L` считается допустимой. Все изображения, где сумма `L` больше определенного порога, считать засвеченными.

Предоставленный пользователю набор данных и решение: https://disk.yandex.ru/d/0yh18sD7t3TPLg.

Задача 2.4.2. Слишком большой груз (20 баллов)

Темы: компьютерное зрение, бинаризация, поиск контуров, контурный анализ.

Условие

Захват беспилотника имеет цилиндрическую форму. Он опускается на захватываемый объект вертикально вниз. Если объект не поместится в круговое основание цилиндра, то он не поместится и в захвате.

Дано изображение с камеры беспилотника, собирающегося воспользоваться захватом. На изображении представлены два объекта: красный куб и некий груз. Известно, что красный куб — куб максимального размера, который может поместиться в захвате. Необходимо написать функцию, определяющую, поместится ли груз на изображении в захвате или нет. Считайте, что камера направлена вертикально вниз.

Анализируйте только то положение груза, которое представлено на изображении!

Выполнение

1. Скачайте материалы задания: https://disk.yandex.ru/d/bEtPQB6oHcF-Wg.
   Ознакомьтесь с материалами задания.
2. Ознакомьтесь с подготовленными файлами `.py`, набором изображений и аннотациями к нему. Среди подготовленных файлов:

- `eval.py` — файл с функцией, определяющей поместится груз в захвате или нет. Именно эту функцию необходимо дописать!
- `main.py` — файл, проверяющий точность работы вашего алгоритма. Не редактируйте его — `main.py` использует написанные вами функции из `eval.py` и сверяет верные ответы с предсказанием вашего алгоритма.
- `annotations.csv` — файл с аннотациями, устанавливающий соответствие между изображениями и верными ответами. В каждой строке содержится путь к файлу с изображением и соответствующим ответом.

3. Прочитайте файл `eval.py`. В файле содержится функция `redict_ability_capture`.
4. Допишите в файле `eval.py` функцию `predict_ability_capture`.
   Функция получает на вход BGR-изображение и должна вернуть `True` или `False`.
5. Запустите файл `main.py` и проверьте свой алгоритм.
   Если программа выдала ошибку, то найдите и исправьте ее в файле `eval`, вновь запустите файл `main.py`.
6. Пришлите решение на онлайн-платформу для проверки.
   В качестве решения, необходимо сдать отредактированный файл `eval.py`, либо архив «`*.zip`» с файлом `eval.py` и остальными файлами, требующимися для его работы. В архиве должны находиться файлы, а не одноименная архиву папка.

Технические ограничения

Размер решения ограничен: не более 0,1 МБ. Если созданный алгоритм успешно проверен платформой, то следующее решение можно прислать только через 10 мин. Если написанный алгоритм в ходе проверки выдал сообщение об ошибке, то следующее решение можно прислать сразу.

Пакеты, ориентированные на работу с изображениями и данными, использующиеся на платформе проверки: `Python 3.8.10 dlib 19.24.0; keras 2.8.0 ; Keras-Preprocessing 1.1.2; imutils 0.5.4; numpy 1.22.4; opencv-python 4.6.0.66; pandas 1.4.3; scikit-image 0.19.3; scikit-learn 1.1.1; scipy 1.8.1; tensorflow-cpu 2.8.2.`

Решение

Размыть изображение для увеличения однородности цвета. Бинаризовать изображение так, чтобы красный куб стал белым, а все остальное — черным. Найти контур куба и вычислить радиус описывающей его окружности.

Бинаризовать исходное изображение так, чтобы фон и красный куб стали черными, а все остальное — белым. Найти контур груза, вычислить радиус окружности, описывающей контур. Если радиус меньше радиуса, найденного для куба, значит, груз помещается в захвате.

Предоставленный пользователю набор данных и решение: https://disk.yandex.ru/d/bEtPQB6oHcF-Wg.

Задача 2.4.3. Разъем для подзарядки (30 баллов)

Темы: компьютерное зрение, бинаризация, контурный анализ, детектирование объектов по цветам, распознавание сравнением с шаблоном.

Условие

На изображении представлены четыре разъема станции для зарядки беспилотников. Из всех разъемов для нашего беспилотника подходит только один, приведенный на рис. 2.4.1.

Рис. 2.4.1. Красный разъем

Необходимо написать функцию, определяющую номер подходящего разъема. Нумерация разъемов начинается с единицы и идет слева направо.

Выполнение

1. Скачайте материалы задания: https://disk.yandex.ru/d/c9Jkt5A2i6HAgA.
   Ознакомьтесь с материалами задания.
2. Ознакомьтесь с подготовленными файлами `.py`, набором изображений и аннотациями к нему. Среди подготовленных файлов:
   - `eval.py` — файл с функцией, определяющей номер подходящего разъема. Именно эту функцию необходимо дописать!
   - `main.py` — файл, проверяющий точность работы вашего алгоритма. Не редактируйте его — `main.py` использует написанные вами функции из `eval.py` и сверяет верные ответы с предсказанием вашего алгоритма.
   - `annotations.csv` — файл, устанавливающий соответствие между изображениями и верными ответами. В каждой строке файла содержится путь к файлу с изображением и соответствующим ответом.
3. Прочитайте файл `eval.py`.
   В файле содержатся функции `predict_connect_number` и `load_etalon`. `predict_connect_number` — определяет номер подходящего разъема.

`load_etalon_img` — загружает эталонные изображения, возвращает список изображений. Если эта функция не используется, пусть возвращает пустой список.

4. Допишите в файле `eval.py` функцию `predict_connect_number`.
   Функция получает на вход `BGR`-изображение и должна вернуть целое число.
5. Запустите файл `main.py` и проверьте свой алгоритм.
   Если программа выдала ошибку, то найдите и исправьте ее в файле `eval`, вновь запустите файл `main.py`.
6. Пришлите решение на онлайн-платформу для проверки.
   В качестве решения, необходимо сдать отредактированный файл `eval.py`, либо архив «`*.zip`» с файлом `eval.py` и остальными файлами, требующимися для его работы. В архиве должны находиться файлы, а не одноименная архиву папка.

Технические ограничения

Размер решения ограничен: не более 0,1 МБ. Если созданный алгоритм успешно проверен платформой, то следующее решение можно прислать только через 10 мин. Если написанный алгоритм в ходе проверки выдал сообщение об ошибке, то следующее решение можно прислать сразу.

Пакеты, ориентированные на работу с изображениями и данными, использующиеся на платформе проверки: `Python 3.8.10; dlib 19.24.0; keras 2.8.0; Keras-Preprocessing 1.1.2; imutils 0.5.4; numpy 1.22.4; opencv-python 4.6.0.66; pandas 1.4.3; scikit-image 0.19.3; scikit-learn 1.1.1; scipy 1.8.1; tensorflow-cpu 2.8.2.`

Решение

Бинаризовать изображение так, чтобы красные разъемы стали белыми, а все остальные пиксели изображения стали черным. Найти контуры красных разъемов. Найти ограничивающие контуры рамки. По координатам рамок вырезать из исходного изображения обнаруженные объекты.

Из предоставленного набора данных вырезать эталонные изображения искомого разъема. Попиксельно сравнить обнаруженные объекты с эталонами, запомнить координаты объекта, совпавшего с эталоном.

Возможен вариант решения через распознавание формы и цвета разъема. Пример подобного решения представлен ниже.

Предоставленный пользователю набор данных и решение: https://disk.yandex.ru/d/c9Jkt5A2i6HAgA.

Задача 2.4.4. Знакомое созвездие (40 баллов)

Темы: компьютерное зрение, бинаризация, контурный анализ, декартовы координаты, полярные координаты, комбинаторика.

Условие

Дана пара изображений, на каждом изображении — одно созвездие. Созвездие характеризуется следующими параметрами: количеством звезд, размером звезд, взаимным расположением звезд.

Необходимо написать функцию, определяющую, одно и то же созвездие на изображениях или нет. Созвездия считаются одинаковыми, если второе изображение можно получить из первого двумя последовательными операциями. Первая операция — масштабирование изображения с сохранением пропорций. Вторая операция — поворот изображения в собственной плоскости относительно любой точки.

Выполнение

1. Скачайте материалы задания: https://disk.yandex.ru/d/T7qC-weButM-eA.
   Ознакомьтесь с материалами задания.
2. Ознакомьтесь с подготовленными файлами `.py`, набором изображений и аннотациями к нему. Среди подготовленных файлов:
   - `eval.py` — файл с функцией, определяющей одинаковые созвездия на изображениях или нет. Именно эту функцию необходимо дописать!
   - `main.py` — файл, проверяющий точность работы вашего алгоритма. Не редактируйте его — `main.py` использует написанные вами функции из `eval.py` и сверяет верные ответы с предсказанием вашего алгоритма.
   - `annotations.csv` — файл, устанавливающий соответствие между парами изображений и верными ответами. В каждой строке файла содержится путь к двум изображениям, соответствующим ответом и пояснением.
3. Прочитайте файл `eval.py`.
4. В файле содержатся функция `is_same_stars`.
   Допишите в файле `eval.py` функцию `is_same_stars`.
   Функция получает на вход `BGR`-изображение и должна вернуть True или False.
5. Запустите файл `main.py` и проверьте свой алгоритм.
   Если программа выдала ошибку, то найдите и исправьте ее в файле `eval`, вновь запустите файл `main.py`.
6. Пришлите решение на онлайн-платформу для проверки.
   В качестве решения, необходимо сдать отредактированный файл `eval.py`, либо архив «`*.zip`» с файлом `eval.py` и остальными файлами, требующимися для его работы. В архиве должны находиться файлы, а не одноименная архиву папка.

Технические ограничения

Размер решения ограничен: не более 0,1 МБ. Если созданный алгоритм успешно проверен платформой, то следующее решение можно прислать только через 10 мин. Если написанный алгоритм в ходе проверки выдал сообщение об ошибке, то следующее решение можно прислать сразу.

Пакеты, ориентированные на работу с изображениями и данными, использующиеся на платформе проверки: `Python 3.8.10; dlib 19.24.0; keras 2.8.0; Keras-Preprocessing 1.1.2; imutils 0.5.4; numpy 1.22.4; opencv-python 4.6.0.66; pandas 1.4.3; scikit-image 0.19.3; scikit-learn 1.1.1; scipy 1.8.1; tensorflow-cpu 2.8.2`.

Решение

Конфигурация созвездия полностью характеризуется следующими данными:

- Координаты геометрического центра созвездия или центра масс созвездия.
- Список размеров звезд.
- Список расстояний от звезд до центра созвездия.
- Список углов, образованных звездами и центром созвездия.

Для каждого изображения необходимо вычислить вышеперечисленные параметры. Если созвездия одинаковые, то выполнятся следующие условия, которые необходимо проверить:

- Число звезд и их размеры должны совпадать.
- Все расстояния от звезд до центра созвездия для двух сравниваемых изображений должны отличаться в одно и тоже количество раз.
- Списки углов, образованных звездами и центрами созвездий, должны совпадать или совпадать со смещением.

Предоставленный пользователю набор данных и решение: https://disk.yandex.ru/d/T7qC-weButM-eA.

3. Второй отборочный этап

3.1. Работа наставника НТО на этапе

На втором отборочном этапе НТО участникам предстоит решать как индивидуальные, так и командные задачи в рамках выбранного профиля. Подготовка к этому этапу требует от них не только глубокого понимания предметной области, но и умения работать в команде, эффективно распределять роли и применять полученные знания на практике. Наставник играет здесь важную роль — он помогает участникам выстроить осмысленную и целенаправленную траекторию подготовки.

Вот основные направления, в которых наставник может поддержать участника:

- **Подготовка по образовательным программам НТО.** Наставник может готовить участников, используя готовые образовательные программы по технологическим направлениям, рекомендованные организаторами, а также адаптировать их под уровень подготовки школьников.
- **Разбор заданий прошлых лет.** Изучение задач второго отборочного этапа прошлых лет помогает участникам понять формат заданий, определить типовые ошибки и выработать стратегии решения.
- **Онлайн-курсы.** Участники могут пройти курсы по разбору задач прошлых лет или курсы, рекомендованные разработчиками отдельных профилей. Наставник может включить эти курсы в план подготовки, а также сопровождать процесс изучения и помогать с возникшими вопросами.
- **Анализ материалов профиля.** Совместный разбор методических материалов, размещенных на страницах профилей, помогает уточнить требования к участникам и направить подготовку на ключевые темы.
- **Практикумы.** Это важный элемент подготовки, позволяющий применять знания на практике. Наставник может:
  - ◇ организовать практикумы по методическим материалам с сайта профиля;
  - ◇ декомпозировать задачи заключительного этапа прошлых лет на отдельные элементы и проработать их с участниками;
  - ◇ провести анализ требуемых профессиональных компетенций и спланировать занятия для развития наиболее значимых из них;
  - ◇ направить участников на практикумы и мероприятия от организаторов, которые анонсируются в официальных сообществах НТО, например, в телеграм-канале для наставников: https://t.me/kruzhok_association.
- **Командная работа.** Одной из ключевых задач наставника на втором этапе является помощь в формировании команды или в поиске подходящей. Наставник может помочь участникам определить их сильные стороны, выбрать роль в команде и сориентироваться в процессе командообразования, включая участие в бирже команд в рамках конкретного профиля.

Если участники не прошли отборочный этап

Случается, что несмотря на усилия и серьезную подготовку, участники не проходят во второй или заключительный этап Олимпиады. В такой ситуации особенно важна поддержка наставника.

- **Поддержка и признание усилий.** Наставнику важно подчеркнуть ценность пройденного пути: полученные знания, навыки, преодоленные трудности и личностный рост. Это помогает участникам сохранить мотивацию и не воспринимать результат как окончательное поражение.
- **Рефлексия.** Полезно организовать встречу для обсуждения впечатления от участия, трудности, с которыми столкнулись школьники и то, что они узнали о себе и команде. Наставник может направить разговор в конструктивное русло: какие выводы можно сделать? Что сработало хорошо? Что можно улучшить?
- **Анализ ошибок и пробелов.** Наставник вместе с участниками анализирует, какие темы вызвали наибольшие затруднения, чего не хватило в подготовке — теоретических знаний, практических навыков, командного взаимодействия. Это позволяет выстроить более эффективную стратегию на будущее.
- **Планирование дальнейшего пути.** Участникам можно предложить:
  - ◇ продолжить углубленное изучение профиля или смежных направлений;
  - ◇ заняться проектной деятельностью, которая укрепит знания и навыки;
  - ◇ сформировать план по подготовке к следующему циклу НТО, начиная с работы над типовыми заданиями и курсами.
- **Создание устойчивой мотивации.** Важно показать школьникам, что участие в НТО — это не просто соревнование, а часть большого образовательного маршрута. Даже неудачный результат может стать толчком к профессиональному росту, если воспринимать его как точку развития, а не как конец пути.

Таким образом, наставник помогает участникам не только готовиться к этапам НТО, но и справляться с неудачами, выстраивать долгосрочную стратегию и сохранять интерес к инженерному и технологическому творчеству.

3.2. Инженерный тур

Задача 3.2.1. Определение прочности дороги (20 баллов)

Темы: компьютерное зрение, машинное обучение, сверточные нейронные сети, регрессия.

Условие

На изображении представлен фрагмент поверхности дорожного полотна. Поверхность может иметь повреждения двух типов: ямы и трещины. Каждое повреждение снижает прочность полотна. Задача — написать функцию, которая определяет прочность дорожного полотна на изображении.

Максимальную прочность имеет неповрежденное полотно из бетонных плит, ее значение принято за 1. Прочность полотна, которое непригодно к использованию из-за множества повреждений, принята за 0. Обратите внимание, что неповрежденные полотна разных типов могут иметь различную прочность.

Рекомендуется использовать методы машинного обучения для решения этой задачи.

Материалы к заданию:

https://disk.yandex.ru/d/zUqLGZ-yTR1Tbg/Задача1/user_task.

Выполнение

1. Скачайте материалы задания.
2. Ознакомьтесь с материалами задания.

   В материалах подготовлены несколько файлов `.py`, набор изображений и аннотации к нему.

   В списке подготовленных файлов содержатся:

   - `eval.py` — файл с функцией, которую вам необходимо дописать;
   - `main.py` — файл, проверяющий точность работы вашего алгоритма; не редактируйте его. `main.py` использует написанные вами функции из `eval.py` и сверяет верные ответы с предсказанием вашего алгоритма;
   - `annotations.csv` — файл, устанавливающий соответствие между изображениями и верными ответами; в каждой строке файла содержится путь к изображению и соответствующим ему значение прочности.
3. Прочитайте файл `eval.py`.

   В файле содержатся функции `load_models` и `get_road_health`:

   - `load_models` — загружает из файлов модели машинного обучения;
   - `get_road_health` — определяет прочность дорожного полотна на изображении, именно ее необходимо дописать.
4. Допишите функции в файле `eval.py`.

5. Запустите файл `main.py` и проверьте свой алгоритм.

   Если программа выдала ошибку, то найдите и исправьте ее в файле `eval`, вновь запустите файл `main.py`.

6. Пришлите решение на онлайн-платформу для проверки.

   В качестве решения необходимо сдать отредактированный файл `eval.py`, либо архив `*.zip` с файлом `eval.py` и остальными файлами, требующимися для его работы. В архиве должны находиться файлы, а не одноименная архиву папка.

Технические ограничения

Размер решения ограничен: не более 15 МБ. Если алгоритм успешно проверен платформой, то следующее решение можно прислать только через 10 мин. Если алгоритм в ходе проверки выдал сообщение об ошибке, то следующее решение можно прислать сразу.

Пакеты, ориентированные на работу с изображениями и данными, использующиеся на платформе проверки:

`Python 3.8.10; catboost 1.1.1; dlib 19.24.0; gast 0.4.0; h5py 3.7.0; imutils 0.5.4; keras 2.9.0; Keras-Preprocessing 1.1.2; matplotlib 3.6.2; numpy 1.23.2; opencv-python 4.6.0.66; pandas 1.5.1; scikit-image 0.19.3; scikit-learn 1.1.3; scipy 1.9.3; tensorflow-cpu 2.9.2; torch 1.13.0; torchaudio 0.13.0; torchvision 0.14.0.`

Используйте совместимые пакеты.

Решение

Для решения задания необходимо обучить модель машинного обучения, решающую задачу регрессии. В качестве данных для обучения нужно использовать материалы, предоставленные пользователям. Использование нейронной сети вполне оправдано при решении данной задачи; главное — в последнем слое оставить один нейрон и линейную функцию активации. Функции потерь при обучении: `MSE`, `MAE`, `Hubel Loss`, `MSLE`.

В приведенном решении используется полносвязная нейронная сеть со следующей архитектурой:

```
super(SimpleNet, self).__init__()
self.fc1 = nn.Linear(400 * 400, 128)
self.relu1 = nn.ReLU()
self.fc2 = nn.Linear(128, 64)
self.relu2 = nn.ReLU()
self.fc3 = nn.Linear(64, 1)
```

Предоставленный пользователю набор данных и решение:

https://disk.yandex.ru/d/zUqLGZ-yTR1Tbg/Задача1.

Файл с решением — `solution.py`.

Задача 3.2.2. Обмен данными через MQTT (10 баллов)

Темы: обмен данными, обработка сообщений, MQTT, подписчик-издатель.

Условие

Устройства современной автономной транспортной системы должны обмениваться данными друг с другом. Один из возможных протоколов обмена данными — протокол `MQTT`. Задача — написать `MQTT`-клиента, который будет прослушивать несколько каналов, обрабатывать данные из них и публиковать результаты обработки в другие каналы.

Для решения задачи необходимо ознакомиться с устройством протокола `MQTT`. Для создания клиента используйте библиотеку `paho-mqtt`.

Клиент должен прослушивать каналы `*2` и `*3`. В эти каналы приходят сообщения в формате «число; имя канала». Если сообщение пришло в канал `*2`, то число в нем нужно умножить на 2 и результат опубликовать в канале, название которого пришло в сообщении. Аналогично для канала `*3`, только число нужно умножить на 3.

Кроме того, клиент должен прослушивать каналы `addend` и `command`. В `addend` будут приходить сообщения с числами. В `command` будут приходить сообщения с названием канала. Как только в `command` приходит сообщение, необходимо суммировать все новые числа из канала `addend` и опубликовать в канале, название которого указано в сообщении.

Еще клиент должен прослушивать канал `numbers`. В него приходят сообщения, содержащие числа. Все нечетные числа клиент должен пересылать в канал `odd`.

Сообщения передаются в формате `bytes`, для их преобразования в текст используйте `decode("utf-8")`.

В сообщениях всегда одно число и/или одно имя канала.

Материалы к заданию:

https://disk.yandex.ru/d/zUqLGZ-yTR1Tbg/3адача2/user_task.

Выполнение

1. Скачайте материалы задания.
2. Ознакомьтесь с материалами задания.

   В материалах подготовлены несколько файлов `.py`. В списке подготовленных файлов содержатся:

   - `eval.py` — файл с функциями, которые необходимо дописать;
   - `main.py` — файл, проверяющий точность работы вашего алгоритма; не редактируйте его. `main.py` использует написанные вами функции из `eval.py` и сверяет верные ответы с предсказанием вашего алгоритма;
   - `annotations.json` — файл, устанавливающий соответствие между тестами и верными ответами.
3. Прочитайте файл `eval.py`.

   В файле содержатся функции `setup` и `main_loop`:

- `setup` — инициализирует `MQTT`-клиент;
- `main_loop` — реализует цикл приема и обработки сообщений.

4. Допишите функции в файле `eval.py`.
5. Запустите файл `main.py` и проверьте свой алгоритм.

   Если программа выдала ошибку, то найдите и исправьте ее в файле `eval`, вновь запустите файл `main.py`.
6. Пришлите решение на онлайн-платформу для проверки.

   В качестве решения необходимо сдать отредактированный файл `eval.py`, либо архив `*.zip` с файлом `eval.py` и остальными файлами, требующимися для его работы. В архиве должны находиться файлы, а не одноименная архиву папка.

Технические ограничения

Размер решения ограничен: не более 1 МБ. Если алгоритм успешно проверен платформой, то следующее решение можно прислать только через 10 мин. Если алгоритм в ходе проверки выдал сообщение об ошибке, то следующее решение можно прислать сразу.

Пакеты, ориентированные на работу с изображениями и данными, использующиеся на платформе проверки:

```
Python 3.8.10; catboost 1.1.1; dlib 19.24.0; gast 0.4.0; h5py
3.7.0; imutils 0.5.4; keras 2.9.0; Keras-Preprocessing 1.1.2;
matplotlib 3.6.2; numpy 1.23.2; opencv-python 4.6.0.66; pandas
1.5.1; scikit-image 0.19.3; scikit-learn 1.1.3; scipy 1.9.3;
tensorflow-cpu 2.9.2; torch 1.13.0; torchaudio 0.13.0;
torchvision 0.14.0.
```

Используйте совместимые пакеты.

Решение

Для решения задания необходимо разобраться в устройстве протокола `MQTT`. Понять, что брокер запускать и настраивать не нужно, он уже запущен на платформе проверки. Необходимо подписаться на каналы и обрабатывать входящие сообщения. При подключении к брокеру следует использовать адрес 127.0.0.1, а не псевдоним `localhost`.

Предоставленный пользователю набор данных и решение:

https://disk.yandex.ru/d/zUqLGZ-yTR1Tbg/3адача2.

Файл с решением — `solution.py`.

Задача 3.2.3. Детектирование квадрокоптеров (35 баллов)

Темы: компьютерное зрение, детектирование объектов на изображении, нейросетевые детекторы, YOLO.

Условие

Задача — написать функцию, определяющую положение квадрокоптера на изображении. На каждом изображении ровно один квадрокоптер. В качестве данных для обучения можно использовать фотографии из материалов к заданию.

Материалы к заданию:

https://disk.yandex.ru/d/zUqLGZ-yTR1Tbg/Задача3/user_task.

Выполнение

1. Скачайте материалы задания.
2. Ознакомьтесь с материалами задания.

   В материалах подготовлены несколько файлов `.py`, набор изображений и аннотации к нему. В списке подготовленных файлов содержатся:

   - `eval.py` — файл с функциями, которые необходимо дописать;
   - `main.py` — файл проверяющий, точность работы вашего алгоритма; не редактируйте его. `main.py` использует написанные функции из `eval.py` и сверяет верные ответы с предсказанием вашего алгоритма;
   - `annotations.csv` — файл, устанавливающий соответствие между изображениями и верными ответами; в каждой строке файла содержится путь к изображению и параметрами ограничивающей квадрокоптер рамки.
3. Прочитайте файл `eval.py`.

   В файле содержатся функции `load_models` и `detect_drone`:

   - `load_models` — загружает из файлов модели машинного обучения.
   - `detect_drone` — обнаруживает квадрокоптер на изображении.
4. Допишите функции в файле `eval.py`.
5. Запустите файл `main.py` и проверьте свой алгоритм.

   Если программа выдала ошибку, то найдите и исправьте ее в файле `eval`, вновь запустите файл `main.py`.
6. Пришлите решение на онлайн-платформу для проверки.

   В качестве решения необходимо сдать отредактированный файл `eval.py`, либо архив `*.zip` с файлом `eval.py` и остальными файлами, требующимися для его работы. В архиве должны находиться файлы, а не одноименная архиву папка.

Технические ограничения

Размер решения ограничен: не более 30 МБ. Если алгоритм успешно проверен платформой, то следующее решение можно прислать только через 10 мин. Если алгоритм в ходе проверки выдал сообщение об ошибке, то следующее решение можно прислать сразу.

Пакеты, ориентированные на работу с изображениями и данными, использую-

щиеся на платформе проверки:

```
Python 3.8.10; catboost 1.1.1; dlib 19.24.0; gast 0.4.0; h5py
3.7.0; imutils 0.5.4; keras 2.9.0; Keras-Preprocessing 1.1.2;
matplotlib 3.6.2; numpy 1.23.2; opencv-python 4.6.0.66; pandas
1.5.1; scikit-image 0.19.3; scikit-learn 1.1.3; scipy 1.9.3;
tensorflow-cpu 2.9.2; torch 1.13.0; torchaudio 0.13.0;
torchvision 0.14.0.
```

Используйте совместимые пакеты.

Решение

Участникам необходимо обучить нейросетевой детектор. Основная сложность заключается в выборе модели детектора, подходящей под ограничения по размеру решения — 30 МБ, и такой, которую получится использовать через библиотеки, установленные на платформе проверки.

Таких детекторов несколько, наиболее часто используемые участниками — `Yolo3-tiny`, `Yolo4-tiny` и `Yolo5`. Два первых детектора удобно обучать в фреймворке `Darknet`, а инференс выполнять через `OpenCV.dnn`. Для запуска и обучения `Yolo5` используется библиотека `Ultralytics`; ее составляющие, необходимые для инференса, нужно было добавить в архив с решением.

В материалах ниже представлено решение с `Yolo5` и его инференсом через `PyTorch`.

Предоставленный пользователю набор данных и решение:

https://disk.yandex.ru/d/zUqLGZ-yTR1Tbg/Задача3.

Файл с решением — `solution.py`.

Задача 3.2.4. Определение загруженности дорог (20 баллов)

Темы: компьютерное зрение, бинаризация, поиск и анализ контуров, цветовые пространства.

Условие

На рис. 3.2.1 представлена карта загруженности дорог.

Черные круги — перекрестки. Перекрестки соединены дорогами. Из каждого перекрестка выходит хотя бы одна дорога. Дороги не пересекаются и не ветвятся. Каждая дорога соединяет только два перекрестка. Дороги состоят из одного или более участков. Участки бывают трех видов загруженности и обозначаются разными цветами: красным, желтым и зеленым.

Красный — максимальная загруженность, кодируется числом 3.

Желтый — средняя загруженность, кодируется числом 2.

Зеленый — минимальная загруженность, кодируется числом 1.

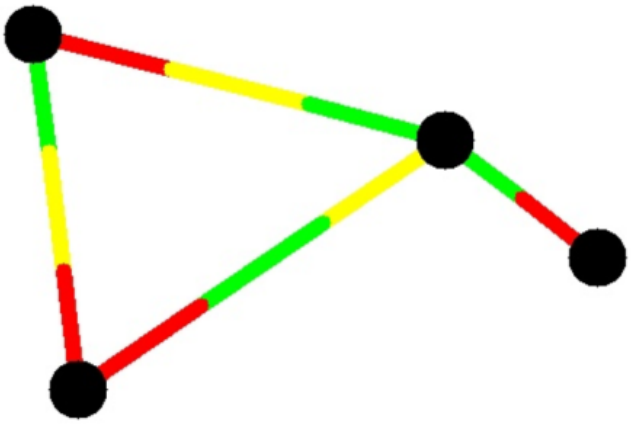

Рис. 3.2.1. Карта загруженности дорог

Задача — написать функцию, получающую «описание» каждого перекрестка на изображении. «Описание» перекрестка состоит из его номера и списков, отражающих загруженность каждой дороги, выходящей из перекрестка. Перекрестки нумеруются слева направо, начиная с единицы, рис. 3.2.2.

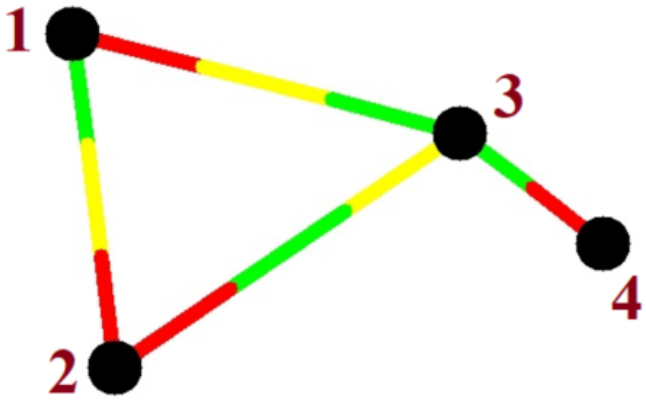

Рис. 3.2.2. Нумерация перекрестков

Описание для перекрестка 3 будет выглядеть следующим образом:

$$\text{«3»} : [[1, 2, 3], [2, 1, 3], [1, 3]],$$

где «3» — номер перекрестка, [1, 2, 3] — список участков, образующих дорогу, выходящую из перекрестка.

Порядок чисел важен и соответствует загруженности участков дороги в направлении от перекрестка.

Если из перекрестка выходит четыре дороги, то и в описании перекрестка должно быть четыре списка, неважно в каком порядке они будут идти. Для третьего перекрестка правильными будут шесть описаний.

Описания «3» : $[[1, 2, 3], [2, 1, 3], [1, 3]]$ и «3» : $[[1, 3], [2, 1, 3], [1, 2, 3]]$ эквивалентны.

Рекомендуем решать эту задачу без использования машинного обучения.

Материалы к заданию:

https://disk.yandex.ru/d/zUqLGZ-yTR1Tbg/Задача4/user_task.

