МАТЕРИАЛЫ ЗАДАНИЙ

Всероссийской междисциплинарной олимпиады школьников
«Национальная технологическая олимпиада»

по профилю
«Технологии виртуальной реальности»

2023/24 учебный год

http://ntcontest.ru

УДК 373.5.016:004.946
ББК 74.263.2
Т38

Авторы:
М.В. Бабушкин, А.А. Баранов, И. А. Зуев, Д.А. Казаков, А.С. Кленин, И.А. Кобец, Д. А. Мерзляков, Е.В. Милованович, Н.А. Серебрянская, А.Ю. Чеботарев, Т.С. Юрова, Т.В. Ян

Т38 Всероссийская междисциплинарная олимпиада школьников 8-11 класса «Национальная технологическая олимпиада». Учебно-методическое пособие
Том 22 **Технологии виртуальной реальности**
—М.: ООО «ВАШ ФОРМАТ», 2024. — 176 с.

ISBN 978-5-00147-612-2

Данное пособие разработано коллективом авторов на основе опыта проведения всероссийской междисциплинарной олимпиады школьников 8-11 класса «Национальная технологическая олимпиада» в 2023/24 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 2023/24 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

ISBN 978-5-00147-612-2

УДК 373.5.016:004.946
ББК 74.263.2

Оглавление

Введение

Национальная технологическая олимпиада

Всероссийская междисциплинарная олимпиада школьников «Национальная технологическая олимпиада» (далее — 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».

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.

Всероссийская междисциплинарная олимпиада школьников 8–11 класса «Национальная технологическая олимпиада» — это командная инженерная Олимпиада, позволяющая школьникам работать в 41-м инженерном направлении. 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 девятом сезоне (2023–24 гг.) самых масштабных в России командных инженерных соревнованиях подали более 141 тысячи школьников и студентов из всех регионов страны и семи зарубежных государств: Азербайджана, Белоруссии, Казахстана, Киргизии, Молдовы, Узбекистана и Черногории. Общий охват олимпиады с 2015 года превысил 660 000 участников. https://journal.kruzhok.org/tpost/pggs3bp7y1-tehnologicheskaya-podgotovka-inzhenernih

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

- 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 и ведущими экспертами, а также найти единомышленников.

Ценности НТО

Национальная технологическая олимпиада — командные инженерные соревнования для школьников и студентов. Особое пространство Олимпиады создают общие ценности и смыслы, которые предлагается разделять всем: участникам, организаторам, наставникам, экспертам.

Основа всей олимпиады — это современное технологическое образование как новый уклад жизни в современном мире. Этот уклад подразумевает доступность качественного образования для каждого заинтересованного человека, возможность постепенно и непрерывно учиться и развиваться, совместно создавать среду, в которой гуманитарное знание и новые технологии взаимно дополняют друг друга. Это идеал будущего общества. Участники Олимпиады уже сейчас попадают в такое будущее.

Как организаторы мы надеемся, что принципы, заложенные в основу НТО, станут общими принципами для всех, кто имеет отношение к Олимпиаде.

Решать прикладные задачи, нацеленные на умножение общественного блага

В соревнованиях и подготовке к ним мы адаптируем реальные задачи современной науки и производства к знаниям и навыкам, которые могут освоить школьники и студенты. Задачи имеют прикладное значение для людей и не оторваны от реальности. Мы стремимся к тому, чтобы участники понимали, для чего нужно решать такие задачи, кому в мире станет лучше, если они будут решаться системно и профессионально. Ценность Олимпиады заключается в том, что здесь можно попробовать себя в этом, и найти единомышленников для решения подобных задач в будущем.

Создавать, а не только потреблять

Создание новых решений мы ставим выше стремления потреблять уже созданное. Создание ценности для других ставим выше поиска личной выгоды. Это не значит, что нужно забыть о себе и самоотверженно посвятить всю свою жизнь делу технологического прогресса. Но творчество всегда приносит большую радость, чем потребление. Это относится и ко всей олимпиаде.

Олимпиада — это общее дело организаторов, партнеров и участников. Способность принимать проблемы олимпиады как свои и пытаться решить их ценнее для творческого человека, чем желание найти недостатки в работе других.

Работать в команде

Способность работать в команде — это не только эффективная стратегия действия в современном мире. Работа в команде не отрицает наличия свободной воли каждого конкретного участника, его значимости и права на собственное мнение. Но в сообща мы стремимся достигнуть общей цели, опираясь на взаимное уважение всех участников, учитывая интересы и слабые и сильные стороны каждого.

Команды формируют целые сообщества, которые имеют сходные цели и ценности и могут очень многое, поскольку сильные горизонтальные связи помогают реализовывать самые дерзкие и амбициозные задачи. Это то, что нужно для технологического развития. Мы заняты построением такого сообщества и надеемся, что вы захотите стать его частью.

Осваивать и ответственно развивать новые технологии

Сообщество Национальной технологической олимпиады — часть Кружкового движения НТИ. Это прежде всего сообщество людей, увлеченных современными технологиями. Нас всех объединяет стремление разобраться в них, создать что-то новое и найти таких же увлеченных единомышленников.

Мы — часть сообщества технологических энтузиастов, и для нас границы возможностей технологий всегда подвижны. Именно поэтому просим не забывать об этике инженера и ученого, ответственности за свои изобретения перед людьми, которых это касается. Творя новое, не навреди!

Играть честно и пробовать себя

Мы признаем, что победа в соревнованиях важна и нужна. Но утверждаем, что для победы не все средства хороши и цель не является оправданием для грязной игры. Победа должна быть заслужена в рамках правил, единых для всех. Человек, который играет честно, не будет списывать, интриговать, подставлять других и заниматься прочей нездоровой конкуренцией.

Человек, который играет честно, — уважает себя, свою команду и соперников. Он принимает правила игры и в заданных рамках доказывает право на победу.

Мы бережем пространство Олимпиады как безопасное для всех участников. Это помогает искать себя, и при этом не бояться пробовать новые задачи, определять свой дальнейший путь, учиться на ошибках и каждый год становиться более сильным и подготовленным.

Быть человеком

Соревнования — это очень сложный и эмоционально насыщенный процесс. Чтобы он приносил радость и пользу всем, мы призываем всех участников вести себя порядочно и думать не только о себе.

Вежливость, эмпатия и забота — вот что сделает процесс комфортным и полезным для всех. Мы ценим уважение труда каждого человека и его позиции, бережное отношение к работе и жизни каждого. И просим отказаться от токсичной оценочной критики — она не решит ваши проблемы, а сделает хуже вам, другому и всей

Олимпиаде в целом.

Человек, который остается человеком, умеет признавать ошибки и отвечать за слова и дела перед другими. Здесь это ценят. Встав перед альтернативой между сиюминутной выгодой, капризом и общей целью соревнования — человек выберет последнее и поможет другим, организаторам и участникам, поддержать эту цель.

Важное замечание. Этот текст — живое выражение смыслов и ценностей Национальной технологической олимпиады. Он будет меняться вместе с развитием нашего сообщества. Авторы с благодарностью примут помощь от всех, кто чувствует сопричастность ценностям и готов включиться в их доработку.

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 направлениям:

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

В 2023/24 учебном году 28 профилей НТО включены в Перечень олимпиад школьников, 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, что 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 старшеклассников проводится в три этапа:

- Первый отборочный этап — заочный индивидуальный. На данном этапе участникам предлагаются задачи по двум предметам, соответствующим тому или

иному профилю, а также задания, формирующие теоретические знания и представления по направлениям выбранных профилей.

- Второй отборочный этап — заочный командный. На данном этапе участникам предлагаются индивидуальные компетентностные и командные задачи, связанные с направлением выбранного профиля.
- Заключительный этап — очный командный. Этап представляет собой очные соревнования длительностью 5–6 дней, куда приезжают команды со всей страны, успешно справившиеся с двумя отборочными этапами, и решают комплексные прикладные инженерные задачи.

Профили НТО 2023/24 учебного года и соответствующий уровень РСОШ

Профили II уровня РСОШ

- Автоматизация бизнес-процессов
- Беспилотные авиационные системы
- Водные робототехнические системы
- Инженерные биологические системы
- Интеллектуальные робототехнические системы
- Нейротехнологии и когнитивные науки
- Технологии беспроводной связи

Профили III уровня РСОШ

- Автономные транспортные системы
- Анализ космических снимков и геопространственных данных
- Аэрокосмические системы
- Большие данные и машинное обучение
- Геномное редактирование
- Интеллектуальные энергетические системы
- Информационная безопасность
- Искусственный интеллект
- Летающая робототехника
- Наносистемы и наноинженерия
- Новые материалы
- Передовые производственные технологии
- Разработка компьютерных игр
- Спутниковые системы
- Технологии виртуальной реальности
- Технологии дополненной реальности
- Технологическое предпринимательство
- Умный город
- Фотоника
- Цифровые технологии в архитектуре
- Ядерные технологии

Профили без уровня РСОШ

- Научная медиакоммуникация
- Программная инженерия в финансовых технологиях
- Современная пищевая инженерия
- Технологическое мейкерство
- Урбанистика
- Цифровое производство в машиностроении
- Цифровой инжиниринг в строительстве
- Цифровые сенсорные системы

Новые профили без уровня РСОШ

- Инфохимия
- Квантовый инжиниринг
- Технологии компьютерного зрения и цифровые сервисы
- Цифровая гидрометеорология
- Цифровое месторождение

Обратите внимание, что в олимпиаде 2024/25 года список профилей, в т.ч. входящих в РСОШ, и уровни РСОШ — могут поменяться.

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

- учащихся технологических кружков, любители инженерных и робототехнических соревнований;
- олимпиадников, которым интересны межпредметные олимпиады;
- фанатов и адептов передовых технологий;
- школьников, участвующих в хакатонах, проектных конкурсах и школах;
- будущих предпринимателей, намеревающихся найти на Олимпиаде единомышленников для будущего стартапа;
- увлекающихся школьников, которые хотят видеть предмет шире учебника.

Познакомить школьников с НТО и ее направлениями, замотивировать принять участие в НТО можно с помощью специальных мероприятий: Урок НТО и Дни НТО. Как педагогу провести Урок НТО, или как в образовательном учреждении организовать День НТО можно познакомиться в методических рекомендациях на сайте НТО. Там же можно выбрать и скачать необходимые уроки и подборки материалов по направлениям https://nti-lesson.ru/.

Участвуя в НТО, школьники получают возможность работать с практикоориентированными задачами в области прорывных технологий, собирать команды единомышленников, включаться в профессиональное экспертное сообщество, а также заработать льготы для поступления в вузы.

У НТО есть площадки подготовки по всей стране, которые занимаются привлечением участников и проводят мероприятия по подготовке к соревнованиям. Они могут быть открыты:

- в организациях общего и дополнительного образования;
- на базе частных кружков в области программирования, робототехники и иных технологий;
- в вузах;
- технопарках

и других организациях.

Каждое образовательное учреждение, ученики которого участвуют в НТО или НТО Junior, может стать площадкой подготовки к олимпиаде, что дает возможность включиться в Кружковое движение НТИ.

На сайте НТО размещены инструкции о том, как организация может стать площадкой подготовки: https://ntcontest.ru/mentors/stat-ploshadkoi/. Условия регистрации и требования к работе площадок подготовки обновляются вместе с развитием олимпиады. Обновленная версия размещается на сайте перед началом нового цикла олимпиады.

Наставники НТО

В НТО большое внимание уделяется работе с наставниками. Наставник НТО оказывает всестороннюю поддержку участникам Олимпиады, помогая решать организационные вопросы и развивать как технические знания и компетенции, так и социальные навыки, связанные с работой в команде.

Наставником может стать любой человек, которому интересно сопровождать участников и помогать им формировать необходимые для решения технологических задач компетенции и готовиться к соревнованиям. Это может быть преподаватель школы или вуза, педагог дополнительного образования, руководитель кружка, эксперт в технологической области, представитель бизнеса и т. п. Если наставнику не хватает собственных знаний, он может привлекать коллег и внешних экспертов и

поддерживать усилия и мотивацию учеников, которые разбирают задачи самостоятельно. На данный момент сообщество наставников НТО включает в себя более 7 тысяч человек.

Главная задача наставника — выстроить комплексную структуру подготовки к Олимпиаде в течение всего учебного года. В области ответственности наставника находится поддержка мотивации участников и помощь в решении возникающих проблем. Не менее важно зафиксировать цели и ожидания от предстоящих соревнований, что поможет оценить прирост профессиональных компетенций, личных и командных навыков за время подготовки.

Примеры организационных задач, которые стоят перед наставником НТО:

- Информирование и работа с мотивацией. На этапе регистрации на Олимпиаду наставник привлекает участников, рассказывая, что такое НТО и какие преимущества она предлагает. Наставнику необходимо разобраться в устройстве НТО, этапах и расписании этапов, а также изучить профили, чтобы помочь каждому ученику выбрать наиболее перспективные и интересные для него направления.
- Формирование программы подготовки. Наставник составляет график подготовки к НТО и следит за его реализацией, руководя процессом подготовки учеников.
- Отслеживание сроков. Наставник следит за сроками проведения этапов НТО и напоминает участникам о необходимости своевременной загрузки решений на платформу.

Примеры задач наставника, связанных с непосредственной подготовкой к соревнованиям:

- Анализ компетенций участников. Наставник вместе с учениками оценивает компетенции, которые необходимы для успешного участия в НТО, выявляет нехватку знаний и навыков и отбирает материалы и задачи, которые ученикам нужно изучить и решить.
- Содержательная подготовка к первому и второму отборочному этапу. Наставник вместе с учениками изучает материалы для подготовки, рекомендованные разработчиками выбранных профилей, а также разбирает и решает задачи НТО прошлых сезонов. Рекомендуется использовать записи вебинаров, материалы и онлайн-курсы профилей.
- Содержательная подготовка к заключительному этапу. Наставник может использовать разборы задач заключительного этапа прошлых лет, а также следить за расписанием подготовительных очных и дистанционных мероприятий и рекомендовать ученикам их посещать.

Примеры задач наставника в области развития социальных навыков, связанных с развитием личной эффективности и взаимодействия с другими участниками:

- Формирование команд. Второй отборочный этап НТО проходит в командном формате. Наставник помогает ученикам сформировать эффективную команду с оптимальным распределением ролей. В ряде случаев он может содействовать в поиске недостающих участников команды, в том числе в других городах и стать наставником такой команды, коммуникация в которой осуществляется через web-сервисы.
- Отслеживание прогресса и анализ полученного опыта. Наставник проводит ре-

флексию прогресса отдельных участников и команды по результатам каждого этапа НТО и после завершения участия в соревнованиях. Это помогает участникам оценить свое движение по траектории соревнований, сильные и слабые стороны, сформулировать, каких компетенций не хватило для более высокого результата и как их можно улучшить в будущем.

- Поддержка и мотивирование участников. Наставник поддерживает интерес учеников к соревнованиям, а также помогает им сохранять высокую мотивацию, что особенно важно, если команда показала результаты хуже, чем ожидалось.
- Выстраивание индивидуальной образовательной траектории. Наставник может помочь ученикам осознанно создать собственную траекторию развития, в том числе вне НТО: подбор обучающих курсов и соревнований, выбор вуза и направления дальнейшего обучения.

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/promo. Курс формирует общие представления о работе наставников в области подготовки участников к инженерным соревнованиям.

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

Наставникам для ведения занятий с учениками предлагаются образовательные программы, разработанные на основе восьмилетнего опыта организации подготовки к НТО. В настоящий момент такие программы представлены по 10-ти передовым технологическим направлениям:

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

и др.

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.

Формируется сообщество наставников НТО. Ежегодно Кружковое движение НТИ проводит Всероссийский конкурс технологических кружков: https://konkurs.kruzhok.org, принять участие в котором может каждый наставник. По итогам конкурса кружки-участники размещаются на Всероссийской карте кружков: https://map.kruzhok.org.

В 2022 году был разработан Навигатор для наставников команд или отдельных участников НТО: https://www.notion.so/bdlv/5a1866975c2744728c2bd8ba80d21ec2.

Навигатор ориентирован на начинающих наставников и помогает погрузиться в работу с НТО. Опытным наставникам Навигатор может быть полезен как сборник важных рекомендаций и статей:

- Смогут ли мои ученики принять участие в НТО.
- Как наставнику зарегистрироваться в НТО.
- Как помочь участникам выбирать профили.
- Что можно успеть сделать, если я и мои ученики начнем участвовать с нового учебного года.
- Как убедить руководство включиться в НТО.
- Что важно знать, начиная подготовку школьников.
- Как организовать подготовку.
- Как проводить рефлексию.
- Как мотивировать участников.
- Как работать с командой участников НТО.

Организаторы Олимпиады также оказывают экспертно-методическую поддержку сообществу наставников. Были разработаны методические рекомендации для наставников: «Технологическая подготовка инженерных команд»: https://journal.kruzhok.org/tpost/pggs3bp7y1-tehnologicheskaya-podgotovka-inzhenernih. Рассмотрены особенности подготовки к 5-ти направлениям:

- Большие данные.
- Машинное обучение.

- Искусственный интеллект.
- Спутниковые системы.
- Летающая робототехника.

Для наставников НТО разработан и постоянно пополняется страница с материалами для профессионального развития: http://clc.to/for-mentor.

