



НТО

МАТЕРИАЛЫ ЗАДАНИЙ
Всероссийской междисциплинарной олимпиады
школьников 8–11 класса
«Национальная технологическая олимпиада»
по профилю
«Интеллектуальные робототехнические системы»

2024/25 учебный год

ntcontest.ru

УДК 373.5.016:[681.5:004.8]

ББК 74.263.2

И73

Авторы:

М. В. Бабушкин, А. А. Бойцев, Л. В. Борель, Н. В. Ведерников, А. А. Ведяков, И. А. Воронцов, А. А. Гаврилюк, Е. Н. Горечин, О. В. Зубков, М. В. Лукина, А. О. Овчаров

И73 Всероссийская междисциплинарная олимпиада школьников 8–11 класса «Национальная технологическая олимпиада». Учебно-методическое пособие
Том 10 **Интеллектуальные робототехнические системы**
— М.: Ассоциация участников технологических кружков, 2025. — 188 с.

ISBN 978-5-908021-09-8

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

13.00.00 Электро- и теплоэнергетика

15.00.00 Машиностроение

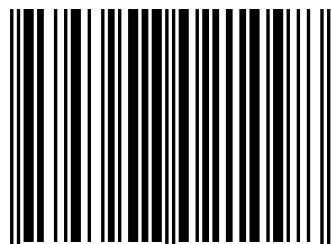
24.00.00 Авиационная и ракетно-космическая техника

27.00.00 Управление в технических системах

ISBN 978-5-908021-09-8

УДК 373.5.016:[681.5:004.8]

ББК 74.263.2



9 785908 021098 >

Оглавление

1 Введение	5
1.1 Национальная технологическая олимпиада	5
1.2 Интеллектуальные робототехнические системы	13
2 Первый отборочный этап	16
2.1 Работа наставника НТО на этапе	16
2.2 Предметный тур. Информатика	17
2.2.1 Первая волна. Задачи 8–11 класса	17
2.2.2 Вторая волна. Задачи 8–11 класса	27
2.2.3 Третья волна. Задачи 8–11 класса	37
2.2.4 Четвертая волна. Задачи 8–11 класса	50
2.3 Предметный тур. Математика	65
2.3.1 Первая волна. Задачи 8–9 класса	65
2.3.2 Первая волна. Задачи 10–11 класса	68
2.3.3 Вторая волна. Задачи 8–9 класса	72
2.3.4 Вторая волна. Задачи 10–11 класса	75
2.3.5 Третья волна. Задачи 8–9 класса	80
2.3.6 Третья волна. Задачи 10–11 класса	85
2.3.7 Четвертая волна. Задачи 8–9 класса	89
2.3.8 Четвертая волна. Задачи 10–11 класса	93
2.4 Инженерный тур	98
3 Второй отборочный этап	122
3.1 Работа наставника НТО на этапе	122
3.2 Инженерный тур	124
3.2.1 Командные задачи	124

4	Заключительный этап	144
4.1	Работа наставника НТО при подготовке к этапу	144
4.2	Предметный тур	146
4.2.1	Информатика. 8–11 классы	146
4.2.2	Математика. 8–9 классы	159
4.2.3	Математика. 10–11 классы	162
4.3	Инженерный тур	167
4.3.1	Общая информация	167
4.3.2	Легенда задачи	167
4.3.3	Требования к команде и компетенциям участников	167
4.3.4	Оборудование и программное обеспечение	168
4.3.5	Описание задачи	169
4.3.6	Система оценивания	179
4.3.7	Решение задачи	182
4.3.8	Материалы для подготовки	184
5	Критерии определения победителей и призеров	186
6	Работа наставника после НТО	188

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. Интеллектуальные робототехнические системы

Интеллектуальные робототехнические системы (ИРС) — это современные технологии, объединяющие в себе множество наук: робототехнику, искусственный интеллект, компьютерное зрение, механику, электронику и многое другое. Они используются для автоматизации производства, эксплуатации техники, управления транспортом, медицинской диагностики, а также для выполнения многих других задач.

Сегодня ИРС представлены различными типами роботов: манипуляторами, мобильными роботами, дронами, автономными транспортными средствами и т. д. Каждый тип имеет особенности и предназначен для выполнения определенных задач.

Одной из главных особенностей интеллектуальных робототехнических систем является возможность их автономной работы. Это означает, что роботы в состоянии самостоятельно определять свое местоположение, принимать решения и выполнять задачи без участия человека — для этого используются различные датчики и алгоритмы обработки информации.

Однако, несмотря на высокую степень автоматизации, ИРС все еще нуждаются в управлении и программировании. Человек может задавать роботу целевые задачи, контролировать его работу, а также управлять его движениями и действиями.

Знакомство с профилем Интеллектуальные робототехнические системы с «Урока НТО», который разработан для проведения в общеобразовательных учреждениях. Материалы для проведения урока находятся на сайте <https://nto-lesson.ru/> и доступны после регистрации на платформе «Талант». Урок создан в виде командной игры, в которой учитель исполняет роль инвестора, планирующего вложить средства в современную робототехническую систему. Команды, основываясь на полученной в ходе урока информации, должны представить описание своего проекта такой системы и объяснить, какими преимуществами она обладает.

Первый отборочный этап (индивидуальный) состоит из двух туров — предметного и инженерного. Предметный тур определяет уровень подготовки школьников по предметам: математика и информатика.

Задачи по математике проверяют знания по алгебре, комбинаторике, геометрии. Решая задачи по программированию, школьники демонстрируют простейшие навыки составления и отладки программ, обрабатывающих массивы данных, и понимание таких тем, как комбинаторика, операции со строками, вычислительная геометрия, теория графов. В целом, на первом этапе проверяются знания, необходимые как для второго, так и для заключительного этапа.

Задачи **второго отборочного этапа** разработаны таким образом, что их сложно решить индивидуально, поэтому участники объединяются в команды. Они требуют погружения в такие робототехнические темы, как:

- планирование маршрутов перемещения;
- расчет допустимых конфигураций манипуляторов;
- расчет конфигурации и положения манипуляторов;
- анализ пространства;

- компьютерное зрение;
- переход из одной системы координат в другую.

Для получения дополнительной информации, необходимой для решения задач второго этапа, командам предлагаются образовательные материалы, опубликованные на странице профиля.

Заключительный этап включает два тура: инженерный и индивидуальный предметный.

Инженерный тур посвящен написанию программы для планирования движения и управления автономным такси (мобильным роботом) в условиях городской среды, где каждая дорога обладает двумя полосами (по одной для направления движения). При этом робот должен двигаться по заданным точкам/меткам в правильном порядке и без нарушения ПДД.

Выполнение командной задачи делится на три основных блока:

1. Базовое управление роботом и навигация: необходимо написать программу управления роботом, слежения за ним по камере и одометрии (робот должен двигаться по координатам с камеры и по собственной одометрии).
2. Движение по дороге без правил: требуется разработать программу планирования маршрута и обработки изображения для выделения проезжей части.
3. Движение по дороге в соответствии с правилами дорожного движения: требуется усовершенствовать решения, сделанные на прошлых этапах, для учета ПДД, где движение должно осуществляться по правой стороне проезжей части, а выезд на разделительную полосу или за пределы проезжей части запрещен. ПДД участникам регламентируются дополнительно.

Для решения поставленной задачи командам предоставляются:

- робототехнические наборы РОББО;
- несколько тестовых полей 2×2 м для непрерывного тестирования алгоритмов навигации, компьютерного зрения и управления;
- основное поле 4×3 м для проведения соревнований, содержащее проезжую часть, дорожную разметку и зоны парковки.

Поле рассчитано на работу 3–5 роботов в один промежуток времени для выполнения тестов алгоритмов навигации, компьютерного зрения и управления, а также для проведения раундов с оцениванием. С этой целью выделяются слоты времени, на которые капитаны должны записываться каждое утро в течение всех дней Олимпиады.

Таким образом, при решении задачи заключительного этапа участники могут опираться на те знания и наработки, которые они приобрели во время второго этапа. Несмотря на то, что задача заключительного этапа заранее неизвестна, ее элементы рассматриваются на предварительных этапах, что значительно упрощает реализацию алгоритма управления робототехническим устройством.

Важной частью заключительного этапа является индивидуальный предметный тур, в ходе которого участники решают задачи по математике и информатике (программированию). Баллы, набранные на предметом туре, влияют на их личные результаты.

Победители и призеры профиля Интеллектуальные робототехнические системы

поступают в ведущие вузы России на специальности, связанные с робототехникой и искусственным интеллектом, принимают участие в научно-технологических проектных школах. Участие школьников в данном профиле позволяет повысить популярность этой тематики и осознанность выбора профессии в области информационных технологий и робототехники. Собственный реальный опыт в этой области дает возможность уже в школьном возрасте понять свое отношение и выбрать целевой профильный вуз, а значит, наметить эффективную образовательную траекторию.

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 $0 \leq t \leq 10^6$.

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++

```
1 #include<bits/stdc++.h>
2 #define int long long
3 using namespace std;
4 signed main(){
5     int t;
6     cin >> t;
7     cout << ((t * (t - 1)) * (t - 2)) << endl;
8 }
```

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 $1 \leq a, b, c, d \leq 10^9$.

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++

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  signed main(){
5      double a, b, c, d;
6      cin >> a >> b >> c >> d;
7      double a0 = min({a, b, c, d});
8      double a1 = min(max(a, b) / 2.0, min(a, b));
9      double a2 = min(max(c, d) / 2.0, min(c, d));
10     double ans = max({a0, a1, a2});
11     if( (int)ans == ans ){
12         int ians = ans;
13         cout << ians << endl;
14         return 0;
15     }
16     cout.precision(1);
17     cout << fixed<< ans << endl;
18 }
```

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 распилит 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 $1 \leq n \leq 10^5$.

Во второй строке заданы $2n$ целых чисел d_i — длины всех кусков, которые получились после «тренировки» внука, где $1 \leq d_i \leq 10^9$. 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++

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  signed main(){
5      int n;
6      cin >> n;
7      vector<pair<int, int> > v(2 * n);
8      for(int i = 0; i < 2 * n; i++){
9          int d;
10         cin >> d;
11         v[i] = {d, i + 1};
12     }
13     sort(v.begin(), v.end());
14     vector<pair<int, int> > ans(n);
15     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 $2 \leq a \leq 10^{18}$, $1 \leq b, c \leq 10^{18}$.

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++
1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  int f1(int M, int a){
5      int res = 0, z = 0;
6      while(1){
7          if(M == 0 && z < a){
8              return res;
9          }
10         res += M;
11         M = M + z;
12         z = M % a;
13         M = M / a;
14     }
15 }

```

```

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 $2 \leq n, m \leq 2 \cdot 10^5$.

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. $1 \leq A, B \leq n$, $1 \leq d \leq 10^9$. 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 , $1 \leq S \leq n$.

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++

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  int n, m;
5  vector<vector<pair<int, int> > > G;
6  vector<int> order, used;
7  void dfs(int a){
8      used[a] = 1;
9      for(auto to : G[a]){
10         if(!used[to.first]){
11             dfs(to.first);
12         }
13     }
14     order.push_back(a);
15 }
16 signed main(){
17     cin >> n >> m;
18     G.resize(n + 1);
19     used.resize(n + 1, 0);
20     for(int i = 0; i < m; i++){
21         int a, b, d;
22         cin >> a >> b >> d;
23         G[a].push_back({b, d});
24     }

```

```

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 $n \times m$.

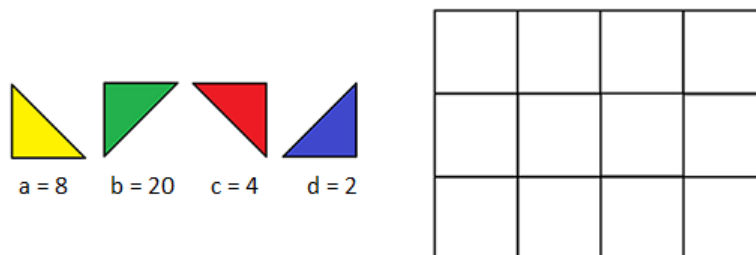


Рис. 2.2.1

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.

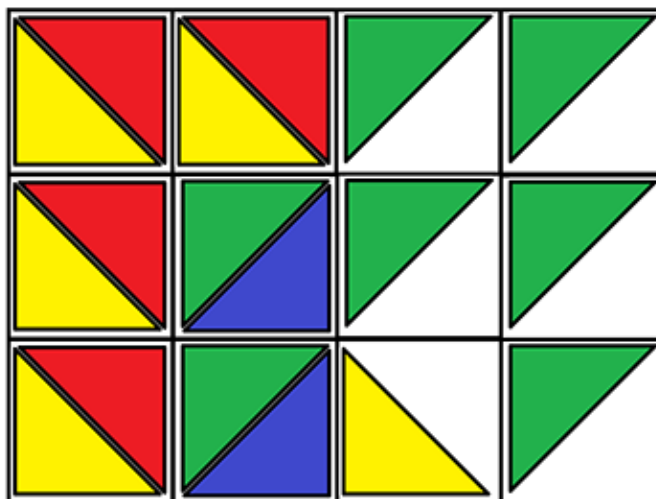


Рис. 2.2.2

Решение

Ниже представлено решение на языке C++.

C++

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  signed main(){
5      int a, b, c, d, n, m;
6      cin >> a >> b >> c >> d >> n >> m;
7      if(a > c){
8          swap(a, c);
9      }
10     if(b > d){
11         swap(b, d);
12     }
13     int f = a + b;
14     int k = n * m;
15     if(k <= f){
16         cout << k * 2;
17         return 0;
18     }
19     k -= f;
20     c -= a;
21     d -= b;
22     cout << f * 2 + min(k, c + d) << endl;
23 }
```

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 $1 \leq n \cdot k \leq 10^{18}$.

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++
1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  signed main(){
5      int n, k;
6      cin >> n >> k;
7      int x = (k * n) / (k * k + 2 * k + 1);
8      cout << x - k << ' ' << x + k << ' ' << x / k << ' ' << x * k << endl;
9  }

```

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, $2 \leq n, m \leq 25$. Во второй строке через пробел находятся координаты (x, y) обязательной для посещения клетки, где $1 \leq x \leq n, 1 \leq y \leq m$. 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.

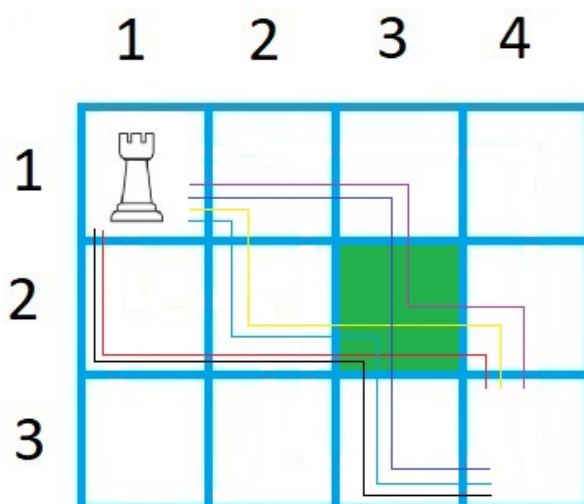


Рис. 2.2.3

Решение

Ниже представлено решение на языке C++.

C++

```

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

```

```

9         bc[i][j] += bc[i - 1][j];
10        if(j - 1 >= 0){
11            bc[i][j] += bc[i - 1][j - 1];
12        }
13    }
14 }
15 int n, m, x, y;
16 cin >> n >> m >> x >> y;
17 int d1 = bc[x - 1 + y - 1][x - 1];
18 int d2 = bc[n - x + m - y][n - x];
19 int ans = d1 * d2;
20 cout << ans << endl;
21 }

```

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 появиться 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++

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  signed main(){
5      string s;
6      cin >> s;
7      string mx = s, mn = s;
8
9      for(int i = 0; i < 5; i++){
10         for(int j = 0; j < 10; j++){
11             mx = max(mx, s);
12             mn = min(s, mn);
13             if(j < 9){
14                 s = s.substr(1) + s[0];
15             }
16         }
17         for(int j = 0; j < 5; j++){
18             swap(s[2 * j], s[2 * j + 1]);
19         }
20     }
21     stringstream ssmn;
22     ssmn << mn;
23     int imn;
24     ssmn >> imn;
25     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 $1, \dots, n$.

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 $1 \leq n \leq 2 \cdot 10^5$.

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++

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  const int LG = 19;
5  int N = (1 << LG);
6  vector<int> tr(2 * N, 0);
7  void upd(int pos, int x){
8      pos += N;
9      tr[pos] = x;
10     pos /= 2;
11     while(pos){
12         tr[pos] = {tr[2 * pos]+ tr[2 * pos + 1]};
13         pos /= 2;
14     }
15 }
16 int get(int l, int r){
17     l += N;
18     r += N;
19     int res = 0;
20     while(l <= r){
21         if(l % 2 == 1){
22             res += tr[l];
23         }
24         if(r % 2 == 0){
25             res += tr[r];
26         }
27         l = (l + 1) / 2;
28         r = (r - 1) / 2;
29     }
30     return res;
31 }
32 signed main(){
33     int n, a;
34     cin >> n;
35     int ans = 0;
36     for(int i = 0; i < n; i++){
37         cin >> a;
38         int sl = get(0, a - 1);
39         int sr = get(a + 1, N - 1);
40         ans += min(sl, sr);
41         upd(a, a);
42     }
43     cout << ans << endl;
44 }
```

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++

```

1 #include<bits/stdc++.h>
2 #define int long long
3 using namespace std;
4 signed main(){
5     int A, C, D;
6     cin >> A >> C >> D;
7     cout.precision(1);
8     cout << fixed << (A + C) / 2.0 << endl;
9 }
```

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

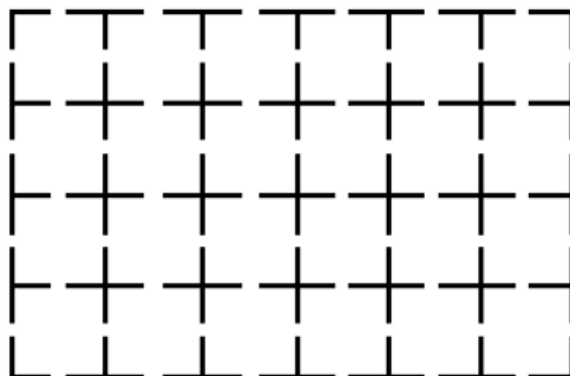


Рис. 2.2.4

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 $1 \leq n, m \leq 10^9$.

Формат выходных данных

Требуется по числам a и b найти размеры исходной решетки n и m . Будем всегда считать, что $n \leq m$, 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 бинарного поиска.

C++

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  signed main(){
5      int a, b;
6      cin >> a >> b;
7      int L = 0, R = a / 4 + 1;
8      while(R - L > 1){
9          int M = (R + L) / 2;
10         int D = a / 2 - M;
11         if(M * D <= b){
12             L = M;
13         }
14         else{
15             R = M;
16         }
17     }
18     cout << L + 1 << ' ' << a / 2 - L + 1 << endl;
19 }
```

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 свечку и предложил ему решить задачу.

«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 $2 \leq n, m \leq 1000$.

Во второй строке находятся два числа x и y через пробел — координаты свечки, где $1 \leq x \leq n - 1, 1 \leq y \leq m - 1$, 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.

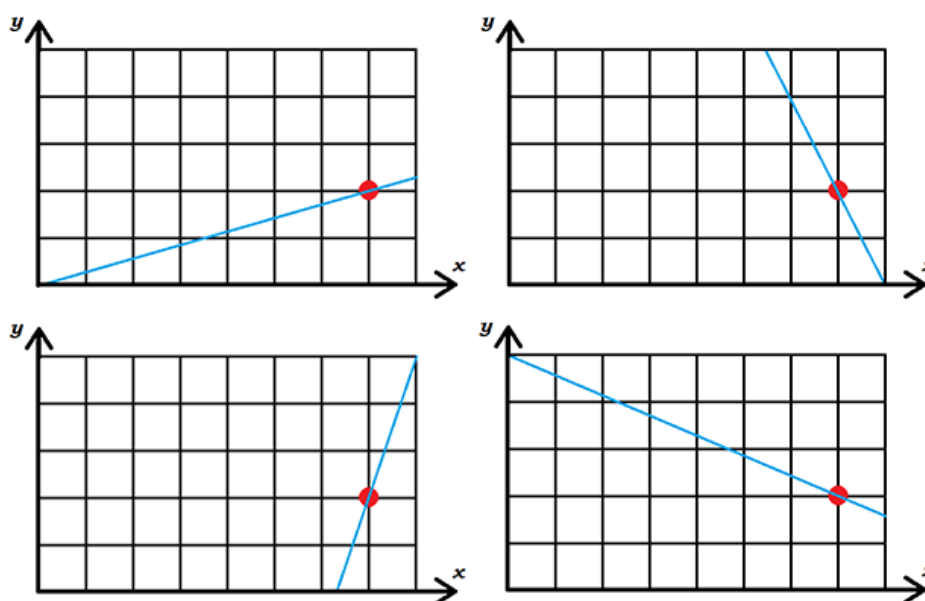


Рис. 2.2.5

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++

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  const int INF = 1e18;
5  double katy(double x, double y, double n){
6      return n * y / x;
7  }
8  double n, m, x, y;
9  double ans = INF;
10 double k1, k2;
11 void upd(){
12     if(k1 < m){
13         double st = k1 * n / 2;
14         ans = min(ans, n * m - 2 * st);
15     }
16     else{
17         double st = k2 * m / 2;
18         ans = min(ans, n * m - 2 * st);
19     }
20 }
21 signed main(){
22     cin >> n >> m >> x >> y;
23     k1 = katy(x, y, n);
24     k2 = katy(y, x, m);
25     upd();
26     k1 = katy(n - x, y, n);
27     k2 = katy(y, n - x, m);
28     upd();
29     k1 = katy(x, m - y, n);
30     k2 = katy(m - y, x, m);
31     upd();
32     k1 = katy(n - x, m - y, n);
33     k2 = katy(m - y, n - x, m);
34     upd();
35     cout.precision(3);
36     cout << fixed << ans << endl;
37 }
```

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 $2 \leq n \leq 2 \cdot 10^5$. Во второй строке через пробел заданы n целых чисел a_i через пробел — длины полосок, где $1 \leq a_i \leq 10^9$.

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.

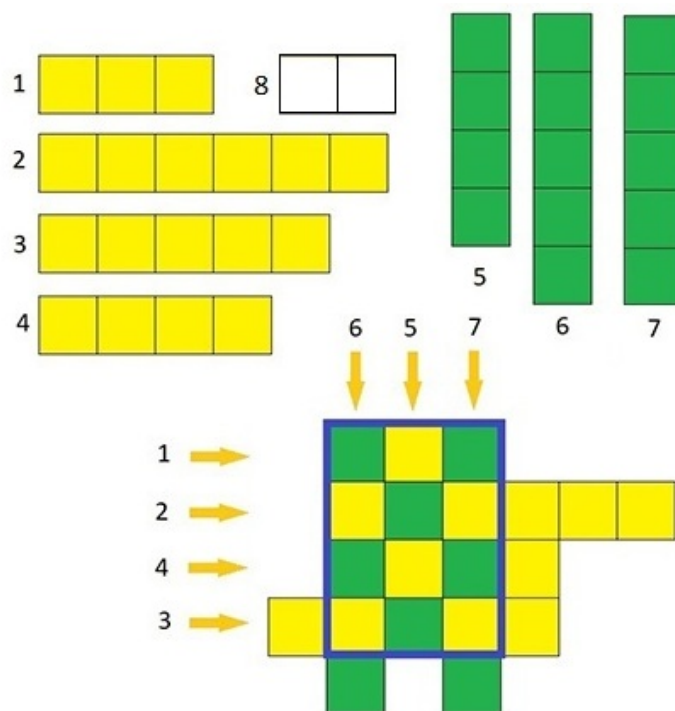


Рис. 2.2.6

Решение

Ниже представлено решение на языке C++.

C++

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  signed main(){
5      int n;
6      cin >> n;
7      deque<int> v(n);
8      for(int i = 0; i < n; i++){
9          cin >> v[i];
10     }
11     sort(v.begin(), v.end());
12     int ans = 0;
13     int cnth = 0, minh;
14     while(1){
15         if(v.size() == 0){
16             break;
17         }
18         cnth++;
19         minh = v.back();
20         v.pop_back();
21         while(v.size() > 0 && v[0] < cnth){
22             v.pop_front();
23         }
24         ans = max(ans, cnth * min(minh, (int)v.size()));
25     }
26     cout << ans << endl;
27 }

```

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 $20 \leq n \leq 100$, 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, $20 \leq n \leq 100$.

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++

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  string frs;
5  int n;
6  vector<string> dict;
7  vector<int> msk(26, 0);
8  int cnt = 0;
9  vector<vector<int>> amsk;
10 vector<string> ans;
11 bool bigok = 0;
12 void p(int pos){
13     if(!bigok){
14         if(cnt == 0){
15             sort(ans.begin(), ans.end());
16             bigok = 1;
17             return;
18         }
19         for(int i = pos; i < n; i++){
20             string ts = dict[i];
21             bool ok = 1;
22             for(int j = 0; j < 26; j++){
23                 if(amsk[i][j] > msk[j]){
24                     ok = 0;
25                 }
26             }
27             if(ok){
28                 ans.push_back(ts);
29                 for(int j = 0; j < 26; j++){
30                     msk[j] -= amsk[i][j];
31                     cnt -= amsk[i][j];
32                 }
33                 p(i + 1);
34                 if(!bigok){
35                     for(int j = 0; j < 26; j++){
36                         msk[j] += amsk[i][j];
37                         cnt += amsk[i][j];
38                     }
39                 }
40                 ans.pop_back();
41             }
42         }
43     }
44 }
45 signed main(){
46     cin >> frs;
47     cin >> n;
48     amsk.resize(n, vector<int>(26, 0));
49
50     string ts;
51     for(int i = 0; i < n; i++){
52         cin >> ts;
53         dict.push_back(ts);
54     }
55     for(int i = 0; i < n; i++){
56         for(auto el : dict[i]){
57             amsk[i][el - 'a']++;
58         }
59     }

```

```
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 $1 \leq a, b, c, d \leq 10^9$.

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

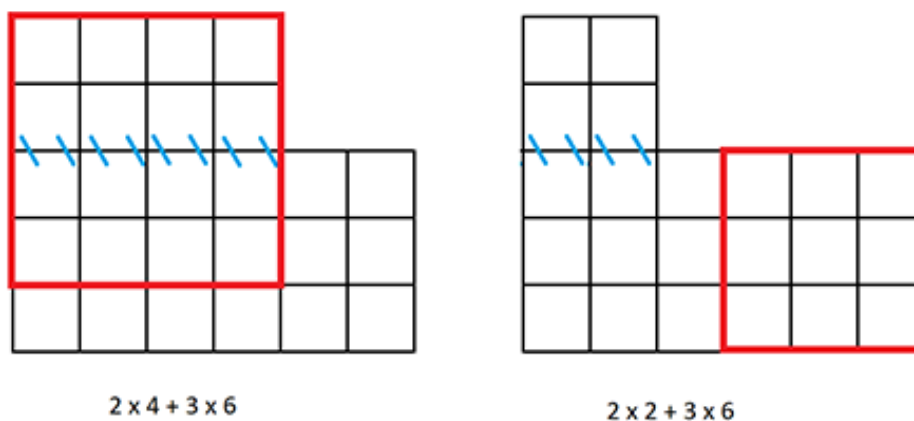
Примечания

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++

```

1 #include<bits/stdc++.h>
2 #define int long long
3 using namespace std;
4 signed main(){
5     int a, b, c, d;
6     cin >> a >> b >> c >> d;
7     int ans = max(min(a, b), min(c, d));
8     int p1 = min(a + c, min(b, d));
9     int p2 = min(a + d, min(b, c));
10    int p3 = min(b + c, min(a, d));
11    int p4 = min(b + d, min(a, c));
12    ans = max({ans, p1, p2, p3, p4});
13    cout << ans << endl;
14 }

```

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 А, С, G, Т.

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 $D_1 = \mathbf{AACGGTAGGT}$, а $D_2 = \text{TCCCGGAACA}$. Выберем $i = 4$ и поменяем местами префиксы длины 4. Получим две новые ДНК, одна из которых будет иметь вид $\mathbf{AACGGGAACA}$, а вторая — TCCCGTAGGT . 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 $10 \leq n \leq 100$, $2 \leq m \leq 100$.

В следующих m строках содержится описание исходной популяции ДНК, каждая задается строкой длины n , состоящей из символов А, С, G и Т.

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
AACCGGTТАА
ACGTACGTAC
AAACCCGGGT
САТТАСТGGA

AAGCGCTТАА
ССАСАСGТGC
ААСТАGGGGT

Стандартный вывод
ААТТССТGAA

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++

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  signed main(){
5      int n, m;
6      cin >> n >> m;
7      vector<string> v1(m);
8      for(int i = 0; i < m; i++){
9          cin >> v1[i];
10     }
11     string d;
12     cin >> d;
13     vector<string> v2(m - 1);
14     for(int i = 0; i < m - 1; i++){
15         cin >> v2[i];
16     }
17     for(int j = 0; j < n; j++){
18         int ss = 0;
19         for(int i = 0; i < m; i++){
20             ss ^= (int)(v1[i][j]);
21         }
22         for(int i = 0; i < m - 1; i++){
23             ss ^= (int)(v2[i][j]);
24         }
25         cout << (char)(ss);
26     }
27     cout << endl;
28 }
```

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 $1 \leq A, B, C, D \leq 1000$, $0 \leq E \leq 1000$.

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++

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  const int INF = 1e18;
5  vector<int> v(4);
6  int e, ans = INF;
7  void p(vector<int> &v1, vector<int> &vr, int tv){
8      if(v1.size() == 2){
9          ans = min(ans, tv + *max_element(v1.begin(), v1.end()));
10         return;
11     }
12     for(int i = 0; i < v1.size() - 1; i++){
13         for(int j = i + 1; j < v1.size(); j++){
14             vector<int> v11;
15             for(int k = 0; k < v1.size(); k++){
16                 if(k != i && k != j){
17                     v11.push_back(v1[k]);
18                 }
19             }
20             vector<int> vr1 = vr;
```

```

21         vrl.push_back(vl[i] + e);
22         vrl.push_back(vl[j] + e);
23         int tmp = max(vl[i], vl[j]);
24         sort(vrl.rbegin(), vrl.rend());
25         vll.push_back(vrl.back() + e);
26         vrl.pop_back();
27         p(vll, vrl, tv + tmp + vll.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 $2 \leq a \leq 10^6$, $0 \leq b \leq 10^6$.

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++

```

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

```

```

7  int binpow (int a, int n) {
8      int res = 1;
9      while (n > 0) {
10         if (n % 2 == 1)
11             (res *= a) %= MOD;
12         (a *= a) %= MOD;
13         n /= 2;
14     }
15     return res;
16 }
17
18 int bc(int n, int k){
19     int res = f[n];
20     int p1 = binpow(f[k], MOD - 2);
21     int p2 = binpow(f[n - k], MOD - 2);
22     (res *= p1) %= MOD;
23     (res *= p2) %= MOD;
24     return res;
25 }
26 signed main(){
27     for(int i = 1; i <= 2e6; i++){
28         f[i] = (f[i - 1] * i) % MOD;
29     }
30     int a, b;
31     int ans = 0;
32     cin >> a >> b;
33     a--;
34     for(int i = 0; i < a + 1; i++){
35         if(2 * i <= b){
36             int d = bc(a, i);
37             if(b - 2 * i <= a - i){
38                 (d *= bc(a - i, b - 2 * i) ) %= MOD;
39                 (ans += d) %= MOD;
40             }
41         }
42     }
43     cout << ans << endl;
44 }

```

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 прогуливающих джентльменов, где $1 \leq d \leq 200$, $2 \leq n \leq 2000$.

Во второй строке находятся n целых чисел v_i через пробел — скорости каждого джентльмена, где $1 \leq v_i \leq 2000$. Гарантируется, что все скорости попарно различны. Скорости даны в порядке возрастания, то есть $v_1 < v_2 < \dots < v_n$.

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

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.

