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

по профилю «Умный город»

2024/25 учебный год

ntcontest.ru

УДК 373.5.016:[711.4:004]
ББК 74.263.2
У54

Авторы:
П. Ф. Баранов, А. В. Бывшенко, О. В. Зубков, А. А. Коломейцев, Н. А. Кузляев, Н. Ю. Кузнецов, С. В. Обухов

У54 Всероссийская междисциплинарная олимпиада школьников 8–11 класса «Национальная технологическая олимпиада». Учебно-методическое пособие
Том 30 **Умный город**
— М.: Ассоциация участников технологических кружков, 2025. — 213 с.

ISBN 978-5-908021-29-6

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:
02.00.00 Компьютерные и информационные науки
09.00.00 Информатика и вычислительная техника
10.00.00 Информационная безопасность
11.00.00 Электроника, радиотехника и системы связи
12.00.00 Фотоника, приборостроение, оптические и биотехнические системы и технологии
27.00.00 Управление в технических системах

ISBN 978-5-908021-29-6
9 785908 021296 >

УДК 373.5.016:[711.4:004]
ББК 74.263.2

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. Умный город

Профиль Умный город Национальной технологической олимпиады посвящен практическому применению знаний электроники и программирования микроконтроллеров в сфере повышения удобства пользования городской и промышленной инфраструктурой с применением цифровых технологий. Профиль направлен на решение задач в области оптимизации, доступности, автоматизации, экономической эффективности и безопасности городской, коммунальной и промышленной инфраструктуры.

Для успешного решения комплексной задачи, которая стоит перед школьниками, им необходимы прочные знания по физике и информатике.

Заключительный этап Олимпиады является логическим завершением подготовительного цикла, включающего два отборочных этапа: индивидуальный и командный.

Знакомство с профилем Умный город начинается с вводного урока, где обсуждается морфологический анализ при разработке новых идей и технологических решений. Школьники разрабатывают концепцию умного пешеходного перехода и погружаются в проблемное и содержательное поле задач профиля. Материалы к уроку можно скачать по ссылке: https://nto-lesson.ru.

Первый отборочный этап состоит из предметного и инженерного туров. Отборочный тур проводится индивидуально дистанционно на платформе Яндекс.Контест. Предметный включает решение задач по профильным предметам — физике и информатике. Данный этап нацелен на отбор участников, обладающих базовыми знаниями по названным дисциплинам.

В ходе инженерного тура первого отборочного этапа проверяются базовые навыки:

- программирования;
- решения алгоритмических задач;
- применения языков программирования для решения физических задач.

Участники решают базовые задачи из области электротехники и электроники — это закладывает фундамент для для успешного прохождения последующих этапов.

Цель **второго отборочного этапа** — выявление наиболее подготовленных школьников для участия в заключительном этапе, а также углубление в тематику профиля. Данный этап предполагает более сложные задачи, справляться с которыми эффективнее в команде.

Таким образом, в ходе второго этапа участники приобретают дополнительные знания и опыт командной работы, необходимые на заключительном этапе. Именно здесь формируются команды-финалисты.

Задачи второго этапа посвящены актуализации основ радиоэлектроники и электрических цепей, поэтому участникам предлагаются задания, мотивирующие к изучению языков программирования и алгоритмов, приобретению навыков декомпозиции комплексных задач, что позволяет выделить из них наиболее подготовленных.

Для команд-финалистов в качестве подготовительного мероприятия проводится

онлайн-хакатон, предоставляющий возможность потренироваться в программировании микроконтроллеров и попробовать в действии различные датчики, с которыми предстоит работать непосредственно на заключительном этапе.

В командном практическом туре заключительного этапа участникам необходимо продемонстрировать свои компетенции и применить навыки, полученные на предыдущих этапах.

Технологическая задача заключительного этапа — создание микрорайона в составе умного города с транспортной системой, включающей в себя умные светофоры и фонари, парковки и шлагбаумы. Это напрямую коррелирует с технологиями умного города, а именно, с созданием единых ресурсов обработки информации внутри него. Разбор задачи помогает участникам лучше понять существующие проблемы, возможности их решения и способы комплексного повышения эффективности городской инфраструктуры.

Для отладки решений за каждой командой закрепляется макет микрорайона, монтирующийся в полигон умного города. Полигон включает трассу с черной линией и места для парковки электрокаров, железную дорогу, проходящую через микрорайоны всех команд.

Команды подключают комплектующие согласно спроектированным схемам. Все элементы должны взаимодействовать с локальным сервером, который, в свою очередь, коммуницирует с центральным городским сервером. Справиться с такой комплексной инженерной задачей в установленные сроки под силу только хорошо подготовленной и слаженной команде, члены которой обладают навыками в программировании и электронике, а также владеют тайм-менеджментом.

Кроме командной задачи участники должны решить задачи повышенной сложности по физике и информатике на индивидуальном предметном туре заключительного этапа. Это позволяет им продемонстрировать свои теоретические знания, умения и уровень подготовки.

Приобретенные в профиле компетенции помогают выпускникам реализовывать свои проекты в области интеллектуальной прикладной электроники и интернета вещей. Полученные навыки отвечают современным трендам экономики — востребованности в создании интегрированных компьютерных систем управления.

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;
    }
}
```

```
16 int f2(int M, int b){
17     return M + M / b;
18 }
19 signed main(){
20     int a, b, c;
21     cin >> a >> b >> c;
22     int L = 0, R = 1;
23     while(f1(R, a) <= c){
24         R *= 2;
25     }
26     while(R - L > 1){
27         int M = (R + L) / 2;
28         if(f1(M, a) < c){
29             L = M;
30         }
31         else{
32             R = M;
33         }
34     }
35     int z = R;
36     L = 0, R = 1;
37     while(f2(R, b) <= z){
38         R *= 2;
39     }
40     while(R - L > 1){
41         int M = (R + L) / 2;
42         if(f2(M, b) < z){
43             L = M;
44         }
45         else{
46             R = M;
47         }
48     }
49     cout << R << endl;
50 }
```

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});
    }
```

```
25      int s;
26      cin >> s;
27      dfs(s);
28      reverse(order.begin(), order.end());
29      vector<int> dp(n + 1, -1);
30      dp[s] = 0;
31      for(auto el : order){
32          for(auto to : G[el]){
33              dp[to.first] = max(dp[to.first], dp[el] + to.second);
34          }
35      }
36      for(int i = 1; i <= n; i++){
37          cout << dp[i] <<' ';
38      }
39  }
```

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 $n \times m$.

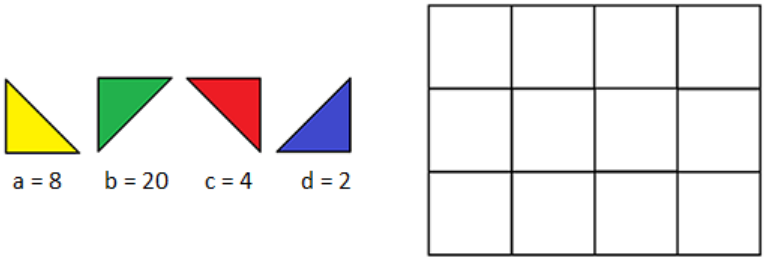

Рис. 2.2.1

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.

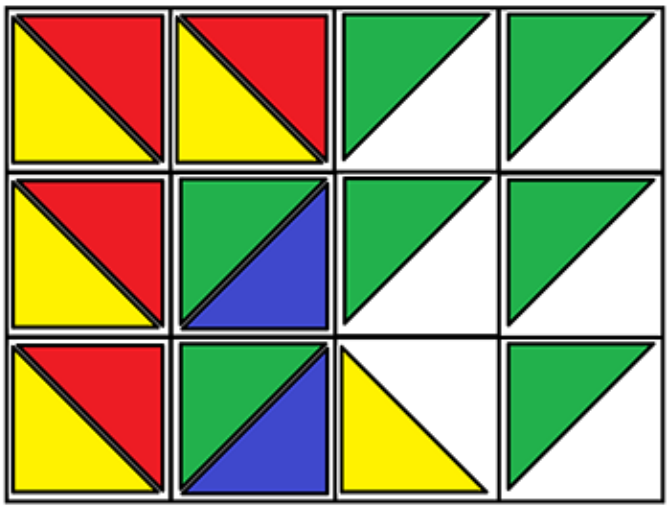

Рис. 2.2.2

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#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.

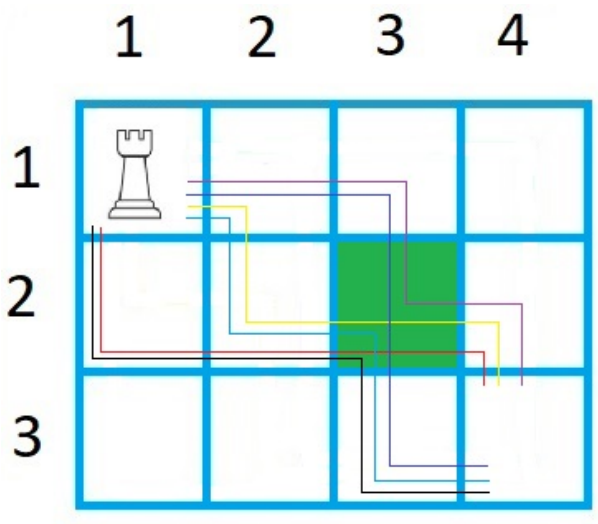

Рис. 2.2.3

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

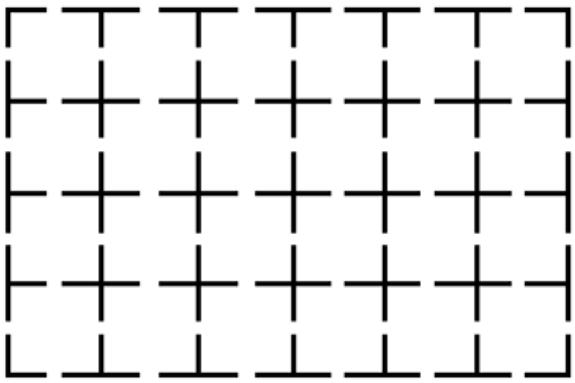

Рис. 2.2.4

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.

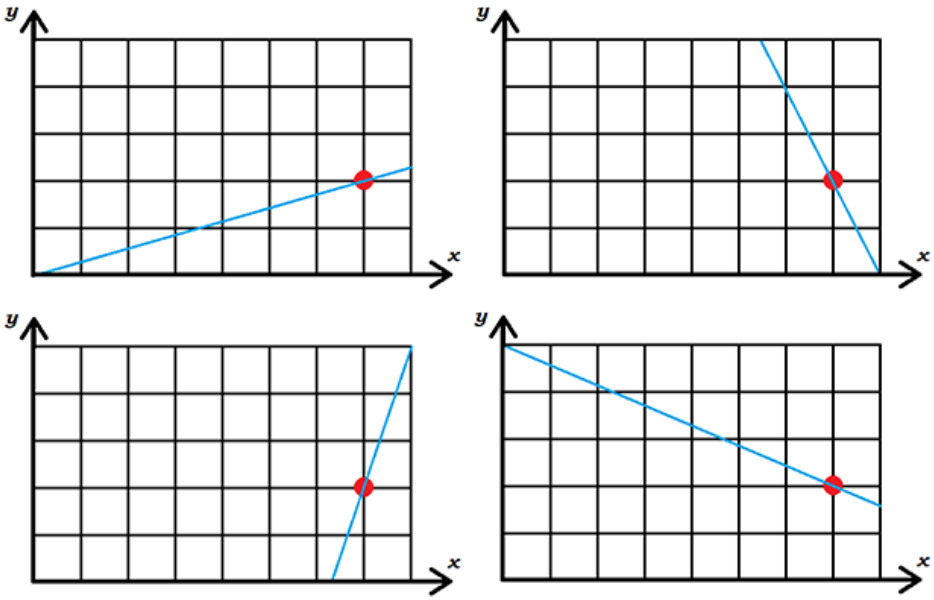

Рис. 2.2.5

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.

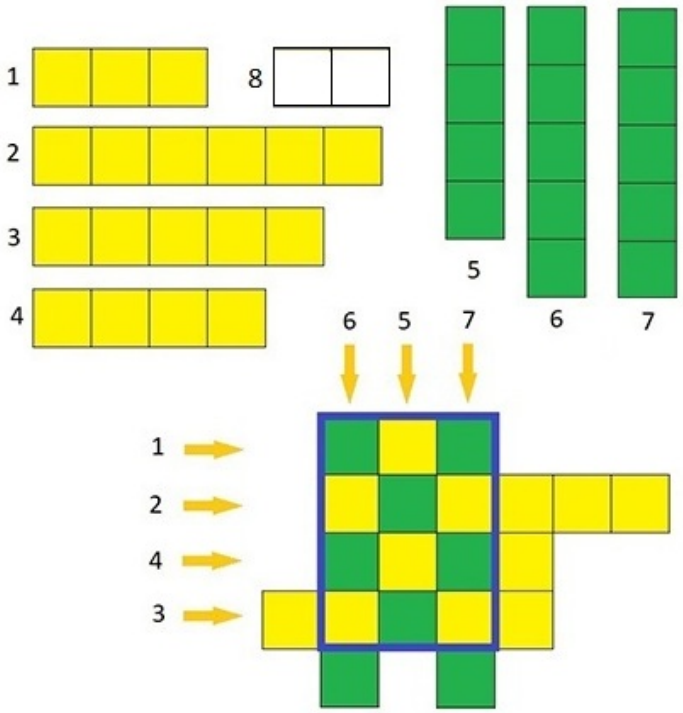

Рис. 2.2.6

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

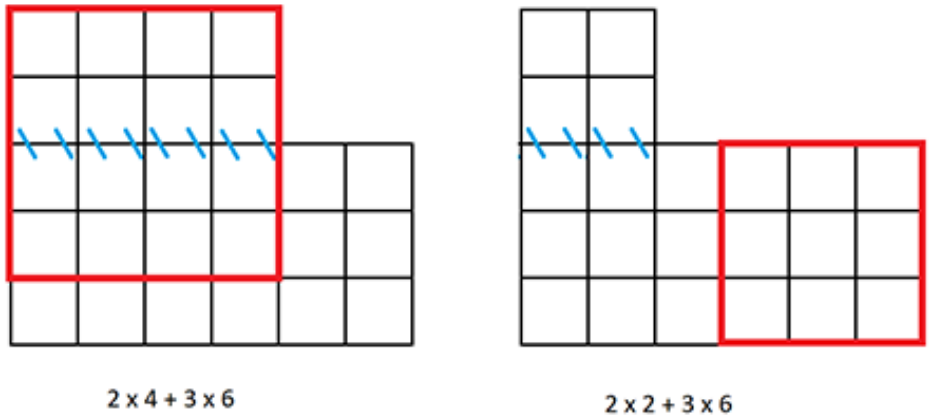

Рис. 2.2.7

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.

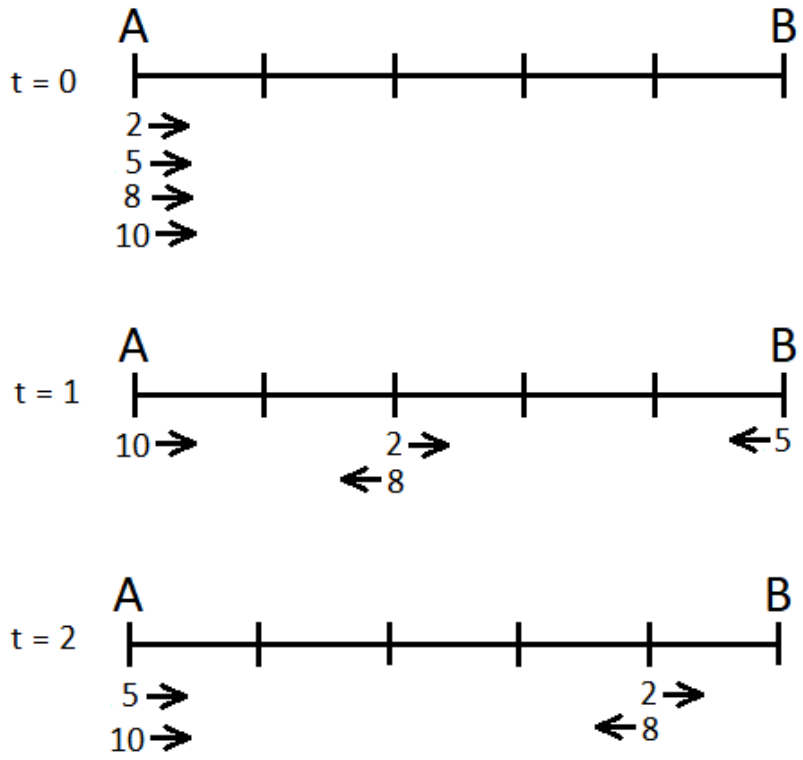

Рис. 2.2.8

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?

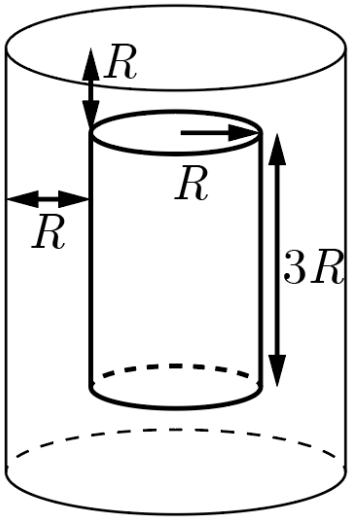

Рис. 2.3.1

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.

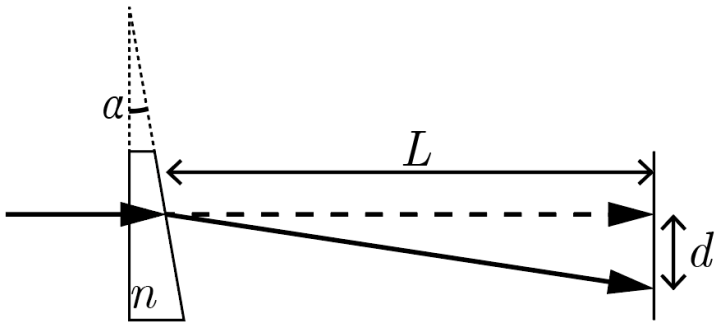

Рис. 2.3.2

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

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

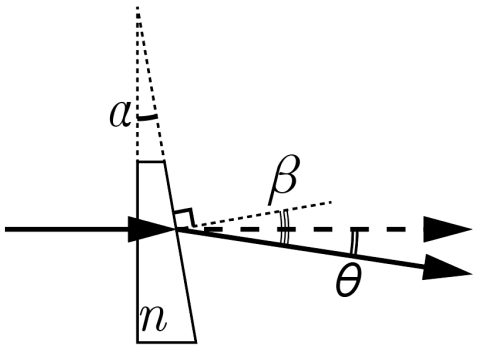

Рис. 2.3.3

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.

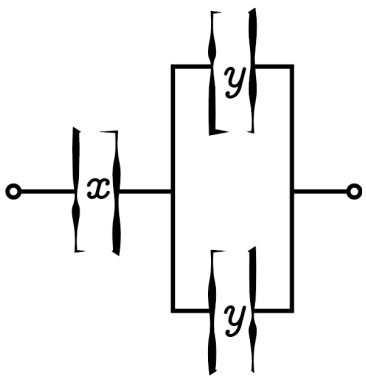

Рис. 2.3.4

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.

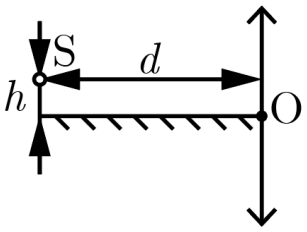

Рис. 2.3.5

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).

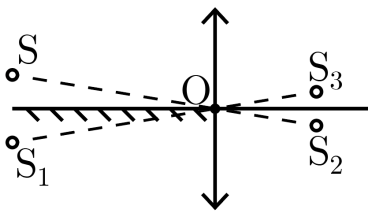

Рис. 2.3.6

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.

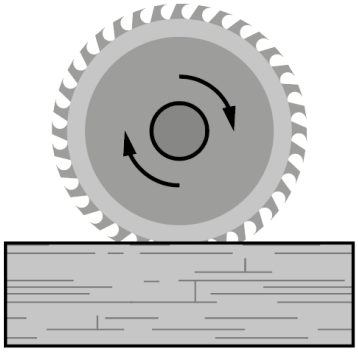

Рис. 2.3.7

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.

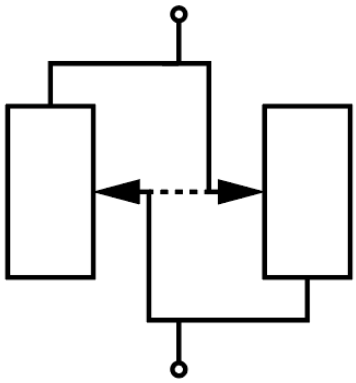

Рис. 2.3.8

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.

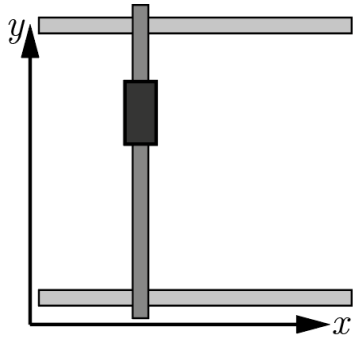

Рис. 2.3.9

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?

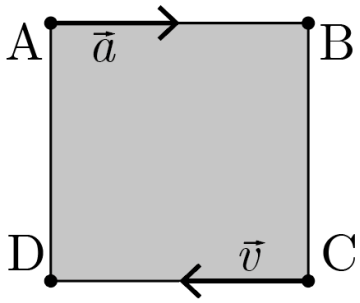

Рис. 2.3.10

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.

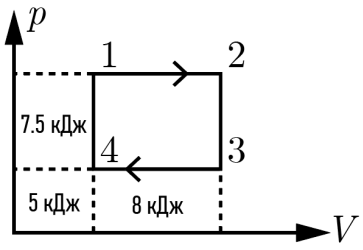

Рис. 2.3.11

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?

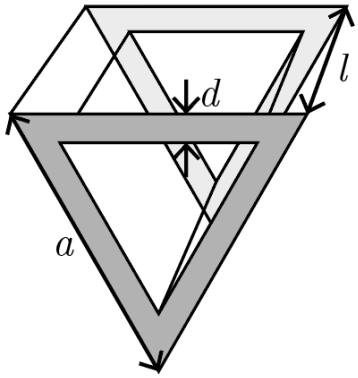

Рис. 2.3.12

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.

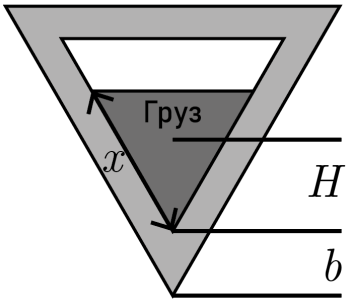

Рис. 2.3.13

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

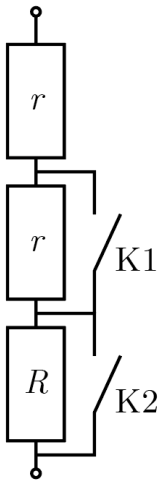

Рис. 2.3.14

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.

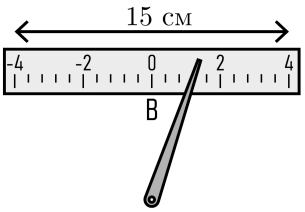

Рис. 2.3.15

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).

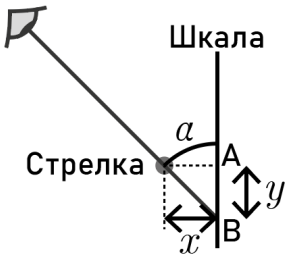

Рис. 2.3.16

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

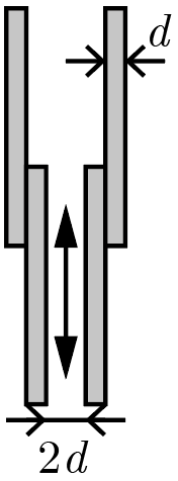

Рис. 2.3.17

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

$$\begin{aligned} m_1 &= \frac{\mu p_0 V}{R(T_0 + \Delta t)}, \\ m_2 &= \frac{\mu p_0 V}{RT_0}, \end{aligned} \tag{2.3.99}$$

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)

$$m_2 \cancel{g} = M\cancel{g} + m_1\cancel{g} \Rightarrow \frac{\mu p_0 V}{RT_0} = M + \frac{\mu p_0 V}{R(T_0 + \Delta t)}, \tag{2.3.100}$$

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:

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

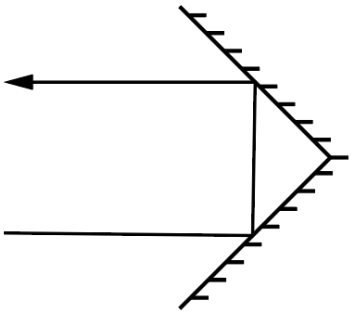

Рис. 2.3.18

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.

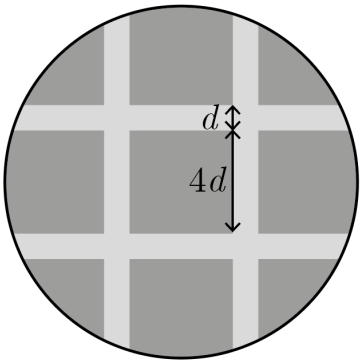

Рис. 2.3.19

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.

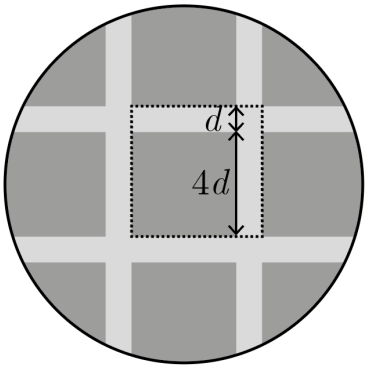

Рис. 2.3.20

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.

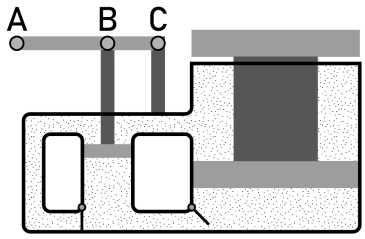

Рис. 2.3.21

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

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

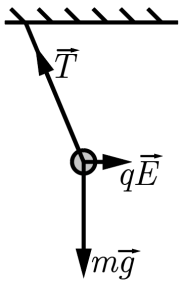

Рис. 2.3.22

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 $g = 9{,}8\,\text{м/с}^2$.