Выполнение

1. Скачайте материалы задания.
2. Ознакомьтесь с материалами задания.

   В материалах подготовлены несколько файлов `.py`, набор изображений и аннотации к нему. В списке подготовленных файлов содержатся:

   - `eval.py` — файл с функцией, определяющей загруженность дорог по изображению, именно эту функцию необходимо дописать;
   - `main.py` — файл, проверяющий точность работы вашего алгоритма; не редактируйте его. `main.py` использует написанные функции из `eval.py` и сверяет верные ответы с предсказанием вашего алгоритма;
   - `annotations.json` — файл, устанавливающий соответствие между изображениями и верными ответами.
3. Прочитайте файл `eval.py`.

   В файле содержится функция `analyze_traffic`. Допишите ее.
4. Запустите файл `main.py` и проверьте свой алгоритм.

   Если программа выдала ошибку, то найдите и исправьте ее в файле `eval`, вновь запустите файл `main.py`.
5. Пришлите решение на онлайн-платформу для проверки.

   В качестве решения необходимо сдать отредактированный файл `eval.py`, либо архив `*.zip` с файлом `eval.py` и остальными файлами, требующимися для его работы. В архиве должны находиться файлы, а не одноименная архиву папка.

Технические ограничения

Размер решения ограничен: не более 1 МБ. Если алгоритм успешно проверен платформой, то следующее решение можно прислать только через 10 мин. Если алгоритм в ходе проверки выдал сообщение об ошибке, то следующее решение можно прислать сразу.

Пакеты, ориентированные на работу с изображениями и данными, использующиеся на платформе проверки:

```
Python 3.8.10; catboost 1.1.1; dlib 19.24.0; gast 0.4.0; h5py
3.7.0; imutils 0.5.4; keras 2.9.0; Keras-Preprocessing 1.1.2;
matplotlib 3.6.2; numpy 1.23.2; opencv-python 4.6.0.66; pandas
1.5.1; scikit-image 0.19.3; scikit-learn 1.1.3; scipy 1.9.3;
tensorflow-cpu 2.9.2; torch 1.13.0; torchaudio 0.13.0;
torchvision 0.14.0.
```

Используйте совместимые пакеты.

Решение

Первый шаг к решению задачи — поиск координат центров перекрестков. Центрами перекрестков считаются центры контуров, найденных на изображении, бинаризованном так, чтобы черные перекрестки стали белыми, а все остальное — черным.

Теперь необходимо попарно перебирать перекрестки. Для каждой пары вычисляется угол наклона прямой, проходящей через центры перекрестков. Он необходим для того, чтобы с некоторым шагом выбирать точки на этой прямой и анализировать их цвет. Если фиксируются изменения цвета точки, значит, начался новый участок дороги или дорога закончилась вовсе. Можно считать цвет предыдущей точки цветом участка дороги. Учитываются только красный, желтый и зеленый цвета. Если все точки белого цвета, значит, дороги нет вовсе.

Таким образом, для каждого перекрестка получается столько дорог, сколько из него выходит. И для каждой дороги определены составляющие ее участки в направлении от перекрестка.

Предоставленный пользователю набор данных и решение:

https://disk.yandex.ru/d/zUqLGZ-yTR1Tbg/Задача4.

Файл с решением — `solution.py`.

Задача 3.2.5. Полет квадрокоптера (15 баллов)

Темы: ROS, полетное задание квадрокоптера, взаимодействие нод в ROS, обработка изображения в ROS.

Условие

Квадрокоптер работает на базе ОС Ubuntu и ROS Noetic. Необходимо разобраться в том, как устроено полетное задание для квадрокоптера, какие события происходят в течение полета и как обрабатывать изображение с видеокамеры. Данные, генерируемые во время реального полета квадрокоптера, записаны и сохранены в формате `rosbag`.

Во время полета в кадр камеры попадает ArUCo метка семейства 6×6 250. Необходимо написать программу (`node`), которая после взлета квадрокоптера распознает метку на изображении. Как только метка найдена, необходимо опубликовать `id` метки в топике `/tag(std_msgs/Int32)`, после этого программа должна завершиться.

Файл `rosbag` можно проиграть встроенными средствами ROS, при проигрывании публикуются все необходимые для выполнения задания топики: `/pioneer_max_camera/image_raw(sensors_msgs/Image)` и `/geoscan/flight/callback_event (std_msgs/Int32)`.

Для решения данной задачи можно использовать только следующие инструменты: Python 3.8, OpenCv, CvBridge, ROS Noetic (ROS-Base).

Для установки ROS Noetic понадобится виртуальная машина с установленной ОС Ubuntu 20.04. Инструкция по установке Virtual Box: https://virtualbox.su/kak-ustanovit-windows-v-virtualbox/.

ОС Ubuntu 20.04: https://releases.ubuntu.com/20.04/.

Инструкция по установке ROS Noetic на Ubuntu 20.04, необходима версия ROS-Base: https://wiki.ros.org/noetic/Installation/Ubuntu.

Инструкция по использованию CvBridge для конвертации изображения из ROS в OpenCV: https://wiki.ros.org/cv_bridge/Tutorials/ConvertingBetweenROSImagesAndOpenCVImagesPython.

Инструкция по проигрыванию `rosbag`: https://wiki.ros.org/rosbag/Commandline#play.

Файл `rosbag`, предоставленный участникам, отличается от файла для проверки решений на онлайн-платформе.

Материалы к заданию:

https://disk.yandex.ru/d/zUqLGZ-yTR1Tbg/Задача5/user_task.

В качестве решения необходимо сдать файл `eval.py`.

Технические ограничения

Размер решения ограничен: не более 30 МБ. Если алгоритм успешно проверен платформой, то следующее решение можно прислать только через 10 мин. Если алгоритм в ходе проверки выдал сообщение об ошибке, то следующее решение можно прислать сразу.

Решение

`Rosbag`-файл необходимо проигрывать самостоятельно только при проверке задания локально, на собственном компьютере.

В решении для онлайн-платформы участникам необходимо подключиться к каналам, указанным в условии, и обрабатывать данные так, как при реальном полете. Система проверки выдаст ошибку, если верный ответ будет дан, например, после посадки.

Предоставленный пользователю набор данных и решение:

https://disk.yandex.ru/d/zUqLGZ-yTR1Tbg/Задача5.

Файл с решением — `solution.py`.

4. Заключительный этап

4.1. Работа наставника НТО при подготовке к этапу

На этапе подготовки к заключительному этапу НТО наставник решает две важные задачи: помощь участникам в подготовке к предстоящим соревнованиям и формирование устойчивой и слаженной команды. Заключительный этап требует высокой слаженности, уверенности и глубоких знаний, и наставник становится тем, кто объединяет усилия участников и направляет их в нужное русло.

Наставник помогает участникам:

- разобрать задания прошлых лет, используя официальные сборники, чтобы понять структуру финальных испытаний, типы задач и ожидаемый уровень сложности;
- изучить организационные особенности заключительного этапа, включая формат проведения, регламент, продолжительность и технические нюансы;
- спланировать подготовку — на основе даты начала финала составляется четкий график занятий, в котором распределены темы, практикумы и командные тренировки;
- обратиться (при необходимости) за консультацией к разработчикам заданий по профилю, уточнить, на какие аспекты подготовки следует обратить особое внимание, и получить дополнительные материалы.

Также рекомендуется участие в мероприятиях от организаторов, таких как:

- установочные вебинары и открытые разборы задач;
- хакатоны, практикумы и мастер-классы для финалистов;
- встречи в онлайн-формате, информация о которых публикуется в группе НТО во «ВКонтакте» и в телеграм-чатах профилей.

Наставнику необходимо уделить внимание работе на формированием устойчивой, продуктивной и мотивированной команды:

- **Сплочение команды.** Это особенно актуально, если участники живут в разных городах. Регулярные онлайн-встречи, совместная работа над задачами и неформальное общение помогают наладить доверие и улучшить командную динамику.
- **Анализ ролей.** Наставник вместе с командой определяет, кто за что отвечает, какие задачи входят в зону ответственности каждого участника. Также обсуждаются возможности взаимозаменяемости на случай непредвиденных ситуаций.
- **Оценка компетенций.** Важно определить, какими знаниями и навыками уже обладают участники, а какие необходимо развить. На основе этого формируется индивидуальный и командный план подготовки.
- **Участие в подготовительных мероприятиях от разработчиков профилей.**

Перед заключительным этапом проводятся установочные вебинары, разборы задач прошлых лет, практикумы, мастер-классы для финалистов. Информация о таких мероприятиях публикуется в группе НТО в VK и в чатах профилей в Telegram.

- **Практика в формате хакатонов.** Наставник может организовать дистанционные хакатоны или практикумы с использованием заданий прошлых лет и методических рекомендаций из официальных сборников.

Таким образом, наставник становится координатором и моральной опорой команды, помогая пройти заключительный этап НТО с максимальной уверенностью и результатом.

4.2. Предметный тур

Задачи третьего этапа предметного тура профиля по информатике открыты для решения. Участие в соревновании доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/72657/enter/.

4.2.1. Информатика. 8–11 классы

Задача 4.2.1.1. Новогодние каникулы на Плюке (10 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

На планете Плюк год длится N дней. При этом месяц на Плюке продолжается M дней, а неделя — K дней. Цикличность лет, месяцев и недель никак не зависит друг от друга и, например, окончание года вполне может прийтись на середину месяца.

Как и у всех нормальных людей, у жителей Плюка имеются новогодние каникулы, которые начинаются с первого дня нового года. Согласно плюкианскому закону, начало рабочего периода (и, соответственно, окончание новогодних каникул) должно приходиться либо на первый день первого месяца, либо на первый день первой недели, наступивший сразу после нового года. Так как в законе не уточняется, какой конкретно из этих двух вариантов нужно выбрать, легкомысленные плюкианцы выбирают начало рабочего года так, чтобы каникулы были как можно более длинными.

В текущем году им не повезло: первый день года стал первым днем недели и первым днем месяца, то есть каникул у них не было. Нужно определить, какую максимально разрешенную законом длительность новогодних каникул они могут позволить себе в следующем после текущего году.

Формат входных данных

На вход подаются три целых числа N, M и K через пробел. Эти числа задают соответственно продолжительность года, месяца и недели на Плюке.

$1 \leqslant M, K \leqslant N \leqslant 10^{18}$.

Формат выходных данных

Вывести одно число — максимально возможную длительность новогодних каникул при этих условиях.

Примеры

Пример №1

Стандартный ввод
17 8 9
Стандартный вывод
7

Пример №2

Стандартный ввод
365 30 7
Стандартный вывод
25

Примечания

В первом примере длительность года 17 дней, длительность месяца 8 дней и длительность недели 9 дней, то есть в текущем году в него вошло 2 месяца и 1 день, а значит, первый рабочий день (первый день следующего месяца) наступит на восьмой день после нового года, а 7 дней из предыдущего месяца могут быть выходными. С другой стороны, в текущем году была 1 неделя и еще 8 дней, то есть по этому параметру в новом году будет всего 1 выходной. Выбираем ответ — 7 дней.

Пример программы-решения

Ниже представлено решение на языке C++.

C++

```
#include<bits/stdc++.h>
#define all(a) a.begin(), a.end()
#define for0(i, n) for(int i = 0; i < n; i++)
#define for1(i, n) for(int i = 1; i <= n; i++)
#define x first
#define y second
#define int long long
using namespace std;
typedef long long ll;
typedef pair<int, int> pii;
```

```cpp
typedef vector<int> vi;
typedef vector<vector<int> > vvi;
const int INF = 1e18;
const int MOD = 1e9 + 7;
const int LG = 19;

signed main(){
        ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);
    int n, m, k;
    cin >> n >> m >> k;

    int a = m - n % m;
    if(a == m){
        a = 0;
    }
    int b = k - n % k;
    if(b == k){
        b = 0;
    }

    cout << max(a, b);
}
```

Задача 4.2.1.2. Правила перевозок пепелацем (15 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Согласно правилам перевозок, на каждые n пацаков, перевозимых пепелацем, должен быть назначен один чатланин, который должен их сопровождать и находиться в этом же пепелаце. Если количество пацаков в пепелаце не делится нацело на n, то для тех, кто в остатке, должен быть выделен еще один чатланин. При этом все находящиеся в пепелаце обязаны сидеть каждый на своем месте. Требуется перевезти k пацаков с Хануда на Плюк, и можно заказать любое количество пепелацев, каждый на m мест. Нужно узнать, какое минимальное количество пепелацев нужно заказать, чтобы вывезти всех k пацаков, не нарушая правила перевозок. Пацаков нужно вывезти за один раз, так как на планете Хануд долго жить нельзя. Можно считать, что всегда найдется необходимое количество чатлан, готовых прийти на помощь и поучаствовать в перевозке.

Формат входных данных

В единственной строке содержится три числа через пробел — n, k и m.

$1 \leqslant n, k \leqslant 10^9$, $2 \leqslant m \leqslant 10^9$.

Формат выходных данных

Вывести одно число — минимальное требуемое количество пепелацев.

Примеры

Пример №1

Стандартный ввод
`10 200 20`
Стандартный вывод
`12`

Примечания

Рассмотрим пример. На каждые 10 пацаков, находящихся в пепелаце, нужно выделить одного сопровождающего их чатланина. Требуется перевезти 200 пацаков, каждый доступный пепелац вмещает 20 человек. Тогда в одном пепелаце максимум можно перевезти 18 пацаков в сопровождении двух чатлан. Откуда получаем, что в 11 пепелацах можно перевезти 198 пацаков и потребуется заказать еще один для двух оставшихся пацаков.

Пример программы-решения

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long

using namespace std;
typedef pair<int, int> pii;


signed main(){
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);
    int n, t, m;
    cin >> n >> t >> m;
    int w = m / (n+1);
    int z = w * n + m % (n+1);
    if(m % (n+1) > 0) z--;
    int ans = t / z + ((t % z) > 0);
     cout << ans;
}
```

Задача 4.2.1.3. Пересыпание песка (20 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Би и Уэф заняты важным делом — добыванием чатлов. Они нашли странную конструкцию, состоящую из n ржавых емкостей. Емкости стоят в один ряд и имеют объемы $v_1, v_2, \ldots, v_n$. Изначально все емкости были пустыми.

Би нагребает песок в первую емкость объемом v_1 до ее верха, а Уэф нажимает кнопку. После этого автоматически происходит пересыпание песка из емкости № 1 в емкость № 2, затем из емкости № 2 в емкость № 3 и т. д. На каждом этапе песок пересыпается либо до тех пор, пока не опустеет емкость № i, либо пока до верху не наполнится емкость № $i + 1$. Из последней емкости № n песок высыпается обратно на поверхность планеты.

После этого Би снова засыпает доверху емкость № 1, и Уэф снова нажимает кнопку.

После каждого такого набора пересыпаний Уэф находит в последней емкости ровно столько чатлов, какова разность между объемом засыпанного в первую емкость песка и высыпавшегося из n-й на этой стадии.

В какой-то момент они заметили, что чатлы перестали появляться, то есть объем засыпанного песка стал равен объему высыпавшегося. Осталось выяснить, сколько всего чатлов они добыли таким образом.

Формат входных данных

В первой строке находится целое число n — количество емкостей, $1 \leqslant n \leqslant 10^5$.

Во второй строке через пробел находятся числа $v_1, v_2, \ldots, v_n$ — объемы соответствующих емкостей, $1 \leqslant v_i \leqslant 10^9$.

Формат выходных данных

Вывести одно число — максимальное количество чатлов, которое смогут добыть Би и Уэф.

Примеры

Пример №1

Стандартный ввод
5 8 9 4 6 3

Стандартный вывод
15

Примечания

На первой стадии в первую емкость Би насыплет 8 мер песка. Далее из первой во вторую пересыплется 8 мер, из второй в третью — 4, из третьей в четвертую — 4, из четвертой в пятую — 3, из пятой на поверхность планеты высыплется 3 меры песка. Таким образом, за первую стадию Би и Уэф получат $8 - 3$, то есть 5 чатлов.

На второй стадии в первую емкость Би насыплет 8 мер песка. Далее из первой во вторую пересыплется 5 мер (так как во второй уже есть 4 меры), из второй в третью — 4, из третьей в четвертую — 4 (в ней окажется уже 5 мер, так как 1 осталась с прошлого раза), из четвертой в пятую — 3, из пятой на поверхность планеты высыплется 3 меры песка. Таким образом, за вторую стадию Би и Уэф получат $8 - 3$, то есть 5 чатлов.

На третьей стадии в первую емкость Би насыплет 5 мер песка, так как с прошлой стадии в первой емкости осталось 3 меры. Далее из первой во вторую пересыплется 4 меры (так как во второй уже есть 5), из второй в третью — 4, из третьей в четвертую — 4 (в ней окажется уже 6 мер, так как 2 осталась с прошлых стадий), из четвертой в пятую — 3, из пятой на поверхность планеты высыплется 3 меры песка. Таким образом, за третью стадию Би и Уэф получат $5 - 3$, то есть 2 чатла.

На четвертой стадии в первую емкость Би насыплет 4 меры песка, так как с прошлой стадии в первой емкости осталось 4 меры. Далее из первой во вторую пересыплется 4 меры (так как во второй уже есть 5), из второй в третью — 4, из третьей в четвертую — 3 (так как в четвертой уже есть 3 меры), из четвертой в пятую — 3, из пятой на поверхность планеты высыплется 3 меры песка. Таким образом, за четвертую стадию Би и Уэф получат $4 - 3$, то есть 1 чатл.

На пятой стадии в первую емкость Би насыплет 4 меры песка. Далее из первой во вторую пересыплется 4 меры, из второй в третью — 3 (в ней осталась 1 мера с прошлой стадии), из третьей в четвертую — 3, из четвертой в пятую — 3, из пятой на поверхность планеты высыплется 3 меры песка. Таким образом, за пятую стадию Би и Уэф получат $4 - 3$, то есть 1 чатл.

На шестой стадии в первую емкость Би насыплет 4 меры песка. Далее из первой во вторую пересыплется 3 меры, из второй в третью — 3, из третьей в четвертую — 3, из четвертой в пятую — 3, из пятой на поверхность планеты высыплется 3 меры песка. Таким образом, за шестую стадию Би и Уэф получат $4 - 3$, то есть 1 чатл.

Далее в первую емкость можно засыпать только 3 меры, что в итоге окажется равно тому, что высыплется на поверхность.

Таким образом, во время этих действий, Би и Уэф добудут $5 + 5 + 2 + 2 + 1 + 1 + 1 = 15$ чатлов.

Пример программы-решения

Ниже представлено решение на языке C++.

C++

```
#include<bits/stdc++.h>
#define sz(a) (int)a.size()
#define pb push_back
```

```cpp
#define all(a) a.begin(), a.end()
#define for0(i, n) for(int i = 0; i < n; i++)
#define for1(i, n) for(int i = 1; i <= n; i++)
#define int long long
using namespace std;
typedef vector<int> vi;
const int INF = 1e18;
const int MOD = 1e9 + 7;
const int LG = 19;
signed main(){
      ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);
    int n, mn = INF;
    cin >> n;
    vi v(n);
    for0(i, n){
        cin >> v[i];
        mn = min(mn, v[i]);
    }
    int ans = 0;
    for0(i, n){
        ans += v[i] - mn;
        if(v[i] == mn){
            break;
        }
    }
    cout << ans << endl;
}
```

Задача 4.2.1.4. Упорядочивание галактик в Тентуре (25 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Немногие знают, что все звезды в Тентуре (обитаемой вселенной) находятся в одной плоскости. Более того, все звезды Тентуры совместно образуют прямоугольник $n \times m$, и в каждой ячейке этого прямоугольника находится ровно одна звезда.

Когда-то давным-давно всеми звездами и планетами Тентуры управлял один правитель — далекий предок Господина ПЖ. Но время шло и, согласно закону Тентуры, управляющий должен разделить свою галактику на две (не обязательно равные) прямоугольные части и отдать их в наследство двум своим старшим потомкам. Делить свою галактику можно только по прямой линии между двумя рядами звезд от края до края галактики. Каждая из этих частей становилась отдельной галактикой. Так поступал каждый правитель, и в итоге все множество звезд Тентуры оказалось разделено на несколько прямоугольных областей. И, что удивительно, всегда выполнялось такое свойство: среди любых четырех звезд, образующих квадрат 2×2 есть хотя бы две из одной галактики. Иными словами, границы галактик всегда образуют T-образные пересечения, и нет ни одного «перекрестка» в котором встречаются

четыре галактики.

Для определенности введем направления в Тентуре «верх-низ» и «лево-право». Тогда любую границу можно характеризовать либо как горизонтальную, либо как вертикальную. Горизонтальная граница делит свою галактику на верхнюю и нижнюю части, а вертикальная — на левую и правую части.

В целях наведения порядка ученые Тентуры решили упорядочить все существующие галактики согласно следующему правилу: для любой разделенной галактики, если она разделена горизонтально, первыми перечисляются галактики сверху, потом галактики снизу; если она разделена вертикально — первыми перечисляются галактики слева, потом галактики справа. И так для каждой из разделенных когда-то галактик. Далее из таких перечислений для меньших галактик складываются по тому же правилу перечисления для старых больших галактик до тех пор, пока не получится перечисления всех галактик Тентуры. Требуется получить такое упорядочение галактик Тентуры по ее карте.

Формат входных данных

В первой строке вводятся через пробел два числа n и m — размеры прямоугольника, который образуют планеты Тентуры.

Далее в следующих n строках, каждая длиной по m символов, без пробелов между ними, указаны планеты. Карта ориентирована таким образом, что направления «верх-низ» и «лево-право» совпадают с теми, что определили ученые Тентуры. Каждая планета задается одной буквой от `a` до `z`. Если две планеты заданы одной буквой, значит, они находятся в одной галактике. Гарантируется, что все галактики удовлетворяют условиям задачи: любая галактика является прямоугольной областью, никакие четыре галактики не сходятся своими углами в одном «перекрестке». Весь набор галактик был получен пошагово, путем разделения на каждом шаге ровно одной галактики на два меньших прямоугольника.

Ограничения: $1 \leqslant n, m \leqslant 50$.

Формат выходных данных

Вывести одну строку, в которой буква каждой галактики карты упомянута ровно один раз, причем все вместе они задают порядок на карте, согласно условию задачи.

Примеры

Стандартный ввод
7 9 zzctooooo zzctooooo zzcdddhbb zzcaaaabb pppaaaaxx pppaaaaxx ppppuuuuu

Стандартный вывод
zcptodhabxu

Пример программы-решения

Ниже представлено решение на языке C++.

C++

```
Авторское решение на C++
#include<bits/stdc++.h>
#define sz(a) (int)a.size()
#define pb push_back
#define all(a) a.begin(), a.end()
#define for0(i, n) for(int i = 0; i < n; i++)
#define for1(i, n) for(int i = 1; i <= n; i++)
#define x first
#define y second
#define int long long
using namespace std;
typedef pair<int, int> pii;
typedef pair<pii, pii> galact;
typedef vector<int> vi;
typedef vector<vector<int> > vvi;
const int INF = 1e18;
const int MOD = 1e9 + 7;
const int LG = 19;

pair<galact, galact> del(galact a, vector<string> &v){
    pair<galact, galact> res = {{{-1, -1},{-1, -1}}, {{-1, -1}, {-1,
    ↪  -1}} };
    int x1 = a.x.x;
    int x2 = a.y.x;
    int y1 = a.x.y;
    int y2 = a.y.y;
    for(int i = x1; i < x2; i++){
        bool ok = 1;
        for(int j = y1; j <= y2; j++){
            if(v[i][j] == v[i + 1][j]){
                ok = 0;
            }
        }
        if(ok){
            return {{{x1, y1}, {i, y2}}, {{i + 1, y1}, {x2, y2}}};
        }
    }
    for(int j = y1; j < y2; j++){
        bool ok = 1;
        for(int i = x1; i <= x2; i++){
            if(v[i][j] == v[i][j + 1]){
                ok = 0;
            }
        }
        if(ok){
            return {{{x1, y1}, {x2, j}}, {{x1, j + 1}, {x2, y2}}};
        }
    }
    return res;
```

```
}

string order(galact a, vector<string> &v){
    string res;
    pair<galact, galact> t = del(a, v);
    if(t.x.x.x == -1){
        res = v[a.x.x][a.x.y];
    }
    else{
        res = order(t.x, v) + order(t.y, v);
    }
    return res;
}

signed main(){
       ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);
    int n, m;
    cin >> n >> m;
    vector<string> v(n);
    for0(i, n){
        cin >> v[i];
    }
    string ans = order({{0, 0}, {n - 1, m - 1}}, v);
    cout << ans << endl;
}
```

Задача 4.2.1.5. Пролетая мимо Плюка (30 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 11 с.

Ограничение по памяти: 64 Мбайт.

Условие

Автоматический зонд стартует из некоторой точки на координатной плоскости и бесконечно долго выполняет записанную в нем программу перемещений. Каждое перемещение — это сдвиг на одну позицию по координате X или по координате Y. Более точно смещение в сторону увеличения координаты X обозначается символом R (вправо), смещение в сторону уменьшения координаты X обозначается символом L (влево), смещение в сторону увеличения координаты Y обозначается символом U (вверх), смещение в сторону уменьшения координаты Y обозначается символом D (вниз). Программа перемещений состоит не более, чем из 10^5 символов, и после того, как зонд ее выполнит, он начинает выполнять ее заново. Известно, что в процессе полета зонд как минимум один раз заканчивает и начинает заново свою программу в точке $(0, 0)$.

Известны координаты планеты Плюк — целые числа XP и YP. Требуется определить минимальное расстояние до Плюка, на котором зонд окажется, находясь в какой-либо целочисленной точке плоскости в ходе своего полета.

Формат входных данных

В первой строке содержится одно натуральное число n — количество команд в программе перемещения зонда, $1 \leqslant n \leqslant 10^5$.

Во второй строке содержится n символов из множества L, R, D, U — программа перемещения.

В третьей строке содержатся координаты планеты Плюк XP, YP — целые числа, по абсолютному значению не превосходящие 10^6. В данной задаче считать планету Плюк точкой плоскости.

Формат выходных данных

Вывести одно число — наименьшее расстояние, на котором окажется зонд в процессе бесконечного выполнения указанной программы перемещений. Ответ вывести с точностью не менее шести знаков после десятичной точки.

Примеры

Пример №1

Стандартный ввод
7 RUURDDD 6 3

Стандартный вывод
2.8284271

Примечания

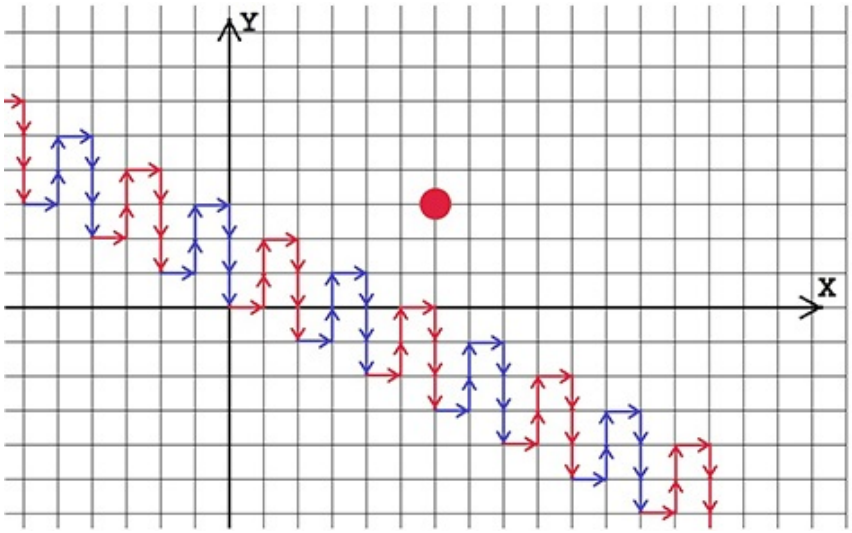

Рис. 4.2.1

На рис. 4.2.1 представлен путь зонда для примера из условия. Чередующимся красным и синим показаны отдельные наборы выполненных команд из программы перемещений.

Видно, что зонд один раз закончил программу в точке (0, 0) и снова начал выполнять ее далее. Видно, что в точке (4, 1) зонд подлетает к Плюку на расстояние $\sqrt{8}$).

Пример программы-решения

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define sz(a) (int)a.size()
#define pb push_back
#define all(a) a.begin(), a.end()
#define for0(i, n) for(int i = 0; i < n; i++)
#define for1(i, n) for(int i = 1; i <= n; i++)
#define x first
#define y second
#define int long long
using namespace std;
typedef pair<int, int> pii;
typedef vector<int> vi;
typedef vector<vector<int> > vvi;
const int INF = 1e18;
const int MOD = 1e9 + 7;
const int LG = 19;

int sqdist(pii a, pii b){
    return ((a.x - b.x)*(a.x - b.x) + (a.y - b.y)*(a.y - b.y));
}

pii pos(pii t, pii v, int M){
    return {t.x + v.x * M, t.y + v.y * M};
}

signed main(){
       ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    map<char, pii> dxy = {{'L',{-1, 0}}, {'R', {1, 0}}, {'U', {0, 1}},
     ↪  {'D', {0, -1}}};
    int n;
    cin >> n;
    string s;
    cin >> s;
    pii P;
    cin >> P.x >> P.y;
    pii t = {0, 0};
    for(auto c : s){
        t.x += dxy[c].x;
        t.y += dxy[c].y;
    }
    int mn = INF;
    if(t.x == 0 && t.y == 0){
        t = {0, 0};
        for(auto c : s){
```

```
            t.x += dxy[c].x;
            t.y += dxy[c].y;
            mn = min(mn, sqdist(t, P));
        }
        cout.precision(7);
        double ans = mn;
        cout << fixed << sqrt(ans);
        return 0;
    }


    pii z = {0, 0};
    for(auto c : s){
        z.x += dxy[c].x;
        z.y += dxy[c].y;
        int L = -1, R = 1;
        pii tl = pos(z, t, L);
        while(abs(tl. x) < 1e7 && abs(tl.y) < 1e7){
            L *= 2;
            tl = pos(z, t, L);
        }
        pii tr = pos(z, t, R);
        while(abs(tr. x) < 1e7 && abs(tr.y) < 1e7){
            R *= 2;
            tr = pos(z, t, R);
        }

        while(R - L > 2){
            int M1 = (2 * L + R) / 3;
            int M2 = (L + 2 * R) / 3;
            pii p1 = pos(z, t, M1);
            pii p2 = pos(z, t, M2);
            if(sqdist(p1, P) < sqdist(p2, P)){
                R = M2;
            }
            else{
                L = M1;
            }
        }
        pii p1 = pos(z, t, L);
        pii p2 = pos(z, t, (R + L) / 2);
        pii p3 = pos(z, t, R);
        mn = min({mn, sqdist(p1, P), sqdist(p2, P), sqdist(p3, P)});
    }
    cout.precision(7);
    double ans = mn;
    cout << fixed << sqrt(ans);
}
```

4.2.2. Физика. 8–9 классы

Задача 4.2.2.1. Гордый буревестник (15 баллов)

Темы: кинематика, системы отсчета, относительность механического движения, принцип относительности Галилея, средняя и мгновенная скорость тела при движении, расчет пути и времени движения.