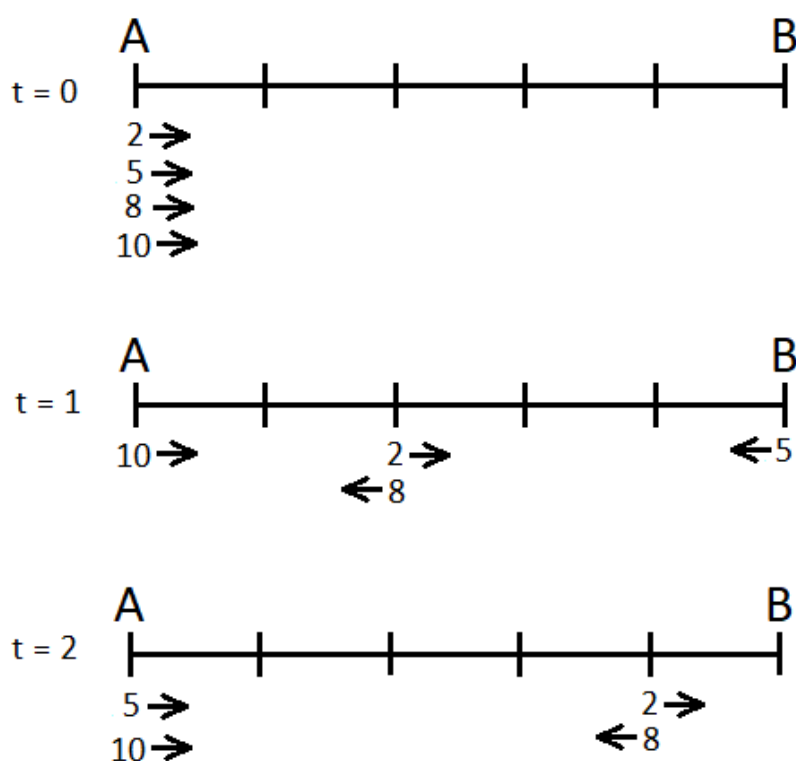


Рис. 2.2.8

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 бинарного поиска с квадратичным нахождением ответа в каждой его итерации.

Решение

Ниже представлено решение на языке C++.

C++

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  const double EPS = 1e-7;
5  double x(double M, int V, int d){
6      double dst = V * M;
7      int cnt = floor((dst + EPS) / d);
8      double pin = dst - cnt * d;
9      if(cnt % 2 == 0){
10         return pin;
11     }
12     else{
13         return d - pin;
14     }
15 }
16 int F(double M, vector<int> &v, int d){
17     int res = 0;
18     for(int i = 0; i < v.size(); i++){
19         double dst = v[i] * M;

```