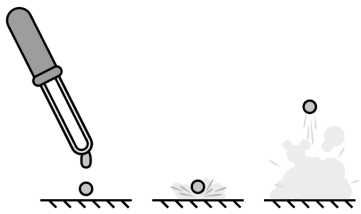

Рис. 2.3.23

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

Задачи первого этапа инженерного тура открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/66700/enter/.

Задача 2.4.1. Анализ круговой диаграммы (5 баллов)

Темы: представление информации, работа с графиками, множества.

Условие

На некоторой площадке по продаже видеоигр есть игры четырех категорий: [Singleplayer], [Adventure], [Strategy], [Multiplayer]. В базе данных системы площадки ведется полный учет по всем играм.

В ежемесячном отчете стажер компании построил круговую диаграмму, отображающую долю игр по категориям, но каждая игра может иметь не одну категорию. Более подробно:

1. любая игра обязательно принадлежит к одной из категорий [Singleplayer] или [Multiplayer], но не сразу к двум;
2. кроме категории [Singleplayer] или [Multiplayer] игра может иметь одну или две (а может не иметь) категории [Adventure] и [Strategy].

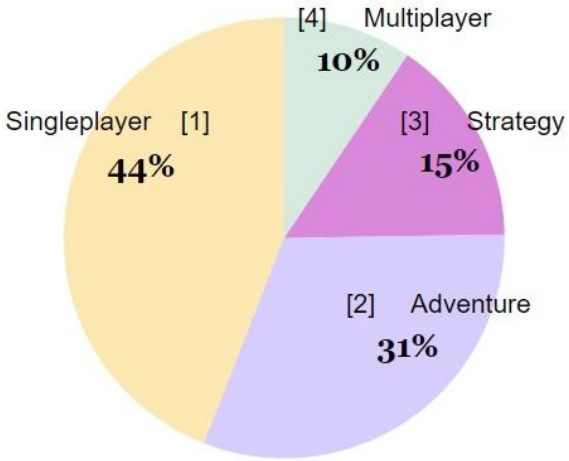

Рис. 2.4.1

Если построить круговую диаграмму, отображающую долю игр с категорией [Singleplayer] и [Multiplayer], то какую долю в процентах будет занимать [Singleplayer]?

Решение

Поскольку игры с метками [Singleplayer] и [Multiplayer] не пересекаются, и каждая игра имеет только одну из этих меток, их процентное соотношение не изменится, если исключить из графика другие метки.

На общей диаграмме [Singleplayer] и [Multiplayer] вместе занимают 54%. Если исключить другие метки, то на графике они будут занимать 100%. Это означает, что их доля увеличится в $k = 100/54$ раза больше. Следовательно, процентное соотношение [Singleplayer] и [Multiplayer] также увеличится в k раз.

Процентное отношение [Singleplayer] будет равно:

$$44 \cdot k = \frac{44 \cdot 100}{54} = 81{,}4814\%.$$

В ответе следует указать 81,48.

Ответ: 81,48.

Задача 2.4.2. Разбор маски (5 баллов)

Темы: комбинаторика, маски.

Условие

Маска строки ω — это шаблон, который определяет формат или структуру строки s, состоящей из строчных латинских символов. В маске ω могут использоваться следующие символы:

1. $*$ (звездочка) — заменяет ноль или несколько любых символов;
2. $?$ (вопросительный знак) — заменяет один любой символ;
3. $[a - z]$ (буква от `a` до `z`) — заменяет одну соответствующую букву.

Например, строка `abb` удовлетворяет маскам $*$, $a?b$, $*b$, $?bb*$, а к маске $? * a*$ подходят строки `ba`, `caba`.

Определите, сколько строк длины от 1 до 4 удовлетворяют маске $\omega = ? * a*$.

Решение

Дана маска $[?*a*]$, где знак вопроса может быть заменен на один из 26 символов. Оставшаяся часть маски $[*a*]$ обозначает любую строку, в которой есть хотя бы одна буква «a».

Так как требуется найти количество строк длиной от 1 до 4, соответствующих маске $[? * a*]$, а знак вопроса уже определен, то требуется определить, сколько строк длины от 1 до 3 соответствуют маске $[*a*]$.

Строк длины 1 с буквой `a`: 1.

Строк длины 2 с буквой `a`: $26^2 - 25^2 = 51$.

Строк длины 3 с буквой `a`: $26^3 - 25^2 = 1\,951$.

Итого $[*a*]$ преобразуется в одну из $(1 + 51 + 1\,951) = 2\,003$ строк.

Итоговый ответ на задачу: $26 \cdot 2\,003 = 52\,078$.

Ответ: 52 078.

Задача 2.4.3. Сбор пирамидок (10 баллов)

Темы: сортировки, жадный алгоритм, бинарный поиск.

Условие

Малыш Макс собирает пирамидки из стержней и колец разного диаметра. У Макса есть a стержней и b колец, все стержни одинаковы и способны собрать на себя три кольца, а вот кольца могут быть разного диаметра. Причем Макс считает, что собранная пирамидка является красивой, если она состоит из трех колец, и кольца в ней идут строго по увеличению диаметра, если смотреть сверху вниз.

Определите максимальное число красивых пирамидок n, которые может собрать Макс. Все n красивых пирамидок должны быть собранными одновременно, то есть каждое кольцо и каждый стержень могут быть задействованы не более, чем в одной пирамидке.

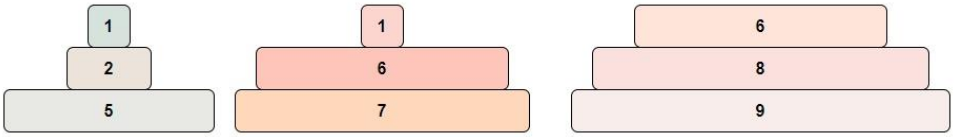

Рис. 2.4.2. Иллюстрация трех пирамидок к первому тесту из примера

Формат входных данных

В первой строке входных данных содержатся два целых числа a, b ($0 \leqslant a, b \leqslant$ $\leqslant 10^5$) — количество стержней и колец в распоряжении Макса соответственно.

Во второй строке содержатся b целых чисел c_i ($1 \leqslant c_i \leqslant 10^9$) — диаметры колец.

Формат выходных данных

Выведите единственное целое число — максимальное число красивых пирамидок.

Примеры

Пример №1

Стандартный ввод
3 12 1 1 2 5 6 6 6 6 7 8 9 9
Стандартный вывод
3

Пример №2

Стандартный ввод
1 3 18 15 3
Стандартный вывод
1

Пример №3

Стандартный ввод
2 11 10 14 3 5 2 10 20 10 4 13 13
Стандартный вывод
2

Решение

Задачу можно решить с помощью бинарного поиска по ответу, это следует из следующего замечания: если можно собрать p пирамидок, то, очевидно, можно собрать и $(p-1)$, и если нельзя собрать p пирамидок, то тем более нельзя собрать и $(p+1)$.

Попробуем понять, получится ли собрать определенное количество пирамидок, равное p. Для этого будем действовать по принципу жадности: отсортируем все кольца независимо от того, в каком порядке они будут расположены — по возрастанию или по убыванию. В результате получится отсортированный массив колец c.

Затем пройдемся по этому массиву и оденем каждое кольцо c_i на стержень, на котором находится минимальное количество колец. Если на этом стержне уже есть кольцо того же диаметра, что и c_i, то пропустим это кольцо. Если в итоге на каждом стержне окажется по три кольца, то получится собрать p пирамидок.

Рассмотрим два неверных подхода к решению задачи.

1. Одевать кольца на стержень с максимальным числом колец. Контрпример: $c = [1, 2, 3, 4, 4, 4, 5, 5, 5]$, если в распоряжении есть минимум три стержня, то ответ 3 достижим: $(1, 4, 5)$, $(2, 4, 5)$, $(3, 4, 5)$, а при неверном подходе получится пирамидка $(1, 2, 3)$ и незаконченная пирамидка $(4, 5, \_)$.
2. Посчитать для кольца каждого диаметра, сколько раз можно его использовать — для колец одного диаметра, которых всего менее a штук, можно их использовать столько раз, сколько их встречается, для колец одного диаметра, которых более a штук, их используем не более a раз (иначе по принципу Дирихле будет пирамидка с двумя кольцами одного диаметра) — посчитаем, сколько есть «доступных» колец и поделим на 3, получив число полных пирамидок. Контрпример: $a = 2$, $c = [1, 1, 2]$. Очевидно, что ни одной пирамидки собрать нельзя, но идея подхода посчитает, что кольца с диаметром 1 можно использовать дважды и кольцо диаметра 2 — всего три кольца, а значит, можно собрать пирамидку.

Сложность: $\mathcal{O}\left(b \cdot \log b + b \cdot \log\left(min\left(a, \frac{b}{3}\right)\right)\right)$.

Ниже представлено решение на языке Python.

Python

```python
import sys

a, b = map(int, sys.stdin.readline().split())
c = list(map(int, sys.stdin.readline().split()))

c.sort()

l, r = 1, min(b // 3, a)

while l <= r:
    m = (l + r) // 2
    cycles = 0

    p = [0 for i in range(m)]
    it = 0
    for i in range(b):
        if p[it] < c[i]:
            p[it] = c[i]
            it += 1
            if it == m:
                it = 0
                cycles += 1

    if cycles < 3:
        r = m - 1
    else:
        l = m + 1

print(max(0, r))
```

Задача 2.4.4. Набор текста (20 баллов)

Темы: динамическое программирование, строки.

Условие

Необходимо напечатать строку s длины n, состоящую из символов в двух разных регистрах: заглавных и строчных латинских букв. Изначально на клавиатуре включен нижний регистр (печатаются строчные буквы).

За одно нажатие клавиши на клавиатуре можно:

1. нажать клавишу `SHIFT`, которая переключит регистр для одного следующего напечатанного символа;
2. нажать клавишу `CAPS`, которая переключает регистр для всех следующих печатаемых символов;
3. нажать клавишу с буквой на клавиатуре, тогда она напечатается в установленном регистре.

Определите минимальное число нажатий клавиш для того, чтобы напечатать строку s.

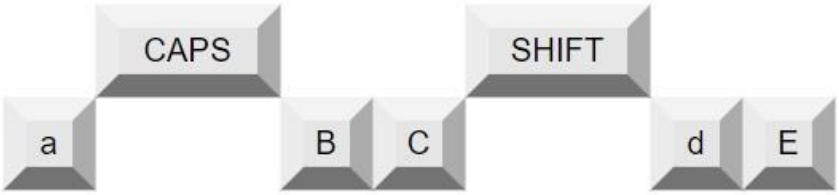

Рис. 2.4.3. Иллюстрация набора строки из второго примера за 7 нажатий

Формат входных данных

В первой строке входных данных содержится целое число n ($1 \leqslant n \leqslant 10^5$) — количество символов в строке s.

Во второй строке содержится строка s длины n. Строка состоит только из строчных и заглавных букв латинского алфавита.

Формат выходных данных

Выведите единственное целое число — минимальное число нажатий клавиш на клавиатуре для того, чтобы напечатать строку s.

Примеры

Пример №1

Стандартный ввод
5 ABCDE

Стандартный вывод
6

Пример №2

Стандартный ввод
5 aBCdE
Стандартный вывод
7

Пример №3

Стандартный ввод
7 AAAAaAA
Стандартный вывод
9

Решение

Задачу можно решить методом динамического программирования. Определим два массива $upper$ и $lower$, оба размера $(n+1)$, следующим образом:

- $upper_i$ — минимальное количество нажатий на клавиатуре для печати первых i символов строки s с установленным в конце верхним регистром;
- $lower_i$ — минимальное количество нажатий на клавиатуре для печати первых i символов строки s с установленным в конце нижним регистром.

База динамики. Изначально $lower_0 = 0$, $upper_0 = 1$, так как изначально клавиатура установлена на нижний регистр, и печатать ничего не нужно, остальные значения $lower$ и $upper$ не определены.

Переход динамики. Будем просматривать строку s слева направо. Пусть сейчас мы смотрим на i-й символ строки, тогда есть два варианта событий:

1. Символ $s_{(i-1)}$ (у строки 0-индексация) в нижнем регистре, тогда $lower_i$ обновится на минимальное значение из следующих:
   - $(lower_{(i-1)}+1)$ — поскольку уже напечатаны $(i-1)$ символов, и у клавиатуры был установлен нижний регистр, то текущий символ напечатаем за одно нажатие;
   - $(upper_{(i-1)}+2)$ — используем нажатие на `CAPS` для переключения регистра на нижний. После того как напечатаны $(i-1)$ символов и включен верхний регистр, нажимаем на клавишу символа.

   $upper_i$ же примет минимальное значение из следующих:
   - $(lower_i+1)$ — используем ответ для $lower_i$, но тратим одно действие на нажатие `CAPS`, переключая регистр на верхний;
   - $(upper_{(i-1)}+2)$ — берем ответ для $upper_{(i-1)}$, нажимаем на `SHIFT`, затем на клавишу с символом. Это требует два действия, и после нажатия на клавишу с символом регистр переключается обратно.

2. Символ $s_{(i-1)}$ в верхнем регистре. Тогда делаем все то же самое, что описано выше, но подменяем понятия $lower$ и $upper$.

Ответ на задачу — минимум из $lower_n$ и $upper_n$. Также можно использовать не два, но один двухмерный массив, как показано в решении ниже.

Сложность: $\mathcal{O}(n)$.

Ниже представлено решение на языке Python.

Python

```python
import sys

n = int(sys.stdin.readline())
s = str(sys.stdin.readline().strip())

dp = [[0 for i in range(n + 1)], [0 for i in range(n + 1)]]
dp[1][0] = 1

for i in range(n):
    if s[i].lower() == s[i]:
        # буква в нижнем регистре
        dp[0][i + 1] = 1 + min(dp[0][i], 1 + dp[1][i])
        dp[1][i + 1] = min(dp[0][i + 1] + 1, 2 + dp[1][i])
    else:
        # буква в верхнем регистре
        dp[1][i + 1] = 1 + min(dp[1][i], 1 + dp[0][i])
        dp[0][i + 1] = min(dp[1][i + 1] + 1, 2 + dp[0][i])

print(min(dp[0][-1], dp[1][-1]))
```

Задача 2.4.5. Набор атлетов (20 баллов)

Темы: сортировки, два указателя, структуры данных.

Условие

В маленькой стране N есть клуб гиревого спорта «Гиря и Гора», в котором преподает поднятие тяжестей небезызвестный в известных кругах уважаемый Мхал Саныч. В клубе занимаются n атлетов, каждый i-й атлет имеет личный рекорд поднятия гири — целое число a_i.

Так как скоро олимпиада, то президент страны N поручил отобрать k атлетов так, чтобы все k атлетов имели различные личные рекорды по поднятию гири. Мхал Саныч хочет выбрать таких k атлетов, чтобы наибольшая разница между личными рекордами любых двух атлетов в команде была как можно меньше, при этом должно выполняться поручение президента.

Определите, какую минимальную разницу между максимальным и минимальным личными рекордами атлетов можно получить, выбрав k атлетов с различными личными рекордами из n имеющихся.

Формат входных данных

В первой строке входных данных содержатся два целых числа n, k ($1 \leqslant n, k \leqslant \leqslant 10^5$) — количество атлетов в клубе «Гиря и Гора» и количество атлетов для отбора на олимпиаду соответственно.

Во второй строке содержатся n целых чисел a ($1 \leqslant a_i \leqslant 10^9$) — личные рекорды каждого атлета по поднятию гири.

Формат выходных данных

Выведите единственное целое число — минимальную из максимальных разниц в наборе из k атлетов с попарно различными личными рекордами. Если ни одного такого набора не существует, то выведите `-1`.

Примеры

Пример №1

Стандартный ввод
`5 2` `3 9 6 10 1`
Стандартный вывод
`1`

Пример №2

Стандартный ввод
`11 3` `4 9 1 1 2 4 5 5 1 2 6`
Стандартный вывод
`2`

Пример №3

Стандартный ввод
`4 3` `1 2 2 1`
Стандартный вывод
`-1`

Решение

Отсортируем массив a и будем искать в нем такой отрезок $[l, r]$, на котором встречается ровно k различных чисел, а разница между $a[r]$ и $a[l]$ будет максимальной разницей в этом наборе. По всем таким наборам нужно найти минимальную разницу.

Решим задачу методом двух указателей. Создадим $left = 0$ и будем итерироваться по массиву переменной $right$, поддерживая на отрезке k различных чисел (это можно сделать, используя счетчик и, например, словарь с количеством вхождений каждого элемента на отрезке).

Если на отрезке менее k различных чисел, то сдвигаем $right + 1$, иначе, если на отрезке k различных чисел, то пытаемся обновить ответ, как минимум, из того, что в нем находится и $a[right]$, и $a[left]$, затем сдвигаем $left + 1$.

Сложность: $\mathcal{O}(n + n \cdot \log n)$, что то же самое, что $\mathcal{O}(n \cdot \log n)$.

Ниже представлено решение на языке Python.

Python

```python
import sys

n, k = map(int, sys.stdin.readline().split())
a = list(map(int, sys.stdin.readline().split()))

a.sort()
mn = 1 << 60
f = {}

r = 0
for i in range(n):
    if a[i] not in f or f[a[i]] == 0:
        f[a[i]] = 1
        k -= 1
    else:
        f[a[i]] += 1

    while k == 0:
        mn = min(mn, a[i] - a[r])
        if f[a[r]] == 1:
            k += 1
        f[a[r]] -= 1
        r += 1

if mn == 1 << 60:
    print(-1)
else:
    print(mn)
```

Задача 2.4.6. Сушилка (8 баллов)

Темы: физика, электричество.

Условие

В умном доме есть сушилка для одежды, ее схема представлена на рис. 2.4.4. К концам схемы приложено постоянное напряжение. Сопротивление $R_1 = 3R$, а сопротивление остальных элементов $R_2 = R_3 = R$. Какое количество теплоты выделится на R_1 за 2 мин, если за 30 с на R_3 выделилось 220 Дж? Ответ укажите в джоулях.

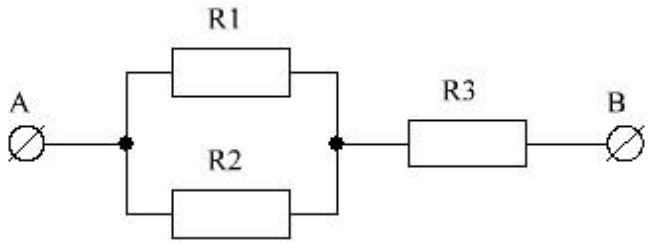

Рис. 2.4.4

Решение

По закону Джоуля – Ленца:

$$Q_1 = I_1^2 \cdot R_1 \cdot t_1;$$
$$Q_3 = I_3^2 \cdot R_3 \cdot t_3.$$

Так как элементы R_1 и R_2 соединены параллельно и подключены к элементу R_3 последовательно, то

$$I_3 = I_1 + I_2 \text{ и } U_1 = U_2.$$

Из закона Ома:

$$I_1 \cdot R_1 = I_2 \cdot R_2 \;\Rightarrow\; I_1 = \frac{I_2 \cdot R}{3R} = \frac{1}{3} I_2;$$
$$I_3 = I_2 + \frac{1}{3} I_2 = \frac{4}{3} I_2 \;\Rightarrow\; I_2 = \frac{3}{4} I_3 \;\Rightarrow\; I_1 = \frac{1}{4} I_3;$$
$$Q_1 = \left(\frac{1}{4} I_3\right)^2 \cdot 3R_3 \cdot t_1 = \left(\frac{1}{4} I_3\right)^2 \cdot 3R_3 \cdot t_3 \frac{t_1}{t_3} = \frac{3}{16} \cdot Q_3 \cdot \frac{t_1}{t_3};$$
$$Q_1 = \frac{3}{16} \cdot 220 \cdot \frac{120}{30} = 165 \text{ Дж}.$$

Ответ: 165 Дж.

Задача 2.4.7. Лампочки (8 баллов)

Темы: физика, электричество.

Условие

Инженер разработал схему, представленную на рис. 2.4.5. Напряжение источника считаем постоянным. Сопротивление резисторов составляет $R = 100$ Ом. Известно, что если в этой цепи вместо одной из ламп подключить резистор с сопротивлением R, то мощность, выделяемая во всей цепи, увеличится в $k = 5$ раз. Найдите сопротивление лампы r, ответ дайте в омах.

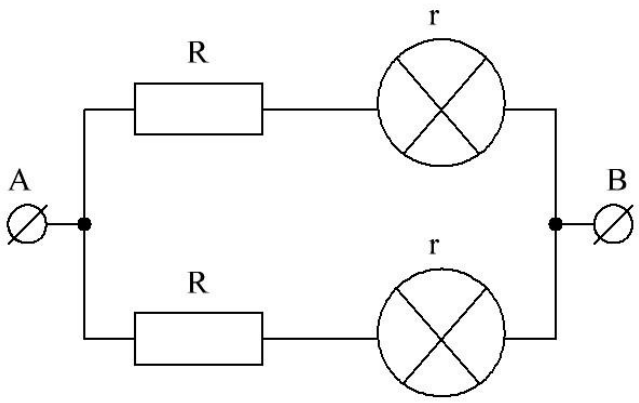

Рис. 2.4.5

Решение

Общее сопротивление цепи в первом случае равно:

$$R_1 = \frac{R + r}{2}.$$

Мощность в цепи равна:

$$P_1 = \frac{U^2}{R_1} = \frac{2 \cdot U^2}{R + r},$$

где U — напряжение между точками A и B.

После того как одну из ламп заменили резистором, общее сопротивление цепи стало равным:

$$R_2 = \frac{2R \cdot (R + r)}{3R + r},$$

а мощность:

$$P_2 = \frac{U^2(3R + r)}{2R \cdot (R + r)}.$$

По условию

$$P_2 = k \cdot P_1.$$

Тогда

$$\frac{U^2(3R + r)}{2R \cdot (R + r)} = k \cdot \frac{2 \cdot U^2}{R + r},$$

отсюда получаем:

$$r = (4k - 3) \cdot R = (4 \cdot 5 - 3) \cdot 100 = 1{,}7 \text{ кОм}.$$

Ответ: 1 700 Ом.

Задача 2.4.8. Конденсаторы (8 баллов)

Темы: физика, электричество.

Условие

В умном городе было три аккумулятора, между которыми нужно было распределить заряд. Для этого их соединили по следующей схеме — рис. 2.4.6. ЭДС источника составляет 12 В. Емкость второго конденсатора в 2 раза больше первого и в 3 раза меньше третьего. Ключи поочередно быстро замкнули и разомкнули, сначала — первый, затем — второй и в конце — третий. Найдите напряжение на всех трех конденсаторах в вольтах.

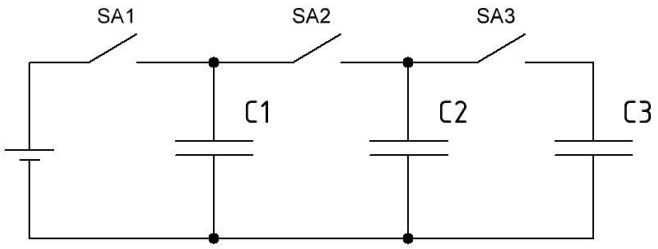

Рис. 2.4.6

Решение

Емкости конденсаторов равны:

$$C_2 = 2C_1;$$
$$C_3 = 3C_2 = 6C_1.$$

Найдем заряд на первом конденсаторе после того, как замкнули первый ключ:

$$q_0 = C_1 \cdot U_{\text{ист}}.$$

Затем найдем отношение зарядов после замыкания второго ключа, зная, что после этого напряжение на C_1 и C_2 стало одинаковым:

$$\frac{q_1}{C_1} = \frac{q_2}{C_2} \;\Rightarrow\; \frac{q_1}{q_2} = \frac{C_1}{C_2} = \frac{1}{2} \;\Rightarrow\; q_1 = \frac{1}{2}q_2.$$

Известно, что

$$q_0 = q_1 + q_2 \;\Rightarrow\; q_2 = \frac{2}{3}q_0 \text{ и } q_1 = \frac{1}{3}q_0.$$

Найдем заряды на конденсаторах после замыкания третьего ключа:

$$\frac{q_2'}{q_3} = \frac{C_2}{C_3} = \frac{1}{3} \;\Rightarrow\; q_2' = \frac{1}{3}q_3.$$

Известно, что

$$q_2 = q_2' + q_3 \;\Rightarrow\; q_3 = \frac{3}{4}q_2 = \frac{1}{2}q_0$$

и

$$q_2' = \frac{1}{4}q_2 = \frac{1}{6}q_0.$$

Находим напряжения на каждом конденсаторе:

$$U_{C_1} = \frac{C_1 \cdot U_{\text{ист}}}{3 \cdot C_1} = 4 \text{ В},$$
$$U_{C_2} = \frac{C_1 \cdot U_{\text{ист}}}{6 \cdot C_2} = \frac{U_{\text{ист}}}{12} = 1 \text{ В},$$
$$U_{C_3} = \frac{C_1 \cdot U_{\text{ист}}}{2 \cdot C_3} = \frac{U_{\text{ист}}}{12} = 1 \text{ В}.$$

Ответ: $U_{C_1} = 4$ В, $U_{C_2} = 1$ В, $U_{C_3} = 1$ В.

Задача 2.4.9. Диод (8 баллов)

Темы: физика, электричество.

Условие

Инженер спроектировал схему, представленную на рис. 2.4.7, вольт-амперная характеристика диода изображена на рис. 2.4.8. Известно, что напряжение источника в два раза больше порогового напряжения U_0, а внутреннее сопротивление источника в восемь раз меньше сопротивления резистора. Найдите отношение мощностей $P_R/P_{\text{диод}}$. Ответ округлите до десятых.