Условие

Согласно прогнозу погоды, в ближайшие несколько часов ожидается сильный ветер, дующий преимущественно в северо-восточном направлении. Ожидаемые порывы ветра будут достигать скорости до 54 км/ч. Поэтому запланированный перелет беспилотного транспортного средства вынужденно будет осуществляться в сложных метеоусловиях.

Неблагоприятные погодные условия могут потребовать существенного увеличения заряда аккумулятора для повышения запаса хода (длительности полета) беспилотника по сравнению с зарядом, способным обеспечить полет на ту же дистанцию в безветренную погоду.

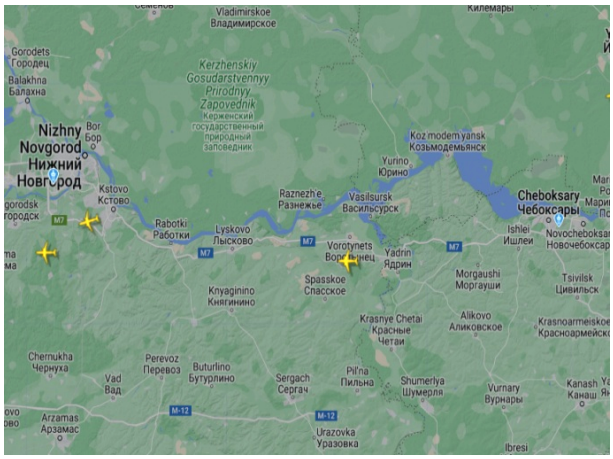

Рис. 4.2.2

На сколько полных минут полета необходимо увеличить заряд аккумулятора беспилотника, чтобы надежно гарантировать доставку груза (в ожидаемых неблагоприятных метеоусловиях) из пункта А в пункт Б, расположенный в 60 км западнее пункта А, если крейсерская скорость беспилотника в условиях полного штиля составляет 30 м/с?

Решение

Крейсерская скорость беспилотника ($V_{кр}$ = 30 м/с) в условиях полного штиля — это скорость его движения относительно воздушной среды. Относительно же поверхности Земли скорость беспилотника будет складываться из скоростей самого беспилотника и скорости движения воздушных масс (т. е. скорости ветра). При этом (согласно заданию) требуется, чтобы беспилотник двигался в строго западном направлении.

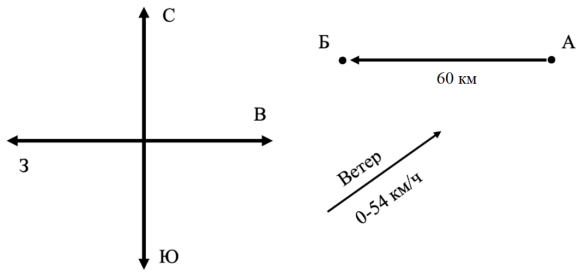

Рис. 4.2.3

Рассмотрим самый негативный (из возможных) сценарий, когда сила (скорость) ветра $V_{\text{в}}$ достигает максимальных значений (до 54 км/ч) на протяжении всего полета.

Скорость ветра, при переводе в метры в секунду, составит 15 м/с.

$$V_{\text{в}} = 54\frac{\text{км}}{\text{ч}} = \frac{54\ \text{км/ч}}{3{,}6\ \text{км/м с/ч}} = 15\ \text{м/с}.$$

При движении из А в Б беспилотник должен двигаться на запад, т. е. в направлении, строго противоположном оси X. Другими словами, скорость вдоль оси Y должна отсутствовать: $V_y = 0$.

Из чего следует, что $V_{\text{кр}} \cdot \sin\alpha = V_{\text{В}} \cdot \sin 45^\circ$ (см. пояснительный рис. 4.2.4).

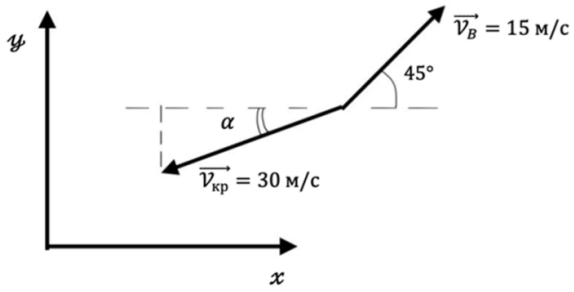

Рис. 4.2.4

Согласно этому, угол α — угол, под которым следует подержать ориентировку, можно получить из следующего соотношения:

$$\sin\alpha = \frac{V_B}{V_{\text{кр}}} \cdot \sin 45^\circ = \frac{15}{30} \cdot \frac{\sqrt{2}}{2} \approx 0{,}35355.$$

Тогда скорость вдоль оси X для беспилотника, сориентированного таким образом, составит:

$$V_x = -V_{\text{кр}} \cdot \cos\alpha + V_B \cdot \cos 45^\circ.$$

Учитывая прямое следствие из основного тригонометрического тождества,

$$\cos\alpha = \sqrt{1 - \sin^2\alpha} = \sqrt{1 - (0{,}35355)^2} = \sqrt{0{,}875} \approx 0{,}93541.$$

Получим итоговое значение для модуля скорости вдоль оси X:

$$|V_x| = V_{\text{кр}} \cdot \sqrt{1 - \left(\frac{V_B}{V_{\text{кр}}} \cdot \sin 45^\circ\right)^2} - V_B \cdot \cos 45^\circ;$$

$$|V_x| = 30 \text{ м/с} \cdot \sqrt{1 - \left(\frac{15}{30} \cdot \frac{\sqrt{2}}{2}\right)^2} - 15 \text{ м/с} \cdot \frac{\sqrt{2}}{2} \approx 17{,}456 \text{ м/с}.$$

Время движения из A в B при сильном ветре составит:

$$t_{\text{В}} = \frac{S}{V_x} = \frac{60 \cdot 10^3 \text{ м}}{17{,}456 \text{ м/с}} \approx 3437{,}21 \text{ с}.$$

При полете в безветренную погоду потребовалось бы:

$$t_0 = \frac{S}{V_{\text{кр}}} = \frac{60 \cdot 10^3 \text{ м}}{30 \text{ м/с}} = 2\,000 \text{ с}.$$

Таким образом, необходимо закладывать дополнительный запас хода (времени полета) в расчете на неблагоприятные погодные условия:

$$\Delta t = t_{\text{В}} - t_0 \approx 3437{,}21 - 2\,000 = 1\,437{,}21 \text{ с} \approx 23{,}95 \text{ мин} \approx 24 \text{ мин}.$$

Примечание. Так как речь в задаче шла о запасе хода, то округление в этом случае нужно делать не по правилам округления, а по логике происходящего, т. е. до целого значения и в большую сторону.

Ответ: на 24 мин.

Критерии оценивания

Критерий	**Баллы**
Сформулирована мысль об относительности механического движения в условиях ветра	2
Сделаны пояснительные рисунки, отображающие влияние ветра на скорость полета беспилотника относительно Земли	2
Получено аналитическое выражение для вычисления скорости движения беспилотника относительно поверхности Земли с учетом действия ветра	4
Вычислено численное значение скорости беспилотника (при движении на запад из пункта А в пункт Б) с учетом ветра	3
Вычислено время движения из пункта А в пункт Б с учетом ветра	2
Вычислено время движения из пункта А в пункт Б в безветренную погоду	1

Критерий	Баллы
Определено, на сколько полных минут полета необходимо увеличить заряд аккумулятора по сравнению с полетом в безветренную погоду	1
Всего	**15**

Задача 4.2.2.2. Наш след на Кэмел-трофи (12 баллов)

Темы: основы молекулярно-кинетической теории строения вещества и взаимодействие тел, равнодействующая сил, давление.

Условие

Для увеличения проходимости на сложном грунтово-каменистом участке дороги во всех четырех шинах транспортного средства снизили давление.

Автомобильного манометра с собой не было, поэтому совершая данную процедуру, приходилось ориентироваться на глаз. Внутренний голос опытного водителя подсказывал, что судить о величине остаточного давления в шинах можно было по возрастающему размеру пятна контакта шин с дорожным покрытием.

Рис. 4.2.5

Стравливание давления в каждом из колес прекращали в тот момент, когда протяженность пятна контакта (A) для данного колеса достигала отметки в 40 см (разумеется, примерно). Пренебрегая жесткостью автомобильных шин и принимая ускорение свободного падения равным 10 м/с^2, оцените абсолютную величину остаточного давления в шинах автомобиля, если снаряженная масса автомобиля составляла 2,4 т, а ширина (B) используемых покрышек была 25 см. Ответ выразите в единицах атмосферного давления с точностью до одной десятой. Примите, что давление в 100 кПа равно атмосферному.

Решение

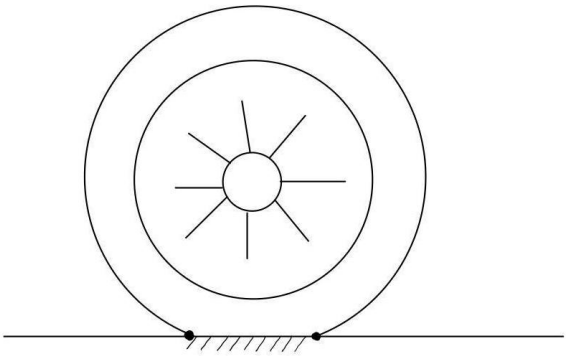

Рис. 4.2.6

Пятно контакта возникает в результате деформации шины колеса (см. пояснительный рис. 4.2.6) под действием части веса автомобиля, приходящегося на данное колесо. На одно колесо (в среднем) приходится 1/4 веса автомобиля.

Эта нагрузка уравновешивается упругой реакцией, которая зависит от внутреннего давления в шинах, жесткости самих шин и той степени деформации, которую они испытывают. Если не учитывать жесткость материала шин, то можно считать, что деформацию шин ограничивает только избыточное внутреннее давление газа в шинах по сравнению с атмосферным давлением снаружи.

Учитывая все вышесказанное, можно записать уравнение баланса сил вдоль вертикальной оси:

$$\frac{1}{4}N = \Delta P \cdot S,$$

где N — вес автомобиля; ΔP — избыточное давление (по сравнению с внешним (атмосферным)); S — площадь пятна контакта, приходящегося на каждое колесо (в среднем).

При известной величине пятна контакта избыточное давление в шинах составит:

$$\Delta P = \frac{N}{4 \cdot S} = \frac{mg}{4 \cdot S} = \frac{2{,}4 \cdot 10^3 \text{ кг} \cdot 10 \text{ м/с}^2}{4 \cdot 40 \cdot 25 \cdot 10^{-4} \text{ м}^2} = 6 \cdot 10^4 \text{ Па} \approx 0{,}6 \text{ атм}.$$

В итоге абсолютное давление в шинах составило:

$$P_{\text{abs}} = \Delta P + P_{\text{атм}} = 0{,}6 + 1 = 1{,}6 \text{ атм}.$$

Ответ: 1,6 атм.

Критерии оценивания

Критерий	**Баллы**
Сформулированы причины возникновения пятна контакта	2
Сделаны пояснительные рисунки	2

Критерий	Баллы
Упомянуто значение нагрузки, приходящейся на каждое колесо (в среднем)	1
Записано уравнение, связывающее избыток внутреннего давления, площадь пятна контакта и уровень нагрузки, приходящейся на колесо	3
Определено численное значение избыточного давления в шинах (переведены единицы измерения из паскалей в единицы атмосферного давления)	2
Вычислено абсолютное давление в шинах	2
Всего	**12**

Задача 4.2.2.3. Нянь бесстрашный укротитель и варенья поглотитель! (25 баллов)

Темы: электрические явления, модель твердых тел, тепловое расширение, электрический ток, удельное электросопротивление, тепловое действие электрического тока, закон Джоуля – Ленца.

Условие

Провода, использованные в основных электрических узлах на борту квадрокоптера, изготовлены из технически чистой меди. Совокупная длина медных проводов (при температуре 20 °C) составляла 100 м.

Про материал проводов известно, что:

- Удельное электросопротивление чистой электротехнической меди имеет температурную зависимость вида:

$$\rho_t = \rho_{20} \cdot [1 + \alpha(t - 20)],$$

где

  - ρ_t — удельное электросопротивления меди при некоторой температуре t;
  - ρ_{20} — удельное электросопротивление меди при температуре 20 °C (равное 0,0172 Ом $\cdot$ мм^2/м), принятое в качестве базового уровня;
  - α — температурный коэффициент, отражающий скорость изменения удельного электросопротивления, составляет 0,004 °C^{-1};
  - t — температура (°C).

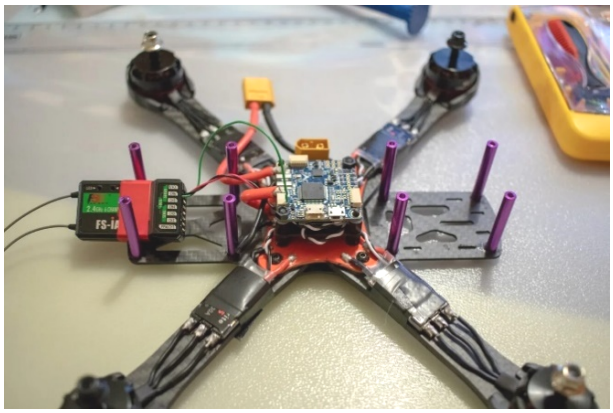

Рис. 4.2.7

- Тепловое расширение чистой электротехнической меди описывается уравнением вида:

$$r_t = r_{20} \cdot [1 + \beta(t - 20)]\,,$$

где

  - r_t — геометрический размер проводника при некоторой температуре t;
  - r_{20} — геометрический размер проводника при температуре 20 °C, принятой в качестве базового (опорного) значения;
  - β — коэффициент теплового расширения, равный $16{,}8 \cdot 10^{-6}$ °C^{-1};
  - t — температура (°C).

Вопрос: на сколько процентов будут различаться уровни электротепловых потерь, связанных с изменением омического сопротивления проводов в условиях тридцатиградусной жары, по сравнению с эксплуатацией квадрокоптера в тридцатиградусный мороз при неизменном значении силы тока, подаваемым с бортового источника питания?

Решение

Обозначим рассматриваемые температуры, при которых нужно сравнить уровни электротепловых потерь, следующим образом:

$$t_1 = +30\ ^\circ\text{C}, \quad t_2 = -30\ ^\circ\text{C}.$$

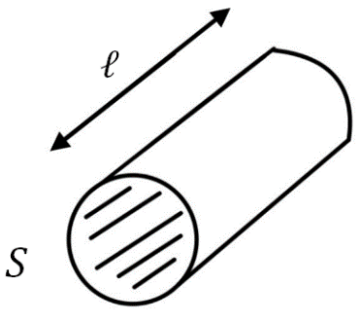

Рис. 4.2.8

Общее изменение сопротивления медных электрических проводов при изменении эксплуатационной температуры происходит под влиянием не только изменения удельного электросопротивления, но также за счет изменения длины и площади сечения проводов. Это связано напрямую с тем, что общее сопротивление проводника связано с удельным сопротивлением формулой:

$$R = \rho \cdot \frac{l}{S}.$$

За счет теплового расширения у медных проводов увеличится не только длина, но и возрастет площадь сечения, причем относительное изменение площади сечения будет пропорциональным квадрату отношения диаметров:

$$\frac{S_2}{S_1} = \left(\frac{d_2}{d_1}\right)^2.$$

Это связано с тем, что площадь сечения связана с диаметром формулой:

$$S = \frac{\pi d^2}{4}.$$

При однородном тепловом расширении относительное изменение диаметра проводника происходит в той же мере, что и относительное изменение длины проводника. Поэтому относительное изменение сечения можно выразить иначе, а именно:

$$\frac{S_2}{S_1} = \left(\frac{d_2}{d_1}\right)^2 = \left(\frac{l_2}{l_1}\right)^2.$$

В результате отношение электросопротивлений проводников можно выразить как:

$$\frac{R_1}{R_2} = \frac{\rho_1 \cdot l_1/S_1}{\rho_2 \cdot l_2/S_2} = \frac{\rho_1}{\rho_2} \cdot \frac{l_1}{l_2} \cdot \frac{S_2}{S_1} = \frac{\rho_1}{\rho_2} \cdot \frac{l_1}{l_2} \cdot \left(\frac{l_2}{l_1}\right)^2 = \frac{\rho_1}{\rho_2} \cdot \frac{l_2}{l_1}.$$

Таким образом, чтобы найти отношение электросопротивлений проводников при температурах: $t_1 = +30$ °C и $t_2 = -30$ °C, нужно вычислить отношения удельных электросопротивлений $\frac{\rho_1}{\rho_2}$ и отношение линейных размеров $\frac{l_2}{l_1}$ при соответствующих температурах.

Удельное электросопротивление чистой электротехнической меди имеет температурную зависимость вида:

$$\rho_t = 0{,}0172 \cdot [1 + 0{,}004 \cdot (t - 20)]\,.$$

Следовательно:

$$\frac{\rho_1}{\rho_2} = \frac{0{,}0172 \cdot [1 + 0{,}004 \cdot (+30 - 20)]}{0{,}0172 \cdot [1 + 0{,}004 \cdot (-30 - 20)]} = \frac{1 + 0{,}004 \cdot 10}{1 - 0{,}004 \cdot 50} = \frac{1{,}04}{0{,}8} = 1{,}3.$$

Тепловое расширение чистой электротехнической меди описывается уравнением вида:

$$r_t = r_{20} \cdot [1 + \beta(t - 20)]\,,$$

где β — коэффициент теплового расширения, равный $16{,}8 \cdot 10^{-6}$ °C^{-1}.

Здесь необходимо учесть, что r_{20} — исходный геометрический размер медного проводника при температуре, принятой в качестве базового (опорного) значения. Этот размер может иметь разные значения, в отличие от удельного сопротивления, имеющего вполне конкретную величину!

Следовательно, отношение линейных размеров $\dfrac{l_2}{l_1}$ при соответствующих температурах будет равно:

$$\frac{l_2}{l_1} = \frac{l_{20} \cdot [1 + \beta(t_2 - 20)]}{l_{20} \cdot [1 + \beta(t_1 - 20)]} = \frac{1 + 16{,}8 \cdot 10^{-6} \cdot (-30 - 20)}{1 + 16{,}8 \cdot 10^{-6} \cdot (30 - 20)} =$$
$$= \frac{1 - 16{,}8 \cdot 10^{-6} \cdot 50}{1 + 16{,}8 \cdot 10^{-6} \cdot 10} \approx \frac{0{,}99916}{1{,}000168} \approx 0{,}998992.$$

В результате отношение электросопротивлений проводников при изменении температуры составит:

$$\frac{R_1}{R_2} = \frac{\rho_1}{\rho_2} \cdot \frac{l_2}{l_1} = 1{,}3 \cdot 0{,}998992 \approx 1{,}2987.$$

Согласно закону Джоуля – Ленца, мощность нагрева от электрического тока можно выразить через силу тока, напряжение и электросопротивление следующими способами:

$$W = IU = \frac{U^2}{R} = I^2 R.$$

При неизменном значении тока удобнее всего пользоваться последним выражением для тепловой мощности. В итоге отношение тепловых потерь при разных температурах будет выражаться через соотношение сопротивлений:

$$\frac{W_1}{W_2} = \frac{I_1^2 R_1}{I_2^2 R_2} = \frac{R_1}{R_2} = \frac{\rho_1}{\rho_2} \cdot \frac{l_2}{l_1} = 1{,}3 \cdot 0{,}998992 \approx 1{,}2987.$$

Данное относительное изменение эквивалентно относительному приросту величины теплопотерь на $29{,}87\%$.

Ответ: увеличится на 29,87%.

Критерии оценивания

Критерий	**Баллы**
Сформулирована мысль о причинах возникновения потерь (закон Джоуля – Ленца)	2
Показана роль и характер влияния факторов, отвечающих за изменение величины электросопротивления	3
Вычислено значение удельного электросопротивления при разных температурах	4
Вычислено или учтено относительное изменение длины проводов в результате теплового расширения	4
Вычислено или учтено относительное изменение сечения проводов в результате теплового расширения	4

Критерий	Баллы
Определено относительное изменение общего омического электросопротивления с ростом температуры с учетом изменения удельного электросопротивления и теплового расширения	4
Вычислено относительное изменение величины тепловых потерь (с учетом закона Джоуля – Ленца при неизменной величине тока)	3
Выражено в процентах относительное изменение тепловых потерь	1
Всего	**25**

Задача 4.2.2.4. Болотоход по кличке «Проходимец» (28 баллов)

Темы: механическое движение, Законы Ньютона, принцип суперпозиции сил, сила тяжести, сила упругости, сила трения, модель абсолютно твердого тела, центр тяжести, момент силы, условия равновесия твердого тела.

Условие

Грунтозацепы гусеничных траков вездехода обеспечивают такой уровень сцепления с грунтом, что вездеход способен тянуть буксировочный трос с усилием, в три раза превышающим его собственный вес. Максимальный крутящий момент, передаваемый от силовой установки на ведущие колеса (при включенной нижней передаче), таков, что соответствующее суммарное сдвиговое усилие, передаваемое от ведущих колес на гусеницы, в пять раз превышает вес вездехода, находящегося в состоянии покоя. Все четыре колесные оси вездехода, когда он стоит на ровной горизонтальной поверхности, нагружены равномерно. Оси расположены на расстоянии 1 м друг от друга (см. фотографию рис. 4.2.9).

Рис. 4.2.9

Определите предельно допустимую высоту в расположении центра тяжести вездехода для предотвращения его возможного опрокидывания при преодолении максимально крутых подъемов, на которые он гипотетически способен взбираться. Грунт считайте всюду однородным, одинаково плотным и достаточно жестким (т. е. вездеход не вязнет и не закапывается в нем).

Решение

Согласно условию задачи и данному в нем описанию вездехода, можно сделать следующие логически непротиворечивые выводы:

1. Центр тяжести вездехода (в продольном направлении) располагается между второй и третьей осями и строго посередине вездехода (см. пояснительный рис.), т. е. на расстоянии по 1,5 м от переднего и заднего краев пятна контакта. Это следует из того, что у вездехода, стоящего на месте и в горизонтальном положении, оси нагружены равномерно, а расстояния между осями равны 1 м, при том, что межосевых пролетов всего три, а это означает, что l — длина пятна контакта гусеницы с грунтом — составляет 3 м.

2. Коэффициент трения (эффективное значение) между гусеницами и грунтом достигает значения $\mu = 3$. Это прямое следствие из того, что максимальное усилие, с которым вездеход может тянуть буксировочный трос, составляет три его веса:

$$F_{\text{тяги (буксир)}} = F_{\text{трения}} = \mu N = \mu mg.$$

Так как $F_{\text{тяги (буксир)}} = 3mg \implies \mu = 3$.

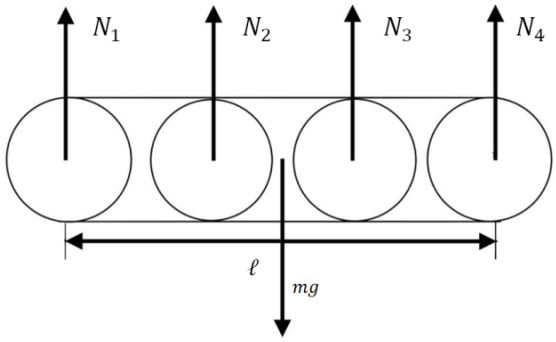

Рис. 4.2.10

3. Ограничение по тяге определяется исключительно трением (сцеплением с грунтом), т. к. запас по крутящему моменту в 5 раз превышает вес вездехода. При неограниченном уровне сцепления $F_{\text{тяги}}$ была бы равна $5mg$. Но в условиях имеющегося уровня сцепления реально воплощаемая сила тяги $F_{\text{тяги}}$ будет ограничена значением, равным μN, а не $5mg$.

4. Максимально возможный угол подъема определяется через соотношение:

$$\operatorname{tg}\alpha = \mu = 3.$$

Это непосредственно следует из того, что при движении под углом α к горизонту сила реакции опоры N определяется из соотношения:

$$N = mg\cos\alpha.$$

При этом сила тяги $F_{\text{тяги}}$, реализуемая вездеходом, должна быть не меньше, чем $mg\sin\alpha$ (проекция силы тяжести на наклонную плоскость), иначе вездеход не сможет двигаться вверх. Из этого следует, что максимально возможный угол подъема определяется через соотношение:

$$\mu \cdot mg \cdot \cos\alpha = mg \cdot \sin\alpha.$$

В результате тангенс предельного угла атаки (максимально возможный угол подъема) составляет: $\operatorname{tg}\alpha = \mu = 3$.

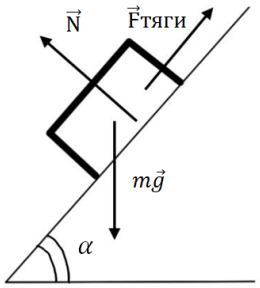

Рис. 4.2.11

5. При подъеме вездеход сохраняет устойчивость (не опрокидывается), если вектор силы тяжести пересекает пятно контакта гусениц с грунтом, а не выходит за его рамки.

   В предельном случае вектор силы тяжести может проходить через крайнюю точку (нижнюю границу) пятна контакта (см. точку A на пояснительном рис. 4.2.12).

   В результате при подъеме с максимально возможным углом атаки положение центра тяжести вездехода должно быть не выше точки O, т. е. располагаться на высоте, относительно поверхности грунта, не большей, чем величина отрезка OH.

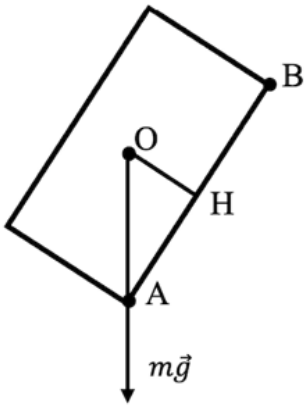

Рис. 4.2.12

Значение OH легко найти из прямоугольного треугольника AOH:

$$OH = \frac{AH}{\operatorname{tg}\alpha} = \frac{AB/2}{\operatorname{tg}\alpha} = \frac{3\,\text{м}/2}{3} = 0{,}5\,\text{м}.$$

Ответ: не более 0,5 м от поверхности.

Критерии оценивания

Критерий	**Баллы**
Сделан вывод о том, где находится центр тяжести в продольном направлении	3
Определено значение для протяженности пятна контакта гусениц вездехода	3
Вычислено значение эффективного коэффициента трения	5
Показано (объяснено), что именно является лимитирующим (сила трения или мощность (крутящий момент) двигателя на способности вездехода преодолевать крутые подъемы	5
Вычислен тангенс (или сам угол) предельно крутого подъема для данного вездехода	4
Сформулировано условие равновесия (или условия опрокидывания) вездехода при движении по наклонной плоскости	5
Вычислено предельное значение для высоты, определяющей точку расположения центра тяжести вездехода	3
Всего	**28**

Задача 4.2.2.5. Господа, соблюдайте дистанцию! (20 баллов)

Темы: динамика, неравномерное прямолинейное движение, ускорение, второй закон Ньютона, импульс тела, кинетическая энергия, законы сохранения.

Условие

Адаптивный круиз-контроль (далее АКК) автоматически поддерживает заданную водителем скорость и варьирует ее в зависимости от текущей дистанции до попутного транспортного средства. Специальный сенсор периодически подает сигнал, который, отразившись от впереди идущего транспорта, возвращается обратно. Исходя из полученных данных, процессор АКК оценивает текущее расстояние, рассчитывает динамику его изменения, после чего принимает решение, нужно ли автомобилю ускориться или притормозить. Для оценки ситуации на дороге и принятия соответствующего решения системе АКК требуется промежуток времени (не менее 0,2 с). Еще около 0,3 с потребуется на приведение тормозной системы в режим интенсивного (аварийного) торможения (время на нагнетание и передачу давления к тормозным цилиндрам, осуществляющим прижим тормозных колодок к тормозным дискам колес).

Определите, какую минимально безопасную дистанцию между автомобилями должен поддерживать АКК, если движение автомобиля с системой АКК происходит в потоке, имеющем скорость 90 км/ч. Примите уровень сцепления шин автомобилей с дорогой таковым, что эффективный коэффициент трения (в зависимости от качества используемых шин) находится в диапазоне от 0,6 до 0,8. Считайте что движение происходит на горизонтальном ровном участке, а ускорение свободного падения равно 10 м/с^2. Ответ выразите в метрах (округлив до целых значений).

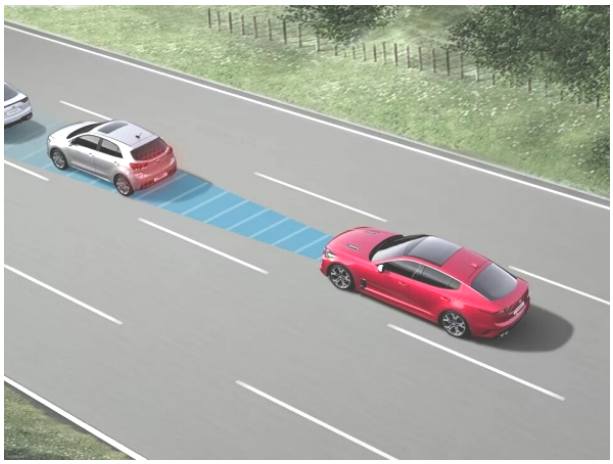

Рис. 4.2.13

Решение

Согласно данным о скорости работы АКК и срабатывания тормозной системы, автомобиль, движущийся следом за впереди идущим автомобилем, совершающим экстренное торможение, начнет совершать торможение в полную силу спустя 0,5 с.

Наиболее неблагоприятная ситуация может реализоваться в том случае, если первый автомобиль оттормаживается лучше, чем следующий за ним (т. е. коэффициент трения у шин первого автомобиля достигает значения 0,8, а у второго — только 0,6). Следовательно, угол наклона (β) на графике изменения скорости от времени будет заметно меньше $\angle\alpha$, т. к. тангенсы этих углов наклона на этих участках графиков численно равны соответствующим величинам ускорений при торможении.