```
20     int cnt = floor((dst + EPS) / d);
21     res += cnt * i;
22     double tx = x(M, v[i], d);
23     for(int j = 0; j < i; j++){
24         double txj = x(M, v[j], d);
25         if(cnt % 2 == 0){
26             res += txj <= tx + EPS;
27         }
28         else{
29             res += txj >= tx - EPS;
30         }
31     }
32 }
33 return res;
34 }
35 signed main(){
36     int d, n;
37     cin >> d >> n;
38     vector<int> v(n);
39     for(int i = 0; i < n; i++){
40         cin >> v[i];
41     }
42     int k;
43     cin >> k;
44     double L = 0, R = 1;
45     while(F(R, v, d) <= k){
46         R *= 2;
47     }
48     R /= 2;
49     while(R - L > 1e-4){
50         double M = (R + L) / 2.0;
51         if(F(M, v, d) < k){
52             L = M;
53         }
54         else{
55             R = M;
56         }
57     }
58     cout.precision(10);
59     cout << fixed << L << endl;
60 }
```

2.3. Предметный тур. Математика

2.3.1. Первая волна. Задачи 8–9 класса

Задачи первой волны предметного тура по математике за 8–9 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: <https://contests.yandex.ru/contest/63459/enter/>.

Задача 2.3.1.1. (15 баллов)

Тема: арифметика.

Условие

Оля в каждую клетку таблицы 3×3 записала по некоторому числу и с удивлением заметила, что сумма чисел в каждой строке и в каждом столбце таблицы равна 23. Внимательный же одноклассник Витя к ее размышлениям добавил информацию, что сумма чисел в каждом получившемся квадрате 2×2 равна 32. Какое число Оля записала в центральную клетку таблицы?

Решение

Проанализируем исходную таблицу и увидим, что при построении всех возможных квадратов 2×2 :

- числа, стоящие в угловых клетках исходной таблицы, входят по одному разу;
- числа, стоящие во второй строке и во втором столбце — по два раза;
- центральное число — четыре раза.

Тогда если найдем сумму чисел во всех квадратах 2×2 и из нее вычтем сумму чисел всей таблицы, а также сумму чисел, стоящих во втором столбце и второй строке, то найдем центральное число, то есть $32 \cdot 4 - 23 \cdot 3 - 23 \cdot 2 = 13$.

Ответ: 13.

Задача 2.3.1.2. (15 баллов)

Тема: комбинаторика.

Условие

Нечетное восьмизначное число назовем «интересным», если оно состоит из простых цифр и одинаковые цифры не стоят рядом. Сколько существует таких «интересных чисел»?

Решение

Простые цифры — это 2, 3, 5 и 7. Тогда так как «интересное» число должно быть нечетным, то в разряде его единиц может стоять только 3, 5 или 7, то есть три варианта. В разряде десятков также может стоять только три варианта, т. к. одинаковые цифры не могут стоять рядом, и т. д. Таким образом, общее количество «интересных» чисел равно $3^8 = 6561$.

Ответ: 6561.

Задача 2.3.1.3. (20 баллов)

Тема: планиметрия.

Условие

В остроугольном треугольнике ABC провели высоты AA_1 и CC_1 . Точки E и F — середины отрезков AC и A_1C_1 соответственно.

Найдите длину отрезка EF , если известно, что $AC = 30$ и $A_1C_1 = 24$.

Решение

В прямоугольном треугольнике AC_1C с гипотенузой AC : $C_1E = \frac{1}{2}AC = 15$. Аналогично в треугольнике A_1C : $A_1E = \frac{1}{2}AC = 15$.

Таким образом, треугольник A_1C_1E является равнобедренным, и его медиана EF является также и высотой.

Тогда по теореме Пифагора: $EF^2 = A_1E^2 - A_1F^2 = 15^2 - 12^2 = 81$, $EF = 9$.

Ответ: 9.

Задача 2.3.1.4. (25 баллов)

Темы: уравнения, формулы сокращенного умножения.

Условие

Найдите значение выражения $x + y + 3z$, если известно, что числа x , y , z удовлетворяют равенству:

$$5x^2 + 4y^2 + 9z^2 + 12z + 13 = 4xy + 12x.$$

Решение

Преобразуем равенство следующим образом:

$$(x^2 - 4xy + 4y^2) + (4x^2 - 12x + 9) + (9z^2 + 12z + 4) = 0,$$

то есть

$$(x - 2y)^2 + (2x - 3)^2 + (3z + 2)^2 = 0.$$

Данное равенство будет выполняться при условии, что каждое слагаемое равно 0.

Отсюда получаем систему

$$\begin{cases} x - 2y = 0, \\ 2x - 3 = 0, \\ 3z + 2 = 0, \end{cases}$$

единственным решением которой будет

$$x = \frac{3}{2}; \quad y = \frac{3}{4}; \quad z = -\frac{2}{3}.$$

Тогда

$$x + y + 3z = \frac{3}{2} + \frac{3}{4} + 3 \cdot \left(-\frac{2}{3}\right) = \frac{1}{4} = 0,25.$$

Ответ: 0,25.

Задача 2.3.1.5. (25 баллов)

Тема: теория вероятностей.

Условие

Шестизначное число будем называть «замечательным», если оно составлено из цифр 1, 2, 3, 4, 5, 6 (каждая цифра используется в числе по одному разу) и кратно 12. Какая вероятность, что сгенерированное компьютером шестизначное число будет «замечательным»?

Ответ выразите в долях и округлите его до четвертого знака после запятой.

Решение

Для того чтобы «замечательное» число делилось на 12, оно должно делиться на три и на четыре. Заметим, что все рассматриваемые числа кратны трем, так как сумма их цифр равна 21.

Для того же чтобы число было кратно четырем, необходимо, чтобы две его последние цифры образовывали число, кратное четырем. В нашем случае это могут быть варианты: 12, 16, 24, 32, 36, 52, 56, 64, всего их восемь. К каждому из них нужно приписать впереди четырехзначное число, составленное из остальных четырех цифр, таких чисел $4! = 24$. Значит, всего «интересных» чисел $24 \cdot 8 = 192$.

Всего же шестизначных чисел $9 \cdot 10^5 = 900\,000$.

Тогда вероятность, что сгенерированное компьютером число будет являться «замечательным», будет равна $\frac{192}{900\,000} \approx 0,0002$.

Ответ: 0,0002.

2.3.2. Первая волна. Задачи 10–11 класса

Задачи первой волны предметного тура по математике за 10–11 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: <https://contest.yandex.ru/contest/63476/enter/>.

Задача 2.3.2.1. (10 баллов)

Темы: комбинаторика, десятичная запись числа, цифры.

Условие

Двузначное число назовем подходящим, если оно состоит из четных цифр, расположенных по возрастанию (например, 26). Сколько существует таких подходящих чисел?

Решение

Число не может начинаться с нуля, так что можно использовать только цифры 2, 4, 6, 8. Выпишем все подходящие: 24, 26, 28, 46, 48, 68.

Ответ: 6.

Задача 2.3.2.2. (15 баллов)

Темы: текстовые задачи, пропорции, составление уравнений.

Условие

На Марсе планируется разместить колонию в 100 тысяч человек. Разные колонисты будут заняты на разных работах и важно, чтобы каждый вид работы выполняли группы из минимального количества человек. Одна из важных задач — обеспечение колонистов сбалансированным питанием. Нормы здорового рациона были рассчитаны таким образом, чтобы обеспечить для каждого человека 350 г картофеля в день. Полный цикл производства картофеля от посадки и до сбора составляет 60 дней, каждые 60 дней часть собранного урожая используется для выращивания нового. В той технологии, которую используют космонавты, с 1 га можно вырастить 250 т картофеля, а для посадки нужно 5 т/га. Специальная обработка почвы позволяет добиться сохранения постоянного уровня урожайности, причем можно засадить и обрабатывать произвольную долю гектара. Чтобы полностью обслуживать один гектар в условиях теплиц на Марсе, требуется труд четырех человек.

Какое минимальное количество человек должны трудиться на выращивании картофеля?

Решение

Один человек за 60 дней по плану должен съесть $60 \cdot 0,35 = 21$ кг картофеля. Следовательно, 100 тысяч человек по плану за это время съедят 2 100 000 кг.

С одного гектара получаем 250 т, но при этом из них 2 т нужно использовать для посадки. Это значит, что с каждого гектара люди получают в свой рацион 245 т картофеля. Если разделить количество картофеля, которое съест по плану колония за 60 дней, на количество картофеля, которое попадет к ним с 1 га, то получится, что требуется приблизительно 8,571 га. Так как каждый гектар должны обрабатывать четыре человека, то для обработки 8,571 га потребуется труд 34,286 человек. Это значит, что 34 человек недостаточно, требуется запланировать труд 35 человек.

Ответ: 35.

Задача 2.3.2.3. (20 баллов)

Темы: уравнение параболы, координаты вершины параболы.

Условие

Две параболы с различными вершинами пересекаются таким образом, что первая парабола проходит через вершину второй параболы, а вторая — проходит через вершину первой. Уравнение первой параболы имеет вид $y = x^2$, второй $y = a \cdot x^2 + b \cdot x + c$. Найдите, чему равна величина $10 \cdot a + c$.

Решение

Координаты вершины первой параболы имеют вид $(0; 0)$, следовательно, коэффициент $c = 0$. Координаты вершины второй параболы имеют вид

$$x = -\frac{b}{2a};$$

$$y = -\frac{b^2}{4a}.$$

Тогда, подставив их в уравнение первой параболы, получаем:

$$-\frac{b^2}{4a} = \frac{b^2}{2a}.$$

Отсюда $a = -1$.

Ответ: -10 .

Задача 2.3.2.4. (25 баллов)

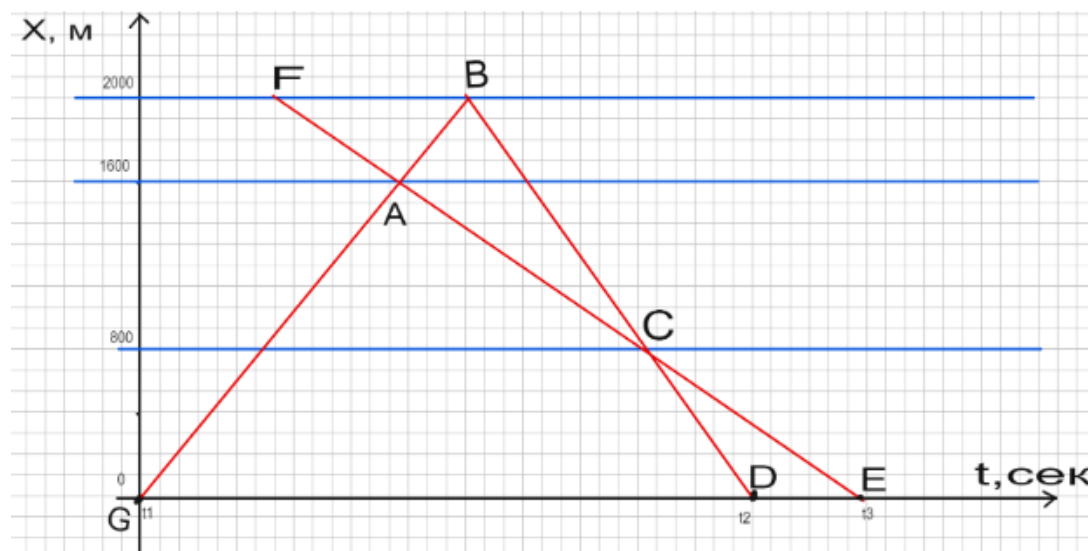
Темы: текстовые задачи и логика, графическое изображение движения, теорема Менелая.

Условие

В 6:00 со дна океана, находящегося на глубине 2 000 м, на поверхность, двигаясь с постоянной скоростью вертикально вверх, начала всплывать подводная лодка. Когда она поднялась до глубины 400 м, капитан заметил, что мимо них вниз плывет глубоководный батискаф. Ему что-то показалось странным. Когда подводная лодка поднялась на поверхность, капитан понял, что на оболочке батискафа были признаки повреждения. Чтобы предотвратить возможную трагедию, в тот же самый момент с подводной лодки вниз спустили спасательный глубоководный аппарат, который спускался с некоторой постоянной скоростью. Когда до дна оставалось 800 м, этот аппарат поравнялся с батискафом. Если бы спасательный аппарат не перехватил батискаф, то спасательный аппарат достиг бы дна к 11:00. Предполагая, что спасательный аппарат все время движения двигался равномерно, определите, в какой момент времени батискаф достиг бы дна, если бы он продолжил движение с той же постоянной скоростью. Ответ введите в виде двух целых чисел, записанных подряд — количество часов и количество минут.

Решение

Самое изящное решение получается графическим методом.



Здесь данные из условия задачи можно обозначить следующим образом:

$$h_1 = 2000 - 400 = 1600, \quad h(2) = 800, \quad H = 2000, \quad t_1 = 6, \quad t_2 = 12.$$

По теореме Менелая получаем:

$$\frac{GA}{AB} \cdot \frac{BC}{CD} \cdot \frac{DE}{GE} = 1.$$

Выразим эти отношения из разных пар подобных треугольников:

$$\begin{aligned}\frac{GA}{AB} &= \frac{h_1}{H - h_1}; \\ \frac{BC}{CD} &= \frac{H - h_2}{h_2}; \\ \frac{DE}{GE} &= \frac{t_3 - t_2}{t_3 - t_1}; \\ \frac{h_1}{H - h_1} \cdot \frac{H - h_2}{h_2} \cdot \frac{t_3 - t_2}{t_3 - t_1} &= 1.\end{aligned}$$

Подставим числа:

$$\begin{aligned}\frac{1\,600}{400} \cdot \frac{1\,200}{800} \cdot \frac{t_3 - 11}{t_3 - 6} &= 1; \\ 6 \cdot (t_3 - 11) &= t_3 - 6; \\ 5 \cdot t_3 &= 60; \\ t_3 &= 12 \text{ ч.}\end{aligned}$$

Ответ: 12.

Задача 2.3.2.5. (30 баллов)

Условие

Инженер-исследователь работает над созданием новой системы гиперпространственной навигации для космических кораблей, которая потребует меньших вычислительных ресурсов. Часть измерений гиперпространства скрыта от нас и устроена не так, как мы привыкли, а именно — являются дискретными (с конечным количеством позиций), позиции в которых следуют друг за другом циклически. Например, если это измерение, в котором 5 позиций, то их можно занумеровать числами от 0 до 4 так, что космический корабль, при прямолинейном движении вдоль этого измерения, будет пролетать позиции $0 - 1 - 2 - 3 - 4 - 0 - \dots$ (конечно, корабль в любой момент может изменить направление своего движения на обратное или начать/продолжить изменять позиции и по другим измерениям гиперпространства).

Оказалось, что в гиперпространстве возможна быстрая (но не мгновенная) телепортация: для такого перемещения требуется особая последовательность перемещений в дискретных подпространствах с остановками лишь в выделенные моменты времени. Ранее для хранения таких сложных гипермаршрутов использовалась технология сплошного хранения всех промежуточных опорных точек пути. Однако из-за воздействия агрессивной космической радиации устройства хранения информации часто выходят из строя, что делает сплошное хранение информации очень дорогим, так как требует многократного резервного копирования,

Инженер корабля предложил хранить не сами последовательности позиций, а формулы для их вычисления (что хранить гораздо дешевле и надежнее). В частности, ему удалось запрограммировать движение в одном из измерений с 13 позициями следующим образом: начальное положение обозначается числом 0 и дальнейшие позиции для остановки вычисляются по формуле: x_{n+1} равно остатку от деления $(x_n^5 + 2)$ на 13.

Переход корабля из одной позиции в соседнюю по прямому или обратному ходу занимает 1 единицу времени, которую называют таймом. Корабль, используя эту формулу, прошел полный цикл по остановкам и вернулся в позицию с номером 0.

Какое минимальное количество таймов могло занимать все его движение между остановками в ходе этого цикла?

Решение

Запишем последовательность позиций, в которых останавливается корабль:

$$0 - 2 - 8 - 10 - 6 - 4 - 12 - 1 - 3 - 11 - 9 - 5 - 7 - 0.$$

Между каждыми двумя позициями корабль может двигаться либо прямым ходом, либо обратным. Нужно выбрать кратчайший из двух.

Тогда общая длительность промежутков будет:

$$T = 2 + 6 + 2 + 4 + 2 + 5 + 2 + 2 + 5 + 2 + 4 + 2 + 6 = 44.$$

Ответ: 44.

2.3.3. Вторая волна. Задачи 8–9 класса

Задачи второй волны предметного тура по математике за 8–9 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: <https://contest.yandex.ru/contest/63460/enter/>.

Задача 2.3.3.1. (15 баллов)

Тема: арифметика.

Условие

Первый поезд мимо телеграфного столба проезжает за 9 с, второй поезд мимо этого же столба — за 14 с, а, двигаясь навстречу мимо друг друга, они проезжают за 10 с (с момента, когда поравнялись их начала, и до момента, когда разминулись концы).

Во сколько раз скорость первого поезда больше скорости второго?

Решение

Пусть x м/с — скорость первого поезда, тогда из условия задачи его длина 9 м. Аналогично, если y м/с — скорость второго поезда, то его длина равна $14y$ м.

Зная, что, двигаясь навстречу мимо друг друга, они проезжают за 10 с, составим уравнение:

$$\frac{9x + 14y}{x + y} = 10.$$

Решив это уравнение, получим $x = 4y$. То есть скорость первого поезда в четыре раза больше скорости второго.

Ответ: 4.

Задача 2.3.3.2. (15 баллов)

Тема: комбинаторика.

Условие

Вася и Петя играют в разведчиков и для этого придумали свой язык шифрования, в котором используются только пять символов. При этом все «слова» в их сообщениях непустые, то есть содержат хотя бы один знак, и длиной не более пяти знаков.

Сколько различных «слов» они имеют в своем арсенале, чтобы передавать друг другу информацию?

Решение

«Слова», которые могут составлять Вася и Петя на своем языке, могут состоять из 1, 2, 3, 4 и 5 символов.

Тогда общее количество слов будет равно $5^1 + 5^2 + 5^3 + 5^4 + 5^5 = 3905$.

Ответ: 3905.

Задача 2.3.3.3. (20 баллов)

Тема: геометрия.

Условие

В треугольнике ABC длина биссектрисы AD равна длине отрезка DC и $AC = 2AB$. Найдите $\angle ABC$.

Решение

В равнобедренном треугольнике ADC из точки D проведем медиану DE на сторону AC , которая также будет являться и высотой.

Тогда $AE = \frac{1}{2}AC = AB$. Треугольники AED и ABD равны по двум сторонам и углу между ними: $AE = AB$, AD — общая сторона и $\angle DAE = \angle DAB$.

Следовательно, $\angle ABC = \angle ABD = \angle AED = 90^\circ$.

Ответ: 90° .

Задача 2.3.3.4. (25 баллов)

Тема: десятичная запись натурального числа.

Условие

В натуральном двузначном числе a цифры поменяли местами и получили двузначное число b . Оказалось, что сумма чисел a и b делится на 5, а их разность — на 27.

Найдите все возможные значения числа a . В ответ запишите сумму всех полученных чисел.

Решение

Пусть $a = \overline{xy} = 10x + y$ и $b = \overline{yx} = 10y + x$.

Тогда

$$a + b = 11x + y = 11(x + y).$$

Так как по условию $a + b = 11(x + y) : 5$ и числа 5 и 11 взаимно просты, то

$$(x + y) : 5. \tag{2.3.1}$$

Далее из второго условия $a - b = 9x - 9y = 9(x - y) : 27$, следует, что

$$(x - y) : 3. \tag{2.3.2}$$

Осталось перебрать все возможные значения цифр x и y , удовлетворяющих условиям (2.3.1) и (2.3.2). Непосредственной проверкой можно убедиться, что этим условиям удовлетворяют пары (1; 4), (2; 8), (4; 1), (5; 5), (6; 9), (8; 2) и (9; 6).

Таким образом, получаем пять чисел, сумма которых равна $14 + 28 + 41 + 55 + 69 + 82 + 96 = 385$.

Ответ: 385.

Задача 2.3.3.5. (25 баллов)

Тема: текстовая задача.

Условие

Команда «Математики» за последние три года, согласно протоколам, приняла участие в 111 матчах по мини-футболу (в это число вошли и игры, которые были отменены по техническим причинам). При анализе результатов было замечено:

- сколько-то игр было выиграно;
- ничьи составляют 45% от всех игр, в которых не были одержаны победы;
- количество матчей, в которых были допущены поражения, к количеству отмененных игр относится как 1 : 2.

Какое количество матчей «Математики» проиграли?

Решение

Пусть было одержано x побед. Тогда количество игр, которые были сыграны вничью, проиграны или были отменены, равно $111 - x$.

Тогда $\frac{9}{20}(111 - x)$ — количество игр, сыгранных вничью.

Найдем количество игр, которые были проиграны, или отменены:

$$(111 - x) - \frac{9}{20}(111 - x) = \frac{1221 - 11x}{20}.$$

Тогда количество игр, в которых были поражения, равно

$$y = \frac{1221 - 11x}{60} \in Z.$$

Получили диофантово уравнение

$$11x + 60y = 1221.$$

Выразим x :

$$x = 111 - 60 \cdot \frac{y}{11}.$$

Таким образом, $y : 11$ и $y > 0$.

Рассмотрим различные случаи относительно y :

1. $y = 11$. Тогда $x = 111 - 60 = 51$.
2. $y = 22$. Тогда $x = 111 - 120 = -9$. Количество игр не может быть отрицательным числом. Следовательно, данный случай, как и все последующие, не подходит.

Таким образом, количество игр, в которых были получены поражения, равно 11.

Ответ: 11.

2.3.4. Вторая волна. Задачи 10–11 класса

Задачи второй волны предметного тура по математике за 10–11 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: <https://contest.yandex.ru/contest/63477/enter/>.

Задача 2.3.4.1. (10 баллов)

Темы: стереометрия, центральная симметрия.

Условие

Прямоугольный параллелепипед имеет объем, равный 30. Его рассекли на две части, проведя плоскость через точку пересечения всех трех его диагоналей.

Чему равно максимальное значение объема одной из этих двух частей?

Решение

При центральной симметрии плоскости переходят в параллельные им плоскости, а прямые — в параллельные им прямые. Диагонали параллелепипеда делятся точкой пересечения пополам, поэтому он имеет центр симметрии. При центральной симметрии любая точка параллелепипеда, не находящаяся на секущей плоскости, перейдет в точку, которая находится с другой стороны от любой плоскости, проходящей через этот центр, так как эти две точки и центр симметрии находятся на одной прямой, которая пересекает эту плоскость.

Таким образом, плоскость делит параллелепипед на две части, которые переходят друг в друга при центральной симметрии. Следовательно, их объемы должны быть равны. Это означает, что часть параллелепипеда имеет объем, равный половине объема параллелепипеда

Ответ: 15.

Задача 2.3.4.2. (15 баллов)

Темы: теорема Виета, многочлены.

Условие

Путешественник достал древнюю карту спрятанных сокровищ на острове Пасхи. Путь к пещере, в которой пиратами был закопан клад, был зашифрован с помощью квадратного уравнения. К сожалению, с течением времени запись одного из коэффициентов стерлась, и поэтому путешественник не смог его точно восстановить.

Оказалось, что оно имеет следующий вид: $x^2 + 6x + a = 0$.

Здесь буквой a обозначен неизвестный коэффициент.

Уравнение использовалось для того, чтобы можно было разделить инструкцию по поиску сокровища на несколько частей таким образом, чтобы совершенно невозможно было бы понять, что и где искать, если хотя бы одной части недостает.

У путешественника были все части инструкции, поэтому он смог понять, что нужно от нужной точки на побережье идти ровно P км на юг вдоль единственной тропы, затем $Q = \frac{P}{2}$ км на запад, а потом повернуться на северо-восток и идти прямо, пока вершина вулкана Теревака, кратер Рано-Арои, не станет виден под углом

ровно $10R^\circ$ над уровнем горизонта. Рядом с этим местом и находится пещера. Здесь P, Q — корни данного квадратного уравнения, упорядоченные по возрастанию, $R = 2P + Q$.

Может ли путешественник, исходя из данных условий, однозначно найти два этих корня?

Если может, напишите в ответ число R . Если не может, напишите в ответ число 0.

Решение

Запишем теорему Виета для квадратного уравнения:

$$\begin{cases} P + Q = -6, \\ PQ = a. \end{cases}$$

В условии указано, что $Q = \frac{P}{2}$. Подставив в первое уравнение, получаем, что $P = -4, Q = -2$.

Ответ: -10 .

Задача 2.3.4.3. (20 баллов)

Темы: арифметическая задача, симметрия.

Условие

Исследователи выращивают экспериментальную культуру грибов. Эти грибы размножаются почкованием. Гриб порождает два новых гриба каждые 4 ч. Только что появившийся гриб слишком маленький, и поэтому он должен еще 6 ч расти, прежде чем размножиться, таким образом, первое потомство от нового гриба возникает лишь через 10 ч после его появления из почки.

Сколько грибов, включая только что появившихся, будет в лаборатории через 28 ч, если изначально там был один гриб, который породит два новых гриба только через 4 ч.

Решение

Самый первый гриб за 28 ч успеет породить только три поколения грибов, так как для появления четвертого поколения нужно 30 ч. Поэтому чтобы ответить на вопрос задачи, нужно посчитать, сколько грибов успеют отпочковаться от грибов, которые породил первый гриб, а потом посчитать также третье поколение.

Первые два гриба, отпочковавшиеся через 4 ч, создадут еще четыре гриба в 14 ч, еще четыре — в 18 ч, еще четыре — в 22 ч и еще четыре в — 26 ч. Всего они породят 16 грибов.

Вторые два гриба, появившиеся через 8 ч, создадут еще четыре гриба в 18 ч, еще четыре — в 22 ч и еще четыре гриба — в 26 ч. Всего они породят 12 грибов.

Третьи два гриба, появившиеся через 12 ч, создадут еще четыре гриба в 22 ч, и еще четыре гриба — в 26 ч. Всего они породят восемь грибов.

Четвертые два гриба, появившиеся через 16 ч, создадут еще четыре гриба в 26 ч.

Пятые два гриба — в 20 ч, шестые два гриба — в 24 ч, а седьмые два гриба в 28 ч не успеют породить никаких новых грибов — это еще шесть грибов.

Таким образом, можно посчитать количество грибов первого и второго поколения:

$$N_1 = 7 \cdot 2 = 14;$$

$$N_2 = 16 + 12 + 8 + 4 = 40.$$

Осталось посчитать третье поколение. Оно образуется в 24 ч и 28 ч из первых четырех грибов из первых двух грибов, в 28 ч из вторых четырех грибов из первых двух грибов и в 28 ч из первых четырех грибов из вторых двух грибов. То есть еще восемь грибов:

$$N_3 = 2 \cdot 2 \cdot 2 + 2 \cdot 2 \cdot 2 + 2 \cdot 2 \cdot 2 + 2 \cdot 2 \cdot 2 = 32.$$

Суммарно получаем:

$$N = 1 + N_1 + N_2 + N_3 = 1 + 14 + 40 + 32 = 87.$$

Ответ: 87.

Задача 2.3.4.4. (25 баллов)

Темы: прямоугольный треугольник, теорема Пифагора, теорема косинусов.

Условие

В прямоугольном равнобедренном треугольнике ABC с прямым углом C проведена биссектриса AL . Из точки L к стороне BC проведен перпендикуляр, который пересек сторону AB в точке M . Перпендикуляр, построенный к стороне AB в точке M , пересекает сторону AC в точке N .

Чему равен угол ANL ? Ответ приведите в градусах.

Решение

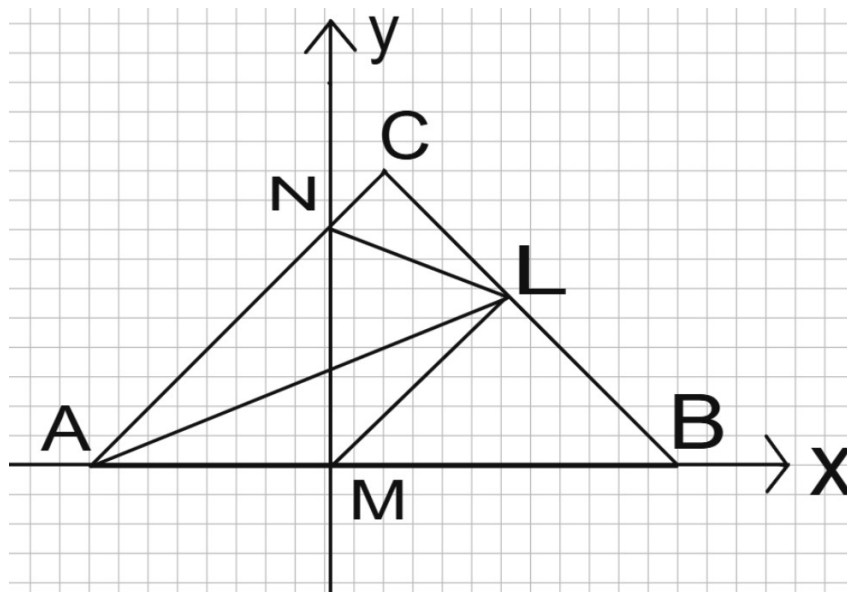
$MN = AM$, значит, угол ANM тоже равен 45° и NM перпендикулярно AB .

Тогда углы NML и BML тоже равны по 45° .

Пусть $AM = 1$ (в условии никаких длин нет, поэтому можем за единицу длины взять любой отрезок). Тогда

$$AN = \sqrt{2};$$

$$AL = 2 \cdot AM \cdot \cos 22,5^\circ = 2 \cdot \sqrt{\frac{\sqrt{2} + 2}{4}} = \sqrt{\sqrt{2} + 2}.$$



Чтобы найти NL , используем метод координат. Проведем горизонтальную ось через AB , вертикальную ось через MN . Тогда точка N имеет координаты $(0; 1)$. Что же касается точки L , то ее координаты x и y совпадают, а длина ML равна 1. Следовательно, они равны $L\left(\frac{\sqrt{2}}{2}; \frac{\sqrt{2}}{2}\right)$.

Используя формулу расстояния между двумя точками, получаем:

$$NL^2 = \frac{1}{2} + \left(1 - \frac{\sqrt{2}}{2}\right)^2 = 2 - \sqrt{2}.$$

Обозначив угол ALN за x , применим теорему косинусов:

$$2 = 2 - \sqrt{2} + \sqrt{2} + 2 - 2 \cdot \sqrt{\sqrt{2} + 2} \cdot \sqrt{2 - \sqrt{2}} \cdot \cos x.$$

Отсюда получаем, что:

$$\cos x = \frac{\sqrt{2}}{2}, \quad x = 45^\circ.$$

Тогда угол ANL равен 180° минус угол ALN и угол NAL :

$$ANL = 180 - 45 - 22,5 = 112,5.$$

Ответ: 112,5.

Задача 2.3.4.5. (30 баллов)

Темы: уравнение параболы, уравнение касательной, угловой коэффициент наклона прямой.

Условие

Для разработки оптической системы на основе параболических отражателей света потребовалось исследовать оптические свойства парабол. Пусть парабола задана уравнением $y = 16x^2$. Требуется на плоскости найти такую точку O , что все проекции этой точки на касательные к параболе лежат на оси абсцисс. Найдите координаты точки O и запишите их в ответ.

Уравнение касательной прямой к параболе (в заданной точке (x_0, y_0)) однозначно устанавливается как уравнение невертикальной прямой, проходящей через (x_0, y_0) и имеющей единственную точку пересечения с параболой.

Решение

Рассмотрим точку с абсциссой x_0 на параболе. Уравнение прямой, проходящей через эту точку, в общем виде имеет вид:

$$y = a \cdot (x - x_0) + 16 \cdot (x_0)^2.$$

Приравняем его к уравнению параболы и найдем, при каком значении a они будут иметь ровно одну точку пересечения:

$$\begin{aligned} 16 \cdot x^2 &= a \cdot (x - x_0) + 16 \cdot (x_0)^2; \\ 16 \cdot x^2 - a \cdot x + x_0 \cdot a - 16 \cdot (x_0)^2 &= 0; \\ D &= a^2 - 4 \cdot 16 \cdot (x_0 \cdot a - 16 \cdot x_0^2) = (a - 32 \cdot x_0)^2 = 0; \\ a &= 32 \cdot x_0. \end{aligned}$$

Итак, запишем уравнение касательной в этой точке к параболе в виде

$$y = 32 \cdot x_0 \cdot (x - x_0) + 16 \cdot (x_0)^2.$$

Эта прямая пересечет ось абсцисс в точке с координатой $x_1 = \frac{x_0}{2}$.

Уравнение прямой, проходящей через эту точку перпендикулярно касательной:

$$y = -\frac{2 \cdot x - x_0}{64 \cdot x_0}.$$

Эта прямая пересечет ось ординат в точке с координатами $(0; 0,015625)$. Координаты этой точки не зависят от значения x_0 , а значит, все такие прямые пройдут через эту точку.

Ответ: $(0; 0,015625)$.

2.3.5. Третья волна. Задачи 8–9 класса

Задачи третьей волны предметного тура по математике за 8–9 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: <https://contest.yandex.ru/contest/63461/enter/>.

Задача 2.3.5.1. (15 баллов)

Тема: текстовая задача.

Условие

Начинающий предприниматель Петров закупил 1 000 единиц некоторого товара и попытался его продать с наценкой 20% за единицу продукции. Однако ожидания предпринимателя не совпали с реальностью, и он смог продать только 40% от своего объема, после чего вынужден был снизить цену на товар на 10%. В результате снижения единица товара стала стоить 5 832 руб. за штуку.

Какую чистую прибыль, то есть разность между деньгами, полученными за продажу товара и затратами на его закупку, получил Петров?

Решение

Пусть x руб. — цена за единицу товара, по которой совершена закупка предпринимателем Петровым. Тогда он первоначально планировал осуществить продажи по цене

$$x + 0,2x = 1,2x.$$

После снижения же цены товар стал стоить

$$1,2x - 0,1 \cdot 1,2x = 1,08x.$$

Так как известно, что после снижения единица товара стала стоить 5 832 руб. за штуку, то

$$1,08x = 5\,832x = 5\,400.$$

Таким образом, товар был закуплен 5 400 руб. за штуку, и общие затраты на его покупку составили 5 400 000 руб.

Согласно условию задачи 400 единиц товара было продано по цене $1,2 \cdot 5\,400 = 6\,480$ руб., и всего было получено за них $6\,480 \cdot 400 = 2\,592\,000$ руб.

Оставшиеся же 600 единиц были проданы по цене 5 832 руб. и получено за них $5\,832 \cdot 600 = 3\,499\,200$ руб.

Тогда чистая прибыль предпринимателя Петрова будет равна

$$2\,592\,000 + 3\,499\,200 - 5\,400\,000 = 691\,200 \text{ руб.}$$

Ответ: 691 200.

Задача 2.3.5.2. (15 баллов)

Тема: комбинаторика.

Условие

Сколько существует нечетных пятизначных чисел, в которых есть хотя бы одна цифра 5?

Решение

Для того чтобы найти количество требуемых чисел, достаточно из общего количества пятизначных нечетных чисел вычесть количество чисел, в которых отсутствует цифра 5.

В десятичной записи нечетного пятизначного числа на последнюю позицию претендует пять вариантов (цифры 1, 3, 5, 7 и 9), на первую — девять вариантов (все цифры, кроме нуля), а на все остальные позиции — по 10 вариантов. Тогда общее количество пятизначных нечетных чисел будет равно

$$9 \cdot 10 \cdot 10 \cdot 10 \cdot 5 = 45\,000.$$

Для записи нечетного пятизначного числа, в десятичной записи которого отсутствует цифра 5, на каждую соответствующую позицию будет на один вариант меньше, тогда общее количество таких чисел будет равно

$$8 \cdot 9 \cdot 9 \cdot 9 \cdot 4 = 23\,328.$$

Тогда количество пятизначных нечетных чисел, в которых присутствует хотя бы одна цифра 5, равно

$$45\,000 - 23\,328 = 21\,672.$$

Ответ: 21 672.

Задача 2.3.5.3. (20 баллов)

Темы: алгебра, система уравнений.

Условие

Наблюдательный Витя для некоторых двух различных чисел заметил интересную особенность: первое число, увеличенное на 4, будет равно квадрату второго числа, уменьшенного на 2; и наоборот, если ко второму числу прибавить 4, то результат будет равен квадрату первого числа, уменьшенного на 2. Найдите сумму квадратов данных двух чисел.

Решение

Пусть x, y — два исходных различных числа. Тогда согласно условиям задачи будем иметь систему уравнений:

$$\begin{cases} x + 4 = (y - 2)^2, \\ y + 4 = (x - 2)^2. \end{cases}$$

Вычитая из первого равенства второе, получим:

$$x - y = (y - 2)^2 - (x - 2)^2 = (y - x)(x + y - 4).$$

Так как числа x, y различны, то отсюда получаем, что $x + y = 3$.

Складывая же уравнения полученной системы, получим

$$x + y + 8 = (y - 2)^2 + (x - 2)^2 = y^2 - 4y + 4 + x^2 - 4x + 4.$$

Из последнего равенства получаем, что

$$x^2 + y^2 = 5(x + y) = 15.$$

Ответ: 15.

Задача 2.3.5.4. (25 баллов)

Темы: теория чисел, остатки.

Условие

Петя записал на доске три числа 391, 604, 888 и задумчиво сказал Васе: «Если я сейчас эти три числа разделю на одно и то же натуральное число, отличное от единицы, то в результате получу один и тот же остаток».

На какое натуральное число Петя планирует произвести деление исходных чисел?

Решение

Обозначим число, на которое производится деление, через x , а остаток через y .

Тогда каждое из записанных Петей чисел можно представить в виде:

$$391 = xm + y,$$

$$604 = xk + y,$$

$$888 = xn + y,$$

где m, k и n — неполные частные, возникающие при делении.

Вычитая из третьего равенства второе, а из второго — первое, получим:

$$284 = x(n - k),$$

$$213 = x(k - m).$$

Вычтем из верхнего равенства нижнее:

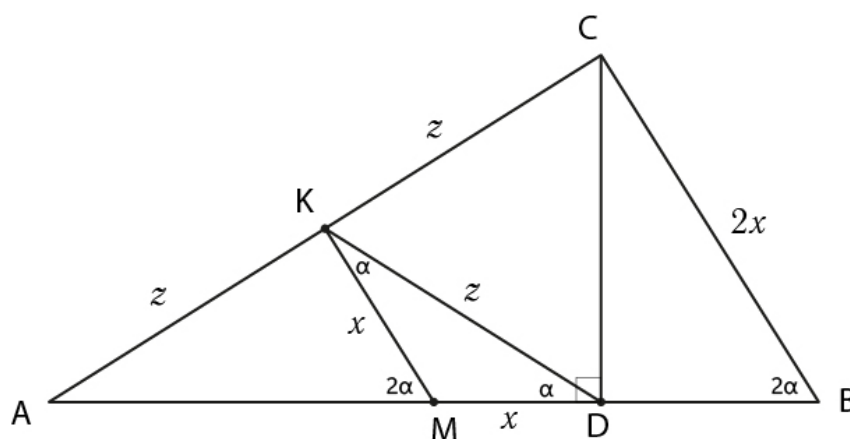
$$71 = x(n - 2k + m).$$

Так как 71 — это простое число, то $71 = 71 \cdot 1$, и, по условию задачи $x \neq 1$, то единственный возможный вариант для делителя Пети равен 71.

Ответ: 71.

Задача 2.3.5.5. (25 баллов)Тема: планиметрия.**Условие**

CD — высота остроугольного треугольника ABC , M — середина стороны AB и $\angle ABC = 2\angle BAC$. Найдите отношение $BC : MD$.

Решение

На стороне AC отметим ее середину — точку K .

Тогда $AK = KC$ и $AM = MB$ (по условию задачи), следовательно, MK — средняя линия треугольника ABC и $BC = 2MK$.

Докажем, что $MK = MD$.

По свойству медианы прямоугольного треугольника, проведенной из вершины прямого угла, в треугольнике ADC : $DK = \frac{1}{2}AC = AK$.

Таким образом, треугольник AKD — равнобедренный и $\angle KAD = \angle KDA$ как углы при основании KD .

Так как MK — средняя линия треугольника ABC , то $MK \parallel BC$ и $\angle AMK = \angle ABC = 2\angle BAC = 2\angle KAD = 2\angle KDA = 2\angle KDM$.

По теореме о внешнем угле для треугольника MKD

$$\angle AMK = \angle KDM + \angle MKD.$$

Тогда из последних двух равенств следует, что $\angle KDM = \angle MKD$ и треугольник MKD — равнобедренный.

Следовательно, $MK = MD$, и так как $BC = 2MK = 2MD$, то $BC : MD = 2 : 1$.

Ответ: 2.

2.3.6. Третья волна. Задачи 10–11 класса

Задачи третьей волны предметного тура по математике за 10–11 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: <https://contest.yandex.ru/contest/63478/enter/>.

Задача 2.3.6.1. (10 баллов)

Темы: осевая симметрия, равнобедренный треугольник, движение.

Условие

Известно, что выпуклая фигура Φ на плоскости устроена таким образом, что она симметрична относительно любой прямой, которая проходит через точку O на этой плоскости. Самое большое расстояние между двумя точками, принадлежащими фигуре Φ , равно дроби, в числителе которой шесть, а в знаменателе квадратный корень из числа π .

Чему равна площадь фигуры Φ ?

Решение

- Докажем, что все точки на границе фигуры равноудалены от центра симметрии O .
 - Возьмем на границе фигуры произвольную точку A . Пусть расстояние $OA = R$.
 - Возьмем любую другую точку B на границе фигуры.
 - Построим прямую I , проходящую через точку O и являющуюся биссектрисой угла AOB .
 - По свойству осевой симметрии, точка A' , симметричная точке A относительно прямой I , также принадлежит фигуре Φ .
 - Поскольку I — биссектриса, точка A' попадет на луч OB . Так как при симметрии расстояние до центра сохраняется ($OA' = OA = R$), точка A' совпадет с точкой B только если $OB = R$.
 - Предположим, что $OB > R$. Тогда точка A лежит внутри отрезка OB . Но поскольку фигура выпуклая, весь отрезок OB должен принадлежать фигуре, а значит, и точка A не может быть граничной. Пришли к противоречию.
 - Предположим, что $OB < R$. Тогда точка B лежит внутри отрезка OA . Это также противоречит тому, что B — граничная точка.
 - Следовательно, единственно возможный вариант — $OB = R$.
 - Поскольку точка B была выбрана на границе произвольно, получается, что все точки границы фигуры Φ находятся на одинаковом расстоянии R от точки O . По определению, это окружность.
- Так как границей является окружность, то сама фигура — круг.
- Используя данное в условии значение диаметра ($6/\sqrt{\pi}$), находим радиус ($3/\sqrt{\pi}$) и вычисляем площадь, которая равна 9.

Ответ: 9.

Задача 2.3.6.2. (15 баллов)

Темы: составление уравнений, составление пропорций, проценты.

Условие

Находясь на борту космического корабля, главный двигатель за первый час израсходовал 40% всего запаса анобтаниума, а вспомогательные двигатели вместе за это же время израсходовали лишь 300 г анобтаниума. За следующий час главный двигатель израсходовал 80% оставшегося топлива, а вспомогательные двигатели израсходовали 100 г топлива на двоих. В итоге на борту корабля осталось 800 г топлива. Сколько килограммов фантастического топлива было на борту до начала полета?

Решение

Найдем массу анобтаниума, оставшегося к концу первого часа.

Не было израсходовано главным двигателем к этому моменту $100 + 800 = 900$ г.

Это составляет $100 - 80 = 20\%$.

Составим пропорцию и решим ее:

$$\begin{array}{l} 20\% - 900, \\ 100\% - ? \end{array}$$

Значит, к концу первого часа оставалось $900 : 0,2 = 4500$ г анобтаниума.

Найдем массу топлива к началу первого часа.

Не было израсходовано главным двигателем к этому моменту $4500 + 300 = 4800$ г, что составляет $100 - 40 = 60\%$.

Составим пропорцию и решим ее:

$$\begin{array}{l} 60\% - 4800, \\ 100\% - ? \end{array}$$

Значит, к началу первого часа было $4800 : 0,6 = 8000$ г, что составляет 8 кг.

Ответ: 8.

Задача 2.3.6.3. (20 баллов)

Темы: уравнение параболы, уравнение прямой.

Условие

Известно, что три различные точки $A(2; 4)$, $B(x; 6)$, $C(6; y)$ расположены на координатной плоскости таким образом, что через них нельзя провести параболу

с вертикальной осью. При этом также известно, что x — минимальное натуральное подходящее число, неравное единице.

Найдите величину $x + y$.

Решение

Через три точки нельзя провести параболу тогда и только тогда, когда они расположены на одной прямой. Действительно, прямая не может пересекать параболу в трех точках, так как квадратное уравнение имеет не больше двух корней. С другой стороны, если три точки не лежат на одной прямой, то через них всегда можно провести параболу. Покажем это.

Пусть на числовой прямой есть три точки с координатами (x_1, y_1) , (x_2, y_2) , (x_3, y_3) . Запишем уравнение параболы в следующем виде:

$$y = y_1 \frac{(x - x_2) \cdot (x - x_3)}{(x_1 - x_2) \cdot (x_1 - x_3)} + y_2 \frac{(x - x_1) \cdot (x - x_3)}{(x_2 - x_1) \cdot (x_2 - x_3)} + y_3 \frac{(x - x_1) \cdot (x - x_2)}{(x_3 - x_1) \cdot (x_3 - x_2)}.$$

Первое слагаемое равно нулю во второй и третьей точке, и равно y_1 в первой, аналогичным образом устроены второе и третье слагаемые, так что это уравнение задает функцию, проходящую через три точки. Однако надо еще проверить, что уравнение задает именно параболу. Для этого нужно, чтобы коэффициент при x^2 не равнялся нулю.

$$\frac{y_1}{(x_1 - x_2) \cdot (x_1 - x_3)} + \frac{y_2}{(x_2 - x_1) \cdot (x_2 - x_3)} + \frac{y_3}{(x_3 - x_1) \cdot (x_3 - x_2)} \neq 0;$$

$$y_1 \cdot (x_2 - x_3) - y_2 \cdot (x_1 - x_3) + y_3 \cdot (x_1 - x_2) \neq 0.$$

Можно убедиться, что это условие означает, что три точки не лежат на одной прямой. А именно, нахождение трех точек на одной прямой можно записать следующим образом:

$$\frac{y_1 - y_2}{x_1 - x_2} = \frac{y_1 - y_3}{x_1 - x_3};$$

$$(y_1 - y_2) \cdot (x_1 - x_3) = (y_1 - y_3) \cdot (x_1 - x_2);$$

$$y_1 \cdot (x_1 - x_3) - y_1 \cdot (x_1 - x_2) = y_2 \cdot (x_1 - x_3) - y_3 \cdot (x_1 - x_2);$$

$$y_1 \cdot (x_2 - x_3) + y_3 \cdot (x_1 - x_2) - y_2 \cdot (x_1 - x_3) = 0.$$

Таким образом, если это условие выполнено, то через три точки проходит прямая, и не проходит никакая парабола. А если оно не выполнено, то проходит единственная парабола, и нельзя провести никакую прямую.

Тогда выразим угловой коэффициент этой прямой тремя разными способами:

$$k = \frac{2}{x - 2} = \frac{y - 6}{6 - x} = \frac{y - 4}{4}.$$

Отсюда получаем, что

$$y = \frac{4 \cdot x}{x - 2}.$$

Минимальное натуральное x , не равное единице, которое подходит — это $x = 3$. Тогда $y = 12$.

Значит, $x + y = 15$.

Ответ: 15.

Задача 2.3.6.4. (25 баллов)

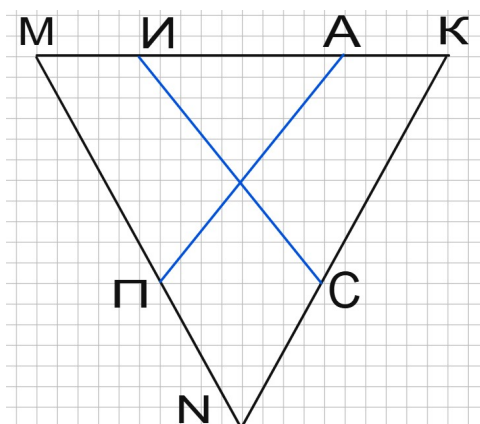
Темы: равносторонний треугольник, первый признак равенства треугольников.

Условие

Три прямые дороги образуют треугольник с равными сторонами, длина которых равна 1 000 м. У этих дорог стоят четыре человека, каждый на обочине одной из трех дорог. Иван и Александр стоят возле одной дороги в 500 м друг от друга. Сергей и Петр стоят у обочин двух других дорог. Сергею идти до Ивана 1 500 м по дорогам кратчайшим путем, Александру до Петра тоже. Между дорогами расположено поле. Какая величина получится, если к расстоянию от Сергея до Ивана по прямой (то есть по полю, а не по дорогам) добавить половину расстояния от Ивана до Александра вдоль дороги, возле которой они стоят, и вычесть расстояние от Александра до Петра по прямой (по полю)?

Решение

Нарисуем расположение всех этих четырех человек. Расположение Сергея и Петра здесь определяется из того условия, что путь до Ивана и Александра соответственно должен занимать 1 500 м, в то время как расстояние по одной стороне не больше 1 000 м, а по другой не больше 500 м.



Используя указанные расстояния, можем записать:

$MA + MP = KI + KC$, а значит, $MI + MP = KA + KC = 1000$ м.

$MI + KA = IA = 500$ м. Кроме того, длина KM равна 1 000 м.

Отсюда выходит, что $KA = 1000 - KC = 1000 - MA$, а значит, $KC = MA$. Аналогично выходит, что $KI = MP$.

Тогда треугольники KIC и MPA равны друг другу по двум сторонам и углу между ними.

Следовательно, $АП = СИ$, а значит, $АП - СИ + 0,5 \cdot ИА = 250$ м.

Ответ: 250.

Задача 2.3.6.5. (30 баллов)

Темы: делители числа, произведение делителей, разложение на множители.

Условие

Количество четных делителей натурального числа в 5 раз больше всех остальных его делителей (рассматриваются все делители, включая само число и единицу). Третья часть всех делителей не делится на 3. Половина четных делителей делится на 5. Само число при этом не превосходит 10 000. Напишите в ответ максимальное число, которое подходит под этим условия.

Решение

Количество четных делителей натурального числа в 5 раз больше всех остальных его делителей.

Это значит, что оно делится на 2^5 степени, но не делится на 2^6 .

Третья часть всех делителей не делится на 3.

Это значит, что оно делится на 3^2 , но не делится на 3^3 .

Половина четных делителей делится на 5.

Это значит, что оно делится на 5, но не делится на 25.

Если перемножим 2^5 на 3^2 и на 5, то получим 1440. Минимальное число, подходящее под условия выше, но большее этого числа, равно $7 \cdot 1440 = 10080 > 10000$.

Следовательно, под все условия подходит только число 1440.

Ответ: 1440.

2.3.7. Четвертая волна. Задачи 8–9 класса

Задачи четвертой волны предметного тура по математике за 8–9 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: <https://contest.yandex.ru/contest/63462/enter/>.

Задача 2.3.7.1. (15 баллов)

Темы: теория чисел, признаки делимости.

Условие

На доске записано число 202420252026. Танечка хочет убрать несколько цифр из исходного числа так, чтобы получившийся результат делился на 45 и являлся наибольшим из всех возможных. Какое число запишет на доске Танечка?

Решение

Для того чтобы число Танечки было бы кратно 45, необходимо выполнение условий делимости на 5 и 9. Следовательно, число должно заканчиваться на 0 или на 5. В данном случае первым делом Танечка должна убрать последние две цифры и получить 2024202520.

Для выполнения условия делимости на 9 необходимо, чтобы сумма цифр числа была бы 9. Сумма цифр сейчас равна 19. Ближайшая сумма, кратная 9, равна 18, но 1 в числе нет, следовательно, следующий вариант — 9. Для этого из оставшегося числа ей нужно вычеркнуть цифры, дающие в сумме 10. Тогда наибольшее число, которое может получить Танечка, — 202050.

Ответ: 202050.

Задача 2.3.7.2. (15 баллов)

Тема: десятичная запись натурального числа.

Условие

Найдите все трехзначные натуральные числа \overline{abc} , удовлетворяющие условию

$$\overline{abc} = \overline{ab} + \overline{bc} + \overline{ca}.$$

В ответ запишите сумму всех найденных чисел.

Решение

Распишем равенство, заданное в условии задач

$$\overline{abc} = \overline{ab} + \overline{bc} + \overline{ca};$$

$$100a + 10b + c = 10a + b + 10b + c + 10c + a;$$

$$100a + 10b + c = 11a + 11b + 11c;$$

$$89a = 10c + b.$$

Так как a, b, c — цифры, то единственным решением данного уравнения является набор $a = 1, b = 9, c = 8$. Следовательно, единственное число, удовлетворяющее условию задачи, это 198.

Ответ: 198.

Задача 2.3.7.3. (20 баллов)

Темы: алгебра, квадратный трехчлен.

Условие

Найдите количество значений параметра b , при которых все корни уравнения $x^2 + bx + 2026 = 0$ целые.

Решение

Пусть x_1 и x_2 — целые корни данного уравнения. Тогда согласно теореме Виета:

$$x_1 \cdot x_2 = 2026.$$

Так как 2026 раскладывается на множители

$$2026 = 1 \cdot 2026 = 2 \cdot 1013,$$

то получаем четыре набора для значений корней

$$(1; 2026), (-1; -2026), (2; 1013), (-2; -1013).$$

Зная значения корней, также по теореме Виета найдем значения параметра b :

$$b = -(x_1 + x_2).$$

Таким образом, всего существует четыре значения параметра $b = \{-2027; 2027; -2015; 2015\}$, при каждом из которых уравнение имеет целые корни.

Ответ: 4.

Задача 2.3.7.4. (25 баллов)

Тема: геометрическая вероятность.

Условие

В треугольнике ABC на биссектрисе BD отмечена точка E так, что $BE = ED$. Найти вероятность, что точка, брошенная в треугольник ABC , попадет в треугольник AED , если $AB = 3$ и $BC = 5$.

Ответ выразите в долях и при необходимости округлите его до четвертого знака после запятой.

Решение

Согласно определению геометрической вероятности, требуемая вероятность будет равна отношению площадей треугольников AED и ABC . AE — медиана треугольника ABE , следовательно, $S_{ABD} = 2S_{AED}$.

Площади треугольников ABD и BDC относятся как длины их оснований AD и DC , то есть

$$\frac{S_{ABD}}{S_{BDC}} = \frac{AD}{DC} = \frac{AB}{BC} = \frac{3}{5}.$$

Последнее равенство выполняется согласно свойству биссектрисы BD в треугольнике ABC . Тогда

$$S_{ABC} = S_{ABD} + S_{BDC} = S_{ABD} + \frac{5}{3}S_{ABD} = \frac{8}{3}S_{ABD} = \frac{16}{3}S_{AED}.$$

Из последнего равенства следует отношение

$$\frac{S_{AED}}{S_{ABC}} = \frac{3}{16} = 0,1875.$$

Таким образом, вероятность того, что точка брошенная в треугольник ABC , попадет в треугольник AED , равна 0,1875.

Ответ: 0,1875.

Задача 2.3.7.5. (25 баллов)

Тема: алгебра.

Условие

При каком значении числа a сумма квадратов чисел x и y будет принимать наибольшее значение, если известно, что сумма этих чисел равна $2a + 1$, а произведение равно $4a^2 + 8a - 4$?

Решение

По условию задачи $x + y = 2a + 1$ и $xy = 4a^2 + 8a - 4$.

Воспользуемся формулой квадрата суммы двух чисел

$$(x + y)^2 = x^2 + 2xy + y^2.$$

Отсюда

$$\begin{aligned} x^2 + y^2 &= (x + y)^2 - 2xy = (2a + 1)^2 - 2(4a^2 + 8a - 4) = 4a^2 + 4a + 1 - 8a^2 - 16a + 8 = \\ &= -4a^2 - 12a + 9 = -(4a^2 + 12a + 9) + 18 = -(2a + 3)^2 + 18. \end{aligned}$$

В полученном выражении первое слагаемое принимает неположительные значения при любом a . Следовательно, сумма квадратов чисел x и y будет максимальной при $2a + 3 = 0$ или $a = -1,5$.

Проверим, что при данном значении параметрам $a = -1,5$ числа x и y действительно существуют. В этом случае $x + y = -2$ и $xy = -7$.

Выразив из первого равенства $y = -x - 2$ и подставив его во второе, после преобразований получим уравнение $x^2 + 2x - 7 = 0$. Дискриминант данного уравнения равен 32, следовательно, корни уравнения существуют, по которым однозначным образом восстанавливаются решения построенной системы. Откуда и следует существования чисел x и y , заданных в условии задачи.

Ответ: $-1,5$.

2.3.8. Четвертая волна. Задачи 10–11 класса

Задачи четвертой волны предметного тура по математике за 10–11 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: <https://contest.yandex.ru/contest/63479/enter/>.

Задача 2.3.8.1. (10 баллов)

Темы: кратчайший путь, параллельный перенос.

Условие

Склад находится в месте, отмеченном на карте точкой A . Нужно проложить дорогу до берега реки, затем построить мост, перпендикулярный течению реки, и от другого берега проложить дорогу до деревни, отмеченной на карте точкой B . Пример подобного построения на рисунке.

Берега реки здесь нарисованы как параллельные прямые. Координатная ось Ox на рисунке отсчитывает положения моста относительно реки в километрах. В примере, приведенном на рисунке, мост проходит через метку 1 км.

Через какую метку должен проходить мост, чтобы сумма длин пути от склада A до реки по дороге и от противоположного берега реки до деревни B была наименьшей? Ответ дайте в километрах.

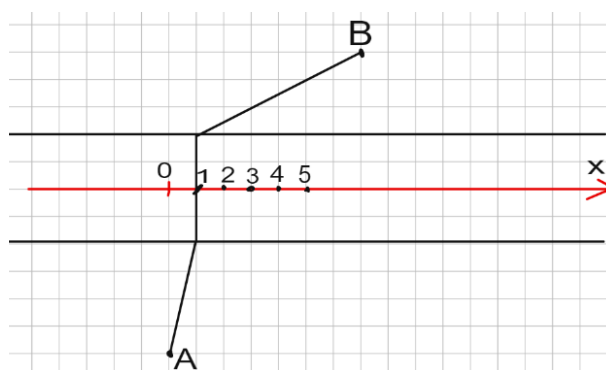


Рис. 2.3.1

Решение

Если вырезать с карты реку и соединить точки A и B прямой, то это и будет кратчайший путь, их соединяющий. Чтобы получить путь до реки, нужно после этого вновь вставить реку. Продемонстрируем эти операции с помощью рис. 2.3.2–2.3.3.

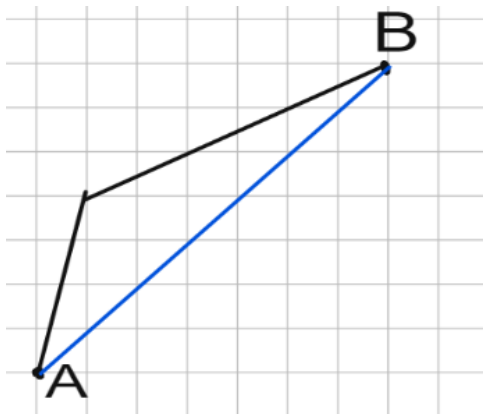


Рис. 2.3.2

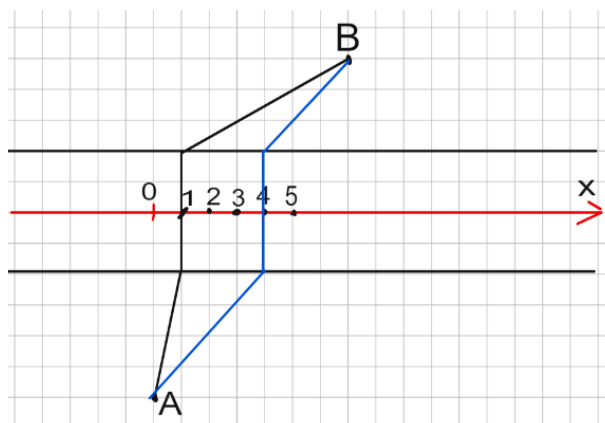


Рис. 2.3.3

Ответ: 4.

Задача 2.3.8.2. (15 баллов)

Темы: составление уравнений, составление пропорций, решение уравнений.

Условие

Два космических корабля стартуют одновременно с одной планеты и направляются к Альфе Центавра, расстояние до которой составляет 4,37 св. лет. Один корабль движется со скоростью 0,1 св. год в год, а другой — со скоростью 0,2 св. год в год.

Через сколько лет расстояние до Альфы Центавра для более быстрого корабля будет в три раза меньше, чем для более медленного корабля? Ответ приведите с точностью до сотых.

Решение

Обозначим время, прошедшее с начала пути, как t лет.

Расстояние, пройденное медленным кораблем, равно $0,1t$ св. год, и оставшееся расстояние до Альфы Центавра для медленного корабля $4,37 - 0,1t$ св. год.

Расстояние, пройденное быстрым кораблем, равно $0,2t$ св. год, и оставшееся расстояние до Альфы Центавра для быстрого корабля $4,37 - 0,2t$ св. год.

По условию задачи, остаток пути для быстрого корабля в 3 раза меньше, чем остаток пути для медленного корабля:

$$4,37 - 0,2t = \frac{1}{3}(4,37 - 0,1t).$$

Умножим обе стороны на 3:

$$3(4,37 - 0,2t) = 4,37 - 0,1t;$$

$$13,11 - 0,6t = 4,37 - 0,1t.$$

Переносим все t в одну сторону и постоянные в другую:

$$13,11 - 4,37 = 0,6t - 0,1t;$$

$$8,74 = 0,5t.$$

Делим обе стороны на 0,5:

$$t = \frac{8,74}{0,5} = 17,48.$$

Таким образом, искомое время равно 17,48 лет.

Ответ: 17,48.

Задача 2.3.8.3. (20 баллов)

Темы: квадратный трехчлен, функции, неопределенные коэффициенты.

Условие

Функция $f(x)$ является квадратным трехчленом и может быть описана следующим образом:

$$f(x) = (f(1) + f(-1) + f(0)) \cdot x^2 + (f(1) + 2 \cdot f(0)) \cdot x - 1.$$

В то же время квадратный трехчлен в общем виде может быть записан так:

$$f(x) = a \cdot x^2 + b \cdot x + c.$$

Найдите минимальное значение величины $a^2 + 2b^2 + 3c^2$ при данных условиях.

Решение

Подставим $f(x)$ в общем виде в первую формулу из условия:

$$a \cdot x^2 + b \cdot x + c = (a + b + c + a - b + c + c) \cdot x^2 + (a + b + c + 2 \cdot c) \cdot x - 1;$$

$$\begin{cases} a = 2 \cdot a + 3 \cdot c, \\ b = a + b + 3 \cdot c, \\ c = -1. \end{cases}$$

Отсюда получаем, что $a = 3$, $c = -1$. Чтобы искомая величина была минимальной, нужно, чтобы коэффициент $b = 0$.

Ответ: 12.

Задача 2.3.8.4. (25 баллов)

Темы: делители числа, произведение делителей, разложение на множители.

Условие

Произведение всех делителей числа 1 000, включая само это число и единицу, равно 10^k .

Чему равно k ?

Решение

$$1\,000 = 2^3 \cdot 5^3.$$

Комбинируя все возможные способы выбрать степень двойки и степень пятерки, входящие в делитель, получаем все $(3+1) \cdot (3+1) = 16$ вариантов, каждый из которых соответствует делителю числа. При этом эти 16 делителей можно разбить на пары, произведение в каждой дает 1 000:

$$1\,000 = 1 \cdot 1\,000 = 2 \cdot 500 = 4 \cdot 250 = 8 \cdot 125 = 5 \cdot 200 = 10 \cdot 100 = 20 \cdot 50 = 25 \cdot 40.$$

Тогда выходит, что это будет число $1\,000^8$, а значит, 10^{24} .

Ответ: 24.

Задача 2.3.8.5. (30 баллов)

Темы: равносторонний треугольник, первый признак равенства треугольников.

Условие

Дан квадрат $ABCD$. На сторонах CB и CD отмечены точки L и K соответственно такие, что $CL = CK$. Из точки C на отрезок LD опущен перпендикуляр в точку E .

Пусть $AE = 60$, $EK = 91$. Найдите длину AK .

Решение

Сделаем рис.2.3.4.

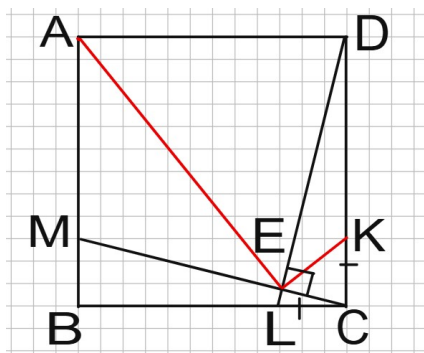


Рис. 2.3.4

Одно из возможных решений заключается в использовании метода координат. Обозначим длину квадрата за единицу, а $CL = CK = x$.

Тогда

$$L(1-x; 0); K(1; x); \vec{LD} = (x; 1); \vec{CE} = t \cdot (1; -x); E(1+t; -x \cdot t); \vec{LE} = (t+x; -x \cdot t).$$

Так как вектора LE и LD должны быть сонаправлены, то

$$\frac{t+x}{x} = -x \cdot t; t = -\frac{x}{1+x^2}; E(1 - \frac{x}{1+x^2}; 1 - \frac{1}{1+x^2});$$

$$\vec{AE} = (1 - \frac{x}{1+x^2}; -\frac{1}{1+x^2}); \vec{EK} = (\frac{x}{1+x^2}; x - \frac{x^2}{1+x^2}).$$

Посчитаем скалярное произведение: $\vec{AE} \cdot \vec{EK} = 0$. Это значит, что треугольник AEK прямоугольный.

Тогда AK можно найти по теореме Пифагора: $60^2 + 91^2 = 109^2$.

Ответ: 109.

2.4. Инженерный тур

Задачи первого этапа инженерного тура открыты для решения. Соревнование доступно на платформе Яндекс.Контест: <https://contest.yandex.ru/contest/66929/enter/>.

Задача 2.4.1. Калибровка одометрии (10 баллов)

Темы: мобильная робототехника, одометрия, численные методы, кинематика, дифференциальная платформа.

Условие

Рассмотрим мобильного робота с дифференциальной платформой. У него есть два колеса (левое и правое), соединенные с двигателем.

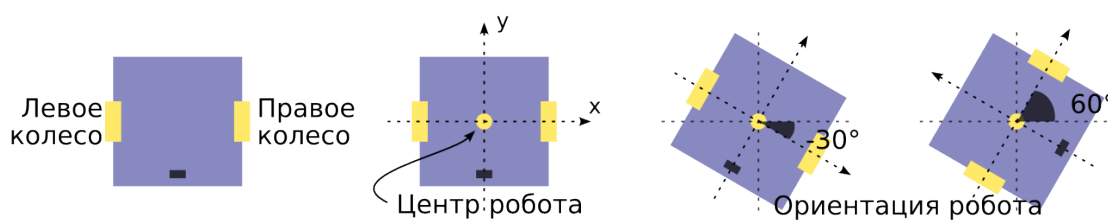


Рис. 2.4.1

Кинематическая модель робота описывается следующим уравнением:

$$\begin{aligned}\dot{p} &= [\dot{x} \quad \dot{y} \quad \dot{\theta}]^T = [-\nu \sin \theta \quad \nu \cos \theta \quad \omega]^T; \\ \nu &= \frac{(\omega_r + \omega_l)r}{2}; \\ \omega &= \frac{(\omega_r - \omega_l)r}{b},\end{aligned}$$

где $r > 0$ и $b > 0$ — радиус колес и база (расстояние между колесами); ω_l, ω_r — скорость вращения колес (левого и правого); x, y — координаты центра робота относительно неподвижной системы отсчета; θ — ориентация робота (угол поворота относительно оси x). Положительное вращение робота соответствует правилу правой руки (положительное направление — против часовой стрелки, отрицательное — по часовой стрелке).

На практике возникает проблема с тем, что параметры робота (радиус колес и расстояние между колесами) оказываются не равны измеренным, например, с помощью штангенциркуля или линейки. В первую очередь это справедливо для роботов, чьи колеса не слишком узкие и имеют мягкое покрытие. В результате возникает задача калибровки параметров одометрии, а именно, по измеренным данным о положении робота и колес определить параметры r и b . Это и требуется сделать в задании.

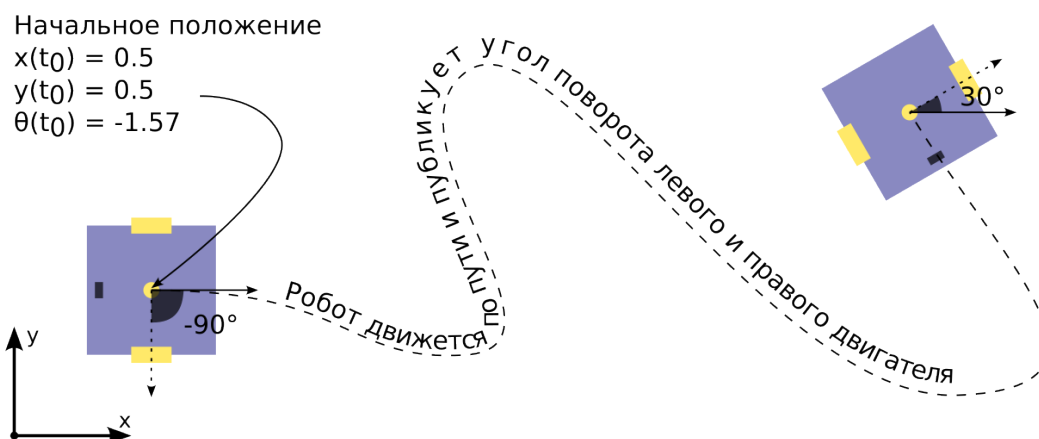


Рис. 2.4.2

Задание

Платформа движется из известного начального положения $x(t_0)$, $y(t_0)$, $\theta(t_0)$, а углы двигателей в начальный момент времени были равны $\varphi_l(t_0)$, $\varphi_r(t_0)$. С постоянным шагом по времени h поступают значения положений робота $x(t_i)$, $y(t_i)$ и углы на двигателях робота $\varphi_l(t_i)$, $\varphi_r(t_i)$, здесь $t_i = t_0 + h \cdot i$, $i \in \mathbb{N}$. Требуется определить радиус колес робота r и расстояние между колесами b .

Формат входных данных

- Первая строка содержит значение $0 < h < 0,1$.
- Вторая строка содержит положение $-100 < x(t_0), y(t_0) < 100$ и ориентацию $-100 < \theta(t_0) < 100$ робота, углы поворота колес $-\pi < \varphi_l(t_0), \varphi_r(t_0) < \pi$ в начальный момент времени, значения разделены пробелом.
- Третья строка содержит значение $5 < n < 10\,000$ количества измерений $x(t_i)$, $y(t_i)$, $\varphi_l(t_i)$, $\varphi_r(t_i)$.
- Следующие n строк хранят значения $-10\,000 < x(t_i), y(t_i), \varphi_l(t_i), \varphi_r(t_i) < 10\,000$, разделенные пробелом.

Все величины указаны в единицах СИ.

Формат выходных данных

- Первая строка содержит значение параметра r и b , разделенные пробелом.

При выводе округлите значения до шести знаков после запятой.

Примеры*Пример №1*

Стандартный ввод					
3.859787684798448e-05					
0.000000	0.000000	1.000000	0.000000	0.000000	0.000000
10					
0.000000	0.000000	1.000000	0.000000	0.000000	0.000000
-0.009552	0.006135	0.999688	0.001930	0.000000	
-0.019102	0.012274	0.999375	0.003860	0.000000	
-0.028651	0.018415	0.999063	0.005790	0.000000	
-0.038197	0.024560	0.998750	0.007720	0.000000	
-0.047741	0.030707	0.998438	0.009649	0.000000	
-0.057284	0.036857	0.998125	0.011579	0.000000	
-0.066825	0.043011	0.997813	0.013509	0.000000	
-0.076363	0.049167	0.997500	0.015439	0.000000	
-0.085900	0.055327	0.997188	0.017369	0.000000	
Стандартный вывод					
11.765247 72.667897					

Пример №2

Стандартный ввод					
3.859787684798448e-05					
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
30					
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
-0.000002	0.013623	0.000250	0.000386	0.001930	
-0.000007	0.027247	0.000500	0.000772	0.003860	
-0.000015	0.040870	0.000750	0.001158	0.005790	
-0.000027	0.054494	0.001000	0.001544	0.007720	
-0.000043	0.068117	0.001250	0.001930	0.009649	
-0.000061	0.081740	0.001500	0.002316	0.011579	
-0.000083	0.095364	0.001750	0.002702	0.013509	
-0.000109	0.108987	0.002000	0.003088	0.015439	
-0.000138	0.122611	0.002250	0.003474	0.017369	
-0.000170	0.136234	0.002500	0.003860	0.019299	
-0.000206	0.149857	0.002750	0.004246	0.021229	
-0.000245	0.163481	0.003000	0.004632	0.023159	
-0.000288	0.177104	0.003250	0.005018	0.025089	
-0.000334	0.190727	0.003500	0.005404	0.027019	
-0.000383	0.204351	0.003749	0.005790	0.028948	
-0.000436	0.217974	0.003999	0.006176	0.030878	
-0.000492	0.231597	0.004249	0.006562	0.032808	
-0.000552	0.245220	0.004499	0.006948	0.034738	
-0.000615	0.258844	0.004749	0.007334	0.036668	
-0.000681	0.272467	0.004999	0.007720	0.038598	
-0.000751	0.286090	0.005249	0.008106	0.040528	
-0.000824	0.299713	0.005499	0.008492	0.042458	
-0.000901	0.313337	0.005749	0.008878	0.044388	
-0.000981	0.326960	0.005999	0.009263	0.046317	
-0.001064	0.340583	0.006249	0.009649	0.048247	
-0.001151	0.354206	0.006499	0.010035	0.050177	
-0.001241	0.367829	0.006749	0.010421	0.052107	
-0.001335	0.381452	0.006999	0.010807	0.054037	
-0.001432	0.395075	0.007249	0.011193	0.055967	

Стандартный вывод

11.765247 72.667897

*Пример №3***Стандартный ввод**