Технологии виртуальной реальности

Технологии виртуальной реальности дают возможность человеку погружаться в цифровую среду, абстрагироваться от ограничений реального мира и многократно получать опыт благодаря реалистичным симуляциям в виртуальной реальности.

С апреля 2024 года в НТО создан новый профиль «Виртуальные миры», который объединил три подпрофиля (ранее профили): «Разработка компьютерных игр», «Технологии виртуальной реальности» и «Технологии дополненной реальности». Такой шаг был сделан с целью укрупнения профилей НТО.

Подпрофиль «Технологии виртуальной реальности» посвящен разработке игровых симуляторов с использованием виртуальной реальности. Участникам ежегодно предлагаются задания, требующие различных компетенций. Задания ориентированы как на программирование, так и на дизайн (UI/UX, 3D моделирование с анимацией). Учитывая разницу компетенций, участникам предлагаются различные категории заданий в зависимости от заявляемой роли, при этом участник может попробовать свои силы в обоих направлениях. Подпрофиль проходит в несколько этапов: 2 отборочных в дистанционном формате и заключительный в очном режиме. Для успешной подготовки участников к финальному соревнованию предоставляются материалы, курсы, задания прошлых лет.

Отборочные этапы и подготовительные мероприятия проводятся в течение года, чтобы учащиеся могли качественно подготовиться к решению комплексной проектной задачи заключительного этапа. Подпрофиль включает в себя задачи по двум школьным предметам: информатика и математика. Первый отборочный этап проводится индивидуально в сети Интернет с использованием онлайн-платформы Stepik. Для каждой возрастной группы (8–9 класс или 10–11 класс) предлагается свой набор заданий, отражающий углубленное изучение указанных выше дисциплин и соответствующий предметным компетенциям уровня участников школьных олимпиад.

Второй этап является командным и состоит из задач, призванных помочь участникам подготовиться к комплексному решению задачи заключительного этапа. Предлагаемые задачи рассчитаны на разные роли участников в команде: программист, 2D и 3D художник, геймдизайнер. В период работы над задачами второго этапа каждый участник получает опыт в своей профессиональной направленности. Область виртуальной реальности подразумевает выполнение более 50% задач по 3D моделированию, включая текстурирование и анимирование.

Параллельно с первым и вторым отборочными этапами для участников проводятся учебно-тренировочные сборы (УТС), в том числе, с привлечением региональных ЦМИТ, площадок федеральной сети детских технопарков «Кванториум».

Задача заключительного этапа профиля в 2023–2024 гг. состояла в разработке прототипа VR-приложения, в котором игрок застревает на плоту посреди воды, и ему необходимо добраться до суши, преодолевая различные препятствия на пути, используя механики, позволяющие совершать маневры.

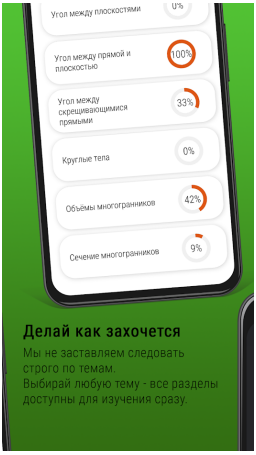
Задача заключительного этапа и критерии оценки разработаны таким образом, чтобы каждая команда стремилась к коллективной работе и могла четко распределить роли и задачи между собой. В качестве ключевых параметров разработки

учитываются качество реализованного функционала, геймдизайнерских решений, а также художественное оформление приложения.

Помимо командной задачи участникам необходимо решить индивидуальные задачи по предметам информатика и математика, которые схожи по темам с задачами первого этапа, но имеют более высокую сложность.

Компетенции, приобретенные в подпрофиле, применяются выпускниками НТО в различных областях: поступление в профильные вузы, разработка игровых приложений виртуальной и дополненной реальности, участие в коммерческих проектах.

Ниже представлено приложение Stearomtry для подготовки к ЕГЭ по стереометрии с использованием AR, разработанное финалистом профиля НТО Дмитрием Мерзляковым, доступно в GooglePlay (https://play.google.com/store/apps/details?id=com.wizard.stearometry).



Работа наставника НТО на первом отборочном этапе

На первом отборочном этапе НТО участникам предлагаются задачи по предметам, соответствующим выбранным профилям. Для подготовки к первому отборочному этапу Олимпиады наставник может использовать следующие рекомендуемые форматы и мероприятия:

- Разбор задач первого отборочного этапа НТО прошлых лет.
- Мини-соревнования по решению задач предметных олимпиад муниципального уровня.
- Углубленные занятия по разделам предметов в соответствии с рекомендациями разработчиков профилей.

Для проверки, самостоятельного решения или проведения мини-соревнований могут использоваться предметные курсы НТО на платформе Stepik. Также возможно привлечение других преподавателей-предметников для проведения занятий в случае, если у наставника недостаточно компетенций в области предметных олимпиад.

Инженерный тур состоит из курса или теоретических материалов, погружающих участников в тематику профиля, и теоретических и практических заданий, как правило связанных с теорией.

Первый отборочный этап

Предметный тур. Информатика и информационные технологии

Первая волна. Задачи 8–11 класса

Задача II.1.1.1. Авиакомпания (9 баллов)

Темы: базы данных.

Условие

Даны фрагменты двух таблиц базы данных некоторой авиакомпании. Исходя из информации данных таблиц, определите, сколько человек вылетели из Москвы в интервале от 12 до 18 часов за 05.07.2023 и 06.07.2023.

Обратите внимание, что в разные даты один и тот же номер рейса может иметь разные пункты вылета и пункты прилета.

Таблица II.1.1: `passengers`

id	first_name	last_name	birth	document	flight_num	flight_date	status
1	Ivan	Ivanov	25.05.1999	**********	104	05.07.2023	True
2	Anna	Smirnova	24.05.2002	**********	104	05.07.2023	False
3	Ekaterina	Kuznetsova	04.02.1996	**********	105	05.07.2023	True
4	Aleksandr	Popov	06.04.1994	**********	103	05.07.2023	True
5	Elena	Vasilieva	03.11.1994	**********	104	05.07.2023	False
6	Sergei	Petrov	25.06.1984	**********	103	05.07.2023	False
7	Daniil	Sokolov	07.12.2000	**********	101	06.07.2023	True
8	Anastasia	Mikhailova	15.12.2002	**********	103	05.07.2023	True
9	Mikhail	Novikov	05.02.1993	**********	105	05.07.2023	True
10	Elizaveta	Fedorova	18.05.2004	**********	102	05.07.2023	True
11	Evgeniy	Morozov	26.09.2001	**********	101	05.07.2023	True
12	Semen	Volkov	16.08.1988	**********	103	05.07.2023	True
13	Vladislav	Alekseev	18.07.1981	**********	102	05.07.2023	True
14	Maksim	Lebedev	20.03.1988	**********	104	05.07.2023	False
15	Aleksandra	Semenova	27.06.1998	**********	102	05.07.2023	True
16	Kristina	Egorova	03.06.1999	**********	101	05.07.2023	True
17	Arina	Pavlova	21.05.1983	**********	102	05.07.2023	True
18	Dmitriy	Kozlov	07.05.1982	**********	101	06.07.2023	False
19	Danil	Stepanov	02.08.1986	**********	101	06.07.2023	True
20	Anna	Nikolaeva	20.04.1981	**********	101	05.07.2023	True
21	Rostislav	Orlov	27.03.1987	**********	101	06.07.2023	False

Таблица II.1.2: `flights`

id	flight_num	departure	arrival	flight_date	flight_date	status
1	101	Moscow	Kazan	05.07.2023	14:00	True
2	102	Moscow	Sochi	05.07.2023	15:30	False
3	103	Vladivostok	Novosibirsk	05.07.2023	09:00	True
4	104	Moscow	Ufa	05.07.2023	17:20	True
5	105	Moscow	Saint Petersburg	05.07.2023	19:00	True
6	101	Kazan	Kaliningrad	06.07.2023	11:15	True

Таблица `passengers` является информацией о пассажирах, которые приобрели билеты на рейсы данной авиакомпании.

В колонках:

- `id` — номер записи в таблице;
- `first_name` — имя пассажира;
- `second_name` — фамилия пассажира;
- `birth` — дата рождения;
- `document` — номер документа, по умолчанию в авиакомпании он скрыт;
- `flight_num` — номер рейса, на который пассажир приобрел билет;
- `flight_date` — дата вылета рейса;
- `status` — активен ли статус пассажира на данный рейс, если `True` — пассажир полетит (или уже полетел), `False` — билет был сдан.

Таблица `flights` является информацией о рейсах авиакомпании.

В колонках:

- `id` — номер записи в таблице;
- `flight_num` — номер рейса;
- `departure` — город вылета;
- `arrival` — город прилета;
- `flight_date` — дата вылета рейса;
- `departure_time` — время вылета рейса;
- `status` — активен ли статус рейса, если `True` — будет выполнен (или уже выполнен), `False` — рейс отменен.

Решение

Исходя из условия задачи, выберем те рейсы, которые подходят, их всего два.

101	Moscow	Kazan	05.07.2023	14:00	True
104	Moscow	Ufa	05.07.2023	17:20	True

Далее идем по таблице и ищем всех людей, которые летят 05.07.2023 номерами рейсов 101 или 104 со статусом `True`.

Людей с номером рейса 101, но датой вылета 06.07.2023 в расчет не берем, так как этот рейс не вылетает из Москвы.

1	Ivan	Ivanov	25.05.1999	**********	104	05.07.2023	True
11	Evgeniy	Morozov	26.09.2001	**********	101	05.07.2023	True
16	Kristina	Egorova	03.06.1999	**********	101	05.07.2023	True
20	Anna	Nikolaeva	20.04.1981	**********	101	05.07.2023	True

Ответ: 4.

Задача II.1.1.2. Вечный XOR (9 баллов)

Темы: алгебра логики.

Условие

Дано число 11011001 в двоичной системе счисления. К данному числу применяется операция XOR на другое, неизвестное нам, восьмизначное число в двоичной системе счисления. После операции выполняется проверка: если результат операции меньше восьмизначного, к нему дописываются незначащие нули. Такой проверкой мы поддерживаем восьмизначный формат числа. После этого операция XOR и проверка выполняются снова в той же последовательность и так до бесконечности...

Определите восьмизначное неизвестное число, которое применяется в операции XOR, если известно, что на 127 применении операции в этом алгоритме результат до проверки был равен 1100011.

Решение

Заметим одну интересную особенность функции XOR: если взять результат операции XOR числа 217 и любого числа x (допустим 3) и к результату вновь применить операцию XOR с числом x (в нашем случае 3), то мы вернемся к исходному числу.

$$\begin{array}{rcl} 11011001 & = & 217 \\ \hat{} \quad 1100011 & = & 99 \\ \hline 10111010 & = & 186 \end{array}$$

Получается, что на 127-й по счету операции XOR, то есть нечетной, будет получено промежуточное число, которое по условию равно 1100011 или 99.

Осталось лишь узнать неизвестное число x, которое будет давать 99 в результате XOR с исходным числом 217.

Для этого можно узнать результат XOR между числами 217 и 99.

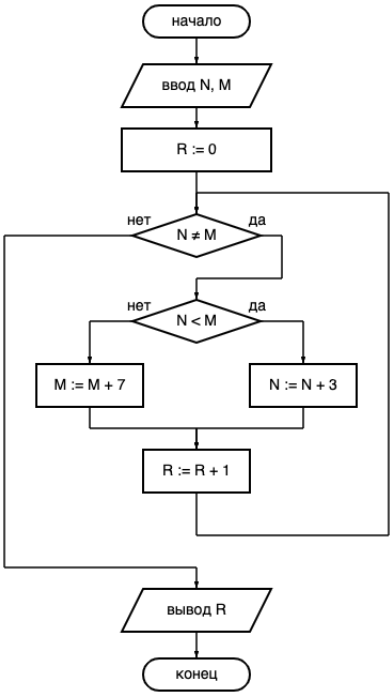
Ответ: 186.

Задача II.1.1.3. Сколько раз (11 баллов)

Темы: анализ алгоритмов.

Условие

Дана блок-схема алгоритма. Какое число будет выведено, если на вход были поданы $N = 41$ и $M = 57$.

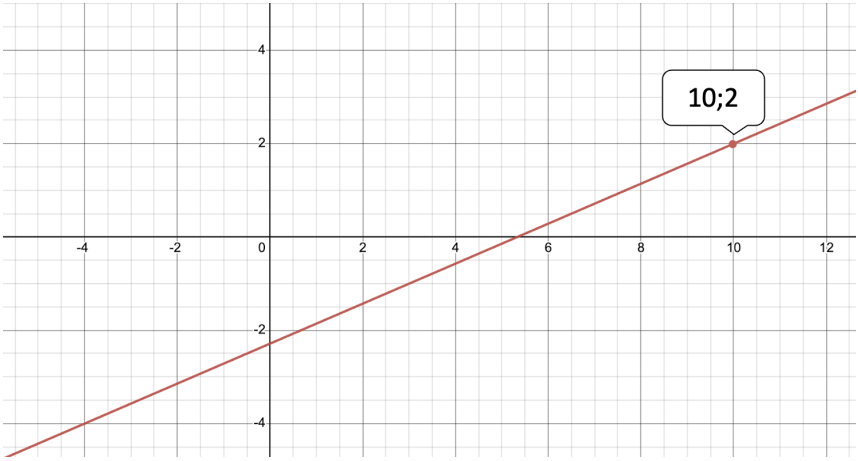


Решение

Начальные знания чисел $n = 41$ и $m = 57$. Как видно из алгоритма, программа будет прибавлять 3 к числу n (если $n < m$) и прибавлять 7 к числу m (если $m < n$) до тех пор, пока эти числа не станут равны. Значит сумма, прибавленная к числу n должна быть больше суммы прибавленной к числу m на $57 - 41 = 16$, из чего можно составить уравнение:

$$3x - 7y = 16.$$

Отсюда можно подобрать два таких целых, минимальных x и y, при которых это уравнение будет верно. Также можно построить график и найти, где он впервые проходит через целые положительные координаты.



Раз $x = 10$, а $y = 2$ то суммарное количество операций будет равно 12.

Ответ: 12.

Задача II.1.1.4. Дорога до работы (11 баллов)

Темы: графы.

Условие

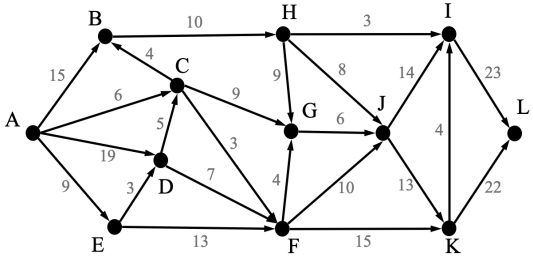
На рисунке приведена схема района «Северный», где каждая вершина графа, показанная латинскими буквами от A до L, обозначают объекты его инфраструктуры, а ребра — дороги между ними.

Гарантируется, что никаких других путей в этом районе нет и что двигаться можно лишь по направлению ребер, которое указано стрелками.

Рядом с каждой дорогой указана ее пропускная способность, которая показывает предельное количество машин, проходящих через эту дорогу за единицу времени.

Буквой A обозначен новый жилой комплекс, а буквой L — IT-парк, в который все ездят на работу с утра.

Ваша задача узнать — какое максимальное количество машин может проходить утром по дорогам этого района в единицу времени или же максимальную пропускную способность данного графа.

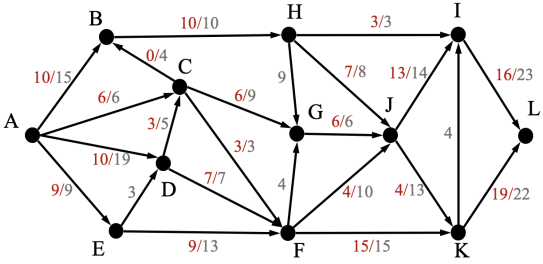


Решение

Для решения этой задачи воспользуемся теоремой **max-flow min cut** о максимальном потоке и минимальном разрезе, которая утверждает, что в сети потоков максимальный объем потока, проходящего от истока к стоку, равен общему весу ребер в минимальном разрезе, т. е. наименьший общий вес ребер, удаление которых отключило бы исток от стока.

Самым минимальным разрезом является удаление ребер BH, CF, DF, AE и GJ с суммой $10 + 3 + 7 + 9 + 6 = 35$, все другие разрезы отключающие исток от стока будут иметь большую сумму.

Стоит отметить, что данную задача можно было решить и используя алгоритм Форда-Фалкерсона.



Ответ тоже получится $16 + 19 = 35$.

Ответ: 35.

Задача II.1.1.5. Уличный транспорт (14 баллов)

Темы: кодирование.