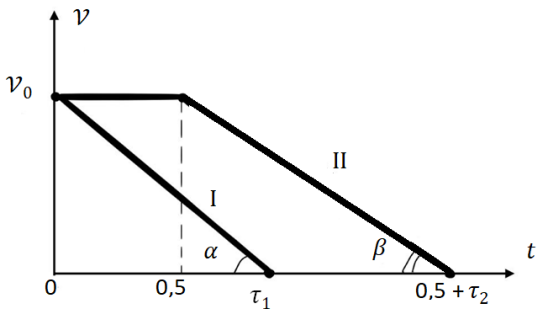

Рис. 4.2.14

В момент времени $t = 0$ оба автомобиля двигались со скоростью 90 км/ч, что соответствует 25 м/с ($90/3{,}6 = 25$).

Ускорения, развиваемые при активной фазе торможения, определяются уровнем сцепления шин с дорожным полотном и величиной ускорения свободного падения:

$$a_i = \frac{F_{\text{тр}}}{m} = \frac{\mu_i N}{m} = \frac{\mu_i m g}{m} = \mu_i g.$$

Длительность фазы активного торможения (до полной остановки) зависит от скорости потока (т. е. от скорости на момент начала торможения) и от величины

развиваемого ускорения при торможении:

$$\tau_i = \frac{\Delta V}{a_i} = \frac{V_0 - 0}{a_i} = \frac{V_0}{a_i}.$$

Тормозной путь, включающий временную задержку до начала активной фазы торможения, определяется площадью под соответствующими графиками. Соответственно, для впереди идущего (первого) автомобиля:

$$S_1 = V_0 \cdot \frac{\tau_1}{2} = \frac{V_0^2}{2\mu_1 g} = \frac{25^2}{2 \cdot 0{,}8 \cdot 10} \approx 39{,}0625\,\text{м}.$$

Для автомобиля (II), движущегося сзади:

$$S_2 = V_0 \cdot 0{,}5 + \frac{V_0 \cdot \tau_2}{2} = V_0 \cdot 0{,}5 + \frac{V_0^2}{2\mu_2 g} =$$
$$= 25 \cdot 0{,}5 + \frac{25^2}{2 \cdot 0{,}6 \cdot 10} \approx 64{,}5833\,\text{м}.$$

Разность тормозных путей определяет тот необходимый минимум безопасной дистанции между автомобилями, который необходимо поддерживать системе АКК во избежание столкновения.

Т. е. необходимая (безопасная) дистанция при движении в потоке (со скоростью 90 км/ч) должна составлять не менее:

$$S_{\text{мин}} \geqslant \Delta S = S_2 - S_1 \approx 64{,}5833\,\text{м} - 39{,}0625\,\text{м} \approx 25{,}5\,\text{м}.$$

Погрешность ответа

При округлении полученного значения до целого значения необходимо округлять непременно в бОльшую сторону (без оглядки на правила округления!), т. к. здесь речь идет о минимально необходимой (безопасной) дистанции. Она однозначно должна быть не меньше найденного значения!

Ответ: $S_{\min} \geqslant 26$ м.

Критерии оценивания

Критерий	**Баллы**
Разъяснена суть и причина для необходимости поддерживать дистанцию между машинами	3
Сформулирована мысль о том, что является благоприятным и неблагоприятным сценарием развития аварийной ситуации	2
Построен график или сделано текстовое пояснение, отражающие характер изменения скоростей автомобилей	4
Определены достижимые уровнем сцепления значения ускорений при торможении	3

Критерий	Баллы
Вычислены времена ускоренного движения, необходимые для подавления скорости до полной остановки	2
Вычислен тормозной путь для движущейся впереди машины	2
Вычислен тормозной путь автомобиля, идущего сзади и снабженного системой АКК	3
Определено значение безопасной дистанции по разности тормозных путей при неблагоприятном сценарии развития аварийной ситуации на дороге	1
Всего	**20**

4.2.3. Физика. 10–11 классы

Задача 4.2.3.1. Тормоза придумал инженер, а не трус! (23 балла)

Темы: динамика, неравномерное прямолинейное движение, ускорение, второй закон Ньютона, импульс тела, кинетическая энергия, импульс силы, механическая работа, работа сил трения, законы сохранения, момент силы относительно оси вращения, условия равновесия твердого тела.

Условие

Колесная база (межосевое расстояние) автомобиля составляет 2,5 м. Центр тяжести автомобиля находится на высоте 50 см от поверхности дорожного полотна и немного смещен в сторону передней оси. В результате такого смещения 60% веса автомобиля приходится на переднюю ось и только 40% — на заднюю.

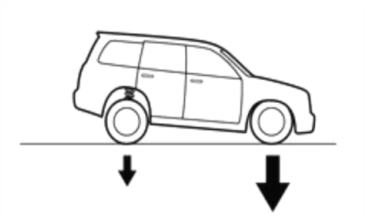

Рис. 4.2.15

Все четыре колеса автомобиля «обуты» в шины одного типа и размера и имеют одну и ту же степень износа. Автомобиль снабжен дисковыми тормозами (вентилируемыми спереди и обычными сзади). Диаметр тормозных дисков при этом одинаков. Тормозные колодки также ничем не отличаются. Так как при торможении на долю передних тормозов обычно приходится существенно более высокая нагрузка, дополнительная вентиляция передних дисков сделана исключительно для предотвращения их преждевременного перегрева.

Рис. 4.2.16

В попытке реализовать максимально эффективное торможение инженер-механик занимался настройкой баланса тормозов. Наилучший результат был достигнут при некотором (вполне конкретном) соотношении усилий, которые создавали тормозные цилиндры, обеспечивая соответствующий прижим тормозных колодок к передним и задним тормозным дискам. При этом полная остановка автомобиля, изначально движущегося со скоростью 108 км/ч, была осуществлена за 45 м пути.

Вопрос: какое соотношение прижимных усилий, переданных на передние и задние тормозные диски, позволило осуществить автомобилю максимально эффективное торможение?

Решение

Когда автомобиль покоится или движется равномерно по горизонтальной поверхности, его оси нагружены соответственно с силой $0{,}6N$ и $0{,}4N$ (где N — вес автомобиля).

При торможении передняя ось автомобиля будет дополнительно перегружена (за счет перераспределения веса). При этом вес, приходящийся на переднюю ось $N_{\text{п}}$, увеличится ровно на столько, на сколько снизится нагрузка $N_{\text{з}}$, приходящаяся на заднюю ось. В сумме эти две силы остаются равными весу автомобиля.

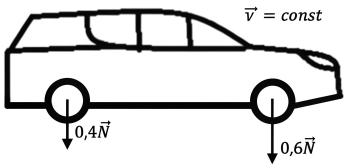

Рис. 4.2.17

Рассмотрим динамику торможения автомобиля, которому нужно погасить скорость 108 км/ч ($\approx$ 30 м/с) за 45 м тормозного пути. Работа сил трения численно равна изменению кинетической энергии автомобиля:

$$F_{\text{тр}} \cdot S = \frac{m\Delta V^2}{2} \Rightarrow F_{\text{тр}} = \frac{m \cdot (\Delta V^2)}{2S}.$$

При этом ускорение, с которым будет замедляться автомобиль, согласно второму закону Ньютона:

$$a = \frac{F_{\text{тр}}}{m} = \frac{(\Delta V^2)}{2S} = \frac{30^2}{2 \cdot 45} = 10 \text{ м/с}^2.$$

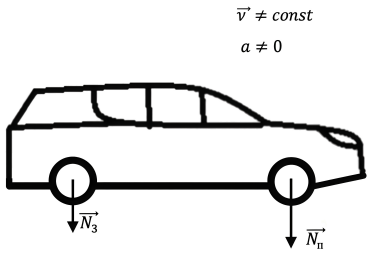

Рис. 4.2.18

Из этого следует, что эффективное значение коэффициента трения (точнее сцепления шин с дорожным полотном) в данном случае составит $\mu = 1$ (т. к. $a = g = 10$ м/с^2), т. е. сила трения численно равна весу автомобиля. Реальный коэффициент трения может быть и большим, т. к. ограничение по степени торможения регламентируется либо силой сцепления шин с дорожным полотном, либо величиной суммарного момента сил трения, создаваемым тормозной системой. Формулировка задачи подразумевает, что за эффективность торможения отвечает именно тормозная система. Она должна быть сбалансирована таким образом, чтобы максимально использовать силы трения, возникающие на контактных поверхностях каждого колеса (по всем осям). Т. е. подразумевается, что при этом фактическое значение коэффициента трения может иметь и большее значение, чем реализуемое при данном торможении.

Далее рассмотрим автомобиль схематично (как твердое тело прямоугольной формы) и покажем силы, создающие моменты сил, пытающиеся перевернуть и удержать автомобиль относительно выбранной оси, проходящей через две условные точки в месте контакта двух передних колес с асфальтом.

При $a = 0$. Согласно условию задачи, размерные величины на вспомогательном рисунке 4.2.19 имеют следующие значения: $h = 0{,}5$ м; $l = 2{,}5$ м.

Так как автомобиль не опрокидывается (не вращается) $\Rightarrow$ относительно точки O:

$$\sum \vec{M} = 0,$$

что равносильно тому, что:

$$N_3 \cdot l - mgx = 0;$$

$$x = \frac{N_3 \cdot l}{mg} = \frac{0{,}4N \cdot l}{N} = 0{,}4l.$$

В результате:

$$x = 0{,}4 \cdot 2{,}5 \text{ м} = 1 \text{ м}.$$

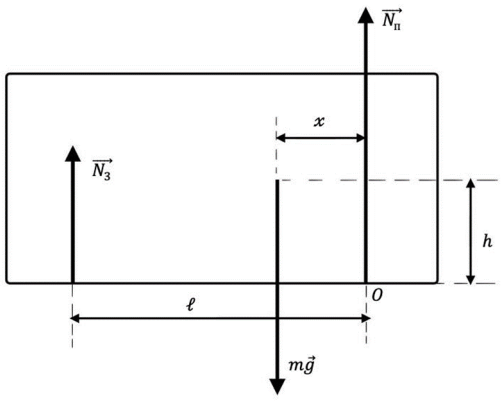

Рис. 4.2.19

Таким образом, установлено, что центр тяжести автомобиля расположен на расстоянии 1 м от передней оси и в 1,5 м от задней, если отсчет вести в горизонтальной плоскости. Дополнительного известно (из условия), что центр тяжести смещен вверх на 0,5 м.

При торможении, когда $a \neq 0$ ($a < 0$), возникают дополнительные силы, действующие на автомобиль (силы трения).

Рассмотрим моменты сил относительно центра тяжести (Ц. Т.) автомобиля, местоположение которого мы ранее определили.

Относительно Ц. Т.:

$$\sum \vec{M} = 0,$$

что равносильно тому, что:

$$N'_з \cdot (l - x) + F_{тр.з} \cdot h + F_{тр.п} \cdot h - N'_п \cdot x = 0.$$

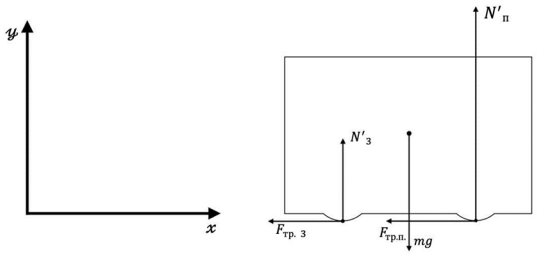

Рис. 4.2.20

При этом сумма сил по вдоль оси Y равна нулю (автомобиль не взлетает и не проваливается). Следовательно:

$$N'_з + N'_п = mg.$$

Откуда:

$$N'_з = mg - N'_п.$$

Таким образом:

$$(mg - N'_{\text{п}}) \cdot (l - x) + \mu N'_{\text{з}} \cdot h + \mu N'_{\text{п}} \cdot h - N'_{\text{п}} \cdot x = 0.$$

$$(mg - N'_{\text{п}}) \cdot (l - x) + \mu (mg - N'_{\text{п}}) \cdot h + \mu N'_{\text{п}} \cdot h - N'_{\text{п}} \cdot x = 0.$$

Учитывая, что: $x = 1$ м, $h = 0{,}5$ м; $l = 2{,}5$ м, $\mu = 1$, получаем:

$$1{,}5mg - 1{,}5N'_{\text{п}} + 0{,}5mg - 0{,}5N'_{\text{п}} + 0{,}5N'_{\text{п}} - N'_{\text{п}} = 0;$$

$$2mg - 2{,}5N'_{\text{п}} = 0.$$

Откуда:

$$N'_{\text{п}} = 0{,}8mg;$$

$$N'_{\text{з}} = mg - N'_{\text{п}} = 0{,}2mg.$$

Получается, что силы реакции опоры при данном торможении соотносятся как $\dfrac{N'_{\text{п}}}{N'_{\text{з}}} = 4$.

Требуемое соотношение тормозных усилий со стороны тормозных колодок для создания соответствующих моментов сил трения (из-за равенства диаметров тормозных дисков) будет тем же: $4:1$. При таком соотношении тормозные усилия будут максимально эффективно задействованы, обеспечивая равновероятный срыв колес в скольжение, если фактический уровень сцепления с дорогой окажется недостаточным для достижения требуемого замедления при торможении.

Таким образом, баланс тормозов (соотношение тормозных усилий, создаваемых тормозными цилиндрами) на передние и задние тормозные диски составляет $4{:}1$, что обеспечит максимально эффективное и полное торможение автомобиля со 108 км/ч за 45 м пути, разумеется, при том условии, что уровень сцепление шин с дорогой достаточен ($\mu \geqslant 1$) для фактической реализации такого торможения.

Ответ: $4:1$.

Критерии оценивания

Критерий	**Баллы**
Найдено положение центра тяжести вдоль горизонтальной оси	3
Определено значение ускорения при торможении	3
Найдено эффективное значение коэффициента трения (сцепления) шин об асфальт	2
Сделаны пояснительные рисунки (схемы) с указанием сил, действующих в период торможения	4
Записан баланс моментов сил при торможении	5
Найдены значения вертикальных сил, действующие на переднюю и заднюю оси при торможении	3

Критерий	Баллы
Определены соотношения тормозных усилий (найден баланс тормозов)	3
Всего	**23**

Задача 4.2.3.2. Высоко сижу, далеко гляжу (22 балла)

Темы: электромагнитные колебания и волны, свойства электромагнитных волн: отражение, преломление, дифракция, интерференция, взаимосвязь периода, частоты, скорости распространения и длины волны.

Условие

Эффект, связанный с изменением частоты и, соответственно, длины волны излучения, которую воспринимает наблюдатель (приемник), вследствие движения источника излучения относительно наблюдателя, назван в честь австрийского физика Кристиана Доплера. Причина данного эффекта заключается в том, что, когда источник волн движется в направлении наблюдателя, каждый последующий гребень волны выходит из положения более близкого к наблюдателю, чем гребень предыдущей волны. Таким образом, каждой последующей волне необходимо немного меньше времени, чтобы добраться до наблюдателя, чем предыдущей волне, что воспринимается наблюдателем как увеличение частоты.

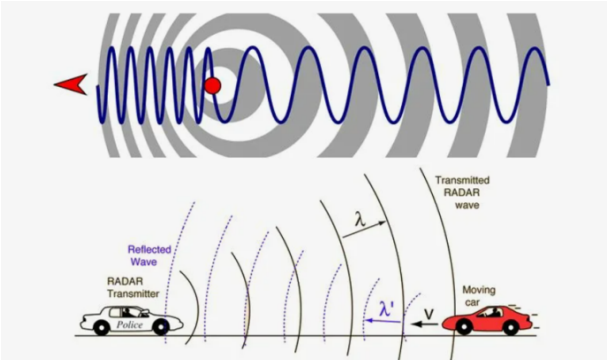

Рис. 4.2.21

Стационарный полицейский радар, работающий на эффекте Доплера, излучает радиосигнал с частотой 24 кГц (являясь первоисточником излучения). Это излучение, достигнув металлических деталей автомобиля, воспринимается в искаженном виде (вследствие сближения автомобиля с источником). Далее происходит переизлучение радиосигнала в обратном направлении с уже искаженной частотой (с частотой, которую воспринял автомобиль).

В результате радиосигнал, который будет зарегистрирован радаром, получит дополнительное изменение частоты, которое также будет зависеть от скорости движения этого автомобиля и скорости распространения электромагнитных волн.

Фиксируя отраженный сигнал и определяя его частоту (и/или длину волны), прибор сопоставляет эти искаженные значения с исходными характеристиками из-

лучения и по величине этих изменений (например, по изменению длины волны) вычисляет скорость движения автомобиля.

Вопрос: с какой допустимой ошибкой полицейский радар должен зафиксировать изменение длины электромагнитной волны, чтобы обеспечить определение скорости автомобиля, движущегося со скоростью 100 км/ч, с погрешностью, не превышающей 1%, предполагая, что прибор идеально отъюстирован и ведет регистрацию в прекрасных условиях видимости на ровном и прямом участке автодороги?

Решение

Эффект Доплера связан с изменением частоты и, соответственно, длины волны излучения, которое воспринимает наблюдатель (прибор) вследствие движения источника (источника и/или наблюдателя). Когда источник волн движется к наблюдателю, каждый последующий гребень волны выходит из положения, более близкого к наблюдателю, чем гребень предыдущей волны. Таким образом, каждой последующей волне необходимо немного меньше времени, чтобы добраться до наблюдателя, чем предыдущей волне, что воспринимается наблюдателем как увеличение частоты.

Если наблюдатель (радар) неподвижен, а движется только источник (в нашем случае автомобиль), то нетрудно показать, что каждый последующий гребень волны отражаемого от автомобиля излучения выходит из положения, более близкого к наблюдателю (к радару), чем гребень предыдущей волны. Если математически формализовать это утверждение, то получим следующее соотношение:

$$V \cdot f'_s = f \cdot (V - V_s),$$

где

- V — скорость распространения волн; в нашем случае $V = c \approx 3 \cdot 10^8$ м/с (равно скорости света);
- V_s — скорость сближения источника (автомобиля) к наблюдателю (к радару);
- f'_s — частота испущенного от автомобиля сигнала;
- f — частота, воспринимаемая прибором.

Следовательно, частота, воспринимаемая прибором (f), будет связана с частотой отраженного от автомобиля сигнала f'_s следующим образом:

$$f = \frac{V}{V - V_s} \cdot f'_s.$$

Однако в рассматриваемом случае прибор регистрирует отраженное излучение, которое уже было искажено по сравнению с исходным сигналом, испущенным с частотой $f_0 = 24$ кГц:

$$f'_S = \frac{V}{V - V_s} \cdot f_0,$$

где $V = c$ — скорость света.

Таким образом:

$$f = \left(\frac{c}{c - V_s}\right)^2 \cdot f_0 \Rightarrow \frac{f_0}{f} = \left(1 - \frac{V_s}{c}\right)^2.$$

Длина волны λ связана с частотой f и скоростью c как:

$$\lambda = \frac{c}{f} \Rightarrow \frac{\lambda_1}{\lambda_2} = \frac{f_2}{f_1}.$$

Следовательно, для нашего случая:

$$\frac{\lambda}{\lambda_0} = \frac{f_0}{f} = \left(1 - \frac{V_S}{c}\right)^2 = 1 - 2\left(\frac{V_S}{c}\right) + \left(\frac{V_S}{c}\right)^2.$$

В силу того, что скорость света на много порядков больше скорости сближения источника (автомобиля) к наблюдателю (к радару), последним членом выражения $\left(\frac{V_S}{c}\right)^2$ можно пренебречь по причине его чрезвычайной малости при $V_S \ll c$.

В результате отношение длин волн и частот можно рассчитывать по формуле:

$$\frac{\lambda}{\lambda_0} = \frac{f_0}{f} \approx 1 - 2\left(\frac{V_S}{c}\right).$$

Тогда абсолютное изменение длины волны, зарегистрированное прибором, должно иметь следующее значение:

$$|\Delta\lambda| = |\lambda - \lambda_0| = 2\left(\frac{V_S}{c}\right) \cdot \lambda_0 = 2\frac{V_s}{c} \cdot \frac{c}{f_0} = 2\left(\frac{V_s}{f_0}\right) \approx 2 \cdot \frac{(100/3{,}6)\ \text{м/с}}{24 \cdot 10^3\ \text{Гц}} \approx 2{,}315 \cdot 10^{-3}\ \text{м}.$$

Прибор должен зафиксировать скорость автомобиля с погрешностью не более 1%, поэтому он должен обладать достаточной точностью и чувствительностью, при которой он будет способен надежно фиксировать изменения длин волн, не превышающих одной сотой доли от найденной нами величины абсолютного изменения, возникшего при движении автомобиля со скоростью 100 км/ч, что соответствует: $2{,}3 \cdot 10^{-5}$ м.

Ответ: $2{,}3 \cdot 10^{-5}$ м.

Критерии оценивания

Критерий	**Баллы**
Записано (выведено) выражение для изменения частоты (и/или длины волны) излучения, воспринимаемое наблюдателем в зависимости от скорости источника и скорости распространения сигнала	7
Сделан учет того, что исходным источником излучения является не автомобиль, а радар	5
Записана взаимосвязь длины волны со скоростью распространения и частотой излучения	5
Определено изменение длины волны, возникающее при регистрации отраженного сигнала от движущегося со скоростью 100 км/ч автомобиля	3

Критерий	Баллы
Вычислена необходимая точность прибора (его разрешающая способность) для погрешности измерения скорости не более 1%	2
Всего	**22**

Задача 4.2.3.3. Если фары зажигают, значит, это кому-то нужно! (20 баллов)

Темы: оптика, основы геометрической и физической оптики, прямолинейное распространение света в однородной среде, точечный источник света, связь мощности излучения и температурой абсолютно черного тела (закон Стефана – Больцмана).

Условие

Фары автомобиля, использующие лампы накаливания (с температурой нагрева нити 2 127 °C), обеспечивают необходимый уровень освещенности только на ближайшие 10 м дорожного полотна, что часто является недостаточным, повышая риск ДТП по причине позднего визуального обнаружения другого транспортного средства, пешехода, велосипедиста, животного или иного вида препятствия или помехи.

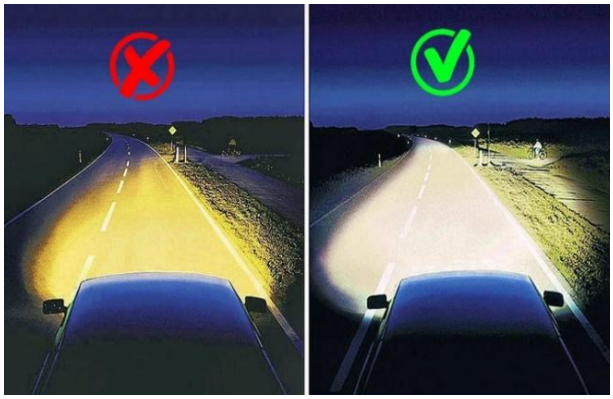

Рис. 4.2.22

Энергетическая светимость (W) нагретого серого тела, коим является нить накала, подчиняется закону Стефана – Больцмана, согласно которому $W = \varepsilon \cdot \sigma \cdot T^4$, где ε — степень черноты тела, σ — постоянная Стефана – Больцмана, T — абсолютная температура.

Определите, на сколько градусов по шкале Цельсия необходимо повысить температуру накала нити, чтобы, не внося изменений в конструкцию фар и ламп, обеспечить надлежащий уровень освещенности на дистанции в 15 м. Интенсивность светового потока, проходящего через основание конуса условно примите распределенным равномерно. Учтите, что распространение света от фары можно считать происходящим внутри некоторого конуса, сохраняющего самоподобие.

Решение

Исходная рабочая температура накала нити лампы составляет: $t_1 = 2\,127$ °C, что эквивалентно 2400 K (по абсолютной шкале температур).

Энергетическая светимость серого тела $W \sim T^4$ ($W = \varepsilon \cdot \sigma \cdot T^4$), где

- ε — степень черноты тела,
- σ — постоянная Стефана – Больцмана,
- T — абсолютная температура.

Так как конструктивных изменений в устройство не вносилось, отношение светимостей будет связано исключительно с изменением температуры накала нити:

$$\frac{W_2}{W_1} = \left(\frac{T_2}{T_1}\right)^4.$$

Световой поток Ф, создаваемый лампой, пропорционален энергетической светимости лампы, следовательно,

$$\frac{\Phi_2}{\Phi_1} = \left(\frac{T_2}{T_1}\right)^4.$$

На расстояниях в 10–15 м нить накала лампы можно смело считать точечным источником света. Следовательно, освещенность (E) будет убывать обратно пропорционально площади поверхности той части сферы, находящейся в пределах телесного угла, в котором распространяется световой поток, испускаемый источником света (фарой):

$$E = \frac{\Phi}{S_{\text{пов.}}}.$$

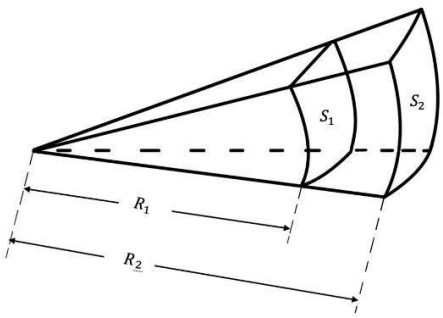

Рис. 4.2.23

Площадь части сферы $S_{\text{пов.}}$ внутри телесного угла ω численно равна:

$$S_{\text{пов.}} = 4\pi R^2 \frac{\omega}{4\pi} = \omega R^2,$$

где ω — телесный угол, R — радиус сферы.

В результате отношение освещенностей, создаваемых лампами, разогретыми до разных температур на разных расстояниях, определяются из соотношения:

$$\frac{E_2}{E_1} = \frac{\Phi_2}{\Phi_1} \cdot \left(\frac{R_1}{R_2}\right)^2 = \left(\frac{T_2}{T_1}\right)^4 \cdot \left(\frac{R_1}{R_2}\right)^2.$$

Из чего следует, что:

$$T_2 = T_1\sqrt{\frac{R_1}{R_2}} \cdot \sqrt[4]{\frac{E_2}{E_1}}.$$

Так как в задаче требуется, чтобы уровень освещенности на дистанции 15 м для более разогретой лампы был тем же, что и на 10 м для исходной лампы, т. е.:

$$E_2 = E_1;$$

$$T_2 = T_1\sqrt{\frac{R_2}{R_1}} = 2\,400\ \text{К}\sqrt{\frac{15\ \text{м}}{10\ \text{м}}} \approx 2\,939\ \text{К};$$

$$\Delta T = T_2 - T_1 = (2\,939 - 2\,400)\ \text{К} = 539\ \text{К} = 539\ ^\circ\text{C}.$$

Ответ: на 539 °C.

Критерии оценивания

Критерий	**Баллы**
Осуществлен перевод температуры накала в абсолютную шкалу температур (из градусов Цельсия в градусы Кельвина)	2
Записано выражение о связи температуры с энергетической светимостью в привязке к выражению светового потока, формируемому фарой	4
Показано, что отношение световых потоков, формируемых серым телом, при неизменности прочих факторов, будет определяться по соотношению абсолютных температур, возведенных в четвертую степень	4
Сделан пояснительный рисунок, показано, что уровень освещенности убывает с квадратом расстояния (объяснено через отношение площадей оснований у подобных конусов или через телесный угол и площадь части сферы)	5
Вычислена температура накала нити, необходимая для создания требуемой освещенности на дистанции в 15 м	4
Получено значение необходимого изменения температуры накала нити	1
Всего	**20**

Задача 4.2.3.4. Мы с тобой на одной волне — ты и я! (15 баллов)

Темы: электромагнетизм, электромагнитное поле, электромагнитная индукция, магнитный поток, индуктивность, электромагнитные колебания и волны, колебательный контур, вынужденные электромагнитные колебания, резонанс.

Условие

Индуктивность соленоида (L) зависит от числа витков соленоида (n), его длины (l), площади витков с током (S), магнитной проницаемости (μ) и некоторого коэффициента (k), зависящего от формы соленоида:

$$L = k \cdot \mu_0 \cdot \mu \cdot \frac{n^2 \cdot S}{l}.$$

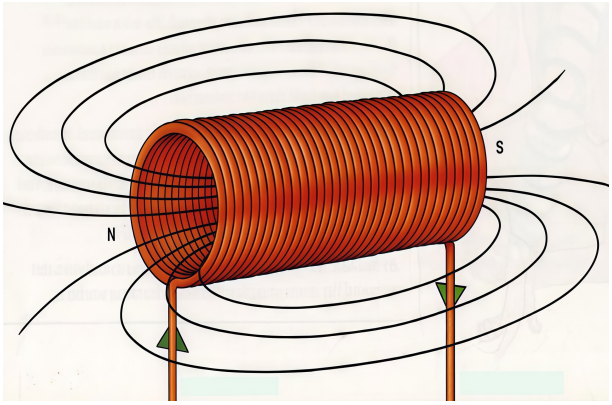

Рис. 4.2.24

В результате небольшой технической аварии была случайно повреждена медная катушка, использованная в качестве соленоида (с индуктивностью L), входящего как один из основных элементов колебательного контура в антенне радиоприемника квадрокоптера.

После анализа состояния пострадавшей катушки и обнаружения мест повреждения выяснилось, что проволока была намотана на цилиндрическую оправку плотно в один слой, и что повторно можно будет использовать не всю проволоку, а только основной ее фрагмент, составляющий 85% от первоначальной длины проволоки. Ремонт катушки заключался в повторной намотке неповрежденной части проволоки на оправку того же диаметра с тем же шагом.

Во сколько раз необходимо будет изменить емкость конденсатора, чтобы после ремонта катушки частота приема сигнала осталась прежней?

Решение

Индуктивность соленоида (L) зависит от числа витков соленоида (n), его длины (l), площади витков с током (S), магнитной проницаемости (μ) и некоторого коэффициента (k), зависящего от формы соленоида:

$$L = k \cdot \mu_0 \cdot \mu \cdot \frac{n^2 \cdot S}{l}.$$

Из-за того что катушка индуктивности частично пострадала, длина проволоки, которую можно было использовать повторно, уменьшилась и составила 85% от первоначальной длины. Если осуществлять повторную намотку с тем же шагом и на том же диаметре оправки, то:

$$n' = 0{,}85 \cdot n_0;$$

$$l' = 0{,}85 \cdot l_0;$$
$$S' = S_0.$$

При этом индуктивности «после» и «до», будут соотносится как:

$$\frac{L'}{L_0} = \frac{0{,}85^2}{0{,}85} = 0{,}85.$$

Электрическую схему приемника радиосигнала можно изобразить в виде трех составных элементов, имеющих соответствующее омическое, емкостное и индуктивное сопротивление.