```

3.859787684798448e-05
0.000000 0.000000 0.000000 0.000000 0.000000
30
0.000000 0.000000 0.000000 0.000000 0.000000
0.000002 -0.013623 0.000250 -0.001930 -0.000386
0.000007 -0.027247 0.000500 -0.003860 -0.000772
0.000015 -0.040870 0.000750 -0.005790 -0.001158
0.000027 -0.054494 0.001000 -0.007720 -0.001544
0.000043 -0.068117 0.001250 -0.009649 -0.001930
0.000061 -0.081740 0.001500 -0.011579 -0.002316
0.000083 -0.095364 0.001750 -0.013509 -0.002702
0.000109 -0.108987 0.002000 -0.015439 -0.003088
0.000138 -0.122611 0.002250 -0.017369 -0.003474
0.000170 -0.136234 0.002500 -0.019299 -0.003860
0.000206 -0.149857 0.002750 -0.021229 -0.004246
0.000245 -0.163481 0.003000 -0.023159 -0.004632
0.000288 -0.177104 0.003250 -0.025089 -0.005018
0.000334 -0.190727 0.003500 -0.027019 -0.005404
0.000383 -0.204351 0.003749 -0.028948 -0.005790
0.000436 -0.217974 0.003999 -0.030878 -0.006176
0.000492 -0.231597 0.004249 -0.032808 -0.006562
0.000552 -0.245220 0.004499 -0.034738 -0.006948
0.000615 -0.258844 0.004749 -0.036668 -0.007334
0.000681 -0.272467 0.004999 -0.038598 -0.007720
0.000751 -0.286090 0.005249 -0.040528 -0.008106
0.000824 -0.299713 0.005499 -0.042458 -0.008492
0.000901 -0.313337 0.005749 -0.044388 -0.008878
0.000981 -0.326960 0.005999 -0.046317 -0.009263
0.001064 -0.340583 0.006249 -0.048247 -0.009649
0.001151 -0.354206 0.006499 -0.050177 -0.010035
0.001241 -0.367829 0.006749 -0.052107 -0.010421
0.001335 -0.381452 0.006999 -0.054037 -0.010807
0.001432 -0.395075 0.007249 -0.055967 -0.011193

```

Стандартный вывод

11.765247 72.667897

Решение

1. Рассчитать линейную и угловую скорости по осям координат (первую производную положений x и y), численно проинтегрировав измерения положений x и y .
2. Из кинематической модели рассчитать линейную скорость и угловое положение робота, используя значение $\theta(t_0)$.
3. Рассчитать угловую скорость, численно проинтегрировав полученные значения ориентации $\theta(t)$.
4. Рассчитать угловые скорости колес робота, численно $\omega_l(t)$ и $\omega_r(t)$, проинтегрировав измерения их угловых положений $\varphi_l(t)$, $\varphi_r(t)$.

5. С помощью выражения $r = \frac{2\nu}{(\omega_r + \omega_l)}$ и полученных значений $\omega_l(t)$ и $\omega_r(t)$ получить оценку радиуса колес.
6. Из выражения $b = \frac{(\omega_r + \omega_l)r}{\omega}$, используя оценку r , найти значение b .

Реализация описанного алгоритма на языке программирования Python представлена в файле https://disk.yandex.ru/d/xhgXxuKfe7QhEQ/solution_1.py.

Задача 2.4.2. Моделирование динамических систем (10 баллов)

Темы: моделирование, динамические системы, численные методы.

Условие

Реальные технические и физические системы на макроуровне отличаются тем, что все сигналы в них непрерывны. При построении их моделей естественным образом возникают производные. Например, скорость — это первая производная положения, а ускорение является его второй производной, через операцию дифференцирования связаны ток в катушке индуктивности и падение напряжения на ней.

В результате математические модели для описания таких систем и их функционирования состоят из дифференциальных уравнений. В этих уравнениях, в отличие от алгебраических, решением является не какое-то значение переменной, а функция.

Например, решением дифференциального уравнения

$$\ddot{y}(t) + y(t) = 0,$$

где точка над символом означает взятие производной по времени, а две точки — вторую производную, является функция $y(t) = a \sin(t + b)$, здесь a и b — некоторые параметры, которые определяются начальными условиями. Например, если $y(0) = 1$, $\dot{y}(0) = 0$, тогда подставив в найденное решение и его первую производную $t = 0$, получим

$$\begin{aligned} 1 &= a \sin(b), \\ 0 &= a \cos(b), \end{aligned}$$

откуда можно найти, что $a = 1$, $b = \pi/2$.

Для проверки найдем вторую производную и подставим в уравнение:

$$\begin{aligned} \dot{y}(t) &= a \cos(t + b); \\ \ddot{y}(t) &= -a \sin(t + b); \\ -a \sin(t + b) + a \sin(t + b) &= 0; \\ 0 &= 0 - \text{верно.} \end{aligned}$$

Далее, имея математическую модель, можно провести численное моделирование и проанализировать ее поведение. Для того чтобы проводить моделирование таких динамических моделей, применяются численные методы решения дифференциальных уравнений.

Самым простым из них является метод Эйлера. Для общности рассмотрим модель следующего вида:

$$\dot{y}(t) = f(t, y(t), u(t)),$$

где t — время, а u — некоторый измеряемый входной сигнал, то есть его значение в текущий момент времени известно, например, это может быть сигнал управления.

Для численного решения дифференциального уравнения нужно взять некоторое начальное условие $y(0)$ и определить шаг интегрирования h . Тогда можно рассчитать значение функции $y(t)$ в момент времени h следующим образом:

$$y(h) = y(0) + hf(0, y(0), u(0)).$$

Затем можно последовательно рассчитывать значение функции $y(t)$ в следующие моменты времени с шагом h :

$$\begin{aligned} y(2h) &= y(h) + hf(h, y(h), u(h)); \\ y(3h) &= y(2h) + hf(2h, y(2h), u(2h)); \\ &\dots \\ y(kh + h) &= y(kh) + hf(kh, y(kh), u(kh)). \end{aligned}$$

Суть метода: предполагается, что в течение периода времени в h секунд (это могут быть и доли секунды, например, 0,0001 с) среднее значение производной будет равно значению производной в начале шага. Это равносильно предположению, что достаточно точно можно аппроксимировать функцию на этом шаге прямым отрезком, а в данном случае следующее значение функции может быть рассчитано как сумма текущего значения функции и значения производной функции, умноженной на временной шаг h .

При использовании достаточно малого h это допущение можно считать вполне оправданным. Другое дело, что в общем случае шаг должен быть очень малым, иначе данный метод может давать значительную ошибку, но это увеличивает количество вычислений при моделировании одного и того же периода времени работы системы.

Более точным является метод Коши или метод средней точки. Идея заключается в том, что можно получить более высокую точность, если в качестве некоторого среднего значения производной возьмем значение не из начала шага, а из его середины. Действительно, в общем случае можно ожидать, что значение производной в конце шага скорее будет ближе к значению производной в середине шага, чем к значению в его начале.

Проблема состоит в том, что невозможно рассчитать значение производной в середине шага, так как для этого требуется знать значение функции в середине шага $y(t + h/2)$:

$$\dot{y}(t) = f(t, y(t), u(t)),$$

а значит,

$$\dot{y}(t + h/2) = f(t + h/2, y(t + h/2), u(t + h/2)).$$

Эту проблему можно решить, используя метод Эйлера:

$$y(t + h/2) = y(t) + \frac{h}{2}f(t, y(t), u(t)).$$

Заметим, что в этом случае шаг для метода Эйлера составляет $h/2$, а не h , что положительно сказывается на точности решения. Теперь можно рассчитать значение в момент времени $y(t + h)$:

$$y(t + h) = y(t) + hf(t + h/2, y(t + h/2), u(t + h/2)).$$

Задание

Напишите программу, выполняющую численное решение дифференциального уравнения методом Коши с постоянным шагом по времени h для заданного начального условия $y(0)$:

$$\dot{y}(t) = \frac{ay(t) + bt^2 + c}{dt + e} + f,$$

где a, b, c, d, e, f — параметры системы.

Формат входных данных

- Первая строка хранит параметры системы a, b, c, d, e, f , разделенные пробелом в виде вещественных чисел в диапазоне $[-100; 100]$. Гарантируется, что параметры являются корректными и решение дифференциального уравнения существует.
- Вторая строка хранит начальное условия для интегрирования $y(0)$.
- Третья строка хранит шаг интегрирования $0 < h < 1$, размер $5 < n < 100\,000$ — сколько шагов при численном решении требуется сделать, разделенных пробелом.

Все величины указаны в единицах СИ.

Формат выходных данных

- Первая строка содержит число $s = n + 1$ с результатами вычислений задания.
- Далее идут s строк со значениями $y(ih)$, где $i = 0, 1, 2, \dots, n$.

При выводе округлите значения до шести знаков после запятой.

Примеры*Пример №1*

Стандартный ввод
1.0 1.0 -2.0 1.0 1.0 0.0
2.0
0.1 9
Стандартный вывод
10
2.000000
2.001818
2.016480
2.053692
2.120697
2.223114
2.365425
2.551294
2.783768
3.065426

Пример №2

Стандартный ввод
1.228789 3.391090 -8.000000 3.000000 4.000000 0.000000
4.923386
0.695078 9
Стандартный вывод
10
4.923386
4.682327
4.818577
5.451902
6.641021
8.419785
10.809752
13.825631
17.477967
21.774620

Пример №3

Стандартный ввод
6.240544 6.747381 5.000000 2.000000 1.000000 0.000000
8.437533
0.637444 9
Стандартный вывод
10
8.437533
75.791111
273.755572
684.289592
1394.034051
2492.819850
4072.940261
6228.718086
9056.214864
12653.019828

Решение

1. Реализовать метод Коши согласно выражению $\dot{y}(t) = \frac{ay(t)+bt^2+c}{dt+e} + f$.
2. Подставить начальное условие и все параметры в выражение для производной.
3. Далее в цикле вычислять решение системы на основе реализации метода (см. пункт 1).

Реализация описанного алгоритма на языке программирования Python представлена в файле https://disk.yandex.ru/d/xhgXxuKfe7QhEQ/solution_2.py.

Задача 2.4.3. Устойчивость динамических систем (9 баллов)

Темы: теория автоматического управления, динамические системы, устойчивость, моделирование.

Условие

Положение равновесия динамической системы — это такое положение, в котором система не движется, все скорости, ускорения и более старшие производные выходной переменной равны нулю. Например, модель маятника имеет два положения равновесия: верхнее и нижнее.

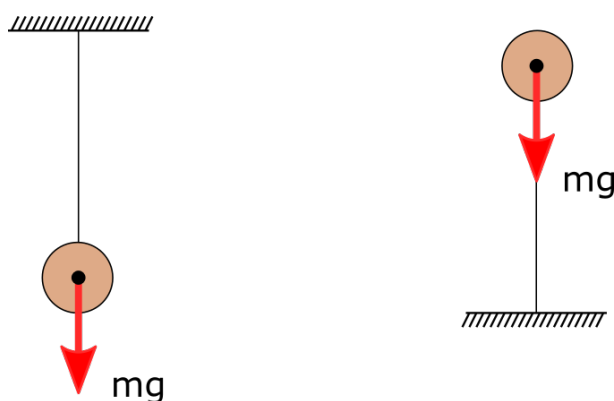


Рис. 2.4.3

Положения равновесия могут быть устойчивыми и неустойчивыми. В устойчивое положение равновесия система возвращается, если находилась в некоторой его окрестности, например, после прекращения внешнего воздействия. У маятника верхнее положение является неустойчивым, а нижнее — устойчивым.

У некоторых систем только одно положение равновесия, в таком случае часто говорят об устойчивости самой системы. Если это единственное положение равновесия устойчиво, то система будет в него возвращаться из любого начального положения при отсутствии внешнего воздействия. В противном случае это будет происходить только из некоторой окрестности положения равновесия.

Есть несколько разных способов рассмотрения устойчивости динамических систем, и в этом задании рассмотрим устойчивость системы по выходу, т. е. как изменится ее выходная переменная. В этом случае, кроме устойчивых и неустойчивых систем (или положений равновесия), выделяют системы, находящиеся на границе устойчивости. То есть такие системы, которые при отклонении от положения равновесия хоть и не возвращаются в него, но и не отдаляются, а так и продолжают находиться в его окрестности. Система при этом может колебаться в этой окрестности или находится неподвижно.

Например, нижнее положение маятника без трения находится на границе устойчивости (колебательной). Если отклонить маятник от него, то он начнет бесконечно колебаться относительно нижнего положения, так как трение отсутствует. Здесь важно заметить, что верхнее положение неустойчиво, оно находится на границе устойчивости. Маятник хоть и не может бесконечно далеко отклониться от него, но он будет колебаться в области, которая не включает верхнее положение равновесия, в отличие от нижнего.

Среди систем, приведенных ниже, требуется определить систему, чье положение равновесия $[x_1, x_2] = [0, 0]$ находится на границе устойчивости.

В обозначениях ниже точка над буквой означает производную, т.е. \dot{x} — это производная x .

Решение

1. Простое решение — численными методами решить систему и посмотреть на то, как меняется выходная переменная при различных начальных условиях из окрестности нулевого положения равновесия. Если она сходится к нулю, значит, нулевое положение равновесия устойчиво.
2. Если участник знаком с теорией устойчивости, то для линейных систем он может проверить один из критериев устойчивости. Например, найти собственные числа матрицы. В случае, если они чисто мнимые, можно сделать вывод, что система находится на (колебательной) границе устойчивости.

Ответ:

1. Неустойчивая:

$$\begin{cases} \dot{x}_1 = x_1 + 2x_2 + u, \\ \dot{x}_2 = 3x_1 + 4x_2 + u. \end{cases}$$

2. На границе устойчивости:

$$\begin{cases} \dot{x}_1 = x_2 + u, \\ \dot{x}_2 = -2x_1 + u. \end{cases}$$

3. На границе устойчивости:

$$\begin{cases} \dot{x}_1 = u, \\ \dot{x}_2 = u. \end{cases}$$

4. Неустойчивая:

$$\begin{cases} \dot{x}_1 = x_1 + 2x_2 + u, \\ \dot{x}_2 = 2x_1 + x_2 + u. \end{cases}$$

5. Устойчивая:

$$\begin{cases} \dot{x}_1 = -x_1 + 2x_2 + u, \\ \dot{x}_2 = -2x_1 - x_2 + u. \end{cases}$$

Задача 2.4.4. Максимизация пропускной способности пути (10 баллов)

Темы: теория графов, обходы в графах, методы планирования траектории, программирование.

Условие

Ограничение: TL — 10 с, ML — 64 МБ.

Задана карта в виде двумерного массива размером $N \times M$, $1 \leq N, M \leq 100$. Каждая клетка задается парой чисел (i, j) , где i ($1 \leq i \leq N$) — номер строки,

j ($1 \leq j \leq M$) — номер столбца. В каждой клетке лежат целые числа в диапазоне от 0 до 255, назовем это число стоимостью клетки, означающей получаемую награду роботом.

Робот Саша хочет добраться из клетки S с координатами (x_s, y_s) в клетку F с координатами (x_f, y_f) , проходя только по клеткам с наградой. Саша занимает собой одну клетку, а двигаться может только в соседние клетки (соседние по ребрам клетки, а не по диагонали), причем робот не будет вставать на клетку, которая не содержит награду (то есть в клетку со стоимостью 0), а также не может посещать уже пройденные клетки, ведь там не осталось награды.

Помогите найти Саше путь из точки S в точке F такой, что минимальная полученная по пути награда является максимальной. Обращаем внимание, что имеется ввиду не сумма наград по пути, а величина самой маленькой награды по пути, полученная в одной из клеток.

Сумму наград по пути будем называть стоимостью пути, его тоже надо будет посчитать, но задача робота — максимизировать самую маленькую награду, которая встретится на пути.

Награда в начальной точке не рассматривается, она не должна входить в рассчитываемую стоимость пути и не должна рассматриваться при определении минимальной награды на пути.

Формат входных данных

- В первой строке заданы два целых числа: N и M .
- Далее идут четыре целых числа: x_s, y_s, x_f, y_f — координаты точек S и F соответственно.
- Далее задана карта в виде двумерного массива размером $N \times M$ в виде таблицы из N строк и M столбцов с целыми числами от 0 до 255.

Формат выходных данных

- В первой строке выведите три числа: k — количество клеток в пути, S — стоимость пути, D — минимальная полученная по пути награда.
- В следующих k строках выведите координаты вершин (i, j) в пути в порядке обхода, по одной вершине в строке (начальная и конечная вершины тоже считаются).
- В случае, если построить путь без пустых клеток нельзя, вывести -1 .

Примеры

Пример №1

Стандартный ввод			
1	4		
1	1	1	4
1	1	1	1

Стандартный вывод
4 4 1
1 1
1 2
1 3
1 4

Пример №2

Стандартный ввод
3 3
3 1 1 3
255 0 255
255 0 255
255 0 255
Стандартный вывод
-1

Решение

1. Представить исходную карту в виде ориентированного графа. Таким образом, стоимость вершины будет отождествляться со стоимостью ребра. Если стоимость вершины равна 0, то данная вершина не имеет ребер (как из нее, так и в нее).
2. Для поиска требуемого пути можно модифицировать алгоритм Дейкстры так, чтобы сохранять не кратчайший путь до вершины, а вес минимального ребра в этом пути, а при релаксации отдавать предпочтение ребрам с большим весом.

Реализация описанного алгоритма на языке программирования Python представлена в файле https://disk.yandex.ru/d/xhgXxuKfe7QhEQ/solution_4.py.

Задача 2.4.5. Создание графа (14 баллов)

Темы: теория графов, обход графа, программирование.

Условие

Ограничение: TL — 10 с, ML — 488,3 МБ.

Дано поле размером $N \times M$, $1 \leq N, M \leq 100$ квадратных единиц, по которому ездит круглый робот Саша. Диаметр робота Саши равен D . Саша начинает путь из точки (x_s, y_s) .

Саше известно, что на поле расположены K препятствий, которые являются прямоугольниками и задаются двумя точками (p_1, p_2) (это диагонально противоположные точки). Гарантируется, что все препятствия находятся в пределах поля,

а также гарантируется, что координаты вершин препятствий являются целыми числами. Саше известна конечная точка (x_f, y_f) , до которой ему нужно доехать. Задача: составить граф на поле так, чтобы робот мог перемещаться по вершинам и ребрам, не сталкиваясь с препятствиями. Таким образом, каждая вершина графа будет задаваться своими координатами (x_i, y_i) , кроме того, начальная и конечная точки также должны быть в графе. Количество ребер и вершин в графе может быть любым, с учетом ограничений задачи.

В ответ необходимо вывести координаты вершин, по которым нужно пройти роботу, чтобы добраться из своей стартовой точки до конечной, не сталкиваясь с препятствиями.

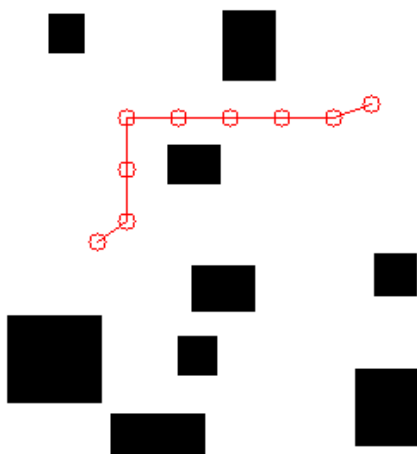


Рис. 2.4.4

Особенности учета расстояний, размеров и границ

- Робот Саша очень осторожен и остановится, если расстояние по любой из осей до препятствия от его центра станет равным или меньше его радиуса, то есть робот не поедет в точку $(1; 1)$, если в точке $(2; 2)$ есть препятствие, несмотря на то, что расстояние между этими точками $\sqrt{2}$.
- Робот может ездить по границе поля.

Формат входных данных

- В первой строке заданы четыре числа: $N \in \mathbb{N}$, $M \in \mathbb{N}$, $K \in \mathbb{N}$, $D \in \mathbb{R}_+$ такие, что $1 \leq N, M \leq 100$, $1 \leq K \leq 10$, $1 \leq D \leq 50$.
- Далее идут четыре вещественных числа: x_s, y_s, x_f, y_f — координаты точек S и F соответственно.
- Далее идут K строк, в каждой описаны четыре точки (каждая задается двумя вещественными числами), задающие препятствия-прямоугольники.

Формат выходных данных

- В первой строке выведите одно число C — количество вершин в пути.
- В следующих C строках выведите координаты вершин (x_i, y_i) в пути в порядке обхода, по одной вершине в строке (начальная и конечная вершины тоже считаются).
- Заметьте, не требуется найти минимальный путь — достаточно найти любой путь.
- В случае, если построить путь без пересечения препятствия нельзя, в том числе из-за того, что начальная или конечная точка слишком близко к препятствию, вывести -1 .

Примеры

Пример №1

Стандартный ввод
8 8 1 1
7.0 1.0 1.0 7.0
1.0 1.0 4.0 4.0
Стандартный вывод
15
7 1
6 1
6 2
6 3
6 4
6 5
6 6
5 6
4 6
3 6
2 6
1 6
1 7

Решение

Один из способов решения данной задачи заключается в инициализации графа карты с учетом размера робота и препятствиями.

1. Чтобы учесть границы препятствий и размеры робота, нужно сделать следующее преобразование. В качестве радиуса робота использовать нулевое значение, а все линейные размеры препятствий увеличить на величину, равную половине радиуса робота.
2. Задать матрицу для представления пространства, где будут отмечены свободные ячейки и ячейки, занятые препятствиями.
3. Убедиться, что после преобразования из предыдущего пункта начальная и конечная точки не оказались в пределах препятствий.

4. Полученную матрицу представить в виде графа: на исходное поле натягивается сетка. Вершины сетки соответствуют точкам, в которые робот может переходить, а ребра соединяют возможные перемещения.
5. Задать шаг, на который робот будет перемещаться. С учетом шага инициализировать новый граф смежности, в котором две вершины будут смежными в том случае, если обе вершины свободны и расстояние между ними равно одному шагу.
6. Для каждой полученной вершины проверить, что существует путь к начальной точке и к конечной точке без пересечения препятствий. Тем самым формируется конечный граф, учитывающий заданный шаг робота, а также свободное и занятое пространство.
7. С помощью алгоритма Дейкстры найти путь от начальной точки к конечной в графе из пункта 6. Повторять шаги 5–7, уменьшая шаг, пока не будет найден путь, или шаг не станет меньше единицы.

Реализация описанного алгоритма на языке программирования Python представлена в файле https://disk.yandex.ru/d/xhgXxuKfe7QhEQ/solution_5.py.

Задача 2.4.6. Песок и камни (14 баллов)

Темы: компьютерное зрение, геометрия, программирование.

Ограничение

TL — 25 с, ML — 320 МБ.

Условие

Беспилотный автономный роботизированный вертолет «Изобретательность», выполняющий исследовательскую миссию на Марсе, после очередного полета сделал снимок местности вокруг точки его приземления.

Требуется написать программу для определения расположения препятствий на полученном от марсианского вертолета изображении песчаной местности. Само изображение было обработано внутренней программой вертолета для поиска препятствий: убрано большинство лишних деталей и изображение бинаризовано (есть только черные и белые пиксели).

К сожалению, несмотря на обработку изображения, на нем, кроме препятствий, остались отдельные следы песчинок. Препятствием для последующей посадки вертолета являются камни, представляющие собой группу окрашенных в черный цвет пикселей размером не меньше, чем 2×2 пикселя.

В качестве ответа для каждого препятствия нужно вывести координаты его центра масс. В данном случае центром масс камня будем считать среднее по всем положениям пикселей, принадлежащих данному камню. Центр массы камня считается найденным правильно, если полученный ответ отличается от истинного не более, чем на 1 пиксель в каждую сторону или по диагонали. Гарантируется, что пиксели, представляющие собой песок, не являются соседними, в том числе по диагонали, и не граничат с камнями.

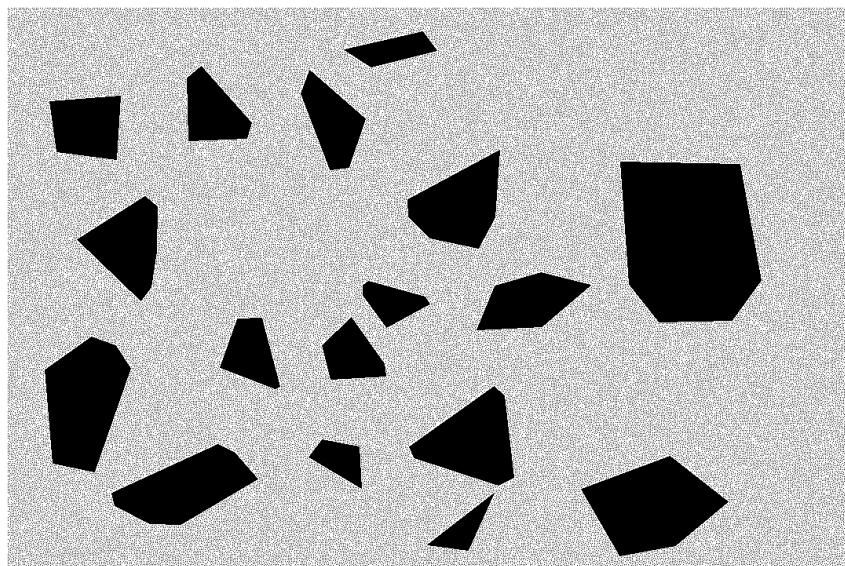


Рис. 2.4.5

Формат входных данных

- В первой строке заданы два числа W и H — ширина и высота изображения.
- Далее следующие H строк построчно поступает бинаризованное изображение в виде двумерного массива, элементы строки разделены пробелами.

Формат выходных данных

- В первой строке поступает количество камней $0 \leq n \leq 50$ на изображении.
- Далее в отдельных строках поступают по два числа, задающих координаты центра масс для соответствующего препятствия.

Примечания

При использовании компилятора (make) Python 3.12.3 будут доступны для использования Python пакеты `scipy`, `sklearn`, `numpy` и `cv2`.

Обратите внимание на то, что направление осей отличается от направления осей, принятого при работе с изображениями на компьютере. В качестве положительного направления осей используйте направление осей OpenCV (ось x смотрит вправо, ось y вниз).

Строки, соответствующие изображениям слишком велики, поэтому вместо примера ввода предлагаем скачать файл по ссылке https://disk.yandex.ru/d/162rd_j5CR013A, содержащий один из тестов, для тестирования на вашем ПК.

Примеры**Пример №1**

Стандартный ввод
2048 2048
255 255 255 255 ...
255 0 255 255 ...
255 255 255 0 ...
255 255 255 255 ...
...
Стандартный вывод
2
251 109
357 152

Решение

В качестве рекомендуемого способа решения данной задачи предлагается использовать метод DBSCAN библиотеки `sklearn`.

1. Методом DBSCAN обнаружить кластеры, группы пикселей, которые будут соответствовать профилю камней. В качестве признаков для кластеризации необходимо использовать положения пикселей, а минимальное расстояние между кластерами задать как 1 пиксель.
2. Для каждого кластера найти центр масс.

В качестве альтернативного решения можно использовать подход, основанный на операции эрозии.

1. Применить метод эрозии к изображению (использовать функцию `cv2.erode` из библиотеки `OpenCV`). Другим способом избавления от шумов является применение медианного фильтра с ядром 3×3 .
2. Осуществить сегментацию изображения. Можно применить такие методы, как K-Means или методы для выделения контуров.
3. Анализ объектов: после сегментации можно вычислить центры оставшихся объектов аналогично тому, как это делалось в предыдущем способе решения.

Реализация описанного алгоритма на языке программирования Python представлена в файле https://disk.yandex.ru/d/xhgXxuKfe7QhEQ/solution_6.py.

Задача 2.4.7. Определение столкновений и опасных ситуаций (16 баллов)

Темы: компьютерное зрение, система позиционирования, мобильная робототехника, программирование.

Ограничение

TL — 25 с, ML — 320 МБ.

Условие

Рассмотрим мобильного робота с дифференциальной платформой. У него есть два колеса (левое и правое), соединенных с мотором. У робота есть передняя и задняя часть (ось y направлена на переднюю часть), его корпус имеет форму квадрата.

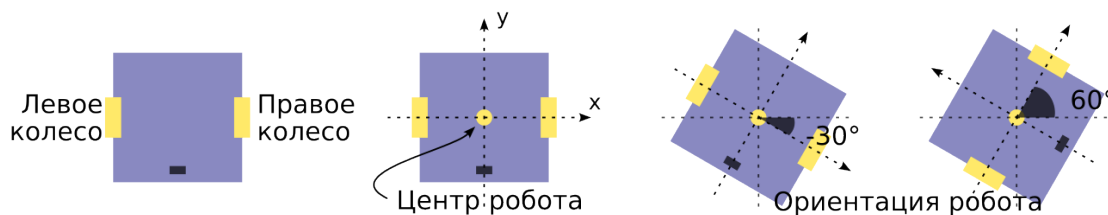


Рис. 2.4.6

Представим, что таких роботов много, они двигаются по полю, имеют разные координаты x, y центра робота и разную ориентацию θ .

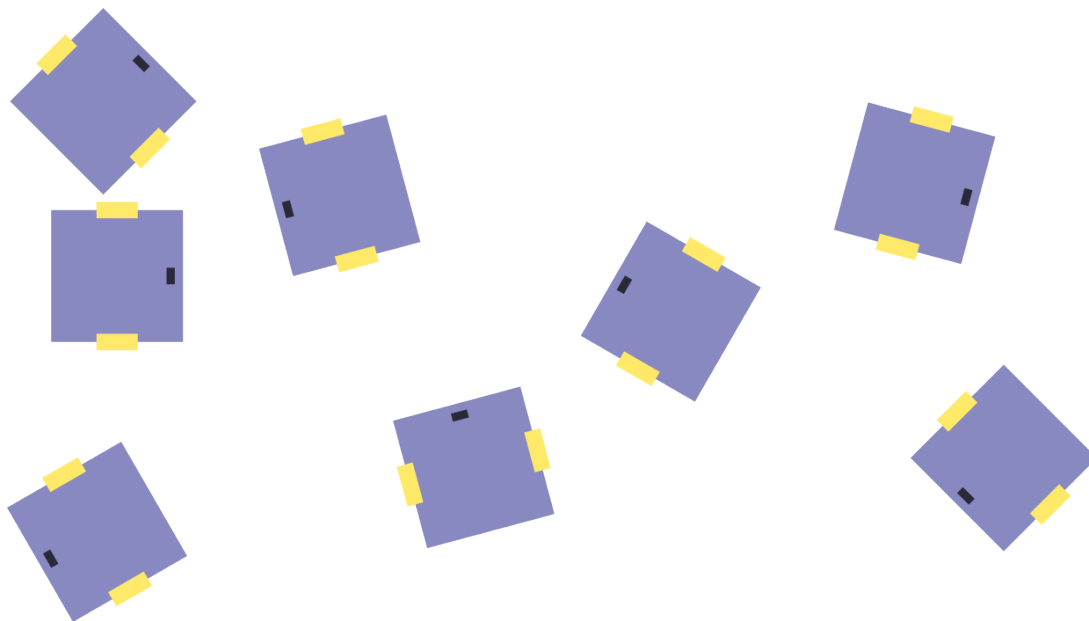


Рис. 2.4.7

Задание. По изображению необходимо определить, какие из роботов столкнулись, и вывести центры (x, y) всех роботов, которые не столкнулись ни с одним другим роботом. Считается, что роботы столкнулись, если расстояние между ними менее двух пикселей, либо их изображения пересекаются. Кроме того, известно, что хотя бы один робот на изображении не пересекается ни с каким другим роботом, а также, что площадь общей части двух пересекающихся роботов не превышает 20% от площади робота. На изображении может находиться от 0 до 50 роботов.

В данной задаче рекомендуется использовать исключительно язык Python и компилятор/окружение (make) Python 3.12.3, где будут доступны для использования Python-пакеты `scipy`, `sklearn`, `numpy` и `cv2`.

Изображения предоставляются в сериализованном виде. Пример сериализации в Python:

Python