Условие

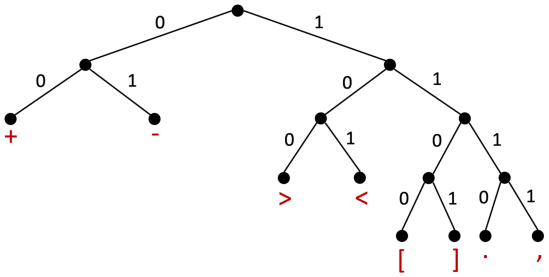
Даня и Ваня на уроке информатики получили очень странное задание. В нем им необходимо определить, какое минимальное количество информации будет содержать сообщение о способе перемещения случайного прохожего. Для этого они

простояли целые сутки на улице и поняли, что прохожие в основном передвигаются одним из пяти следующих вариантов: пешком, на самокате, велосипеде, скейтборде или роликах. Так как Ваня опаздывал на свидание, он решил, что все варианты транспорта равновероятные и убежал. Но Даня заметил одну особенность: пешеходы встречаются с вероятностью 50%, а прохожие на самокате, велосипеде, роликах и скейтбордах с вероятностью 12,5% каждый.

На сколько бит количество информации, содержащееся в сообщении о транспорте прохожего, которое посчитает Ваня, будет отличаться от количества информации, рассчитанного Даней?

Решение

При равномерном посимвольном кодировании Вани: мощность алфавита равна 8, так как всего 8 команд, для кодирования которых по формуле Хартли потребуется 3 бита. Всего в программе 100 символов, а значит, вся программу будет весить $3 \cdot 100 = 300$ бит при неравномерном кодировании Дани.



Тогда получается, что программа будет весить $2 \cdot 2 \cdot 32 + 3 \cdot 2 \cdot 6 + 4^4 \cdot 6 = 260$ бит, что на 40 меньше веса, полученного Ваней.

Ответ: 40.

Задача П.1.1.6. Нули и единицы (14 баллов)

Темы: системы счисления.

Условие

Существует два целых числа x и y, удовлетворяющих выражению $x = 2^i - 1$, $y = 2^i - 1$, где $x \in [1; 64]$.

Определите, сколько существует вариантов выбрать x и y при следующих условиях:

1. $x > y$.
2. Произведение данных чисел в двоичной записи содержит хотя бы одну единицу и хотя бы один ноль.

3. Произведение данных чисел в двоичной записи имеет разницу между количеством единиц и нулей не более 13.

Решение

По условию у нас есть два числа $x = 2^i - 1$, $y = 2^j - 1$, где $1 \leqslant i, j \leqslant 54$.

Зная, что любое число $z = 2^n - 1$, где $n \in \mathbb{N}$, выглядит как n единиц:

$$2^3 - 1 = 111_2$$

$$2^4 - 1 = 1111_2$$

и так далее, делаем вывод, что наши числа — это тоже набор от 1 до 64 единиц в двоичной системе счисления.

Кроме того, заметим одно интересное свойство, что если перемножать числа такого вида друг на друга, то результат всегда будет содержать ровно столько нулей, сколько было единиц в меньшем числе, и ровно столько единиц, сколько их было в большем.

Например:

$$\begin{array}{cccccccc} & & & & & 1 & 1 & 1 \\ \times & & & 1 & 1 & 1 & 1 & 1 \\ \hline & & & & & 1 & 1 & 1 \\ & & & & 1 & 1 & 1 & \\ & & & 1 & 1 & 1 & & \\ & & 1 & 1 & 1 & & & \\ \hline 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{array}$$

или

$$\begin{array}{ccccccccccc} & & & & & & & 1 & 1 & 1 & 1 \\ \times & & & & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline & & & & & & & 1 & 1 & 1 & 1 \\ & & & & & & 1 & 1 & 1 & 1 & \\ & & & & & 1 & 1 & 1 & 1 & & \\ & & & & 1 & 1 & 1 & 1 & & & \\ & & & 1 & 1 & 1 & 1 & & & & \\ & & 1 & 1 & 1 & 1 & & & & & \\ & 1 & 1 & 1 & 1 & & & & & & \\ \hline 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}$$

Получается, что для выполнения третьего условия нам необходимо перемножать числа с разницей не более 13 разрядов. При этом для выполнения второго условия число j не должно быть 1, ведь любое число, умноженное на 1, не будет изменяться. А первое условие, что x всегда строго больше y, позволяет избежать повторы, взяв число y за меньшее левое число пары, а x — за большее правое.

Получается, что для $j = 2$ в пару можно взять любое $i \in [3; 15]$, и такое правило будет работать для всех $j \in [2; 51]$

Выходит, что на 50 вариантов взятия j существует по 13 вариантов взятия i, а это уже $50 \cdot 13 = 650$ пар.

Для каждого $j \geqslant 52$ количество возможных i будет уменьшаться, так как i не может быть больше 64, а значит, для $j = 52$ будет всего 12 возможных вариантов i, для $j = 53$ будет всего 11 возможных вариантов i и так далее, что выливается в арифметическую прогрессию: $12 + 11 + \ldots + 2 + 1 = 78$ вариантов.

Всего получается $650 + 78 = 728$ пар.

Ответ: 728.

Задача II.1.1.7. Кольцевой сборщик (17 баллов)

Темы: программирование.

Условие

На некотором заводе решили расфасовать детали. Каждая деталь имеет свой размер, выраженный как целое число. Для фасовки сотрудники взяли кольцевой сборщик. Кольцевой сборщик — это некий механизм с ячейками разного размера, в которые можно положить деталь. Изначально выбрана для приема детали ячейка под номером 1, каждую секунду она сдвигается на следующую: через секунду будет выбрана для приема ячейка под номером 2, через две секунды — под номером 3 и так далее... Если сборщик дойдет до последней ячейки, он на следующем шагу окажется на ячейке под номером 1. Чтобы разместить деталь в сборщик, необходимо, чтобы размер выбранной для приема ячейки был равен размеру детали. Всего необходимо погрузить n деталей, каждая — имеет свой уникальный размер от 1 до n включительно. Сами они загружаются в сборщик по возрастанию, сначала с размером 1, потом с размером 2, и так далее до размера n включительно. Работники завода попросили у Вас помощи. Они сообщили вам, сколько у них деталей, а также порядок ячеек в кольцевом сборщике, и просят Вас написать программу, которая рассчитает, через сколько все детали будут погружены в кольцевой сборщик. Считайте, что деталь укладывается в кольцевой сборщик моментально.

Формат входных данных

В первой строке входных данных записано число n ($1 \leqslant n \leqslant 10^5$) — количество деталей.

Во второй строке записано n целых чисел s_i ($1 \leqslant s_i \leqslant n$) — последовательность размеров ячеек в кольцевом сборщике. Все размеры ячеек являются уникальными числами.

Выбранной при старте ячейкой считать первое число последовательности.

Формат выходных данных

Выведите одно число — количество времени, затраченное на расфасовку всех деталей.

Методика проверки

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 0,5 балла. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
4 3 1 4 2
Стандартный вывод
6

Пояснения к примеру

Первая на вход идет деталь с размером 1, чтобы добраться до ячейки с размером 1 необходимо затратить одну су: $3 \to 1$. Следующая на вход идет деталь с размером 2, чтобы добраться до ячейки с размером 2 необходимо затратить две сы: $1 \to 4 \to 2$. Следующая на вход идет деталь с размером 3, чтобы добраться до ячейки с размером 3 необходимо затратить одну су: $2 \to 3$. Последняя на вход идет делать с размером 4, чтобы добраться до ячейки с размером 4 необходимо затратить две сы: $3 \to 1 \to 4$. Итого было затрачено на расфасовку всех деталей 6 с.

Решение

Заведем отдельный список/словарь, который в качестве индексов будет использовать размеры деталей, а в качестве значений — индексы ячеек для деталей на ленте. Так как детали укладываются последовательно, пройдем циклом по деталям размерами от 1 до N включительно. На момент начала укладки мы расположены над ячейкой под номером 1. Для вычисления времени до нужной нам ячейки, зная, что лента меняет ячейку каждую секунду, воспользуемся следующей формулой «точка расположения ячейки для нужной детали — наше нынешнее положение». Тем самым мы вычислим расстояние до ячейки, что и будет эквивалентно в рамках нашей задаче времени до ячейки.

Если точка расположения нашей ячейки находится позади нашей позиции, проверен полный круг по ленте, вернувшись в стартовое положение, и добавим расстояние до нужной ячейки: «N — наше нынешнее положение + точка расположения ячейки для нужной детали». После каждого перемещения по ленте обновляем нынешнюю позицию на точку, до которой мы дошли на этом шагу: «наше нынешнее положение = = точка расположения ячейки для нужной детали». Суммируем все рассчитанные расстояния и получаем полное время, за которое мы обойдем всю ленту и уложим все детали.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```python
n = int(input())
indexes = dict()
arr = list(map(int, input().split()))
for i in range(n):
    indexes[arr[i]] = i
    current_index = result = 0
for i in range(n):
    if current_index < indexes[i + 1]:
        result += indexes[i + 1] - current_index
    else:
        result += n + indexes[i + 1] - current_index
    current_index = indexes[i + 1]
print(result)
```

Задача II.1.1.8. Кредиты в банке (17 баллов)

Темы: программмирование.

Условие

В некотором банке регулярно проходит огромное количество транзакций в сутки. Все эти транзакции (без указания личных данных клиентов) отображаются в логах банка. Это сделано для того, чтобы можно было анализировать количество денег, которые клиенты внесли в банк. Как вы знаете, банки обладают возможностью выдавать кредиты своим клиентам, но они их выдают из денег, которые вложили другие клиенты. И, естественно, чтобы выдать кредит, банк должен иметь в наличии ту сумму, на которую он это хочет сделать. Нормальной системы контроля денег у банка, о котором у нас в задаче идет речь, нет, поэтому они это делают через логи. Они узнают по ним гарантированное количество уникальных денежных единиц, которое было зафиксировано, и тем самым определяют гарантированную сумму, которую могут выдать в кредит.

Вам был дан некий отрезок из логов этого банка. Каждый клиент закодирован уникальным номером. Определите, какое гарантированное количество уникальных денежных единиц есть у банка на кредит. Для подробного понимания, как высчитывается гарантированная величина уникальных денежных единиц, смотрите пояснение к примеру.

Формат входных данных

На вход программе в первой строке поступает целое число n $1 \leqslant n \leqslant 10^5$ — количество операций в логах. В следующих n строках записано по три целых числа $from$ ($1 \leqslant from \leqslant 500$), to ($1 \leqslant to \leqslant 500$), $from \neq to$, и $amount$ ($1 \leqslant amount \leqslant 10^9$) — клиенты, которые отправили и получили деньги соответственно, а также количество денежных единиц.

Формат выходных данных

Программа должна вывести одно число — гарантированное количество уникальных денежных единиц, которые были зафиксированы по логам.

Примеры

Пример №1

Стандартный ввод
3 1 2 50 2 3 30 3 1 40
Стандартный вывод
60

Пояснение к примеру

В примере первая транзакция производится между клиентами 1 и 2 на величину 50 денежных единиц. До этого эти деньги не были в логах, а значит, это 50 уникальных денежных единиц. Дальше идет транзакция между клиентами 2 и 3 на величину 30 денежных единиц. Как мы знаем из первой транзакции, у клиента под номером 2 есть 50 денежных единиц, и, соответственно, эти 30 денежных единиц могли быть пересланы из этих 50, поэтому мы не можем заявлять, что это гарантировано уникальные денежные единицы. В случае, если клиент 2 отправит 30 денежных единиц клиенту 3, то у него может остаться $50 - 30 = 20$ денежных единиц. Следующая транзакция происходит между клиентами 3 и 1 на величину 40 денежных единиц. Так как у клиента 3 нам известно только 30 денежных единиц, которые были отправлены от клиента 2, то оставшиеся $40 - 30 = 10$ будут уникальными единицами денег, так как до этого о них речь нигде в логах не шла. Итого, у нас получается $50 + 10 = 60$ гарантировано уникальных денежных единиц.

Решение

Заведем некий список/словарь, который будет хранить, сколько на данный момент у клиентов денег, которые нам известны, а также переменную, в которую будем записывать количество уникальных денег. Изначально мы не знаем ни одной транзакции, следовательно, про каждого клиента мы знаем о наличии 0 денег. Запускаем цикл, в котором обрабатываем каждую транзакцию следующим образом: от отправителя мы вычитаем сумму денег, которая указана в переводе, которую он отправил, а получателю их начисляем. Если счет отправителя становится отрицательным, следовательно, были отправлены деньги, о которых мы ранее не знали, следовательно, обновляем значение уникальных денег, добавляя модуль отрицательного баланса (той части денег, о которых мы ранее не знали). После этого запишем на баланс отправителя, что у него 0 денег, так как больше нет неизвестных денег. Обработав все транзакции таким образом, в конце выводим переменную с количеством уникальных денег.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```python
n = int(input())
bank_accounts = dict()
unique_moneys = 0
for i in range(n):
    from_user, to_user, amount = map(int, input().split())
    if to_user not in bank_accounts:
        bank_accounts[to_user] = 0
    bank_accounts[to_user] += amount
    if from_user not in bank_accounts:
        bank_accounts[from_user] = 0
    bank_accounts[from_user] -= amount
    if bank_accounts[from_user] < 0:
        unique_moneys += abs(bank_accounts[from_user])
        bank_accounts[from_user] = 0
print(unique_moneys)
```

Вторая волна. Задачи 8–11 класса

Задача II.1.2.1. Жилой дом (7 баллов)

Темы: базы данных.

Условие

Дан фрагмент таблицы базы данных некоторого жилого дома.

Таблица II.1.3: `livers`

id	first_name	last_name	birth	sex	flight_num
1	Ivan	Ivanov	25.05.1999	male	101
3	Ekaterina	Kuznetsova	04.02.1996	female	103
4	Aleksandr	Popov	06.04.1994	male	102
5	Elena	Vasilieva	03.11.1994	female	103
6	Sergei	Petrov	25.06.1984	male	102
7	Daniil	Sokolov	07.12.2000	male	102
8	Anastasia	Mikhailova	15.12.2002	female	103
9	Mikhail	Novikov	05.02.1993	male	103
10	Elizaveta	Fedorova	18.05.2004	female	104
11	Evgeniy	Morozov	26.09.2001	male	105
12	Semen	Volkov	16.08.1988	male	106
13	Vladislav	Alekseev	18.07.1981	male	104
14	Maksim	Lebedev	20.03.1988	male	106
15	Aleksandra	Semenova	27.06.1998	female	105
16	Kristina	Egorova	03.06.1999	female	107
17	Arina	Pavlova	21.05.1983	female	107
18	Dmitriy	Kozlov	07.05.1982	male	107
19	Danil	Stepanov	02.08.1986	male	108
20	Anna	Nikolaeva	20.04.1981	female	109
21	Rostislav	Orlov	27.03.1987	male	109

Таблица `livers` является информацией о пассажирах, которые проживают в доме.

В колонках:

- `id` — номер записи в таблице;
- `first_name` — имя проживающего;
- `second_name` — фамилия проживающего;
- `birth` — дата рождения;
- `sex` — пол проживающего: `male` — мужчина, `female` — женщина;
- `flat_num` — в какой квартире проживает человек.

Исходя из информации данной таблицы, определите, сколько есть потенциальных пар/семей в доме. Потенциальной парой/семьей будем называть таких проживающих, которые живут в одной квартире, имеют разный пол, а также разница их возрастов не превышает пять лет. В каждой квартире может проживать только одна пара, но не обязательно только два человека.

Решение

Учитывая, что в каждой квартире может проживать только одна пара, но не обязательно только два человека, надо проверить каждую квартиру на наличие хотя бы одной такой пары, удовлетворяющей условию задачи:

101: жители 1 и 2 разных полов с разницей в возрасте менее пяти лет — подходит;

102: жители 4, 6 и 7 одинаковых полов — не подходит;

103: жители 3, 5, 8 и 9, при этом у жителей 5 и 9 разный пол с разницей в возрасте менее пяти лет — подходит;

104: жители 10 и 13 разных полов с разницей в возрасте более пяти лет — не подходит;

105: жители 11 и 15 разных полов с разницей в возрасте менее пяти5 лет — подходит;

106: жители 12 и 14 одинаковых полов — не подходит;

107: жители 16,17 и 18, при этом у жителей 17 и 18 разный пол с разницей в возрасте менее пяти лет — подходит;

108: житель 19 — не подходит;

109: жители 20 и 21 разных полов с разницей в возрасте более пяти лет — не подходит.

Итого получается четыре пары.

Ответ: 4.

Задача II.1.2.2. Десятки (9 баллов)

Темы: системы счисления.

Условие

Назовите максимальную систему счисления, где для чисел 10^i $(1 \leqslant i \leqslant 9)$ при переводе в выбранную систему счисления их длина равна i.

Решение

Чтобы выполнялось условие, описанное в задаче, необходимо подставить под i максимальное значение (в рамках задачи это 9), и выбирать систему счисления до того момента, пока длина числа 10^i в некоторой системе счисления равна i. После того, как условие не будет выполняться, число никак не увеличится в размере, а, следовательно, не будет больше систем счислений, удовлетворяющих условию.

Так как длина числа 10^9 в десятичной системе счисления больше 9, начнем с 11-ричной системы счисления:

11-ричная система счисления — $10^9 = 47352388a$ (длина 9).

12-ричная система счисления — $10^9 = 23aa93854$ (длина 9).

13-ричная система счисления — $10^9 = 12c23a19c$ (длина 9).

14-ричная система счисления — $10^9 = 96b4b6b6$ (длина 8) — условие не выполнено.

Максимальная система счисления 13-ричная.