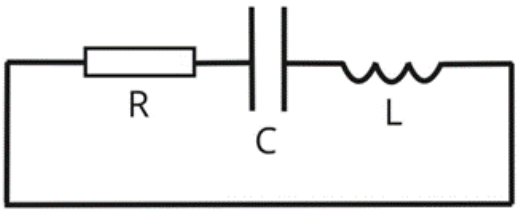

Рис. 4.2.25

Условие резонанса в такой цепи будет удовлетворять циклическая частота $\omega_0 = \frac{1}{\sqrt{LC}}$, при которой полное сопротивление достигает минимума (постепенно достигая величины омического сопротивления), т. е. когда $\omega_0 L = \frac{1}{\omega_0 C}$ (когда индуктивное равно емкостному сопротивлению), при этом:

$$C = \frac{1}{L \cdot \omega_0^2}.$$

Так как по условию задачи требуется сохранить резонансную частоту (частоту приема), то:

$$\frac{C'}{C_0} = \frac{L_0}{L'} = \frac{L_0}{0{,}85 \cdot L_0} \approx 1{,}17647 \approx 1{,}18.$$

Ответ: необходимо увеличить примерно в 1,18 раза.

Критерии оценивания

Критерий	**Баллы**
С учетом формулы для вычисления индуктивности соленоида показано, какие параметры, влияющие на индуктивность соленоида, изменились и насколько сильно они изменились при сокращении длины проволоки	3
Численно оценено изменение индуктивности после ремонта по сравнению со значением индуктивности до аварии	3
Сформулировано условие резонанса в цепи колебательного контура	4

Критерий	Баллы
Записано аналитическое выражение для резонансной частоты	3
Получено численное значение, отражающее степень требуемого изменения емкости конденсатора для сохранения резонансной частоты (частоты приема сигнала) при измененном значении индуктивности соленоида	2
Всего	**15**

Задача 4.2.3.5. «Сила есть — ума не надо!»... Или все же надо?! (20 баллов)

Темы: механика, механическое движение и его виды, движение по окружности, законы Ньютона, момент сил, законы сохранения в механике, импульс тела, момент импульса, работа, мощность, работа как мера изменения энергии.

Условие

На испытательном стенде при проверке и обкатке двигателя была получена зависимость развиваемой им мощности от частоты оборотов двигателя (см. соответствующий график и таблицу на рис. 4.2.26).

На основании полученных экспериментальных данных оцените, при какой частоте вращения (из числа приведенных в таблице) испытуемый двигатель выдал максимальный крутящий момент и какова при этом была его величина?

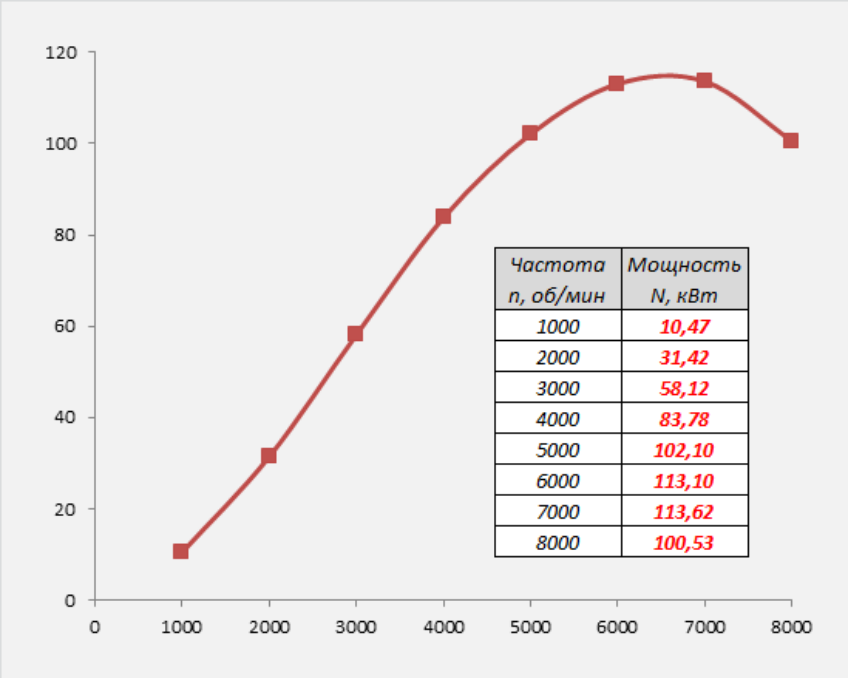

Рис. 4.2.26

Решение

Момент силы — векторная величина, которая характеризует силовое воздействие, вызывающее вращательное движение.

Крутящий момент двигателя можно выразить как векторное произведение радиус-вектора $\vec{r}$ — вектора, проведенного от оси вращения до точки приложения (генерации) силы $\vec{F}$, на вектор силы $\vec{F}$:

$$\vec{M} = [\vec{r} \times \vec{F}].$$

Мощность двигателя можно воспринять как некий объем работы, который может выполнять двигатель за рассматриваемый промежуток времени и отнесенный к длительности этого промежутка времени.

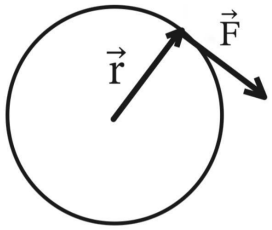

Рис. 4.2.27

Мощность связана с действием силы, совершаемом на некотором преодолеваемом расстоянии:

$$N = \frac{\Delta A}{\Delta t} = \frac{F \cdot \Delta S \cdot \cos\alpha}{\Delta t} = F \cdot V \cdot \cos\alpha,$$

где α — угол между векторами $\vec{F}$ и $\vec{V}$; $\vec{V}$ — скорость материальной точки.

В случае воздействия момента силы совершаемое действие выражается через «угловое расстояние» $\Delta\varphi$ (изменение угла):

$$N = \frac{\Delta A}{\Delta t} = \frac{M \cdot \Delta\varphi}{\Delta t} = M \cdot \omega,$$

где ω — угловая скорость (рад/сек): $\omega = 2\pi n$; n — частота вращения (об/с).

В результате:

$$N = \frac{2\pi M n^*}{60},$$

где n^* — частота оборотов двигателя, выраженная в (об/мин).

В итоге для выражения крутящего момента можно получить выражение, связывающее его с мгновенной мощностью и частотой оборотов:

$$M = \frac{30 \cdot N}{\pi \cdot n^*}.$$

Согласно данным, приведенным в таблице, и сделанным соответствующим вычислениям момента, можно сделать вывод о том, что максимальное значение момента $M_{\text{мах}} \approx 200$ Н·м было достигнуто при частоте $n^* = 4\,000$ об/мин.

Таблица 4.2.10

Частота, об/мин	Мощность, кВт	Крутящий момент, Н·м
1 000	10,47	100
2 000	31,42	150
3 000	58,12	185
4 000	**83,78**	**200**
5 000	102,1	195
6 000	113,1	180
7 000	113,62	155
8 000	100,53	120

Ответ: 200 Н·м при 4 000 об/мин.

Критерии оценивания

Критерий	**Баллы**
Дано определение, что такое крутящий момент	1
Приведена формула, выражающая значение крутящего момента через приложенную силу и плечо (сделан поясняющий рисунок)	2
Дано определение мощности	2
Показана взаимосвязь между воздействием момента силы и совершаемым действием с определенной циклической частотой, которое можно выразить как мощность	5
Получено итоговое выражение для вычисления крутящего момента, связывающее его с мгновенной мощностью и частотой оборотов двигателя	4
Согласно данным, приведенным в таблице, сделаны соответствующие вычисления крутящего момента	4
Согласно вычислениям момента сделан вывод о том, что максимальное значение момента $M_{\text{мах}} \approx 200$ Н·м было достигнуто при частоте $n^* = 4\,000$ об/мин.	2
Всего	**20**

4.3. Инженерный тур

4.3.1. Общая информация

На заключительном этапе профиля АТС необходимо запустить и отладить автономную транспортную систему, состоящую из трех программируемых устройств:

- беспилотного автомобиля;
- квадрокоптера;
- сервисного центра.

Участники разрабатывают и отлаживают программы, управляющие беспилотными устройствами. Они не только создают систему для транспортировки грузов, но и интегрируют ее в информационную инфраструктуру города. Беспилотники собирают информацию, необходимую для доставки грузов, и другие полезные данные.

В этом году система должна обнаружить дефектные участки дорожного полотна и определить степень их поврежденности.

4.3.2. Легенда задачи

Администрация города НТО устраивает конкурс на разработку полностью автономной транспортной системы с функцией анализа состояния дорожного покрытия.

Для создания и отладки прототипов систем выделяется полигон с обширной сетью дорог и программируемые наземные и воздушные беспилотные аппараты. Беспилотники могут собирать три типа данных, определяющих состояние дорожного полотна:

- наличие и характер внешних повреждений;
- уровень вибраций при преодолении участка дороги;
- пробы материала дорожного полотна.

На основе сведений, полученных от беспилотников, после анализа проб система должна определить, какие участки городских дорог нуждаются в ремонте в первую очередь.

В конкурсе участвуют несколько команд разработчиков. Победителем станет та команда, которая к концу конкурса продемонстрирует наиболее совершенный прототип слаженно работающей транспортной системы. Качество ее работы определяется точностью предсказаний и количеством объектов городской среды, с которыми беспилотники корректно взаимодействуют.

4.3.3. Требования к команде и компетенциям участников

Количество участников в команде 3–4 человека, среди которых:

1. **Программист беспилотного автомобиля**: базовые знания в электронике; работа с алгоритмами локального позиционирования беспилотных автомобилей; написание алгоритмов обнаружения и взаимодействия с объектами воссозданной городской среды.
2. **Программист квадрокоптера**: знание ROS; работа с компьютерным зрением и нейронными сетями в задачах квадрокоптеров, осуществляющих навигацию над полигоном воссозданной городской среды, детектирование наземных объектов.
3. **Программист лаборатории визуального анализа**: определение трехмерных координат объектов по изображению с камеры; работа с алгоритмами детектирования и классификации цветовых маркировок; программирование промышленного манипулятора.
4. **Капитан команды**: распределение задач среди участников команды, отслеживание дедлайнов, реализация обмена данными между устройствами транспортной системы; определение степени сложности финального испытания.

4.3.4. Оборудование и программное обеспечение

Таблица 4.3.1. Оборудование

Наименование	**Описание**
Полигон «АЙКАР Стенд», версия «Город НТО» и макеты зданий для полигона «Город НТО»	Интерактивный полигон городской среды со зданиями, дорожной разметкой и объектами городской среды: пешеходами, дорожными знаками, светофорами. Моделирует город, в котором необходимо разработать и запустить автономную транспортную систему. Доступ к полигону участники получают по расписанию. На полигоне участники отлаживают всю транспортную систему в целом, решают подзадачи беспилотного автомобиля и квадрокоптера.
УМК АЙКАР	Программируемая учебная модель беспилотного автомобиля АЙКАР, испытательный полигон «АЙКАР Стенд», версия «Восьмерка» и объекты городской среды: пешеходы, дорожные знаки. Доступен участникам в течение всего заключительного этапа. Используется для запуска и отладки программ локального позиционирования беспилотного автомобиля.
Квадрокоптер «Пионер» модификации НТО	Доступен участникам в течение всего заключительного этапа. Используется для запуска и отладки программ квадрокоптера.

Лаборатория визуального анализа (ЛВА)	Устанавливается на полигоне «Город НТО». Моделирует здание с системой хранения проб дорожного полотна. Оснащена промышленным манипулятором, которым участники управляют программно. Манипулятор захватывает пробы и располагает их перед объективом камеры. Алгоритм обработки изображений с камеры и его синхронизацию с действиями манипулятора реализуют финалисты. Доступ к ЛВА участники получают по расписанию.
Объекты моделирующие повреждения дорожного покрытия	Изображения повреждений дорожного полотна. Устанавливаются на полигонах. Доступны участникам в течение всего заключительного этапа.
Стационарный компьютер для обучения нейронных сетей	Доступен участникам в течение всего заключительного этапа. Используется для обучения нейронных сетей и отладки программ, использующих их.
Ноутбук для инференса нейронных сетей	Доступен участникам в течение всего заключительного этапа. Используется для взаимодействия с программируемыми устройствами на полигоне и отладки алгоритмов компьютерного зрения и инференса нейросетей.
Ноутбук для редактирования программного кода	Доступен участникам в течение всего заключительного этапа. Используется для чтения документации, коммуникации с организаторами, поиска данных в интернете и редактирования программного кода.

Таблица 4.3.2. Программное обеспечение

Наименование	**Описание**
Tengine	Механизм для конвертации моделей нейросетей и их высокопроизводительного инференса во встраиваемых устройствах. Используется при квантовании нейросетевого детектора и запуске квантованного нейросетевого детектора на бортовом компьютере беспилотного автомобиля.
Python3	Основной язык для написания алгоритмов компьютерного зрения и работы с нейросетями. Используется для написания программ всех устройств транспортной системы.
OpenCV (библиотека для python3)	OpenCV — библиотека алгоритмов компьютерного зрения, обработки изображений, численных алгоритмов и инференса нейросетей с открытым кодом. Используется для написания программ всех устройств транспортной системы.
Darknet (фреймворк)	Darknet — фреймворк для обучения нейросетевых детекторов с открытым исходным кодом, написанный на языке C с использованием программно-аппаратной архитектуры параллельных вычислений CUDA. Используется для обучения нейросетевых детекторов.

ROS (операционная система)	ROS — экосистема для программирования роботов, предоставляющая функциональность для распределенной работы. Используется при написании и запуске программ для квадрокоптера.
LabelImg	Приложение для разметки датасетов.
PuTTY	Клиент для различных протоколов удаленного доступа. Используется для подключения к устройствам транспортной системы по SSH.

4.3.5. Описание задачи

Общая информация о командной задаче заключительного этапа

Правила проведения заключительного этапа: https://disk.yandex.ru/i/06BGpIFsc2I4Rw.

Участникам необходимо разработать и отладить автоматизированную транспортную систему с функцией анализа состояния дорожного, состоящую из трех устройств:

- беспилотного автомобиля;
- квадрокоптера;
- автоматизированной лаборатории, анализирующей пробы дорожного полотна.

Все перечисленные устройства работают на полигоне, воссоздающем городскую среду, с макетами зданий, дорогами, перекрестками, дорожными знаками, пешеходами, светофорами и объектами, моделирующими разрушающиеся участки дороги (рис. 4.3.1).

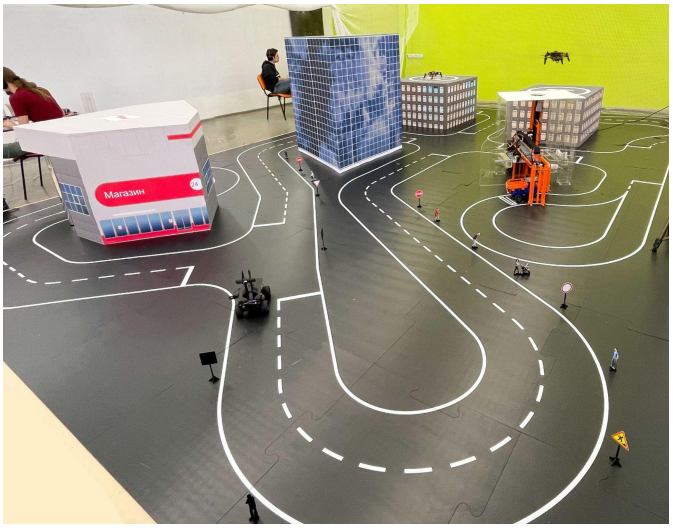

Рис. 4.3.1. Общая схема проточной части турбины Полигон «АЙКАР Стенд», версия «Город НТО»

Участки дороги разрушаются при каждом новом проезде наземного беспилотника, соответственно, изменяется их внешний вид, данные от акселерометров беспилотного автомобиля и результаты анализа проб дорожного покрытия.

Задача участников — вывести зависимость между количеством и видом внешних повреждений дороги, ее внутренней структурой, данными от акселерометров и количеством проездов, которые участок дороги сможет выдержать в будущем. Для этого необходимо собрать статистические данные от беспилотников и лаборатории.

На заключительном этапе беспилотники выступают средством для сбора и анализа информации. Каждый из трех типов полученных от них сведений позволяет улучшить точность предсказания необходимости ремонта участков дороги. Причем все данные должны анализироваться автоматически без участия человека.

Для каждого устройства необходимо написать разработать несколько алгоритмов.

Квадрокоптер:

- Взлет, патрулирование города, посадка.
- Поиск на изображении с камеры разрушающихся участков дороги и визуальный анализ их повреждений.
- Передача координат и данных о визуальных повреждениях беспилотным автомобилям.

Беспилотный автомобиль:

- Перемещение по дорожной сети города с соблюдением ПДД из пункта А в пункт Б.
- Получение данных о поврежденных участках дороги, требующих изучения.
- Сбор и анализ данных с акселерометров, установленных на колесной паре автомобиля.
- Построение маршрута в городе с учетом разрушенных участков дороги.
- Обмен данными с лабораторией визуального анализа при отгрузке проб дорожного полотна.
- Обнаружение на изображении с камеры различных объектов городской среды.

Лаборатория визуального анализа (ЛВА):

- Обмен данными с беспилотным автомобилем при отгрузке проб дорожного полотна.
- Захват проб дорожного полотна и их транспортировка к визуальному анализатору.
- Алгоритм визуального анализа структуры проб дорожного полотна.

Задача разработки транспортной системы разбита на несколько подзадач. Сдавать подзадачи и получать за них баллы участники могут до 10:00 (МСК) 7 марта.

Во время финальных испытаний участники демонстрируют слаженную работу нескольких устройств транспортной системы.

Материалы, предоставленные участникам на старте заключительного этапа, находятся в архиве: https://disk.yandex.ru/d/IaRr08YUFMhdDQ/Material/.

Файлы с решениями подзадач находятся в папке: https://disk.yandex.ru/d/NJ2o0pOxDW162A.

Задача 4.3.5.1. Подзадачи беспилотного автомобиля (45 баллов)

Подзадача БПА-1. Коммутация электронных модулей беспилотного автомобиля

Условие

Необходимо изучить инструкцию по работе с моделью беспилотного автомобиля АЙКАР, соединить электронные модули согласно схеме подключения и продемонстрировать работу базового программного кода.

Для выполнения подзадачи необходимо:

- показать разработчику профиля скоммутированные электронные модули и получить разрешение на включение питания модели беспилотного автомобиля;
- запустить на модели беспилотного автомобиля АЙКАР базовый программный код и продемонстрировать его работу разработчику профиля.

Критерии оценивания подзадачи:

1 балл — базовый код работает (беспилотник движется по разметке).

Комментарии

Выполнив это подзадание, участники убеждаются в исправности оборудования и работоспособности базового программного кода, а также осваивают умения, необходимые для дальнейшей работы с беспилотным автомобилем, а именно:

- установка аккумуляторов;
- включение питания систем беспилотника;
- передача файлов на бортовой компьютер;
- управление операционной системой бортового компьютера;
- запуск программ на бортовом компьютере.

Это задание является обязательным для выполнения всеми участниками. Его выполнение гарантирует, что они располагают исправным оборудованием и минимальными необходимыми навыками для работы.

Типовые ошибки при выполнении подзадания:

- невнимательное чтение инструкции;
- включение питания без разрешения разработчика профиля.

Решение

Файлы с решениями подзадач находятся в папке: https://disk.yandex.ru/d/NJ2o0pOxDW162A.

Для решения задачи нужно:

1. Безошибочно подключить соединительные провода к электронным модулям беспилотного автомобиля.
2. Подключиться к бортовому компьютеру беспилотного автомобиля через SSH.

3. Скопировать файлы базового программного кода на бортовой компьютер и запустить их для исполнения.

Эти действия выполняются по предоставленным инструкциям.

Подзадача БПА-2. Старт при исчезновении запрещающего знака и остановка в зоне приема проб дорожного полотна ЛВА

Условие

Порядок действий:

1. АЙКАР устанавливается на расстоянии 2 м перед зоной приема проб дорожного полотна ЛВА, которая отмечена зеленым прямоугольником.
2. В 50 см от беспилотника (за пределами проезжей части) устанавливается знак «Движение запрещено».
3. По команде организатора участники запускают программу.
4. Беспилотный автомобиль начинает движение в тот момент, когда организатор убирает знак из зоны видимости камеры.
5. Автомобиль должен остановиться в зоне приема проб ЛВА (зона приема проб ЛВА обозначены зеленым прямоугольником, расположенным между ограничивающими дорожную полосу линиями разметки).

Подзадача считается выполненной, если:

- беспилотный автомобиль начал движение при исчезновении знака из поля зрения камеры;
- после полной остановки беспилотника зеленый прямоугольник находится между передней и задней колесными осями;
- подряд проведены два успешных испытания.

Критерии оценивания подзадачи:

- *3 балла — беспилотник начал движение, как только знак «Движение запрещено» исчез из его зоны видимости.*
- *2 балл — беспилотник остановился в зоне приема проб ЛВА.*

Типовые ошибки при выполнении подзадачи:

- недостаточно точно подобранные пороги бинаризации для обнаружения запрещающего знака;
- реагирование на все зеленые прямоугольники, обнаруживаемые на идущих друг за другом кадрах.

Решение

Файлы с решениями подзадач находятся в папке: https://disk.yandex.ru/d/NJ2o0pOxDW162A.

Для решения поставленной задачи следует разобраться в базовом коде и добавить в него два алгоритма: алгоритм обнаружения запрещающего движение знака и

алгоритм поиска зоны приема проб ЛВА.

Для поиска дорожного знака на изображении можно применять различные способы:

- обучение нейросети;
- использование каскада Хаара;
- поиск объекта по цвету.

Самый простой вариант решения — последний, однако этот метод требует четко подобранных порогов бинаризации.

В решении, представленном ниже, перед основным циклом обработки кадров расположен дополнительный цикл, который выполняется, пока запрещающий знак не исчезнет из кадра. Внутри цикла проводится бинаризация и поиск контуров. Для каждого найденного контура вычисляется минимальная окружность, которая его охватывает.

Если площадь контура составляет более 80% от площади этой окружности, считается, что знак найден.

Если круг не найден в течение 2 с, то цикл завершается и начинается основной цикл обработки кадров.

Python

```python
while True:
    _, frame = cap.read()
    height, width, _ = frame.shape
    new_width = int(width * 0.3)
    cropped_frame = frame[50:250, new_width : width - new_width]
    hsl_frame = cv2.cvtColor(cropped_frame, cv2.COLOR_BGR2HLS)
    binary = cv2.inRange(hsl_frame, (132, 86, 102), (179, 255, 255))
    contours, _ = cv2.findContours(binary, cv2.RETR_EXTERNAL,
     cv2.CHAIN_APPROX_SIMPLE)
    circle_found = False
    for contour in contours:
        (x, y), radius = cv2.minEnclosingCircle(contour)
        center = (int(x), int(y))
        radius = int(radius)
        try:
            if cv2.contourArea(contour) / (np.pi * radius * radius) >
             ↪  0.8:
                cv2.circle(cropped_frame, center, radius, (0, 255, 0),
                 ↪  2)
                circle_found = True
                circle_last_seen = time.monotonic()
                print("Circle found")
                break
        except Exception as e:
            print(e)
    if not circle_found:
        print("Circle not found")
    if not circle_found and
                           (time.monotonic() - circle_last_seen) > 2:
        print("Circle not found for 2 seconds, exiting...")
        break
```

Вторая часть задания связана с остановкой в зоне приема проб ЛВА. Задача сводится к детектированию зеленого прямоугольника и отсчету времени для остановки

беспилотника в нужный момент. В решении, представленном ниже, проводится бинаризация так, чтобы зеленые пиксели стали белыми, а все остальные – черными.

Подсчитывается число белых пикселей: если оно больше определенного значения, то линия считается обнаруженной (это должно произойти два раза подряд). Если линия обнаружена, то засекается время и через 0,514 с беспилотник останавливается.

Следующий фрагмент кода располагается в основном цикле обработки кадров.

Python

```python
mark_frame = frame[150:200, 216:316]
hsv_frame = cv2.cvtColor(mark_frame, cv2.COLOR_BGR2HSV_FULL)
bin_frame = cv2.inRange(hsv_frame, (0, 76, 0), (179, 255, 255))
green = np.count_nonzero(bin_frame)
# print(f"Current green: {green}")
if green > 600 and not green_flag:
    # print("Found green > 2000")
    green_flag = True
if green_flag and green < 600 and SUPER_FLAG == False:
    print("Green end")
    LAST_TIME_DETECTED = time.monotonic()
    SUPER_FLAG = True
if SUPER_FLAG == True and time.monotonic() > LAST_TIME_DETECTED +
↪  0.514:
    STATE = STOP
    print("STOP")
    arduino.stop()
```

Подзадача БПА-3. Получение данных от акселерометров на колесной паре беспилотного автомобиля

Условие

Акселерометры установлены на колесной паре автомобиля и непрерывно фиксируют изменения в ускорении автомобиля. Чтобы получить данные от них, необходимо в момент нахождения беспилотника на поврежденном участке отправить сообщение: `get_sample`. Данные представляют собой 200 чисел от 0 до 100, разделенных пробелом. Это значения вертикального ускорения в условных единицах, измеренные через равные промежутки времени.

Порядок действий:

1. АЙКАР устанавливается на расстоянии 1–2 м от поврежденного участка дороги.
2. По команде организатора участники запускают программу.
3. Беспилотный автомобиль должен проехать по поврежденному участку дороги и вывести данные от акселерометров в терминал SSH-соединения с бортовым компьютером автомобиля.

Подзадача считается выполненной, если:

- беспилотный автомобиль отправил сообщение акселерометру в момент движения через поврежденный участок дороги;

- в терминал SSH соединения с автомобилем вывелись данные от акселерометров;
- разработчик просмотрел код запущенных программ и признал его рабочим.

Критерии оценивания подзадачи:

4 балла — получены данные от акселерометров на колесной паре беспилотного автомобиля.

Комментарии

В IP-сети беспилотных аппаратов был развернут MQTT-сервер, моделирующий акселерометры. Для получения данных от акселерометра следовало отправить запроса MQTT-серверу и получить ответное сообщение.

Типовые ошибки при выполнении подзадачи:

- множественное детектирование одного и того же поврежденного участка;
- неверно определенные и указанные в коде IP-адреса устройств;
- отсутствие декодирования сообщения от сервера при его получении;
- непрерывная серия запросов к серверу вместо одного запроса.

Решение

Файлы с решениями подзадач находятся в папке: https://disk.yandex.ru/d/NJ2o0pOxDW162A.

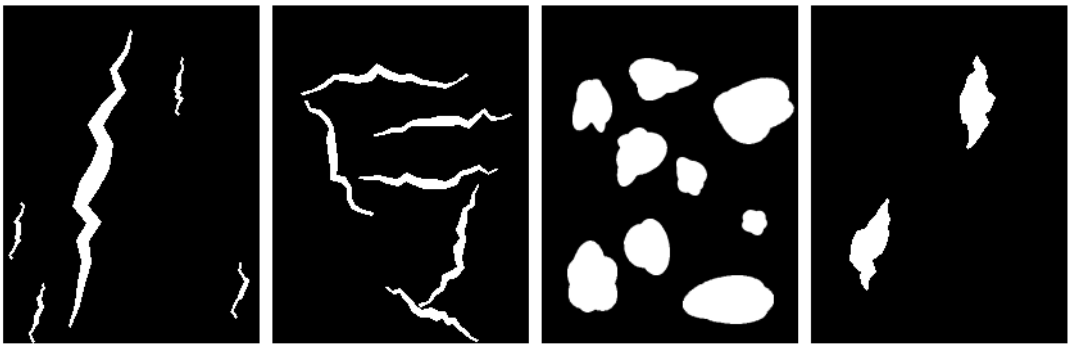

Рис. 4.3.2. Изображения поврежденных участков дорожного полотна

Такие изображения накладываются на полигон для беспилотных автомобилей между линиями дорожной разметки. Базовый алгоритм движения по разметке не отличает друг от друга белые повреждения и линии дорожной разметки, из-за чего беспилотный автомобиль съезжает с трассы на поврежденных участках. Поэтому часть задачи — модернизация алгоритма движения по разметке.

В базовый алгоритм детектирования разметки можно добавить функцию, которая будет удалять белые повреждения с изображения участка дороги перед колесами автомобиля.

Python

```
def fill_points(bin_frame):
    contours, _ = cv2.findContours(bin_frame,
                          cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    new_mask = np.zeros_like(bin_frame)
    if contours:
        min_x = min(cv2.boundingRect(cnt)[0] for cnt in contours)
        max_x = max(cv2.boundingRect(cnt)[0] +
                    cv2.boundingRect(cnt)[2] for cnt in contours)
        tolerance = 50
        left_contours = [cnt for cnt in contours if
                         cv2.boundingRect(cnt)[0] <= min_x + tolerance]
        right_contours = [cnt for cnt in contours if
                          cv2.boundingRect(cnt)[0] +
                         cv2.boundingRect(cnt)[2] >= max_x - tolerance]
        cv2.drawContours(new_mask, left_contours, -1, 255,
         thickness=cv2.FILLED)
        cv2.drawContours(new_mask, right_contours, -1, 255,
          thickness=cv2.FILLED)
    return new_mask
```

Функция получает на вход бинаризованное изображение участка дороги, на котором производится поиск всех контуров, определяются координаты самого правого и левого контура. Контуры в интервале $tolerance = 50$ от самого левого и правого переносятся на новое черно-белое изображение, идентичное по размерам исходному. Таким образом все контуры повреждений между линиями дорожной разметки исчезают. Полученное изображение можно передавать в базовый алгоритм движения по разметке.

Вторая часть задачи — обнаружение поврежденных участков дороги. Надежнее всего использовать для этой цели нейросетевой детектор. Необходимо собрать и разметить датасет со всеми видами поврежденных участков дорожного полотна — сфотографировать объекты с разнообразных ракурсов, под которыми беспилотный автомобиль может их увидеть.

Если предполагается обнаружение объектов на одной конкретной дистанции, то в датасет нужно добавлять только фотографии с объектами на этом конкретном удалении от камеры; в настройках обучения детектора нужно отключить масштабирование при аугментации данных.

На полученном датасете необходимо обучить детектор Yolo v4-tiny и выполнить квантование модели для запуска на бортовом компьютере беспилотного автомобиля. Ниже приведен код, использующий детектор для обнаружения поврежденных участков дорожного полотна. Для избежания множественных срабатываний детектора вводится задержка 2 с.

Перед циклом чтения кадров выполняются инструкции, загружающие модель нейронной сети из файла и настраивающие ее якорные рамки.

Python

```
model = yolopy.Model('best.tmfile', use_uint8=True,
use_timvx=True, cls_num=1)
model.set_anchors([18, 33, 33, 48, 25, 71, 58, 76,
40, 113, 87, 140])
```

Следующая функция возвращает `True`, если в кадре появляется поврежденный

участок дороги.

Python

```python
def detect_yolo(frame):
    frame = apply_perspective(frame, [(511, 392),
(725, 391), (406, 713), (841, 715)])
    classes, scores, boxes = model.detect(frame)
    return len(classes) > 0
```

В цикле обработки кадров добавлена проверка наличия поврежденных участков дороги. Если подобные участки есть и в течение последних 2 с подобных участков в кадре не было, то отправляется запрос MQTT-серверу, ответ от которого выводится в терминал.