```
1 import cv2
2 import base64
3 import numpy as np
4
5 buffer = cv2.imencode('.jpg', img)
6 jpgBufferText = base64.b64encode(buffer)
7 imageString = jpgBufferText.decode('utf-8')
```

Пример десериализации. Здесь обратно преобразуем изображение из строки:

Python

```
1 import cv2
2 import base64
3 import numpy as np
4
5 buffer = base64.b64decode(imageString)
6 array = np.frombuffer(buffer, dtype=np.uint8)
7 img = cv2.imdecode(array, flags=1)
```

Формат входных данных

- В первой строке заданы два числа N и M — ширина и высота изображения.
- Во второй строке передается преобразованное к строке изображение.

Значения элементов массива изображения лежат в диапазоне от 0 до 255 и имеют тип `uint8`.

Обратите внимание, что каждая ячейка изображения в OpenCV хранит три числа: B, G, R . Соответственно, $\text{img}[0, 0] = [B, G, R]$. В качестве положительного направления осей используйте направление осей OpenCV (ось x смотрит вправо, ось y — вниз) — положительный угол отсчитывается по часовой стрелке.

Формат выходных данных

- В первой строке поступает количество роботов $0 \leq n \leq 50$ на изображении, которые не столкнулись с другими роботами.
- Далее в отдельных строках поступают координаты центра каждого такого робота, разделенные пробелом. Координаты центра должны быть целыми числами.

Примеры

Пример входных данных приведен в файле https://disk.yandex.ru/d/xhgXxuKfe7QhEQ/input_problem_7.txt

Стандартный вывод
4
344 345
340 225
114 168
249 154

Решение

1. Преобразовать изображение к двум цветам так, чтобы черный цвет означал, что этот пиксель является частью какого-то робота, а белый цвет — что пиксель не является частью какого-либо робота.
2. Применить операцию эрозии к изображению на два пикселя, чтобы заполнить пространство между роботами, которые находятся на расстоянии до двух пикселей включительно.
3. Определить все контуры на изображении с помощью `canny`.
4. Посчитать площадь каждого контура, если все роботы на изображении одинаковы, и гарантируется, что есть робот, который ни с кем не столкнулся, то контур с наименьшей площадью и есть такой робот.
5. Найти другие контуры, площади которых превышают минимальную площадь не более, чем на 80%.
6. Определить центр квадрата каждого подходящего робота с помощью поиска контуров `canny` и вывести ответ.

Реализация описанного алгоритма на языке программирования Python представлена в файле https://disk.yandex.ru/d/xhgXxuKfe7QhEQ/solution_7.py.

Задача 2.4.8. Расстановка грузов (16 баллов)

Темы: компьютерное зрение, геометрия, программирование.

Ограничение

TL — 20 с, ML — 64 МБ.

Условие

По конвейеру движется кубическая деталь в случайной ориентации. Рядом с конвейером работает робот-манипулятор, и ему необходимо следить, чтобы деталь имела заданную ориентацию (рис. 2.4.8).

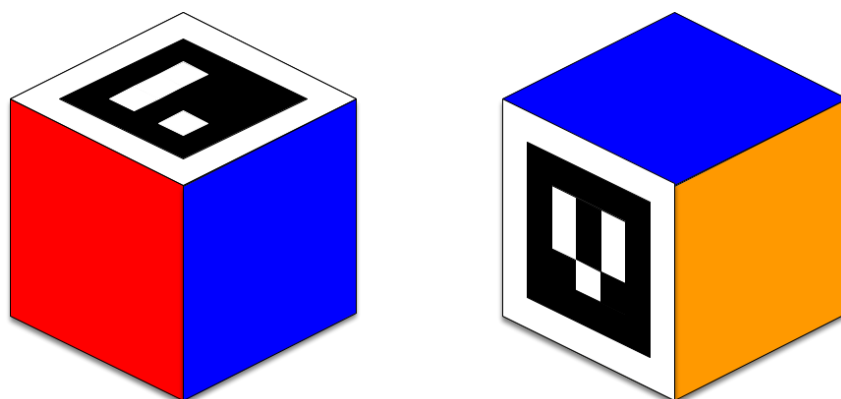


Рис. 2.4.8. Исходная ориентация — слева; заданная ориентация — справа

Робот должен повернуть деталь, если ее исходная ориентация будет отличаться от заданной. В качестве обратной связи используется камера, расположенная на конвейере.

Внешний вид детали можно представить в виде развертки (так как деталь имеет кубическую форму), причем каждая из граней может быть раскрашена в сплошной цвет, либо содержать нанесенную метку (ArUco-маркер). Пример развертки представлен на рис. 2.4.9 на левом изображении. На правом изображении представлен порядок граней на развертке (он понадобится позже).

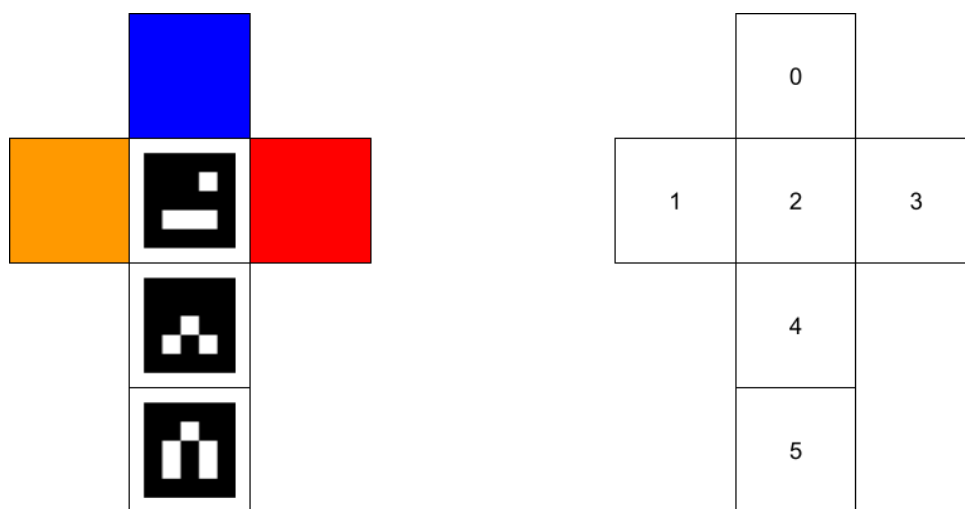


Рис. 2.4.9

Вращение детали осуществляется вокруг осей на 90° , как показано на рис. 2.4.10. Набор возможных команд ограничивается шестью командами и обозначает ось вращения и направление («+X», «-X», «+Y», «-Y», «+Z», «-Z»).

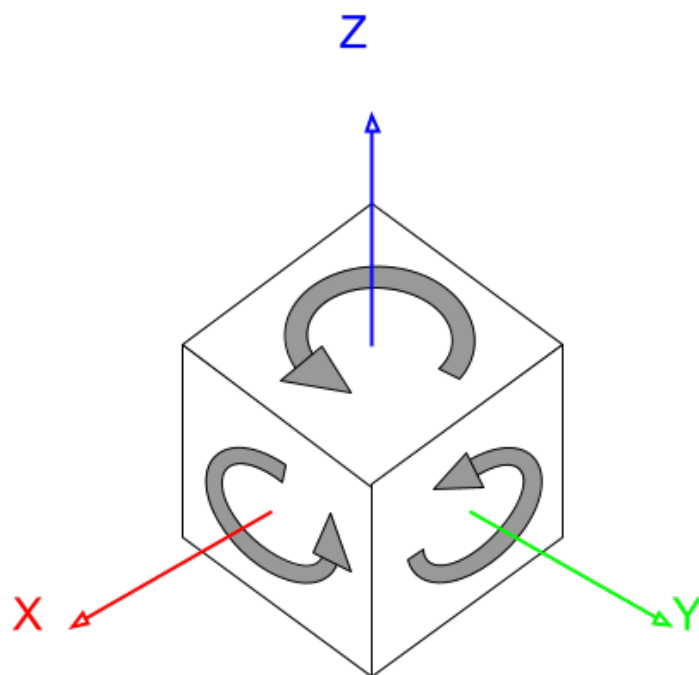
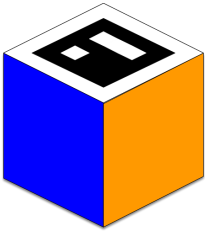
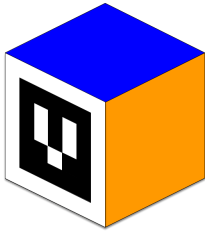
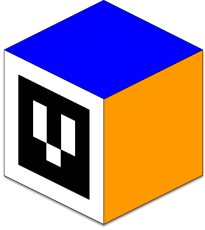
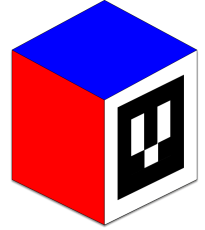
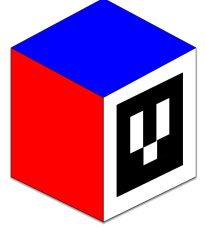
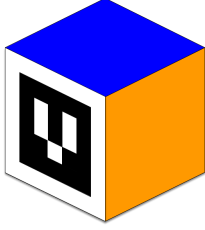


Рис. 2.4.10

Примеры вращений представлено в таблице 2.4.1.

Таблица 2.4.1. Примеры вращений

Команда	До	После
«+X»		
«-X»		
«+Y»		

Команда	До	После
«-Y»		
«+Z»		
«-Z»		

Задание. Требуется на основе исходного и заданного изображений определить набор команд, при помощи которых робот-манипулятор сможет переложить деталь в заданную ориентацию. Максимально разрешенное количество команд равно 30.

Формат входных данных

- В первой строке задается развертка детали в виде массива, содержащего шесть элементов. Номер элемента соответствует порядковому номеру грани развертки, которые были представлены ранее. Если грань содержит сплошной цвет, то она будет описываться словарем, содержащим ключ `color` и значение в виде массива чисел `int`, задающие цвет согласно цветовой модели RGB, например, `{"color": [R, G, B]}`. Если грань содержит метку, то она будет описываться словарем, содержащим ключ `marker_id` и значение `int`, которое соответствует ID значению маркера, например, `{"marker_id": 173}`.
- Вторая строка содержит два целых числа N и M — ширина и высота изображения.
- В следующих трех строках содержатся RGB-каналы исходного изображения соответственно в виде $N \times M$ элементов, разделенных пробелами (сначала идут элементы первой строки, далее — второй и т. д.).
- В следующих трех строках содержатся RGB-каналы заданного изображения соответственно в виде $N \times M$ элементов, разделенных пробелами (сначала идут элементы первой строки, далее — второй и т. д.).

Формат выходных данных

Выходными данными служит одна строка, содержащая в себе команды (описаны ранее в таблице), разделенные пробелами. Всего не более тридцати команд.

Примеры

Пример №1

Стандартный ввод
[{'color': [0, 0, 255]},{ 'marker_id': 173}, ... 640 480 255 255 255 255 ... 255 255 255 255 255 255 ... 255 255 255 255 255 255 ... 255 255 255 255 255 255 ... 255 255 255 255 255 255 ... 255 255 255 255 255 255 ... 255 255
Стандартный вывод
+X +X +Y -Z -Z

Решение

1. Сначала ведется поиск видимых вершин кубов (деталей) на двух изображениях. Для каждой из граней, согласно известной развертке, надо определить или ее цвет, или $AgUco$ -метку, а также ее центр.
2. По координатам центров граней определить, какая грань находится сверху, какая — слева, а какая — справа на каждом изображении, из чего определить текущую и заданную ориентацию куба.
3. Выбрать две смежные грани куба, представляющего желаемую ориентацию, на основе которых будет производиться поиск последовательности команд. Последующие пункты описывают последовательность при выборе левой и правой граней.
4. Убедиться, что грань, которая должна оказаться в левом положении, не находится на верхней или нижней грани. В противном случае с помощью поворотов вокруг оси Y перевести грань в центральное положение (т.е. такое положение, при котором она не находится ни сверху, ни снизу).
5. Окончательно за счет поворотов вокруг оси Z привести грань из предыдущего пункта в левое положение.
6. С помощью поворотов вокруг оси X выставить необходимую грань в правое положение. Заметим, что вместе с тем будет выставлена и верхняя грань в нужное положение.

Реализация описанного алгоритма на языке программирования Python представлена в файле https://disk.yandex.ru/d/xhgXxuKfe7QhEQ/solution_8.py.

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

Задачи второго этапа знакомят участников с одним из основных вызовов робототехники — автономной навигацией мультиагентных робототехнических систем. Подобные задания можно встретить в таких международных соревнованиях, как: RoboCup, Robotex International, Darpa Robotic Challenge. Они максимально приближены к реальности, их развитие будет представлено на заключительном этапе уже на реальных роботах. Аналогичные задачи решаются по отношению к любому автопилоту на автономных складах, в роботах-пылесосах и системе доставки.

Количество участников в команде: 3–4 человека.

Роли:

1. Математик-алгоритмист. Подбор и разработка алгоритмов для программистов: управления и навигации, создания графа занятости, алгоритмов поиска пути в графе, построения траектории и т. д. В приоритете хорошее знание математики, алгоритмов планирования пути и машинного зрения.
2. Программист системы управления и навигации робота. Программирование системы управления роботом, планирования пути и следование по нему. Понимание того, как работает теория графов и как настраиваются регуляторы. В приоритете хорошее знание программирования, алгоритмов планирования пути, теории управления (ПИД-регулятор и др.).
3. Программист машинного зрения. Программирование системы машинного зрения, определение координат объектов на изображении, классификации объектов, определения препятствий. В приоритете хорошее знание программирования, алгоритмов машинного зрения.
4. Капитан команды. Распределение задач, расстановка приоритетов, распределение ресурсов, разрешение споров, проектирование архитектуры решения, программирование высокоуровневой интеллектуальной системы управления, создание связи между системой управления робота, системой навигации и планирования движения и системой машинного зрения. В приоритете хорошее знание программирования, алгоритмов планирования пути, машинного зрения (OpenCV) и теории управления (ПИД-регулятор и др.), также приветствуются хорошие коммуникативные навыки.

Участники должны понимать основы математического анализа, алгебры, геометрии, основы программирования на C/C++ и Python. Им потребуется владение алгоритмами поиска пути, планирования движения, алгоритмов управления, алгоритмов обработки изображения, знание библиотеки OpenCV. Также очень важным аспектом является знание и умение работы в операционной системе Linux.

3.2.1. Командные задачи

Командные задачи второго этапа инженерного тура открыты для решения. Соревнование доступно на платформе Яндекс.Контест:

- 1 блок: <https://contest.yandex.ru/contest/69982/enter/>.
- 2 блок: <https://contest.yandex.ru/contest/69985/enter/>.
- 2 блок: <https://contest.yandex.ru/contest/69987/enter/>.

Задача 3.2.1.1. Планирование движения робота (8 баллов)

Темы: навигация, компьютерное зрение, программирование.

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

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

Ограничение по времени выполнения программы: 100 с.

Ограничение по памяти: 400 Мбайт.

Условие

Рассматривается мобильный робот с дифференциальной платформой и карта местности со статичными препятствиями. Требуется написать программу для планирования пути для посещения целевых точек (в правильном порядке), при этом избегая столкновений с препятствиями. Целевые точки задаются в виде координат на изображении карты местности в системе координат OpenCV (ось x смотрит вправо, ось y вниз), положительный угол отсчитывается против часовой стрелки относительно оси x .

В этой задаче рассматривается только форма робота, без учета его кинематики. Принимается, что в точках найденного пути робот может поворачиваться, а между ними он движется прямолинейно. Считается, что по пути передвигается центр масс маски робота (имеется в виду центр масс, определяемый по изображению маски робота), и вокруг нее происходит вращение робота при повороте на месте. Найденный путь должен проходить через заданные целевые точки.

Для обеспечения безопасного движения по пути требуется, чтобы расстояние от робота до препятствий было не менее трех пикселей. Причем в точках пути робот должен быть способен выполнить полный оборот.

Карта местности, где требуется построить маршрут, и форма робота (вид сверху) задаются в виде черно-белых изображений. Целевые точки задаются в виде координат на изображении карты местности. Порядок прохождения точек имеет значение.

Гарантируется, что:

- при расположении центра масс робота в ключевой точке столкновения с препятствиями отсутствуют, в том числе при выполнении полного оборота;
- путь до каждой целевой точки существует.

На рис. 3.2.1 приведен пример сегментированной карты местности.

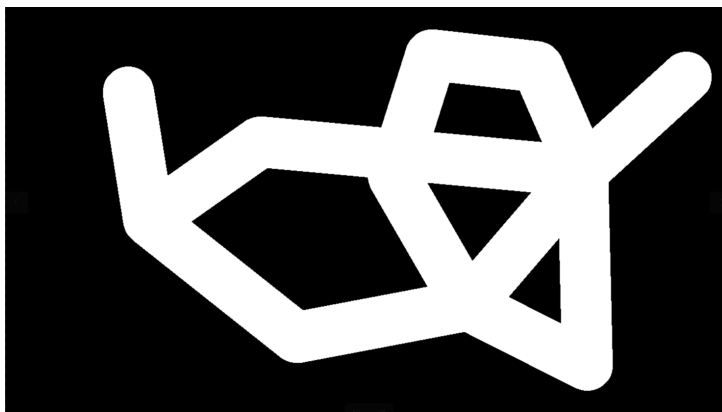


Рис. 3.2.1

Формат входных данных

- Первая строка содержит два числа N и M — ширина и высота изображения сегментированной карты.
- Вторая строка содержит изображение заранее сегментированной карты пространства (черным цветом обозначены препятствия, белым — пространство, где можно двигаться) в виде $N \times M$ элементов, разделенных пробелами (сначала элементы первой строки, далее второй и т. д.).
- Третья строка содержит два числа K и L — ширина и высота изображения маски робота.
- Четвертая строка содержит изображение маски робота (черным цветом обозначен фон, белым цветом — сам робот) в виде $K \times L$ элементов, разделенных пробелами (сначала элементы первой строки, далее — второй и т. д.).
- Пятая строка содержит два целых числа, соответствующие начальному положению робота XU .
- Шестая строка содержит начальную ориентацию робота.
- Следующие строки содержат координаты целевых точек в формате XU , то есть два целых числа, разделенных пробелом. Обратите внимание, что порядок прохождения заданных точек важен!

Формат выходных данных

- Первая строка содержит целое число C — количество точек в найденном пути.
- Следующие строки содержат координаты маршрута в формате XU , то есть два целых числа, разделенных пробелом.

Примеры

Строки, соответствующие изображениям, слишком велики, поэтому вместо примеров ввода и вывода предлагаем скачать файлы, содержащие один из тестов, для тестирования на вашем ПК.

Пример №1

Стандартный ввод

<https://disk.yandex.ru/d/oAF6TMss93XjJg/1/input.1.txt>.

Стандартный вывод

<https://disk.yandex.ru/d/oAF6TMss93XjJg/1/output.1.txt>.

Пример №2

Стандартный ввод

<https://disk.yandex.ru/d/oAF6TMss93XjJg/1/input.2.txt>.

Стандартный вывод

<https://disk.yandex.ru/d/oAF6TMss93XjJg/1/output.2.txt>.

Решение

1. Сначала необходимо произвести эрозию на изображение карты с использованием маски робота в различных ориентациях. Желательно после произвести повторную операцию эрозии с использованием круга с радиусом три пикселя, чтобы избежать возможных проблем с погрешностью отрисовки робота.
2. Далее следует построить граф, включающий целевые точки. Сделать это можно несколькими способами, например, через скелетизацию, сетку с равномерными шагом или через точную/примерную декомпозицию пространства на ячейки.
3. Затем нужно последовательно применить методы поиска пути (например, алгоритм Дейкстры, A*, волновой и т. д.) для построения пути.

Пример программы-решения

<https://disk.yandex.ru/d/oAF6TMss93XjJg/1/solution.py>.

Критерии оценивания

- Баллы начисляются пропорционально последовательно успешно пройденным тестам (до первого неуспешного).
- Максимальное количество баллов зависит от времени отправки решения.

Задача 3.2.1.2. Сегментация изображений (20 баллов)

Тема: компьютерное зрение.

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

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

Ограничение по времени выполнения программы: 100 с.

Ограничение по памяти: 631 Мбайт.

Условие

Рассматривается задача сегментации изображения складского помещения. Пример такого изображения представлен на рис. 3.2.2. На исследуемом изображении можно найти:

- карту-схему складского помещения;
- техническую зону для роботов (обозначена зеленым цветом с заполнением);
- запрещенные зоны, на которых роботу нельзя находиться (обозначены красным цветом со штрихом);
- зоны хранения (обозначены желтым цветом без заполнения, но с ArUco-маркером в центре);
- хранящиеся на складе грузы (обозначены светлыми фигурами).

Необходимо написать программу, которая по полученному изображению будет производить сегментацию складского помещения:

- на техническую зону;
- на зоны со свободными местами хранения;
- на зоны, где робот может передвигаться.

Также требуется определить массив идентификаторов ArUco-маркеров свободных мест и координаты центров масс каждого груза. Форма грузов может быть произвольной, как и цвет, но при этом цвет груза не совпадает с цветами зон, и расположение груза находится в пределах зон хранения. Площадь груза не менее девятой части площади зоны хранения. Если в зоне хранения расположен груз, то эта зона будет считаться запрещенной для перемещения робота. По пустым зонам хранения передвигаться можно. Зоны хранения могут иметь различные размеры. Возможная вариация расцветок зон не отличается в открытой и закрытой части тестов.

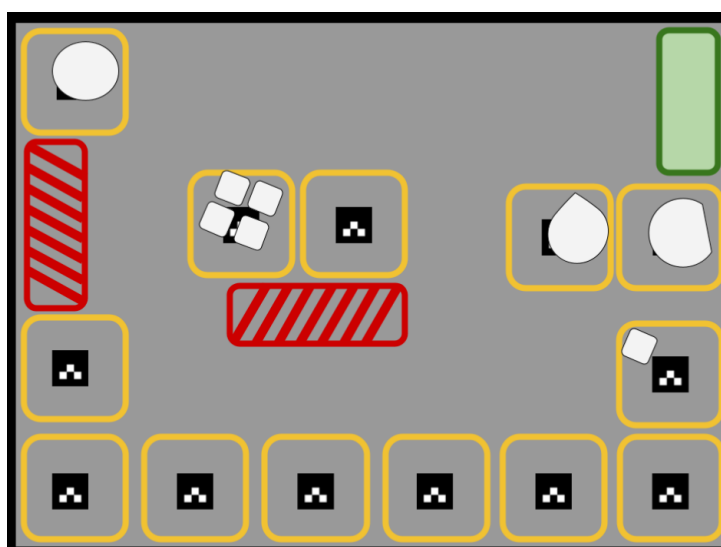


Рис. 3.2.2. Пример карты (примерный)

Более точный пример можно найти в формате ввода.

Формат входных данных

- Первая строка содержит два числа N и M — ширина и высота изображения сегментированной карты.
- Следующие три строки содержат BGR-каналы изображения, которое нужно сегментировать, в виде $N \times M$ элементов, разделенных пробелами (сначала идут элементы первой строки, далее второй и т. д.).

Формат выходных данных

- Первая строка содержит маску технической зоны.
- Вторая строка содержит маску зоны, где робот может передвигаться.
- Третья строка содержит маску мест свободных зон хранения.
- Четвертая строка содержит словарь, в котором в качестве ключей содержатся ID ArUco-маркеров свободных зон хранения, значения которых будут содержать центр соответствующей зоны хранения.
- Пятая строка содержит двумерный массив центров масс каждого из грузов.

Примеры

Поскольку строки, соответствующие изображениям, слишком велики, вместо первого примера ввода и вывода предлагаем скачать файлы, содержащие один из тестов, для тестирования на вашем ПК.

Пример №1

Стандартный ввод

<https://disk.yandex.ru/d/oAF6TMss93XjJg/2/input.txt>.

Стандартный вывод

<https://disk.yandex.ru/d/oAF6TMss93XjJg/2/output.txt>.

Пример №2

Стандартный ввод
1280 720
0 0 0 0 ... 0 0
0 0 0 0 ... 0 0
0 0 0 0 ... 0 0
[[10,10],[30,30],[150,50],[50,150]]
Стандартный вывод
0 0 0 0 ... 0 0
0 0 0 0 ... 0 0
0 0 0 0 ... 0 0
{"27": [110, 250], "51": [660, 351]}

Решение

Переведем изображение из RGB-формата в HSV. На тестовом изображении найдем значения для каждого из цветов, обозначающих конкретный цвет. Затем определим, к каким зонам относятся найденные значения цветов. После этого на каждом новом изображении мы будем каждое значение цвета сопоставлять в пару с ранее найденными значениями. Определив наиболее близкие пары, сможем получить маски разметок.

Для поиска цвета мест хранения можно использовать ArUco-маркеры.

Для технической зоны маска разметки будет соответствовать маске самой зоны. Для запрещенной зоны необходимо найти контуры, после чего отфильтровать контуры, которые находятся внутри других контуров, затем, заполнив эти контуры, получим маску запрещенных зон, которую после дополним занятыми зонами хранения.

Далее, для зон хранения определяем контуры и так же, как и для запрещенных зон, фильтруем контуры, которые находятся внутри других контуров.

Теперь находим контуры грузов. Проводим фильтрацию по площади, чтобы исключить элементы маркеров. Далее, проверяем, внутри каких контуров зон хранения находятся контуры грузов, и при обнаружении такого контура дополняем маску запрещенных зон занятой зоной хранения. Из оставшихся контуров составляем маску свободных зон хранения.

Также, исходя из оставшихся контуров, проходимся по каждому контуру, маской вырезаем зону хранения из основного изображения, переведенного в оттенки серого, после чего, используя пороговую фильтрацию, получаем маску ArUco-маркера, по которой определяем его значение. Зная контуры свободных зон хранения, определяем координаты их центров масс и, найдя все соответствующие ArUco-маркеры, составляем словарь.

Возвращаясь к контурам грузов, для каждого контура находим координату центра масс и заносим их в отдельный массив.

Примечания

При использовании компилятора (make) Python 3.12.3 будут доступны пакеты `scipy`, `numpy` и `cv2`. В качестве маркеров используются ArUco-маркеры 3×3 , которые были сгенерированы, согласно задаче «Определение идентификаторов», представленной в теоретических материалах. Для определения этих маркеров необходимо написать свою функцию, основываясь на том, как они создаются (средствами OpenCV детектировать эти маркеры не получится).

Пример программы-решения

<https://disk.yandex.ru/d/oAF6TMss93XjJg/2/solution.py>.

Критерии оценивания

- Баллы начисляются пропорционально последовательно успешно пройденным тестам (до первого неуспешного).
- Максимальное количество баллов зависит от времени отправки решения.

Задача 3.2.1.3. Траекторное управление (12 баллов)

Темы: навигация, компьютерное зрение, программирование, теория управления, траекторный регулятор.

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

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

Ограничение по времени выполнения программы: 100 с.

Ограничение по памяти: 400 Мбайт.

Условие

В этой задаче рассматривается мобильный робот с дифференциальной платформой, которому необходимо посещать целевые точки в пространстве со статичными препятствиями, избегая столкновений с ними. Напишите программу для расчета требуемых скоростей колес робота с заданной частотой дискретизации, при которых робот посетит все целевые точки, избегая столкновений с препятствиями. Кинематическая модель робота описывается следующими уравнениями:

$$\dot{p} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v \cos \theta \\ -v \sin \theta \\ \omega \end{bmatrix},$$

$$v = \frac{(w_r + w_l)r}{2} \cdot \frac{\pi}{180},$$

$$\omega = \frac{(w_r - w_l)r}{b},$$

где

- r — радиус колес,
- b — расстояние между колесами,
- w_r и w_l — угловые скорости правого и левого колес соответственно (град/с),
- x, y — координаты центра робота относительно неподвижной системы отсчета,
- θ — ориентация робота (град, угол поворота относительно оси x).

Положительное вращение робота соответствует правилу правой руки (против часовой стрелки), отрицательное — по часовой стрелке.

Шаг времени при генерации требуемых угловых скоростей колес должен быть равен 0,01 с. При проверке предоставленного решения дифференциальные уравнения кинематики робота будут решаться с использованием метода Коши (см. задание 2.4.2 инженерного тура отборочного этапа). Робот должен посетить все целевые точки не

более, чем за 100 с. Для обеспечения безопасного движения по пути требуется, чтобы расстояние от робота до препятствий было не менее трех пикселей.

Целевая точка считается пройденной, если центр робота прошел не дальше, чем три пикселя от точки. Порядок прохождения точек имеет значение.

Формат входных данных

- Первая строка содержит два числа N и M — ширина и высота изображения сегментированной карты.
- Вторая строка содержит изображение заранее сегментированной карты пространства (черным цветом обозначены препятствия, белым — пространство, где можно двигаться) в виде $N \times M$ элементов, разделенных пробелами (сначала элементы первой строки, далее — второй и т. д.).
- Третья строка содержит два числа K и L — ширина и высота изображения маски робота.
- Четвертая строка содержит изображение маски робота в виде $K \times L$ элементов, разделенных пробелами (сначала элементы первой строки, далее — второй и т. д.). Робот имеет круглую форму с центром в середине, однако его размеры четко не определены.
- Следующая строка содержит словарь, который будет содержать следующие поля:
 - ◇ `init` — массив из трех чисел, которые обозначают начальное положение и ориентацию робота $[x, y, \theta]$.
 - ◇ `params` — массив, содержащий параметры робота $[r, b, \text{max_speed}]$, где `max_speed` — максимальная угловая скорость колес.
 - ◇ `points` — массив целевых точек в формате $[x, y]$, которые робот должен посетить.

Формат выходных данных

- Первая строка содержит целое число C — количество точек траектории.
- Следующие строки содержат временную метку (время задания скоростей), а также задаваемые угловые скорости для левого и правого колеса в формате $\langle t, \omega_l, \omega_r \rangle$.

Примеры

Строки, соответствующие изображениям, слишком велики, поэтому вместо примеров ввода и вывода предлагаем скачать файлы, содержащие один из тестов, для тестирования на вашем ПК.

Пример №1

Стандартный ввод

<https://disk.yandex.ru/d/oAF6TMss93XjJg/3/input.1.txt>.

Стандартный вывод

<https://disk.yandex.ru/d/oAF6TMss93XjJg/3/output.1.txt>.

Пример №2

Стандартный ввод

<https://disk.yandex.ru/d/oAF6TMss93XjJg/3/input.2.txt>.

Стандартный вывод

<https://disk.yandex.ru/d/oAF6TMss93XjJg/3/output.2.txt>.

Пример №3

Стандартный ввод

<https://disk.yandex.ru/d/oAF6TMss93XjJg/3/input.3.txt>.

Стандартный вывод

<https://disk.yandex.ru/d/oAF6TMss93XjJg/3/output.3.txt>.

Решение

1. Из первого задания имеется требуемый путь робота. По условию, робот движется линейно между точками пути, следовательно, можно реализовать подобное движение и в данной задаче.
2. Разделим движение между точками на два этапа: движение по прямой (когда скорости обоих колес равны), поворот робота (скорости колес противоположны).
3. Для движения по прямым линиям и поворота имеет смысл задавать максимальные угловые скорости колес (по абсолютному значению), причем после движения по прямой линии должна следовать остановка, как и после поворота.
4. В конце остается только реализовать замедление и ускорение робота согласно ограничениям на максимальную и минимальную скорость.

Пример программы-решения

<https://disk.yandex.ru/d/oAF6TMss93XjJg/3/solution.py>.

Критерии оценивания

- Баллы начисляются пропорционально последовательно успешно пройденным тестам (до первого неуспешного).
- Максимальное количество баллов зависит от времени отправки решения.

Задача 3.2.1.4. Траекторное управление группой роботов (20 баллов)

Темы: навигация, компьютерное зрение, программирование, теория управления, траекторный регулятор.

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

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

Ограничение по времени выполнения программы: 100 с.

Ограничение по памяти: 400 Мбайт.

Условие

Задача является продолжением задачи «Траекторное управление» 3.2.1.3. Рассматривается команда мобильных роботов с дифференциальной платформой, которым необходимо посещать целевые точки в пространстве со статичными препятствиями. Напишите программу для расчета требуемых скоростей колес для каждого из роботов с заданной частотой дискретизации, при которых они посетят все целевые точки, избегая столкновений с препятствиями и другими роботами.

Кинематическая модель робота описывается следующими уравнениями:

$$\dot{p} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v \cos \theta \\ -v \sin \theta \\ \omega \end{bmatrix},$$

$$v = \frac{(w_r + w_l)r}{2} \cdot \frac{\pi}{180},$$

$$\omega = \frac{(w_r - w_l)r}{b},$$

где

- r — радиус колес,
- b — расстояние между колесами,
- w_r и w_l — угловые скорости правого и левого колес соответственно (град/с),
- x, y — координаты центра робота относительно неподвижной системы отсчета,
- θ — ориентация робота (град, угол поворота относительно оси x).

Положительное вращение робота соответствует правилу правой руки (против часовой стрелки), отрицательное — по часовой стрелке.

Шаг времени при генерации требуемых угловых скоростей колес должен быть равен 0,01 с. При проверке предоставленного решения дифференциальные уравнения кинематики робота будут решаться с использованием метода Коши (см. задание 2.4.2 инженерного тура отборочного этапа). Роботы должны посетить все целевые точки не более, чем за 100 с.

Целевая точка отмечается как пройденная, если центр робота прошел не дальше, чем два пикселя от точки. Порядок прохождения точек имеет значение. Для обеспечения безопасного движения по пути требуется, чтобы расстояние от робота до препятствий и других роботов было не менее трех пикселей.

Формат входных данных

- Первая строка содержит два числа N и M — ширина и высота изображения сегментированной карты.
- Вторая строка содержит изображение заранее сегментированной карты пространства (черным цветом обозначены препятствия, белым — пространство, где можно двигаться) в виде $N \times M$ элементов, разделенных пробелами (сначала элементы первой строки, далее — второй и т. д.).
- Третья строка содержит два числа K и L — ширина и высота изображения маски робота.
- Четвертая строка содержит изображение маски робота в виде $K \times L$ элементов, разделенных пробелами (сначала элементы первой строки, далее — второй и т. д.). Робот имеет круглую форму с центром в середине, однако его размеры четко не определены.
- Следующая строка содержит словарь, который будет содержать следующие поля:
 - ◇ `init` — массив из трех чисел, которые обозначают начальное положение и ориентацию робота $[x, y, \theta]$.
 - ◇ `params` — массив, содержащий параметры робота $[r, b, \text{max_speed}]$, где `max_speed` — максимальная угловая скорость колес.
 - ◇ `points` — массив целевых точек в формате $[x, y]$, которые робот должен посетить.

Формат выходных данных

- Первая строка содержит целое число C — количество точек траектории.
- Следующие строки содержат временную метку (время задания скоростей), а также задаваемые угловые скорости для левого и правого колеса в формате $\langle t, \omega_l, \omega_r \rangle$.

Примеры

Строки, соответствующие изображениям, слишком велики, поэтому вместо примеров ввода и вывода предлагаем скачать файлы, содержащие один из тестов, для тестирования на вашем ПК.

Пример №1

Стандартный ввод

<https://disk.yandex.ru/d/oAF6TMss93XjJg/4/input.1.txt>.

Стандартный вывод

<https://disk.yandex.ru/d/oAF6TMss93XjJg/4/output.1.txt>.

Пример №2

Стандартный ввод

<https://disk.yandex.ru/d/oAF6TMss93XjJg/4/input.2.txt>.

Стандартный вывод

<https://disk.yandex.ru/d/oAF6TMss93XjJg/4/output.2.txt>.

Пример №3

Стандартный ввод

<https://disk.yandex.ru/d/oAF6TMss93XjJg/4/input.3.txt>.

Стандартный вывод

<https://disk.yandex.ru/d/oAF6TMss93XjJg/4/output.3.txt>.

Решение

Решение задачи будет продолжением решения задания «Траекторное управление» 3.2.1.3. Дополнительно требуется добавить проверку расстояния между роботами и, если расстояние будет меньше определенного значения, то нужно остановить обоих роботов и спланировать их поведение для объезда друг друга. Один из способов реализации данного алгоритма заключается в следующих шагах:

- выбрать одного из роботов и найти недалеко от него положение, которое не будет пересекаться с маршрутом второго робота (проверку пересечения следует производить с учетом габаритов робота);
- переместить первого робота в положение из предыдущего пункта;
- переместить второго робота и, как только он проедет пересечение новых маршрутов, проложить новый путь для первого робота или вернуть его обратно и продолжить движение.

Пример программы-решения

<https://disk.yandex.ru/d/oAF6TMss93XjJg/4/solution.py>.

Критерии оценивания

- Баллы начисляются пропорционально последовательно успешно пройденным тестам (до первого неуспешного).
- Максимальное количество баллов зависит от времени отправки решения.

Задача 3.2.1.5. Детекция роботов и трекинг (20 баллов)

Тема: компьютерное зрение.

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

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

Ограничение по времени выполнения программы: 100 с.

Ограничение по памяти: 917 Мбайт.

Условие

Рассматривается несколько последовательных изображений складского помещения (вид сверху), на которых помимо статических объектов также присутствуют роботы, которые передвигаются от кадра к кадру. Требуется написать программу для детекции и отслеживания перемещений роботов. Для первого кадра положения роботов известны. Для всех остальных кадров необходимо определить положение центра (центра масс маски робота). Всего дано не менее трех кадров в каждом тесте. На рис. 3.2.3 представлен пример одного кадра.

Раскраска роботов заранее неизвестна и является случайной, однако она отличается у разных роботов в рамках одного теста. Форма роботов является произвольной и может совпадать у разных роботов в рамках одного теста. Отдельно отметим, что робот явно виден на фоне, то есть исключено наличие случаев частичного перекрытия или слияния робота с окружением. Все остальные элементы, кроме роботов, на всех кадрах статичны, то есть не изменяются от кадра к кадру, а роботы между последовательными кадрами передвигаются. Кроме того, никакие два робота во время своего движения не сталкивались, то есть отсутствуют кадры, где изображения роботов бы перекрывались.

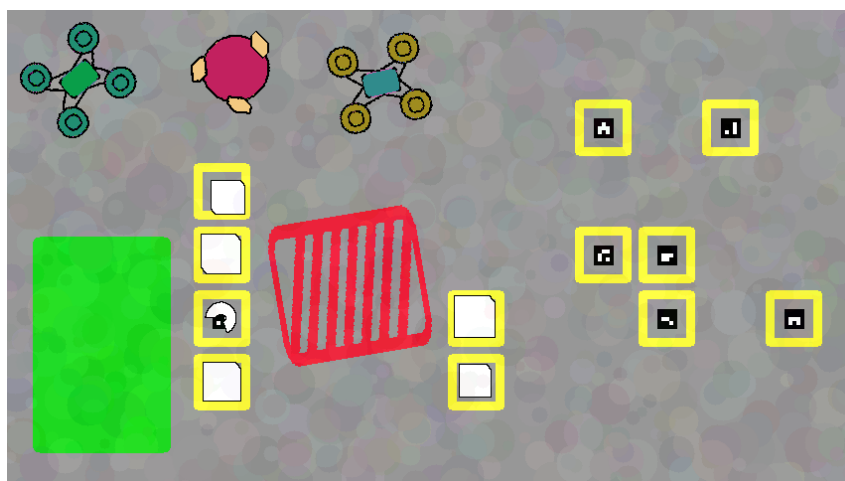


Рис. 3.2.3

Формат входных данных

- Первая строка содержит два целых числа N и M — ширина и высота изображения сегментированной карты в пикселях.
- Следующие строки содержат последовательные изображения карты с роботами в виде $N \times M$ элементов, разделенных пробелами (сначала идут элементы первой строки, далее — второй и т. д.). Каждый кадр представлен тремя строками, в которых последовательно содержатся три канала изображения BGR.
- Последняя строка содержит положения роботов на первом кадре в виде двумерного массива, см. примеры.

Формат выходных данных

- Для каждого робота на отдельной строке должны быть выведены координаты положений роботов, соответствующих порядку последовательных изображений, в виде целых чисел, разделенных пробелами. Например, для трех кадров строка будет иметь вид $\langle X1 Y1 X2 Y2 X3 Y3 \rangle$, и так для каждого робота.
- Порядок вывода, то есть в какой строке координаты какого робота, неважен.
- Координаты рекомендуется округлить до целых чисел, и выводить сначала координату по оси OX , потом OY .

Примечания

В примере ниже приведены три последовательных кадра, на которых изображены четыре робота.

Пример №1

Стандартный ввод
640 480
0 0 0 0 ... 0 0
0 0 0 0 ... 0 0
0 0 0 0 ... 0 0
0 0 0 0 ... 0 0
0 0 0 0 ... 0 0
0 0 0 0 ... 0 0
0 0 0 0 ... 0 0
0 0 0 0 ... 0 0
0 0 0 0 ... 0 0
0 0 0 0 ... 0 0
0 0 0 0 ... 0 0
[[231, 105], [316, 464], [344, 239], [420, 334]]
Стандартный вывод
231 105 150 444 150 444
316 464 50 72 400 410
344 239 151 63 344 237
420 334 321 321 211 312

Поскольку строки, соответствующие изображениям, слишком велики, полные примеры входных данных можно найти по ссылкам ниже. Только один из примеров содержится в тестах.

Примеры

Пример №1

Стандартный ввод

<https://disk.yandex.ru/d/oAF6TMss93XjJg/5/input.1.txt>.

Стандартный вывод

<https://disk.yandex.ru/d/oAF6TMss93XjJg/5/output.1.txt>.

Пример №2

Стандартный ввод

<https://disk.yandex.ru/d/oAF6TMss93XjJg/5/input.2.txt>.

Стандартный вывод

<https://disk.yandex.ru/d/oAF6TMss93XjJg/5/output.2.txt>.

Пример №3

Стандартный ввод

<https://disk.yandex.ru/d/oAF6TMss93XjJg/5/input.3.txt>.

Стандартный вывод

<https://disk.yandex.ru/d/oAF6TMss93XjJg/5/output.3.txt>.

Решение

1. Сначала нужно попиксельно сравнить два последовательных изображения с целью составить маску, на которой выделены положения всех или части роботов.
2. Далее выделяются отдельные контуры, области внутри которых сравниваются с окружением на каждом из сравниваемых изображений. Если на изображении делается вывод, что контур обозначает робота, то в отдельный массив заносится информация с внешним видом робота, а также определяется его местоположение на карте, исходя из центра масс области внутри контура.
3. Для сравнения уже опознанных роботов предлагается сравнивать расцветку найденного робота с записанной, постепенно вращая ее на небольшой фиксированный угол. Если сходство будет более 90%, то можно считать, что робот опознан.
4. Описанный алгоритм необходимо вызывать между каждыми двумя последовательными изображениями, чтобы найти всех роботов, в том числе и тех, что на первом изображении были неподвижны.

Пример программы-решения

<https://disk.yandex.ru/d/oAF6TMss93XjJg/5/solution.py>.

Критерии оценивания

- Баллы начисляются пропорционально последовательно успешно пройденным тестам (до первого неуспешного).
- Максимальное количество баллов зависит от времени отправки решения.

Задача 3.2.1.6. Многоагентное управление (20 баллов)

Тема: мультиагентное управление.

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

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

Ограничение по времени выполнения программы: 100 с.

Ограничение по памяти: 400 Мбайт.

Условие

Рассматривается мультиагентная система из S ($S \leq 7$) одинаковых круглых роботов на омниплатформе (роботы могут двигаться в любом направлении), расположенная на некоторой плоской местности. На местности присутствует одно круглое препятствие.

Задача роботов: из произвольных начальных положений образовать формацию, описываемую графом расстояний между каждым роботом и всеми остальными. Это значит, что для каждого робота заданы желаемые расстояния до всех других роботов.

Формация должна быть образована не более, чем за 1 000 с, при этом расстояния между роботами должны отличаться от желаемых не более, чем на 5%. Гарантируется, что желаемые расстояния между роботами строго больше их диаметра, кроме того, формирование указанной формации возможно.

Напишите одну программу-регулятор, которая будет исполняться на каждом из роботов (на всех роботах будет одновременно запущена одна и та же программа), и формировать сигнал управления, обеспечивающий решение задачи, так что роботы не будут сталкиваться между собой и с препятствием, не будут выезжать за границы карты. Сигналом управления для каждого из роботов являются проекции линейной скорости, то есть модель i -го робота выглядит следующим образом:

$$\begin{cases} \dot{x}_i(t) = u_{x,i}(t), \\ \dot{y}_i(t) = u_{y,i}(t), \end{cases}$$

где $u_{x,i}(t)$ и $u_{y,i}(t)$ — проекции желаемой скорости на соответствующие оси. Решение этих дифференциальных уравнений производится методом Эйлера.

За основу решения задачи возьмите следующий пропорциональный регулятор, в котором специальным образом определен сигнал ошибки, на основе координат роботов и желаемых расстояний. Очевидно, что ошибку можно посчитать, например, как

$$e_d = d_{ij}^2 - m_{ij}^2,$$

где d_{ij} — фактическое расстояние между i -м и j -м роботом, а m_{ij} — желаемое расстояние между ними. Однако из разницы между расстояниями или квадратами расстояний непонятно, в каком направлении следует двигаться на плоскости, поэтому предлагается рассмотреть модифицированную ошибку, которая может быть рассчитана по каждой из осей:

$$\begin{cases} e_{x,ij} = (d_{ij}^2 - m_{ij}^2)(x_j - x_i), \\ e_{y,ij} = (d_{ij}^2 - m_{ij}^2)(y_j - y_i), \end{cases}$$

где x_i и x_j — координаты i -го и j -го роботов по оси OX соответственно, y_i и y_j — координаты i -го и j -го роботов по оси OY соответственно.

Тогда управление для i -го робота с помощью пропорционального регулятора можно рассчитать, как сумму модифицированных ошибок по расстоянию между i -м роботом и всеми остальными, умноженную на некоторый коэффициент:

$$\begin{cases} u_{x,i}(t) = \alpha \sum_{j=1, j \neq i}^S e_{x,ij}, \\ u_{y,i}(t) = \alpha \sum_{j=1, j \neq i}^S e_{y,ij}, \end{cases}$$

где $\alpha > 0$ — коэффициент регулятора, $u_{x,i}(t)$ и $u_{y,i}(t)$ — элементы вектора управления для i -го робота.

Если у всех роботов реализовать такой регулятор, то гарантируется сходимость расстояний между роботами к желаемым значениям. Однако он не учитывает препятствия, возможные столкновения между роботами, не предотвращает выезд робота за пределы местности. Внесите модификацию, чтобы решить эту проблему. Кроме того, можно реализовать любой другой регулятор.

Формат входных данных

Участникам требуется подготовить файл на языке Python, содержащий функцию следующего вида:

Python