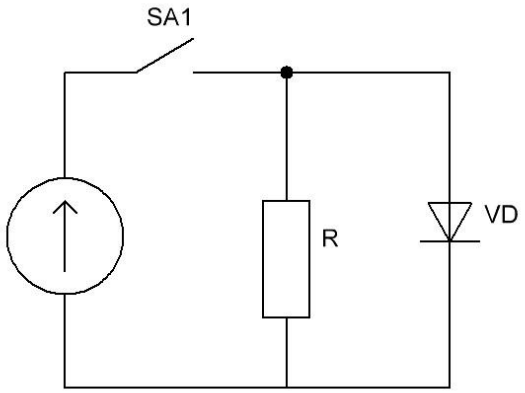

Рис. 2.4.7

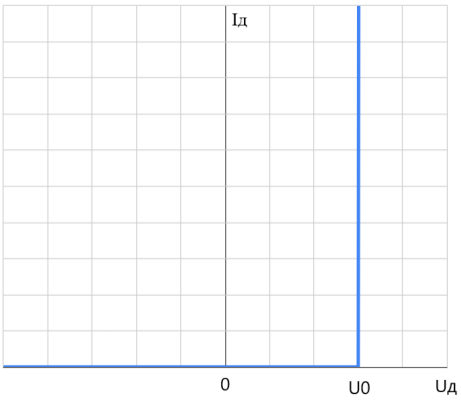

Рис. 2.4.8

Решение

Диод, очевидно, находится в открытом состоянии.

Мощности, выделяемые на элементах:

$$P_R = U_0 \cdot I_R$$

и

$$P_д = U_0 \cdot I_д.$$

Отсюда

$$\frac{P_R}{P_д} = \frac{I_R}{I_д}.$$

Эквивалентная схема.

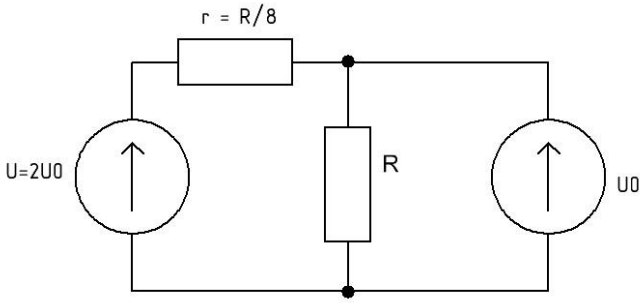

Рис. 2.4.9

Методом наложенных контуров находим токи:

$$I_R = 0 + \frac{U_0}{R}$$

и

$$I_д = \frac{16U_0}{R} - \frac{9R \cdot U_0}{R^2},$$

$$\frac{P_R}{P_д} = \frac{I_R}{I_д} = \frac{\frac{U_0}{R}}{\left(\frac{16U_0}{R} - \frac{9R \cdot U_0}{R^2}\right)} \approx 0{,}1.$$

Ответ: 0,1.

Задача 2.4.10. Оптопара (8 баллов)

Темы: физика, электричество.

Условие

В умном городе используется схема с оптопарой, представленная на рис. 2.4.10. Имеется зависимость входного тока оптопары от выходного (рис. 2.4.11). Известно, что напряжение входного источника равно 32 В, а выходного — 15 В, сопротивление $R_1 = 4$ кОм, $R_2 = 3$ кОм. Найдите ток в выходной ветви в миллиамперах.

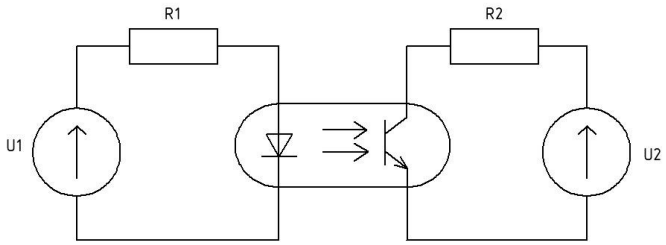

Рис. 2.4.10

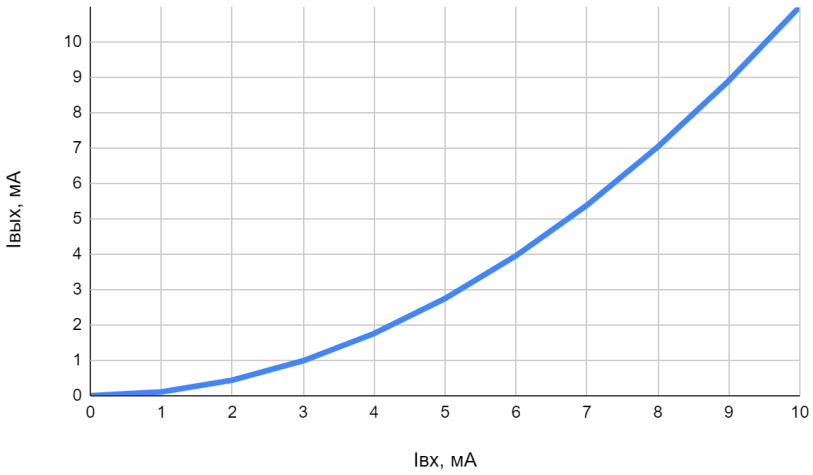

Рис. 2.4.11

Решение

Найдем ток во входной цепи:

$$I_{вх} = \frac{U_1}{R_1} = \frac{32}{4000} = 8 \text{ мА}.$$

По графику находим выходной ток, он равен 7 мА, но необходимо проверить, возможен ли такой ток. Найдем максимально возможный ток в выходной цепи:

$$I_{\text{вых}} = \frac{U_2}{R_2} = \frac{15}{3000} = 5 \text{ мА}.$$

Можно сделать вывод, что фототранзистор вышел в насыщение, и ток в выходной цепи составляет 5 мА.

Ответ: 5 мА.

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

3.2.1. Командные задачи

Командные задачи второго этапа инженерного тура открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/69929/enter/.

Задача 3.2.1.1. XOR-функция (600 баллов)

Тема: логические операции.

Условие

Определим функцию $\hbar(N)$ как значение выражения $1 \oplus 2 \oplus \ldots \oplus N$, где $\oplus$ обозначает логическое исключающее «или».

Для скольких N от 1 до 2^{60} включительно значение функции $\hbar(N)$ принимает значение 0?

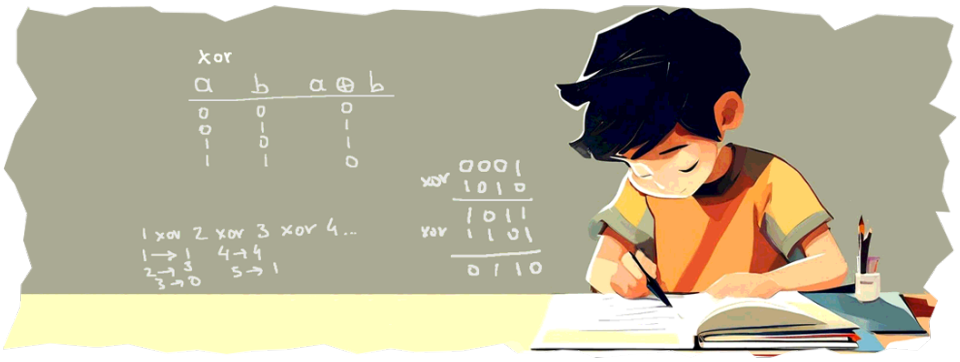

Рис. 3.2.1

Примечания

Результат функции xor между двумя бинарными числами — это число, полученное путем применения xor (исключающего «или») попарно между каждым битом двух чисел. Например, $11_{10} \oplus 6_{10} = 13_{10}$, так как $11_{10} = 1011_2$ и $6_{10} = 110_2$, а $1011_2 \oplus 0110_2 = 1101_2 = 13_{10}$.

Решение

Приведенная функция для всех чисел от 1 до 2^k принимает значение 0 ровно 2^{k-2} раза. В частности, для всех чисел от 1 до 2^{60} значение 0 будет получено $2^{58} = 288\,230\,376\,151\,711\,744$ раза.

Ответ: 288 230 376 151 711 744.

Задача 3.2.1.2. Разбор интервью (800 баллов)

Темы: теория игр, турниры, логика.

Условие

В шахматном турнире принимали участие шесть человек: Антон, Богдан, Вера, Глеб, Даша и Ева. Каждый сыграл с каждым по одной игре. За победу в игре победитель получает 2 очка, а проигравший — 0, в случае ничьи оба игрока получают по 1 очку.

Турнир был закрытым, и его результаты неизвестны, но из интервью, данных каждым игроком, получена некоторая информация о турнире. Гарантируется, что все игроки сказали правду. Ниже приведены утверждения каждого из них.

- Антон: «Я не сыграл ни одной игры вничью, а еще не занял последние два места, поэтому я доволен».
- Богдан: «Я опередил моего давнего соперника Антона. Кроме того, я не проиграл ни одной игры».
- Вера: «Сложилось так, что все получили разное количество очков в конце турнира, и места определись однозначно».
- Глеб: «Я не проиграл ни одной игры, тем не менее я не смог занять первое место».
- Даша: «Я выиграла у своей подруги Евы, мне кажется, она немного расстроилась».
- Ева: «У меня ни разу не получилось выиграть, но я сыграла вничью с Глебом, который очень хорошо играет в шахматы».

Из предложенных ниже утверждений выберите те, про которые можно сказать, что они верны вне зависимости от исходов конкретных игр (разумеется, при соблюдении истинности ответов участников).

Варианты ответа (множественный выбор):

1. В турнире выиграл или Богдан, или Глеб.
2. Последнее место заняла Ева.
3. Глеб мог занять второе место.
4. В турнире победил мальчик.
5. Вера могла оказаться на первом месте.
6. Богдан и Глеб сыграли между собой вничью.
7. Глеб выиграл игру у Антона.
8. Первые три места заняли мальчики.

Решение

Опровергнем ряд утверждений примером, см. таблицу 3.2.1.

Таблица 3.2.1. Таблица результатов

	А	**Б**	**В**	**Г**	**Д**	**Е**	**Место**	**Очки**
А		0	0	0	2	4	4	4
Б	2		1	1	1	1	3	6
В	2	1		1	1	2	1	8
Г	2	1	1		2	1	2	7
Д	0	1	0	0		2	5	3
Е	0	1	0	1	0		6	2

Неверные утверждения:

- В турнире выиграл или Богдан, или Глеб.
- В турнире победил мальчик.
- Первые три места заняли мальчики.

Этот пример также подтверждает следующие утверждения:

- Глеб мог занять второе место.
- Вера могла оказаться на первом месте.

Заметим, что хоть следующие ниже утверждения выполнены в примере, это не доказывает их истинность при любом другом исходе игр ввиду их формулировки:

- Последнее место заняла Ева.
- Богдан и Глеб сыграли между собой вничью.
- Глеб выиграл игру у Антона.

Рассмотрим их по отдельности:

- «Глеб выиграл игру у Антона»: это верное утверждение, потому что Антон не мог сыграть вничью, а Глеб не проиграл ни одной игры.
- «Богдан и Глеб сыграли между собой вничью»: это тоже верное утверждение, потому что Богдан и Глеб не проиграли ни одной игры, но они провели между собой игру на турнире — единственный исход в таком случае ничья.
- «Последнее место заняла Ева»: опровергнем это утверждение примером, см. таблицу 3.2.2.

Таблица 3.2.2. Таблица результатов

	А	**Б**	**В**	**Г**	**Д**	**Е**	**Место**	**Очки**
А		0	0	0	2	2	4	4
Б	2		2	1	2	1	1	8
В	2	0		1	2	1	3	6
Г	2	1	1		2	1	2	7
Д	0	0	0	0		2	6	2

	А	Б	В	Г	Д	Е	Место	Очки
Е	0	1	1	1	0		5	3

Ответ: 3, 5, 6, 7.

Задача 3.2.1.3. Покупка Милки (1000 баллов)

Темы: жадный алгоритм, структуры данных.

Условие

Петя открыл для себя вкусный напиток «Милки» (молочный коктейль), который продается в магазине неподалеку от его дома. Он решил пить его каждый день в течение n дней по одной банке в день.

Однако в этом магазине действует активное ценообразование, то есть цены на все товары, включая «Милки», меняются каждый день. К счастью, тетя Пети работает в этом магазине и знает цены на «Милки» на ближайшие n дней. Теперь Петя тоже знает эти цены.

Но есть одно ограничение: в один день Петя может купить не более k банок «Милки», потому что именно столько помещается в его рюкзак, а ходить в магазин чаще одного раза в день он не хочет.

Помогите Пете определить минимальную сумму денег, которая ему понадобится, чтобы пить «Милки» каждый день в течение следующих n дней.

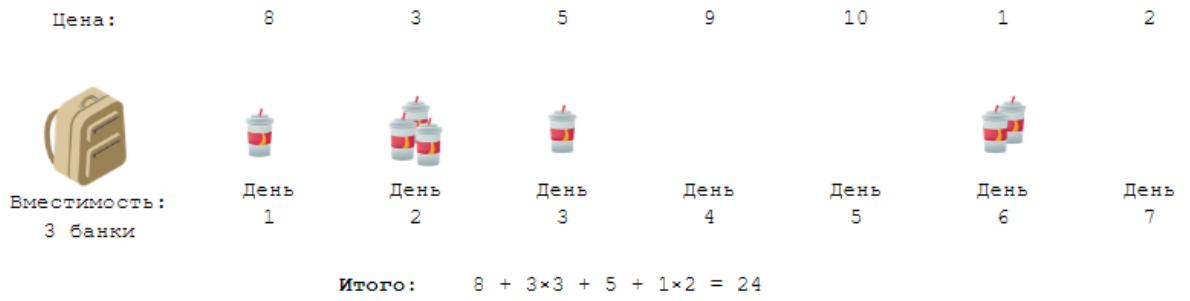

Рис. 3.2.2. Иллюстрация к третьему примеру

Формат входных данных

В первой строке входных данных содержатся два целых числа n, k ($1 \leqslant n, k \leqslant \leqslant 10^5$) — количество дней и размер рюкзака Пети. Во второй строке входных данных содержатся n целых чисел p_i ($1 \leqslant p_i \leqslant 10^9$) — цена напитка «Милки» в i-й день.

Формат выходных данных

Выведите единственное целое число — минимальное число денег, необходимое Пете, чтобы иметь возможность пить по банке «Милки» в каждый из следующих n дней.

Примеры

Пример №1

Стандартный ввод
6 2 2 4 3 9 10 7
Стандартный вывод
18

Пример №2

Стандартный ввод
5 3 5 4 3 2 1
Стандартный вывод
15

Пример №3

Стандартный ввод
7 3 8 3 5 9 10 1 2
Стандартный вывод
24

Решение

Рассмотрим один из вариантов решения этой задачи.

Определим пустую очередь с приоритетом h — эта очередь будет работать с парами чисел, выполняя операции вставки и извлечения за $\mathcal{O}(\log |h|)$.

Начнем итерироваться по массиву p, начиная с первого элемента. Запомним, что в день i цена на напиток равна p_i, и в этот день можно было купить k банок — выполним вставку в очередь h пары чисел (p_i, k). Затем выполним операцию извлечения пары $(price, rem)$ (цена напитка в какой-то день и количество банок, которое можно было купить в этот день) из очереди h — ввиду природы очереди с приоритетом, $price$ будет наименьшим из всех существующих $price$ в очереди h, прибавим

к ответу $price$, если при этом $rem \geqslant 2$, то в тот же день, которому соответствует цена $price$, можно было купить еще минимум одну банку — запомним это и выполним вставку в h пары $(price, rem - 1)$.

Сложность: $\mathcal{O}(n \cdot \log n)$.

Ниже представлено решение на языке Python.

Python

```python
import sys
from heapq import heappop, heappush

n, k = map(int, sys.stdin.readline().split())
p = list(map(int, sys.stdin.readline().split()))

res = 0
h = []

for i in range(n):
    heappush(h, (p[i], k))

    price, rem = heappop(h)
    res += price)

    if rem > 1:
        heappush(h, (price, rem - 1))

print(res)
```

Задача 3.2.1.4. Ограничивающая фигура (1200 баллов)

Темы: оптимизация, сортировка.

Условие

На столе лежат n прямоугольников из цветного картона, каждый прямоугольник имеет целочисленные длины сторон.

Все прямоугольники раскладываются так, что стороны каждого прямоугольника или параллельны, или перпендикулярны сторонам любого другого. Прямоугольники могут частично или полностью лежать друг на друге.

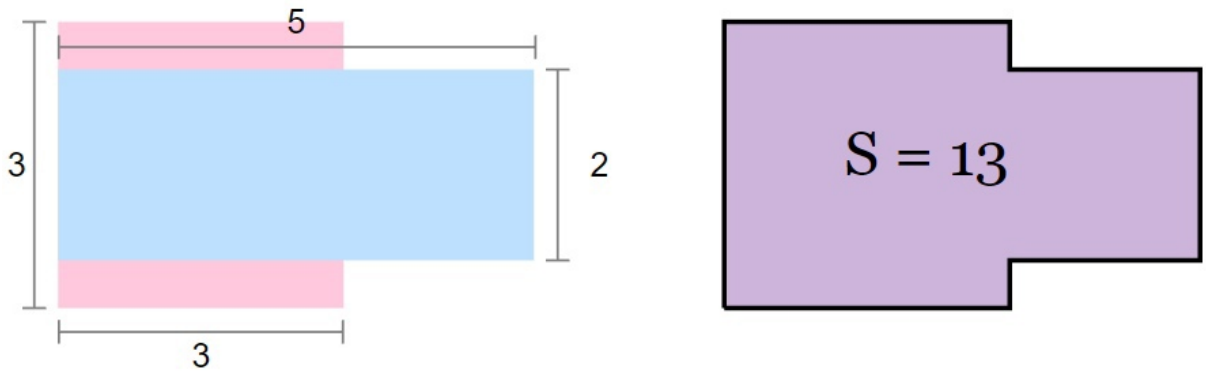

Рис. 3.2.3

Определите минимальную площадь фигуры (необязательно прямоугольника), ограничивающей все лежащие прямоугольники при их оптимальном расположении. Фигуру назовем ограничивающей, если внутрь нее помещаются все прямоугольники.

Формат входных данных

В первой строке входных данных содержится целое число n ($1 \leqslant n \leqslant 2 \cdot 10^5$) — количество прямоугольников.

Далее следуют n строк с описанием каждого прямоугольника: в i-й строке содержится два целых числа w_i, h_i ($1 \leqslant w_i, h_i \leqslant 10^9$) — длины смежных сторон прямоугольника i.

Формат выходных данных

Выведите единственное целое число — минимальную площадь фигуры, покрывающей все прямоугольники.

Примеры

Пример №1

Стандартный ввод
2 5 4 3 3
Стандартный вывод
20

Пример №2

Стандартный ввод
2 3 3 2 5
Стандартный вывод
13

Пример №3

Стандартный ввод
4 5 7 5 4 3 10 16 2
Стандартный вывод
56

Решение

Можно доказать, что оптимальной стратегией для минимизации площади является расположение всех прямоугольников с фиксацией в одном углу (возможно, существует несколько способов, но этот также является оптимальным).

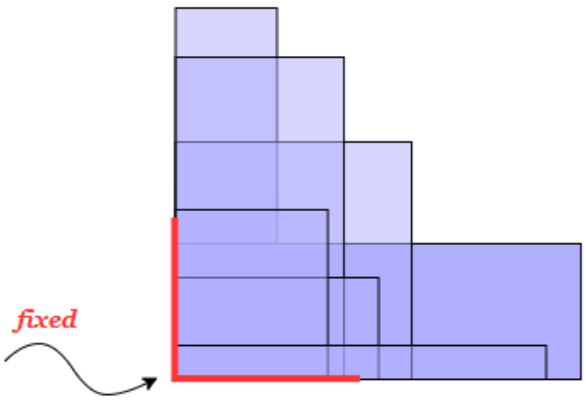

Рис. 3.2.4

Предположим, $h_i \leqslant w_i$, если это не так, то поменяем числа в каждой такой паре, чтобы ширина каждого прямоугольника была не меньше, чем высота. После этого отсортируем все прямоугольники в первую очередь по невозрастанию ширины, а затем — по невозрастанию высоты.

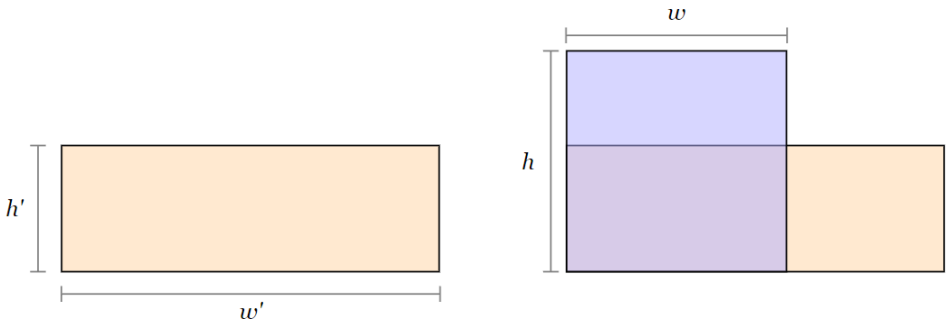

Рис. 3.2.5

Выберем первый прямоугольник, который имеет наибольшую ширину. Если таких прямоугольников несколько, то выберем самый высокий из них. Пусть ширина этого прямоугольника равна w', а высота — h'. Запомним эти значения.

Затем мы будем последовательно рассматривать остальные прямоугольники в порядке их сортировки. Если текущий прямоугольник имеет ширину w и высоту h ($w \geqslant h$), а $h \leqslant h''$, то это означает, что текущий прямоугольник полностью помещается внутри уже существующей фигуры и не увеличивает ее площадь. Поэтому такие прямоугольники пропустим.

Более интересен случай, когда $h > h'$, в такой ситуации прибавим к ответу площадь прямоугольника, ограниченного по оси X с w по w' и по оси Y с 0 по h'.

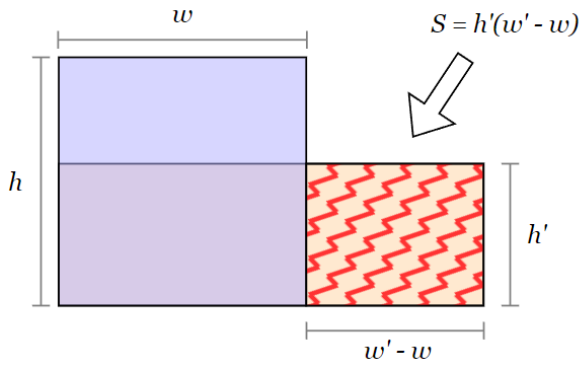

Рис. 3.2.6

Заменим w' на w, то есть присвоим переменной w' значение w. Аналогично присвоим переменной h' значение h. Продолжая этот процесс для оставшихся прямоугольников, можно вычислить площадь всей фигуры за исключением последнего прямоугольника, который совпадает с самым высоким.

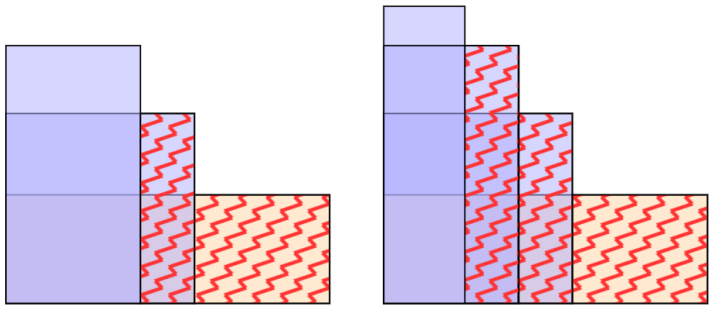

Рис. 3.2.7

В конце прибавим к ответу $w' \cdot h'$ — это площадь «неучтенного» прямоугольника.

Сложность: $\mathcal{O}(n \cdot \log n)$.

Ниже представлено решение на языке Python.

Python

```python
import sys

n = int(sys.stdin.readline())

d = []
for i in range(n):
    w, h = map(int, sys.stdin.readline().split())
    if w >= h:
        w, h = h, w
    d.append([w, h])

d.sort(key=lambda x: (-x[0], -x[1]))

res = 0
a, b = d[0]

for w, h in d[1:]:
    if h > b:
        res += b * (a - w)
        b = h
        a = w

res += a * b

print(res)
```

Задача 3.2.1.5. Высота башни (1400 баллов)

Темы: графы, структуры данных, продвинутые алгоритмы.

Условие

Построим фигурку из блоков конструктора весьма необычным способом: изначально берем крепежную деталь — фундамент конструкции, который является отрезком длины w, затем поднимаемся на возвышенность и сбрасываем с нее поочередно n блоков конструктора. Они падают вниз и прикрепляются к первой задетой

ими детали; если блок не задевает ни одну деталь, то он бесследно пропадает.

Считаем, что изначально деталь фундамента занимает отрезок $[0; w]$, блок i-й по счету падает вниз в вертикальном отрезке $[l_i; r_i]$. Падающий блок задевает деталь, если внутрь отрезка $[l_i; r_i]$ попадает строго больше одной точки детали.

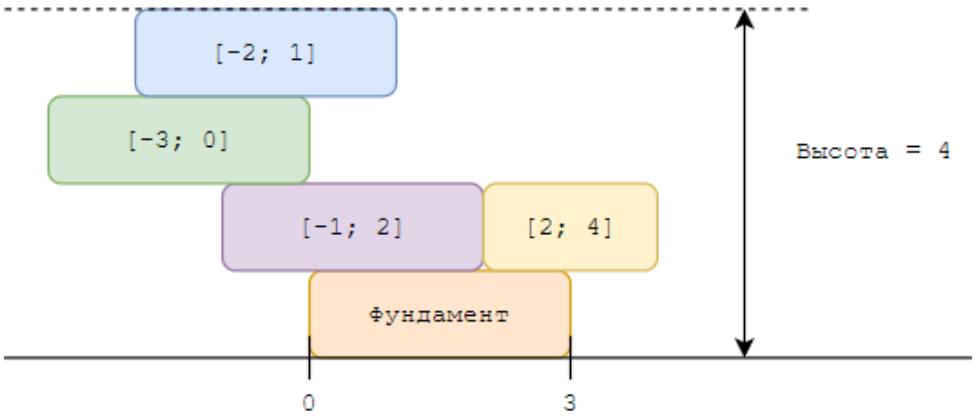

Рис. 3.2.8

Если каждый блок конструктора, в том числе и фундамент, имеет одинаковую высоту, равную 1, то какой высоты получилась фигура?

Формат входных данных

В первой строке входных данных содержится целое число w ($1 \leqslant w \leqslant 10^9$) — длина крепежной детали. Крепежная деталь занимает отрезок $[0; w]$.

Во второй строке входных данных содержится целое число n ($0 \leqslant n \leqslant 10^5$) — количество сбрасываемых блоков.

Далее следуют n строк, в i-й строке содержится описание i-го блока как два целых числа l_i, r_i ($-10^9 \leqslant l_i < r_i \leqslant 10^9$) — левая и правая граница вертикального отрезка, в котором падает блок i.

Формат выходных данных

Выведите единственное целое число — высоту фигуры после сброса всех n блоков.