Python

```python
if detect_yolo(frame):
    if time.time() - last_detect > 2:
        client.send()
        response = client.get_output()
        print(response)
    last_detect = time.time()
```

Подзадача БПА-4. Получить карту расположения поврежденных участков дорожного полотна от квадрокоптера

Условие

Под картой подразумевается набор данных, позволяющий построить изображение полигона с отмеченными поврежденными участками дорог. У каждого участка необходимо:

- обозначить тип повреждений;
- указать порядковый номер;
- отметить число посещений беспилотным автомобилем.

Поврежденные участки должны быть обозначены так, чтобы было ясно, на какой стороне дороги повреждение, и, с точностью до одного пунктирного штриха разметки, понятно его положение.

По команде организатора участники запускают три программы:

- имитатор-квадрокоптер, отправляющий карту;
- программу, принимающую данные от квадрокоптера на борту беспилотного автомобиля;
- программу, визуализирующую карту на компьютере оператора.

Подзадача считается полностью выполненной, если:

- на компьютере оператора отобразилась карта поврежденных участков;
- в терминал SSH-соединения с автомобилем выведены данные, пересылаемые беспилотником, и участник объяснил методику их интерпретации для получения карты;
- разработчик просмотрел код запущенных программ и признал его рабочим.

Критерии оценивания подзадачи:

5 баллов — получена карта расположения поврежденных участков дорожного полотна от квадрокоптера.

Комментарии

Стартовое положение поврежденных участков на карте участники задавали самостоятельно.

Типовые ошибки при выполнении подзадачи:

- попытки передавать не метаданные, а изображение карты;
- неверно определенные и указанные в коде IP-адреса устройств;
- отсутствие отображения типа повреждения, порядкового номера точки или числа посещений беспилотником.

Решение

Файлы с решениями подзадач находятся в папке: https://disk.yandex.ru/d/NJ2o0pOxDW162A.

Задачу можно разбить на две части: создание программы для пересылки метаданных о поврежденных участках дороги, а также программы для построения карты по метаданным и ее отображения.

Ниже представлен код программы сервера, принимающего и хранящего данные о поврежденных участках дороги. Серверу передаются новые данные, после чего информация о поврежденных участках обновится.

У сервера можно запросить те данные, которые он сейчас хранит. Программа сервера использует FastAPI. Определен класс для хранения данных о точках — поврежденных участках дороги. Задан `endpoint` для отправки данных о точках на сервер и `endpoint` для получения данных обо всех точках, хранящихся на сервере.

Python

```python
import copy
import uvicorn
import time
from fastapi import FastAPI, File, UploadFile
from fastapi.responses import JSONResponse
from pydantic import BaseModel
from typing import List, Dict
class PointsRequest(BaseModel):
    type: str
    number: int
    count: int
    x: int
    y: int
@app.post("/points")
async def post_points(request: List[PointsRequest]):
    global points
    points = []
    for point in request:
        points.append({"type": point.type, "number": point.number,
        ↪  "count": point.count, "x": point.x, "y": point.y})
```

```
    return {"status": "OK", "detail": None}
@app.get("/points")
async def get_points():
    print(points)
    return {"status": "OK", "detail": points}
if __name__ == "__main__":
    uvicorn.run(app, host="0.0.0.0", port=8000)
```

Сервер является промежуточным звеном для передачи метаданных через IP-сеть.

Для обращения к серверу необходимо организовать соответствующие функции. Ниже представлен пример их реализации в классе `Server`.

Python

```python
import requests
class Server:
    def __init__(self):
        self.base_url = "http://172.16.65.50:8000"
    def send_points(self, points: list) -> bool:
        ans = requests.post(f"{self.base_url}/points", json=points)
        if ans.status_code != 200:
            return False
        return ans.json()["status"] == "OK"
    def get_points(self) -> list:
        ans = requests.get(f"{self.base_url}/points")
        if ans.status_code != 200:
            return []
        return ans.json()["detail"]
```

Ниже представлен код приложения, которое отправляет данные о поврежденных участках дороги на сервер, а затем получает их обратно и отображает карту на основе этих данных.

Класс `DamagePoint` реализует структуру для хранения информации о поврежденных участках. Метод `json` возвращает словарь с данными о поврежденной точке, который можно использовать для отправки на сервер.

Python

```python
class DamagePoint:
    def __init__(self, x, y, damage_type, visit_count, number):
        self.x: int = x
        self.y: int = y
        self.damage_type = damage_type
        self.visit_count: int = visit_count
        self.number: int = number
    def json(self):
        return {"type": self.damage_type, "number": self.number,
        ↪  "count": self.visit_count, "x": self.x, "y": self.y}
```

Функция `send_all_points` отправляет все поврежденные точки на сервер. Если отправка успешна, выводит сообщение об этом.

Python

```python
def send_all_points(points: List[DamagePoint]):
    data = []
    for point in points:
        data.append(point.json())
    if server.send_points(data):
        print("Data was sent to server!")
    else:
        print("Error in sanding data!")
```

Функция `draw_points` отображает поврежденные точки на изображении. На месте каждой точки рисуется круг, и рядом с ним отображается информация о типе повреждения, количестве посещений беспилотником и порядковый номер.

Python

```python
image_path = "./image.jpg"
image = cv2.imread(image_path)
image = cv2.resize(image, (815, 600))
original_image = image.copy()
damage_points = []
def draw_points():
    global image
    image = original_image.copy()
    for idx, point in enumerate(damage_points):
        cv2.circle(image, (point.x, point.y), 5, (0, 0, 255), -1)
        text = f"{point.damage_type}/{point.visit_count}/
         {point.number}"
        cv2.putText(image, text, (point.x - 20, point.y - 15),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)
    cv2.imshow("Map", image)
```

Функция `update_points` обновляет список поврежденных точек, получая их с сервера, заново рисует точки и выводит актуальную информацию о них.

Python

```python
def update_points():
    global damage_points, biggest_num
    damage_points = []
    server_points = server.get_points()
    biggest_num = 0
    for point in server_points:
        biggest_num = max(biggest_num, point["number"])
        damage_points.append(DamagePoint(point["x"], point["y"],
     point["type"], point["count"], point["number"]))
    draw_points()
def main():
    draw_points()
    while True:
        key = cv2.waitKey(1) & 0xFF
        update_points()
        if key == 27:
            break
    cv2.destroyAllWindows()
if __name__ == "__main__":
    main()
```

Подзадача БПА-5. Посещение всех поврежденных участков на карте и обновление информации о них

Условие

Порядок действий:

1. Участники демонстрируют организатору карту полигона с возможностью ручного добавления и удаления поврежденных участков дороги.
2. Организатор случайным образом размещает их на полигоне, а участники — наносят на карту.
3. Организатор устанавливает АЙКАР на трассу.
4. По команде организатора участники запускают программы.
5. Беспилотный автомобиль проезжает по всем поврежденным участкам, и в этот момент получает данные от акселерометров.

Подзадача считается выполненной, если:

- беспилотный автомобиль побывал на всех разрушенных участках дороги;
- при каждом посещении поврежденного участка беспилотник отправлял запрос акселерометрам;
- счетчик посещений каждого такого участка отображал верное число посещений и увеличивался в реальном времени;
- после остановки беспилотника на компьютере оператора выводятся все данные, полученные от акселерометров.

Критерии оценивания подзадачи:

16 баллов — посещены все поврежденные участки на карте и обновлена информация о них. Представлены три набора статистических данных.

Комментарии

Это самая сложная и ресурсозатратная подзадача из подзадач беспилотного автомобиля! Для ее решения следует дополнить код из предыдущей подзадачи.

Необходимо реализовать:

- интерфейс, позволяющий установить на карте поврежденные участки;
- алгоритм поиска маршрута по городу, который охватывает все разрушенные участки;
- такое взаимодействие беспилотника и карты, при котором счетчик посещений поврежденных участков на карте обновляется в реальном времени (то есть как только беспилотник проезжает по поврежденному участку, на карте отображается новое число посещений).

Типовые ошибки при выполнении подзадачи:

- интерфейс не позволяет задать положение поврежденного участка;
- некорректно изменяется число посещений беспилотным автомобилем: увеличивается больше, чем на 1, либо не увеличивается вовсе;
- беспилотник неверно строит маршрут и не посещает все поврежденные участки дороги.

Решение

Файлы с решениями подзадач находятся в папке: https://disk.yandex.ru/d/NJ2o0pOxDW162A.

Для передачи данных между беспилотным автомобилем и компьютером, отображающим карту, удобно использовать FastAPI-сервер из предыдущей подзадачи, дополненный функциями для передачи задачи беспилотному автомобилю.

Задача — последовательность поворотов, которые необходимо совершить, и `id` поврежденых точек, которые нужно посетить. Кроме того, необходимо добавить функцию для увеличения на сервере счетчика посещений точки с конкретным `id`. Ниже представлен программный код, реализующий эти дополнения.

Python

```python
@app.post("/task")
async def post_task(request: List[str]):
    global tasks
    tasks = request
    return {"status": "OK", "detail": None}
@app.get("/task")
async def get_task():
    global tasks
    tasks_copy = copy.deepcopy(tasks)
    tasks = []
    return {"status": "OK", "detail": tasks_copy}
@app.post("/point")
async def visit_point(number: int):
    for i in range(len(points)):
        if points[i]["number"] == number:
            points[i]["count"] += 1
            return {"status": "OK", "detail": None}
    return {"status": "ER", "detail": "Number not found"}
```

К классу `Server` из предыдущей подзадачи добавлены методы для новых взаимодействий с сервером.

Python

```python
def send_task(self, tasks: list) -> bool:
    ans = requests.post(f"{self.base_url}/task", json=tasks)
    print(ans.json())
    if ans.status_code != 200:
        return False
    return ans.json()["status"] == "OK"
def get_task(self) -> list:
    ans = requests.get(f"{self.base_url}/task")
    if ans.status_code != 200:
        return []
    return ans.json()["detail"]
def visit_point(self, number) -> list:
    ans = requests.post(f"{self.base_url}/point?number={number}")
    if ans.status_code != 200:
        return []
    return ans.json()["status"]
```

Наибольшее количество изменений требуется внести в программу, отвечающую за отображение карты, а именно:

- возможность размещать точки на карте с помощью мыши;
- алгоритм для формирования задания для беспилотного автомобиля.

Новая функция `mouse_callback` обрабатывает события, связанные с мышью. При нажатии левой кнопки мыши добавляется новая поврежденная точка или удаляется существующая. При нажатии правой кнопки мыши устанавливаются стартовая и конечная точки для беспилотника.

Python

```python
robot_zone = None
stop_zone = None
freeze_zone = None
def mouse_callback(event, x, y, flags, param):
    global damage_points, robot_zone, biggest_num, stop_zone,
    ↪ freeze_zone
    if event == cv2.EVENT_LBUTTONDOWN:
        for i, point in enumerate(damage_points):
            if abs(point.x - x) < 10 and abs(point.y - y) < 10:
                print(f"Удаление точки: {point.damage_type}")
                del damage_points[i]
                draw_points()
                send_all_points(damage_points)
                return
        damage_type = input("Тип повреждения: ")
        damage_point = DamagePoint(x, y, damage_type, 0, biggest_num +
        ↪ 1)
        biggest_num += 1
        damage_points.append(damage_point)
        send_all_points(damage_points)
        draw_points()
    elif event == cv2.EVENT_RBUTTONDOWN:
        if not robot_zone:
            robot_zone, _ = DamagePoint(x, y, 0, 0, 0).get_zone()
            print(f"Робот установлен в зону {robot_zone}")
            freeze_zone = input("Зона для установки: ")
            print("Нажмите на точку остановки")
        else:
            stop_zone, _ = DamagePoint(x, y, 0, 0, 0).get_zone()
        if robot_zone and stop_zone:
            plan_robot_route()
```

Функция `plan_robot_route` вызывается после установки всех точек на карте. Она формирует список действий, которые должен выполнить робот, и отправляет его на сервер.

Python

```python
def plan_robot_route():
    if not robot_zone:
        print("Зона беспилотника не установлена.")
        return
    target_zones = {dp.get_zone()[0] for dp in damage_points}
    if not target_zones:
        print("Точек нет")
        return
    route, visited = find_optimal_route(robot_zone, target_zones)
    vis_cnt = 0
    task = []
```

```
    for i in range(len(route) - 1):
        for neighbor, action in graph.get(route[i], []):
            if neighbor == route[i + 1]:
                task.append(action)
                while (vis_cnt < len(visited) and i + 1 ==
             visited[vis_cnt][0] - 1):
                    if freeze_zone == visited[vis_cnt][1]:
                        pass
                    else:
                        task.append(visited[vis_cnt][1])
                    vis_cnt += 1
                break
    task.append("f")
    print(task)
    print(visited)
    server.send_task(task)
```

В приведенном коде используется функция `find_optimal_route`, которая определяет последовательность поворотов, необходимых беспилотнику для посещения всех поврежденных участков дорожного покрытия.

Python

```
def find_optimal_route(start, targets):
    print(targets)
    targets = set(targets)
    visited = []
    current_zone = start
    route = [current_zone]
    while targets:
        best_next_zone = None
        best_path = None
        min_turns = 1e9
        for target in targets:
            path = find_shortest_path(current_zone, target)
            turns = len(path)
            if turns < min_turns:
                min_turns = turns
                best_next_zone = target
                best_path = path
        if best_next_zone:
            route.extend(best_path[1:])
            for pt in find_all_points(best_next_zone, damage_points):
                visited.append((len(route), pt))
            targets.remove(best_next_zone)
            current_zone = best_next_zone
    path = find_shortest_path(current_zone, stop_zone)
    route.extend(path[1:])
    return route, visited
```

Для работы с точками и поиска пути через них используются зоны. Зоны — это абстракция, представляющая каждый участок дорожной полосы между двумя перекрестками. Чтобы определить, в какой именно зоне находится конкретная точка, используется заранее сформированный `json`-файл с соответствием пикселей изображения и `id` конкретных зон.

Python

```python
def get_zone(self):
    nearest_zone = None
    min_distance = 1e9
    percent_position = 0
    for zone_id, points in lines.items():
        points = np.array(points)
        distances = np.linalg.norm(points - np.array((self.x,
                                                      self.y)), axis=1)
        min_idx = np.argmin(distances)
        min_dist = distances[min_idx]
        if min_dist < min_distance:
            min_distance = min_dist
            nearest_zone = zone_id
            percent_position = (min_idx / (len(points) - 1)) * 100
    return nearest_zone, float(round(percent_position, 2))
```

Ниже приведен код еще двух функций `find_shortest_path` и `find_all_points`, они используются при формировании задания для беспилотника.

`find_all_points` находит все поврежденные точки в заданной зоне и возвращает их номера, отсортированные по порядку расположения в зоне.

`find_shortest_path` находит кратчайший путь от начальной точки до целевой точки, используя алгоритм поиска в ширину (BFS). Возвращает список зон, составляющих путь. Граф, в котором осуществляется поиск, задается словарем.

Python

```python
graph = {
    "6": [("10", "STRAIGHT"), ("9", "LEFT")],
    "7": [("15", "RIGHT")],
    "8": [("7", "RIGHT"), ("10", "LEFT")],
    "9": [("13", "RIGHT"), ("16", "LEFT")],
    "10": [("14", "LEFT")],
    "11": [("9", "RIGHT"), ("7", "STRAIGHT")],
    "13": [("11", "RIGHT")],
    "14": [("16", "STRAIGHT"), ("8", "LEFT")],
    "15": [("8", "RIGHT"), ("13", "STRAIGHT")],
    "16": [("6", "LEFT")]
}
def find_shortest_path(start, target):
    queue = deque([(start, [], None)])
    visited = set()
    while queue:
        current_zone, path, last_direction = queue.popleft()
        if current_zone == target:
            return path + [current_zone]
        if current_zone in visited:
            continue
        visited.add(current_zone)
        for neighbor, direction in graph.get(current_zone, []):
            if neighbor not in visited:
                queue.append((neighbor, path + [current_zone],
                 ↪  direction))
    return []
def find_all_points(zone, damage_points):
    data = []
    for point in damage_points:
```

```
        if point.get_zone()[0] == zone:
            data.append((point.number, point.get_zone()[1]))
    return [f"{dt[0]}" for dt in sorted(data, key=lambda x: x[1])]
```

Таким образом, беспилотный автомобиль может запрашивать с сервера задание, состоящее из последовательности поворотов и `id` точек, которые необходимо посетить. Беспилотник делает это до тех пор, пока не получит его, а затем формирует список маневров, которые необходимо совершить, и список `id` точек для посещения.

Python

```
data = server.get_task()
while len(data) == 0:
    data = server.get_task()
    time.sleep(0.5)
algorithm = []
points = []
points_cnt = 0
prev_r = 0
prev_l = 0
for step in data:
    if step == "STRAIGHT":
        algorithm.append(CROSS_STRAIGHT)
    elif step == "RIGHT":
        algorithm.append(CROSS_RIGHT)
    elif step == "LEFT":
        algorithm.append(CROSS_LEFT)
    else:
        points.append(step)
```

В основном цикле обработки кадров, аналогично БПА-3 выполнено обнаружение поврежденных участков дороги.

Python

```
if detect_yolo(trans_perspective(frame, TRAP, RECT, SIZE)):
    detect_count += 1
    print("temp detected", detect_count)
    if time.time() - last_acur_detect > 2 and detect_count > 2:
        last_acur_detect = time.time()
        print("DETECTED!")
        server.visit_point(points[points_cnt])
        points_cnt += 1
        # Получение и вывод данных от акселерометра
        client.send()
        response = client.get_output()
        print(response)
    if detect_count > 2:
        last_acur_detect = time.time()
    last_detect = time.time()
```

При обнаружении очередного поврежденного участка дороги беспилотный автомобиль увеличивает счетчик посещенных участков — по номеру встреченного участка находит его идентификатор в заранее полученном списке, а затем отправляет серверу сообщение о его посещении. Сервер обновляет счетчик для данного участка, и эта информация отображается на карте, так как карта постоянно запрашивает сервер о статусе точек и обновляет их отображение.

Подзадача БПА-6. Взятие проб дорожного полотна и их доставка в ЛВА

Условие

Порядок действий:

1. Участники демонстрируют организатору карту полигона с возможностью ручного добавления и удаления поврежденных участков дороги.
2. Организатор случайным образом размещает на полигоне поврежденные участки, а участники наносят их на карту.
3. Организатор устанавливает АЙКАР на трассу.
4. По команде участники запускают программы.
5. Организатор сообщает номер участка, пробы которого необходимо взять.
6. Участники передают номер участка в программу через стандартный поток ввода.
7. Беспилотник доезжает до указанного участка, берет пробы дорожного полотна, доставляет в зону приема проб ЛВА.

Подзадача считается полностью выполненной, если:

- АЙКАР доехал до указанного участка, остановился и вывел в терминал SSH соединения сообщение `collecting samples of the road surface`;
- после вывода сообщения беспилотник доехал и остановился в зоне приема проб ЛВА;
- разработчик просмотрел код запущенных программ и признал его рабочим.

Критерии оценивания подзадачи:

6 баллов — получен один набор статистических данных.

Комментарии

Решение этой подзадачи предполагает объединение программ, разработанных для решения БПА-2, БПА-3, БПА-5.

Типовые ошибки при выполнении подзадания:

- не останавливается для взятия пробы;
- берет пробу не того поврежденного участка, который указан на старте испытаний;
- не останавливается в зоне приема проб ЛВА.

Решение

Файлы с решениями подзадач находятся в папке: https://disk.yandex.ru/d/NJ2o0pOxDW162A.

В решение для БПА-5 необходимо внести изменение. Из всех точек (поврежденных участков) необходимо выделить ту, на которой необходимо остановиться и взять пробы дорожного полотна. После составлении задания для беспилотника ему передается список с `id` точек посещения. Точка, на которой необходимо взять пробы, отмечается отдельно! Метка добавляется в функции `plan_robot_route`.

Python

```
while vis_cnt < len(visited) and i + 1 == visited[vis_cnt][0] - 1:
    if freeze_zone == visited[vis_cnt][1]:
        task.append(visited[vis_cnt][1] + "_f")
    else:
        task.append(visited[vis_cnt][1])
    vis_cnt += 1
break
```

Беспилотный автомобиль, выполняя задание, проверяет, к какой точке он прибыл, и если это точка для взятия проб, переходит в специальное состояние `FREEZE`.

Python

```
if detect_yolo(trans_perspective(frame, TRAP, RECT, SIZE)) and not
 ↪  need_we_stop(frame):
    detect_count += 1
    print("temp detected", detect_count)
    if time.time() - last_acur_detect > 2 and detect_count > 4:
        last_acur_detect = time.time()
        print("DETECTED!")
        point = points[points_cnt]
        if "_f" in point:
            print("collecting samples of the road surface")
            point = point.split("_")[0]
            STATE = FREEZE
            freeze_start = time.time()
        server.visit_point(point)
        points_cnt += 1
```

Специальное состояние обрабатывается так, чтобы беспилотник остановился на 2 с.

Python

```
if STATE == FREEZE and time.time() - freeze_start > 2:
    STATE = GO
if STATE != STOP and STATE != FREEZE:
    arduino.set_speed(CAR_SPEED)
    arduino.set_angle(angle)
else:
    arduino.stop()
```

Функции поиска поврежденных участков дороги аналогичны БПА-3. Функции обнаружения зоны приема проб ЛВА и остановки в ней аналогичны БПА-2.

Подзадача БПА-7. Определение оставшегося срока службы поврежденного участка дороги по данным акселерометра

Условие

Порядок действий:

1. Участники демонстрируют организатору карту полигона с возможностью ручного добавления и удаления поврежденных участков дороги.
2. Организатор случайным образом размещает на полигоне поврежденные участ-

ки.

3. Участники наносят поврежденные участки на карту.
4. Организатор устанавливает АЙКАР на трассу.
5. По команде организатора участники запускают программы.
6. Организатор сообщает номер участка, который необходимо обследовать.
7. Участники передают номер участка в программу через стандартный поток ввода.
8. Беспилотник проезжает по поврежденному участку дороги, получает данные от акселерометров и выводит в терминал SSH-соединения оставшийся срок службы дорожного полотна.

Подзадача считается выполненной, если:

- беспилотный автомобиль отправил запрос акселерометрам в момент движения через поврежденный участок дороги;
- в терминал SSH-соединения с автомобилем выведен оставшийся срок службы участка дороги с ошибкой менее 10%,
- разработчик просмотрел код запущенных программ и признал его рабочим;
- испытание успешно пройдено два раза подряд.

Критерии оценивания подзадачи:

6 баллов — получен один набор статистических данных.

Комментарии

Для решения этой подзадачи нужно обучить регрессионную модель машинного обучения — по показаниям акселерометра предсказывает оставшийся срок службы поврежденного участка дороги. Срок измеряется в количестве проездов беспилотного автомобиля, которое участок сможет выдержать до полного разрушения.

За выполнение подзадач БПА-5, БПА-6, ЛВА-3, ЛВА-4, БПЛА-4, БПЛА-5 участники получают не только баллы, но и наборы статистических данных — информацию о разрушении конкретного участка дорожного покрытия. В одном таком наборе данных представлены:

- изображения повреждений дорожного полотна;
- данные от акселерометров;
- изображение пробы покрытия дорожного полотна для всех возможных сроков службы этого участка.

В CSV-файле, прилагающемуся к набору данных, устанавливается соответствие между изображениями повреждений, показаниями акселерометра и оставшимся сроком службы дорожного полотна.

При выполнении подзадачи участникам предоставляются только те данные от акселерометра, которые позволяют однозначно определить оставшийся срок службы покрытия.

Типовые ошибки при выполнении подзадачи:

- отсутствие анализа и очистки датасета перед обучением нейронной сети;
- разные версии фреймворка на устройстве инференса и устройстве для обуче-

ния нейронных сетей.

Решение

Файлы с решениями подзадач находятся в папке: https://disk.yandex.ru/d/NJ2o0pOxDW162A.

Для определения оставшегося срока службы дорожного участка лучше всего использовать нейронные сети. Анализируя статистические наборы данных, становится ясно, что дорожное полотно разрушается по восьми различным сценариям.

Трудность заключается в том, что в некоторых сценариях на начальном или конечном этапе жизни дорожного полотна показания акселерометра неизменны. Когда половина данных не содержит вариаций, модель не получает полезных градиентов для обучения на этой части.

Такое положение может приводить к тому, что оптимизатор «перестанет обращать внимание» на эту часть данных или будет пытаться компенсировать отсутствие различий в сигналах за счет искусственного завышения весов на изменяемой части. Это снизит общую обобщающую способность модели и не позволит достичь необходимой точности. Для преодоления проблемы участникам следует удалить из датасета неинформативные данные.

Задачу можно решить с помощью единственной модели машинного обучения, которая будет одновременно выполнять классификацию сценариев и регрессию для каждого из них. На начальном этапе обработки данных можно объединить представление классификационной ветви с представлением энкодера через конкатенацию, что позволит регрессионной ветви адаптировать свои предсказания в зависимости от идентифицированного сценария разрушения. Однако такой подход требует глубоких знаний в области машинного обучения и большого объема данных для обучения.

Более простой вариант заключается в том, чтобы обучить классификатор, который определит сценарий разрушения, а затем для каждого сценария обучить отдельную регрессионную модель.

Ниже представлен код для формирования архитектуры классификатора, его обучения и сохранения обученной модели. Параметры обучения подбираются экспериментально.

Python

```python
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten,
    ↪ Dense, Dropout
# Параметры
SEQ_LENGTH = 200
NUM_CLASSES = 8
# Создаем модель классификатора с использованием одномерных сверточных
    ↪ слоев.
input_shape = (SEQ_LENGTH, 1)
model = Sequential(name="Conv1D_Classifier")
model.add(Conv1D(filters=32, kernel_size=3,
                 activation='relu', padding='same',
                  ↪ input_shape=input_shape))
```

```
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.3))
model.add(Conv1D(filters=64, kernel_size=3, activation='relu',
    ↪ padding='same'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.3))
model.add(Conv1D(filters=128, kernel_size=3, activation='relu',
    ↪ padding='same'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
# Выходной слой: NUM_CLASSES нейронов, softmax активация для
    ↪ классификации
model.add(Dense(NUM_CLASSES, activation='softmax'))
# learning_rate подбирается экспериментально
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-3),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.summary()
# Параметры обучения
epochs = 10
batch_size = 10
# Обучение модели
history = model.fit(
   X_train, Y_train,
   epochs=epochs,
   batch_size=batch_size,
   validation_data=(X_test, Y_test))
# Сохранение модели
model.save("Class.h5")
```

Модель различает восемь классов сценариев разрушения, и для каждого обучается регрессионная модель, при этом их архитектуры будут одинаковы. Параметры обучения для каждой модели подбираются экспериментально. Ниже представлен код создания, обучения и сохранения регрессионной модели.