```
1 def controller(robot_position, other_robot_positions,
2   ↪ target_distances, map_image, robot_mask, max_velocity, data):
3   ...
4   return velocity, data
```

где

- `robot_position` — положение робота на карте в системе координат OpenCV (`list` с двумя числами типа `float`).
- `other_robot_positions` — положения остальных роботов на карте в системе координат OpenCV (`list` из `list`-ов, которые аналогичны `robot_position`).
- `target_distances` — желаемые расстояния до роботов. Порядок соответствует порядку координат в `other_robot_positions`.
- `map_image` — изображение карты.
- `robot_mask` — маска робота.
- `max_velocity` — максимальная скорость по одной из осей.
- `data` — словарь, куда можно сохранять то, что нужно участнику. Он будет передаваться от цикла к циклу. У каждого робота свой словарь.

Так как данная задача является интерактивной, то простой набор входных и выходных данных отсутствует. Однако примеры задач можно найти в папке по ссылке: <https://disk.yandex.ru/d/oAF6TMss93XjJg/6/input/>.

Здесь:

- первая строка содержит два числа N и M — ширина и высота изображения местности;
- вторая строка содержит изображение местности (черным цветом обозначено препятствие, белым — пространство, где можно двигаться) в виде $N \times M$ элементов, разделенных пробелами (сначала элементы первой строки, далее — второй и т. д.);
- третья строка содержит два числа K и L — ширина и высота изображения маски робота;
- четвертая строка содержит изображение маски робота в виде $K \times L$ элементов, разделенных пробелами (сначала элементы первой строки, далее — второй и т. д.), робот имеет круглую форму с центром в середине, однако его размеры четко не определены;
- следующая строка содержит двумерный массив, представляющий собой `list` начальных положений роботов в системе координат OpenCV, сами начальные положения тоже представляют собой `list`, содержащий два числа типа `float`.

Задав желаемые расстояния между роботами, можно провести тестирование своей программы.

Формат выходных данных

Ожидается, что функция участника вернет рассчитанную скорость для робота в системе координат OpenCV в виде массива с двумя числами типа `float` и словарь `data`.

Решение

Для решения задачи требуется реализовать П-регулятор по ошибке формирования требуемого расстояния от управляемого робота до соседнего. Результирующее управляющее воздействие рассчитывать как сумму управляющих воздействий по каждому соседу.

Основная сложность состоит в корректном формировании сигнала ошибки. Одним из вариантов решения данной проблемы является рассмотрение разницы между желаемым и фактическим расстоянием между роботами, умноженным на разницу координат для учета взаимного расположения роботов.

Для избежания столкновений робота с соседними, необходимо:

- выделить проекцию вектора скорости на вектор, соединяющий его с управляемым роботом для каждого соседа;
- умножить полученную проекцию на коэффициент, зависящий от расстояния между управляемым роботом и соответствующим соседом и корректирующий данную проекцию вектора скорости, тем самым уменьшая ее по достижении определенного порогового значения.

Аналогично необходимо поступить для избежания столкновения с препятствием или границей карты.

Пример программы-решения

<https://disk.yandex.ru/d/oAF6TMss93XjJg/6/solution.py>.

Критерии оценивания

Баллы начисляются пропорционально последовательно успешно пройденным тестам (до первого неуспешного).

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

4.2.1. Информатика. 8–11 классы

Задача 4.2.1.1. Черно-белая таблица (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: 256 Мбайт.

Условие

Робот двигается по таблице 15×15 .

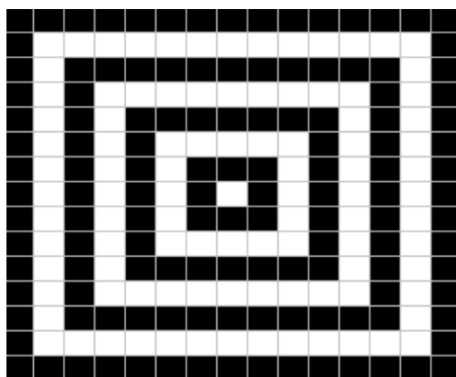


Рис. 4.2.1

Помогите роботу ответить на несколько запросов: по заданным координатам клетки нужно определить ее цвет.

Формат входных данных

В первой строке дано одно число t ($1 \leq t \leq 225$) — количество тестовых запросов.

В следующих t строках дано два целых числа r и c ($1 \leq r, c \leq 15$) — номер строки и столбца. Строки и столбцы нумеруются с левого верхнего угла с единицы.

Формат выходных данных

Выведите t строк — ответ на каждый запрос. Если клетка черная, выведите `black`, иначе — `white`.

Примеры

Пример №1

Стандартный ввод
2 3 5 4 5
Стандартный вывод
black white

Решение

Можно получить решение, встроив информацию о 15×15 сетке в исходный код, но это занимает много времени. Поскольку сетка симметрична относительно центрального квадрата (8-я строка и 8-й столбец), можно решить задачу более простым способом.

В данной сетке квадрат окрашен в черный цвет, если «расстояние» от центрального квадрата до него нечетное, и в белый, если четное. В частности, квадрат на r -й строке и c -м столбце черный тогда и только тогда, когда расстояние Чебышева (шахматное расстояние) от центрального квадрата, $\max(|r - 8|, |c - 8|)$, нечетное (здесь $|x|$ обозначает абсолютное значение x).

Пример программы-решения

Ниже представлено решение на языке Python.

Python

```

1 t = int(input())
2 for _ in range(t):
3     n, m = map(int, input().split())
4     x = min(n, 15 - n + 1)
5     x = min(x, min(m, 15 - m + 1))
6     print("black" if x % 2 == 1 else "white")

```

Задача 4.2.1.2. Робот-манипулятор (18 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: 256 Мбайт.

Условие

Робот находится на прямой в ячейке 1.

Робот последовательно выполняет команды, которые написаны на ячейках. В ячейке под номером i находится команда переместиться в ячейку под номером a_i . Возможно, что $a_i = i$.

Сколько команд выполнит робот, чтобы добраться в ячейку под номером n ?

Формат входных данных

В первой строке задано число n ($2 \leq n \leq 10^5$) — количество ячеек.

Во второй строке задано n чисел a_i ($1 \leq a_i \leq n$) — номер ячейки, в которую переместится робот, если окажется на i -й ячейке.

Формат выходных данных

Выведите единственное число — сколько команд выполнит робот, прежде чем окажется в n -й ячейке. Если робот не сможет оказаться в ячейке под номером n , то выведите -1 .

Примеры

Пример №1

Стандартный ввод
3
3 1 2
Стандартный вывод
1

Пример №2

Стандартный ввод
4
3 4 1 2
Стандартный вывод
-1

Пример №3

Стандартный ввод
5
3 5 4 2 4
Стандартный вывод
4

Решение

1. Инициализация:
 - Текущая позиция: $pos = 1$.
 - Счетчик команд: $count = 0$.
 - Множество посещенных ячеек: $visited = \emptyset$.
 2. Основной цикл:
 - Пока $pos \neq n$ и $pos \notin visited$:
 - ◊ Добавляем pos в $visited$.
 - ◊ Перемещаемся: $pos \leftarrow a_{pos}$.
 - ◊ Увеличиваем счетчик: $count \leftarrow count + 1$.
 3. Проверка результата:
 - Если $pos = n$: возвращаем $count$.
 - Иначе (если $pos \in visited$): возвращаем -1 (робот заиклился).
- Максимальное количество шагов не превосходит n , так как после n шагов робот гарантированно войдет в цикл (по принципу Дирихле).
 - Использование множества $visited$ позволяет детектировать циклы за $\mathcal{O}(1)$ на каждом шаге.
 - Алгоритм имеет временную сложность $\mathcal{O}(n)$ и пространственную сложность $\mathcal{O}(n)$.

Пример программы-решения

Ниже представлено решение на языке Python.

Python

```

1  n = int(input())
2  a = list(map(int, input().split()))
3
4  cur_pos = 1
5  steps = 0
6
7  while cur_pos != n:
8      steps += 1
9      cur_pos = a[cur_pos - 1]
10     if steps > n:
11         steps = -1
12         break
13
14  print(steps)

```

Задача 4.2.1.3. Сборка робота (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: 256 Мбайт.

Условие

Как известно, чтобы собрать робота, надо его составить из трех частей: верхней (головы), средней (туловища) и нижней (ног).

У вас есть n деталей каждого из трех типов. Размер i -й детали, которую можно использовать в качестве верхней части робота, равен a_i . Размер i -й детали, которую можно использовать в качестве средней части робота, равен b_i . Наконец, размер i -й детали, которую можно использовать в качестве нижней части робота, равен c_i .

Для того чтобы построенный робот правильно работал, необходимо соблюсти следующие условия:

- размер средней части должен быть строго больше, чем размер верхней части;
- размер нижней части должен быть строго больше, чем размер средней части.

Вам стало интересно, сколько существует различных способов построить робота, удовлетворяющего описанным условиям. Два способа считаются различными, если хотя бы одна из трех частей робота в них различается.

Формат входных данных

Первая строка содержит одно целое число n ($1 \leq n \leq 100\,000$) — количество деталей каждого из трех типов.

Вторая строка содержит n целых чисел a_1, a_2, \dots, a_n ($1 \leq a_i \leq 10^9$) — размеры деталей, которые можно использовать в качестве верхней части робота.

Третья строка содержит n целых чисел b_1, b_2, \dots, b_n ($1 \leq b_i \leq 10^9$) — размеры деталей, которые можно использовать в качестве средней части робота.

Четвертая строка содержит n целых чисел c_1, c_2, \dots, c_n ($1 \leq c_i \leq 10^9$) — размеры деталей, которые можно использовать в качестве нижней части робота.

Формат выходных данных

Выведите одно целое число — количество различных способов построить робота, удовлетворяющего описанным условиям.

Примеры*Пример №1*

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

Пример №2

Стандартный ввод
3 1 1 1 2 2 2 3 3 3
Стандартный вывод
27

Пример №3

Стандартный ввод
6 3 14 159 2 6 53 58 9 79 323 84 6 2643 383 2 79 50 288
Стандартный вывод
87

Примечания

В первом примере можно составить трех различных роботов:

- верхняя часть: первая деталь, средняя часть: первая деталь, нижняя часть: первая деталь;
- верхняя часть: первая деталь, средняя часть: первая деталь, нижняя часть: втора деталь;
- верхняя часть: первая деталь, средняя часть: втора деталь, нижняя часть: втора деталь.

Решение

Задача заключается в подсчете количества троек (i, j, k) таких, что $A_i < B_j < C_k$. Если зафиксировать j , то искомое количество будет равно произведению количества i , для которых $A_i < B_j$, и количества k , для которых $B_j < C_k$.

Таким образом, для каждого j можно подсчитать количество подходящих i и k , перемножить их и сложить результаты.

Как же подсчитать количество i и k ? Если просто перебирать все элементы массива, то сложность будет $\mathcal{O}(N^2)$, что неприемлемо.

Предварительно отсортируем массивы A и C . Тогда количество i и k можно найти с помощью бинарного поиска. В C++ для этого удобно использовать стандартные функции `lower_bound` и `upper_bound`.

Итоговая временная сложность составит $\mathcal{O}(N \log N)$, что является приемлемым.

Пример программы-решения

Ниже представлено решение на языке Python.

Python