Примеры

Пример №1

Стандартный ввод
3 4 -1 2 -3 0 2 4 -2 1
Стандартный вывод
4

Пример №2

Стандартный ввод
4 3 -2 5 -3 -2 5 7
Стандартный вывод
2

Пример №3

Стандартный ввод
4 5 2 5 4 7 -2 1 0 3 1 2
Стандартный вывод
4

Решение

Нужно уметь быстро прибавлять на отрезке. Идея, которая напрашивается сразу — дерево отрезков (ДО), однако в стандартном виде это будет работать слишком медленно.

Во-первых, координаты находятся в диапазоне от -10^9 до 10^9, что не позволит задать координаты и использовать в дереве явно. Применим сжатие координат —

рассмотрим все координаты, которые встречаются в задаче, отсортируем их и переназначим значения: так, самой малой координате будет соответствовать 1, второй по величине будет соответствовать 2 и так далее. Итого имеем не более $2n$ различных координат.

Во-вторых, пусть даже с $2n$ различными координатами, ДО в привычном виде все еще не справится с задачей, так как блок в общем случае занимает не одну координату, а отрезок, то есть нужно обновлять значения в ДО для каждой точки в отрезке, а это не пройдет по времени и ничем не лучше реализации без ДО.

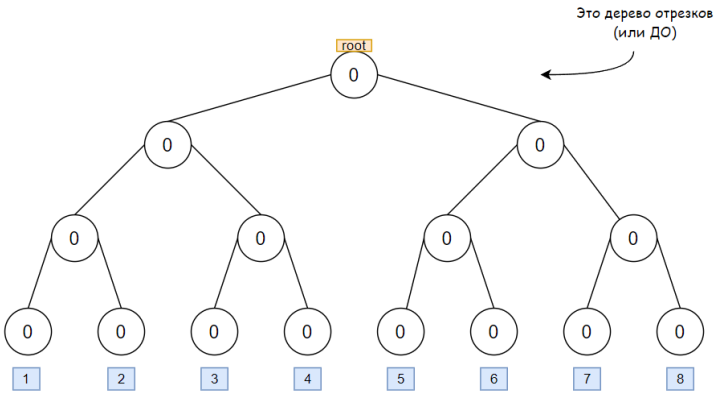

Рис. 3.2.9

Применим оптимизацию, позволяющую прибавлять $+1$ в ДО не в одной точке, а на отрезке. Эта оптимизация использует идею «ленивых вычислений» или «отложенных» (https://ru.algorithmica.org/cs/segment-tree/lazy-propagation/, pdf-версия веб-страницы: https://disk.yandex.ru/i/Nj8_d-2lzCfb9w), а именно: когда текущая вершина в ДО покрывает целиком часть отрезка, на котором выполняется прибавление, то значение в этой вершине увеличивается, а далее не спускаемся в дочерние вершины. Такая операция сопряжена с рядом возможных случаев: если в текущей вершине уже было записано обновление ранее — то спускаем его в дочерние вершины, а только после записываем в текущую вершину новое значение. Вообще, всегда оказавшись в какой-то вершине, спускаем обновления вниз по дереву.

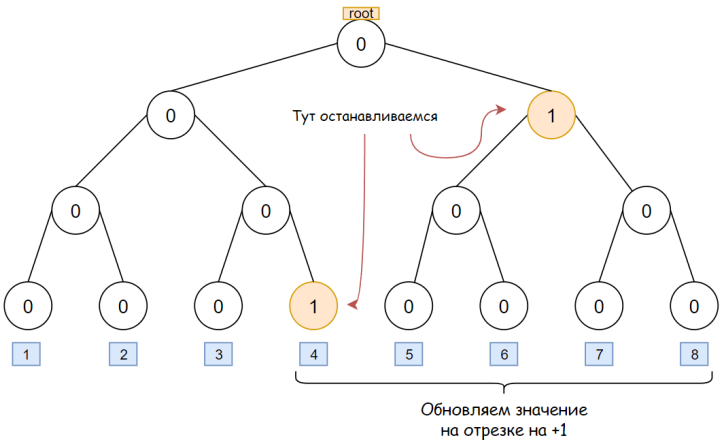

Рис. 3.2.10

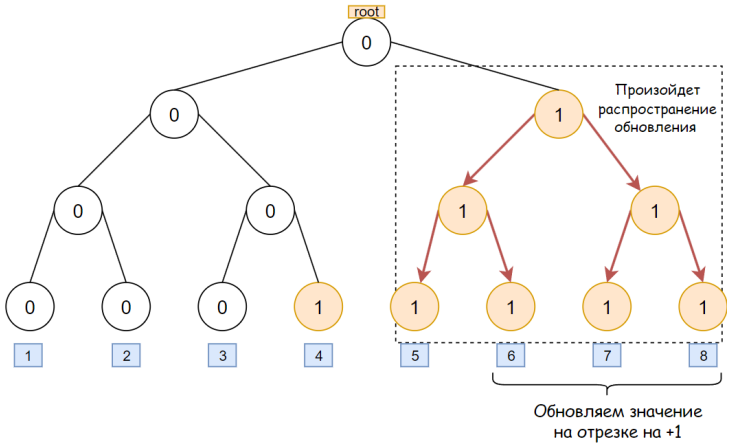

Рис. 3.2.11

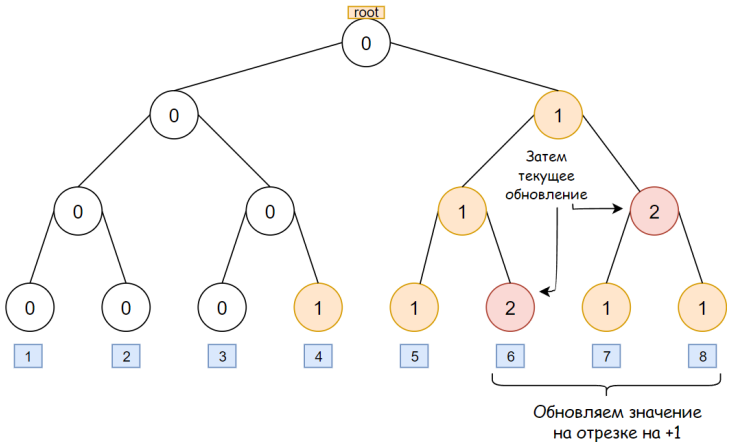

Рис. 3.2.12

Ответ на задачу — максимальное число в вершине дерева.

Всего тратим время на сортировку координат $n \cdot \log n$.

Затем n раз выполняем операции на дереве. Каждая операция затрагивает $4 \cdot \log n$ вершин дерева.

Сложность: $\mathcal{O}(n \cdot \log n)$.

Ниже представлено решение на языке Python.

Python

```python
import sys


class SegmentTree:
    def __init__(self, size):
        self.size = size
        self.tree = [0] * (4 * size)
        self.lazy = [None] * (4 * size)

    def _apply(self, node, start, end, value):
        self.tree[node] = value
        if start != end:
            self.lazy[node * 2] = value
```

```
            self.lazy[node * 2 + 1] = value
        self.lazy[node] = None

    def _update_range(self, node, start, end, l, r, value):
        if self.lazy[node] is not None:
            self._apply(node, start, end, self.lazy[node])

        if start > end or start > r or end < l:
            return

        if start >= l and end <= r:
            self._apply(node, start, end, value)
            return

        mid = (start + end) // 2
        self._update_range(node * 2, start, mid, l, r, value)
        self._update_range(node * 2 + 1, mid + 1, end, l, r, value)
        self.tree[node] = max(self.tree[node * 2], self.tree[node * 2
        ↪  + 1])

    def update_range(self, l, r, value):
        self._update_range(1, 0, self.size - 1, l, r, value)

    def _query_range(self, node, start, end, l, r):
        if start > end or start > r or end < l:
            return -float('inf')

        if self.lazy[node] is not None:
            self._apply(node, start, end, self.lazy[node])

        if start >= l and end <= r:
            return self.tree[node]

        mid = (start + end) // 2
        left_query = self._query_range(node * 2, start, mid, l, r)
        right_query = self._query_range(node * 2 + 1, mid + 1, end, l,
        ↪  r)
        return max(left_query, right_query)

    def query_range(self, l, r):
        return self._query_range(1, 0, self.size - 1, l, r)


def solve():
    w = int(sys.stdin.readline())
    n = int(sys.stdin.readline())

    lr = []
    uniq = {0, w - 1}

    for i in range(n):
        l, r = map(int, sys.stdin.readline().split())
        lr.append([l, r])
        uniq.add(l)
        uniq.add(r - 1)

    sorted_uniq = sorted(list(uniq))
    new_x = {sorted_uniq[i]: i for i in range(len(sorted_uniq))}

    seg_tree = SegmentTree(len(uniq))
```

```
    seg_tree.update_range(l=sorted_uniq.index(0),
                          r=sorted_uniq.index(w - 1),
                          value=1)

    for l, r in lr:
        nl, nr = new_x[l], new_x[r - 1]
        mx = seg_tree.query_range(nl, nr)
        if mx >= 1:
            seg_tree.update_range(nl, nr, mx + 1)

    max_height = seg_tree.query_range(0, len(uniq))
    print(max_height)
    return max_height


if __name__ == '__main__':
    solve()
```

Задача 3.2.1.6. Сгорающий резонансный контур (1000 баллов)

Тема: электричество.

Условие

Была собрана цепь, представленная на рис. 3.2.13. В начальный момент времени оба ключа открыты. В какой-то момент времени первый ключ замыкается и держится закрытым до тех пор, пока ток через резистор R не станет меньше в k_1 раз максимального значения. После этого первый ключ открывается, и закрывается второй. Предохранитель рассчитан на ток I. Какой заряд на конденсаторе C будет в момент перегорания предохранителя?

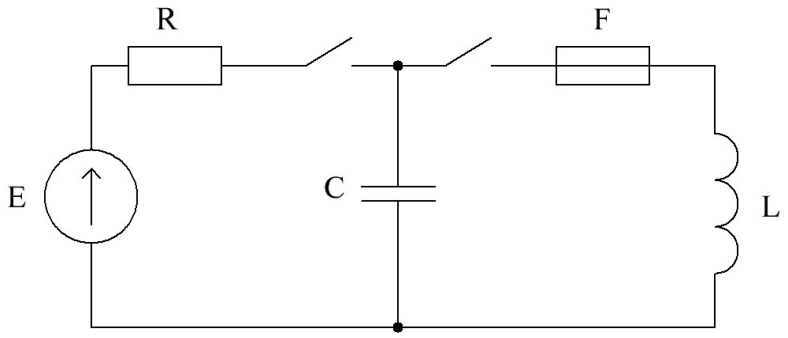

Рис. 3.2.13

Формат входных данных

На вход в строчку через пробел подаются следующие переменные:

- R — сопротивление в омах (Ом), вещественное число от 1 до 10^6;
- k_1 — вещественное число от 2 до 10^6;

- I — ток в амперах (А), вещественное число от 10^{-6} до 10^{2};
- E — напряжение в вольтах (В), вещественное число от 10^{-6} до 10^{3};
- C — емкость в фарадах (Ф), вещественное число от 10^{-6} до 10^{2};
- L — индуктивность в генри (Гн), вещественное число от 10^{-6} до 10^{2}.

Формат выходных данных

На выход необходимо вывести заряд на конденсаторе в кулонах с точностью до шести знаков после запятой.

Примеры

Пример №1

Стандартный ввод
`1000 2 0.1 12 0.001 0.001`
Стандартный вывод
`0.005999`

Решение

Напряжение конденсатора после закрытия первого ключа:

$$U_{Co} = E \cdot \left(1 - \frac{1}{k_1}\right).$$

Полная энергия колебательной системы:

$$W = \frac{C \cdot U_{Co}^2}{2}.$$

Энергия, запасенная в конденсаторе в момент разрыва предохранителя:

$$W_C = W - \frac{L \cdot I^2}{2}.$$

Заряд на конденсаторе:

$$q = \sqrt{2 \cdot W_C \cdot C}.$$

Ниже представлено решение на языке C++.

C++

```
#include <stdio.h>
#include <cmath>
int main() {
    float R, k, I, E, C, L, U, W, Wc, q;
    scanf("%f", &R);
```

```
    scanf("%f", &k);
    scanf("%f", &I);
    scanf("%f", &E);
    scanf("%f", &C);
    scanf("%f", &L);
    U=E*(1-1/k);
    W=C*U*U/2;
    Wc=W-L*I*I/2;
    q=sqrt(Wc*C*2);
    printf("%.6f", q);
    return 0;
}
```

Задача 3.2.1.7. Горное чаепитие (1200 баллов)

Темы: термодинамика, электричество.

Условие

Один заядлый любитель чая и гор отправился в путешествие со своими любимыми аккумулятором, чайником и кружкой. Местность, по которой он гуляет, представлена на рис. 3.2.14 в виде сверху, с указанием высоты каждого квадрата.

Горе-путешественника поднимает ветром и переносит в случайное место на этих горах. Оказавшись в клетке номер N, путешественник не растерялся и тут же включил чайник, налив в него воды ровно на одну кружку чая, и запитал его от аккумулятора. Емкость аккумулятора Q мАч с напряжением 12 В. Объем кружки чая 0,4 л, начальная температура воды 20 °C, плотность воды 1 000 кг/м^3, а удельная теплоемкость воды 4 200 Дж/(кг·°C). Включив чайник, он понял, что из-за высоты гор вода закипает при другой температуре, а именно, температура кипения изменяется на 10 °C при изменении высоты на 300 м, при этом на высоте 1 000 м температура кипения равна 100 °C, температура изменяется по линейному закону. Вот такие вот странные горы.

Наш любитель чая решил во что бы то ни стало выпить максимальное количество кружек, при этом он будет пить чай в каждой посещенной им области, передвигаться между ними он может только по горизонтали и вертикали. Путешественник может оставаться в одной области и пить там неограниченное количество кружек чая, насколько хватит аккумулятора. Определите максимальное число выпитых кружек чая.

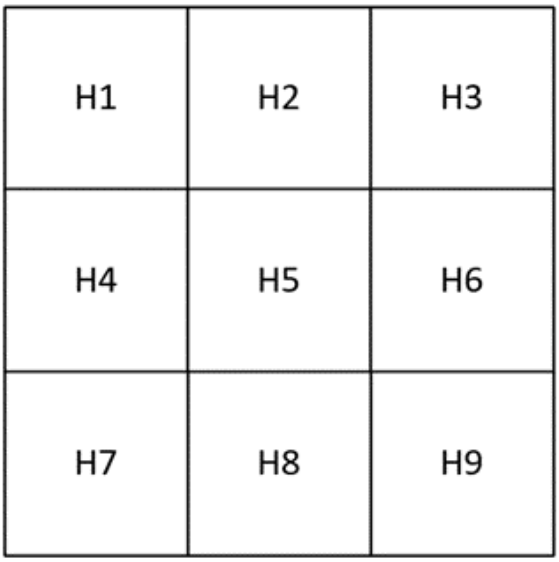

Рис. 3.2.14. Горная местность, вид сверху

Формат входных данных

На вход приходит число N от 1 до 9, означающее местонахождение любителя чая, числа $H1, H2, \ldots, H9$, означающие высоту области над уровнем моря в метрах и емкость аккумулятора Q в миллиампер-час (мАч).

- N — число от 1 до 9;
- $H1, H2, H3$ — числа от 0 до 2 000;
- $H4, H5, H6$ — числа от 0 до 2 000;
- $H7, H8, H9$ — числа от 0 до 2 000;
- Q — число от 10 000 до 100 000.

Формат выходных данных

Целое число, максимальное количество выпитых кружек чая.

Решение

Для начала необходимо пересчитать матрицу высот в матрицу энергетических затрат по формуле:

$$Q_{\text{кип}} = C \cdot m \cdot \Delta t,$$

где:

- $Q_{\text{кип}}$ — количество энергии на кипячение воды в чайнике, Дж;
- C — удельная теплоемкость воды, Дж/(кг·°C);
- Δt — разница между температурой кипения воды на данной высоте и начальной температурой воды, °C.

Для определения Δt воспользуемся следующей формулой:

$$\Delta t = \frac{1000 - h}{30} + 80,$$

где h — высота горы, м.

Также необходимо знать количество запасенной энергии в аккумуляторе в джоулях (Дж):

$$Q_{\text{ак}} = Q \cdot 3{,}6 \cdot V,$$

где Q — емкость аккумулятора, мАч, V — напряжение аккумулятора, В.

Решение заключается в переборе возможных вариантов прохождения. Приведенное ниже решение представляет собой рекурсивный проход с оценкой пути. В зависимости от положения любителя чая на поле (в матрице) 3×3 клетки необходимо учитывать возможность его движения. Оценка пути заключается в вычислении количества выпитых чашек чая.

Ниже представлено решение на языке Python.

Python

```python
n = int(input())
h = [(1000 - x) / 30 + 80 for _ in range(3) for x in map(int, input().split())]
q = int(input())

C_water = 4200
m_water = 0.4
V = 12
Q = q * 3.6 * V

q_for_cup = [C_water * m_water * h[i] for i in range(9)]


def walking(n, tea_drink, energy_remain, used):
    best = tea_drink + int(energy_remain / q_for_cup[n])

    if q_for_cup[n] > energy_remain:
        return best

    for dn in [1, -1, 3, -3]:
        if not 0 <= n + dn <= 8:
            continue
        if abs(dn) != 3 and not 0 <= n % 3 + dn <= 2:
            continue
        if (1 << n + dn) & (used) != 0:
            continue
        best = max(best, walking(n + dn, tea_drink + 1, energy_remain - q_for_cup[n], used | (1 << n + dn)))

    return best


print(walking(n, 0, Q, 1 << n))
```

Задача 3.2.1.8. Психрометр (800 баллов)

Тема: термодинамика.

Условие

Иногда инженерам приходится решать задачу, используя только то, что есть непосредственно под рукой или на складе. Именно такая ситуация случилась у инженера Бори. К нему обратились сотрудники из ботанического сада с просьбой как можно быстрее собрать устройство, которое бы измеряло температуру и влажность в оранжерее, поскольку предыдущее вышло из строя, а некоторые растения очень прихотливы. Осмотрев углы своего КБ на наличие датчиков, Борис нашел только два датчика температуры DS18B20 и BMP280. И тут он вспомнил, что есть такое замечательное устройство, как психрометр (https://ru.wikipedia.org/wiki/Психрометр). «Это то, что нужно!» — воскликнул Борис. И, нагуглив таблицу зависимости плотности насыщенного пара от температуры, принялся писать код. Но так как в КБ сейчас очень много срочных заказов, Боря обратился за помощью к участникам Олимпиады, чтобы вы написали программу, которая бы по разности температур и атмосферному давлению выдавала влажность в помещении.

Таблица 3.2.3

t, °C	ρ, г/м^3	t, °C	ρ, г/м^3	t, °C	ρ, г/м^3
−30	0,28	−8	2,54	14	12,1
−28	0,35	−6	2,99	16	13,6
−26	0,43	−4	3,51	18	15,4
−24	0,52	−2	4,13	20	17,3
−22	0,64	0	4,84	22	19,4
−20	0,77	2	5,6	24	21,8
−18	0,94	4	6,4	26	24,4
−16	1,13	6	7,3	28	27,2
−14	1,36	8	8,3	30	30,3
−12	1,63	10	9,4		
−10	2,14	12	10,4		

Формат входных данных

t_1 — температура сухого термометра, вещественное число с двумя знаками после запятой от −30 до 30 °C.

t_2 — температура сухого термометра, вещественное число с двумя знаками после запятой от −30 до 30 °C ($t_2 < t_1$).

P — атмосферное давление в текущий момент времени, вещественное число с двумя знаками после запятой. Принять, что в оранжерее естественная вентиляция без сквозняков. Входные данные подаются в одну строку через пробел.

Примеры

Пример №1

Стандартный ввод
20 19 760
Стандартный вывод
90%

Пример №2

Стандартный ввод
27.36 17.34 750.49
Стандартный вывод
25%

Пример №3

Стандартный ввод
24.02 17.01 750.68
Стандартный вывод
40%

Пример №4

Стандартный ввод
17.36 10.34 740.49
Стандартный вывод
26%

Пример №5

Стандартный ввод
-4.01 -7.02 758.00
Стандартный вывод
7%

Решение

Для решения данной задачи необходимо было воспользоваться предложенной статьей и таблицей зависимости плотности насыщенного пара от температуры. Из статьи про психрометр надо было подметить два момента:

1. начение психрометрического коэффициента равно 0,0011 (по условию в оранжерее естественная вентиляция без сквозняков);
2. формула для расчета значения влажности:

$$\varphi = 100\left(\frac{d_w}{d_s} - \alpha \cdot B\frac{t - t_w}{d_s}\right).$$

В этой формуле уже известны две температуры (t и t_w), атмосферное давление, а также психрометрический коэффициент. Остается найти плотность насыщенного пара. Для этого была предоставлена таблица соответствия плотности насыщенного пара от температуры, но в качестве входных данных может выступать любое вещественное число от -30 до 30.

Первое, что приходит в голову — округлять температуру до ближайшего целого четного, но точнее будет использовать кусочно-линейную функцию. Такой подход активно используется для преобразования аналоговых сигналов с помощью микроконтроллеров. Суть в том, что, зная приращение по температуре и по плотности, можно найти наклон линейной функции для участка температур в 2°. Из таких небольших участков составляется кусочно-линейная функция, стремящаяся к нелинейной функции изменения плотности насыщенного пара. Значения плотностей насыщенного пара, а также наклоны линейных функций заносятся в соответствующие массивы. На основе известной температуры рассчитывается, к какой линейной функции принадлежит необходимая плотность насыщенного пара, и через линейную функцию находится ее приближенное значение. Все известные величины подставляются в формулу, и вычисляется относительная влажность воздуха. При выводе значение округляется до целых.

Ниже представлено решение на языке C.

C

```c
#include <stdio.h>
#include <stdint.h>
#include <math.h>

#define ALPHA                           0.0011
#define DEFAULT_PREASURE          760
#define VALID_INDEX(index, max) ((index > max) ? max : index)

float density_table[] = {0.28, 0.35, 0.43, 0.52, 0.64, 0.77, 0.94, 1.13, 1.36, 1.63, 2.14, 2.54,
2.99, 3.51, 4.13, 4.84, 5.6, 6.4, 7.3, 8.3, 9.4, 10.4, 12.1, 13.6, 15.4, 17.3, 19.4, 21.8, 24.4,
27.2};

float k_table[] = {0.035, 0.04, 0.045, 0.06, 0.065, 0.085, 0.095, 0.115, 0.135, 0.255, 0.2,
0.225, 0.26, 0.31, 0.355, 0.38, 0.4, 0.45, 0.5, 0.55, 0.5, 0.85, 0.75, 0.9, 0.95, 1.05, 1.2, 1.3,
1.4, 1.55};

```

```
int main(){
float t1, t1w;
float P1 = DEFAULT_PREASURE;
scanf("%f%f%f", &t1, &t1w, &P1);
uint8_t i1, i1w;
i1 = VALID_INDEX((uint8_t)(t1 + 30) / 2, 29);
i1w = VALID_INDEX((uint8_t)(t1w + 30) / 2, 29);
float d1, d1w;
d1 = k_table[i1] * (t1 - (-30 + 2 * i1)) + density_table[i1];
d1w = k_table[i1w] * (t1w - (-30 + 2 * i1w)) + density_table[i1w];
float phi = 100.0f * (d1w - 0.0011 * P1 * (t1 - t1w)) / d1;
printf("%.0f%%\n", phi);
return 0;
}
```

Задача 3.2.1.9. Планер «Добрыня» (800 баллов)

Условие

Борис, как и многие его ровесники, мечтал стать пилотом. Мечте не суждено было сбыться, но из этой мечты исходит его хобби. Борис занимается авиамоделизмом. В преддверии отпуска он днями и ночами набрасывал какие-то идеи по улучшению своего планера «Добрыня». Но для того чтобы понимать, улучшилась ситуация или нет, необходим бортовой компьютер. Он будет писать логи полета, из которых можно будет сделать соответствующий вывод

Борис обратился к участникам Олимпиады с просьбой помочь ему написать часть программы для бортового компьютера. К бортовому компьютеру с Arduino nano на борту будет подключен трехосевой акселерометр ADXL335 (https://disk.yandex.ru/d/4wGwp_aq_ShAlQ/adxl335.pdf). Arduino считывает показания с трех аналоговых выводов, пересчитывает полученные значения в проекции ускорения по трем осям (в метрах в секунду за секунду), а также вычисляет углы планера: тангаж и крен (в градусах).

Примечания

Гарантируется, что напряжение питания Arduino 5 В, датчика 3,3 В, а рыскание отсутствует.

Формат входных данных

N — количество обрабатываемых показаний.
N строчек по три значения.
x_1 — показания АЦП по оси X, 10-битное беззнаковое целое.
x_2 — показания АЦП по оси Y, 10-битное беззнаковое целое.
x_3 — показания АЦП по оси Z, 10-битное беззнаковое целое.

Формат выходных данных

N строчек по пять значений.
a_x — проекция ускорения по оси X, м/с^2 до двух знаков после запятой.

a_y — проекция ускорения по оси Y, м/с^2 до двух знаков после запятой.
a_z — проекция ускорения по оси Z, м/с^2 до двух знаков после запятой.
Roll — крен, град. с точностью до одного знака до запятой.
Pitch — тангаж, град. с точностью до одного знака до запятой.

Примеры

Стандартный ввод
1 263 306 342

Стандартный вывод
-7.01 -0.14 5.60 1 51

Решение

Для решения данной задачи необходимо понять, каким образом ускорение, измеряемое акселерометром, преобразуется в напряжение на входе. Для этого обратимся к документации на акселерометр, которая была приложена к заданию. Акселерометр рассчитан на ускорение до 3g. Смещение нуля, согласно документации, составляет 1,65 В, при питании в 3,3 В. Приращение напряжения от изменения ускорения по оси 300 мВ/g. Отсюда выведем формулу расчета:

$$a = \frac{\left(\frac{adc \cdot 5}{1023} - 1{,}65\right)}{0{,}300} \cdot 9{,}8 = \frac{\left(\frac{adc \cdot 5000}{1023} - 1650\right)}{300} \cdot 9{,}8 = \left(\frac{adc \cdot 50}{3069} - 5\right) \cdot 9{,}8.$$

Для каждого показания АЦП по трем осям рассчитывается ускорение в метрах секунду за секунду. Далее применим формулу из `application note` для расчета крена и тангажа. Эти формулы справедливы для большинства акселерометров и легко находятся в статьях:

$$roll = \operatorname{arctg}\left(\frac{a_y}{a_z}\right);$$

$$pitch = \operatorname{arctg}\left(\frac{-a_x}{\sqrt{a_y^2 + a_z^2}}\right).$$

Подставляя в эти формулы рассчитанные ранее ускорения, вычисляются углы крен и тангаж в радианах, остается только пересчитать радианы в градусы.