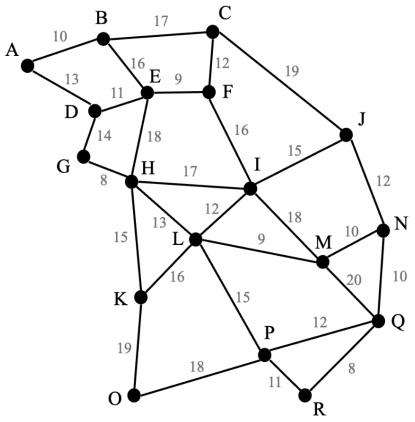
Ответ: 13.

Задача II.1.2.3. Дорожные работы (11 баллов)

Темы: теория графов.

Условие

Министерству транспорта некоторого города поступил запрос с обновлением асфальтоукладочного покрытия между важными элементами инфраструктуры. Однако совсем скоро зима, поэтому автомагистрали и дороги нужны проложить как можно скорее. Все возможные варианты прокладки дорог с требуемым для этого временем указаны на рисунке.



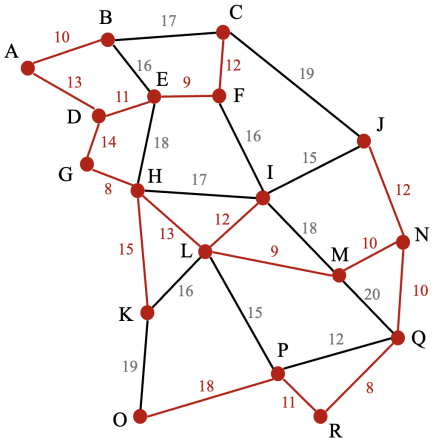
Главным условием является то, что до каждого из зданий должна быть проложена хотя бы одна дорога.

Определите минимальное время, которое потребуется на асфальтирования этого города.

Решение

Для решения этой задачи достаточно построить минимальное остовное дерево графа. Самый простой способ сделать это — воспользоваться алгоритмом Краскала, который каждый раз берет ребро с минимальным весом и, если такое взятие не образует цикла, присоединяет его к скелету.

Получим такую картину.



Ответ: 195.

Задача II.1.2.4. Необычный отель (11 баллов)

Темы: анализ алгоритмов.

Условие

Ваня поехал в отпуск и заселился в очень необычный отель. В нем ровно 9999 номеров. Перед заселением все двери этих номеров открыты, а странный консьерж каждый раз после уборки номера меняет состояние двери: с открытой на закрытую и наоборот, чтобы проветривать комнату. При этом сначала он делает уборку во всех комнатах с номерами кратными 1, потом — 2, потом — 3 . . . и так до 9999.

Определите самый большой номер комнаты, который будет открыт после уборки.

Решение

Из условия известно, что консьерж проходит номера последовательно их кратности (сначала номера с кратностью 1, затем с кратностью 2 и т. д.).

Сделаем вывод: сколько раз номер в отеле кратен некоторым числам, столько раз его и посетят. Все числа, на которые номер комнаты кратен — это делители нашего номера, следовательно, сколько делителей у номера комнаты — столько раз ее и посетят.

Определим, в каком порядке происходят действия с дверьми:

- каждую нечетную операцию дверь меняет свое состояние с открытой на закрытую;
- каждую четную операцию дверь меняет свое состояние с закрытой на открытую (дверь будет закрыта, так как до этого была нечетная операция).

Следовательно, номер с последней закрытой дверью — это самое большое число номера, над которым проведено нечетное количество операций, или, исходя из ранее выведенного условия, нечетное количество делителей.

Нечетное количество различных делителей имеют только числа, которые являются квадратами (например, $4 = 2^2$, $81 = 9^2$). Следовательно, найдем самый большой квадрат, который меньше 9999. Ближайший полный квадрат к 9999, это $10000 = 100^2$. 10000 является большим результатом, поэтому возьмем меньший на единицу (минимальный шаг) квадрат: $(100 - 1)^2 = 99^2 = 9801$. $9801 < 9999$, а также является наибольшим квадратом, так как следующий квадрат уже превышает номер последней комнаты.

Ответ: 9801.

Задача II.1.2.5. Футбольный турнир (14 баллов)

Темы: кодирование.

Условие

В этом году проходит ежегодный футбольный турнир среди Assembler-программистов. Ежегодно это соревнование объединяет миллионы людей со всего мира, каждый с нетерпением ждет его проведения. Сейчас на соревнование было зарегистрировано 512 команд. Все соревнование проходит в три этапа: отборочный этап, групповой этап и финальный этап. Во время отборочного этапа проходит четыре стадии турнира: $\frac{1}{256}$, $\frac{1}{128}$, $\frac{1}{64}$ и $\frac{1}{32}$. Все матчи проходят по 90 мин основного времени, и, в случае ничейного результата, добавляется дополнительное время 30 мин. Если после 120 мин матча не удается выяснить победителя, проходит серия пенальти.

После отборочного этапа остается 32 команды, и они попадают в групповой этап. Все эти команды случайным образом распределяются по восьми группам, и в процессе этапа они сыграют каждый с каждым по два раза, то есть любая команда на этой стадии сыграет 6 матчей. Во время группового этапа матчи проходят только по 90 мин, независимо от результата.

По итогам группового этапа в финальную стадию проходит 16 лучших команд, и они начинают играть за кубок футбольного ассемблера. Всего проходит 4 стадии: $\frac{1}{8}$,

$\frac{1}{4}$, $\frac{1}{2}$ и финал, матча за третье место нет. Во время финальной стадии сохраняются те же правила проведения матчей, что и в отборочном этапе: 90 +30 + серия пенальти.

Данный турнир проводится не первый год, и организаторы прекрасно знают из своей статистики, что в дополнительное время в отборочном этапе заканчивается не более 10% матчей, а также не более 20% матчей в финальном этапе.

Организаторы хотят, чтобы весь турнир прошел на высшем уровне и без нареканий, но у них возник вопрос: сколько памяти надо выделить, чтобы гарантировано сохранить все результаты матча. Организаторы хотят хранить отчеты по матчам поминно, выделяя на каждую мину по 2 байта. Серию пенальти они решили не хранить, а записывать гол на счет победителя на 120 мине. Каждый этап соревнования хранится отдельно, независимо от других, в килобайтах. Исходя из статистических данных процентов матчей, заканчивающихся в основное или дополнительное время, а также формата турнира, рассчитайте, какое минимальное целое количество памяти нужно выделить в килобайтах, чтобы гарантировано удалось сохранить все результаты турнира поминно.

Считать, что 1 Кбайт равен 1024 байтам.

Решение

Для сохранения матча длительностью 90 мин потребуется 180 байт, для 120-минутного матча — 240 байт.

Отборочный этап

В $\frac{1}{256}$ стадии пройдет $\frac{512}{2}$ (количество команд, участвующих в матче) = 256 матчей, и, следовательно, в следующую стадию пройдет 256 команд.

В $\frac{1}{128}$ стадии пройдет $\frac{256}{2}$ (количество команд, участвующих в матче) = 128 матчей, и, следовательно, в следующую стадию пройдет 128 команд.

В $\frac{1}{64}$стадии пройдет $\frac{128}{2}$ (количество команд, участвующих в матче) = 64 матча, и, следовательно, в следующую стадию пройдет 64 команды.

В $\frac{1}{32}$ стадии пройдет $\frac{64}{2}$ (количество команд, участвующих в матче) = 32 матча, и, следовательно, в следующую стадию пройдет 32 команды.

Всего за отборочную стадию пройдет 256+128+64+32 = 480 матчей, не более 10% из которых могут закончиться в дополнительное время с серией пенальти: $480 \cdot 0,1 =$ = 48 матчей; в основное время закончится: 480 − 48 = 432 матча.

Следовательно, для хранения данных о матчах в отборочном этапе потребуется: $432 \cdot 180 + 48 \cdot 240 = 89280$ байт.

Групповой этап

Всего будет 32 команды, поделенных равномерно на 8 групп, следовательно, в каждой группе по 4 команды. Каждая команда сыграет друг против друга по два раза, следовательно, всего будет 6 туров между командами, а в каждом туре будет по 2 матча. Посчитаем, сколько матчей будет проведено всего: $6 \cdot 2 \cdot 8 = 96$ матчей. Все матчи пройдут только в основное время: $96 \cdot 180 = 17280$ байт.

Финальный этап

В $\frac{1}{8}$ стадии пройдет $\frac{16}{2}$(количество команд, участвующих в матче) = 8 матчей, и, следовательно, в следующую стадию пройдет 8 команд.

В $\frac{1}{4}$ стадии пройдет$\frac{8}{2}$ (количество команд, участвующих в матче) $= 4$ матча, и, следовательно, в следующую стадию пройдет 4 команды.

В $\frac{1}{2}$ стадии пройдет $\frac{4}{2}$ (количество команд, участвующих в матче) $= 2$ матча, и, следовательно, в следующую стадию пройдет 2 команды.

В финале пройдет всего 1 матч.

Всего за финальную стадию пройдет $8 + 4 + 2 + 1 = 15$ матчей, не более 25% из которых могут закончиться в дополнительное время с серией пенальти: $15 \cdot 0,25 = 3,75$ матча. Так как в условии задачи указано **не более**, то округляем в меньшую сторону: 3 матча; в основное время закончится: $15 - 3 = 12$ матчей.

Следовательно, для хранения данных о матчах в отборочном этапе потребуется: $12 \cdot 180 + 3 \cdot 240 = 2880$ байт.

Переведем все значения из байт в КБайт:

- для хранения отборочного этапа потребуется $89280/1024 = 87,1875 = 88$ Кбайт;
- для хранения группового этапа потребуется $17280/1024 = 16,875 = 17$ Кбайт;
- для хранения финального этапа потребуется $2880/1024 = 2,8125 = 3$ Кбайт.

Всего для хранения таблицы потребуется: $88 + 17 + 3 = 108$ Кбайт.

Ответ: 108.

Задача II.1.2.6. Фиктивные переменные (14 баллов)

Темы: алгебра логики.

Условие

Дана логическая функция, состоящая из семи переменных:

$$(((a \wedge b) \vee (\neg a \wedge (\neg a \vee e) \wedge b)) \to (c \wedge (d \vee e)) \vee (\neg c \wedge d) \vee (e \wedge \neg c)), \wedge (\neg f \vee (g \wedge f) \vee \neg g).$$

Фиктивными переменными называются те переменные, которые не влияют на результат функции. Выясните, какие переменные являются фиктивными. В ответе укажите их в любом порядке слитно, без пробелов, запятых и иных знаков. Гарантируется, что есть минимум две фиктивные переменные, а также существует хотя бы одна переменная, от которой зависит результат функции.

Решение

Преобразуем выражение:

$$\neg a \wedge (\neg a \vee e) = \neg a.$$

Рассмотрим левую часть, заметим, что:

$$((((a \wedge b) \vee (\neg a \wedge b)) \to (c \wedge (d \vee e)) \vee (\neg c \wedge d) \vee (e \wedge \neg c))) \wedge (\neg f \vee (g \wedge f) \vee \neg g).$$

Теперь обратим внимание на $(a \wedge b) \vee (\neg a \wedge b)$, потому что по свойству склеивания это будет просто b:

$$((b \to (c \wedge (d \vee e)) \vee (\neg c \wedge d) \vee (e \wedge \neg c))) \wedge (\neg f \vee (g \wedge f) \vee \neg g).$$

Во второй скобке можно заметить общий множитель $\neg c$, который можно вынести за скобки (свойство дистрибутивности):

$$((b \to (c \wedge (d \vee e)) \vee (\neg c \wedge (d \vee e)))) \wedge (\neg f \vee (g \wedge f) \vee \neg g).$$

Далее общий множитель $(d \vee e)$, который тоже можно вынести за скобки (свойство дистрибутивности):

$$((b \to (d \vee e) \wedge (c \vee \neg c))) \wedge (\neg f \vee (g \wedge f) \vee \neg g).$$

Здесь $c \vee \neg c = 1$, а значит функция принимает вид:

$$((b \to (d \vee e))) \wedge (\neg f \vee (g \wedge f) \vee \neg g).$$

Уберем лишние скобки:

$$(b \to (d \vee e)) \wedge (\neg f \vee (g \wedge f) \vee \neg g).$$

Теперь преобразуем правую часть, по закону поглощения:

$$\neg f \vee (g \wedge f) = \neg f \vee g,$$

после этого логическое выражение имеет следующий вид:

$$(b \to (d \vee e)) \wedge (\neg f \vee g \vee \neg g).$$

Так как $g \vee \neg g = 1$, то и вся скобка тоже превращается в 1, следовательно, функция принимает вид:

$$\neg b \vee d \vee e.$$

Значит, фиктивными переменными являются a, c, f, g.

Ответ: acfg.

Задача II.1.2.7. Прогнозирование (17 баллов)

Темы: программирование.

Условие

Сегодня проходит финал по перетягиванию каната. В нем принимают участие две команды: синих и красных. Обе команды проделали большой путь до этого финала ради призового фонд с конфетами. Но на днях команда красных предложила главному тренеру команды синих конфет за то, чтобы они проиграли. И те и другие будут в плюсе, ведь тогда команда красных заберет призовой фонд, а команда синих получит гарантированные конфеты за проигрыш.

После того как он получил конфеты, руководители команды красных попросили узнать, сколько матчей они смогут гарантированном проиграть. Они дали ему один день на обдумывание, чем он и занялся. Тренер помнит, что финал проходит по следующих правилам: от команды представляются n человек, и в рамках финала

проходит также n матчей. В первом матче канат тянут по одному человеку с каждой стороны, во втором матче канат тянут по два человека с каждой стороны, на третий три, и так далее до того, пока канат не будут тянуть с каждой стороны по n человек. Побеждает в матчах та команда, у которой больше суммарная сила на сторону. Если силы равны, объявляется ничья. Тренер знает силы и своей команд, и команды соперника, и вправе на каждый матч сам решать, кто участвует за команду синих. Также он знает, что команда красных будет ставить максимально оптимально своих участников на матчи.

Исходя из этого, он просит вас написать программу, которая посчитает, какое максимальное количество матчей он может проиграть, если будет сам решать кто в каком матче участвует.

Формат входных данных

В первой строке входных данных записано целое число n ($1 \leqslant n \leqslant 2 \cdot 10^5$) — количество участников в каждой команде и одновременно количество матчей в финале. Во второй строке записано n целых чисел a_i ($1 \leqslant a_i \leqslant 10^9$) — силы участников команды синих. В третьей строке записано n целых чисел b_i ($1 \leqslant b_i \leqslant 10^9$) — силы участников команды красных.

Формат выходных данных

Выведите одно целое число — максимальное количество матчей, которое команда синих может проиграть.

Методика проверки

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 0,5 балла. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

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

Пояснения к примеру

Команда синих может проиграть первых два матча. В первом матче они поставят участника с силой 1 против участника команды красных с силой 2.

Во втором матче они поставят участников с силой 1 и 2 против участников команды красных с силами 2 и 2.

В третьем матче можно сделать ничью, но проиграть не получится. В четвертом и пятом матче команда синих может только выиграть.