Python

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten,
    ↪ Dense, Dropout
# Параметры генератора
SEQ_LENGTH = 200
BATCH_SIZE = 32
# Создаем модель для регрессии.
input_shape = (SEQ_LENGTH, 1)
model = Sequential(name="Conv1D_Regressor")
model.add(Conv1D(filters=32, kernel_size=3, activation='relu',
    ↪ padding='same', input_shape=input_shape))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.3))
model.add(Conv1D(filters=64, kernel_size=3, activation='relu',
    ↪ padding='same'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.3))
```

```
model.add(Conv1D(filters=128, kernel_size=3, activation='relu',
    ↪  padding='same'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(1, activation='linear'))  # выходной нейрон для
    ↪  регрессионного предсказания
# learning_rate подбирается экспериментально
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-3),
              loss='mse',
              metrics=['mae'])
model.summary()
# Параметры обучения
steps_per_epoch = 100  # батчей на эпоху
epochs = 10
# Обучение модели
history = model.fit(
    X_train, Y_train,
    epochs=epochs,
    batch_size=batch_size,
    validation_data=(X_test, Y_test))
# Сохранение модели
model.save("Regress_0.h5")
```

Для каждого сценария разрушения обучаем свою модель. В ходе работы беспилотника каждую модель нужно применить лишь один раз, соответственно, загружать их из файлов и проводить инференс уместно в момент, когда беспилотник получил данные от акселерометра для целевого поврежденного участка.

Сначала беспилотник останавливается, затем загружается и используется классификатор. Он определяет, по какому сценарию разрушается участок дороги, далее используется регрессионная модель для конкретного сценария.

Python

```
if detect_yolo(trans_perspective(frame, TRAP, RECT, SIZE)):
    detect_count += 1
    print("temp detected", detect_count)
    if time.time() - last_acur_detect > 2 and detect_count > 2:
        last_acur_detect = time.time()
        print("DETECTED!")
        server.visit_point(points[points_cnt])
        points_cnt += 1
        # Останавливаем беспилотник
        arduino.stop()
        # Получаем данные от акселерометра
        client.send()
        response = client.get_output()
        # print(response)
        # Загружаем модель классификатора из файла
        model = load_model('Class.h5')
        X_test = string_to_numpy_array(strintg_chis)
        X_test = np.expand_dims(X_test, axis=1)
        X_test = np.expand_dims(X_test, axis=0)
        # Определяем класс сценария разрушения
        y_pred = model.predict(X_test)
        predicted_classes = np.argmax(y_pred, axis=1)
        # print("Предсказанный класс:", predicted_classes[0])
```

```
        next_model_name = ["Regress_1.h5", "Regress_2.h5",
        ↪  "Regress_3.h5",
                           "Regress_4.h5", "Regress_5.h5",
                            ↪  "Regress_6.h5",
                           "Regress_7.h5", "Regress_8.h5"]
        # Загружаем регрессионную модель для конкретного сценария
        model = load_model(next_model_name[predicted_classes[0]])
        # Определяем оставшийся срок службы и выводим в терминал
        y_pred = model.predict(X_test)
        y_pred = round(y_pred[0][0])
        print(y_pred)
```

При подготовке данных от акселерометра для передачи на вход нейронной сети используется функция, переводящая строку чисел разделенных пробелами в массив numpy:

Python

```
# Функция для перевода данных от акселерометра к массиву numpy
def string_to_numpy_array(data_string: str, dtype=float) ->
↪  np.ndarray:
    string_numbers = data_string.split()
    numeric_values = list(map(dtype, string_numbers))
    numpy_array = np.array(numeric_values)
    return numpy_array
```

Система оценивания

Таблица 4.3.3

Описание подзадачи	**Примечания**	**Баллы за подзадачу**
Коммутация электронных модулей беспилотного автомобиля		1
Старт при исчезновении запрещающего знака и остановка в зоне приема проб дорожного полотна ЛВА	2 — беспилотник остановился в зоне приема проб ЛВА; 3 — беспилотник стартовал при исчезновении знака.	5
Получение данных от акселерометров на колесной паре беспилотного автомобиля		4
Получение карты расположения поврежденных участков дорожного полотна от квадрокоптера		5
Посещение всех поврежденных участков на карте и обновление информации о них		16
Взятие проб дорожного полотна и их доставка в ЛВА.		6

Описание подзадачи	Примечания	Баллы за подзадачу
Определение оставшегося срока службы поврежденного участка дороги по данным акселерометра		6
Сумма баллов за все подзадачи		43

Задача 4.3.5.2. Подзадачи лаборатории визуального анализа (20 баллов)

Подзадача ЛВА-1. Транспортировка пробы

Условие

Порядок действий:

1. По команде организатора участники запускают программу.
2. Организатор сообщает участникам номер пробы от 1 до 12. Через стандартный ввод номер пробы передается в программу.
3. Манипулятор захватывает пробу с указанным номером, подносит ее к камере и возвращает на место.

Задание считается полностью выполненным, если последовательно выполнены следующие условия:

- манипулятор захватил пробу и поднес ее к камере;
- на экране оператора появилось изображение с камеры ЛВА;
- манипулятор вернул пробу в исходное положение;
- в процессе транспортировки проба не касалась других проб;
- испытание успешно пройдено два раза подряд.

Критерии оценивания подзадачи:

2 балла — произведена транспортировка пробы.

Типовые ошибки при выполнении подзадачи:

- попытки перемещать захват через запрещенную область;
- одновременное понижение высоты и сжимание захвата.

Решение

Файлы с решениями подзадач находятся в папке: https://disk.yandex.ru/d/NJ2o0pOxDW162A.

Для выполнения этого задания участникам необходимо разобраться в предоставленном им API управления промышленным манипулятором.

Для каждого действия, которое может совершать манипулятор, нужно организовать отдельные функции. Функция `capture_sample` применяется для захвата пробы с конкретным номером.

Python

```python
def capture_sample(robot, sample_number):
    #   Функция для захвата пробы с указанным номером
        sample_positions = {
        1: (450, -400, 95),
        2: (550,-400, 95),
        3: (650, -400, 95),
        4: (650,-300, 95),
        5: (550, -300, 95),
        6: (650,-200,95 ),
        7: (650, 260,95),
        8: (650,360,95),
        9: (550, 360,95),
        10: (650,460,95),
        11: (550, 460,95),
        12: (450,460,95),    }
    if sample_number not in sample_positions:
        print("Некорректный номер пробы")
        return False
    x, y, z = sample_positions[sample_number]
    robot.move("Robot1_1", 489, -131.1, 424, 0, 0)
    # Перемещение к пробе
    robot.move("Robot1_1", x, y, z, 0, 0)
    time.sleep(2)  # Ожидание завершения движения
    # Захват пробы
    robot.move("Robot1_1", x, y, z, 0, 1)  # Активировать захват
    time.sleep(1)
    return True
```

Функция `transport_to_camera` отвечает за перемещение пробы к объективу камеры.

Python

```python
def transport_to_camera(robot):
    #Функция для транспортировки пробы к камере.
    # Координаты камеры
    camera_position = (670, -511, 311.5)
    # Перемещение к камере
    robot.move("Robot1_1", *camera_position, 0, 1)
    time.sleep(2)
    # Получение изображения с камеры
    image_byte = robot.getCamera1Image()
    if image_byte:
        image_np = np.frombuffer(image_byte, np.uint8)
        image_np = cv2.imdecode(image_np, cv2.IMREAD_COLOR)
        cv2.imshow("Camera Image", image_np)
        cv2.waitKey(1)
        time.sleep(5)  # Ожидание для просмотра изображения
        cv2.destroyAllWindows()
    else:
        print("Ошибка получения изображения с камеры")
        return False
    return True
```

Функция `return_sample` возвращающая пробу на исходную позицию.

Python

```python
 def return_sample(robot, sample_number):
 # Функция для возврата пробы на место.

sample_positions = {
     1: (450, -400, 95),
    2: (550,-400, 95),
    3: (650, -400,95),
    4: (650,-300, 95),
    5: (550, -300, 95),
    6: (650,-200, 95),
    7: (650, 260,95),
    8: (650,360,95),
    9: (550, 360,95),
    10: (650,460,95),
    11: (550, 460,95),
    12: (450,460,95),}
if sample_number not in sample_positions:
    print("Некорректный номер пробы")
    return False
x, y, z = sample_positions[sample_number]
# Перемещение к месту возврата
robot.move("Robot1_1", x, y, z, 0, 1)
time.sleep(2)
# Отпускание пробы
robot.move("Robot1_1", x, y, z, 0, 0)  # Деактивировать захват
time.sleep(1)
return True
```

Основная логика работы реализована в функции `main`. Она запускает цикл, который продолжается, пока не будет выполнено одно успешное испытание. У пользователя запрашивается номер пробы (образца). Проводится проверка корректности номера, после чего вызываются функции `capture_sample`, `transport_to_camera` и `return_sample`.

Если какая-то из функций возвращает `False`, выводится сообщение об ошибке и продолжается цикл. В случае, когда все операции выполнены успешно, счетчик увеличивает число успешных запусков и выводит сообщение об успешном завершении испытания. После успешного завершения всех операций выводится сообщение о завершении задания.

Python

```python
def main():
   robot = MCX()
   successful_runs = 0
   while successful_runs < 1:
       # Запрос номера пробы
       sample_number = int(input("Введите номер пробы (1-12): "))
       if sample_number < 1 or sample_number > 12:
           print("Некорректный номер пробы.                Введите
            ↪  число от 1 до 12.")
           continue
       # Захват пробы
       if not capture_sample(robot, sample_number):
           print("Ошибка захвата пробы")
           continue
       # Транспортировка к камере
```

```
        if not transport_to_camera(robot):
            print("Ошибка транспортировки к камере")
            continue
        # Возврат пробы на место
        if not return_sample(robot, sample_number):
            print("Ошибка возврата пробы")
            continue
        # Успешное выполнение
        print(f"Испытание {successful_runs + 1} успешно завершено")
        successful_runs += 1
    print("Задание выполнено успешно 2 раза подряд.")
if __name__ == "__main__":
    main()
```

Подзадача ЛВА-2. Осмотр пробы с разных ракурсов

Условие

Стартовые условия аналогичны предыдущей подзадаче.

Задание считается полностью выполненным, если последовательно выполнены следующие условия:

- манипулятор захватил пробу и поднес ее к камере;
- на экране оператора появились восемь изображений пробы под разными углами с шагом в 45°;
- манипулятор вернул пробу в исходное положение;
- в процессе транспортировки проба не касалась других проб.

Критерии оценивания подзадачи:

4 балла — произведен осмотр пробы с разных ракурсов.

Типовые ошибки при выполнении подзадачи:

- неверно определенные углы, под которыми нужно фотографировать образец перед объективом камеры;
- вызов API манипулятора без проверки завершения выполнения предыдущего вызова.

Решение

Файлы с решениями подзадач находятся в папке: https://disk.yandex.ru/d/NJ2o0pOxDW162A.

При решении этой подзадачи используются программы из предыдущей подзадачи.

Необходимо сократить функцию `transport_to_camera` и вынести в функцию `capture_images` процесс получения восемь изображений пробы под разными углами.

Python

```python
def transport_to_camera(robot):
    #Функция для транспортировки пробы к камере.
    camera_position = (670, -511, 311.5)    # Координаты камеры
    # Перемещение к камере
    robot.move("Robot4_1", *camera_position, 0, 1)
    while robot.getManipulatorStatus() == 1:  # Ожидание завершения
    ↪ движения
        time.sleep(0.1)
    return True
def capture_images(robot):
    #Функция для получения 8 изображений пробы под разными углами.
    angles = [0, 45, 90, 135, 180, -135, -90, -45]  # Углы для поворота
    ↪ (в пределах [-180°, 180°])
    images = []
    for angle in angles:
        # Поворот кисти манипулятора на заданный угол
        robot.move("Robot4_1", 670, -511, 311.5, angle, 1)
        while robot.getManipulatorStatus() == 1:  # Ожидание завершения
        ↪ движения
            time.sleep(0.1)
        # Получение изображения с камеры
        image_byte = robot.getCamera1Image()
        if image_byte:
            image_np = np.frombuffer(image_byte, np.uint8)
            image_np = cv2.imdecode(image_np, cv2.IMREAD_COLOR)
            images.append(image_np)
            # Отображение изображения в отдельном окне
            window_name = f"Camera Image {angle}°"
            cv2.imshow(window_name, image_np)
            cv2.waitKey(500)  # Кратковременное отображение изображения
        else:
            print(f"Ошибка получения изображения под углом {angle}°")
            return False
    # Ожидание нажатия клавиши для закрытия окон
    print("Нажмите любую клавишу, чтобы закрыть окна с изображениями.")
    cv2.waitKey(0)
    cv2.destroyAllWindows()
    return True
```

Функция `main` теперь выглядит следующим образом.

Python

```python
def main():
    robot = MCX()
    successful_runs = 0
    while successful_runs < 1:  # Достаточно одного успешного
    ↪ выполнения
        # Запрос номера пробы
        sample_number = int(input("Введите номер пробы (1-12): "))
        if sample_number < 1 or sample_number > 12:
            print("Некорректный номер пробы. Введите число от 1 до
            ↪ 12.")
            continue
        # Захват пробы
        if not capture_sample(robot, sample_number):
            print("Ошибка захвата пробы")
            continue
        # Транспортировка к камере
```

```
        if not transport_to_camera(robot):
            print("Ошибка транспортировки к камере")
            continue
        # Осмотр пробы с разных ракурсов
        if not capture_images(robot):
            print("Ошибка получения изображений")
            continue
        # Возврат пробы на место
        if not return_sample(robot, sample_number):
            print("Ошибка возврата пробы")
            continue
        # Успешное выполнение
        print(f"Испытание успешно завершено")
        successful_runs += 1
```

Если в процессе выполнения задания появляются ошибки, и манипулятор не выполнит какую-то из промежуточных операций, то выполнение задания начинается заново.

Подзадача ЛВА-3. Сбор статистики по имеющимся в лаборатории пробам

Условие

Стартовые условия аналогичны предыдущей подзадаче, за исключением того, что оператор сообщает, а участники вводят в программу четыре номера проб.

Задание считается полностью выполненным, если последовательно выполнены следующие условия:

- манипулятор поочередно поднес указанные пробы к камере;
- манипулятор вернул все пробы в исходное положение;
- по итогам работы программы на компьютере оператора появились четыре видеозаписи с камеры лаборатории;
- на видео видно пробы со всех 360°.

Критерии оценивания подзадачи:

7 баллов — три набора статистических данных.

Типовые ошибки при выполнении подзадачи:

- получение всех сообщений сразу, без выполнения соответствующих запросам действий;
- попытка подключения обеих программ к инициализированному сокету;
- отсутствие ответных сообщений на запросы.

Решение

Файлы с решениями подзадач находятся в папке: https://disk.yandex.ru/d/NJ2o0pOxDW162A.

Для решения подзадачи нужно использовать код из подзадачи ЛВА-2.

Дополнительно необходимо организовать функцию `record_video` для записи и сохранения видеофайла.

Python

```python
def record_video(robot, sample_number, output_folder):
# Функция для записи видео с поворотом пробы на 360°.
    # Создаем папку для сохранения видео, если ее нет
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)
    # Настройки записи видео
    video_filename = os.path.join(output_folder,
      f"sample_{sample_number}.avi")
    fourcc = cv2.VideoWriter_fourcc(*'MJPG')
    fps = 10  # Частота кадров
    frame_size = (640, 480)
    # Размер кадра (зависит от разрешения камеры)
    # Инициализация VideoWriter
    out = cv2.VideoWriter(video_filename, fourcc, fps, frame_size)
    # Углы для поворота (360° с шагом 45°)
    angles = [-135, -90, -45, 0, 45, 90, 135, 180]
    for angle in angles:
        # Поворот кисти манипулятора на заданный угол
        robot.move("Robot4_1", 670, -511, 311.5, angle, 1)
        while robot.getManipulatorStatus() == 1:
            # Ожидание завершения движения
            time.sleep(1)
        # Получение изображения с камеры
        image_byte = robot.getCamera1Image()
        if image_byte:
            image_np = np.frombuffer(image_byte, np.uint8)
            image_np = cv2.imdecode(image_np, cv2.IMREAD_COLOR)
            out.write(image_np)  # Запись кадра в видео
        else:
            print(f"Ошибка получения изображения под углом {angle}°")
            return False
    # Завершение записи видео
    out.release()
    print(f"Видео для пробы {sample_number} сохранено в
         {video_filename}")
    return True
```

Логику работы функции `main` необходимо скорректировать для получения видеороликов с обзором четырех проб.

Python

```python
def main():
    robot = MCX()
    output_folder = "videos"  # Папка для сохранения видео
    # Запрос 4 номеров проб
    sample_numbers = []
    for i in range(4):
        sample_number = int(input(f"Введите номер пробы
                                                {i + 1} (1-12): "))
        if sample_number < 1 or sample_number > 12:
            print("Некорректный номер пробы. Введите число от 1 до
             ↪  12.")
            return
        sample_numbers.append(sample_number)
```

```
    # Обработка каждой пробы
    for sample_number in sample_numbers:
        # Захват пробы
        if not capture_sample(robot, sample_number):
            print(f"Ошибка захвата пробы {sample_number}")
            continue
        # Транспортировка к камере
        if not transport_to_camera(robot):
            print(f"Ошибка транспортировки пробы
             {sample_number} к камере")
            continue
```

Подзадача ЛВА-4. Определение типа конкретной пробы

Условие

Стартовые условия аналогичны ЛВА-1. После осмотра пробы в терминал SSH-соединения выводится тип (класс) пробы.

Задание считается полностью выполненным, если выполнены следующие условия:

- манипулятор захватил пробу и поднес ее к камере;
- манипулятор вернул пробу в исходное положение;
- в процессе транспортировки проба не касалась других проб;
- в терминале SSH-соединения выведен верный тип пробы;
- испытание успешно пройдено три раза подряд.

Критерии оценивания подзадачи:

7 баллов — получен один набор статистических данных.

Комментарии

Для решения этой подзадачи участники должны были обучить модель машинного обучения, которая по изображениям образцов дорожного полотна определит их тип. Датасет для обучения нейронной сети участники собирают и размечают самостоятельно.

За выполнение подзадач БПА-5, БПА-6, ЛВА-3, ЛВА-4, БПЛА-4 и БПЛА-5 участники получают не только баллы, но и наборы статистических данных. Среди прочего в них содержатся схематичные изображения образцов дорожного полотна с обозначенными классами.

Типовые ошибки при выполнении подзадачи:

- обучение нейронной сети на изображениях из статистических наборов данных;
- использование необработанных изображений с камеры ЛВА для обучения нейронной сети.

Решение

Файлы с решениями подзадач находятся в папке: https://disk.yandex.ru/d/NJ2o0pOxDW162A.

Для решения задачи нужно получить достаточное количество статистических наборов данных за БПА-5, БПА-6, ЛВА-3, ЛВА-4, БПЛА-4 и БПЛА-5, а также установить соответствие между пробами дорожного полотна в ЛВА и их классами. В статистических наборах данных содержатся схематичные изображения образцов дорожного полотна с обозначенными классами.

После того как соответствие установлено, следует собрать обучающий датасет для классификатора. Пробы дорожного полотна имеют цилиндрическую форму, поэтому для определения типа пробы необходимы их изображения со всех сторон. Для сбора датасета удобно использовать изображения проб с разных ракурсов, получаемые в ЛВА-2. Из каждого вырезается участок непосредственно с пробой, а затем из них создается новое изображение, которое передается на вход нейросети для классификации. Функция `extract_and_concatenate` его возвращает.

Python

```python
def extract_and_concatenate(image_list):
    """
    Извлекает из каждого изображения в списке область [150:500, 400:570]
    и склеивает их по горизонтали, возвращая результат как новое изображение.    """
    pieces = []
    for img in image_list:
        # Проверяем, что изображение имеет достаточную размерность
        if img.shape[0] < 500 or img.shape[1] < 570:
            raise ValueError("Размер изображения меньше требуемых для вырезания области [150:500, 400:570].")
        # Извлекаем область. С помощью slice: [150:500, 400:570]
        piece = img[150:500, 400:570].copy()
        # cv2.imshow("Piece", piece)
        # cv2.waitKey(0)
        pieces.append(piece)
    # Склейка по горизонтали, если pieces не пустой
    if pieces:
        concatenated_image = np.hstack(pieces)
    else:
        concatenated_image = None
    return concatenated_image
```

Ниже представлен код для формирования архитектуры классификатора, его обучения и сохранения обученной модели. Параметры обучения подбираются экспериментально.

Python

```python
import numpy as np
import cv2
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization
from sklearn.model_selection import train_test_split
```

```
# Параметры исходного изображения
orig_height = 350
orig_width = 1360
channels = 3
# Множитель для уменьшения изображения втрое (по оси H и W)
scale = 1 / 3
# Вычисляем размеры изображения после масштабирования
resized_height = int(orig_height * scale)
resized_width = int(orig_width * scale)
# Количество классов
num_classes = 5
# Создание модели классификатора
input_shape = (resized_height, resized_width, channels)
model = Sequential(name="Image_Classifier")
model.add(Conv2D(32, (3, 3), activation='relu', padding="same",
    input_shape=input_shape))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(64, (3, 3), activation='relu', padding="same"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, (3, 3), activation='relu', padding="same"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
# выход: 5 классов
model.compile(optimizer=tf.keras.optimizers.
Adam(learning_rate=1e-3),
loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.summary()
# Обучаем модель
epochs = 50
batch_size = 32
history = model.fit(X_train, y_train, epochs=epochs,
 ↪  batch_size=batch_size,
validation_data=(X_test, y_test))
# Сохранение модели
model.save("Class.h5")
```

Функция `predict_sample_type` использует список изображений, полученный в `capture_images` из ЛВА-2. Из списка формируется изображение для передачи на вход нейросетевому классификатору.

Python

```
def predict_sample_type(images_list):
    model = load_model('Class.h5')
    image = extract_and_concatenate(images_list)
    resized_height = int(350 / 3)
    resized_width = int(1360 / 3)
    single_img = cv2.resize(image, (resized_width, resized_height),
     ↪  interpolation=cv2.INTER_LINEAR)
    single_img = single_img / 255.0
```

```
    # Добавляем измерение батча (получим форму (1, resized_height,
    ↪  resized_width, 3))
    single_img_batch = np.expand_dims(single_img, axis=0)
    # Получаем предсказание от модели (вероятности для каждого класса)
    prediction = model.predict(single_img_batch)
    # Определяем класс с максимальной вероятностью
    predicted_class = np.argmax(prediction, axis=1)[0]
    print("Тип пробы:", predicted_class)
```

Система оценивания

Таблица 4.3.4

Описание подзадачи	**Баллы за подзадачу**
Транспортировка пробы	2
Осмотр пробы с разных ракурсов	4
Сбор статистики по имеющимся в лаборатории пробам	7
Определение типа конкретной пробы	7
Сумма за все подзадачи	20

Задача 4.3.5.3. Подзадачи квадрокоптера (37 баллов)

Подзадача БПЛА-1. Коммутация электронных модулей

Условие

Необходимо изучить образец БПЛА и соединить электронные модули аналогичным образом. После подключения всех модулей продемонстрировать работу базового программного кода.

Для выполнения подзадачи необходимо:

- показать разработчику профиля скоммутированные электронные модули и получить разрешение на подключение аккумулятора и запуск программ,
- подключиться к квадрокоптеру по SSH и ввести команду:

```text
roslaunch gs_example test_led.launch --screen
```

Критерии оценивания подзадачи:

1 балл — базовый код работает (светодиодная подсветка загорается зеленым, красным, синим и фиолетовым цветами поочередно).

Комментарии

Выполнив это подзадание, участники убеждаются в исправности оборудования и работоспособности базового программного кода, а также осваивают умения, необходимые для дальнейшей работы с квадрокоптером:

- установка аккумуляторов;
- включение питания систем беспилотника;
- передача файлов на бортовой компьютер;
- управление операционной системой бортового компьютера;
- запуск программ на бортовом компьютере.

Задание является обязательным для выполнения всеми участниками, его выполнение гарантирует, что они располагают исправным оборудованием и минимальными необходимыми навыками для работы.

Типовые ошибки при выполнении подзадачи:

- шлейф видеокамеры не до конца защелкнут в разъем видеокамеры квадрокоптера;
- не до конца защелкнут шлейф в разъем Micro-Match, соединяющий полетный контроллер «Геоскан Пионер» и модуль ультразвуковой навигации.

Решение

Файлы с решениями подзадач находятся в папке: https://disk.yandex.ru/d/NJ2o0pOxDW162A.

Для решения задачи необходимо:

- безошибочно подключить соединительные провода к электронным модулям квадрокоптера;
- подключиться к бортовому компьютеру беспилотника через SSH;
- скопировать файлы базового программного кода на бортовой компьютер и запустить их для исполнения.

Эти действия выполняются по предоставленным инструкциям.

Подзадача БПЛА-2. Обнаружение поврежденных участков дороги

Условие

Порядок действий:

1. Участники выбирают, какие типы поврежденных участков дороги необходимо обнаружить.
2. Организатор случайным образом размещает от 2 до 6 поврежденных участков дороги в полетной зоне квадрокоптера.
3. Квадрокоптер устанавливается на стартовую площадку.
4. По команде организатора участники запускают программу.
5. Квадрокоптер должен взлететь, облететь всю дорожную сеть, вывести в терминал число поврежденных участков дороги, приземлиться на крыше здания.

Подзадача считается полностью решенной, если квадрокоптер последовательно

выполнил следующие действия:

- взлетел со стартовой площадки на произвольную высоту;
- облетел всю доступную дорожную сеть и вывел в терминал число поврежденных участков дороги;
- приземлился на крыше здания и после полной остановки остался в ее пределах.

Критерии оценивания подзадачи:

1 балл — за взлет.

1 балл — за вывод правильного числа поврежденных участков дороги.

+1 балл — за каждый выбранный участниками тип поврежденных участков.

2 балла — за посадку.

Типовые ошибки при выполнении подзадачи:

- совмещение логики полета и логики детектирования повреждений в одном узле;
- обработка всех кадров видеопотока при полете,
- использование тяжеловесных нейросетевых классификаторов;
- отсутствие фильтрации одинаковых объектов;
- непрерывный вывод количества повреждений.

Решение

Файлы с решениями подзадач находятся в папке: https://disk.yandex.ru/d/NJ2o0pOxDW162A.

Для решения задачи требуется разработать два узла в системе ROS. Первый узел будет отвечать за управление перемещением квадрокоптера (далее — полетное задание), а второй — за распознавание и подсчет поврежденных участков дорожного покрытия.

Дрон будет перемещаться в системе координат ультразвуковой навигационной системы «Геоскан Локус 2», поэтому для составления маршрута полета необходимо задать определенные точки. Они должны быть расположены таким образом, чтобы фотографии, сделанные в этих точках, имели частичное перекрытие с предыдущими кадрами. Такой подход необходим для последующей фильтрации обнаруженных повреждений, которые могут встречаться на соседних кадрах.

Для определения необходимых координат можно использовать самый простой метод – измерение их вручную, перемещая квадрокоптер по полигону. Текущие координаты квадрокоптера можно получить, выполнив следующую команду в SSH-терминале:

Python

```
rostopic echo /geoscan/navigation/local/position
```

После выполнения измерений координаты следует вставить в полетное задание.

Необходимо организовать связь между двумя узлами. Оптимальным способом связи является использование сервиса. Узел распознавания будет выступать в роли сервера сервиса, а узел полетного задания — клиентом. При достижении очередной точки маршрута узел полетного задания будет отправлять запрос на обработку кадра.

Следует разработать сервис для вывода статистики полета, включая количество поврежденных участков дорожного покрытия.

Код полетного задания.

Python

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import rospy
from rospy import ServiceProxy
from std_srvs.srv import Empty, Trigger
from gs_flight import FlightController, CallbackEvent
from gs_board import BoardManager
from gs_navigation import NavigationManager

rospy.init_node("flight_test_node")  # инициализируем узел

# получаем координаты стартовой точки
nav = NavigationManager()
HOME_POINT = nav.lps.position()
HOME_POINT.z += 1.0
HOME_POINT = [HOME_POINT.x, HOME_POINT.y, HOME_POINT.z]

coordinates = [ # массив координат точек
    HOME_POINT,
    ... # необходимо вставить замеренные точки в формате [x, y, z]
    HOME_POINT
]

run = True # переменная отвечающая за работу программы
position_number = 0 # счетчик пройденных точек

def callback(event): # функция обработки событий Автопилота
    global ap
    global run
    global coordinates
    global position_number
    global detect_client
    global stat_client

    event = event.data
    if event == CallbackEvent.ENGINES_STARTED: # блок обработки события
    ↪  запуска двигателя
        print("engine started")
        ap.takeoff() # отдаем команду взлета
    elif event == CallbackEvent.TAKEOFF_COMPLETE: # блок обработки
    ↪  события завершения взлета
        print("takeoff complete")
        position_number = 0
```

```
43          ap.goToLocalPoint(coordinates[position_number][0],
            ↪  coordinates[position_number][1],
            ↪  coordinates[position_number][2]) # отдаем команду полета в
            ↪  точку
44      elif event == CallbackEvent.POINT_REACHED: # блок обработки события
        ↪  достижения точки
45          print(f"point {position_number} reached")
46          detect_client()
47          position_number += 1 # наращиваем счетчик точек
48          if position_number < len(coordinates): # проверяем количество
            ↪  текущее количество точек с количеством точек в полетном
            ↪  задании
49              ap.goToLocalPoint(coordinates[position_number][0],
                ↪  coordinates[position_number][1],
                ↪  coordinates[position_number][2]) # отдаем команду
                ↪  полета в точку
50          else:
51              ap.landing() # отдаем команду посадки
52      elif event == CallbackEvent.COPTER_LANDED: # блок обработки события
        ↪  приземления
53          print("finish programm")
54          run = False # прекращем программу
55          response = stat_client()
56          print(f"Кол-во поврежденных участков {response.message}")
57
58  detect_client = ServiceProxy("/get_photo", Empty) # клиент сервиса
    ↪  распознавания
59  stat_client = ServiceProxy("/get_stat", Trigger)
60  board = BoardManager() # создаем объект бортового менеджера
61  ap = FlightController(callback) # создаем объект управления полета
62
63  once = False # переменная отвечающая за первое вхождение в начало
    ↪  программы
64
65  while not rospy.is_shutdown() and run:
66      if board.runStatus() and not once: # проверка подлкючения RPi к
        ↪  Пионеру
67          print("start programm")
68          ap.preflight() # отдаем команду выполенения предстартовой
            ↪  подготовки
69          once = True
```

Для определения поврежденных участков требуется обучить нейросетевой классификатор Yolo v4-tiny.

Алгоритм работы узла детекции следующий:

1. После вызова сервиса распознавания производится получение изображения с камеры.
2. Осуществляется конвертирование в формат, совместимый с классификатором.
3. Выполняется распознавание объектов.
4. Выбирается наиболее вероятный класс на изображении.

Из условия задачи следует, что на одном участке дороги может быть только одно повреждение, что позволяет сузить область поиска. После этого происходит сравнение с объектами, обнаруженными на предыдущем кадре.

Если объекты совпадают, алгоритм считает, что это тот же самый объект; в противном случае в массив объектов добавляется метка класса нового объекта. При

вызове сервиса статистики производится подсчет количества ненулевых значений в массиве детекции, после чего возвращается итоговый результат.

Программный код узла распознавания приведен ниже.

Python

```python
import rospy
from sensor_msgs.msg import Image
from std_srvs.srv import Empty, EmptyResponse
from std_srvs.srv import Trigger, TriggerResponse
import cv2
from cv_bridge import CvBridge
from rospy import Service
import numpy as np
rospy.init_node("detect_node")  # инициализируем узел
bridge = CvBridge()  # создаем объект преобразования сообщений ROS в OpenCV
weights_path = "yolov4-tiny-obj_best.weights"
config_path = "yolov4-tiny-obj.cfg"
net = cv2.dnn.readNet(config_path, weights_path)
yolo_model = cv2.dnn.DetectionModel(net)
all_object = []
# Выдает самый вероятный объект на изображении
def get_most_probable_class(class_ids, confidences):
    if len(class_ids) == 0 or len(confidences) == 0:
        return None
    most_probable_index = np.argmax(confidences)
    return class_ids[most_probable_index]
# Считает количество элементов в списке, которые не равны None.
def count_not_none(all_objects):
    return len(list(filter(lambda x: x is not None, all_objects)))
def detect_handler(request):
    global bridge
    global yolo_model
    global all_object
    image_msg = rospy.wait_for_message("/pioneer_max_camera/image_raw", Image)
    try:
        cv_image = bridge.imgmsg_to_cv2(image_msg, "bgr8")
        class_ids, scores, boxes = yolo_model.detect(cv_image, 0.6, 0.4)
        most_class = get_most_probable_class(class_ids, scores)
        if (most_class is None) or (len(all_object) == 0):
            all_object.append(most_class)
        elif most_class == all_object[-1]:
            all_object.append(None)
        else:
            all_object.append(most_class)
    except:
        pass
    return EmptyResponse()
def stat_handler(request):
    global all_object
    response = TriggerResponse()
    response.success = True
    response.message = str(count_not_none(all_object))
    return response

detect_server = Service("/get_photo", Empty, detect_handler)  # создаем сервис для получения фотографии
```

```
stat_server = Service("/get_stat", Trigger, stat_handler)
rospy.spin()
```

Подзадача БПЛА-3. Определение типов поврежденных участков дороги

Условие

Порядок действий:

1. Участники выбирают, какие типы поврежденных участков дороги необходимо распознать.
2. Организатор случайным образом размещает от четырех поврежденных участков дороги в полетной зоне квадрокоптера.
3. Квадрокоптер устанавливается на стартовую площадку.
4. По команде организатора участники запускают программу.

Подзадача считается полностью решенной, если квадрокоптер последовательно выполнил следующие действия:

- взлетел со стартовой площадки на произвольную высоту;
- обнаружил все поврежденные участки и в момент обнаружения каждого участка вывел в консоль его тип;
- приземлился на крыше здания и после полной остановки остался в пределах посадочной площадки.

Критерии оценивания подзадачи:

1 балл — за каждый обнаруженный участок с правильно определенным типом.

1 балл — за каждый выбранный участниками тип поврежденных участков.

4 балла — за посадку.

Решение

Файлы с решениями подзадач находятся в папке: https://disk.yandex.ru/d/NJ2o0pOxDW162A.