Ниже представлено решение на языке C++.

C++

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

#define PI 3.1415926535
#define PI_2 (PI / 2.0)
#define PI_4 (PI / 4.0)

#define ABS(x) ((x < 0) ? -x : x)
#define SQR(x) (x * x)
```

```
#define CONV(x) ((x - 1) / (x + 1))
#define DEGREES(x) ((float)x * 180.0f / PI)
#define ATAN(x) (x * (0.9992150 + SQR(x) * (-0.3211819 + SQR(x) *
    (0.1462766 + SQR(x) * (-0.0389929)))))

float fast_rsqrt(float number){
        const float x2 = number * 0.5F;
        const float threehalfs = 1.5F;

        union {
                float f;
                uint32_t i;
        } conv = {number};
        conv.i = 0x5f3759df - ( conv.i >> 1 );
        conv.f *= threehalfs - x2 * conv.f * conv.f;
        return conv.f;
}

float fast_atan(float number){
        if (number > 0)
                return PI_4 + ATAN(CONV(number));
        else
                return -(PI_4 + ATAN(CONV(-number)));
}

int main(){
        uint32_t N;
        scanf("%u", &N);
        uint32_t* array = (uint32_t*)calloc(N * 3, sizeof(uint32_t));
        for (uint32_t i = 0; i < N * 3; i += 3)
                scanf("%u %u %u", &array[i], &array[i + 1], &array[i +
                    2]);
        for (uint32_t i = 0; i < N * 3; i += 3){
                float ax_x = ((float)array[i] * 50 / 3069 - 5) * 9.8;
                float ax_y = ((float)array[i + 1] * 50 / 3069 - 5) *
                    9.8;
                float ax_z = ((float)array[i + 2] * 50 / 3069 - 5) *
                    9.8;
                float roll = fast_atan(ax_y / -ax_z);
                float pitch = fast_atan(-ax_x * fast_rsqrt(SQR(ax_y) +
                    SQR(ax_z)));
                printf("%.2f %.2f %.2f %.0f %.0f\n", ax_x, ax_y, ax_z,
                    DEGREES(roll), DEGREES(pitch));
        }
        free(array);
        return 0;
}
```

Задача 3.2.1.10. Умный регулировщик (1200 баллов)

Темы: алгоритмы, ПДД.

Условие

Беспилотных автомобилей в умном городе становится все больше и больше. Они уже классно ездят по прямой и поворачивают, пока никто не мешает. Но последнее время здесь ужасные пробки из-за того, что беспилотные автомобили не

могут разобраться на перекрестке, кто же должен ехать первым. Надо было учить ПДД, прежде чем садиться за руль!

В умном городе существует несколько типов нерегулируемых перекрестков, среди которых наиболее распространены обычные (состоящие из двух пересекающихся дорог) и Т-образные перекрестки. На таких перекрестках отсутствуют светофоры, но у некоторых дорог есть знаки приоритета. Задача заключается в разработке алгоритма, который будет управлять движением беспилотных автомобилей на перекрестках, получая информацию от умного регулировщика о типе перекрестка и наличии приоритетных полос.

Формат входных данных

От умного регулировщика поступает `JSON`-сообщение, содержащее следующую информацию:

- Тип перекрестка (`type`): может быть обычным (`regular`) или Т-образным (`T-shaped`).
- Перечень приоритетных полос (`priority_lanes`): массив строк, указывающий, какие полосы движения имеют приоритет. Если список пустой, значит, ни одна из полос не имеет приоритет.
- Количество автомобилей на перекрестке (`vehicle_count`): общее количество автомобилей на перекрестке.
- Информация о каждом автомобиле:
  - ◇ `vehicle_id`: уникальный идентификатор автомобиля;
  - ◇ `lane`: полоса, на которой находится автомобиль;
  - ◇ `direction`: направление, в которое движется автомобиль (например, `west`, `north` и т. д.).

Дороги в умном городе могут быть в восьми направлениях:

- `north` — север,
- `north-east` — северо-восток,
- `east` — восток,
- `south-east` — юго-восток,
- `south` — юг,
- `south-west` — юго-запад,
- `west` — запад,
- `north-west` — северо-запад.

Считать, что каждая дорога имеет две полосы движения: в прямом и во встречном направлениях.

Задача — разработать алгоритм, который на основании входных данных определит порядок проезда автомобилей через перекресток и вернет его в формате `JSON`. В случае, если два и более автомобилей могут разъехаться одновременно, не создавая помеху друг другу, регулировщик должен назначить им равный приоритет. Любой автомобиль может ехать из любой полосы в любую полосу.

Формат выходных данных

Алгоритм должен вернуть JSON-ответ, содержащий порядок проезда автомобилей через перекресток. Ответ должен содержать порядок движения автомобилей. Порядок появления автомобилей в ответе сортируется по `vehicle_id`. В ответе не может несколько раз появиться один и тот же автомобиль, но у разных автомобилей может быть одинаковый приоритет (если они могут двигаться одновременно, не создавая помеху друг другу). При формировании ответа необходимо соблюдать следующую разметку (количество и места пробелов должны строго соответствовать примеру, см. рис. 3.2.15).

```
{
 "order": [
    {
      "vehicle_id": 1,
      "priority": 1
    },
    {
      "vehicle_id": 2,
      "priority": 1
    },
    {
      "vehicle_id": 3,
      "priority": 1
    },
    {
      "vehicle_id": 4,
      "priority": 1
    }
  ]
}
```

Рис. 3.2.15

Примеры

Стандартный ввод

```
{
  "intersection": {
    "type": "regular",
    "priority_lanes": ["north", "south"],
    "vehicle_count": 4
  },
  "vehicles": [
    {
      "vehicle_id": 1,
      "lane": "north",
      "direction": "west"
    },
    {
      "vehicle_id": 2,
      "lane": "east",
      "direction": "north"
    },
    {
      "vehicle_id": 3,
      "lane": "south",
      "direction": "east"
    },
    {
      "vehicle_id": 4,
      "lane": "west",
      "direction": "south"
    }
  ]
}
```

Стандартный вывод

```
{
  "order": [
    {
      "vehicle_id": 1,
      "priority": 1
    },
    {
      "vehicle_id": 2,
      "priority": 1
    },
    {
      "vehicle_id": 3,
      "priority": 1
    },
    {
      "vehicle_id": 4,
      "priority": 1
    }
  ]
}
```

Решение

Все тестовые задания созданы на основе официальных заданий из теоретических билетов ГИБДД на право управления транспортным средством категории B. В частности, были использованы задания на нерегулируемые перекрестки. Соответственно, все правила, определяющие порядок проезда, указаны в ПДД на момент 2024 года.

Основная идея решения в следующем: на перекрестке возможны восемь направлений. Для каждого направления условно можно выделить две полосы: въезд и выезд. Так как на территории Российской Федерации движение правостороннее,

то въезд всегда будет «правее» или «дальше по часовой стрелке», чем выезд.

Таким образом, можно представить все возможные направления в виде окружности, на которой с равным шагом расположены 16 точек. Каждая из точек обозначает въезд или выезд для каждого направления. Каждый из автомобилей стремится отправиться из начальной точки в конечную. Начальную и конечную точку можно соединить линией и получить отрезок или вектор.

Примем изначально приоритет каждого автомобиля равным 1 (должен проехать первым). Далее необходимо проверить, пересекаются ли отрезки/вектора автомобилей. Если они не пересекаются, значит, автомобили на перекрестке могут разъехаться, не пересекая траекторию друг друга (в реальности, конечно, может существовать перекресток, где такое окажется невозможным, но команда разработчиков провела наблюдение за несколько перекрестками и решила, что такое допущение работает в подавляющем большинстве случаев). Автомобили, чьи траектории не пересекаются, могут оставить равный приоритет между собой.

В случае, если траектории все-таки пересеклись, необходимо определить, чей приоритет останется более высоким. В ПДД это определяется «помехой справа». Простыми словами помехой справа является такая помеха, которая в моменте находится справа от водителя. Если водитель 1 едет прямо в одном направлении, а перпендикулярно ему едет водитель 2 с правой стороны, водитель 1 должен уступить дорогу, т. к. водитель 2 является для него помехой справа.

С точки зрения программирования для определения помехи справа можно использовать разные способы. Самым простым, с точки зрения реализации, является расчет векторного произведения двух векторов движения. Если векторное произведение положительное, то помехой справа является один автомобиль, если отрицательное, то другой. Здесь может возникнуть сложная ситуация при расчете, если конечная точка у обоих автомобилей одинаковая. Для устранения этой проблемы каждый вектор/отрезок можно немного продлить.

Таким образом стоит проверить каждую пару автомобилей. Во избежание неточностей при решении рекомендуем сделать проверку итерационной. Так, в каждой итерации приоритет автомобиля может увеличиться только на 1. Всего итерационным образом нужно провести проверки 3 раза, что по условию задачи будет достаточным (максимум 4 автомобиля, максимальный приоритет может быть 4).

С точки зрения распределения приоритета, с учетом существования главной дороги, логика значительно не меняется, за исключением того факта, что оба автомобиля с главной дороги распределяют приоритет между собой согласно правилам, описанным выше; автомобиль с главной дороги имеет приоритет относительно автомобиля со второстепенной дороги безусловно; автомобили со второстепенной дороги распределяют приоритет согласно правилам, описанным выше. Далее остается только сформировать посылку по правилам, приведенным в задании.

В коде ниже есть закомментированные строки, выводящие промежуточные результаты. Это позволяет лучше понять логику кода.

Ниже представлено решение на языке Python.

Python

```
1  import json
2  import math
```

```
3
4   def polar_to_cartesian(radius, angle_deg):
5       """Преобразует полярные координаты в декартовые."""
6       angle_rad = math.radians(angle_deg)
7       x = radius * math.cos(angle_rad)
8       y = radius * math.sin(angle_rad)
9       return x, y
10
11  def is_intersecting_and_right(line1, line2):
12      """
13      Проверяет, пересекаются ли траектории, включая проверку "помеха
        ↪  справа"
14      для продленных линий в случае совпадения конечных точек.
15      """
16      def determinant(a, b, c, d):
17          return a * d - b * c
18
19      def vector_cross_product(ax, ay, bx, by):
20          """Векторное произведение для определения относительного
            ↪  положения точки."""
21          return ax * by - ay * bx
22
23      def are_points_equal(p1, p2, epsilon=1e-6):
24          """Сравнивает два пункта с учетом допустимой погрешности."""
25          return abs(p1[0] - p2[0]) < epsilon and abs(p1[1] - p2[1]) <
            ↪  epsilon
26
27      def extend_point(p1, p2, factor=1.0):
28          """Продлевает точку p2 на основе направления от p1 к p2."""
29          dx = p2[0] - p1[0]
30          dy = p2[1] - p1[1]
31          return (p2[0] + dx * factor, p2[1] + dy * factor)
32
33      x1, y1, x2, y2 = *line1[0], *line1[1]
34      x3, y3, x4, y4 = *line2[0], *line2[1]
35
36      # Проверяем совпадение конечных точек
37      if are_points_equal((x2, y2), (x4, y4)):
38          '''print("Конечные точки совпадают. Проверяем продленные
            ↪  линии.")'''
39          # Продление линий
40          extended1 = extend_point((x1, y1), (x2, y2), factor=0.1)
41          extended2 = extend_point((x3, y3), (x4, y4), factor=0.1)
42          '''print(f"Продленная точка для линии 1: {extended1}")
43          print(f"Продленная точка для линии 2: {extended2}")'''
44
45          # Проверяем пересечение для продленных линий
46          vec1_x, vec1_y = extended1[0] - x1, extended1[1] - y1
47          vec2_x, vec2_y = extended2[0] - x3, extended2[1] - y3
48          cross = vector_cross_product(vec1_x, vec1_y, vec2_x, vec2_y)
49          return True, cross < 0
50
51      # Проверяем пересечение обычным образом
52      det = determinant(x2 - x1, x3 - x4, y2 - y1, y3 - y4)
53      if det == 0:
54          return False, False
55
56      t = determinant(x3 - x1, x3 - x4, y3 - y1, y3 - y4) / det
57      u = determinant(x2 - x1, x3 - x1, y2 - y1, y3 - y1) / det
58
```

```
    if 0 <= t <= 1 and 0 <= u <= 1:
        vec1_x, vec1_y = x2 - x1, y2 - y1
        vec2_x, vec2_y = x4 - x3, y4 - y3
        cross = vector_cross_product(vec1_x, vec1_y, vec2_x, vec2_y)
        return True, cross < 0

    return False, False

def process_intersection(data, iterations=3):
    radius = 1
    step_angle = 22.5

    # Определение точек и превосходства
    lanes_map = {
        "north": 0, "north-east": 2, "east": 4,
        "south-east": 6, "south": 8, "south-west": 10,
        "west": 12, "north-west": 14
    }
    directions_map = {
        "north": 1, "north-east": 3, "east": 5,
        "south-east": 7, "south": 9, "south-west": 11,
        "west": 13, "north-west": 15
    }
    superior_points = {lanes_map[lane] for lane in
    ↪  data["intersection"]["priority_lanes"]}
    '''print(f"Точки превосходства: {superior_points}")'''

    # Формирование траекторий
    vehicles = data["vehicles"]
    trajectories = []
    for vehicle in vehicles:
        start_idx = lanes_map[vehicle["lane"]]
        end_idx = directions_map[vehicle["direction"]]
        start_point = polar_to_cartesian(radius, start_idx *
        ↪  step_angle)
        end_point = polar_to_cartesian(radius, end_idx * step_angle)
        trajectories.append((start_point, end_point, start_idx))
        '''print(f"Траектория автомобиля {vehicle['vehicle_id']}:
        ↪  стартовая точка {start_idx}, конечная точка {end_idx}")'''


    # Итеративный процесс перераспределения приоритетов
    priorities = [1] * data["intersection"]["vehicle_count"]
    for iteration in range(iterations):
        '''print(f"\nИтерация {iteration + 1}")'''
        new_priorities = priorities[:]
        for i in range(len(trajectories)):
            for j in range(i + 1, len(trajectories)):
                start_idx_i = trajectories[i][2]
                start_idx_j = trajectories[j][2]

                # Проверяем "превосходство" только если траектории
                ↪  пересекаются
                intersects, is_right = is_intersecting_and_right(
                    (trajectories[i][0], trajectories[i][1]),
                    (trajectories[j][0], trajectories[j][1])
                )

                if intersects:
```

```
                    if new_priorities[i] == new_priorities[j]:  # Меняем приоритет только если они равны
                        if (start_idx_i in superior_points) != (start_idx_j in superior_points):
                            if start_idx_i not in superior_points:
                                if new_priorities[i] <= iteration + 2:
                                    new_priorities[i] += 1
                                    '''print(f"Траектория {j + 1} имеет превосходство над траекторией {i + 1}. Приоритет траектории {i + 1} увеличен до {new_priorities[i]}.")'''
                            elif start_idx_j not in superior_points:
                                if new_priorities[j] <= iteration + 2:
                                    new_priorities[j] += 1
                                    '''print(f"Траектория {i + 1} имеет превосходство над траекторией {j + 1}. Приоритет траектории {j + 1} увеличен до {new_priorities[j]}.")'''
                            else:
                                if is_right:
                                    if new_priorities[i] <= iteration + 2:
                                        new_priorities[i] += 1
                                        '''print(f"Автомобиль {vehicles[j]['vehicle_id']} помеха справа для автомобиля {vehicles[i]['vehicle_id']}. Приоритет {vehicles[i]['vehicle_id']} увеличен до {new_priorities[i]}.")'''
                                else:
                                    if new_priorities[j] <= iteration + 2:
                                        new_priorities[j] += 1
                                        '''print(f"Автомобиль {vehicles[i]['vehicle_id']} помеха справа для автомобиля {vehicles[j]['vehicle_id']}. Приоритет {vehicles[j]['vehicle_id']} увеличен до {new_priorities[j]}.")'''
            priorities = new_priorities[:]

    # Формирование результата
    result = {"order": []}
    for i, vehicle in enumerate(vehicles):
        result["order"].append({
            "vehicle_id": vehicle["vehicle_id"],
            "priority": priorities[i]
        })

    '''print(f"\nИтоговые приоритеты: {result['order']}")'''
    return result

# Пример входных данных
with open("input.txt", encoding="UTF-8") as file_in:
    input_data = json.load(file_in)

```

```
# Вывод результата
output = process_intersection(input_data)
print(json.dumps(output, indent=2))
```

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 Codeforces: https://codeforces.com/group/kKzhclDrqZ.

4.2.1. Информатика. 8–11 классы

Задача 4.2.1.1. Временное решение (13 баллов)

Темы: перебор, реализация.

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: 256 Мбайт.

Условие

В одном из лабораторных центров умного города ученики разработали искусственный интеллект, который анализирует возраст зданий по их фотографиям, помогая в прогнозировании их технического обслуживания. Однако из-за ошибки в коде система дает лишь косвенные подсказки о возрасте объекта.

После загрузки фотографий очередного здания были получены следующие подсказки:

1. возраст здания ближе к x годам, чем к y;
2. разница между возрастом здания и z, умноженная на k, более, чем в p раз больше половины возраста здания.

Вот такая замудренная подсказка! Ошибку, конечно, нужно найти и исправить, а сейчас разработайте временное решение: по указанным подсказкам определите возраст здания. Считаем, что подсказки верны.

Формат входных данных

Входные данные представляют собой пять строк с одним целым числом в каждой строке x, y, z, k, p соответственно. Каждое из чисел — целое положительное и не превосходит 200. Гарантируется, что ответ при заданных числах существует.

Формат выходных данных

Выведите единственное целое положительное число — возраст здания. Если возможных вариантов несколько — выведите минимальный. Гарантируется, что все

здания в городе имеют возраст, не превышающий 120.

Примеры

Пример №1

Стандартный ввод
70 30 76 4 3
Стандартный вывод
51

Пример №2

Стандартный ввод
101 16 141 9 5
Стандартный вывод
59

Примечания

«Ближе к x годам, чем к y» — подразумевается, что если возраст здания H, то $|H - x| < |H - y|$.

Решение

Переберем возможный возраст здания, начиная с 1 до 120, до тех пор, пока не выполнятся все условия, а именно: $|H - x| < |H - y|$ и $|H - z| \cdot k \cdot 2 > H \cdot p$.

Сложность решения: $\mathcal{O}(C)$.

Пример программы-решения

Ниже представлено решение на языке Python.

Python

```python
def solve():
    x, y, z, k, p = [int(input()) for i in range(5)]
```

```
3
4       for H in range(1, 121):
5           if abs(H - x) >= abs(H - y):
6               continue
7
8           dif = abs(H - z)
9
10          if dif * k * 2 > H * p:
11              print(H)
12              return
13
14
15  if __name__ == '__main__':
16      solve()
```

Критерии оценивания

Проверка в задаче полная — тестирование не прерывается при неверном ответе. Баллы будут начисляться пропорционально количеству пройденных тестов. Баллы за тестовые примеры не предусмотрены. Всего оцениваемых тестов 20 штук.

Максимальный балл за задачу — 13.

Задача 4.2.1.2. Типичные растения (17 баллов)

Темы: сортировки, перебор, жадный алгоритм.

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: 256 Мбайт.

Условие

В умном городе запущен экспериментальный агротехнический комплекс с теплицей. В теплице установлены n горшочков, в каждом из которых растет особый вид растения, дающий плоды. Растение в горшочке i вырастает и дает первые плоды за t_i дней, после этого растение ждет, пока соберут его плоды. Если растение i выросло, и его плоды собрали, то оно не растет, но продолжает давать плоды каждый раз через ровно t_i дней с момента прошлого сбора его плодов.

Изначально в нулевой день экспериментаторы посадили в этой теплице все n растений. Общая длительность эксперимента составляет T дней. Экспериментаторы хотят собрать из теплицы максимальное число плодов за эти T дней, однако, чтобы минимизировать внешние воздействия на микроклимат, в теплицу разрешается зайти в теплицу и собрать поспевшие плоды лишь единожды между днями 0 и T.

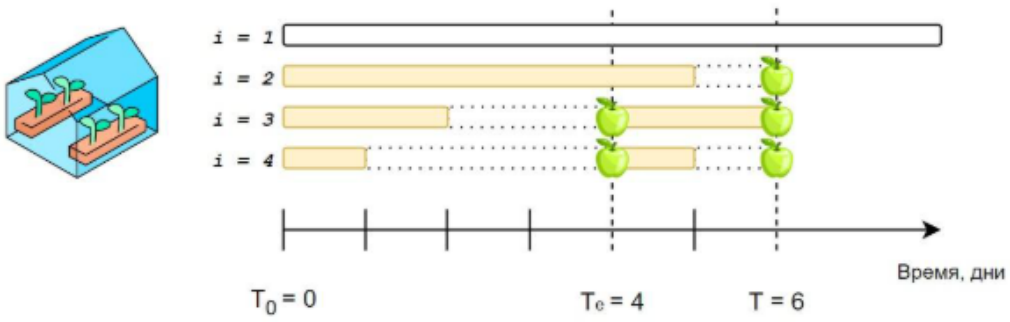

Рис. 4.2.1

По рис. 4.2.1 видно, что оптимально посетить теплицу в день 4 (также можно в день 2 или 3) и собрать плоды с горшков 3 и 4. В последний — шестой — день плоды собираются с растений в горшках 2, 3 и 4.

Определите, какое максимальное число плодов можно суммарно собрать за T дней, если в теплицу можно зайти не более одного раза для сбора плодов растений, не включая дни 0 и T. Заметьте, что в последний день эксперимента разрешается заходить в теплицу и собирать все поспевшие плоды.

Формат входных данных

В первой строке содержатся два целых числа n, T ($1 \leqslant n \leqslant 2{\cdot}10^5$; $1 \leqslant T \leqslant 10^9$) — количество горшочков в теплице и время в днях, доступное для выращивания плодов растений.

Во второй строке содержатся n целых чисел t_i ($1 \leqslant ti \leqslant 10^9$) — время вырастания и созревания плодов каждого растения.

Формат выходных данных

Выведите максимальное число плодов, которые можно собрать, согласно описанным условиям.

Примеры

Пример №1

Стандартный ввод
4 6 8 5 2 1
Стандартный вывод
5

Пример №2

Стандартный ввод
4 4 10 6 4 9
Стандартный вывод
1

Пример №3

Стандартный ввод
5 10 8 3 3 2 5
Стандартный вывод
9

Примечания

Обратите внимание, что если у растения не собрать плоды, то оно не начнет выращивать новые плоды, пока кто-то не соберет текущие плоды.

После того как растение i посадили в нулевой день, в день t_i растение вырастает, и в этот же день с него уже можно собрать первые плоды.

Решение

Условие задачи сводится к тому, чтобы собрать двойной урожай с максимального числа растений. Легко показать, что собрать двойной урожай можно только с растений, время созревания которых принадлежит некоторому отрезку $[0;\ X]$.

Не будем рассматривать такие $t_i > T$, так как с таких растений ни разу нельзя собрать урожай.

Время, в которое стоит зайти в теплицу и собрать урожай — это такое максимальное $ti \leqslant \dfrac{T}{2}$ (но не всегда только это значение, тем не менее, оно всегда не хуже). Если ни одного такого значения нет, то можно зайти в теплицу, когда угодно или не заходить ни разу, так как двойной урожай собрать нельзя.

Сложность решения: $\mathcal{O}(n \cdot \log n)$ или $\mathcal{O}(n)$, если не делать сортировку.

Пример программы-решения

Ниже представлено решение на языке Python.

Python

```python
import sys


def solve():
    n, T = map(int, sys.stdin.readline().split())
    t = list(map(int, sys.stdin.readline().split()))

    t.sort()
    while len(t) > 0 and t[-1] > T:
        t.pop()

    res = len(t)
    for i in range(len(t)):
        if t[i] << 1 <= T:
            res += 1
        else:
            break

    sys.stdout.write(str(res) + '\n')
    return


if __name__ == '__main__':
    solve()
```

Критерии оценивания

Ниже представлена таблица оценки за группы тестов.

Таблица 4.2.1

Группа	Дополнительные ограничения	Баллы
1	n=2	2
2	$n \leqslant 10;\ T \leqslant 100$	3
3	$n \leqslant 1000$	5
4	—	7

Проверка в каждой группе тестов полная — тестирование не прерывается при неверном ответе. Баллы будут начисляться пропорционально количеству пройденных тестов. Балл за задачу — сумма баллов за каждую подгруппу. Баллы за тестовые примеры не предусмотрены.

Задача 4.2.1.3. Восстановление фундамента (22 балла)

Темы: рекурсия, динамическое программирование.

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: 256 Мбайт.

Условие

В игре «Гексокрафт 2D», где все объекты представлены шестиугольными блоками, Вася построил на сервере пирамиду, состоящую из таких блоков. Однако его друг Петя, развлекаясь с различными механиками игры, случайно уничтожил несколько блоков в постройке Васи.

Вася расстроился — давайте поможем ему и оценим ущерб в цифрах. Главное, по его словам, починить основание пирамиды, так что сосредоточимся именно на нем. Изначально пирамида состояла из n уровней. Уровни в пирамиде нумеруются с верхнего — он имеет номер 1, соответственно самый нижний уровень имеет номер n. Блоки в каждом уровне h нумеруются слева направо, начиная с 1 и заканчивая h.

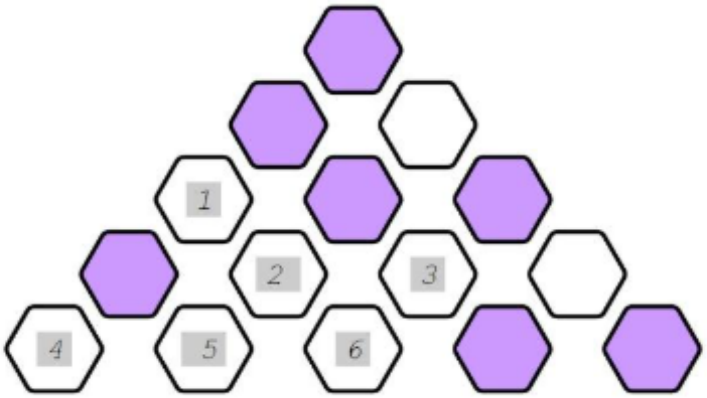

Рис. 4.2.2

Каждый блок в пирамиде может быть или заполнен, или быть пустым. Состояние каждого блока будет известно. Некоторые пустые блоки можно заполнить за один ход, но для того чтобы заполнить блок i на уровне $h > 1$, должны быть заполнены блоки $i - 1$ (если существует) и i (если существует) на уровне $h - 1$.

Определите минимальное число ходов для заполнения всех n блоков нижнего уровня пирамиды.

Формат входных данных

В первой строке входных данных содержится целое число n $(1 \leqslant n \leqslant 1000)$ — количество уровней в пирамиде. Далее содержатся n строк, описывающие начальное состояние пирамиды: в i-й строке содержится бинарная строка s_i длины i, если j-й блок i-го уровня пирамиды заполнен, то $s_{i,j} = 1$, иначе $s_{i,j} = 0$. Гарантируется, что блок пирамиды в самом верхнем уровне всегда заполнен ($s_{1,1} = 1$).

Формат выходных данных

Выведите единственное целое число — минимальное число ходов для того, чтобы заполнить n блоков нижнего уровня пирамиды.

Примеры

Пример №1

Стандартный ввод
2 1 01
Стандартный вывод
1

Пример №2

Стандартный ввод
4 1 00 101 1011
Стандартный вывод
4

Пример №3

Стандартный ввод
5 1 10 011 1000 00011
Стандартный вывод
6

Решение

Задачу можно решить рекурсивно.

Пусть текущий ответ равен нулю. Определим $F(i, j)$ — это функция, которая заполняет все блоки, необходимые для заполнения j-го блока в i-м уровне пирамиды, включая сам этот блок. Принцип ее работы таков:

- если текущий блок уже заполнен — возвращаем 0;
- если вызов некорректен (например, на i-м уровне нет j-го блока) — возвращаем 0;

- если текущий блок пустой, то надо убедиться в том, что заполнены блоки выше него — заполняем текущий блок и возвращаем $1 + F(i-1, j-1) + F(i--1, j)$.

Будем по порядку рассматривать блоки нижнего уровня пирамиды слева направо, как только очередной блок пустой — будем запускать функцию F из текущего блока и прибавлять к ответу возвращаемое ей значение.

Также задачу можно решить без рекурсии — итеративно, заполняя необходимые блоки уровень за уровнем, начиная с нижнего.

Сложность решения: $\mathcal{O}(n^2)$.

Пример программы-решения

Ниже представлено решение на языке Python.

Python