Решение

Отсортируем силы участников обеих команд. Также создадим две переменные, в которых будут хранится суммарные силы участников команд на определенный раунд. Эти суммы на каждый раунд будут наполняться следующим образом: в команду синих мы будем добавлять самого слабого свободного участника из команды, в то время как в команду красных мы будем добавлять самого сильного свободного участника из команды. Тем самым мы постоянно будем задавать команде синих наиболее слабый состав на каждый раунд, а команде красных, наоборот, наиболее сильный. Посчитаем, в скольких случаях команда синих была слабее команды красных, и выведем данный результат.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n = int(input())
blue_team = sorted(list(map(int, input().split())))
red_team = sorted(list(map(int, input().split())))
blue_sum = 0
red_sum = 0
res = 0
for i in range(n):
    blue_sum += blue_team[i]
    red_sum += red_team[-i - 1]
    if blue_sum < red_sum:
        res += 1
print(res)

```

Задача II.1.2.8. Магические ключи (17 баллов)

Темы: программирование.

Условие

Даня попал в магический коридор, в котором он видит n дверей с разными замочными скважинами. Неизвестный голос говорит ему повернуть голову влево, что он без каких-либо сомнений делает. Перед ним открылась следующая картина: стоит стол, а на нем — неограниченное количество m видов ключей, а также карта, на которой расписано, какая дверь каким ключом открывается.

Все бы ничего, но Даня снова услышал неизвестный голос, который произнес следующие слова: «Эти ключи не простые, а магические. Как только ты используешь ключ, у тебя есть k у. е. времени, чтобы воспользоваться им повторно, иначе он разрушится. Но если ты повторно воспользуешься ключом, он обновится, и у тебя снова будет k у. е. времени, чтобы им воспользоваться повторно. Каждая дверь

открывается ключом за 1 у. е. времени. Если ты хочешь выбраться из этого коридора, воспользуйся картой и собери все ключи, которые тебе нужны, иначе ты здесь останешься на века».

В этой ситуации каждый будет брать все и как можно больше, но не Даня. Он решил быть рациональным и не забивать все карманы ненужными ключами. Он отправил вам по «аське» карту и информацию про все магические свойства ключей, и просит написать программу, которая рассчитает минимальное количество ключей каждого вида, которые должен взять Даня, а также их суммарное количество.

Формат входных данных

В первой строке входных данных записано три целых числа n ($1 \leqslant n \leqslant 10^6$), m ($1 \leqslant m \leqslant 1000$) и k ($1 \leqslant k \leqslant 2000$) — количество дверей, ключей и время действия ключа после первого использования соответственно.

Во второй строке записано n целых чисел a_i ($1 \leqslant a_i \leqslant m$) — номер ключа, которым можно открыть дверь под номером i. Гарантируется, что на каждый вид ключа будет не более 1000 дверей, которые им открываются.

Формат выходных данных

Выведите в первой строке одно число — общее количество ключей, которое необходимо с собой взять. Во второй строке выведите n чисел — сколько ключей надо взять на каждый вид по отдельности. Вывод количества ключей идет по порядку: сначала количество ключей с номером 1, затем — с номером 2, и так далее.

Примеры

Пример №1

Стандартный ввод
5 2 2 2 1 2 2 1

Стандартный вывод
3 2 1

Пояснение к примеру

Для открытия первой двери нужен новый ключ с номером 2. Для открытия второй нужен новый ключ с номером 1. Для открытия третьей двери мы можем воспользоваться ранее взятым ключом 2, так как его время действия еще не закончилось еще. Для открытия четвертой двери воспользуемся ранее взятым ключом 2, так как мы его на предыдущей двери обновили, и теперь отсчет его времени действия начался снова. Для открытия пятой двери нужен новый ключ с номером 1, так как предыдущий ключ потерял свое действие. Итого нам нужно два ключа с номером 1 и один ключ с номером 2.

Решение

Создадим отдельный список/словарь, в который будем записывать в качестве индексов/ключей номера ключей от дверей, а в качестве значений под индексами/ключами будет хранится список индексов дверей, которые открываются этими ключами. После этого запускаем цикл, доставая индексы дверей по определенному ключу и вычисляем, сколько нужно ключей определенного типа, чтобы открыть все двери, которые подходят под него. Для того чтобы понимать, нужен новый ключ или нет, воспользуемся следующим условием: если разница между позицией двери и предыдущей двери, открываемой данным ключом, больше времени активности ключа, то требуется новый ключ, в ином случае нет. Если у нас есть хотя бы одна дверь, которая открывается определенным типом ключа, нужно взять минимум один ключ, в ином случае ключи не нужны. Суммируем количество раз, сколько раз, исходя из условия, потребовалось взять еще ключей для дверей, а также добавляем один (чтобы взять первый ключ для дверей). Сохраняем для двери в списке данный результат. В итоге проходимся по всем ключам и дверям для них и выводим сумму всех ключей, а также по отдельности необходимое количество ключей.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n, m, active_time = map(int, input().split())
keys = list(map(int, input().split()))
arr = [[] for i in range(m)]
for i in range(n):
    arr[keys[i] - 1].append(i)
res = 0
count_keys = []
for doors_by_someone_key in arr:
    count_keys.append(0)
    if len(doors_by_someone_key) == 0:
        continue
    prev_door = doors_by_someone_key[0]
    res += 1
    count_keys[-1] += 1
    for door in doors_by_someone_key:
        if door - prev_door > active_time:
            res += 1
            count_keys[-1] += 1
        prev_door = door
print(res, count_keys, sep='\n')
```

Третья волна. Задачи 8–11 класса

Задача II.1.3.1. Аренда авто (7 баллов)

Темы: базы данных.

Условие

Даны фрагменты двух таблиц базы данных некоторой каршеринговой компании.

Таблица II.1.4: Операции

id	Имя	Фамилия	Пол	Дата аренды	id авто	Сумма аренды	Штраф
1	Данил	Смирнов	м	03.08.2023	104	242	Есть
2	Екатерина	Кузнецова	ж	04.08.2023	106	314	Нет
3	Сергей	Попов	м	06.08.2023	105	147	Есть
4	Анастасия	Васильева	ж	08.08.2023	103	150	Нет
5	Елизавета	Штольц	м	10.08.2023	103	219	Есть
7	Дмитрий	Солоков	м	10.08.2023	10	300	Нет
8	Елена	Новикова	ж	12.08.2023	103	258	Есть
9	Михаил	Федоров	м	17.08.2023	10	294	Есть
10	Филипп	Морозов	м	18.08.2023	102	190	Нет
11	Евгений	Волков	м	20.08.2023	101	178	Нет
12	Владислав	Алексеев	м	25.08.2023	103	218	Нет
13	Максим	Лебедев	м	25.08.2023	102	176	Нет
14	Александра	Семенова	ж	28.08.2023	104	315	Есть
15	Арина	Егорова	ж	01.09.2023	102	233	Есть
16	Кристина	Павлова	ж	03.09.2023	101	166	Есть
17	Даниил	Казаченко	м	03.09.2023	102	252	Нет
18	Иван	Козлов	м	04.09.2023	101	323	Есть
19	Агата	Орлова	ж	06.09.2023	106	181	Нет
20	Владимир	Николаев	м	06.09.2023	101	271	Нет
21	Ростислав	Никифоров	м	07.09.2023	106	199	Есть

Таблица II.1.5: Автомобили

id	id авто	Марка	Модель	Номер	Год выпуска	Тип двигателя
1	101	Renault	Kaptur	К123ДЖ|50	2019	бензиновый
2	102	Renault	Logan	К015ТИ|50	2019	бензиновый
3	103	Skoda	Octavia	К329ЮТ|50	2019	дизельный
4	104	Skoda	Octavia	К841ГМ|50	2018	бензиновый
5	105	Audi	A3	К418ДВ|50	2013	дизельный
6	106	Renault	Kaptur	К641ЛТ|50	2017	бензиновый

Таблица `Операции` является информацией о арендаторах, которые воспользовались услугами каршеринговой компании.

В колонках:

- id — номер записи в таблице;
- имя — имя клиента;
- фамилия — фамилия клиента;
- пол — пол клиента;
- дата аренды — дата, когда клиент арендовал автомобиль;
- id авто — номер автомобиля, который арендовал клиент;
- сумма аренды — итоговая сумма аренды автомобиля клиентом;
- штраф — имеет ли клиент штраф за поездку.

Таблица `Автомобили` является информацией об автомобилях компании.

В колонках:

- id — номер записи в таблице;
- id авто — номер автомобиля, который арендовал клиент;
- марка — марка автомобиля;
- модель — модель автомобиля;
- номер — серийный номер автомобиля;
- год выпуска — год, когда был выпущен автомобиль;
- тип двигателя — тип двигателя автомобиля (бензиновый или дизельный).

Исходя из информации данных таблиц, определите, на сколько больше денег заработала компания на мужчинах, которые арендовали бензиновые автомобили, по сравнению с женщинами, арендовавшими дизельные?

Решение

Автомобили с бензиновыми двигателями имеют id 101, 102, 104, 106.

Автомобили с дизельными двигателями имеют id 103, 105.

Найдем всех мужчин, которые арендовали автомобили с id 101, 102, 104, 106:

id	**Имя**	**Фамилия**	**Пол**	**Дата аренды**	**id авто**	**Сумма аренды**	**Штраф**
1	Данил	Смирнов	м	03.08.2023	104	242	Есть
7	Дмитрий	Солоков	м	10.08.2023	101	300	Нет
10	Филипп	Морозов	м	18.08.2023	102	190	Нет
11	Евгений	Волков	м	20.08.2023	101	178	Нет
13	Максим	Лебедев	м	25.08.2023	102	176	Нет
17	Даниил	Казаченко	м	03.09.2023	102	252	Нет
18	Иван	Козлов	м	04.09.2023	101	323	Есть
20	Владимир	Николаев	м	06.09.2023	101	271	Нет
21	Ростислав	Никифоров	м	07.09.2023	106	199	Есть

Суммарно получается 2131 руб. Теперь найдем сколько компания заработала на девушках, арендовавших машины с id 103 и 105.

id	**Имя**	**Фамилия**	**Пол**	**Дата аренды**	**id авто**	**Сумма аренды**	**Штраф**
4	Анастасия	Васильева	ж	08.08.2023	103	150	Нет
8	Елена	Новикова	ж	12.08.2023	103	258	Есть

Итого выходит 408 руб. А значит компания заработал на мужчинах на $2131 - 408 = 1723$ руб. больше.

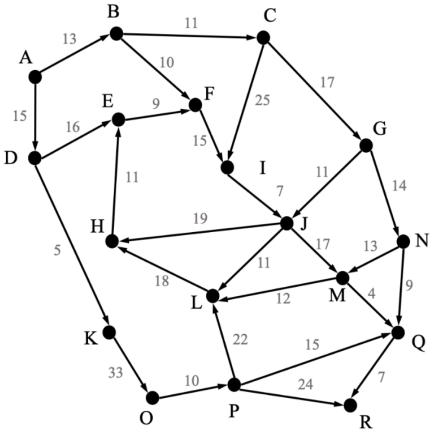
Ответ: 1723.

Задача II.1.3.2. Гонщик (9 баллов)

Темы: теория графов.

Условие

Даня и Ваня играют в одну известную видеоигру «Нужна скорость». Они прошли ее практически всю, за исключением последней сложной миссии. Им нужно как можно дольше скрываться от преследования на время. После многих безуспешных попыток они решили нарисовать карту гоночной локации, которая приведена на рисунке.

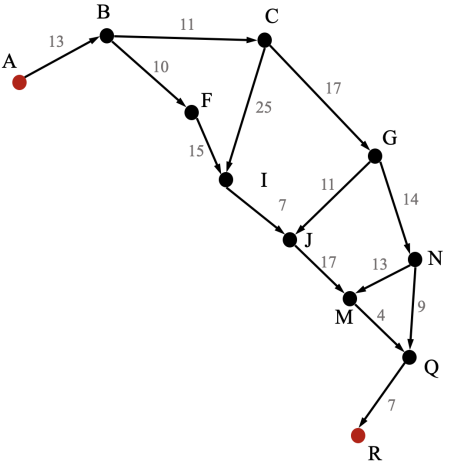


Стартом заезда считается пункт A, а финишем пункт Q. Они замерили максимальное время, которое им удается продержаться на каждом дорожном участке. После замеров Даня и Ваня просят у вас помощи. Найдите максимально возможное время заезда при условии, что через каждый пункт можно проезжать только один раз и двигаться разрешено только в том направлении, куда указана стрелка.

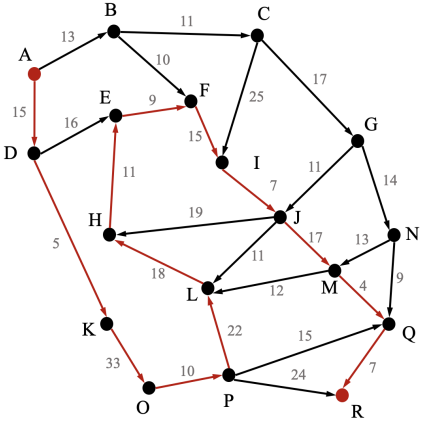
Решение

Заметим, что в графе присутствуют циклы, поэтому попытаемся их разобрать. Очевидно, что после старта есть два основных направления: либо в пункт D (направо), либо в пункт D (вниз).

Если мы идем в пункт B, то мы никогда не сможем попасть в пункт L и H, потому что они вынудят нас пройти через один пункт несколько раз, поэтому картина возможных путей там выглядит очень просто.



Однако если мы идем в пункт D, то мы можем захватить пункт L и H при этом также захватывая и другие пункты из правой части графа.



Применяя обратный алгоритм Дейкстры, сравниваем оба варианта и получаем, что самый долгий путь был на второй картинке, и он составляет 173.

Ответ: 173.

Задача II.1.3.3. Киновечер (11 баллов)

Темы: кодирование.

Условие

Недавно Даня и Ваня ходили в кино на показ новой короткометражки «Опенгеймер».

Они, конечно, были впечатлены актерской игрой и сюжетом, но больше всего им стало интересно, какое же максимальное количество цветов используется в картине. Они решили воспользоваться приложением для скачивания фильмов «Толлент». Из него они узнали, что суммарно произведение весит около 17 Гб при разрешении 1280×720 и частоте кадров 25 к/с, при этом сама картина длится приблизительно 20 мин, а звук кодировался отдельно и весит 1 Гб памяти.

Из этих данных определите, какое максимальное количество цветов могло использоваться в кадре.

Учтите, что в данной задаче:

- 1 Гб = 1024 Мб;
- 1 Мб = 1024 Кб;
- 1 Кб = 1024 Байта.

Решение

Первым делом определим общий объем памяти, в который необходимо уложиться для кодирования визуальной составляющей фильма. Если звук занимает 1 Гб, а весь фильм целиком — 17 Гб, то на кодирование картинки остается 16 Гб.

Вес одного любого кадра фильма будет составлять $1280 \cdot 720 \cdot i$, где i — глубина цвета. Видео — это набор картинок, которые показываются с частотой 25 кадров в секунду (по условию) на протяжении 20 мин (также по условию), вес всего видеофайла можно записать как $V = 1280 \cdot 720 \cdot i \cdot 25 \cdot 20 \cdot 60$.

Выражаем отсюда неизвестную i, а вместо V подставляет найденные 16 Гб, предварительно переведенные в биты:

$$i \leqslant \frac{16 \cdot 2^{33}}{1280} \cdot 720 \cdot 25 \cdot 20 \cdot 60,$$

$$i \leqslant 4,971.$$

Очевидно, что глубина цвета быть дробной не может, и округлить в большую сторону ее тоже нельзя, т.к. мы превысим наш размер видео, значит, максимальное количество бит на кодирование цвета, которые мы можем взять, равно 4.

Количество цветов можно легко найти по формуле $N = i^2$, откуда следует, что оно равно 16.

Ответ: 16.

Задача II.1.3.4. Кубическая разница (14 баллов)

Темы: системы счисления.

Условие

Существует некоторое четырехзначное число $x = abcd$ в четверичной системе счисления.

Кроме этого, есть его копия, записанная в обратном порядке, назовем ее $y = dcba$.

Сколько можно выбрать пар чисел x и y так, чтобы модуль их разности являлся кубом какого-либо целого числа?

Решение

По условию задачи имеется четверичное число x, которое можно так и представить $x = abcd$, где переменные a,b,c,$d \in [0;3]$, так как являются цифрами четверичного алфавита.

Кроме того имеется число y — инвертированная запись числа x, которая равна $y = dcba$, где переменные a, b, c, $d \in [0;3]$, так как являются цифрами четверичного алфавита.

Представляя числа в десчтичной системе счисления, запишем уравнение, что разность чисел x и y равна кубу некоторого числа e:

$$(a \cdot 4^3 + b \cdot 4^2 + c \cdot 4^1 + d \cdot 4^0) - (d \cdot 4^3 + c \cdot 4^2 + b \cdot 4^1 + a \cdot 4^0) = e^3,$$

$$(64a + 16b + 4c + d) - (64d + 16c + 4b + a) = e^3,$$

$$64a + 16b + 4c + d - 64d - 16c - 4b - a = e^3,$$

$$63a + 12b - 12c - 63d = e^3,$$

$$63(a - d) + 12(b - c) = e^3.$$

Учитывая, что a,b,c,$d \in [0;3]$, так как являются цифрами четверичного алфавита, как можно получить куб в разнице 63 и 12?

Такой вариант всего 1 и это: $63 - 12 \cdot 3 = 27$ Значит $a - d = 1$, $b - c = -3$.

Тогда подходит пара чисел (2031; 1302). И еще одна пара (3032; 2303).

Ответ: 2.

Задача II.1.3.5. Кубическая разница (14 баллов)

Темы: алгебра логики.

Условие

Даны две логический функции:

$$F_1 = (\neg y \vee (y \wedge \neg z) \wedge (y \vee \neg e)) \to (x \wedge w \vee w \wedge x),$$

$$F_2 = ((\neg x \vee \neg y \vee \neg z) \wedge (x \vee y \wedge z)) \wedge (\neg w \vee (e \wedge w \vee w \wedge \neg e)).$$

Определите, в скольких из всех возможных значений пяти переменных x, y, z, w, e результаты двух функций будут отличаться друг от друга?

Решение

Упростим обе функции.

Функцию $F_1 = (\neg y \vee (y \wedge \neg z) \wedge (y \vee \neg e)) \to (x \wedge w \vee \neg w \wedge x)(x \wedge w \vee \neg w \wedge x)$ можно упростить по свойствам дистрибутивности: $(x \wedge (w \vee \neg w))$ и $w \vee \neg w$ всегда будет истинно: $x \wedge 1 = x$.

Получим

$$F_1 = (\neg y \vee (y \wedge \neg z) \wedge (y \vee \neg e)) \to x.$$

$(y \wedge \neg z) \wedge (y \vee \neg e)$ можно расширить по свойствам дистрибутивности, приняв, что $(y \wedge \neg z) = a$, тогда получим:

$$a \wedge (y \vee \neg e) = (y \wedge a) \vee (\neg e \wedge a) = (y \wedge (y \wedge \neg z)) \vee (\neg e \wedge (y \wedge \neg z)).$$

Передвинем в левой части скобки по свойству ассоциативности:

$$((y \wedge y) \wedge \neg z) \vee (\neg e \wedge (y \wedge \neg z)).$$

Упростим $y \wedge y$по свойству идемпотентности: $(y \wedge \neg z) \vee (\neg e \wedge (y \wedge \neg z))$.