```

1  import bisect
2
3  n = int(input())
4  a = list(map(int, input().split()))
5  b = list(map(int, input().split()))
6  c = list(map(int, input().split()))
7
8  a.sort()
9  b.sort()
10 c.sort()
11
12 cnt = 0
13
14 for i in range(n):
15     p = bisect.bisect_left(a, b[i])
16     q = bisect.bisect_right(c, b[i])
17     q = n - q
18     cnt += p * q
19
20 print(cnt)

```

Задача 4.2.1.4. Две матрицы (22 балла)

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: 256 Мбайт.

Условие

Даны матрица A с H_1 строками и W_1 столбцами, а также матрица B с H_2 строками и W_2 столбцами.

Для всех целочисленных пар (i, j) таких, что $1 \leq i \leq H_1$ и $1 \leq j \leq W_1$, элемент в i -й строке и j -м столбце матрицы A обозначается как $A_{i,j}$.

Для всех целочисленных пар (i, j) таких, что $1 \leq i \leq H_2$ и $1 \leq j \leq W_2$, элемент в i -й строке и j -м столбце матрицы B обозначается как $B_{i,j}$.

Вы можете выполнять следующие операции над матрицей A любое количество раз (возможно, ноль) в любом порядке:

- выбрать произвольную строку матрицы A и удалить ее;
- выбрать произвольный столбец матрицы A и удалить его.

Определите, возможно ли сделать матрицу A равной матрице B .

Формат входных данных

Первая строка содержит два целых числа: H_1 (количество строк матрицы A) и W_1 (количество столбцов матрицы A), где $(1 \leq H_1, W_1 \leq 10)$.

Следующие H_1 строк описывают матрицу A :

- каждая строка содержит W_1 целых чисел, разделенных пробелами;
- каждый элемент матрицы A удовлетворяет условию $1 \leq A_{i,j} \leq 10^9$.

Затем идет строка с двумя целыми числами: H_2 (количество строк матрицы B) и W_2 (количество столбцов матрицы B), где $(1 \leq H_2, W_2 \leq 10)$.

Следующие H_2 строк описывают матрицу B :

- каждая строка содержит W_2 целых чисел, разделенных пробелами;
- каждый элемент матрицы B удовлетворяет условию $1 \leq B_{i,j} \leq 10^9$.

Все значения во входных данных являются целыми числами.

Формат выходных данных

Выведите Yes, если возможно сделать матрицу A равной матрице B ; в противном случае выведите No.

Примеры*Пример №1*

Стандартный ввод
4 5
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
2 3
6 8 9
16 18 19
Стандартный вывод
Yes

Пример №2

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

Стандартный вывод
№

Решение

Вместо того чтобы удалять строки и столбцы по одному, как в исходной постановке задачи, можно рассмотреть следующую операцию, выполняемую ровно один раз: «выбрать некоторые (возможно, ни одной) строки и столбцы исходной матрицы A и удалить выбранные строки и столбцы одновременно».

Например, для следующей матрицы A , приведенной в примере 1,

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

можно выбрать первую строку, третью строку, второй столбец и пятый столбец и удалить их одновременно, чтобы получить следующую матрицу B , приведенную в примере 1:

6	8	9
16	18	19

Чтобы определить, можно ли сделать матрицу A равной матрице B с помощью таких операций, достаточно перебрать все 2^{H_1} способов выбора «удалять ли каждую из H_1 строк» и все 2^{W_1} способов выбора «удалять ли каждый из W_1 столбцов» (всего $2^{(H_1+W_1)}$ способов) и проверить, делает ли какой-либо из них матрицу A равной матрице B .

Перебор всех способов

Для перебора всех $2^{(H_1+W_1)}$ способов выбора строк и столбцов можно использовать метод битового перебора (**bit brute-forcing**), который опишем ниже:

- Каждый способ выбора «удалять ли каждую из H_1 строк» представляется H_1 -битовым двоичным неотрицательным числом X , где:
 - ◊ Для $i = 0, 1, 2, \dots, H_1 - 1$:
 - * Если i -я строка удаляется, то i -й бит (считая с младшего разряда) числа X равен 1.
 - * Если i -я строка не удаляется, то i -й бит равен 0.
- Например, выбор удаления первой и третьей строк из 5 строк может быть представлен числом 00101 в двоичной системе. Таким образом, все 2^{H_1} способов выбора строк могут быть представлены числами 000...00, 000...01, ..., 111...11 в двоичной системе или числами $0, 1, \dots, 2^{H_1} - 1$ в десятичной системе.
- Аналогично способы выбора «удалять ли каждый из W_1 столбцов» могут быть представлены числами $0, 1, \dots, 2^{W_1} - 1$.

Реализация перебора

Таким образом можно перебрать все комбинации способов выбора строк и столбцов с помощью двойного вложенного цикла, где:

- первый цикл перебирает способы выбора строк: $0, 1, \dots, 2^{H_1} - 1$;
- второй цикл перебирает способы выбора столбцов: $0, 1, \dots, 2^{W_1} - 1$.

Пример программы-решения

Ниже представлено решение на языке Python.

Python

```

1  h1, w1 = map(int, input().split())
2  a = [list(map(int, input().split())) for _ in range(h1)]
3
4  h2, w2 = map(int, input().split())
5  b = [list(map(int, input().split())) for _ in range(h2)]
6
7  for i in range(1 << h1):
8      for j in range(1 << w1):
9          hvec = []
10         wvec = []
11         for k in range(1, h1 + 1):
12             if (i & (1 << (k - 1))) == 0:
13                 hvec.append(k)
14         for k in range(1, w1 + 1):
15             if (j & (1 << (k - 1))) == 0:
16                 wvec.append(k)
17         if len(hvec) != h2 or len(wvec) != w2:
18             continue
19
20         match = True
21         for k in range(1, h2 + 1):
22             for l in range(1, w2 + 1):
23                 if a[hvec[k - 1] - 1][wvec[l - 1] - 1] != b[k - 1][l -
24                     ↪ 1]:
25                     match = False
26                     break
27             if not match:
28                 break
29         if match:
30             print("Yes")
31             exit()
32     print("No")

```

Задача 4.2.1.5. Хитрый пароль (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: 256 Мбайт.

Условие

В качестве пароля дан клетчатый прямоугольник, каждый квадрат (1×1) которого раскрашен в какой-то свой цвет.

Чтобы расшифровать пароль, требуется подсчитать количество прямоугольников, раскрашенных ровно в один цвет.

Формат входных данных

В первой строке даны два числа n и m ($1 \leq n, m \leq 500$) — длина и ширина прямоугольника.

В следующих n строках содержится по m символов — строчных латинских букв, отвечающих за цвет.

Формат выходных данных

Вывести пароль, зашифрованный в разноцветном прямоугольнике.

Примеры*Пример №1*

Стандартный ввод
2 2 aa aa
Стандартный вывод
9

Пример №2

Стандартный ввод
2 2 aa ab
Стандартный вывод
6

Решение**Постановка задачи**

Дан прямоугольник размером $n \times m$, где каждая ячейка раскрашена в один из цветов. Требуется подсчитать количество прямоугольников, раскрашенных ровно в один цвет.

Идея решения

Для каждой строки d и каждого столбца l перебираем все возможные прямоугольники, начиная с (d, l) . Для каждого прямоугольника находим минимальную высоту столбцов в диапазоне $[l, r]$ и добавляем ее к ответу.

Переменные

- n — количество строк.
- m — количество столбцов.
- a — матрица размера $n \times m$, содержащая цвета ячеек.
- h — массив высот столбцов, где $h[i]$ — количество подряд идущих ячеек одного цвета сверху в столбце i .
- ans — итоговый ответ (количество одноцветных прямоугольников).

Алгоритм

1. Инициализируем массив h единицами.
2. Для каждой строки d от 0 до $n - 1$:
 - Для каждого столбца l от 0 до $m - 1$:
 - ◊ Инициализируем $min_h = h[l]$ и $r = l$.
 - ◊ Пока $r < m$ и $a[d][r] = a[d][l]$:
 - * Обновляем $min_h = \min(min_h, h[r])$.
 - * Добавляем min_h к ans .
 - * Увеличиваем r на 1.
 - Если $d \neq n - 1$, обновляем массив h :
 - ◊ Для каждого столбца i от 0 до $m - 1$:
 - * Если $a[d][i] \neq a[d + 1][i]$, сбрасываем $h[i] = 1$.
 - * Иначе увеличиваем $h[i]$ на 1.
3. Выводим ans .

Сложность

- Временная сложность: $\mathcal{O}(n \cdot m^2)$.
- Пространственная сложность: $\mathcal{O}(m)$.

Пример программы-решения

Ниже представлено решение на языке Python.

Python

```
1 n, m = map(int, input().split())
2 a = [input().strip() for _ in range(n)]
3 h = [1] * m # Массив высот столбцов
```

```
4
5 ans = 0 # Итоговый ответ
6 for d in range(n):
7     for l in range(m):
8         min_h = h[l] # Минимальная высота в текущем прямоугольнике
9         r = l
10        # Идём вправо, пока цвет ячеек совпадает
11        while r < m and a[d][r] == a[d][l]:
12            min_h = min(min_h, h[r]) # Обновляем минимальную высоту
13            ans += min_h # Добавляем количество прямоугольников
14            r += 1
15        # Обновляем высоты столбцов для следующей строки
16        if d != n - 1:
17            for i in range(m):
18                if a[d][i] != a[d + 1][i]:
19                    h[i] = 1 # Сбрасываем высоту, если цвет меняется
20                else:
21                    h[i] += 1 # Увеличиваем высоту, если цвет совпадает
22        print(ans)
```

4.2.2. Математика. 8–9 классы

Задача 4.2.2.1. (15 баллов)

Темы: проценты, текстовая задача.

Условие

Четверо друзей собираются купить подарок своему товарищу на день рождения, разделив стоимость подарка поровну. Ребята подсчитали расходы и сошлись во мнении, что было бы гораздо лучше, если бы взнос каждого оказался хотя бы на 30% меньше. Какое наименьшее число друзей следует еще позвать для покупки подарка?

Решение

Пусть стоимость подарка равна S руб. Тогда изначально друзья должны скинуться по $S/4$ руб., но хотели бы не более, чем по $\frac{S}{4} - 0,3 \cdot \frac{S}{4} = \frac{0,7S}{4}$ руб.

Обозначим через n количество друзей, которых нужно для этого позвать. Тогда должно выполняться неравенство

$$(4 + n) \cdot \frac{0,7S}{4} \geq S.$$

Отсюда

$$n \geq \frac{4}{0,7} - 4 = \frac{12}{7}.$$

Так как n — целое, то нужно позвать еще хотя бы двух друзей.

Ответ: 2.

Задача 4.2.2.2. (20 баллов)

Темы: делимость, основная теорема арифметики.

Условие

Есть два натуральных числа x и y . Известно, что их произведение равно 66 000. Какое максимальное значение может принимать НОД этих чисел?

Решение

Представим произведение чисел x и y в виде произведения простых множителей:

$$x \cdot y = 66\,000 = 2^4 \cdot 3 \cdot 5^3 \cdot 11.$$

Наибольший общий делитель чисел x и y не может содержать 3 и 11, так как они не могут быть одновременно делителями и x , и y . Число 2 может входить

в НОД максимум во второй степени, число 5 — в первой. Таким образом, наибольшее значение $\text{НОД}(x, y) = 2^2 \cdot 5 = 20$.

Ответ: 20.

Задача 4.2.2.3. (20 баллов)

Темы: неравенство, свойства квадратного корня, свойства квадратичной функции.

Условие

Решить неравенство

$$\sqrt{5-x} - \sqrt{3+x} \geq x^2 + 6x - 7.$$

Решение

Заметим, что левая часть

$$f(x) = \sqrt{5-x} - \sqrt{3+x}$$

определена при $x \in [-3, 5]$ и убывает на области определения, причем $f(1) = 0$. Правая часть неравенства может быть переписана в виде

$$g(x) = (x+3)^2 - 16,$$

откуда видно, что она возрастает при $x \in [-3, 5]$ и $g(1) = 0$. Значит, ответом служит промежуток $[-3, 1]$.

Ответ: $[-3, 1]$.

Задача 4.2.2.4. (20 баллов)

Темы: площадь треугольника, вписанный угол, подобные фигуры.

Условие

A, B, C — точки плоскости, не лежащие на одной прямой. Окружность с диаметром AB пересекает отрезки AC и BC в точках D и E соответственно. Найти площадь четырехугольника $ABED$, если известно, что $AC = 5$, $BC = 8$, а угол между ними $\angle C = 45^\circ$.

Решение

Имеем

$$S_{ABC} = \frac{1}{2} \cdot AC \cdot BC \cdot \sin C = \frac{1}{2} \cdot 5 \cdot 8 \cdot \frac{\sqrt{2}}{2} = 10\sqrt{2}.$$

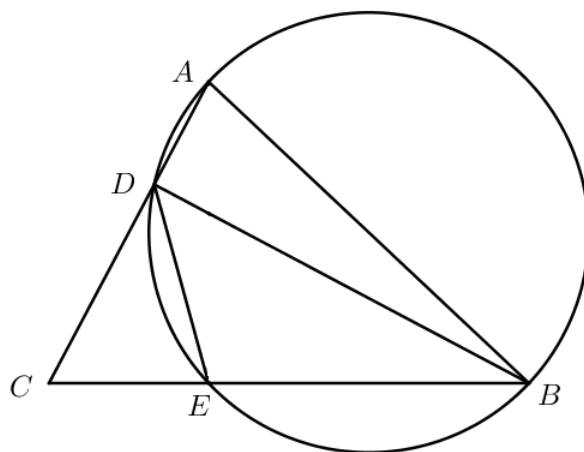


Рис. 4.2.2

Четырехугольник $ABED$ вписанный, значит, $\angle ADE + \angle ABE = 180^\circ$. Сумма смежных углов $\angle ADE + \angle CDE = 180^\circ$. Следовательно, $\angle ABE = \angle CDE$. Тогда треугольники ABC и EDC подобны по двум углам ($\angle ABE = \angle CDE$, угол $\angle C$ — общий).

Проведем отрезок DB . Угол $\angle ADB$ опирается на диаметр, поэтому $\angle ADB = 90^\circ$, значит, смежный с ним $\angle CDB = 90^\circ$.

Треугольник CDB прямоугольный, поэтому

$$\cos C = \frac{CD}{BC} = \cos 45^\circ = \frac{\sqrt{2}}{2}.$$

Таким образом, треугольники EDC и ABC подобны с коэффициентом подобия $\frac{CD}{BC} = \frac{\sqrt{2}}{2}$. Следовательно,

$$\frac{S_{EDC}}{S_{ABC}} = \left(\frac{\sqrt{2}}{2}\right)^2 = \frac{1}{2}.$$

Значит, $S_{EDC} = \frac{1}{2}S_{ABC} = 5\sqrt{2}$, $S_{ABED} = S_{ABC} - S_{EDC} = 5\sqrt{2}$.

Ответ: $5\sqrt{2} \approx 7,07$.

Задача 4.2.2.5. (25 баллов)

Темы: число сочетаний, связность графа.

Условие

Пусть на окружности некоторого радиуса случайным образом выбраны 5 точек. Сколько существует различных способов построить ровно 4 отрезка с концами в этих точках так, чтобы из любой точки можно было попасть в любую другую точку (возможно, проходя через другие точки)?

Решение

Всего существует

$$C_5^2 = 10$$

способов выбрать пару точек для проведения отрезка между ними. Значит, всего существует

$$C_{10}^4 = 210$$

способов выбрать 4 пары точек для соединения. Осталось вычленить плохие соединения. Они бывают двух типов. Первый тип — точка оказалась изолированной, то есть она не соединена ни с одной из оставшихся точек. Способов выбрать эту точку оказывается 5, остальные точки можно соединять произвольно: количество способов выбрать 2 точки из 4 равно

$$C_4^2 = 6,$$

количество способов организовать 4 связи тогда равно

$$C_6^4 = 15,$$

а значит, всего способов организовать такое соединение

$$5 \cdot 15 = 75.$$

Второй тип — ситуация, когда есть ровно 2 связные структуры — 2 точки соединены отрезком, а остальные — какой-то ломаной. Количество способов выбрать 2 точки для соединения отрезком равно

$$C_5^2 = 10,$$

и остается соединить оставшиеся 3 точки при помощи трех отрезков. Итого искомое число равно

$$210 - 75 - 10 = 125.$$

Ответ: 125.

4.2.3. Математика. 10–11 классы

Задача 4.2.3.1. (15 баллов)

Темы: системы счисления, свойства степеней.

Условие

Известно, что

$$11 \dots 1_2 = 100 \dots 0_{1024} - 1,$$

где число в левой части записано в двоичной системе счисления и содержит только единицы, а первое число в правой части записано в 1024-ичной системе счисления и содержит одну единицу и 2025 нулей. Сколько единиц содержит число в левой части?

Решение

Пусть число слева содержит n единиц. Тогда данное в условии уравнение равносильно

$$2^n - 1 = 1024^{2025} - 1.$$

Значит, $2^n = 2^{10 \cdot 2025}$, то есть $n = 20\,250$.

Ответ: 20 250.

Задача 4.2.3.2. (20 баллов)

Темы: неравенство, свойства квадратичной функции, свойства логарифма.

Условие

Решить неравенство

$$\log_3(7 - x) - \log_6(x - 5) \leq x^2 - 10x + 24.$$

Решение

Заметим, что левая часть

$$f(x) = \log_3(7 - x) - \log_6(x - 5)$$

определена при $x \in (5, 7)$ и убывает на области определения, причем $f(6) = 0$. Правая часть неравенства может быть переписана в виде

$$g(x) = (x - 5)^2 - 1,$$

откуда видно, что она возрастает при $x \in (5, 7)$ и $g(6) = 0$. Значит, ответом служит промежуток $[6, 7)$.

Ответ: $[6, 7)$.

Задача 4.2.3.3. (20 баллов)

Темы: задача на работу, система алгебраических уравнений.

Условие

Два менеджера обрабатывают заявки клиентов на подключение к интернету. В понедельник вместе они обработали все заявки за 3 ч. Во вторник первый работал 2 ч, а второй — 1,5, в сумме они обработали 36 заявок. В среду первый обработал в пять раз больше заявок, чем второй в понедельник, а второй — в три раза больше, чем первый во вторник. При этом в среду первый работал на 1 ч дольше второго. Сколько заявок в 1 ч обрабатывает каждый менеджер?

Решение

Обозначим через v_1 и v_2 количество заявок в 1 ч, обрабатываемых первым и вторым менеджером соответственно.

Предложение из условия, касающееся вторника, можно записать в виде уравнения так:

$$2v_1 + 1,5v_2 = 36. \quad (4.2.1)$$

В среду первый менеджер обработал $5(3v_2)$ заявок, а второй — $3(2v_1)$. Тогда информация о среде формулируется в виде уравнения следующим образом:

$$\frac{5(3v_2)}{v_1} - \frac{3(2v_1)}{v_2} = 1.$$

Положим $x = \frac{v_2}{v_1}$. Тогда

$$15x - \frac{6}{x} = 1,$$

что равносильно

$$15x^2 - x - 6 = 0.$$

Это уравнение имеет два корня, один из которых отрицательный и не подходит по смыслу, а второй равен $2/3$.

Подставляя $v_2 = \frac{2}{3}v_1$ в уравнение (4.2.1), получаем $v_1 = 12$. Следовательно, $v_2 = 8$.

Ответ: 12 и 8.

Задача 4.2.3.4. (20 баллов)

Темы: правильная призма, площадь проекции, построение сечения.

Условие

В правильной призме $ABCD A_1 B_1 C_1 D_1$ боковые ребра равны $10\sqrt{2}$, а стороны основания — 5. Точка E принадлежит DC , а точка F принадлежит AD , причем $AF = EC = 2$. Точка S — середина ребра DD_1 . Найти площадь сечения призмы, проходящего через точки F и E параллельно прямой SB_1 .

Решение

Сначала построим сечение, площадь которого требуется вычислить.

1. $FE \cap BD = K$.
2. $KL \parallel SB_1$, $L = KL \cap BB_1$, $H_2 = KL \cap HH_1$.
3. Искомая плоскость содержит $FE \parallel AC$, значит, сама параллельна AC .
4. $H_2 \in MN$, $MN \parallel AC$, $M = MN \cap AA_1$, $N = MN \cap CC_1$.
5. $FMLNE$ — искомое сечение.

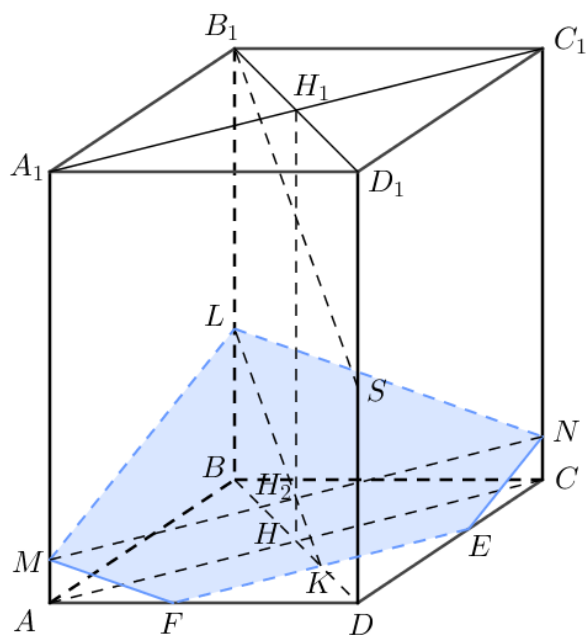


Рис. 4.2.3

Теперь вычислим площадь сечения.

1. Найдем косинус угла между построенной плоскостью и плоскостью основания. FE — ребро двугранного угла. $FE \perp BK$, т. к. $FE \parallel AC$, а $AC \perp BK$ (диагонали квадрата). $FE \perp LK$ (теорема о трех перпендикулярах). Далее

$$\angle(FML, ABC) = \angle(LK, BK) = \angle BKL = \alpha.$$

$B_1D_1 = 5\sqrt{2}$, следовательно, треугольник B_1D_1S равнобедренный,

$$\angle BKL = \angle SB_1D_1 = 45^\circ \Rightarrow \cos \alpha = \frac{\sqrt{2}}{2}.$$

2. $FABCE$ — проекция $FMLNE$ на плоскость основания.

$$AC = 5\sqrt{2}, FE = 3\sqrt{2}, KH = \frac{2}{5} \cdot \frac{1}{2} \cdot 5\sqrt{2} = \sqrt{2}.$$

Найдем

$$S_{FABCE} = S_{ABC} + S_{ACEF} = \frac{25}{2} + \frac{5\sqrt{2} + 3\sqrt{2}}{2} \cdot \sqrt{2} = \frac{41}{2}.$$

- 3.

$$S_{FMLNE} = \frac{S_{FABCE}}{\cos \alpha} = \frac{41 \cdot 2}{2 \cdot \sqrt{2}} = \frac{41\sqrt{2}}{2}.$$

Ответ: $\frac{41\sqrt{2}}{2} \approx 29$.

Задача 4.2.3.5. (25 баллов)

Темы: комбинаторика, правило суммы, число сочетаний.

Условие

Пусть на окружности некоторого радиуса случайным образом выбраны 2025 точек. Сколько существует способов выбрать из них два раза по две различные точки так, чтобы отрезки, соединяющие выбранные две точки соответственно не пересекались? Порядок выбора точек считать неважным.

Решение

Подсчет будем вести следующим образом: искомое число — это разность общего количества возможных отрезков и количества пересекающихся отрезков. Последнее число распадается на два слагаемых: первое отвечает количеству пар пересекающихся отрезков с точкой пересечения во внутренней точке для каждого из отрезков, а второе — количество пар пересекающихся отрезков с точкой пересечения на границе (обоих) отрезков — количество пар отрезков, выходящих из одной точки. Проведем подсчеты.

Не учитывая порядок, всего можем провести

$$C_{2025}^2 = \frac{2025 \cdot 2024}{2} = 2025 \cdot 1012$$

отрезков. Значит, не учитывая порядок, всего существует

$$C_{2025 \cdot 1012}^2$$

пар отрезков. Заметим, что любым четырем точкам из условия соответствует ровно одна конфигурация проведенных отрезков с внутренней точкой пересечения и наоборот. Значит, число пар пересекающихся отрезков с точкой пересечения во внутренней точке равно

$$C_{2025}^4.$$

Осталось вычислить число «смежных» отрезков. Из каждой точки можно провести C_{2024}^2 пар отрезков, а значит, всего таких пар

$$2025 \cdot C_{2024}^2.$$

Итого, искомое число равно

$$C_{2025 \cdot 1012}^2 - C_{2025}^4 - 2025 \cdot C_{2024}^2 = 1\,397\,112\,324\,300.$$

Ответ: 1 397 112 324 300.

4.3. Инженерный тур

4.3.1. Общая информация

На заключительном этапе команды разрабатывают программно-аппаратное решение для управления небольшим, полностью автоматизированным складом площадью 2×2 м.

Участникам предстоит заранее разместить грузы в зонах хранения с учетом известного приблизительного времени их востребования, задействовав при этом двух мобильных роботов и видеопоток с IP-камеры, маркированные ArUco-метки и интерфейсы управления по Wi-Fi/Bluetooth.

Затем, по мере поступления запросов, нужно максимально быстро доставить грузы в зону разгрузки, учитывая реальные нестабильные условия работы за пределами идеальной симуляции.

4.3.2. Легенда задачи

На складе компании МоДист много маленьких роботов, которые могут перемещать только малые и средние МТР. Командам предстоит применить методы мульти-агентного управления и масштабировать склад с маленькими мобильными роботами для транспортировки крупных и тяжелых грузов.

4.3.3. Требования к команде и компетенциям участников

Количество участников в команде: 3–4 человека.

Компетенции, которыми должны обладать члены команды:

- **Специалист AI и обработки изображения:** программирование системы машинного зрения, определение координат объектов на изображении, классификация объектов, определение препятствий; в приоритете хорошее знание программирования, алгоритмов машинного зрения и основ использования нейронных сетей.
- **Специалист по навигации и теории управления:** программирование системы управления роботом, планирования пути и следование по нему; понимание того, как работает теория графов и как настраиваются регуляторы; в приоритете хорошее знание программирования, алгоритмов планирования пути (графы), теории управления (ПИД-регуляторы и др.).
- **Системный администратор/программист высокоуровневых систем управления:** проектирование архитектуры решения и высокоуровневой интеллектуальной системы управления, создание связи между системой управления

робота, системой навигации и планирования движения и системой машинного зрения; в приоритете хорошее знание программирования, алгоритмов планирования пути, машинного зрения (OpenCV) и теории управления (ПИД-регуляторы и др.), а также опыт работы в операционной системе Linux системами и Docker.

4.3.4. Оборудование и программное обеспечение

Схема, приведенная на рис. 4.3.1, отражает взаимодействие всех компонентов системы в следующем порядке:

1. изображение с IP-камеры поступает на сервер, где обрабатывается с помощью алгоритмов компьютерного зрения;
2. сервер, используя данные задания из API, рассчитывает траектории и передает команды на робоплатформы через Raspberry Pi 4B;
3. роботы выполняют действия на поле и отправляют обратную связь серверу, обеспечивая замкнутый цикл управления.

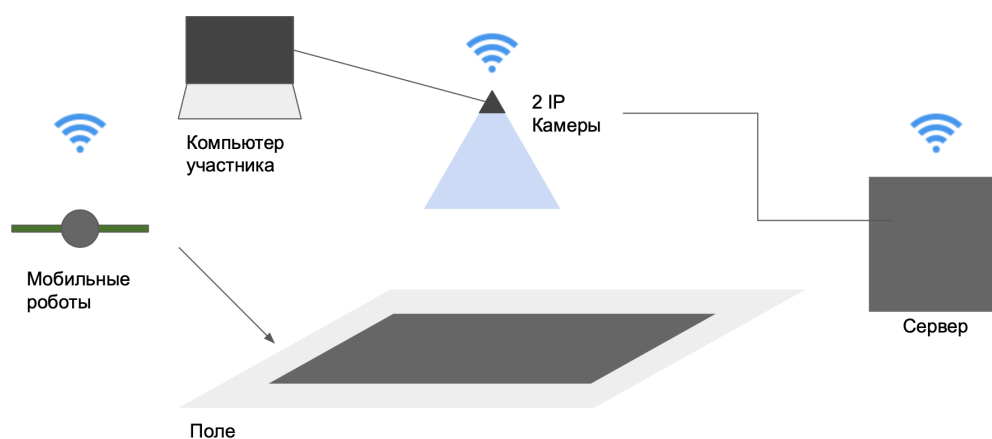


Рис. 4.3.1. Взаимодействие компонентов системы

На рис. 4.3.2 представлен общий вид полигона с размещенными роботами и грузами с IP-камерами; в таблице 4.3.1 приведено описание оснащения полигона.

Таблица 4.3.1. Оснащение полигона

Наименование	Описание	Назначение
Модификация «Роббо Робоплатформы» на ESP32 — https://robbo.ru/products/robbokit/	Мобильная дифференциальная платформа, управляемая по Wi-Fi или Bluetooth; оснащена моторами, поддержкой регулировки напряжения и возможностью подключения внешних модулей	Перемещение и взаимодействие с грузами (захват, доставка, позиционирование)

Наименование	Описание	Назначение
Raspberry Pi 4B — https://www.raspberrypi.com/products/raspberry-pi-4-model-b/	Одноплатный компьютер, выступающий в качестве промежуточного управляющего устройства между роботом и сервером	Управление роботом, обработка видеопотока, отправка и получение данных по сети
IP-камера Axis M3014 — https://www.group-sb.ru/catalog/kupolnye-ip/axis-m3014/	Камера с поддержкой потоковой трансляции (30 Гц), установлена над полем на высоте 2,5 м	Передача изображения поля для сегментации, распознавания объектов и построения карты
Серверный ПК	Компьютер с установленной ОС Linux, подключенный к IP-камере и получающий команды от роботов (доступ по Wi-Fi или SSH)	Запуск пользовательских программ управления, обработка заданий, координация взаимодействия роботов и системы
Программное обеспечение (Python, OpenCV, библиотеки управления)	Средства разработки и библиотеки: Python 3, OpenCV (в том числе модуль агисо), модули управления двигателями	Обработка изображения, сегментация объектов, построение алгоритмов управления, взаимодействие с API задания

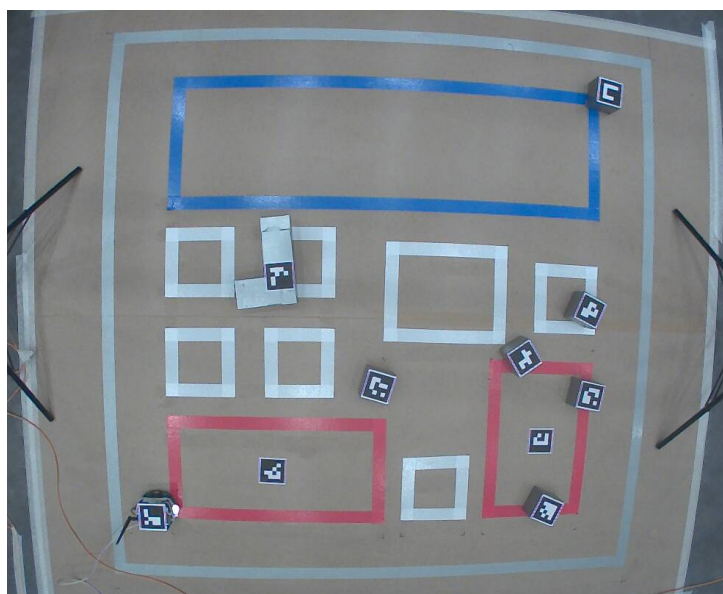


Рис. 4.3.2. Полигон. Общий вид

4.3.5. Описание задачи

В тексте заданий используются следующие термины:

Задача — набор действий и алгоритмов, которые команда должна выполнить (решить).

Этап — подзадача, на которую разбивается задача заключительного этапа.

Попытка — сдача решения задачи в рамках этапа с целью получения баллов (как правило, проверка происходит в установленные временные слоты).

Тест — проверка решения во время попытки при определенной конфигурации грузов и роботов. Другими словами, допускается добавление/удаление/перемещение груза на поле, а также перемещение роботов, после чего участник запускает решение задачи. Во время попытки может проводиться несколько тестов, при которых жюри изменяет условия на поле, а участник перезапускает решение.

На заключительном этапе задачи второго этапа объединены в одну, актуальную для складской логистики задачу, которая сфокусирована на апробации полученных решений в реальных условиях на реальных роботах.

Команды должны написать программу для управления небольшим, полностью автоматизированным складом. Управляя двумя мобильными роботами, нужно расставить грузы в места хранения до их востребования, а после востребования как можно быстрее перевезти их в зону разгрузки.

Ориентировочное время востребования считается известным и необходимо для оптимального расположения грузов в зонах хранения, поскольку важно выдать их как можно быстрее с момента востребования.

Этап 1. Базовое управление роботом и сегментация карты (10 баллов)

Содержание:

- Описание подзадачи.
- Ключевые результаты выполнения подзадачи.
- Возможные проблемы при ее решении.
- Что дает для решения общей задачи? Что происходит, если команда не справилась с решением?

Команде требуется написать программу, осуществляющую сегментацию/детекцию зон и роботов на поле, а также управление роботом в пределах поля. Нужно получить изображение с IP-камеры по соответствующему адресу от организаторов (пример см. ниже).

Python