```python
import sys


def solve():
    n = int(sys.stdin.readline())

    f = [[0 for i in range(h + 1)] for h in range(n)]

    for i in range(n):
        s = str(sys.stdin.readline().strip())
        for j in range(i + 1):
            if s[j] == '1':
                f[i][j] = 1

    need_to_fill = set()
    for i in range(n):
        if f[-1][i] == 0:
            need_to_fill.add(i)

    cur_level = n - 1
    res = 0

    while len(need_to_fill) != 0:
        fill_in_next_level = set()
        res += len(need_to_fill)

        for i in need_to_fill:
            if i - 1 >= 0:
                if f[cur_level - 1][i - 1] == 0:
                    fill_in_next_level.add(i - 1)
                    f[cur_level - 1][i - 1] = 1

            if i < len(f[cur_level - 1]):
                if f[cur_level - 1][i] == 0:
                    fill_in_next_level.add(i)
                    f[cur_level - 1][i] = 1

        need_to_fill, fill_in_next_level = fill_in_next_level, set()
        cur_level = cur_level - 1

```

```
41      sys.stdout.write(str(res) + '\n')
42      return
43
44
45  if __name__ == '__main__':
46      solve()
```

Критерии оценивания

Ниже представлена таблица оценки за группы тестов.

Таблица 4.2.2

Группа	Дополнительные ограничения	Баллы
1	$n \leqslant 3$	4
2	$n \leqslant 5$	4
3	$n \leqslant 50$	4
4	—	10

Проверка в каждой группе тестов полная — тестирование не прерывается при неверном ответе. Баллы будут начисляться пропорционально количеству пройденных тестов. Балл за задачу — сумма баллов за каждую подгруппу. Баллы за тестовые примеры не предусмотрены.

Задача 4.2.1.4. Оптимизация обслуживания (23 балла)

Темы: конструктив, жадный алгоритм, сортировки.

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: 256 Мбайт.

Условие

В умном городе работают две вышки интернета, которые обслуживают n кварталов города. Сейчас администрация планирует установку третьей вышки, и им нужна помощь.

Каждая вышка описывается перестановкой[1] из n элементов, элемент последовательности определяет приоритет, которым обладает обслуживание соответствующего квартала. Чем число больше, тем выше приоритет и, соответственно, выше качество интернет-соединения. Первая вышка описывается перестановкой a, вторая вышка — перестановкой b, третья вышка — перестановкой c.

[1] Перестановкой длины n является массив из n целых чисел, в котором каждое число от 1 до n встречается ровно один раз.

Общее качество интернета в умном городе сейчас (до установки третьей вышки) определяется последовательностью из s из n элементов, где $s_i = max(a_i, b_i)$, а эффективностью третьей вышки является количество индексов i таких, что $c_i > s_i$. Задача: определить оптимальную конфигурацию третьей вышки для достижения ее максимальной эффективности. Если возможных вариантов несколько — допускается вывести любой из них.

Формат входных данных

В первой строке содержится единственное целое число n ($1 \leqslant n \leqslant 10^5$) — количество кварталов в умном городе. Во второй строке содержатся n целых чисел a ($1 \leqslant a \leqslant n$) — конфигурация первой вышки. В третьей строке содержатся n целых чисел b ($1 \leqslant b \leqslant n$) — конфигурация второй вышки. Гарантируется, что a и b являются перестановками.

Формат выходных данных

Выведите любую перестановку для конфигурации третьей вышки, которая максимизирует значение эффективности третьей вышки.

Примеры

Пример №1

Стандартный ввод
3 1 2 3 3 2 1
Стандартный вывод
1 3 2

Пример №2

Стандартный ввод
6 1 5 2 3 4 6 2 6 3 1 5 4
Стандартный вывод
3 1 4 5 6 2

Пример №13

Стандартный ввод
5 1 2 3 4 5 1 2 3 4 5

Стандартный вывод
2 3 4 5 1

Решение

Обозначим массив m такой, что $m_i = max(a_i, b_i)$. Найдем минимальное значение в этом массиве — пусть это $m_x = y$, тогда присвоим p_x такое значение, что больше y, но не больше n, и его еще не использовали. Затем повторим операцию со вторым по самой малой величине значением в m. И так далее, пока не сможем больше сделать ни одного присваивания в p. Остальные значения в p (незаполненные) можно заполнить любыми числами, чтобы в конце была перестановка, какие именно числа и в каком порядке — на ответ не повлияет.

Чтобы быстро находить k-е минимальное число в m и знать его индекс, можно запомнить с каждым значением его индекс (сделать пару «значение» — «индекс») и отсортировать по неубыванию «значений».

Сложность решения: $\mathcal{O}(n \cdot \log n)$.

Пример программы-решения

Ниже представлено решение на языке Python.

Python

```python
import sys


def solve():
    n = int(sys.stdin.readline())
    a = list(map(int, sys.stdin.readline().split()))
    b = list(map(int, sys.stdin.readline().split()))

    for i in range(n):
        a[i] = [max(a[i], b[i]), i]

    a.sort()
    b = [-1 for i in range(n)]
    not_used = set([i for i in range(1, n + 1)])

    st = min(a[0][0], n) + 1
    for v, ind in a:
        while st <= v:
            st += 1

        if st > n:
            break
```

```
            else:
                not_used.discard(st)
                b[ind] = st
                st += 1

        for i in range(n):
            if b[i] == -1:
                b[i] = not_used.pop()

        sys.stdout.write(' '.join(map(str, b)) + '\n')
        return


if __name__ == '__main__':
    solve()
```

Критерии оценивания

Ниже представлена таблица оценки за группы тестов.

Таблица 4.2.3

Группа	Дополнительные ограничения	Баллы
1	n=3	2
2	$n \leqslant 10$	4
3	$n \leqslant 500$	6
4	—	11

Проверка в каждой группе тестов полная — тестирование не прерывается при неверном ответе. Баллы будут начисляться пропорционально количеству пройденных тестов. Балл за задачу — сумма баллов за каждую подгруппу. Баллы за тестовые примеры не предусмотрены.

Задача 4.2.1.5. Эффективный доставщик (25 баллов)

Темы: графы, динамическое программирование.

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: 256 Мбайт.

Условие

На складском пространстве умного города имеется один свободный автономный грузовой автомобиль, который можно сдавать в аренду для перевозки грузов между поселениями. Грузовик отправляется в путешествие, но ему нужно вернуться

в умный город через T ч, так как его услугами снова потребуется воспользоваться в умном городе.

В умном городе есть n поселений, некоторые из них соединены дорогами. Два поселения соединяет не более, чем одна дорога. Всего дорог m штук, и каждая из них описывается своим числом w_i — временем в часах для перемещения между соответствующими поселениями.

Заказы на перевозку грузов есть всегда, поэтому считаем, что каждая поездка между двумя поселениями считается как выполненный заказ. Полагаем, что погрузка и разгрузка выполняются моментально, а грузовику не требуется время на техническое обслуживание.

Определите, какое максимальное число заказов может выполнить грузовик в пределах отведенного времени.

Формат входных данных

В первой строке входных данных содержится единственное целое число n — количество поселений, $2 \leqslant n \leqslant 100$. Во второй строке содержится единственное целое число m — количество дорог между всеми поселениями, $1 \leqslant m \leqslant \dfrac{n(n-1)}{2}$. Умный город является поселением с номером один.

Далее следуют m строк, описывающих дороги между поселениями. Каждая дорога описывается тремя целыми числами u, v, w ($1 \leqslant u < v \leqslant n;\ 1 \leqslant w \leqslant 200$), где u и v — это номера двух поселений, между которыми есть дорога, а w — время в часах для перемещения по этой дороге. Гарантируется, что каждая пара (u, v) присутствует во входных данных не более одного раза.

В последней строке содержится единственное целое число $T(1 \leqslant T \leqslant 100)$ — время в часах, отводимое на аренду грузового автомобиля.

Формат выходных данных

Выведите единственное целое число — максимальное число заказов, которые может выполнить грузовик в указанное время. Помните, что через T часов грузовик должен быть на складе умного города.

Примеры

Пример №1

Стандартный ввод
3 3 1 2 8 2 3 2 1 3 1 5
Стандартный вывод
4

Пример №2

Стандартный ввод
3 3 1 2 8 2 3 2 1 3 1 6
Стандартный вывод
6

Пример №3

Стандартный ввод
5 7 1 2 4 1 3 9 3 5 2 2 3 1 4 5 13 1 4 8 2 5 1 10
Стандартный вывод
4

Решение

Пусть $dp[i][j]$ — максимальное число заказов, которое можно выполнить в поселении j в момент времени i. Матрица dp имеет размер $(T+1) \times n$, изначально заполнена $-inf$, кроме ячейки $dp[0][0]$ — там находится значение 0.

Будем последовательно перебирать i с 1 до T включительно.

Внутри будем также перебирать текущее поселение j.

Посмотрим на каждое поселение u из текущего, в которое можем переместиться.

Можем обновить $dp[i][j] = max(dp[i-1][j],\ dp[i][j], dp[i-w_{ju}][u]+1)$.

Некоторые особые случаи:

- $dp[i-w_{j\to u}][u]$ рассматривается только когда $i-w_{j\to u}$ больше или равно нулю;
- $dp[i-w_{j\to u}][u]+1$ — слагаемое $(+1)$ используется только когда текущая комната не является стартовой (т. е. в ней есть кнопка).

Сложность решения: $\mathcal{O}(T \cdot m)$.

Так как цикл на T итераций, в нем рассматривается каждое ребро два раза, остальными операциями пренебрегаем.

Пример программы-решения

Ниже представлено решение на языке Python.

Python

```python
import sys

INF = 1 << 90


def solve():
    n = int(sys.stdin.readline())
    m = int(sys.stdin.readline())

    g = [[] for i in range(n)]

    for i in range(m):
        u, v, w = map(int, sys.stdin.readline().split())
        u, v = u - 1, v - 1
        g[u].append([v, w])
        g[v].append([u, w])

    T = int(sys.stdin.readline())

    dp = [[-INF for i in range(n)] for time in range(0, T + 1)]
    dp[0][0] = 0

    for time in range(1, T + 1):
        for node in range(n):
            dp[time][node] = max(dp[time - 1][node], dp[time][node])

            for u, w in g[node]:

                if time - w < 0:
                    continue

                dp[time][node] = max(dp[time][node], dp[time - w][u] +
                ↪  1)