Вернув $a = (y \wedge \neg z)$, упростим выражение по свойству поглощения:

$$a \vee (\neg e \wedge a) = a = (y \wedge \neg z),$$

$$F_1 = (\neg y \vee (y \wedge \neg z)) \to x.$$

В левой части выражения разложим выражение по свойству дистрибутивности:

$$\neg y \vee (y \wedge \neg z) = (\neg y \vee y) \wedge (\neg y \vee \neg z);$$

$(\neg y \vee y)$ всегда будет истинно:

$$1 \wedge (\neg y \vee \neg z) = \neg y \vee \neg z.$$

$$F_1 = (\neg y \vee \neg z) \to x.$$

Разложим импликацию:

$$(\neg y \vee \neg z) \to x = \neg(\neg y \vee \neg z) \vee x.$$

Применим на скобку закон Де Моргана: $y \wedge z \vee x$.

$$F_1 = y \wedge z \vee x,$$

$$F_2 = ((\neg x \vee \neg y \vee \neg z) \wedge (x \vee y \wedge z)) \wedge (\neg w \vee (e \wedge w \vee w \wedge \neg e)).$$

$(e \wedge w \vee w \wedge \neg e)$ можно упростить по свойствам дистрибутивности:

$(w \wedge (\neg e \vee e))$; $\neg e \vee e$ всегда истина: $w \wedge 1 = w$.

$F_2 = ((\neg x \vee \neg y \vee \neg z) \wedge (x \vee y \wedge z)) \wedge (\neg w \vee w)\neg w \vee w$ всегда истина: 1.

$F_2 = ((\neg x \vee \neg y \vee \neg z) \wedge (x \vee y \wedge z))$.

Выражения упрощены до трех переменных, следовательно, две переменные не влияют на результат.

Также если менять значения этих переменных, то ответ, зависимый от трех других, будет повторяться.

Следовательно, ответы будут повторяться в 2^2 (выборка вариантов переменной (0, 1) в степени количества переменных) = 4 раза.

Составим таблицу истинности.

x	y	z	F_1	F_2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	0

Результат функций различается только в одном случае.

Так как у нас есть переменные, не влияющие на результат, но повторяющиеся значения функций четыре раза, умножим количество повторений на количество различающихся значений функций: $1 \cdot 4 = 4$.

Ответ: 4.

Задача II.1.3.6. Трасса (14 баллов)

Темы: программирование.

Условие

В новом современном городе строят новую современную скоростную трассу длиной s м. Ее необходимо оборудовать так, чтобы она могла выдерживать большое количество машин и чтобы она не создавала больших пробок и аварийных ситуаций. Поэтому было принято решение посмотреть на другой, аналогичный город с такой же успешной трассой и запросить с камер записи о том, сколько машин там фиксируется за день.

Всего с камер было получено n машин, и по каждой была информация во сколько она заезжает на трассу и с какой скоростью ехала в м/с. После получения этой информации было решено узнать максимальную нагрузку в какую-то из секунд на трассе. От этого значения они и хотят понимать, какую нагрузку должна выдерживать трасса. Вы, как опытный программист и сотрудник ИТ-отдела города, взялись за эту задачу.

Напишите программу, которая по этим данным опередит максимальную нагрузку на трассу в какую-то из секунд.

Формат входных данных

В первой строке входных данных записано два целых числа n ($1 \leqslant n \leqslant 10^5$) и s ($1 \leqslant s \leqslant 10^6$) — количество зафиксированных машин и длина трассы. В следующих n строках по два целых числа t ($1 \leqslant t \leqslant 10^6$) и v ($1 \leqslant v \leqslant s$) — время заезда на трассу и скорость на трассе в м/с соответственно.

Гарантируется, что длина трассы кратна каждой скорости во входных данных.

Формат выходных данных

Выведите одно число — максимальное количество машин на трассе в некоторую секунду.

Методика проверки

Первая машина заедет на третьей секунде и выйдет на 5: [3, 5).

Вторая машина заедет на второй секунде и выйдет на 8: [2, 8).

Третья машина заедет на первой секунде и выйдет на 13: [1, 13).

Четвертая машина заедет на пятой секунде и выйдет на 6: [5, 6).

Итого максимальное количество машин будет замечено на четвертой секунде. Одновременно на трассе будет первая, вторая и третья машины.

Примеры

Пример №1

Стандартный ввод
4 60 3 30 2 10 1 5 5 60
Стандартный вывод
3

Решение

Определим два события, которые у нас возможны в задаче:

- машина заехала на трассу в определенный момент времени, обозначим это как $+1$ машина;
- машина выехала с трассы в определенный момент времени, обозначим это как -1 машина.

Каждое событие мы можем без особых проблем сохранять в массив и после работать с ним.

Первое событие мы можем сохранить в массив как пару (время заезда, 1), где время заезда — параметр из входных данных, а 1 — это аналог +1, дающий нам сигнал, что на трассе появилась новая машина.

Второе событие мы можем сохранить в массив как пару (время выезда, −1), где время выезда — это сумма времени заезда на трассу и длины трассы, поделенной на скорость машины, а −1 — сигнал о том, что машина выехала с трассы (−1 машина).

Отсортируем массив по первому параметру пар чисел: временам заезда и выезда с трассы. Тем самым мы получим последовательность действий на трассе. Запускаем цикл по массиву и, если действие равняется заезду машины, увеличиваем количество машин на трассе, в ином случае — уменьшаем.

Так как у нас действия помечены как 1 (+1) и −1, можем в количество машин добавлять именно их. Заведем отдельно переменную, в которой будем хранить максимальное количество машин, которое было за все время на трассе. Его мы будем обновлять после каждого действия следующий способом: если количество машин на трассе в определенный момент времени больше, чем записано в переменной, то обновляем ее значение.

По окончании цикла выводим максимальное количество машин, которое было зафиксировано.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n, s = map(int, input().split())
arr = []
for i in range(n):
    time_in, speed = map(int, input().split())
    arr.append((time_in, 1))
    arr.append((time_in + s // speed, -1))
arr.sort()
max_cars_per_sec = 0
cur_cars_per_sec = 0
for i in range(2 * n):
    cur_cars_per_sec += arr[i][1]
    max_cars_per_sec = max(max_cars_per_sec, cur_cars_per_sec)
print(max_cars_per_sec)
```

Задача II.1.3.7. Игра +1 (14 баллов)

Темы: программирование.

Условие

Игра +1 — это современная, набирающая популярность игра в просторах интернета. Она завлекает всех своей простотой и желанием добиваться высоких результатов за минимальное количество действий.

Давайте немного познакомимся с ее сутью. Нам выложено некое поле размером $1 \times n$ клеток. В каждой клетке записано некоторое число.

Если на поле есть два одинаковых числа, то их можно объединить. Операция объединения удаляет два числа, над которыми была произведена операция, а также создает новое число (на одной из освободившийся клетке), которое на единицу больше удаленных.

Например, если была объединена пара двоек, то они будут удалены, а новым числом будет 3.

Игра считается законченной, если было получено некоторое загаданное число m или на поле больше нет одинаковых чисел.

Как мы обсудили ранее, игроки хотят побеждать за минимальное количество действий. Так как единственное действие, которое существует — это объединение, то, соответственно, побеждать за минимальное количество объединений. Один из игроков решил считерить и попросил вас написать ему программу, которая, исходя из поля, будет определять, сколько минимально чисел с первоначального поля надо объединить между собой, чтобы закончить игру, или выведите -1, если невозможно собрать нужное число.

В ответе не учитывайте объединения между новыми числами, которые получаются после объединения.

Формат входных данных

В первой строке входных данных записано два целых числа n ($1 \leqslant n \leqslant 10^6$) и m ($2 \leqslant m \leqslant 100$) — количество чисел и цель, которую надо получить.

Во второй строке записано n целых чисел a_i ($1 \leqslant a_i < 100$, $max(a) < m$) — числа на поле.

Формат выходных данных

Выведите одно число — минимальное количество чисел из первоначального поля, которое надо объединить для получения нужного результата.

Примеры

Пример №1

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

Пояснение к примеру

Для получения результата 4 достаточно выбрать $[2, 3, 2][2, 3, 2]$:

- четверки суммируются как одинаковая пара чисел, получая новое число: $[3, 3][3, 3]$;

- восьмерки суммируются как одинаковая пара чисел, получая новое число: [4][4].

Решение

Создадим список/словарь, в котором подсчитаем количество каждого числа, которые нам даны на вход.

Подсчет будем ввести следующим образом: в качестве индекса/ключа будем использовать само число, а в качестве значения — сколько раз оно встретилось.

После этого заведем переменную, в которой будем хранить число, которое мы хотим достичь на определенном шагу, а также необходимое количество этих чисел.

В начальный момент времени у нас значение этой переменной равно конечному результату, который дан во входных данных, а необходимое количество — 1 (само число).

Запускаем цикл, который будет работать до тех пор, пока не соберем все числа, либо пока число, которое мы хотим достичь, не дойдет до нуля (несуществующего числа).

На каждом шагу проверяем через список/словарь, есть ли у нас необходимое количество выбранного числа. Если их достаточно, добавляем недостающее количество чисел и указываем, что собрали все числа (указывает, что нужно 0 чисел). В ином случае отнимаем часть, которую мы можем покрыть, и оставшееся необходимое количество чисел умножаем на два (так как чтобы собрать число x, необходимо два числа $x-1$, описано подробнее в условии), а также меняем нынешнее число, которые нам нужно собрать, уменьшая его значение на 1.

Помимо этого мы ведем на каждом шагу подсчет того, сколько чисел мы взяли, для этого заранее заведем переменную.

После окончания работы циклы проверяем: если остались числа, которые мы не смогли набрать, выводим -1, в ином случае выводим переменную, в которой мы ввели подсчет, сколько чисел взято на каждом шагу.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n, goal = map(int, input().split())
arr = list(map(int, input().split()))
counted = [0] * 101
for value in arr:
    counted[value] += 1
current_need = 1
current_goal = goal
res = 0
while current_need > 0 and current_goal > 0:
    res += min(current_need, counted[current_goal])
    current_need = max(0, current_need - counted[current_goal]) * 2
    current_goal -= 1
if current_need > 0:
    print(-1)
```

```
15  else:
16      print(res)
```

Предметный тур. Математика

Первая волна. Задачи 8–9 класса

Задача II.2.1.1. (15 баллов)

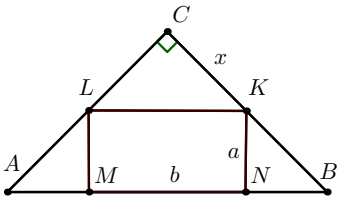
Темы: планиметрия.

Условие

Прямоугольник $MNKL$ вписан в равнобедренный прямоугольный треугольник ABC таким образом, что две его вершины M и N лежат на гипотенузе AB, а две K и L — на катетах BC и AC соответственно. Найдите гипотенузу треугольника ABC, если площади треугольников AML и CLK соответственно равны S_1 и S_2.

Формат ответа: приближенный ответ с точностью до 0,01.

Решение



Поскольку треугольник ABC равнобедренный и прямоугольный, то углы при вершинах A и B равны по $45°$. В силу того, что $MNKL$ — прямоугольник, имеем $\angle AML = 90°$. Следовательно, треугольник AML — прямоугольный равнобедренный. Пусть $AM = ML = a$. Тогда

$$S_1 = \frac{1}{2}a^2.$$

Отсюда $a = \sqrt{2S_1}$.

Так как $\angle CLK = 180° - \angle MLA - \angle MLK = 180° - 45° - 90° = 45°$, то треугольник CLK — прямоугольный равнобедренный. Пусть $CL = CK = x$. Тогда

$$S_2 = \frac{1}{2}x^2.$$

Пусть $LK = b$. По теореме Пифагора $x^2 + x^2 = b^2$. Тогда $x^2 = \frac{b^2}{2}$. Следовательно,

$$S_2 = \frac{1}{2} \cdot \frac{b^2}{2} = \frac{b^2}{4}.$$

Тогда $b = 2\sqrt{S_2}$.

Получаем

$$AB = 2a + b = 2\sqrt{2S_1} + 2\sqrt{S_2} = 2(\sqrt{2S_1} + \sqrt{S_2}).$$

Погрешность 0,01.

Варианты

$S_1 = 3, 4, \ldots, 10$; $S_2 = 11, 12, \ldots, 20$.

Ответ: $2(\sqrt{2S_1} + \sqrt{S_2})$.

Задача II.2.1.2. (20 баллов)

Темы: делимость и остатки.

Условие

Николай решил расставить оловянных солдатиков в колонну по a в ряд, однако ему не хватило k штук, чтобы заполнить последний ряд. Тогда он перестроил солдатиков по b в ряд, при этом ему снова не хватило k солдатиков, чтобы заполнить последний ряд. Наконец он построил их в колонну по c в ряд, и опять ему не хватило k игрушек, чтобы заполнить последний ряд. Какое наименьшее количество солдатиков может быть у Николая, если известно, что их не менее M штук?

Решение

Пусть N — количество солдатиков. Тогда по условию задачи $N + k$ делится на a, на b и на c. Поэтому $N + k$ делится на $\text{НОК}(a, b, c)$. Имеем

$$N = l \cdot \text{НОК}(a, b, c) - k \geqslant M,$$

где $l \in \mathbb{N}$. Наименьшее значение N соответствует наименьшему возможному значению l, равному

$$\left\lceil \frac{M + k}{\text{НОК}(a, b, c)} \right\rceil,$$

где $\lceil \cdot \rceil$ — операция округления вверх до ближайшего целого. Значит, наименьшее количество солдатиков

$$N = \left\lceil \frac{M + k}{\text{НОК}(a, b, c)} \right\rceil \cdot \text{НОК}(a, b, c) - k.$$

Погрешность 0.

Варианты

$a = 6, 7, 8$; $b = 9, 10, 11$; $c = 12, 13, 14$; $k = 1, 2, \ldots, 5$, $M = 200, 250, 300$.

Ответ: $\left\lceil \frac{M + k}{\text{НОК}(a, b, c)} \right\rceil \cdot \text{НОК}(a, b, c) - k.$

Задача II.2.1.3. (20 баллов)

Темы: теория множеств, логика.

Условие

Чтобы обсудить последние новости, n приятелей решили встретиться в кафе. Каждый заказал себе лимонад, но не более двух бокалов. Причем те, кто заказал два бокала, выбрали разные вкусы. В кафе подавали лимонады трех различных вкусов. Оказалось, что для любой пары приятелей вкусы совпали хотя бы для одного бокала, а самый популярный вкус выбрали ровно k приятелей. Определите наименьшее возможное значение k.

Примечание: самых популярных вкусов может быть несколько, когда каждый из этих вкусов выбирается одинаковым количеством приятелей.

Решение

Оценка. Докажем, что самый популярный вкус выбрали не менее $\lceil \frac{2}{3}n \rceil$ приятелей (здесь $\lceil \cdot \rceil$ — операция округления вверх до ближайшего целого). Составим таблицу вида.

	1	2	...	n
A	0	1	...	1
B	1	0	...	1
C	1	1	...	0

Здесь A, B, C — вкусы лимонада. На пересечении i-й строки и j-го столбца стоит 1, если и только если j-й приятель выбрал i-й вкус.

Допустим A — самый (один из самых) популярный вкус. Если в строке A все единицы, то доказывать нечего.

Пусть в строке A есть хотя бы один ноль. Пусть, например, он стоит в первом столбце. Тогда первый приятель выбрал хотя бы один из оставшихся вкусов. Пусть, например, он выбрал B. Если первый приятель заказал только один бокал лимонада, то по условию все остальные приятели тоже выбрали бокал лимонада вкуса B. Но тогда получается, что вкус B популярнее вкуса A. Это противоречие показывает, что первый приятель заказал лимонады обоих вкусов, B и C.

Поскольку у первого приятеля есть хотя бы один общий вкус с каждым из остальных, и каждый заказал не более двух бокалов, то в каждом столбце таблицы стоит ровно две единицы. Тогда во всей таблице записано $2 \cdot n$ единиц. Но тогда в строке A записано не менее $\frac{2n}{3}$ единиц, так как иначе во всей таблице будет менее $2n$ единиц. Поскольку количество единиц в строке A — целое число, то это количество не меньше $\lceil \frac{2}{3}n \rceil$.

Пример. Учитывая, что $n = 3k + 1$, составим такую таблицу.

	1	...	$k+1$	$k+2$	...	$2k+1$	$2k+2$	...	$3k+1$
A	1	...	1	1	...	1	0	...	0
B	0	...	0	1	...	1	1	...	1
C	1	...	1	0	...	0	1	...	1

Здесь два самых популярных вкуса — A и C. В соответствующих строках записано $2k + 1 = \lceil \frac{2}{3}(3k + 1) \rceil = \lceil \frac{2}{3}n \rceil$ единиц.

Погрешность 0.

Варианты

$n = 10, 13, \ldots, 43$.

Ответ: $\left\lceil \frac{2}{3}n \right\rceil$.

Задача II.2.1.4. (20 баллов)

Темы: классическая вероятность.