```

1 import cv2 as cv
2 video_stream =
  ↪ cv.VideoCapture('http://root:admin@10.128.73.50/mjpg/video.mjpg')
3 while True:
4     _, frame = video_stream.read()
5     video_stream.release()

```

В части сегментации необходимо продемонстрировать судьям обработанное изображение с камеры, на котором выделены контуры всех зон, грузов и роботов. Для зон нужно добавить подпись или обозначения (для однозначной интерпретацией жюри) с их ID (при наличии). Контуры грузов и роботов следует обозначить разными цветами (в зависимости от типа выделенного объекта: зона, робот, груз) и подписать их ID (пример — см. рис. 4.3.3–4.3.4).

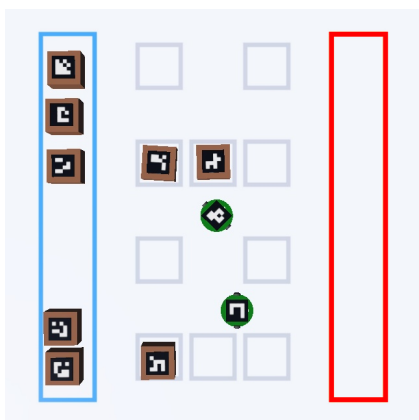


Рис. 4.3.3. Пример сегментации поля:
начальное изображение поля

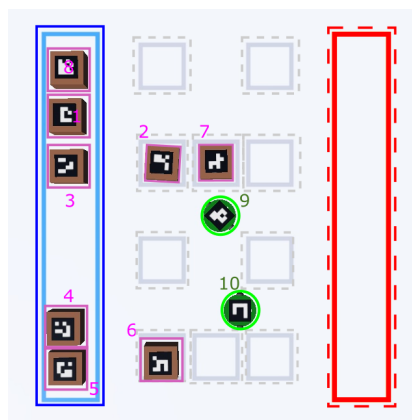


Рис. 4.3.4. Пример сегментации поля:
сегментация зон, роботов и грузов

В части управления роботом команда получает задание на движение к грузу с заданным (целевым) ArUco-маркером по ID. При этом груз будет считаться достигнутым, как только робот произведет касание.

Касанием считается непосредственный контакт робота с грузом, однако груз не должен сдвинуться более, чем на половину наименьшей стороны (например, для груза 10×10 см допустимый сдвиг — 5 см).

Для получения задания необходимо добавить следующие строки в программу управления.

Python

```
1 task = Task()
2 task.start()
3 task.getTask()
```

Здесь для старта обязательно необходимо вызвать `task.start()`. Функция вернет словарь в виде строки (далее при необходимости задание можно будет получить командой `task.getTask()`):

Python

```
1 [{"cargo_id": 1}, {"cargo_id": 2}, {"cargo_id": 3}]
```

где `"cargo_id": 1` задает id ArUco-маркера груза (в примере `id = 1`). Необходимо найти груз с заданным ID на поле, после чего привести робота к грузу и произвести касание.

Командам предоставляется мобильная дифференциальная платформа с возможностью управления через Wi-Fi или Bluetooth. Участники могут управлять мобильной платформой, задавая напряжение на левом и правом двигателе.

Формат оценивания

Для получения максимального балла нужно успешно пройти все тесты. Предусмотрено начисление баллов за достижение маркеров (робот доехал до груза и произвел касание).

Порядок действий:

1. Команда готовится к запуску робота **не более 5 мин** до начала временного слота. Участники обновляют решение с `git`-репозитория, проверяют изображение с камеры, подключаются к роботу.
2. Судья устанавливает грузы с маркерами и одного–двух роботов в соответствии со спецификацией к тесту.
3. Команда запускает съемку проезда робота на телефон, а также сохраняет видео с IP-камеры в процессе прохождения теста.
4. Члены команды демонстрирует обработанное изображение с камеры, на котором обведены и подписаны все зоны, роботы и грузы (с которыми связаны `AgUco`-маркеры).
5. Команда запускает систему управления роботом, которая должна выполнить `task.getTask()` с соответствующими аргументами и отобразить результат в терминале с **задержкой в 5 с**, чтобы члены жюри сфотографировали задание.
6. Программа должна рассчитать траекторию и начать движение к первому грузу из списка (например: `"cargo_id": 1`).
7. После касания с первым грузом из списка робот должен рассчитать траекторию и начать движение ко второму грузу из списка и так далее.
8. Судьи устанавливают грузы на старые места (из п. 2) и робота в другое начальное положение.
9. Команда заново запускает решение и повторяет пп. 2–9 при наличии следующего теста.
10. Попытка прерывается, если у участника не осталось времени на ее выполнение.

Важно сохранять последовательность касания грузов, поскольку получить максимальный балл за задание возможно только при последовательном касании всех заданных грузов.

Баллы начисляются за каждое касание по достижению, в соответствии с порядком задания.

Тесты

В качестве первого теста команде нужно предъявить сегментацию карты, далее следует продемонстрировать распознавание роботов и грузов, а также управление роботом по камере, для чего необходимо, чтобы робот коснулся заданных грузов в заданном порядке.

Пример разметки поля, разделенного на объекты, представлен на рис. 4.3.3–4.3.4. На поле выделяются:

- зона погрузки,
- зоны разгрузки,
- зоны хранения,
- роботы и грузы (обведены контуром, исходя из их формы).

Цвет обводки (контура) выбран в зависимости от типа объекта поля, и его ориентация совпадает с ориентацией объекта. Линия обводки может иметь любую форму (сплошная, штриховая, пунктирная и т. д).

Помимо целевых грузов, задействованных в тесте (в соответствии с ID), на поле также могут лежать и грузы с другим ID.

Принципы оценивания

- Верная сегментация зон хранения — максимально 1 балл.
- Верная сегментация зон погрузки — максимально 1 балл.
- Верная сегментация зон разгрузки — максимально 1 балл.
- Верная сегментация грузов и роботов — максимально 1 балл.
- Касание суммарно всех грузов за тест — 1,5 балла, то есть за каждый груз по $1,5/n$ балл, где n — количество целевых грузов в словаре с заданием.
- Зоны хранения выделены разными цветами — 0 баллов за верную сегментацию зон хранения.
- Зоны погрузки выделены разными цветами — 0 баллов за верную сегментацию зон погрузки.
- Аналогично в случае сегментации зон разгрузки, грузов и роботов.
- Все зоны выделены одним цветом — 0 баллов за сегментацию в целом.
- Ориентация контура отличается от ориентации объекта более, чем на 30° — 0 баллов за сегментацию соответствующего типа объектов поля.
- Не все объекты поля обведены контуром — 0 баллов за сегментацию соответствующего типа объектов поля.
- Контур не соответствует форме объектов — 0 баллов за сегментацию соответствующего типа объектов поля.
- Контур роботов и грузов отступают от границ объекта более, чем на половину от их минимальной длины — 0 баллов за сегментацию соответствующего типа объектов поля.
- ID объектов не соответствуют реальным значениям — 0 баллов за сегментацию соответствующего типа объектов поля.

Тест прерывается и начинается следующий в случаях, если:

- целевой груз сбит (груз сместился более, чем на половину наименьшей стороны груза);
- произведено более двух **касаний** нецелевых грузов;
- нарушен порядок **касания** грузов;
- осуществлен выезд за пределы поля (когда проекция робота полностью выезжает за пределы линии поля).

За тест начисляется 0 баллов, если:

- не было ни одного **касания** целевого груза, ни одна из зон не сегментирована верно;
- команда исправляет код во время прохождения попытки.

Задача прерывается и оценивание прекращается в случае тайм-аута (10 мин).

Технический параметр задачи: частота обновления изображения с камеры 30 Гц (IP-камера).

Этап 2. Передвижение грузов роботом (10 баллов)

Команда должна написать программу управления роботом для перемещения небольших грузов на основе обратной связи по изображению. Для этого выдается задание на перемещение нескольких грузов, которые обозначаются ArUco-маркерами, расположенными на поле.

Целевые грузы могут располагаться в любой точке, из которой возможно выполнение задания. Местами доставки служат зоны хранения и погрузки. Зоны погрузки определяются конкретно по их ID. Зоны хранения никак не обозначены, конкретная зона не задается, однако в одной зоне должно располагаться не более одного груза.

Чтобы получить информацию о задании, в программу интеллектуального управления мобильной платформой необходимо добавить следующие строки.

Python

```
1 task = Task()
2 task.start()
3 task.getTask()
```

Здесь для старта обязательно необходимо вызвать `task.start()`. Функция вернет словарь в виде строки (далее при необходимости задание можно будет получить командой `task.getTask()`):

Python

```
1 [{"unloading_zone": "1", "cargo_id": 1},
2 {"storage_zone": "", "cargo_id": 2},
3 {"storage_zone": "", "cargo_id": 41},
4 {"unloading_zone": "2", "cargo_id": 25}]
5
```

В случае `"unloading_zone": "1", "cargo_id": 1` задает id ArUco-маркера зоны разгрузки (в примере id = 1) маркера груза. Участнику необходимо сегментировать карту для обнаружения необходимых зон и детектировать все маркеры на поле. Демонстрировать сегментацию на данном этапе не требуется.

В случае `"storage_zone": "", "cargo_id": 2` доставлять груз необходимо в зону хранения. Эти зоны обозначаются лишь цветом, поэтому достаточно перевести груз в любую зону хранения.

Команде предоставляется мобильная дифференциальная платформа с возможностью управления через Wi-Fi или Bluetooth. Участник может управлять мобильной платформой, задавая напряжение на левом и правом двигателе.

Формат оценивания

Для получения полного балла нужно успешно пройти два теста. Предусмотрено начисление баллов за доставку грузов в целевые зоны.

Порядок действий:

1. Команда готовится к запуску робота **не более 5 мин.** Участники обновляют решение с git-репозитория, проверяют изображение с камеры, подключаются к роботу.

2. Судья устанавливает грузы с маркерами и одного–двух роботов в соответствии со спецификацией к тесту.
3. Команда запускает съемку на телефон проезда робота, а также сохраняет видео с IP-камеры в процессе прохождения теста.
4. Команда запускает систему управления роботом, которая должна выполнить `task.getTask()` с соответствующими аргументами и отобразить результат в терминале с **задержкой в 5 с**, чтобы члены жюри сфотографировали задание.
5. После запуска робот должен рассчитать траекторию и начать движение к первому грузу из списка.
6. После касания необходимо рассчитать траекторию движения робота в сцепке с грузом для его перемещения в целевую зону.
7. Робот должен рассчитать траекторию и начать движение ко второму грузу из списка и так далее.
8. Для прохождения следующего теста судья устанавливает грузы с маркерами и одного-двух роботов в соответствии со спецификацией к тесту.
9. Команда заново запускает решение и повторяет пункты при наличии следующего теста.
10. Тест прерывается, и начинается следующий, если робот полностью полностью выехал за пределы поля, включая линию, определяющую ее границы.
11. Попытка прерывается, если у участника не осталось времени на ее выполнение.

Важно сохранять последовательность передвигаемых грузов, поскольку получить максимальный балл за задание возможно только при последовательном перемещении всех грузов.

Баллы начисляются за каждое перемещение груза в целевую зону, в соответствии с порядком задания (см. словарь терминов).

Считается, что груз расположен в зоне, если он полностью располагается внутри нее без перекрывания цветных границ зоны (линий). Если груз перекрывает цветную линию зоны и не выходит за пределы зоны, считается, что он расположен частично успешно (за это вычитаются баллы).

Таблица 4.3.2

				
ОК	не ОК	не ОК	частично ОК	ОК

Принципы оценивания

- Перемещение каждого груза в заданную зону (за каждый груз: $5/n$, где n — количество грузов в тесте) — максимально 5 баллов.
- Расположение груза в зоне, частично перекрывающей цветные границы зоны — половина от балла $5/n$, начисленного за перемещение груза.

- Касание с перемещением нецелевого груза — минус $5/2n$ баллов.
- Не полностью доставленный груз (частично находится за пределами целевой зоны, включая цветовую линию) — 0 баллов за доставку данного груза.

Тест прерывается и начинается следующий в случаях, если:

- целевой груз сбит (груз сместился более, чем на половину наименьшей стороны груза);
- нарушен порядок доставки грузов;
- груз доставлен не в целевую зону, а за ее пределы;
- осуществлен выезд за пределы поля (когда проекция робота полностью выезжает за пределы линии поля).

Участник получает 0 баллов при прохождении теста, если ни один груз не был переставлен в требуемую зону.

Задача прерывается и оценивание прекращается в случае тайм-аута (10 мин).

Технический параметр задачи: частота обновления изображения с камеры 30 Гц (USB-камера).

Этап 3. Организация работы автоматизированного складского помещения (10 баллов)

Третий этап моделирует работу реального склада. По сценарию вначале все грузы располагаются на зоне погрузки, затем программа начинает попытку и получает задание.

В задании содержится информация о той зоне погрузки, в которую нужно привезти груз (после востребования), а также примерное время, когда он будет востребован. Все грузы необходимо расположить в зоны хранения до их востребования. Участники должны разместить их так, чтобы в случае надобности можно было доставить груз как можно быстрее.

Важно! Грузы не должны располагаться вне зон без непосредственного контакта с роботом.

Примерное время востребования отличается от действительного времени востребования на ± 30 с по аналогии с ожидаемым временем прибытия такси. (Оценка этих значений происходит, исходя из усредненных известных данных, и может отличаться как в большую сторону (например, водитель пропустил поворот и пришлось объезжать), так и в меньшую (например, водитель нашел быстрый объезд, который не учитывался при оценке времени прибытия).

Следует иметь в виду, что сервер не будет инициатором сообщения о том, что какой-то из грузов стал востребован — решение этой задачи входит в программу участников (способ проверки указан далее).

Во избежание спорных моментов, при необходимости, учитывается время доставки грузов от момента востребования. Оно рассчитывается от момента востребования (Внимание! Не от того момента, когда программа узнала о востребовании.) до того времени, когда программа участника доложит серверу о том, что груз доставлен (способ будет указан далее).

Для начала задания команде необходимо выполнить следующие строки кода.

Python

```
1 task = Task()
2 task.start()
3 task.getTask()
```

В первую очередь следует вызвать `task.start()`. Для получения задания — `task.getTask()`. Функция вернет словарь в виде строки:

Python

```
1 {"task_token": "EV1saAwSasd12Wweqedl", "cargoes": [
2 {"cargo_id": 1, "time": 125, "loading_zone": 1},
3 {"cargo_id": 2, "time": 452, "loading_zone": 1},
4 {"cargo_id": 5, "time": 223, "loading_zone": 2},
5 ]}
```

где

- `task_id`: токен попытки. Данный токен необходимо указывать при всех последующих вызовах функций класса `Task`, поскольку задание завязано на время, и использование данного токена позволит привязать последующие обращения к попытке.
- `cargoes`: массив словарей, содержащий:
 - ◊ `cargo_id`: ID груза;
 - ◊ `time`: примерное время востребования (может отличаться от действительного на ± 30 с);
 - ◊ `loading_zone`: id зоны погрузки.

Исходя из знания примерного времени востребования и зоны погрузки, необходимо расставить грузы по зонам хранения (на это участнику выделяется определенное время, в котором гарантируется отсутствие востребованных грузов).

Далее члену команды требуется постоянно отправлять запрос на проверку востребованности грузов через функцию `task.getCargoDemand()` с указанием токена попытки `task_token`, полученного из команды `task.start()`. Функция вернет список словарей в виде строки:

Python

```
1 [{"cargo_id": 1, "unloading_zone": 1},
2 {"cargo_id": 2, "unloading_zone": 2}]
```

где

- `cargo_id`: ID груза;
- `unloading_zone`: зона разгрузки, на которой ожидается груз.

Как только груз становится востребованным, его можно доставлять до заданной зоны разгрузки, после чего участнику требуется отправить запрос, что груз был доставлен через функцию `task.cargoDelivered()` с указанием токена попытки `task_token`, робота(-ов) которые доставляли груз, id доставленного груза и id зоны разгрузки.

Важно! На этом этапе возможно наличие большого груза, взаимодействовать

с которым получится только с использованием двух роботов.

Команде предоставляется мобильная дифференциальная платформа с возможностью управления через Wi-Fi или Bluetooth. Член команды может управлять мобильной платформой, задавая напряжение на левом и правом двигателе.

Формат оценивания

Этап состоит из одного теста. Для получения полного балла нужно успешно перевезти все грузы в зону погрузки за 10 мин. Предусмотрено начисление баллов за выдачу груза после его востребования в заданной зоне разгрузки.

Порядок действий:

1. Команда готовится к запуску робота **не более 5 мин.**
2. Судья устанавливает грузы в соответствии с заданием.
3. После запуска, исходя из предварительной информации, роботы должны расставить грузы из зоны разгрузки по зонам хранения. В одной зоне хранения должен находиться лишь один груз.
4. Далее необходимо постоянно поддерживать коммуникацию с серверами, отслеживая статус востребованности грузов.
5. После востребованности груза необходимо доставить его до зоны погрузки.
6. Далее необходимо подтвердить, что груз был доставлен, после чего перейти к п. 5.
7. Тест прерывается, если:
 - робот полностью выехал за проезжую часть (он может двигаться только по проезжей части и остановкам);
 - у участника не осталось времени на его выполнение.

Считается, что груз расположен в зоне, если он полностью располагается внутри зоны без касания цветных границ зоны. Если груз перекрывает цветную линию зоны и не выходит за пределы зоны, считается, что он расположен частично успешно (за это будут вычитаться баллы).

Таблица 4.3.3

				
ОК	не ОК	не ОК	частично ОК	ОК

Принципы оценивания

- Доставка груза в зону хранения (5 / количество грузов) — максимально 5 баллов.
- Доставка груза в зону разгрузки по востребованию (5 / количество грузов) — максимально 5 баллов (при спорных ситуациях время от востребования до доставки учитывается для получения дополнительных баллов).

- Расположение груза в зоне, частично перекрывающей цветные границы зоны — минус 2,5 / количество грузов.
- Ошибочная зона выгрузки и/или ошибочный груз при выгрузке — минус 5 / количество грузов.
- Нарушение последовательности, в которой должны перемещаться грузы — минус 5 / количество грузов.

Участник получает 0 баллов при прохождении теста, если ни один груз не был доставлен в требуемую зону разгрузки при востребовании.

Задача прерывается и оценивание прекращается:

- при пересечении хотя бы одним из роботов внешних границ поля,
- в случае тайм-аута (15 мин).

Этап 4. Защита решений (15 баллов)

На данном этапе команда представляет свое решение перед жюри, демонстрируя наработки, архитектуру системы и алгоритмы, использованные для решения задач инженерного тура. Защита включает:

- презентацию краткого видео с демонстрацией работы системы,
- объяснение кода и алгоритмов,
- ответы на вопросы жюри.

Структура защиты:

1. Презентация команды (до 1 мин). Команда кратко представляет себя и поясняет общую концепцию решения.
2. Демонстрация видео (1–2 мин). Включает нарезку лучших попыток работы системы, позволяющую жюри увидеть алгоритмы в действии. Разрешается комбинировать видеозаписи с экрана и реальные съемки.
3. Объяснение алгоритмов и кода (2–3 мин). Представление кода: через `README.md` в `git`-репозитории, пояснение принципов работы всех алгоритмов с визуальной поддержкой.
4. Ответы на вопросы жюри. Участникам задаются вопросы по коду, математическим алгоритмам и разным аспектам решения:
 - компьютерное зрение,
 - планирование движения,
 - управление роботом,
 - логистика грузов,
 - коммуникация между роботом и компьютером.

4.3.6. Система оценивания

Команды готовят решения для этапов задачи (подзадачи) и сдают его в течение трех дней инженерного тура. Количество попыток не ограничено.

Для успешной сдачи необходимо записаться во временной слот. Каждый слот занимает по 10 мин, между ними выделяется 5 мин на подготовку команды к сдаче этапа. Соответственно каждый этап ограничен интервалом в 10 мин.

В одно время доступно 2 слота. Команда может занимать только каждый четвертый (по времени) слот, при этом возможны исключения, если:

- одна из команд пожертвовала своим слотом,
- слот никто не занял, а его время уже началось.

Важно! За каждой командой закреплено одно из четырех полей. Команда не может выбрать время, в котором уже занято поле.

Критерии оценивания этапов:

1. Баллы за прохождение «Базовое управление роботом и сегментация карты»

Максимальное количество баллов за этап — **10**, при этом:

- команда получает баллы за корректную детекцию, трекинг и сегментацию поля, грузов и робота;
- баллы начисляются в случае корректного движения в сторону груза и его достижения;
- ограничение времени на попытку — 10 мин.

В своем слоте команда последовательно выполняет **5 тестов** (количество запусков при разных условиях), в которых судья переставляет грузы и робота, а участники перезапускают решение. За выполнение каждого теста команда получает максимум **2 балла**.

2. Баллы за прохождение «Передвижение грузов роботом»

Максимальное количество баллов за этап — **10**, при этом:

- команда получает баллы за транспортировку груза в целевую зону;
- команда получает неполный балл, если груз, расположенный в целевой зоне, частично перекрывает цветную линию зоны;
- команда теряет баллы при столкновении;
- рестарт происходит при жестком или множественном столкновении, а также при выезде за пределы поля;
- ограничение времени на попытку — 10 мин.

В своем слоте команда последовательно выполняет **2 теста**. Команда получает баллы за каждый успешный тест и теряет при пересечении запрещенной зоны грузом или роботом, выезде за пределы поля, а также при перемещении неверного груза.

3. Баллы за прохождение «Организация работы автоматизированного складского помещения»

Максимальное количество баллов за этап — **10**, при этом:

- команда получает баллы за транспортировку грузов в зону хранения и целевую зону погрузки при востребованности;
- команда получает неполный балл, если груз, расположенный в целевой зоне, частично перекрывает цветную линию зоны;

- команда теряет баллы при столкновении;
- рестарт происходит при жестком или множественном столкновении, а также при выезде за пределы поля;
- ограничение времени на попытку — 10 мин.
- учитывается время, за которое доставили груз в зону погрузки при его востребованности;

Проводится всего **один тест**, в котором участники получают условия (смотри описание этапов выше и условие задачи) и начинают выполнение этапа.

4. Баллы за прохождение «Защита решений»

Максимальное количество баллов за этап — **15**, при этом:

- команда выступает перед жюри и получает от 0 до 17 баллов в соответствии с критериями ниже;
- если команда набрала больше 15 баллов, то за этап она получает 15 баллов за этап.

Критерии оценивания

Общие критерии, которые применяются ко всем аспектам решения:

- **Гибкость решения:** насколько алгоритмы способны адаптироваться к изменениям, а не зависят от жестко заданных условий (входит в другие пункты).
- **Распределение вклада:** оценивается вклад всех участников, т.е. если весь код написан одним человеком, команда получает 0 баллов за этот пункт; вклад оценивается через коммиты в репозитории команды. **2 балла**
- **Продвинутость решений:** использование современных и оптимальных подходов вместо неоправданно сложных или избыточных алгоритмов (входит в другие пункты).
- **Качество оформления README:** структурированность, наличие схем, разборчивость кода и пояснений. **2 балла**

Частные критерии для каждой отдельной задачи:

- **Компьютерное зрение:**
 - ◇ Применение как стандартных алгоритмов обработки изображений и выделения объектов, так и нестандартных (например, в рамках библиотеки OpenCV). **1 балл**
 - ◇ Алгоритмы работы изображения при разных условиях освещенности. **0,5 балла**
 - ◇ Алгоритмы фильтрации данных и устойчивость к шумам. **0,5 балла**
 - ◇ Алгоритмы обработки изображений. (входит в другие пункты)
 - ◇ Обоснованность выбора алгоритмов (сравнение с другими). **1 балл**
- **Планирование движения:**
 - ◇ Гибкость решения (с точки зрения выбора коэффициентов и расширяемости на двух и более роботов) построения траекторий и обхода препятствий. **1 балл**
 - ◇ Устойчивость алгоритмов навигации с учетом препятствий и мультиточечного перемещения груза. **2 балла**

- ◇ Реализация методов динамического или статического планирования пути. **1 балл**
- ◇ Обоснованность выбора алгоритмов (сравнение с другими). **1 балл**
- **Управление роботом:**
 - ◇ Учет физических параметров системы (скорость, инерция, задержки). **1 балл**
 - ◇ Обоснованность алгоритмов управления мобильной платформой через напряжение двигателей. **1 балл**
- **Логистика грузов:**
 - ◇ Оптимизация расстановки грузов с учетом временных ограничений. **1 балл**
 - ◇ Эффективность (с точки зрения минимизации пути, пройденного роботами) маршрутизации грузов с минимальными задержками. **1 балл**
- **Коммуникация между роботом и компьютером:**
 - ◇ Обоснованность выбора методов коммуникации. **0,5 балла**
 - ◇ Схема коммуникации между агентами (ПК, Raspberry, колесная платформа) в решении. **0,5 балла**

После дедлайна сдачи решений этапа №№ 2 и 3 проводится защита решений для всех команд.

Командный балл рассчитывается как сумма баллов за все этапы плюс балл за защиту решения. (Выбирается максимальный балл из всех попыток сдать этап.)

Например, команда успешно выполнила все три этапа. Первый этап она сдала за две попытки на 8 баллов, т. к. выбирается максимальный балл за все попытки. За второй этап команда получила 6,5, за последний (третий) — 7. Итого: $8 + 6,5 + 7 = 21,5$ балл.

Далее командный балл приводится к 100-балльной системе. Для этого вычисляется коэффициент $k = 100 / (\text{макс. балл из всех команд})$, далее баллы всех команд умножаются на этот коэффициент.

Итоговый индивидуальный балл участника равен: математика $\times 0,2$ + информатика $\times 0,2$ + командный балл $\times 0,6$.

В случае равенства итогового индивидуального балла призеры и победители определяются по наилучшим результатам инженерного тура (с максимальным количеством командного балла).

В случае равенства **командного балла** победителем инженерного тура будет считаться команда, набравшая наибольший балл за третий этап, который рассчитывается по формуле: $\text{балл_за_доставку_груза} / \text{ВД}$, где ВД — время доставки (подробный расчет балла за доставку груза — в условии задачи).

4.3.7. Решение задачи

Компьютерное зрение

Для распознавания зон на поле (погрузки, разгрузки, хранения) используется маскирование по цвету в HSV-пространстве с заранее заданными диапазонами.

Изображение «пустого» поля используется как эталон для определения границ зон.

Контуры зон извлекаются с помощью морфологических операций (открытие, закрытие) и функции `cv.findContours`, затем приводятся к нужному виду и обрезаются по границам поля.

Маска с расположением контуров записывается в файл. При запуске она накладывается на изображение с поля, что позволяет определять контура объектов, когда робот перекрывает контуры объектов.

Для определения координат роботов используется распознавание ArUco-маркеров и вычисление вектора ориентации по положению сторон маркера.

Для каждого найденного робота или груза сохраняется центр и угол поворота. Контур груза восстанавливается по заранее заданным шаблонам (малый и большой) и трансформируются с учетом положения ArUco.

Код разделен на сегменты, обрабатывающие зоны, грузы и роботов, с возможностью настройки масок и параметров через свойства класса.

По ссылке можно найти программный код на Python, который выполняет описанные выше операции: <https://gitlab.com/ovcharov.alex.o/nto2025-solution/-/blob/feature/segmentation/segmentation.py>.

Управление роботом

Для определения текущего положения и ориентации робота используется IP-камера, установленная над полем. Обработка изображения включает выделение ArUco-маркера и вычисление центра робота и угла его ориентации.

Управление основано на кинематической модели робота с дифференциальной кинематикой. Вначале рассчитываются линейная и угловая скорости на основе желаемого и измеренного положения робота (x, y, θ) . Далее из кинематической модели рассчитываются желаемые напряжения двигателей.

Для стабилизации движения к цели применяется ПИД-регулятор по расстоянию и углу между направлением робота и вектором на цель. Это позволяет роботу плавно и точно достигать заданных точек.

Команды управления передаются на робота по Bluetooth или Wi-Fi через промежуточный контроллер Raspberry Pi, обеспечивая обратную связь в реальном времени.

При достижении цели робот останавливается и ждет следующей команды — в рамках одного теста таких команд может быть несколько (как минимум три).

По следующей ссылке можно найти программный код на Python, который выполняет описанные выше операции: <https://gitlab.com/ovcharov.alex.o/nto2025-solution/-/tree/feature/robot-control>.

Планирование движения

Алгоритм планирования учитывает расположение всех объектов на поле:

- грузов,

- роботов,
- занятых и свободных ячеек хранения, распознанных по результатам сегментации.

В начальной фазе задачи все грузы размещаются в зоне погрузки, и необходимо рассчитать оптимальные маршруты для их перемещения в свободные ячейки хранения.

Система компьютерного зрения передает бинарную карту с участками, где можно передвигаться и где запрещено. На карту накладывается дилатация с радиусом, равным размеру робота. Это позволяет планировать маршрут для центра робота и не сталкиваться со стенками.

При помощи RRT строится направленный граф (путь для груза) так, что изменение направления движения меняется на 90° .

При помощи PRM строится маршрут до начальной точки пути для груза, а также маршрут, когда робот должен переехать с одной грани на другую для изменения направления движения груза.

Логистика

Для хранения используется правило: один груз — одна ячейка, при этом предпочтение отдается ближним ячейкам, чтобы минимизировать время будущей доставки.

Планирование логистики происходит с учетом примерного времени востребования груза, позволяя расставлять грузы так, чтобы позже сократить путь к зоне разгрузки.

В процессе выполнения система регулярно опрашивает API на предмет востребованности грузов и перестраивает приоритеты движения на основе обновленных данных.

По следующей ссылке можно найти программный код на Python, который выполняет описанные выше операции: https://gitlab.com/ovcharov.alex.o/n-to2025-solution/-/tree/feature/planning?ref_type=heads.

Команда, которая справилась с заданием лучше всех, успешно реализовала все вышеперечисленные шаги решения, разработав быстрый и надежный алгоритм: <https://disk.360.yandex.ru/d/acdepDCaHocbhw>.

4.3.8. Материалы для подготовки

Математика

- Курс. Суть матанализа. Глава 1: <https://www.youtube.com/watch?v=qd0rzmSGPWg&list=PLVjLpKXnAGLVbrcJdDb0a2RS6MmRCgxJz>.
- Курс. Сущность линейной алгебры. Введение: <https://www.youtube.com/watch?v=RNTYicPvWQ&list=PLVjLpKXnAGLXPaS7FRBjd5yZeXwJxZi12>.

Теория управления и навигация

- Курс. Hardcore Math Materials: <https://www.lektorium.tv/hardcoremath-materials>.
- Курс. Control of Mobile Robots — 1.1 Control of Mobile Robots: https://www.youtube.com/watch?v=aSwCMK96NOw&list=PLp8ijpvp8iCvFDYdcXqqYU5Ibl_a0qwjr.

Машинное зрение

- Тьюториал. OpenCV-Python Tutorials: https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html.
- Курс. Computer Vision — AVT: <https://avt.global/cv>.

Навигация и планирование пути

- Казаков К. А. и др. Обзор современных методов планирования движения.
- Павловский И. Обзор алгоритмов планирования пути: <https://habr.com/ru/post/349044/>.
- Russian Blogs. Обзор методов планирования пути: <https://russianblogs.com/article/8569373974/>.
- PythonRobotics. Path Planning: <https://github.com/AtsushiSakai/PythonRobotics#path-planning>.

5. Критерии определения победителей и призеров

Первый отборочный этап

В первом отборочном этапе участники решали задачи предметного тура по двум предметам: математике 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 45,0 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 9,0 баллов, независимо от уровня.

Заключительный этап

Индивидуальный предметный тур

- математика — 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 29,39 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 (математика) 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 математике (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	78,25 и выше
Призеры	От 59,57 до 77,20

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 (Р2Р). 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».