    sys.stdout.write(str(dp[T][0]) + '\n')
```

```
35      return
36
37
38  if __name__ == '__main__':
39      solve()
```

Критерии оценивания

Ниже представлена таблица оценки за группы тестов.

Таблица 4.2.4

Группа	**Дополнительные ограничения**	**Баллы**
1	$n \leqslant 3$	2
2	$n \leqslant 5;\ T \leqslant 10$	3
3	$n \leqslant 10;\ m = n - 1$	6
4	—	14

Проверка в каждой группе тестов полная — тестирование не прерывается при неверном ответе. Баллы будут начисляться пропорционально количеству пройденных тестов. Балл за задачу — сумма баллов за каждую подгруппу. Баллы за тестовые примеры не предусмотрены.

4.2.2. Физика. 8–9 классы

Задача 4.2.2.1. Необычные качели (20 баллов)

Тема: закон сохранения импульса.

Условие

В умном городе поставили новые необычные качели. Но прежде чем пускать на них ребят, для их безопасности, надо все просчитать! Упрощенная схема качелей выглядит так: на абсолютно гладкой поверхности лежит брусок массой M, к которому прикреплен математический маятник длиной L и массой m (см. рис. 4.2.3). Найдите скорость бруска относительно поверхности земли в момент времени, когда угловая скорость маятника в нижней точки равна ω.

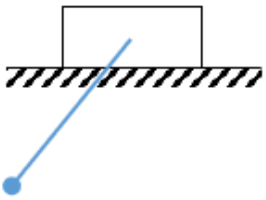

Рис. 4.2.3

Решение

На рис. 4.2.3 $\vec{u}$ — скорость бруска относительно земли, $\vec{v'}$ — скорость маятника относительно бруска.

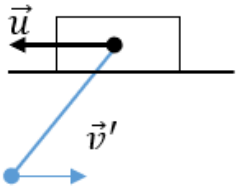

Рис. 4.2.4

Найдем скорость маятника относительно земли:

$$v = v' - u, \text{ где } v' = \omega L \to v = \omega L - u.$$

Запишем закон сохранения импульса относительно CO, связанной с землей:

$$uM = m(\omega L - u).$$

Преобразуем:

$$uM + mu = m\omega L.$$

Найдем:

$$u = m\omega\frac{L}{m+M}.$$

Ответ: $m\omega\dfrac{L}{m+M}$.

Критерии оценивания

Таблица 4.2.5

Критерий	**Баллы**	**Подзадача**
1	8	Найдена скорость маятника относительно земли
2	8	Записан закон сохранения импульса
3	4	Получено выражение для скорости бруска

Задача 4.2.2.2. Сосуд с воздухом (15 баллов)

Темы: второй закон Ньютона, сила Архимеда.

Условие

В умном городе провели эксперимент. Из сосуда, заполненного жидкостью, равномерно достают два связанных невесомой нитью груза, сделанных из одного материала, к одному из которых прикреплен динамометр (см. рис. 4.2.5).

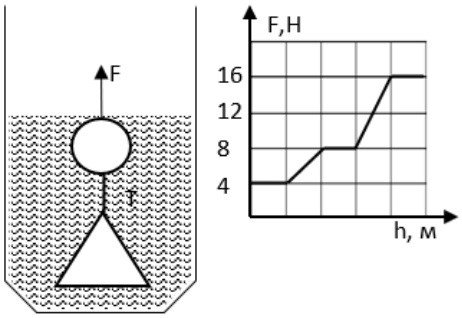

Рис. 4.2.5

Зависимость показаний динамометра F от высоты подъема представлена на графике. Найдите натяжение нити T в момент времени, когда первый груз полностью выйдет из жидкости. Ответ округлите до десятых.

Решение

Сила F остается постоянной, пока не изменяется сила Архимеда, действующая на грузы. Поэтому горизонтальные участки 1, 3, 5 на рис. 4.2.6 отвечают случаям,

когда уровень жидкости находится ниже нижнего груза, между верхним и нижним грузами и выше верхнего груза соответственно. Наклонные участки 2 и 4 соответствуют частичному погружению нижнего и верхнего грузов.

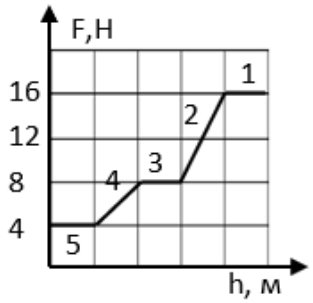

Рис. 4.2.6

На участке 1 сила $F = 16$ Н равна сумме весов верхнего P_1 и нижнего P_2 грузов:

$$16 = P_1 + P_2. \tag{4.2.1}$$

На участке 3 сила $F = 8$ Н равна сумме P_1 и P_2 за вычетом силы Архимеда F_{A2}, действующей на нижний груз:

$$8 = P_1 + P_2 - F_{A2}. \tag{4.2.2}$$

На участке 5 сила $F = 4$ Н равна сумме P_1 и P_2 за вычетом силы Архимеда F_{A1} и F_{A2}, действующих на нижний и верхний груз:

$$4 = P_1 + P_2 - F_{A1} - F_{A2}. \tag{4.2.3}$$

Вычитая (4.2.2) из (4.2.1), получим $F_{A2} = 8$ Н.

А вычитая (4.2.3) из (4.2.2), получим $F_{A1} = 4$ Н.

На третьем участке графика верхний груз полностью вышел из жидкости, и искомая сила натяжения T нижней нити равна разности сил тяжести и Архимеда, действующих на груз:

$$T = P_2 - F_{A2}. \tag{4.2.4}$$

Чтобы найти P_2, заметим, что так как грузы выполнены из одинакового материала, отношение весов грузов $\dfrac{P_2}{P_1}$ равно отношению их объемов, а следовательно, сил Архимеда:

$$\frac{P_2}{P_1} = \frac{F_{A2}}{F_{A1}} = 2. \tag{4.2.5}$$

Из (4.2.1) и (4.2.5) найдем вес нижнего груза: $P_2 = 10{,}7$ Н.

Подставив найденные значения P_2 и F_{A2} в (4.2.4), получим $T = 2{,}7$ Н.

Примечание. Соотношение (4.2.5) также можно получить, сравнивая наклоны кривой на участках 2 и 4 графика, представленного на рис. 4.2.6.

Ответ: 2,7 Н.

Критерии оценивания

Таблица 4.2.6

Критерий	Баллы	Подзадача
1	1	Установлено, какой участок графика соответствует положению грузов в жидкости
2	2	Записано уравнение сил для полностью извлеченных из воды грузов
3	2	Записано уравнение сил для случая, когда из воды извлечен только верхний груз
4	2	Записано уравнение сил для случая, когда оба груза под водой
5	2	Найдены значения сил Архимеда, действующих на каждый груз
6	3	Найдено значение натяжения нити T
7	3	Получено соотношение объемов грузов

Задача 4.2.2.3. Короткие ролики (25 баллов)

Темы: количество теплоты, теплоемкость.

Условие

Все мы в умном городе любим иногда вечером посмотреть короткие ролики в любимых социальных сетях. Нам попался один интересный ролик, где раскаленный шарик кладут на лед. Мы решили рассчитать, как глубоко этот шарик сможет «проплавить» лед, прежде чем остановится.

Возьмем оловянную картечь массой 70 г, нагреем до температуры 200 °C и положим на лед. Определите глубину (самую нижнюю точку), на которую картечь погрузится в лед, если удельная теплоемкость олова 230 Дж/(кг °C), плотность олова 7,3 г/см^3, плотность льда 0,9 г/см^3, удельная теплота плавления льда $3{,}3 \cdot 10^5$ Дж/кг. Температура окружающей среды 0 °C. Картечь считайте шаром.

Решение

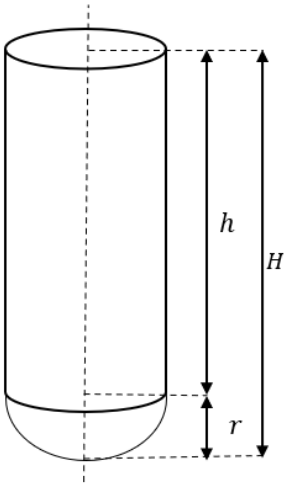

Рис. 4.2.7

Так как картечь — шар, радиус этого шара равен

$$r = \sqrt[3]{\frac{3 - V_c}{4\pi}},$$

где $V_c = \dfrac{M}{\rho_0}$. Отсюда

$$r = \sqrt[3]{\frac{3M}{4\pi\rho_0}}. \tag{4.2.6}$$

Поставим значение $r = \sqrt[3]{\dfrac{3 \cdot 0{,}07}{4\pi 7\,300}} = 0{,}013$ м.

Картечь радиуса r расплавит объем льда, состоящий из объема цилиндра $\pi r^2 h$ и объема половины шара $\dfrac{2}{3}\pi r^3$, т. е. $\pi r^2 h + \dfrac{2}{3}\pi r^3$, где h — высота цилиндра.

Для плавления льда нужно потратить количество теплоты равное:

$$Q = \lambda(\pi r^2 h + \frac{2}{3}\pi r^3)\rho_1, \tag{4.2.7}$$

где λ — удельная теплота плавления льда, ρ_1 — плотность льда.

Количество теплоты, выделенное картечью при остывании:

$$Q = \frac{4}{3}\pi r^3 \rho_0 c(t - t_1), \tag{4.2.8}$$

где c — удельная теплоемкость олова.

Приравнивая (4.2.7) и (4.2.8), получим

$$\frac{4}{3}\pi r^3 \rho_0 c(t - t_1) = \lambda(\pi r^2 h + \frac{2}{3}\pi r^3)\rho_1,$$

откуда

$$h = \frac{r(4\rho_0 c(t - t_1) - 2\lambda\rho_1)}{3\lambda\rho_1}. \tag{4.2.9}$$

Подставим (4.2.6) в (4.2.9):

$$h = \sqrt[3]{\frac{3M}{4\pi\rho_0}} \cdot \frac{4\rho_0 c(t - t_1) - 2\lambda\rho_1}{3\lambda\rho_1}.$$

Подставим численные значения:

$$h = \sqrt[3]{\frac{3 \cdot 0{,}07}{4\pi \cdot 7\,300}} \cdot \frac{4 \cdot 7\,300 \cdot 230 \cdot 200 - 2 \cdot 3{,}3 \cdot 10^5 \cdot 900}{3 \cdot 3{,}3 \cdot 10^5 \cdot 900} = 0{,}011 \text{ м}.$$

Глубина погружения картечи $H = 0{,}013 + 0{,}011 = 0{,}024$ м (высота цилиндра плюс радиус шара).

Ответ: 0,024 м.

Критерии оценивания

Таблица 4.2.7

Критерий	**Баллы**	**Подзадача**
1	3	Записано выражение (4.2.6) для радиуса шара
2	4	Записано выражение для объема льда
3	4	Записано выражение для нахождения количества теплоты, потраченной на плавление льда
4	4	Записано выражение для нахождения количества теплоты, отданной картечью
5	6	Получено выражение для высоты цилиндра h
6	3	Найдено числовое значение h
7	1	Найдена глубина погружения картечи

Задача 4.2.2.4. Квадрокоптеры (15 баллов)

Тема: относительная скорость.

Условие

В умном городе стало много квадрокоптеров, и понадобилось срочно разработать алгоритм, помогающий предотвратить их столкновение. Два квадрокоптера летают на одной высоте с постоянными и одинаковыми по модулю скоростями:

$$|\vec{v_1}| = |\vec{v_2}| = |\vec{v}|.$$

В какой момент времени расстояние между ними будет минимальным и чему оно будет равно, если их начальное взаимное расположение показано на рис. 4.2.8? В начальный момент времени расстояние между ними S.

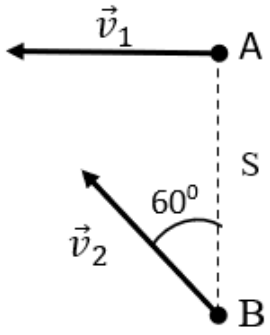

Рис. 4.2.8

Решение

Перейдем в систему отсчета, связанную с «коптером» 1. Ее начало отсчета поместим в точку A. На рисунке скорость $\vec{u}$ — это относительная скорость, которая по модулю равна v и направлена под углом 30° к линии $AB = S$. Найдем минимальное расстояние между «коптерами». Для этого опустим перпендикуляр AC из A на продолжение скорости u. Из рис. 4.2.9 $AC = S \cdot \sin 30^\circ = \dfrac{S}{2}$.

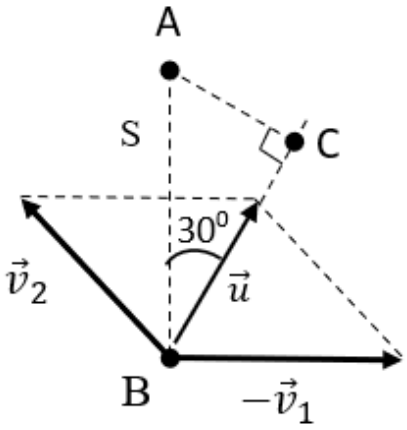

Рис. 4.2.9

Время, за которое они сблизятся на минимальное расстояние:

$$t = \frac{S \cdot \cos 30^\circ}{v} = \frac{\sqrt{3}S}{2v}.$$

Ответ: $t = \dfrac{\sqrt{3}S}{2v}$.

Критерии оценивания

Таблица 4.2.8

Критерий	**Баллы**	**Подзадача**
1	3	Проведены рассуждения о выборе системы отсчета
2	2	Нарисован рисунок
3	2	Показано минимальное расстояние на рисунке между «коптерами»
4	4	Найдено аналитическое выражение для минимального расстояние и проведены расчеты
5	4	Найдено время

Задача 4.2.2.5. Тепло для умного города (25 баллов)

Тема: законы постоянного тока.

Условие

Для умного города разработали модульный электрический нагреватель, нагревательные элементы которого можно подключать по-разному. Систему испытывали с двумя нагревательными элементами разных сопротивлений. При включении только первого элемента электрический нагреватель нагрелся до температуры 110 °C, а если включить только второй — нагреется до температуры 200 °C. До какой температуры нагреется электрический нагреватель, если нагревательные элементы будут включены последовательно или параллельно? Температуру окружающей среды примите равной 20 °C.

Решение

Пусть сопротивление нагревательных элементов будет соответственно R_1 и R_2, а напряжение в цепи U. Тогда

$$\frac{U^2}{R_1} = B(t_1 - t_0), \tag{4.2.10}$$

$$\frac{U^2}{R_2} = B(t_2 - t_0). \tag{4.2.11}$$

Последовательное соединение:

$$\frac{U^2}{R_1 + R_2} = B(t_3 - t_0). \tag{4.2.12}$$

Параллельное соединение:

$$\frac{U^2(R_1 + R_2)}{R_1 R_2} = B(t_4 - t_0), \tag{4.2.13}$$

где t_3, t_4 — температуры, до которых нагреется электрический нагреватель, B — коэффициент пропорциональности.

Выразим из (4.2.10) и (4.2.11) $\dfrac{B}{U^2}$ и приравняем левые части:

$$\frac{t_1 - t_0}{R_1} = \frac{t_2 - t_0}{R_2} \Rightarrow \frac{R_2}{R_1} = \frac{180}{90} = 2.$$

Выразим из (4.2.12) и (4.2.13) $\dfrac{B}{U^2}$ и приравняем левые части к (4.2.10):

$$\frac{90}{R_1} = \frac{t_3 - t_0}{R_1 + R_2} \Rightarrow t_3 = 290 \text{ °C},$$

$$\frac{90}{R_1} = \frac{(t_4 - t_0)(R_1 + R_2)}{R_1 R_2} \Rightarrow t_4 = 80 \text{ °C}.$$

Ответ:

- при последовательном — $t = 290$ °C;
- при параллельном — $t = 80$ °C.

Критерии оценивания

Таблица 4.2.9

Критерий	Баллы	Подзадача
1	1	Записаны уравнения, описывающие связь тепловой мощности с температурой для отдельных включений первого и второго элементов
2	3	Записано уравнение, описывающее связь тепловой мощности с температурой при последовательном включении элементов
3	3	Записано уравнение, описывающее связь тепловой мощности с температурой при параллельном включении элементов
4	6	Найдено аналитическое выражение для температуры при последовательном включении элементов
5	3	Произведен расчет температуры при последовательном включении элементов
6	6	Найдено аналитическое выражение для температуры при параллельном включении элементов
7	3	Произведен расчет температуры при параллельном включении элементов

4.2.3. Физика. 10–11 классы

Задача 4.2.3.1. Диоды (20 баллов)

Тема: законы Кирхгофа.

Условие

В схеме, изображенной на рис. 4.2.10, найдите ток, идущий через сопротивление R_2.

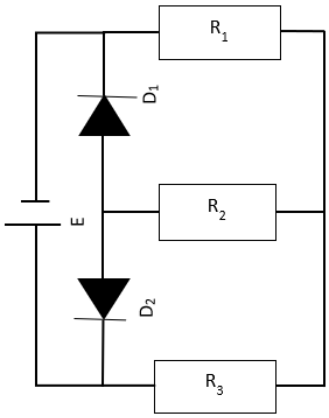

Рис. 4.2.10

ЭДС батареи $E = 9$ В, остальные параметры элементов схемы показаны на рисунке. $R_1 = R_3 = 2R$, $R_2 = R = 4{,}5$ Ом. Диоды и ЭДС являются идеальными.

Решение

Первый вариант решения

Диод D_2 закрыт, диод D_1 открыт.

Альтернативная схема представлена на рис. 4.2.11.

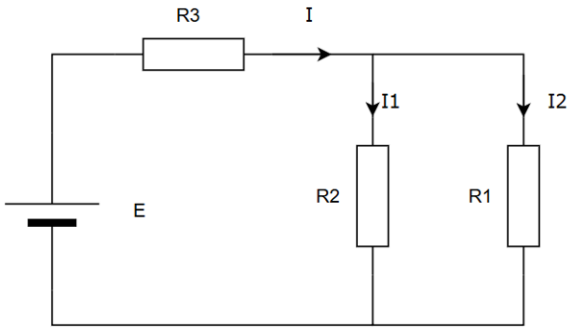

Рис. 4.2.11

Система уравнений по законам Кирхгофа:

$$I = I_1 + I_2;$$

$$E = R_1 I_1 + R_3 I;$$

$$E = R_2 I_2 + R_3 I.$$

Решая систему уравнений, получим:

$$I = \frac{E(R_1 + R_2)}{R_1R_3 + R_1R_2 + R_2R_3};$$

$$I_1 = \frac{ER_2}{R_1R_3 + R_1R_2 + R_2R_3};$$

$$I_2 = \frac{ER_1}{R_1R_3 + R_1R_2 + R_2R_3}.$$

Второй вариант решения

Расставим токи в цепи.

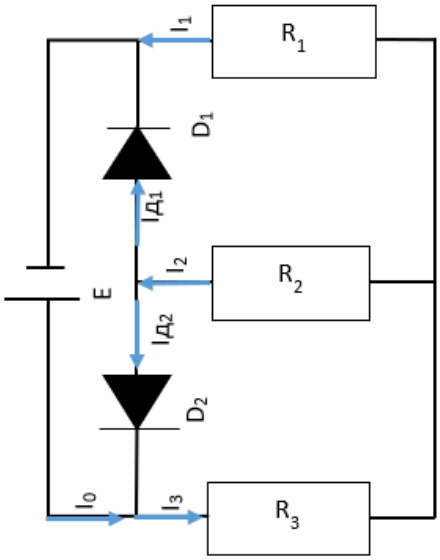

Рис. 4.2.12

Запишем систему уравнений по законам Кирхгофа:

$$I_0 + I_{D2} = I_3; \tag{4.2.14}$$

$$I_{D1} + I_{D2} = I_2; \tag{4.2.15}$$

$$I_1 + I_2 = I_3; \tag{4.2.16}$$

$$E = R_1I_1 + R_3I_3; \tag{4.2.17}$$

$$E = R_2I_2 + R_3I_3; \tag{4.2.18}$$

$$0 = R_1I_1 - R_2I_2. \tag{4.2.19}$$

Подставим численные значения в уравнений (4.2.17)–(4.2.19), получим:

$$9 = 9I_1 + 9I_3 \Rightarrow 1 = I_1 + I_3; \tag{4.2.20}$$

$$9 = 4{,}5I_2 + 9I_3 \Rightarrow 2 = I_1 + 2I_3; \tag{4.2.21}$$

$$0 = 9I_1 - 4{,}5I_2 \Rightarrow I_2 = 2I_1. \tag{4.2.22}$$

Подставим (4.2.22) в (4.2.16):

$$I_3 = 3I_1,$$

а теперь подставим в (4.2.20) $I_1 = \frac{1}{4}$ А, из (4.2.22) следует, что $I_2 = \frac{1}{2}$ А.

Примечания

$$0 = R_2I_2 + R_3I_3.$$

Так как сумма двух положительных величин не может быть равной 0, ток через диод 2 не идет (токи I_3 и I_2 не могут быть отрицательными), поэтому получаем первое решение.

Возможно другое направление токов в схеме.

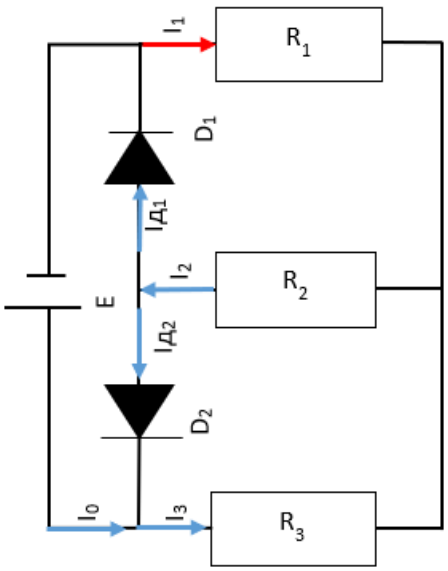

Рис. 4.2.13

Запишем систему уравнений по законам Кирхгофа:

$$I_0 + I_{D2} = I_3; \tag{4.2.23}$$

$$I_{D1} + I_{D2} = I_2; \tag{4.2.24}$$

$$I_2 = I_1 + I_3; \tag{4.2.25}$$

$$E = R_1 I_1 + R_3 I_3; \tag{4.2.26}$$

$$E = R_2 I_2 + R_3 I_3; \tag{4.2.27}$$

$$0 = R_2 I_2 + R_3 I_3; \tag{4.2.28}$$

$$0 = R_1 I_1 + R_2 I_2. \tag{4.2.29}$$

Сумма двух положительных величин не может быть равной 0, поэтому из равенства (4.2.29) вытекает противоречие (уравнение неверное).

Сумма двух одних и тех же величин равна разным значениям, поэтому из (4.2.27) и (4.2.28) вытекает противоречие.

Единственное правильное направление токов представлено в решении 1.

Ответ: 0,5 А.

Критерии оценивания

Таблица 4.2.10

Критерий	**Баллы**	**Подзадача**
1	6	Правильно расставлены возможные направления токов в цепи и представлена эквивалентная схема
2	6	Правильно записаны законы Кирхгофа
3	4	Правильно найдено выражение для тока
4	4	Правильно рассчитан ток

Задача 4.2.3.2. Рамка в магнитном поле (20 баллов)

Тема: электромагнитная индукция.

Условие

В умном городе решили добывать электроэнергию из магнитного поля. Для этого собрали экспериментальную установку. В однородном магнитном поле индукцией 6 мТл перпендикулярно силовым линиям находится квадратная рамка из латунной проволоки со стороной 30 см. Какой заряд протечет по рамке за время ее поворота на угол 45°? Площадь поперечного сечения латунной проволоки 4 мм^2, удельное сопротивление 0,025 (Ом·мм^2)/м. Результат округлите до целого и представьте в милликулонах (мКл).

Решение

$$I = \frac{\Delta q}{\Delta t} \Rightarrow \Delta q = I \cdot \Delta t. \tag{4.2.30}$$

Сила тока в проводнике найдем по закону Ома:

$$I = \frac{|\varepsilon_i|}{R}. \tag{4.2.31}$$

Площадь квадрата $S = a^2$.

Сопротивление проволоки:

$$R = \rho\frac{l}{S_c} = \rho\frac{4a}{S_c}. \tag{4.2.32}$$

S_c — площадь поперечного сечения проводника.

ЭДС индукции:

$$\varepsilon_i = -\frac{\Delta\Phi}{\Delta t} = \frac{\Phi_1 - \Phi_2}{\Delta t}, \tag{4.2.33}$$

где:

$$\Phi_1 = B \cdot S \cdot \cos 0,$$

$$\Phi_2 = B \cdot S \cdot \cos 45.$$

Подставив (4.2.31), (4.2.32) и (4.2.33) в (4.2.30) получим:

$$\Delta q = I\Delta t = -\frac{\Delta\Phi}{\Delta t R}\Delta t = \frac{\Phi_1 - \Phi_2}{R} = \frac{B \cdot S}{R}(\cos 0 - \cos 45) = \frac{B \cdot S_c \cdot a}{4\rho \cdot l} \cdot \frac{2 - \sqrt{2}}{2};$$

$$\Delta q = \frac{6 \cdot 10^{-3} \cdot 4 \cdot 10^{-6} \cdot 0{,}3}{4 \cdot 0{,}025 \cdot 10^{-6}} \cdot \frac{2 - \sqrt{2}}{2} = 0{,}021 \text{ Кл}.$$

Ответ: 21 мКл.

Критерии оценивания

Таблица 4.2.11

Критерий	**Баллы**	**Подзадача**
1	6	Правильно записаны формулы расчета ЭДС индукции, силы тока и сопротивления
2	2	Правильно записана формула расчета сопротивления
3	8	Правильно записана формула нахождения протекающего по проводнику заряда
4	4	Правильно выполнены расчеты

Задача 4.2.3.3. Стеклянная пластинка (15 баллов)

Тема: законы геометрической оптики.

Условие

Инженеры умного города проектируют прибор, измеряющий угол между падающим и отраженным лучом. При очередном испытании прибора использовали погруженную в воду плоскопараллельную пластинку из акрилового стекла. При горизонтальном положении пластинки прибор измерил угол в 60° 30′. Найдите изменение угла γ, образованного отраженным и преломленным лучами (как показано на рис. 4.2.14), если повернуть пластинку на $\tau = 15°$. Показатели преломления воды ($n_в$) и акрилового стекла ($n_с$) равны 1,33 и 1,49 соответственно. Ответ представьте в виде 0° 0′, округлите до целого значения угловых минут.

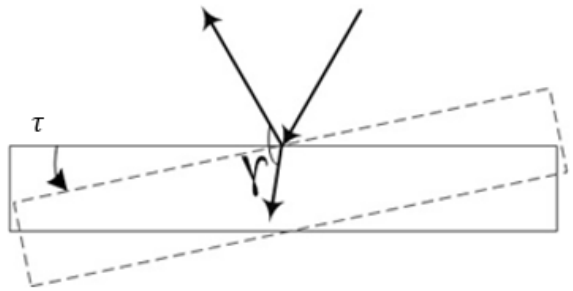

Рис. 4.2.14

Решение

Из рис. 4.2.15 видно, что первоначальный угол падения $\alpha = 30° 15′$.

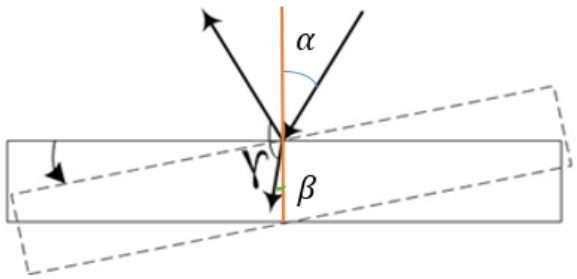

Рис. 4.2.15

Так как стеклянная поверхность повернулась, то на такой же угол повернулся перпендикуляр, опущенный на нее, следовательно, угол падения изменился на такое же значение и равен $\alpha_1 = 45° 15′$.

Найдем угол γ:

$$\gamma = 2\pi - \alpha - \beta,$$

где β — это угол преломления.

$$\frac{\sin\alpha}{\sin\beta} = \frac{n_{\text{с}}}{n_{\text{в}}} \Rightarrow \sin\beta = \frac{n_{\text{в}}}{n_{\text{с}}}\sin\alpha.$$

Такие же рассуждения проводим после поворота пластины:

$$\gamma_1 = 2\pi - \alpha_1 - \beta_1,$$

$$\frac{\sin\beta_1}{\sin\beta_1} = \frac{n_{\text{с}}}{n_{\text{в}}} \Rightarrow \sin\beta_1 = \frac{n_{\text{в}}}{n_{\text{с}}}\sin\beta_1.$$

Найдем изменение угла γ:

$$\gamma - \gamma_1 = \alpha_1 + \beta_1 - \alpha - \beta.$$

Подставим численные значения:

$$\gamma - \gamma_1 = 45{,}25 + 39{,}34 - 30{,}25 - 26{,}726 = 27{,}617 \text{ или } 27^\circ\ 37'.$$

Ответ: $27^\circ\ 37'$.

Критерии оценивания

Таблица 4.2.12

Критерий	**Баллы**	**Подзадача**
1	5	Правильно найдены углы α и β
2	5	Правильно найдены углы α_1 и β_1
3	5	Правильно найдено выражение для $\gamma - \gamma_1$ и выполнены расчеты

Задача 4.2.3.4. Рычажные весы (20 баллов)

Темы: второй закон Ньютона, момент сил, сила Архимеда.

Условие

Коллеги из соседнего умного города подарили нам весы собственной разработки, но совсем не объяснили, как они работают, а значит, надо разобраться. Сделали все по инструкции. Однородный медный куб со стороной 1 см подвесили на неравноплечих весах и погрузили куб в воду наполовину. В каком отношении должны делиться плечи весов, чтобы весы находились в равновесии? Также в опоре весов установлен динамометр. Что он показывает, когда весы находятся в равновесии? Примите плотность воды за 1 г/см^3, плотность меди — 8,9 г/см^3, ускорение свободного падения — 10 м/с^2 и массу коромысла — 2 г.

Решение

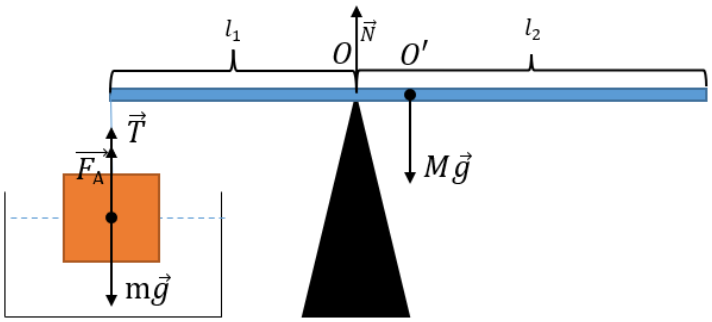

Рис. 4.2.16

Центр масс коромысла точка O' находится на расстоянии $\dfrac{l_1+l_2}{2}$ от краев и на $\dfrac{l_2-l_1}{2}$ от точки подвеса куба.

Запишем второй закон Ньютона для куба в проекции на ось:

$$T + F_A - mg = 0 \Rightarrow T = mg - F_A, \tag{4.2.34}$$

где

$$F_A = \rho_{\text{ж}} g \frac{V}{2} = \rho_{\text{ж}} g \frac{a^3}{2}. \tag{4.2.35}$$

Запишем уравнение моментов для точки O:

$$Tl_1 = Mg\frac{l_2 - l_1}{2}. \tag{4.2.36}$$

Подставим в (4.2.36) (4.2.34) и (4.2.35):

$$\left(\rho_{\text{к}} g a^3 - \rho_{\text{ж}} g \frac{a^3}{2}\right) l_1 = Mg\frac{l_2 - l_1}{2}.$$

Упростим:

$$\frac{l_2}{l_1} = \frac{a^3(2\rho_{\text{к}} - \rho_{\text{ж}})}{M} + 1. \tag{4.2.37}$$

Подставим данные и рассчитаем:

$$\frac{l_2}{l_1} = \frac{10^{-6}(2 \cdot 8\,900 - 1\,000)}{0{,}042} + 1 = 1{,}4.$$

Запишем второй закон Ньютона для коромысла в проекции на ось:

$$N - T - Mg = 0;$$

$$N = T + Mg = (\rho_{\text{к}} - \frac{\rho_{\text{ж}}}{2}) g a^3 + Mg. \tag{4.2.38}$$

Подставим данные и рассчитаем:

$$N = 10^{-5}(8\,900 - 500) + 0{,}042 \cdot 10 = 0{,}504 \text{ Н}.$$

Ответ: 0,504 Н.

Критерии оценивания

Таблица 4.2.13

Критерий	Баллы	Подзадача
1	4	Нарисован рисунок к задаче и расставлены силы
2	3	Записан второй закон Ньютона для куба
3	3	Записано уравнение моментов
4	3	Получено выражение для нахождения отношения плеч
5	2	Найдено численное значение отношения плеч
6	2	Записан второй закон Ньютона для коромысла
7	3	Получена и посчитана сила реакции

Задача 4.2.3.5. Воздушный пузырь (25 баллов)

Темы: Уравнение Менделеева – Клайперона, второй закон Ньютона.

Условие

На побережье умного города поселился кит. Кит очень любит пускать пузыри. Мы заметили, что пузырь воздуха меняет размер в зависимости от глубины. Так, кит плывущий на глубине h, выпускает пузырь воздуха. Пузырь поднимается к поверхности океана. Считая, что температура в течении всего подъема не меняется, определите, какой путь пузырь прошел, если его объем увеличился в η раз. Оцените ускорение, которое приобретет пузырь в конце подъема, если пренебрегать сопротивлением воды в расчетах. Возможно, информация о том, что киты редко погружаются глубже 100 м, поможет сделать правильный вывод. Молярная масса воздуха μ и атмосферное давление P_0.

Решение

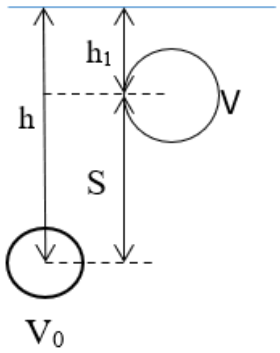

Рис. 4.2.17

Уравнение Менделеева – Клайперона:

$$PV = \frac{m}{\mu}RT;$$

$$V = \frac{m}{\mu P}RT;$$

$$S = h - h_1;$$

$$P = P_0 + \rho gh \text{ и } P_1 = P_0 + \rho gh_1;$$

$$\frac{V_1}{V} = \eta = \frac{P_1}{P} = \frac{P_0 + \rho gh}{P_0 + \rho gh_1};$$

$$h_1 = \frac{P_0 + \rho gh - \eta P_0}{\rho g \eta};$$

$$S = \frac{\eta - 1}{\eta}(h + \frac{P_0}{\rho g});$$

$$\frac{V\mu P_1}{RT} = m;$$

$$F_A - mg = ma \Rightarrow a = \frac{\rho gV}{m} - g;$$

$$a = g\left(\frac{\rho TR}{\mu(P_0 + \rho gh_1)} - 1\right).$$

Подставим произвольные численные значения: $h_1 = 10$ м, $T = 300$ К.

$$a = 9{,}81\left(\frac{1\,000 \cdot 300 \cdot 8{,}31}{0{,}029(10^5 + 1\,000 \cdot 9{,}81 \cdot 10)} - 1\right) = 7\,670 \text{ м/с}^2.$$

Вывод: ускорение получилось много больше ускорения свободного падения, следовательно, сопротивлением воды пренебрегать нельзя.

Ответ: $7\,670$ м/с^2.

Критерии оценивания

Таблица 4.2.14

Критерий	**Баллы**	**Подзадача**
1	4	Записано выражение для давления на глубине h и h_1
2	4	Найдено отношение давлений

Критерий	**Баллы**	**Подзадача**
3	8	Найдено выражение для S
4	6	Записан второй закон Ньютона и найдено выражение для a
5	4	Сделан правильный вывод

4.3. Инженерный тур

4.3.1. Общая информация

С увеличением численности населения мегаполисы сталкиваются с проблемами многокилометровых пробок, нехватки парковочных мест и ухудшения экологии из-за транспортной инфраструктуры.

Участникам третьего этапа предлагается разработать микрорайон умного города, оснащенный технологиями, которые помогут снизить трафик, оптимизировать использование парковок и повысить экологическую устойчивость. Решение должно включать:

- умные светофоры,
- фонари,
- парковки,
- шлагбаумы,

которые интегрируются в единую городскую сеть для обмена данными и быстрого реагирования на экстренные ситуации. Команда должна реализовать эти системы, включая локальный сервер для мониторинга и рекомендаций жителям, чтобы сделать инфраструктуру удобной и эффективной.

4.3.2. Легенда задачи

С ростом численности населения становится все более остро проблема многокилометровых пробок, нехватки парковочных мест и траты времени на дорогу у большинства жителей мегаполисов. Следствием также является загрязнение воздуха, а обеспечение транспортной инфраструктуры увеличивает энергопотребление.

Умные сервисы для мониторинга систем в микрорайонах города — это идеальное решение для комфортной среды проживания людей и транспортной инфраструктуры, где технологии работают на их удобную логистику, эргономичность, экологичность и безопасность.

Так, американский город Сан-Диего решил проблему многокилометровых пробок путем создания крупнейшей городской платформы «Интернет вещей». Ее основу составляет система освещения из 3 500 уличных фонарей, оборудованных датчиками.

Каждый такой фонарь теперь не только светит, но и видит, слышит город благодаря встроенным сенсорам — камерам, микрофонам, динамикам и датчикам окружающей среды. На основе этих данных автоматическая система управляет потоком трафика и отслеживает достаточность свободных мест на парковках, регулирует интенсивность освещения улиц и контролирует качество воздуха.

А как внедрить технологии в каждый российский город? Именно такая задача и поставлена перед инженерными командами: создать микрорайон в составе умного города с транспортной системой, включающей в себя умные светофоры и фонари,

парковки и шлагбаумы. Каждый микрорайон является частью общей технологичной системы умного города с единым центром и логистикой, позволяющей получить, обработать и реагировать на экстренные ситуации.

Система микрорайона должна отслеживать и анализировать данные датчиков и реагировать на запросы городского центра по тому или иному обращению, будь то корректировка освещения или изменение графика перекрытия переезда.

Умные светофоры и парковки позволят сократить трафик на дорогах и упростят взаимодействие жильцов с микрорайоном.

Создаваемая система, охватывающая все необходимые системы для комфортной жизни, позволит применить ее в любом городе России, которые в перспективе, благодаря новому поколению инженеров, могут стать «умными».