Условие

Случайным образом выбираются 4 различные вершины правильного n-угольника (любой выбор равновозможен). Какова вероятность того, что выбранные вершины образуют прямоугольник?

Дайте ответ в процентах с точностью до 0,01.

Решение

Количество способов выбрать 4 вершины равно C_n^4.

Теперь подсчитаем количество возможных прямоугольников. Диагональ одного такого прямоугольника совпадает с диаметром окружности, описанной около n-угольника, поскольку на диагональ опирается угол в 90° с вершиной, лежащей на окружности. Таким образом, каждому прямоугольнику взаимно однозначно сопоставляются две пары диаметрально противоположных вершин n-угольника. Всего таких пар $C_{n/2}^2$.

Следовательно, искомая вероятность равна:

$$\frac{C_{n/2}^2}{C_n^4} = \frac{\frac{n}{2}\left(\frac{n}{2}-1\right)}{2} \cdot \frac{4!}{n(n-1)(n-2)(n-3)} = \frac{3}{(n-1)(n-3)}.$$

Для получения количества процентов остается умножить результат на 100.

Погрешность 0,01.

Варианты

$n = 8, 10, \ldots, 60$.

Ответ: $\dfrac{300}{(n-1)(n-3)}$.

Задача II.2.1.5. (25 баллов)

Темы: алгебра, неравенства.

Условие

При каком наибольшем значении параметра a неравенство:

$$x^2 - 14xy \geqslant -50y^2 + \frac{2}{\beta}ay - 529$$

верно для всех вещественных значений x и y?

Решение

Выделим полные квадраты:

$$x^2 - 14xy \geqslant -50y^2 + \frac{2}{\beta}ay - 529 \iff$$
$$\iff (x^2 - 14xy + 49y^2) + \left(y^2 - \frac{2}{\beta}ay + \frac{a^2}{\beta^2}\right) \geqslant \frac{a^2}{\beta^2} - 529 \iff$$
$$\iff (x - 7y)^2 + \left(y - \frac{a}{\beta}\right)^2 \geqslant \left(\frac{a}{\beta} - 23\right)\left(\frac{a}{\beta} + 23\right).$$

Заметим, что левая часть неотрицательна при всех значениях x и y. Поэтому если правая часть неположительна, то исходное неравенство верно при всех $x, y \in \mathbb{R}$. Если же правая часть строго больше нуля, то при $y = \frac{a}{\beta}$, $x = 7y$ исходное неравенство нарушается.

Таким образом, требуется найти наибольшее значение a, при котором

$$\left(\frac{a}{\beta} - 23\right)\left(\frac{a}{\beta} + 23\right) \leqslant 0.$$

Имеем $a = 23\beta$.

Погрешность 0.

Варианты

$\beta = 3, 5, 7, \ldots, 21$.

Ответ: 23β.

Первая волна. Задачи 10–11 класса

Задача II.2.2.1. (15 баллов)

Темы: алгебра, квадратный трехчлен.

Условие

Найдите расстояние между точками пересечения графиков двух различных квадратных трехчленов, если они отличаются лишь перестановкой старшего коэффициента и свободного члена, а многочлен, равный их сумме, имеет единственный корень и пересекает ось ординат в точке l. Формат ответа: приближенный с точностью до 0,01.

Решение

Пусть f и g — данные квадратные трехчлены,

$$f(x) = ax^2 + bx + c, \quad g(x) = cx^2 + bx + a.$$

Их сумма $h(x) = (a + c)x^2 + 2bx + (a + c)$.

Найдем точки пересечения графиков f и g. Имеем:

$$f(x) = g(x) \iff (a - c)x^2 = a - c.$$

Так как по условию трехчлены f и g различны, то $a \neq c$. Поэтому $x = \pm 1$. Соответствующие ординаты: $y = a + b + c$ для $x = 1$ и $y = a - b + c$ для $x = -1$.

Расстояние между точками пересечения равно:

$$\rho = \sqrt{(1 - (-1))^2 + (a + b + c - (a - b + c))^2} = 2\sqrt{1 + b^2}.$$

По условию трехчлен h имеет единственный корень. Следовательно, его дискриминант, разделенный на 4, равен нулю:

$$b^2 - (a + c)^2 = 0.$$

Отсюда $b^2 = (a + c)^2$. По условию также имеем $h(0) = l$, то есть $a + c = l$.

Таким образом,

$$\rho = 2\sqrt{1 + (a + c)^2} = 2\sqrt{1 + l^2}.$$

Погрешность 0,01.

Варианты

$l = 2, 3, \ldots, 50$.

Ответ: $2\sqrt{1 + l^2}$.

Задача II.2.2.2. (20 баллов)

Темы: текстовая задача.

Условие

Бригада комбайнеров, имеющих одинаковые машины, обрабатывает два поля одинаковой площади. На первом поле комбайны начинают работу по очереди через равные промежутки времени, и к моменту начала работы последнего остается неубранной $1/n$ часть поля. После уборки первого поля бригада приступает к уборке второго. При этом промежутки времени между началом работы комбайнов становятся на $p\%$ больше, чем при работе на первом поле. Во сколько раз время уборки второго поля больше времени уборки первого?

Формат ответа: приближенный с точностью до $0,01$.

Решение

Обозначим: k — количество комбайнов, x — производительность одного комбайна, t — время работы бригады при обработке первого поля до начала работы последнего комбайна, T — время уборки первого поля, τ — время уборки второго поля. Площадь каждого поля примем за 1.

К моменту времени t на первом поле была убрана площадь, равная $1-1/n$. При этом первый комбайн обработал площадь, равную xt, второй — $x\left(t-\frac{t}{k-1}\right)$, третий — $x\left(t-\frac{2t}{k-1}\right)$ и т. д. Поэтому

$$\begin{cases} xt + x\left(t-\frac{t}{k-1}\right) + x\left(t-\frac{2t}{k-1}\right) + \ldots + x\left(t-\frac{(k-2)t}{k-1}\right) = 1-\frac{1}{n}, \\ xk(T-t) = \frac{1}{n}. \end{cases}$$

Пользуясь формулой для суммы арифметической прогрессии, для левой части первого уравнения имеем

$$\begin{aligned} &xt + x\left(t-\frac{t}{k-1}\right) + x\left(t-\frac{2t}{k-1}\right) + \ldots + x\left(t-\frac{(k-2)t}{k-1}\right) = \\ &= xt(k-1) - \frac{xt}{k-1}\left(1+2+\ldots+(k-2)\right) = \\ &= xt(k-1) - \frac{xt}{k-1}\frac{k-1}{2}(k-2) = \frac{xtk}{2}. \end{aligned}$$

Тогда $t = \left(1-\frac{1}{n}\right)\frac{2}{xk}$.

Подставляя во второе уравнение системы, получаем

$$xk\left(T - \left(1-\frac{1}{n}\right)\frac{2}{xk}\right) = \frac{1}{n}.$$

Отсюда

$$T = \frac{1}{nxk} + \frac{2(n-1)}{nxk} = \frac{2n-1}{nxk}.$$

По условию промежутки времени между началом работы двух последующих комбайнов на втором поле равно $\frac{at}{k-1}$, где $a = 1+\frac{p}{100}$. Тогда для второго поля имеем

$$x\tau + x\left(\tau - a\frac{t}{k-1}\right) + x\left(\tau - a\frac{2t}{k-1}\right) + \ldots + x\left(\tau - a\frac{(k-1)t}{k-1}\right) = 1.$$

Используя формулу для суммы арифметической прогрессии, получаем

$$x\tau k - \frac{xat}{k-1}\left(1+2+..+(k-1)\right) = x\tau k - \frac{xat}{k-1}\frac{k}{2}(k-1) = xk\left(\tau - \frac{at}{2}\right).$$

Поэтому, учитывая ранее найденное значение $t = \left(1-\frac{1}{n}\right)\frac{2}{xk}$, находим

$$\tau = \frac{1}{xk} + \frac{at}{2} = \frac{1}{xk} + \frac{a}{2}\left(1-\frac{1}{n}\right)\frac{2}{xk} = \frac{1}{xk}\left(1+a\left(1-\frac{1}{n}\right)\right).$$

Теперь вычислим отношение времени работы бригады на втором поле ко времени работы на первом:

$$\frac{\tau}{T} = \frac{1}{xk}\left(1+a\left(1-\frac{1}{n}\right)\right)\cdot\frac{nxk}{2n-1} = \frac{n+a(n-1)}{n}\frac{n}{2n-1} = \frac{n+a(n-1)}{2n-1}.$$

Подставляя значение a, находим

$$\frac{\tau}{T} = \frac{n + (1 + \frac{p}{100})(n-1)}{2n-1} = \frac{2n - 1 + \frac{p(n-1)}{100}}{2n-1} = 1 + \frac{p(n-1)}{100(2n-1)}.$$

Погрешность 0,01.

Варианты

$p = 10, 11, \ldots, 30$; $n = 5, 6, \ldots, 15$; $k = 16, 17, \ldots, 20$.

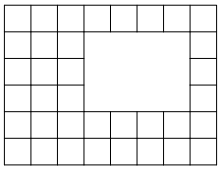
Ответ: $1 + \dfrac{p(n-1)}{100(2n-1)}$.

Задача II.2.2.3. (20 баллов)

Темы: графы.

Условие

Рыболовная сеть имеет форму прямоугольника размера $n \times k$ клеток. Внутри сети имеется прямоугольная дыра размером $l \times m$ клеток (внешняя граница сети цела). Какое наибольшее число нитей, соединяющих узлы сети, можно перерезать так, чтобы сеть не распалась на части?



Решение

Представим рыболовную сеть в виде графа, в котором вершины — узлы сети, а ребра — соединяющие их нити.

Для того чтобы сеть не распалась на части, граф должен быть связным. Связный граф с наименьшим числом ребер — это дерево, в котором, как известно, число ребер на единицу меньше числа вершин.

Подсчитаем число вершин в графе, соответствующем данной в условии рыболовной сети. Если бы дыры не было, то всего было бы $(n+1)(k+1)$ вершин. Из-за наличия дыры в графе отсутствует $(l-1)(m-1)$ вершин. Таким образом, наименьшее число нитей, необходимое для того, чтобы сеть не распалась на части, равно $(n+1)(k+1) - (l-1)(m-1) - 1$.

Подсчитаем количество ребер. Сеть без дыры можно представить составленной из уголков в виде буквы L и двух отрезков — верхней и правой границы. Тогда

число ребер в случае отсутствия дыры равно $2nk + k + n$. Из-за наличия дыры в графе отсутствует $2lm - l - m$ ребер. Значит, в графе всего ребер

$$2nk + k + n - (2lm - l - m).$$

Таким образом, для получения дерева необходимо перерезать количество нитей, равное

$$2nk + k + n - (2lm - l - m) - ((n + 1)(k + 1) - (l - 1)(m - 1) - 1) = kn - lm + 1.$$

Погрешность 0.

Варианты

$n = 200, 205, \ldots, 250$; $k = 300, 305, \ldots, 350$;

$l = 50, 55, \ldots, 100$; $m = 150, 155, \ldots, 200$.

Ответ: $kn - lm + 1$.

Задача II.2.2.4. (20 баллов)

Темы: геометрическая вероятность, выпуклый четырехугольник.

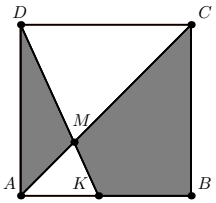
Условие

Сторона квадрата $ABCD$ равна $a\sqrt{2}$. На диагонали AC отмечена точка M на расстоянии b от точки A. Внутри квадрата случайно выбирается точка X. Вычислите вероятность того, что точки C, D, M и X, взятые в некотором порядке, образуют вершины выпуклого четырехугольника.

Ответ дайте в процентах с точностью до 0,01.

Решение

Докажем, что четырехугольник получится выпуклым, если точка X попадет в область, закрашенную на рисунке.



Напомним, что четырехугольник является выпуклым, если он лежит по одну сторону от каждой прямой, проходящей через две его соседние вершины. Разберем четыре случая.

1. Возьмем точку X внутри треугольника MCD и соединим ее отрезком с одной из вершин этого треугольника, построив тем самым одну из сторон четырехугольника. Тогда две другие вершины треугольника окажутся по разные стороны от прямой, проходящей через построенную сторону. Значит, четырехугольник будет невыпуклым.
2. Возьмем точку X внутри треугольника AKM. В этом случае получается невыпуклый четырехугольник, поскольку:
   - если MX — сторона четырехугольника, то D и C лежат по разные стороны от прямой MX;
   - если MD — сторона четырехугольника, то C и X лежат по разные стороны от прямой MD;
   - если MC — сторона четырехугольника, то D и X лежат по разные стороны от прямой MC.
3. Возьмем точку X внутри четырехугольника $KBCM$. Нетрудно видеть, что четырехугольник $XCDM$ выпуклый.
4. Возьмем точку X внутри треугольника AMD. Нетрудно видеть, что четырехугольник $XMCD$ выпуклый.

Искомая вероятность — отношение площади закрашенной области к площади квадрата. Найдем площадь незакрашенной области.

Высота треугольника MCD, проведенная из точки D, — это половина диагонали квадрата, поэтому она равна $\frac{a\sqrt{2}}{\sqrt{2}} = a$. Тогда площадь треугольника MCD равна

$$S_{MCD} = \frac{1}{2}a \cdot (2a - b).$$

Треугольники AKM и MCD подобны (по двум углам). Тогда их площади относятся как квадрат отношения сторон. Следовательно, площадь S_{AKM} треугольника AKM равна

$$S_{AKM} = S_{MCD}\left(\frac{AM}{MC}\right)^2 = \frac{a}{2} \cdot (2a - b)\left(\frac{b}{2a - b}\right)^2 = \frac{ab^2}{2(2a - b)}.$$

Тогда искомая вероятность p равна

$$p = \frac{S_{ABCD} - (S_{AKM} + S_{MCD})}{S_{ABCD}} = \frac{2a^2 - \left(\frac{ab^2}{2(2a-b)} + \frac{a(2a-b)}{2}\right)}{2a^2} = \frac{2a^2 - b^2}{2a(2a - b)}.$$

Для получения ответа в процентах остается умножить полученное выражение на 100.

Погрешность 0,01.

Варианты

$a = 11, 12, \ldots, 30$; $b = 1, 2, \ldots, 10$.

Ответ: $\dfrac{50(2a^2 - b^2)}{a(2a - b)}$.

Задача II.2.2.5. (25 баллов)

Темы: теория множеств, биекция, комбинаторика.

Условие

На дворовой площадке устраивается турнир по пионерболу. В турнире участвуют n ребят, среди них соседи Саша и Маша. Для турнира составляются всевозможные команды, которые можно образовать из ребят, но так, чтобы в каждой команде играли как минимум два человека. Каждая команда играет в турнире ровно один раз. Сколько матчей Саша и Маша будут соперниками?

Решение

Занумеруем участников от 1 до n. Пусть Саша имеет номер 1, а Маша — номер 2. Каждой команде можно поставить в соответствие строку из нулей и единиц, поставив 1 на позиции k, если k-й участник входит в команду, и 0 — иначе.

Всего команд столько же, сколько строк длины n из нулей и единиц, в которых хотя бы две единицы (по условию в команде минимум два человека), но при этом не более $n-2$ единицы (если единиц больше, то невозможно подобрать команду соперников). Значит, вычитая из общего числа строк строки, состоящие только из нулей, только из единиц, содержащих одну единицу и содержащих $n-1$ единицу, получаем, что всего команд:

$$2^n - 1 - 1 - n - n = 2^n - 2n - 2.$$

В каждом матче участвуют две команды, поэтому матчей в два раза меньше числа команд: $2^{n-1} - n - 1$.

Рассмотрим участника под номером k. Подсчитаем количество команд, в которых он участвует — количество строк с единицей на k-й позиции таких, что на остальных позициях может быть что угодно, кроме всех нулей, всех единиц либо одного нуля. Всего таких строк:

$$2^{n-1} - 1 - 1 - (n-1) = 2^{n-1} - n - 1.$$

Получили, что количество матчей совпадает с количеством команд, в которых участвует k-й игрок. Но каждая команда играет ровно один раз. Таким образом, k-й игрок участвует в каждом матче. А значит, в силу произвольного выбора k и каждый игрок участвует в каждом матче. То есть строка из нулей и единиц однозначно задает не только первую команду (с помощью единиц), но и вторую команду (с помощью нулей).

Таким образом, матчи, в которых Саша и Маша — соперники, задаются строками, в которых цифры на позициях 1 и 2 разные. Каждому матчу соответствуют две строки, получаемые одна из другой заменой единиц нулями и наоборот. Поэтому достаточно подсчитать количество строк, в которых на первой позиции стоит единица, на второй — ноль, а на оставшихся что угодно, кроме всех единиц либо всех нулей. Таких строк:

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

Погрешность 0.

Варианты

$n = 10, 11, \ldots, 20.$

Ответ: $2^{n-2} - 2$.

Вторая волна. Задачи 8–9 класса

Задача II.2.3.1. (15 баллов)

Темы: текстовая задача.

Условие

Школе требуется N новых парт. Заказ на их изготовление получили три мебельных завода. Первый завод за три дня может выпустить n парт, второй —за четыре дня выпускает $p\%$ от того количества, которое первый и третий выпускают за два дня. Третий завод за 5 дней выпускает m парт. За сколько дней будет выполнен заказ? Ответ округлите вверх до ближайшего целого.