Для решения подзадачи необходимо внести изменения в полетное задание из БПЛА-1, исключив вывод статистики поиска объектов, а также модифицировать узел обнаружения повреждений. В функции `detect_handler` узла обнаружения, после добавления обнаруженного объекта в массив всех найденных объектов, следует добавить следующий код для вывода распознанного повреждения.

Python

```
if all_object[-1] is not None:
    print(f"Обнаруженный тип повреждения: {all_object[-1]}")
```

Необходимо удалить сервис статистики.

Код измененного полетного задания.

Python

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import rospy
from rospy import ServiceProxy
from std_srvs.srv import Empty
from gs_flight import FlightController, CallbackEvent
from gs_board import BoardManager
from gs_navigation import NavigationManager
rospy.init_node("flight_test_node")  # инициализируем узел
# получаем координаты стартовой точки
nav = NavigationManager()
HOME_POINT = nav.lps.position()
HOME_POINT.z += 1.0
HOME_POINT = [HOME_POINT.x, HOME_POINT.y, HOME_POINT.z]
coordinates = [ # массив координат точек
    HOME_POINT,
    ... # необходимо вставить измеренные точки в формате [x, y, z]
    HOME_POINT
]
run = True # переменная отвечающая за работу программы
position_number = 0 # счетчик пройденных точек
def callback(event): # функция обработки событий Автопилота
    global ap
    global run
    global coordinates
    global position_number
    global detect_client
    event = event.data
    if event == CallbackEvent.ENGINES_STARTED: # блок обработки события
    ↪  запуска двигателя
        print("engine started")
        ap.takeoff() # отдаем команду взлета
    elif event == CallbackEvent.TAKEOFF_COMPLETE: # блок обработки
    ↪  события завершения взлета
        print("takeoff complete")
        position_number = 0
        ap.goToLocalPoint(coordinates[position_number][0],
        ↪  coordinates[position_number][1],
        ↪  coordinates[position_number][2]) # отдаем команду полета в
        ↪  точку
    elif event == CallbackEvent.POINT_REACHED: # блок обработки события
    ↪  достижения точки
        print(f"point {position_number} reached")
        detect_client()
        position_number += 1 # наращиваем счетчик точек
        if position_number < len(coordinates): # проверяем количество
        ↪  текущее количество точек с количеством точек в полетном
        ↪  задании
            ap.goToLocalPoint(coordinates[position_number][0],
                              coordinates[position_number][1],
                              coordinates[position_number][2]) # отдаем
                               ↪  команду полета в точку
        else:
            ap.landing() # отдаем команду посадки
    elif event == CallbackEvent.COPTER_LANDED: # блок обработки события
    ↪  приземления
        print("finish programm")
        run = False # прекращаем программу
```

```
49 detect_client = ServiceProxy("/get_photo", Empty) # клиент сервиса
   ↪ распознавания
50 board = BoardManager() # создаем объект бортового менеджера
51 ap = FlightController(callback) # создаем объект управления полета
52 once = False # переменная отвечающая за первое вхождение в начало
   ↪ программы
53 while not rospy.is_shutdown() and run:
54     if board.runStatus() and not once: # проверка подключения RPi к
       ↪ Пионеру
55         print("start programm")
56         ap.preflight() # отдаем команду выполнения предстартовой
           ↪ подготовки
57         once = True
58     pass
```

Подзадача БПЛА-4. Составить карту поврежденных участков

Условие

Под картой подразумевается набор данных, позволяющий построить изображение полигона с отмеченными поврежденными участками дорог. У каждого участка необходимо:

- обозначить тип повреждений;
- указать порядковый номер;
- отметить число посещений беспилотным автомобилем.

Поврежденные участки должны быть обозначены так, чтобы было ясно, на какой стороне дороги повреждение и с точностью до одного пунктирного штриха разметки понятно его положение.

Порядок действий:

1. Участники сообщают, какие типы участков дороги обнаруживает и распознает их квадрокоптер.
2. Организатор случайным образом размещает четыре поврежденных участка дороги в полетной зоне квадроптера.
3. Квадрокоптер устанавливается на стартовую площадку.
4. По команде организатора участники запускают программу.

Подзадача считается полностью решенной, если последовательно выполнены следующие действия:

- БПЛА взлетел со стартовой площадки на произвольную высоту;
- квадрокоптер облетел всю доступную дорожную сеть;
- БПЛА приземлился на крыше здания и после полной остановки остался в ее пределах;
- на компьютере оператора появилась карта полигона, соответствующая реальному расположению участков.

Критерии оценивания подзадачи:

+1 балл — за каждый участок, верно обозначенный на карте, учитывается положение участка и его тип.

+1 балл — за каждый выбранный участниками тип поврежденных участков.

2 балла — за посадку.

Должны быть предоставлены три набора статистических данных.

Решение

Файлы с решениями подзадач находятся в папке: https://disk.yandex.ru/d/NJ2o0pOxDW162A.

Участникам было предоставлено изображение полигона 1 208×889 px (пикселей).

Красным прямоугольником выделена полетная зона квадрокоптера.

Для выполнения задачи необходимо выполнить привязку координат ультразвуковой навигационной системы к пикселям изображения. Размер зоны задается системой навигации «Геоскан Локус» в метрах, где 0 координат находится в нижнем левом углу, ось X направлена горизонтально, а ось Y — вертикально.

Полетная зона имеет размеры 3×6 м. Следовательно, длина 1 px (пикселя) изображения составляет приблизительно 0,0067 м в системе координат ультразвуковой навигации. Эта величина является константой и должна использоваться для расчета местоположения повреждения.

Для точного указания координат объекта требуется модификация функции его поиска таким образом, чтобы она возвращала смещение центра объекта относительно центра камеры. Полученные данные необходимо перевести в метры и прибавить к текущим координатам квадрокоптера.

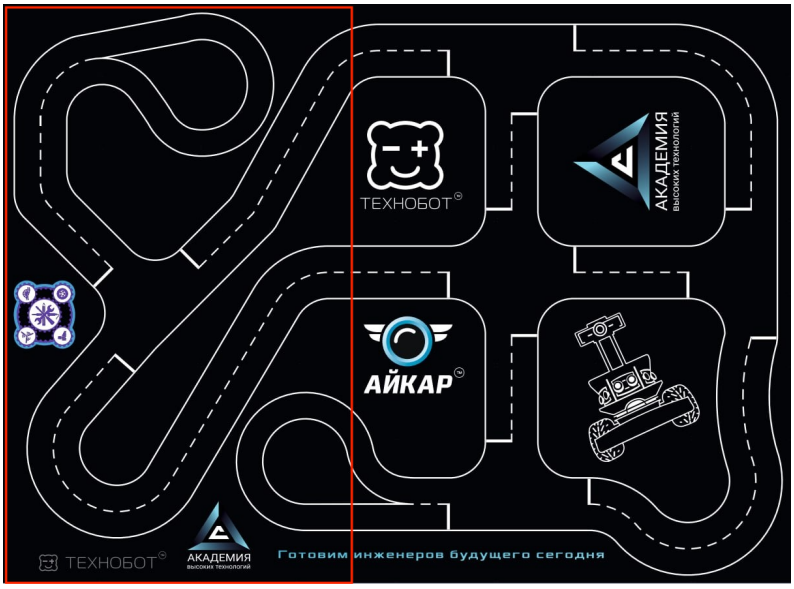

Рис. 4.3.3. Полетная зона квадрокоптера

Модифицированная функция поиска объекта.

Python

```python
# Выдает самый вероятный объект на изображении и координаты центра объекта относительно центра изображения с камеры
def get_most_probable_class(class_ids, confidences, boxes, image_shape):
    if len(class_ids) == 0 or len(confidences) == 0 or len(boxes) == 0:
        return None, None, None  # Если объекты не найдены, возвращаем (None, None, None)
    # Находим индекс максимальной уверенности
    most_probable_index = np.argmax(confidences)
    # Получаем соответствующий class_id и box
    class_id = class_ids[most_probable_index]
    box = boxes[most_probable_index]
    # Рассчитываем центр рамки
    x, y, w, h = box  # Координаты рамки: (x, y, ширина, высота)
    box_center_x = x + w / 2
    box_center_y = y + h / 2
    # Рассчитываем центр изображения
    image_height, image_width = image_shape
    image_center_x = image_width / 2
    image_center_y = image_height / 2
    # Рассчитываем смещение центра рамки относительно центра изображения
    center_offset_x = box_center_x - image_center_x
    center_offset_y = box_center_y - image_center_y
    center_offset = (center_offset_x, center_offset_y)
    return class_id, center_offset
```

Необходимо разработать функцию для перевода смещения из пикселей в метры. Для этого следует использовать высоту квадрокоптера в момент съемки, а также угол обзора камеры. На квадрокоптере «Геоскан Пионер» в модификации НТО установлена камера Raspberry Pi Camera Module 2, которая имеет следующие углы обзора: вертикальный угол – 48,8°, горизонтальный угол – 62,2°.

Функция конвертации смещения из пикселей в метры будет использовать данные о высоте квадрокоптера и углах обзора камеры.

Функция конвертации смещения из пикселей в метры.

Python

```python
def convert_pixels_to_meters(pixel_offset_x, pixel_offset_y, camera_height, camera_fov_h, camera_fov_v):
    # Переводим углы обзора из градусов в радианы
    fov_h_rad = np.radians(camera_fov_h)
    fov_v_rad = np.radians(camera_fov_v)

    # Рассчитываем фокусные расстояния камеры в пикселях
    focal_length_h = camera_height / np.tan(fov_h_rad / 2)
    focal_length_v = camera_height / np.tan(fov_v_rad / 2)

    # Переводим смещение из пикселей в метры
    x_meters = (pixel_offset_x * camera_height) / focal_length_h
    y_meters = (pixel_offset_y * camera_height) / focal_length_v

    return x_meters, y_meters
```

Для создания карты недостаточно сохранять только метку класса объекта, необходимо фиксировать и его положение. Для хранения карты необходимо использовать

сервер из БПА-4. Функция упаковки данных в словарь.

Python

```python
from gs_navigation import NavigationManager
PIXEL_CONST = 0.0067
# Инициализирует NavigationManager для получения координат
nav = NavigationManager()
# Создает словарь с классом и смещением
def create_dict(class_id, object_offset, number):
    global nav
    if class_id is None:
        return None
    else:
        position = nav.lps.position()
        object_offset_x, object_offset_y = convert_pixels_to_meters(
            *object_offset,
            position.z,
            62.2,
            48.8
        )
        return {
            "type": class_id,
            "number": number,
            "count" : 0,
            "x": (object_offset_x + position.x) / PIXEL_CONST,
            "y": (object_offset_y + position.y) / PIXEL_CONST
        }
```

Модифицированная функция детектирования объекта.

Python

```python
# Считает количество элементов в списке, которые не равны None.
def count_not_none(all_objects):
  return len(list(filter(lambda x: x is not None, all_objects)))
def detect_handler(request):
    global bridge
    global yolo_model
    global all_object

    image_msg = rospy.wait_for_message("/pioneer_max_camera/image_raw",
    ↪  Image)
    try:
        cv_image = bridge.imgmsg_to_cv2(image_msg, "bgr8")
        class_ids, scores, boxes = yolo_model.detect(cv_image, 0.6,
        ↪  0.4)
        most_class, object_offset = get_most_probable_class(
            class_ids,
            scores,
            boxes,
            cv_image.shape
        )
        object_dict = create_dict(
            most_class,
            object_offset,
            count_not_none(all_object)
        )
        if (object_dict is None) or (len(all_object) == 0):
            all_object.append(object_dict)
        elif all_object[-1] is None:
```

```
            all_object.append(object_dict)
        elif object_dict["type"] == all_object[-1]["type"]:
            all_object.append(None)
        else:
            all_object.append(object_dict)
        if all_object[-1] is not None:
            print(f"Обнаруженный тип повреждения: {all_object[-1]}")
    except:
        pass
    return EmptyResponse()
```

Функция отправки данных на сервер:

Python

```
import requests
SERVER_URL = "http://172.16.65.50:8000"
def send_points(points):
    ans = requests.post(f"{SERVER_URL}/points", json=points)
    if ans.status_code != 200:
        return False
    return ans.json()["status"] == "OK"
```

Необходимо создать сервис, который будет отправлять данные на сервер после окончания посадки для обновления карты.

Python

```
def send_points_handler(request):
    global all_object
    send_points(all_object)
    return EmptyResponse()
send_points_server = Service("/send_points", Empty,
 ↪  send_points_handler)  # создаем сервис для отправки точек
```

Следует изменить код полетного задания, чтобы после посадки совершалась отправка данных на сервер (корректировка коснется только функции обработки событий автопилота).

Python

```
def callback(event): # функция обработки событий Автопилота
    global ap
    global run
    global coordinates
    global position_number
    global detect_client
    event = event.data
    if event == CallbackEvent.ENGINES_STARTED: # блок обработки события
     ↪  запуска двигателя
        print("engine started")
        ap.takeoff() # отдаем команду взлета
    elif event == CallbackEvent.TAKEOFF_COMPLETE: # блок обработки
     ↪  события завершения взлета
        print("takeoff complete")
        position_number = 0
        ap.goToLocalPoint(coordinates[position_number][0],
                          coordinates[position_number][1],
```

```
                            coordinates[position_number][2]) # отдаем
                             ↪  команду полета в точку
    elif event == CallbackEvent.POINT_REACHED: # блок обработки события
     ↪  достижения точки
        print(f"point {position_number} reached")
        detect_client()
        position_number += 1 # наращиваем счетчик точек
        if position_number < len(coordinates): # проверяем количество
         ↪  текущее количество точек с количеством точек в полетном
         ↪  задании
            ap.goToLocalPoint(coordinates[position_number][0],
                              coordinates[position_number][1],
                              coordinates[position_number][2]) # отдаем
                               ↪  команду полета в точку
        else:
            ap.landing() # отдаем команду посадки
    elif event == CallbackEvent.COPTER_LANDED: # блок обработки события
     ↪  приземления
        print("finish programm")
        send_points_client()
        run = False # прекращем программу
    send_points_client = ServiceProxy("/send_points", Empty) # клиент
     ↪  сервиса распознавания
```

Подзадача БПЛА-5. Обновить данные о конкретном участке дороги

Условие

Порядок действий:

1. Участники демонстрируют организатору карту полигона с возможностью ручного добавления и удаления поврежденных участков дороги (участки на карте должны быть пронумерованы).
2. Организатор случайным образом размещает на полигоне поврежденные участки.
3. Участники наносят поврежденные участки на карту.
4. Квадрокоптер устанавливается на стартовую площадку.
5. По команде организатора участники запускают программу.
6. Организатор сообщает номер участка, данные о котором необходимо обновить.
7. Участники передают номер участка в программу через стандартный поток ввода.
8. Квадрокоптер должен сфотографировать поврежденный участок и передать его фото MQTT-серверу на обработку.
9. Обработанное фото направляется в ответ.
10. На компьютере оператора выводится исходное и обработанное фото.

Подзадача считается полностью решенной, если последовательно выполнены следующие условия:

- БПЛА взлетел со стартовой площадки на произвольную высоту;
- квадрокоптер долетел и завис над выбранным участком;

- на компьютере оператора отобразились исходное и обработанное изображение (центр исходного изображения находится в пределах поврежденного участка дороги);
- БПЛА приземлился на крыше здания и после полной остановки остался в ее пределах.

Критерии оценивания подзадачи:

- *4 балла — на компьютере оператора выведено изображение целевого участка.*
- *2 балла — за посадку.*

Должен быть предоставлен один набор статистических данных.

Решение

Файлы с решениями подзадач находятся в папке:https://disk.yandex.ru/d/NJ2o0pOxDW162A.

Для выполнения подзадачи следует внести изменения в сервер, разработанный в рамках подзадачи БПА-4. Требуется добавить функцию постановки задачи квадрокоптеру и получения этой задачи, а также функцию добавления и сохранения изображений, необходимых для выполнения подзадачи.

Python

```python
class ImageRequest(BaseModel):
  original: str
  processed: str
last_images = None
@app.post("/copter_task")
async def post_copter_task(request: PointsRequest):
  global copter_task
  copter_task = request
  return {"status": "OK", "detail": None}
@app.get("/copter_task")
async def get_copter_task():
  global copter_task
  return {"status": "OK", "detail": copter_task}
@app.post("/images")
async def post_images(request: ImageRequest):
  global last_images
  last_images = request
  return {"status": "OK", "detail": None}
@app.get("/images")
async def get_images():
  global last_images
  return {"status": "OK", "detail": last_images}
```

Необходимо модифицировать код приложения карты, разработанного в рамках подзадачи БПА-5. Требуется добавить функции отправки данных о полетной задаче на сервер в класс `Server`.

Python

```python
def post_copter_task(self, task: dict) -> bool:
      ans = requests.post(f"{self.base_url}/copter_task", json=task)
      if ans.status_code != 200:
          return False
      return ans.json()["status"] == "OK"
```

Далее нужно изменить функцию `mouse_callback`, переназначив правую кнопку мыши на отправку задания квадрокоптеру.

Python

```python
if event == cv2.EVENT_RBUTTONDOWN:
      id = input("Введите id точки")
      map_point = server.get_points()
      task = None
      for point in map_point:
          if point["id"] == id:
              task = point
      if task is not None:
          server.post_copter_task(task)
```

В полетном задании перед запуском двигателя необходимо добавить ожидание полетной задачи. Изменения приведены ниже.

Python

```python
import requests
coordinates = [
   HOME_POINT
]
def callback(event): # функция обработки событий Автопилота
   global ap
   global run
   global coordinates
   global position_number
   global detect_client
   event = event.data
   if event == CallbackEvent.ENGINES_STARTED: # блок обработки события
    ↪ запуска двигателя
      print("engine started")
      ap.takeoff() # отдаем команду взлета
   elif event == CallbackEvent.TAKEOFF_COMPLETE: # блок обработки
    ↪ события завершения взлета
      print("takeoff complete")
      position_number = 0
      ap.goToLocalPoint(coordinates[position_number][0],
       ↪ coordinates[position_number][1],
       ↪ coordinates[position_number][2]) # отдаем команду полета в
       ↪ точку
   elif event == CallbackEvent.POINT_REACHED: # блок обработки события
    ↪ достижения точки
      print(f"point {position_number} reached")
      position_number += 1 # наращиваем счетчик точек
      if position_number < len(coordinates): # проверяем количество
       ↪ текущее количество точек с количеством точек в полетном
       ↪ задании
          ap.goToLocalPoint(coordinates[position_number][0],
                            coordinates[position_number][1],
```

```
                                coordinates[position_number][2]) # отдаем
                                ↪  команду полета в точку
        elif position_number == len(coordinates):
            detect_client()
            ap.goToLocalPoint(*HOME_POINT)
        else:
            ap.landing() # отдаем команду посадки
    elif event == CallbackEvent.COPTER_LANDED: # блок обработки события
    ↪  приземления
        print("finish programm")
        send_points_client()
        run = False # прекращаем программу
SERVER_URL = "http://172.16.65.50:8000"
def get_copter_task():
    ans = requests.get(f"{SERVER_URL}/copter_task")
    if ans.status_code != 200:
        return None
    return ans.json()["detail"]
PIXEL_CONST = 0.0067
while not rospy.is_shutdown() and run:
    if board.runStatus() and not once: # проверка подключения RPi к
    ↪  Пионеру
        print("start programm")
        while True:
            task = get_copter_task()
            if task is not None:
                coordinates.append([task["x"] * PIXEL_CONST, task["y"]
                ↪  * PIXEL_CONST, HOME_POINT[2]])
                break
        ap.preflight() # отдаем команду выполнения предстартовой
        ↪  подготовки
        once = True
    pass
```

Необходимо внести изменения в узел обнаружения дефектов. Требуется удалить функциональность сохранения объектов и добавить запрос к MQTT-серверу для получения обработанного изображения, а также реализовать отправку изображения на общий сервер. Функция отправки двух изображений на общий сервер.

Python

```
import base64
def send_image_to_server(original_image, processed_image):
    # Кодируем исходное изображение в JPEG
    _, buffer_orig = cv2.imencode('.jpg', original_image)
    # Кодируем обработанное изображение в JPEG
    _, buffer_proc = cv2.imencode('.jpg', processed_image)

    # Конвертируем в base64
    img_orig_base64 = base64.b64encode(buffer_orig).decode('utf-8')
    img_proc_base64 = base64.b64encode(buffer_proc).decode('utf-8')

    # Отправляем на сервер
    response = requests.post(
        f"{SERVER_URL}/images",
        json={
            "original": img_orig_base64,
            "processed": img_proc_base64
        }
```

```
    )
    return response.status_code == 200
```

Исправленная функция сервиса детекции.

Python

```python
def detect_handler(request):
    global bridge

    image_msg = rospy.wait_for_message("/pioneer_max_camera/image_raw", Image)
    try:
        cv_image = bridge.imgmsg_to_cv2(image_msg, "bgr8")
        client.send(cv_image)
        output = client.get_output()

        if output is not None:
            send_image_to_server(cv_image, output)

    except Exception as e:
        print(f"Ошибка в detect_handler: {str(e)}")
    return EmptyResponse()
```

Необходимо разработать скрипт, который будет запускаться на компьютере оператора и выводить изображения, полученные с квадрокоптера. Код программы приведен ниже.

Python

```python
import cv2
import numpy as np
import requests
import base64
import time
SERVER_URL = "http://172.16.65.50:8000"
def base64_to_image(base64_string):
    # Декодируем base64 строку в байты
    img_data = base64.b64decode(base64_string)
    # Преобразуем байты в numpy массив
    nparr = np.frombuffer(img_data, np.uint8)
    img = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
    return img
def main():
    # URL сервера

    while True:
        try:
            # Получаем данные с сервера
            response = requests.get(f"{SERVER_URL}/images")
            data = response.json()

            if data["status"] == "OK" and data["detail"] is not None:
                # Получаем изображения
                original_img = base64_to_image(data["detail"]["original"])
                processed_img = base64_to_image(data["detail"]["processed"])

                # Создаем окно с двумя изображениями рядом
```

```
                combined = np.hstack((original_img, processed_img))

                # Показываем изображения
                cv2.imshow("Original and Processed Images", combined)

                # Ждем нажатия клавиши (1 мс)
                if cv2.waitKey(1) & 0xFF == ord('q'):
                    break

            # Небольшая задержка перед следующим запросом
            time.sleep(0.1)

        except Exception as e:
            print(f"Ошибка: {e}")
            time.sleep(1)  # Ждем секунду перед повторной попыткой

    cv2.destroyAllWindows()
if __name__ == "__main__":
    main()
```

Финальные испытания

Совместный запуск устройств

Решение

Файлы с решениями подзадач находятся в папке: https://disk.yandex.ru/d/NJ2o0pOxDW162A.

Для совместного запуска устройств необходимо добавить функцию записи и передачи логического флага, который будет указывать, что необходимо запустить все устройства. Изменения будут реализованы относительно сервера, разработанного в подзадаче БПЛА-5.

Python

```
class BooleanRequest(BaseModel):
  value: bool
start_value = False
@app.post("/start")
async def post_start(request: BooleanRequest):
    global start_value
    start_value = request.value
    return {"status": "OK", "detail": None}
@app.get("/start")
async def get_start():
    global start_value
    return {"status": "OK", "detail": start_value}
```

В код полетного задания квадрокоптера необходимо добавить получение логического флага старта. Требуется реализовать функцию запроса флага и модифицировать основной цикл полетного задания, пример для которого приведен в подзадаче БПЛА-5.

Python

```python
def get_start():
    ans = requests.get(f"{SERVER_URL}/start")
    if ans.status_code != 200:
        return False
    return ans.json()["detail"]
while not rospy.is_shutdown() and run:
    if board.runStatus() and not once: # проверка подключения RPi к Пионеру
        print("start programm")
        while True:
            if get_start():  # ожидаем сигнала старта
                task = get_copter_task()
                if task is not None:
                    coordinates.append([task["x"] * PIXEL_CONST, task["y"] * PIXEL_CONST, HOME_POINT[2]])
                    break
            rospy.sleep(0.1)  # небольшая задержка между проверками
        ap.preflight() # отдаем команду выполнения предстартовой подготовки
        once = True
    pass
```

Для отправки команды старта с компьютера оператора можно использовать приведенную ниже программу, которая ожидает ввода команды с клавиатуры.

Python

```python
import requests
SERVER_URL = "http://172.16.65.50:8000"
def send_start_command():
    try:
        response = requests.post(f"{SERVER_URL}/start", json={"value": True})
        if response.status_code == 200:
            print("Команда старта успешно отправлена")
            return True
        else:
            print(f"Ошибка при отправке команды: {response.status_code}")
            return False
    except Exception as e:
        print(f"Ошибка при отправке команды: {e}")
        return False
def main():
    print("Ожидание ввода команды 'start'...")
    while True:
        command = input("Введите команду: ").strip().lower()
        if command == "start":
            if send_start_command():
                print("Программа завершена")
                break
        elif command == "exit":
            print("Программа завершена")
            break
        else:
            print("Неизвестная команда. Используйте 'start' или 'exit'")
if __name__ == "__main__":
    main()
```

4.3.6. Материалы для подготовки

1. Программа подготовки к профилю «Автономные транспортные системы» Национальной технологической олимпиады [Электронный ресурс]. — Режим доступа: https://avt.global/nto_program.
2. Задачи профиля «Автономные транспортные системы», 2022–2023 [Электронный ресурс]. — Режим доступа: https://ntcontest.ru/docs/ats-assignements1.pdf.
3. Задачи профиля «Автономные транспортные системы», 2021–2022 [Электронный ресурс]. — Режим доступа: https://ntcontest.ru/docs/ats-assignements.pdf.
4. Задачи профиля «Автономные транспортные системы», 2020–2021 [Электронный ресурс]. — Режим доступа: https://drive.google.com/file/d/1jJwI_5MgX-wvmwK7WUh_mhQrEcFAhB_K/view.
5. Руководство по OpenCV [Электронный ресурс]. — Режим доступа: https://docs.opencv.org/4.x/d9/df8/tutorial_root.html.
6. SDK для программирования квадрокоптера Пионер модификации НТО [Электронный ресурс]. — Режим доступа: https://github.com/geoscan/geoscan_pioneer_max.

5. Критерии определения победителей и призеров

Первый отборочный этап

В первом отборочном этапе участники решали задачи предметного тура по двум предметам: физике и информатике и инженерного тура. В каждом предмете максимально можно было набрать 100 баллов, в инженерном туре 100 баллов. Для того чтобы пройти во второй этап, участники должны были набрать в сумме по обоим предметам и инженерному туру не менее 30,0 баллов, независимо от уровня.

Второй отборочный этап

Количество баллов, набранных при решении всех задач второго отборочного этапа, суммируется. Победители второго отборочного этапа должны были набрать не менее 92,0 баллов, независимо от уровня.

Заключительный этап

Индивидуальный предметный тур

- физика — максимально возможный балл за все задачи — 100 баллов;
- информатика — максимально возможный балл за все задачи — 100 баллов.

Командный инженерный тур

Команды заключительного этапа получали за командный инженерный тур от 0 до 100,00 баллов: команда, набравшая наибольшее число баллов среди других команд, становилась командой-победителем.

Все результаты команд нормировались по формуле:

$$\frac{100 \times x}{MAX},$$

где x — число баллов, набранных командой,

MAX — число баллов, максимально возможное за инженерный тур.

В заключительном этапе олимпиады индивидуальные баллы участника складываются из двух частей, каждая из которых имеет собственный вес: баллы за индивидуальное решение задач по предмету 1 (физика) с весом $K_1 = 0{,}15$, по предмету 2

(информатика) с весом $K_2 = 0{,}15$, баллы за командное решение задач инженерного тура с весом $K_3 = 0{,}7$.

Итоговый балл определяется по формуле:

$$S = K_1 \cdot S_1 + K_2 \cdot S_2 + K_3 \cdot S_3,$$

где S_1 — балл первой части заключительного этапа по физике (предметный тур) ($S_{1\ \text{макс}} = 100$);

S_2 — балл первой части заключительного этапа по информатике (предметный тур) ($S_{2\ \text{макс}} = 100$);

S_3 — итоговый балл инженерного командного тура ($S_{3\ \text{макс}} = 175$).

Итого максимально возможный индивидуальный балл участника заключительного этапа $= 152{,}5$ баллов.

Критерий определения победителей и призеров

Чтобы определить победителей и призеров (независимо от класса) на основе индивидуальных результатов участников, был сформирован общий рейтинг всех участников заключительного этапа. С начала рейтинга были выбраны 3 победителя и 7 призеров (первые 25% участников рейтинга становятся победителями или призерами, из них первые 8% становятся победителями, оставшиеся — призерами).

Критерий определения победителей и призеров (независимо от уровня)

Категория	**Количество баллов**
Победители	63,80 и выше
Призеры	От 46,55 до 63,00

6. Работа наставника после НТО

Участие школьника в Олимпиаде может завершиться после любого из этапов: первого или второго отборочных, либо после заключительного этапа. В каждом случае после завершения участия наставнику необходимо провести с учениками рефлексию — обсудить полученный опыт и проанализировать, что позволило достичь успеха, а что привело к неудаче. Подробные материалы о проведении рефлексии представлены в курсе «Наставник НТО»: https://academy.sk.ru/events/310.

Наставнику важно проинформировать руководство образовательного учреждения, если его учащиеся стали финалистами, призерами и победителями. Публичное признание высоких результатов дополнительно повышает мотивацию.

В процессе рефлексии с учениками, не ставшими призерами или победителями, рекомендуется уделить особое внимание особенностям командной работы: распределению ролей, планированию работы, возникающим проблемам. Для этого могут использоваться опросники для самооценки собственной работы и взаимной оценки участниками других членов команды (P2P). Они могут выявить внутренние проблемы команды, для решения которых в план подготовки можно добавить мероприятия, направленные на ее сплочение.

Стоит рассказать, что в истории НТО было много примеров, когда не победив в первый раз, на следующий год участники показывали впечатляющие результаты, одержав победу сразу в нескольких профилях. Конечно, важно отметить, что так происходит только при учете прошлых ошибок и подготовке к Олимпиаде в течение года.

Важным фактором успешного участия в следующих сезонах НТО может стать поддержка родителей учеников. Знакомство с ними помогает наставнику продемонстрировать важность компетенций, развиваемых в процессе участия в НТО, для будущего образования и карьеры школьников. Поддержка родителей помогает мотивировать участников и позволяет выделить необходимое время на занятия в кружке.

С участниками-выпускниками наставнику рекомендуется обсудить их дальнейшее профессиональное развитие и его связь с выбранными профилями НТО. Отдельно можно обратить внимание на льготы для победителей и призеров, предлагаемые в вузах с интересующими ученика направлениями. Кроме того, ряд вузов предлагает льготы для всех финалистов НТО, а также учитывает результаты Конкурса цифровых портфолио «Талант НТО».