4.3.3. Требования к команде и компетенциям участников

Количество участников в команде: 4–5 человек.

В команде должно быть не менее двух программистов для следующего функционала:

- Программист электронных устройств.
- Программист верхнего уровня (веб-система).
- Программист взаимодействия электронных устройств (тестировщик).

Кроме того, два инженера, которые способны не только собирать конструкции, но в целом координировать проект и продемонстрировать его работоспособность:

- Архитектор систем/проектировщик/монтажник.
- Тим-лидер.

Количество тех или иных ролей в каждой команде может быть свободным и не фиксируется строго за человеком, принявшем на себя определенную роль.

4.3.4. Оборудование и программное обеспечение

1. Микрорайон для каждой команды, монтирующийся в полигон умного города. Полигон включает трассу с черной линией и места для парковки электрокаров, железную дорогу, проходящую через микрорайоны всех команд.
2. Трафик в виде мобильных роботов «Динамика ЙоТик М1».
3. Набор электроники (контроллер «ЙоТик 32», исполнительные устройства, датчики, крепеж, провода, отвертки) для создания инфраструктуры микрорайона для каждой команды.
4. Ноутбуки.
5. Роутер Wi-Fi на 2,4 ГГц с доступным интернетом.
6. ПО Arduino IDE.
7. Паяльные станции.

8. Набор инструментов для монтажа объектов микрорайона (монтаж проводов, сверление и др.).

4.3.5. Описание задачи

Каждой команде требуется собрать и запрограммировать несколько составляющих своего микрорайона умного города с транспортной системой.

Задание 1. Контроль трафика

Требуется собрать и запрограммировать умный светофор, который выполняет контроль трафика, переключая с определенным интервалом разрешающий и запрещающий сигнал светофора. При помощи датчика расстояния он определяет количество электрокаров, которые проехали через него за определенное время.

Необходимо посылать уведомления на локальный сервер: при обнаружении чрезмерного загрязнения воздуха отправлять на сервер состояние светофора, данные о загрязнении воздуха и показателях температуры, влажности воздуха.

Задание 2. Автоматическое освещение

Нужно собрать и запрограммировать умный фонарь, который будет автоматически включать свет при обнаружении низкой освещенности, отправлять на сервер команды информацию об освещенности и состоянии света (включен/выключен).

Задание 3. Умная парковка

Необходимо собрать и запрограммировать умную парковку, которая будет постоянно общаться с сервером команды и отправлять данные:

- о занятости парковки,
- об уровне зарядки электрокара,
- о времени до полной зарядки,

кроме того, посылать уведомления на сервер команды о нештатных ситуациях: возникновении возгорания или утечки жидкого электролита из батареи электромобиля.

Задание 4. Перекрытие железнодорожного переезда

Требуется собрать и запрограммировать умный шлагбаум, способный получать данные с городского сервера о расписании поезда и за некоторое время до его приезда перекрывать переезд, а также получать с соседнего шлагбаума данные о задержке поезда и вводить корректировки для своего расписания.

Задание 5. Локальный сервер

Локальный сервер собирает и отображает информацию о состоянии различных устройств и датчиков, кроме того, дает рекомендации жителям. Он выполняет следующие функции:

- Мониторинг состояния устройств и датчиков. Сервер отображает текущее состояние всех подключенных устройств и собирает данные с датчиков.
- Рекомендации для жителей. На основе температуры и влажности сервер советует, какую одежду стоит надеть. Учитывая уровень загрязнения воздуха, он предлагает воздержаться выходить из дома или отменить поездку в загрязненный район.
- Жители могут отслеживать время зарядки своего электрокара и заранее бронировать парковочное место.
- Шлагбаум, основываясь на расписании и задержке прибытия поезда на соседний переезд, производит корректировки для точного перекрытия переезда без задержек.
- При обнаружении нештатных ситуаций, например, возгорании парковочного места, камера, которая расположена на улице с происшествием, производит фотофиксацию и выводит ее на сервер.

Задание 6. Городской сервер

Необходимо подключить локальный сервер к городскому для обеспечения возможностей:

- изменять уровень освещения в районе;
- увеличивать интервал, разрешающий движение с одной из улиц на светофоре;
- выводить фотографии на сервер для фиксации нештатных ситуаций;
- получать данные с соседнего шлагбаума о времени, на которое опоздал поезд, с целью корректировки времени перекрытия железнодорожного переезда;
- бронировать парковочное место, отслеживать занятость парковок и уровень заряда электрокара, который стоит на парковке.

Локальный сервер должен посылать усредненные показатели по району и отправлять рекомендации о том:

- какую одежду лучше надеть, основываясь на значениях температуры и влажности;
- следует ли воздержаться от выхода на улицу вследствие загрязнения воздуха;
- стоит ли поехать в район с повышенным уровнем загазованности.

4.3.6. Система оценивания

Задания оцениваются жюри в соответствии с описанием критерия. Количество баллов за выполненное задание прописано и не делится на два в случае, если команда выполнила задание всего на половину.

Таблица 4.3.1. Критерии оценивания

	Баллы	Критерии
Максимальное количество баллов: 161 Примечание: задания делятся на два типа: • демонстрационные, • сценарные. Выполненные демонстрационные задания показывают в начале защиты: это правильно настроенные процессы отслеживания датчиков, управление исполнительными устройствами. Баллы за сценарные задания начисляются за правильное взаимодействие умных устройств микрорайона с локальным сервером, локальный сервер, в свою очередь, должен взаимодействовать и подчиняться городскому серверу.		
Демонстрационные задания		
Мониторинг системы	0–20	1 балл за каждое устройство, которое корректно изменяет свое состояние с определенным интервалом (определяется участником), состояние устройства дублируется в монитор порта. 1 балл за каждое устройство, которое корректно выводит измеряемые показатели в монитор порта с единицами измерения. Если устройство не имеет единиц измерения (например, датчик протечки), то в монитор порта выводится его состояние: есть протечка/нет протечки. *Умный фонарь:* • Включение/выключение белых светодиодов на каждом модуле MGL-RGB3. (максимум 4 балла) • Вывод информации об уровне освещенности с каждого датчика MGS-L75. (максимум 4 балла) *Умная парковка:* • Вывод расстояния с датчика расстояния MGS-D20. • Вывод наличия/отсутствия протечки с датчика протечки MGS-WT1. • Вывод уровня инфракрасного и видимого излучения в монитор порта с датчика пламени. *Умная камера:* вывод видеопотока с камеры на веб-сайт. *Умный железнодорожный переезд:* • Вывод данных с датчика расстояния MGS-D20. • Включение/выключение красного сигнала светофора на модуле светодиодов MGL-RGB3. • Вращение сервопривода MG90S из положения «перекрыто» в положение «открыто». *Умный светофор:* • Вывод информации с каждого датчика расстояния MGS-D20. (максимум 2 балла) • Переключение между 3 сигналами светофора (красный — верхний, желтый — средний, зеленый — нижний) на адресной светодиодной ленте WS2812B. • Вывод информации с датчика концентрации CO и летучих органических соединений MGS-CO30. • Вывод информации о температуре, влажности и атмосферном давлении с датчика MGS-THP80.
Максимально баллов: 20		
Сценарные задания		
Сценарий № 1 **Контроль трафика**		

	Баллы	**Критерии**
Цветовые сигналы	0–6	Для каждой стороны светофора: • 0 — адресная светодиодная лента не имеет возможности отображать цветовые сигналы или не запрограммирована. • 1 — адресная светодиодная лента изменяет цветовой сигнал, согласно последовательности: красный — 15 с, желтый — 2 с, зеленый — 15 с. • 2 — каждому световому сигналу соответствует свой светодиод на ленте, при этом два и более светодиода гореть не должны. Соответствие сигналов с расположением светодиода: • красный — нижний/левый светодиод; • желтый — средний светодиод; • зеленый — верхний/левый светодиод.
Определение трафика	0–4	Для каждого датчика расстояния: • 0 — датчики расстояния, расположенные на улицах, не имеют возможности определять трафик, не запрограммированы. • 2 — датчики расстояния определяют индивидуально для каждой улицы количество электрокаров за 1 мин.
Загрязнение воздуха	0–2	Пункты не суммируются: • 0 — датчик концентрации CO_2 и летучих органических веществ не имеет возможности определять уровень загрязнения воздуха или не запрограммирован. • 1 — датчик определяет повышение концентрации CO_2 и летучих органических веществ. • 2 — датчик определяет повышение концентрации CO_2 и летучих органических веществ и присваивает ему уровень загрязнения воздуха по шкале 0–3, где 0 — нет загрязнения, 3 — предельный уровень загрязнения.
Определение параметров окружающей среды	0–2	• 0 — датчик ТВА не имеет возможности определить температуру, влажность воздуха и атмосферное давление. • 2 — датчик ТВА определяет температуру, влажность воздуха и атмосферное давление.
Изменение периодов разрешающего сигнала для определенной улицы	0–4	• 0 — увеличение времени разрешающего сигнала светофора со стороны улицы с большим трафиком не производится. • 2 — производится увеличение времени разрешающего сигнала светофора со стороны улицы с большим трафиком. • 2 — при изменении времени разрешающего сигнала светофора на определенной улице на противоположной улице разрешающий сигнал светофора становится меньше на то же время (не допускается включение разрешающего сигнала светофора со стороны двух улиц).
Отправка данных на локальный сервер	0–8	• 0 — данные не отправляются на локальный сервер. • 2 — на локальный сервер отправляются сведения о состоянии уровня загрязнения воздуха. • 2 — отправляются данные о состоянии разрешающего сигнала светофора для каждой улицы. • 2 — на локальный сервер отправляются данные о температуре, влажности воздуха и атмосферном давлении. • 2 — на локальный сервер отправляются данные о количестве электрокаров за 1 мин на каждой улице.
Максимально баллов: 26		

	Баллы	Критерии
Сценарий № 2 **Автоматическое освещение**		
Определение уровня освещенности	0–5	Для каждого датчика освещенности: • 0 — датчик освещенности не имеет возможности определять уровень освещенности на улице. • 1 — датчик освещенности определяет низкую освещенность на улице.
Автоматическое включение света	0–5	Для каждого светодиодного модуля: • 0 — светодиодный модуль не включает белую подсветку при обнаружении низкой освещенности. • 1 — светодиодный модуль автоматически включает свет при обнаружении низкой освещенности.
Отправка данных на локальный сервер	0–6	• 0 — данные не отправляются на локальный сервер. • 3 — отправляются данные об уровне освещенности на улице с каждого фонаря. • 3 — отправляются данные о состоянии каждого фонаря (вкл/выкл).
Максимально баллов: 16		
Сценарий № 3 **Умная парковка**		
Определение занятости парковки	0–1	• 0 — датчик расстояния не имеет возможности определить занятость парковки. • 1 — датчик расстояния определяет занятость парковки.
Определение протечки	0–1	• 0 — датчик протечки не имеет возможности определить протечку. • 1 — датчик протечки определяет протечку.
Определение возгорания	0–1	• 0 — датчик пламени не имеет возможности определить возгорание. • 1 — датчик пламени определяет возгорание.
Определение заряда электрокара	0–3	*Известно, что электрокар для зарядки заезжает на парковку с зарядом батареи равным 0%, зарядка до 100% занимает ровно 1 мин (60 с):* • 0 — умная парковка не имеет возможности определить уровень заряда электрокара по шкале 0–100%. • 3 — умная парковка фиксирует, в какой момент электрокар заехал на парковку, принимает это за старт зарядки и рассчитывает уровень заряда, основываясь на том, что зарядка до 100% занимает ровно 1 мин.
Отправка данных на локальный сервер	0–8	• 0 — умная парковка не отправляет данные на локальный сервер. • 2 — умная парковка отправляет на локальный сервер состояние парковки (занята/не занята). • 2 — при нахождении электрокара на парковке на сервер передается уровень заряда электрокара в диапазоне 0–100%. • 2 — отправляется состояние протечки (есть/нет). • 2 — отправляется состояние возгорания (есть/нет).
Максимально баллов: 14		
Сценарий № 4 **Перекрытие железнодорожного переезда**		

	Баллы	**Критерии**
Определение поезда	0–1	• 0 — датчик расстояния не имеет возможности определить появление поезда. • 1 — датчик расстояния определяет появление поезда.
Включение запрещающего сигнала переезда	0–1	• 0 — модуль светодиодов не имеет возможности включить запрещающий сигнал (красный свет). • 1 — модуль светодиодов включает запрещающий сигнал (красный свет).
Изменение состояния шлагбаума	0–2	• 0 — сервопривод не имеет возможности опустить/поднять шлагбаум. • 1 — сервопривод имеет возможность опустить/поднять шлагбаум. • 1 — положение шлагбаума должно быть либо четко горизонтальным, либо вертикальным, допускается погрешность сервопривода.
Обмен данными с локальным серверов	0–5	• 0 — умный шлагбаум не имеет возможности отправлять и получать данные с локального сервера. • 1 — с локального сервера принимается информация о времени прибытия поезда. • 1 — с локального сервера принимается индекс отклонения поезда от расписания. • 3 — при обнаружении отклонения поезда на сервер отправляет индекс отклонения в секундах, при задержке индекс положительный, при раннем приезде индекс отрицательный.
Перекрытие ж/д переезда	0–6	*Пункты не суммируются:* • 0 — умный шлагбаум не имеет возможности перекрыть железнодорожный переезд. • 3 — перекрытие железнодорожного переезда, используя только расписание. • 6 — перекрытие железнодорожного переезда, используя расписание с учетом индекса отклонения с предыдущего переезда.
	0–1	*За 15 с до прибытия поезда:* • 0 — модуль светодиодов не имеет возможности включить запрещающий сигнал (красный свет). • 1 — модуль светодиодов включает запрещающий сигнал (красный свет).
	0–2	*За 10 с до прибытия поезда:* • 0 — сервопривод не имеет возможности опустить/поднять шлагбаум. • 1 — сервопривод имеет возможность опустить/поднять шлагбаум. • 1 — положение шлагбаума должно быть либо четко горизонтальным, либо вертикальным, допускается погрешность сервопривода.
Открытие ж/д переезда	0–2	• 0 — умный шлагбаум не открывает переезд. • 1 — после проезда поезда через 3 с железнодорожный переезд открывается. • 1 — после проезда поезда через 3 с выключается запрещающий сигнал светофора.
Максимально баллов: 20		
Сценарий 5 **Локальный сервер**		

	Баллы	**Критерии**
Вывод информации с умных устройств в веб-интерфейс	0–10	0 — вывод информации с датчиков умных устройств не производится. *1 балл за каждый выведенный показатель:* • *Умный фонарь:* 4 показателя освещенности. • *Умный светофор:* ◇ температура, влажность воздуха и атмосферное давление. ◇ степень загрязнения воздуха. ◇ количество электрокаров в минуту для каждой улицы. • Умная парковка: ◇ наличие возгорания. ◇ наличие протечки. ◇ занятость парковки (свободно/занято/забронировано). ◇ уровень заряда электрокара. • Умный шлагбаум: ◇ время до закрытия шлагбаума. ◇ индекс отклонения от расписания.
Отправка команд на умные устройства	0–10	0 — локальный сервер не имеет возможности отправлять команды управления на умные устройства. *2 балла за каждый пункт:* • *Умный светофор:* возможность изменить длительность разрешающего сигнала светофора для определенной улицы. • *Умный фонарь:* ◇ изменение режима работы для каждого фонаря: ручной/автоматический; ◇ изменение состояния каждого фонаря вкл/выкл (не должно быть доступно в автоматическом режиме). • *Умный шлагбаум:* ◇ изменение режима работы для шлагбаума: ручной/автоматический; ◇ изменение состояния шлагбаума открыто/закрыто (не должно быть доступно в автоматическом режиме). *Примечание:* управление устройствами с локального сервера становится активным, если разрешающий флаг для управления устройствами, полученный с городского сервера, активен.
Рекомендации для жителей	0–7	Рекомендации по одежде: • 0 — рекомендации отсутствуют, основываясь на температуре и влажности. • 2 — при рекомендации учитывается влажность воздуха. • 2 — при рекомендации учитывается температура воздуха. • 3 — существует 3 вида рекомендаций, содержание рекомендации определяется командой, по содержанию можно четко понять, какие показатели завышены.
	0–5	Рекомендации о воздержании выхода на улицу, приезда в город: • 0 — рекомендации отсутствуют, основываясь на уровне загрязнения воздуха. • 5 — на основе уровня загрязненности даются рекомендации. *Примечание:* всего существует 3 вида рекомендаций, их содержание определяется командой, по содержанию можно четко понять, насколько высок уровень загрязнения.

	Баллы	Критерии
Фотофиксация нештатных ситуаций	0–5	*Нештатной ситуацией считается возгорание или протечка на умной парковке, превышение уровня загрязнения воздуха до предельного уровня:* • 0 — при обнаружении нештатных ситуаций не производится фотофиксация и отправка фото в телеграм-бот. • 5 — при обнаружении нештатной ситуации производится фотофиксация и отправка фото в телеграм-бот.
Максимально баллов: 37		
Сценарий № 6 Городской сервер		
Обмен данными с городским сервером	0–2	• 0 — локальный сервер не имеет возможности получать и отправлять данные на городской сервер. • 2 — локальный сервер отправляет информацию с умных устройств, получает команды для управления, данные с других микрорайонов.
Взаимодействие с умным фонарем	0–6	• 0 — локальный сервер не имеет возможности, основываясь на команды с городского сервера, изменять степень освещения улиц. • 1 — локальный сервер передает данные о состоянии каждого фонаря: вкл/выкл. • 1 — локальный сервер передает на городской сервер усредненные данные об уровне освещенности улиц. • 2 — локальный сервер, основываясь на данных с городского сервера, изменяет степень освещения улиц, передается яркость в процентах 0–100%. • 2 — локальный сервер, основываясь на данных с городского сервера, имеет возможность в ручном режиме включить или выключить определенный фонарь.
Взаимодействие с умным светофором	0–4	• 0 — локальный сервер не имеет возможности управлять интервалом разрешающего сигнала светофора, основываясь на данных с городского сервера. • 1 — локальный сервер передает на городской сервер усредненные данные о загруженности каждой улицы. • 1 — локальный сервер отправляет данные о сингале светофоров. • 2 — локальный сервер управляет состоянием сигнала светофора, основываясь на данных с городского сервера.
Фотофиксация нештатных ситуаций	0–5	• 0 — локальный сервер не имеет возможности отправить фотографию на городской сервер при возникновении нештатной ситуации. • 5 — локальный сервер отправляет фотографию в беседу города при возникновении нештатной ситуации.
Взаимодействие с умным шлагбаумом	0–3	• 0 — локальный сервер не имеет возможности передавать и получать время отклонение поезда от расписания на городской сервер. • 1 — локальный сервер передает индекс отклонения от расписания. • 1 — локальный сервер передает данные о состоянии переезда: открыто/закрыто. • 1 — локальный сервер управляет состоянием железнодорожного переезда, основываясь на данных с городского сервера.

	Баллы	Критерии
Взаимодействие с умной парковкой	0–5	• 0 — локальный сервер не может передавать и получать данные с городского сервера для работы с умной парковкой. • 1 — локальный сервер передает на городской сервер состояние парковки (занята/не занята). • 1 — локальный сервер передает на городской сервер уровень заряда электрокара, если он стоит на парковке. • 3 — при получении команды с городского сервера о бронировании места, парковка ставит статус забронировано, пока на нее не приедет электрокар, который занял место.
Рекомендации для жителей	0–3	• 0 — локальный сервер не имеет возможности передавать на городской сервер информацию с датчиков или рекомендации по одежде, воздержании прибытия в микрорайон. • 1 — локальный сервер передает данные о температуре, влажности воздуха и атмосферного давления. • 1 — локальный сервер передает на городской сервер рекомендации по одежде. • 1 — локальный сервер передает на городской сервер информацию о рекомендации о воздержании прибытия в микрорайон, основываясь на уровне загрязнения воздуха.
Максимально баллов: 28		

Дополнительные баллы*

1. При использовании проводов для подключения оборудования, которые были спаяны самостоятельно, присваивается 5 баллов. Количество баллов зависит от чистоты пайки (от 1 до 5 баллов).
2. Автоматическая отправка в гугл-таблицу лог процессов.

4.3.7. Решение задачи

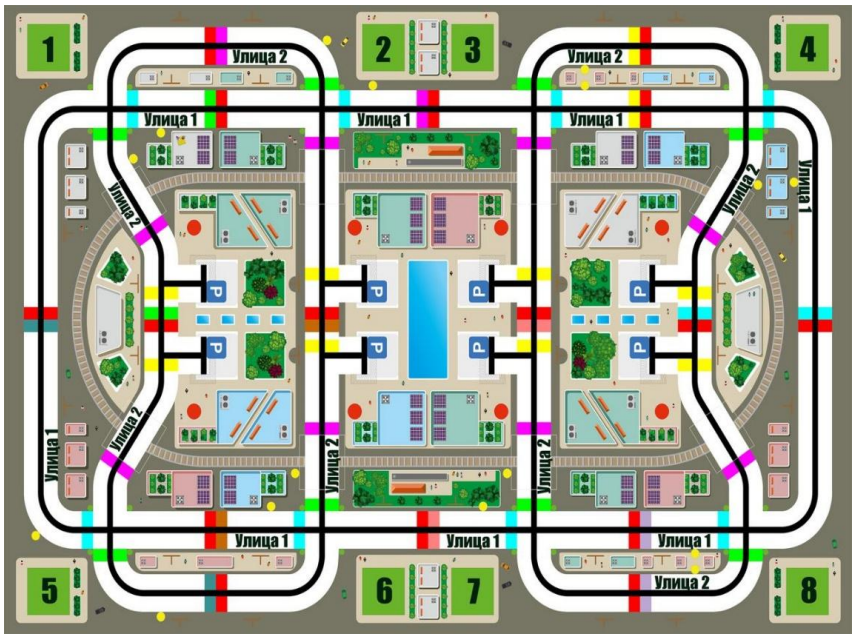

Рис. 4.3.1. Изображение полигона

Каждой команде была выдана часть микрорайона, которая составляла 1/8 от общего полигона.

Общий алгоритм решения

Так как задача комплексная, рекомендуется придерживаться следующего алгоритма решения поставленных задач (см. рис. 4.3.2).

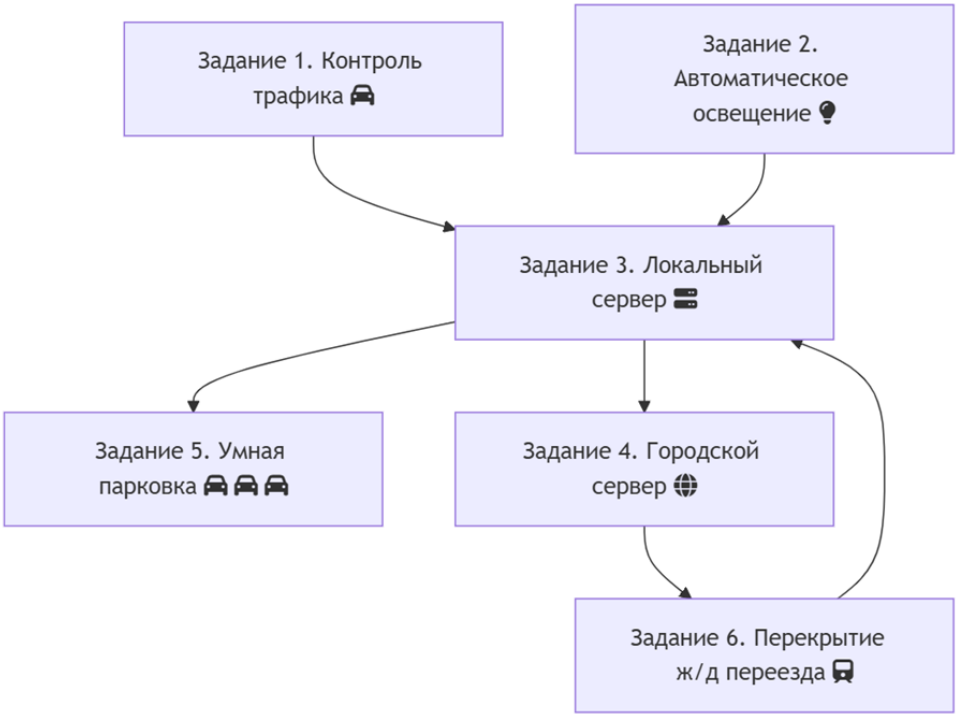

Рис. 4.3.2. Алгоритм решения командной задачи

Сборка микрорайона

Сначала нужно проанализировать состав и расположение умных устройств на выданной части полигона. Все шлейфы и кабели питания паяются участниками самостоятельно, поэтому необходимо придумать длину кабелей и оптимизировать их количество.

Пайка должна быть аккуратной, потому что при равенстве баллов данный критерий тоже оценивается. Кроме того, аккуратность коммуникаций поможет самим участникам не запутаться в том, что куда подключается. После сборки и пайки все устройства располагаются непосредственно на полигоне для дальнейшего тестирования и представления экспертам

Программирование умных устройств

Параллельно сборке идет процесс программирования отдельных устройств. Необходимо определить состав устройства, прописать его логику работы и способ взаимодействия с локальным сервером.

Локальный веб-сервер

Чтобы обеспечить эффективное взаимодействие умных устройств с локальным сервером, изначально важно четко сформулировать принципы коммуникации и перечень передаваемых параметров. Необходимо также определиться с рабочим режимом каждого устройства — управляется ли оно исключительно локальным сервером, либо функционирует согласно заранее заданному сценарию, который активируется с локального сервера.

После того как внутренняя инфраструктура связи между устройствами микрорайона успешно развернута и функционирует стабильно, следующим этапом производится подключение к городскому серверу посредством предварительно заданной конфигурации сети и протокола UDP.

Для этого требуется разработать специализированный обработчик входящих пакетов от городского сервера, обеспечивающий корректный прием и обработку поступающих данных. Дополнительно необходимо настроить регулярную отправку показателей состояния микрорайона на городской сервер.

Городской сервер

После успешного установления обмена данными с городским сервером управление устройствами микрорайона осуществляется исходя из специального флага («тумблера»), определяющего режим управления:

- когда флаг отключен, устройства функционируют автономно, либо регулируются непосредственно с локального сервера («рычажки»);
- когда флаг активирован, контроль над устройствами переходит к городскому серверу, позволяя осуществлять удаленное управление с использованием соответствующих настроек («рычажков») на стороне города.

Повышение точности прогнозирования расписания поездов требует регулярного получения с городского сервера актуальной информации о текущих отклонениях от графика, происходящих на смежных участках путей. Здесь особую роль играет протокол UDP, благодаря которому возможна одновременная передача данных сразу с нескольких контроллеров, расположенных в различных районах (всего восемь), объединенных единым сервером.

Для обработки этих сведений целесообразно создать структуру данных, представленную массивом из восьми элементов, каждый из которых соответствует отдельному району. Основываясь на полученных значениях отклонений, алгоритм сможет своевременно корректировать график движения собственного района, улучшая общую эффективность транспортного процесса.

Программный код эталонных решений этой задаче можно найти по ссылкам ниже:

- Репозиторий от разработчиков задачи: https://gitverse.ru/volkova_k/NTO.
- Работу команды-победителя «Коломята» можно найти по ссылке: https://disk.yandex.ru/d/Fb9cysw_DUZnUA.

Важнейший элемент работы — постоянное самостоятельное тестирование своей

работы. Полезным будет изучение современной методологии разработки программного кода, например, Scrum. При испытаниях с большим объемом работы и ограниченным временем грамотное управление ресурсами и четкая постановка задач может сыграть ключевую роль.

4.3.8. Материалы для подготовки

- Курс «Введение в программирование (C++)»: https://stepik.org/course/363/promo#toc.
- Курс «Программирование на языке C++»: https://stepik.org/course/7/promo.
- Уроки Arduino и робототехники от AlexGyver: https://alexgyver.ru/lessons/.
- Онлайн-симулятор контроллера Йотик 32: https://books.mgbot.ru/books/wokwi.html#1.

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 50,0 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 6400,0 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 161,00 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 ($S_{3\ \text{макс}} = 100$).

Итого максимально возможный индивидуальный балл участника заключительного этапа — 100 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 2 победителя и 5 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	42,82 и выше
Призеры	От 37,20 до 42,02

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».