Решение

Обозначим через x, y, z производительности в ед./сут. для первого, второго и третьего заводов соответственно. Пусть t — время выполнения заказа.

По условию задачи имеем

$$\begin{cases} x = \frac{n}{3}, \\ z = \frac{m}{5}, \\ y = \frac{p}{100}\frac{2(x+z)}{4}. \end{cases}$$

Тогда

$$N = (x + y + z)t = \left(\frac{n}{3} + \frac{m}{5} + \frac{p}{100}\frac{n/3 + m/5}{2}\right)t.$$

Следовательно,

$$t = \frac{N}{\left(\frac{n}{3} + \frac{m}{5}\right)\left(1 + \frac{p}{200}\right)}.$$

Погрешность 0.

Варианты

$N = 600, 650, 700$; $n = 15, 20, 25$; $m = 35, 40, 45, 50$; $p = 20, 25, \ldots, 80$.

Ответ: $\left\lceil \frac{N}{\left(\frac{n}{3} + \frac{m}{5}\right)\left(1 + \frac{p}{200}\right)} \right\rceil$ (здесь $\lceil \cdot \rceil$ — округление вверх до ближайшего целого).

Задача II.2.3.2. (20 баллов)

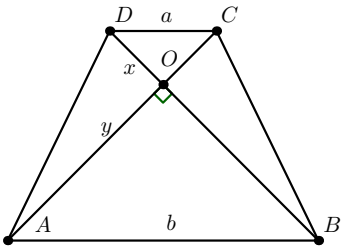
Темы: планиметрия, трапеция.

Условие

Меньшее основание равнобедренной трапеции, диагонали которой взаимно перпендикулярны, равно a. Найдите большее основание трапеции, если ее площадь равна S.

Формат ответа: приближенный с точностью до 0,01.

Решение



Через $S(X)$ обозначим площадь фигуры X. Имеем

$$S = S(ABCD) = S(DOC) + S(ABO) + 2S(AOD) = \frac{1}{2}x^2 + \frac{1}{2}y^2 + 2 \cdot \frac{1}{2}xy.$$

По теореме Пифагора для треугольника DOC будет $x^2 + x^2 = a^2$, тогда $x^2 = \frac{a^2}{2}$. Аналогично из треугольника ABO получаем $y^2 = \frac{b^2}{2}$.

Следовательно,

$$S = \frac{1}{2}\frac{a^2}{2} + \frac{1}{2}\frac{b^2}{2} + \frac{a}{\sqrt{2}}\frac{b}{\sqrt{2}}.$$

Перенося все слагаемые в одну часть и умножая на 4, получаем

$$b^2 + 2ab + a^2 - 4S = 0.$$

Решая это квадратное уравнение и оставляя только положительный корень, находим

$$b = -a + 2\sqrt{S}.$$

Замечание. Задачу можно решить быстрее, если знать свойство равнобедренной трапеции со взаимно перпендикулярными диагоналями: высота h в такой трапеции равна средней линии. Поэтому

$$S = \frac{1}{2}(a+b)h = \left(\frac{a+b}{2}\right)^2.$$

Отсюда получается тот же ответ.

Погрешность 0,01.

Варианты

$a = 3, 4, \ldots, 10$; $S = 110, 120, \ldots, 200$.

Ответ: $2\sqrt{S} - a$.

Задача II.2.3.3. (20 баллов)

Темы: теория множеств, комбинаторика, делимость.

Условие

В соревнованиях по шахматам участвует N команд. Организаторы соревнований придумали следующий способ разбиения команд на группы. Каждой команде присвоен уникальный номер от 1 до N. В первую группу входят команды, номера которых делятся на a, во вторую — те из оставшихся, номера которых делятся на b, в третью — те из оставшихся, номера которых делятся на c, а в четвертую попадают все остальные. Сколько команд будут соревноваться между собой в четвертой группе?

Решение

Пусть A — множество номеров, делящихся на a, B — делящихся на b, C — делящихся на c. Обозначим через $N(X)$ количество элементов в множестве X, $\lfloor x \rfloor$ — округление вниз числа x.

В четвертую группу попадут такие команды, номера которых не делятся ни на одно из чисел a, b или c. Согласно формуле включений и исключений количество N' таких команд равно

$$N' = N - N(A) - N(B) - N(C) + N(A \cap B) + N(A \cap C) + N(B \cap C) - N(A \cap B \cap C).$$

Имеем

$$N(A) = \left\lfloor \frac{N}{a} \right\rfloor, \quad N(B) = \left\lfloor \frac{N}{b} \right\rfloor, \quad N(C) = \left\lfloor \frac{N}{c} \right\rfloor.$$

Далее

$$N(A \cap B) = \left\lfloor \frac{N}{\mathrm{HOK}(a,b)} \right\rfloor, \quad N(A \cap C) = \left\lfloor \frac{N}{\mathrm{HOK}(a,c)} \right\rfloor, \quad N(B \cap C) = \left\lfloor \frac{N}{\mathrm{HOK}(b,c)} \right\rfloor.$$

Наконец, для пересечения всех трех множеств получаем

$$N(A \cap B \cap C) = \left\lfloor \frac{N}{\mathrm{HOK}(a,b,c)} \right\rfloor.$$

Таким образом, в четвертой группе соревнуются команды в количестве

$$N' = N - \left\lfloor \frac{N}{a} \right\rfloor - \left\lfloor \frac{N}{b} \right\rfloor - \left\lfloor \frac{N}{c} \right\rfloor + \\ + \left\lfloor \frac{N}{\mathrm{HOK}(a,b)} \right\rfloor + \left\lfloor \frac{N}{\mathrm{HOK}(a,c)} \right\rfloor + \left\lfloor \frac{N}{\mathrm{HOK}(b,c)} \right\rfloor - \left\lfloor \frac{N}{\mathrm{HOK}(a,b,c)} \right\rfloor.$$

Погрешность 0.

Варианты

$N = 201, 202, \ldots, 209$; $a = 12, 20$; $b = 6, 18$; $c = 9, 10$.

Ответ:

$$N - \left\lfloor \frac{N}{a} \right\rfloor - \left\lfloor \frac{N}{b} \right\rfloor - \left\lfloor \frac{N}{c} \right\rfloor + \\ + \left\lfloor \frac{N}{\text{НОК}(a,b)} \right\rfloor + \left\lfloor \frac{N}{\text{НОК}(a,c)} \right\rfloor + \left\lfloor \frac{N}{\text{НОК}(b,c)} \right\rfloor - \left\lfloor \frac{N}{\text{НОК}(a,b,c)} \right\rfloor.$$

Задача II.2.3.4. (20 баллов)

Темы: классическая вероятность, комбинаторика.

Условие

На клетчатом листе бумаги размера n клеток в высоту и m клеток в ширину случайно закрашивают 3 клетки (любой выбор клеток равновозможен). Какова вероятность того, что для каждой закрашенной клетки будет также закрашена хотя бы одна соседняя, имеющая с ней общую сторону?

Дайте ответ в процентах с точностью до 0,01.

Решение

На листе nm клеток, поэтому число способов выбрать 3 из них равно C_{nm}^3.

Всевозможные расположения закрашенных клеток, когда каждая клетка имеет хотя бы одну соседнюю, тоже закрашенную, изображены на рисунке.

Теперь подсчитаем число способов расположить каждую такую фигуру на клетчатом листе. Для первой фигуры имеется $n(m-2)$ способов, для второй, третьей, четвертой и пятой — $(n-1)(m-1)$ способов, для последней — $(n-2)m$ способов. Значит, искомая вероятность равна

$$\frac{n(m-2) + 4(n-1)(m-1) + (n-2)m}{C_{nm}^3} = \frac{12(3nm - 3n - 3m + 2)}{nm(nm-1)(nm-2)}.$$

Для получения ответа в процентах остается умножить полученное выражение на 100.

Погрешность 0,01.

Варианты

$n = 6, 7, \ldots, 15$; $m = 6, 7, \ldots, 15$.

Ответ: $\dfrac{1200(3nm - 3n - 3m + 2)}{nm(nm-1)(nm-2)}$.

Задача II.2.3.5. (25 баллов)

Темы: алгебра, задача на максимум и минимум.

Условие

Найдите наименьшее значение выражения

$$\frac{(x^2 - ax + b)^2}{\left(x - \frac{a}{2}\right)^2}.$$

Решение

Заметим, что $x^2 - ax + b > 0$ при всех x. Тогда наименьшее значение выражения достигается в той же точке, что и для выражения

$$F = \frac{x^2 - ax + b}{\left|x - \frac{a}{2}\right|}.$$

Выделяя полный квадрат в числителе, получаем

$$\frac{x^2 - ax + b}{\left|x - \frac{a}{2}\right|} = \frac{\left|x - \frac{a}{2}\right|^2 + b - \frac{a^2}{4}}{\left|x - \frac{a}{2}\right|} = \left|x - \frac{a}{2}\right| + \frac{b - \frac{a^2}{4}}{\left|x - \frac{a}{2}\right|}.$$

Пусть $t = \left|x - \frac{a}{2}\right|$, $c = b - \frac{a^2}{4}$. Тогда

$$F = t + \frac{c}{t} = \sqrt{c}\left(\frac{t}{\sqrt{c}} + \frac{\sqrt{c}}{t}\right).$$

Сумма двух положительных взаимнообратных чисел не меньше 2, а значение 2 достигается, когда эти числа равны 1. Таким образом, наименьшее значение выражения F равно

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

Следовательно, наименьшее значение исходного выражение равно $4b - a^2$.

Погрешность 0.

Варианты

$a = 3, 4, \ldots, 10$; $b = 26, 27, \ldots, 50$.

Ответ: $4b - a^2$.

Вторая волна. Задачи 10–11 класса

Задача II.2.4.1. (15 баллов)

Темы: теория чисел, алгебра.

Условие

Число $\frac{n}{36^k}$ записали в 24-ичной системе счисления. Сколько знаков после запятой получилось?

Решение

Имеем $36 = 2^2 \cdot 3^2$, поэтому $36^k = 2^{2k} \cdot 3^{2k}$. Так как $24 = 2^3 \cdot 3$, то, умножив числитель и знаменатель на 2^{4k}, получим

$$\frac{n}{36^k} = \frac{2^{4k}n}{2^{6k} \cdot 3^{2k}} = \frac{2^{4k}n}{24^{2k}}.$$

Значит, данное число имеет $2k$ или меньше знаков после запятой. Поскольку n не делится на 3, то $2^{4k}n$ не делится на 24, а значит, число имеет ровно $2k$ знаков после запятой в 24-ичной системе счисления.

Погрешность 0.

Варианты

$n = 109, 112, \ldots, 169$; $k = 3, 4, \ldots, 15$.

Ответ: $2k$.

Задача II.2.4.2. (20 баллов)

Темы: текстовая задача, логика.

Условие

Пастбище для овец ограждено забором в форме пятиугольника, в вершинах которого вбиты столбы. На территории пастбища вбили еще n столбов. Некоторые столбы соединили между собой непересекающимися бревнами так, что все пастбище разбилось на пятиугольные огражденные участки. Сколько таких участков получилось?

Решение

Рассмотрим граф, в котором вершины — столбы, вбитые внутри и на границе пастбища. Между вершинами проведено ребро, если соответствующие столбы соединены ограждением. Прямолинейные части забора по условию не пересекаются,

поэтому полученный граф — планарный. Следовательно, справедлива формула Эйлера В − Р + Г = 2, где В — число вершин, Р — число ребер, Г — число граней (грань — участок пастбища либо внешняя территория).

Число вершин известно: В = n+5. Число участков, на которые разбито пастбище, равно Г − 1.

Свяжем число ребер с числом граней. Назовем как-нибудь все грани и все ребра (можно, например, занумеровать их). Представим таблицу с двумя столбцами. Запишем в первый столбец все грани. Во втором столбце напротив соответствующей грани перечислим все ребра, которые ее ограничивают. Тогда каждое ребро встретится во втором столбце таблицы ровно два раза, так как каждое ребро отделяет две грани. Напротив каждой грани будет выписано 5 ребер. Таким образом, во втором столбце всего будет 5Г = 2Р записей. Поэтому Р = 5Г/2.

Возвращаясь к формуле Эйлера, находим

$$n + 5 - \frac{5\Gamma}{2} + \Gamma = 2.$$

Отсюда

$$\Gamma = \frac{2(n+3)}{3}.$$

Поэтому число участков, на которые разбито пастбище, равно $\frac{2(n+3)}{3} - 1$.

Погрешность 0.

Варианты

$n = 30, 33, \ldots, 120$.

Ответ: $\dfrac{2(n+3)}{3} - 1$.

Задача II.2.4.3. (20 баллов)

Темы: комбинаторика.

Условие

Туристическая компания предлагает экскурсионные программы по городу, в котором имеется N достопримечательностей. На ближайший сезон компании нужно составить k программ так, чтобы в каждой программе была хотя бы одна достопримечательность, и каждая достопримечательность города оказалась ровно в одной программе. Экскурсионные программы продаются независимо, поэтому их порядок неважен, а порядок обхода достопримечательностей в программе имеет значение (например, «Музей, Парк» и «Парк, Музей» — это разные программы; любая достопримечательность участвует в программе только один раз). Сколько вариантов организовать туристический сезон есть у компании?

Решение

Существует $N!$ способов выписать все N достопримечательностей. Обозначим j-ю достопримечательность через a_j.

Выпишем последовательность из всех N символов a_j в некотором порядке. Мы можем разбить выписанную последовательность на k групп, поставив $k-1$ перегородку между какими-нибудь буквами. Всего есть $N-1$ позиция, где можно поставить перегородку. Таким образом, существует C_{N-1}^{k-1} способов разбить выписанную последовательность на k групп.

Итак, имеется $N!C_{N-1}^{k-1}$ способов выписать все символы a_j вместе с разбиением их на k групп. Каждая такая строка соответствует некоторой организации туристического сезона. Однако по условию порядок групп (экскурсионных программ) неважен. Все описанные строки можно разбить на наборы по $k!$ строк, так что в каждом наборе строки отличаются только перестановкой групп. Каждый такой набор отвечает ровно одному способу организовать сезон. Следовательно, всего у туристической компании имеется

$$\frac{N!C_{N-1}^{k-1}}{k!}$$

вариантов.

Погрешность 0.

Варианты

$N = 10, 11, \ldots, 20$; $k = 4, 5, 6, 7$.

Ответ: $\dfrac{N!C_{N-1}^{k-1}}{k!}$.

Задача II.2.4.4. (20 баллов)

Темы: стереометрия, геометрическая вероятность.

Условие

Из отрезка $[0, a]$ случайно выбираются три вещественных числа. Найдите вероятность того, что наибольшее число отличается от наименьшего не менее, чем на b.

Выразите ответ в процентах с точностью до 0,01.

Решение

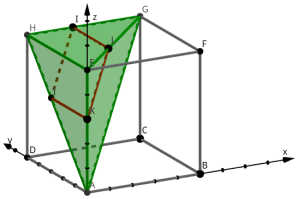
Выбирая три числа из $[0, a]$, назовем x — наименьшее из них, z — наибольшее, а y — лежащее между x и z. Выбор чисел x, y, z равносилен выбору точки $\alpha(x, y, z)$ в пространстве, удовлетворяющей условиям:

- так как $x, y, z \in [0, a]$, то точка α лежит в кубе $ABCDEFGH$ с координатами вершин: $A(0, 0, 0)$, $B(a, 0, 0)$, $C(a, a, 0)$, $D(0, a, 0)$, $E(0, 0, a)$, $F(a, 0, a)$, $G(a, a, a)$, $H(0, a, a)$;

- так как $x \leqslant y$, то точка α лежит по ту же сторону от плоскости $x = y$, что и точка H;
- так как $y \leqslant z$, то точка α лежит по ту же сторону от плоскости $y = z$, что и точка E.

Пересекая три указанных множества (куб и два полупространства), получаем, что точка α выбирается случайно из тетраэдра $AGEH$.

Условию $z - x \geqslant b$ соответствуют те точки тетраэдра $AGEH$, которые лежат выше плоскости $z = x + b$. Эта плоскость пересекает тетраэдр в точках $K(0, 0, b)$, $L(a - b, a - b, a)$, $I(a - b, a, a)$, $J(0, b, b)$.



Искомая вероятность p равна отношению объемов многогранника $HEKJIL$ и тетраэдра $AGEH$.

Объем тетраэдра $AGEH$ равен

$$V_{AGEH} = \frac{1}{3}AE \cdot S_{GEH} = \frac{1}{3}a \cdot \frac{1}{2}a^2 = \frac{a^3}{6}.$$

Объем многогранника $HEKJIL$ вычислим как сумму объемов тетраэдров $IHEKJ$ и $KLEI$.

Имеем

$$S_{HEKJ} = S_{AHE} - S_{AJK} = \frac{1}{2}AE \cdot HE - \frac{1}{2}AK \cdot JK = \frac{1}{2}\left(a^2 - b^2\right).$$

Тогда

$$V_{IHEKJ} = \frac{1}{3}IH \cdot S_{HEKJ} = \frac{1}{3}(a - b) \cdot \frac{1}{2}(a^2 - b^2) = \frac{1}{6}(a - b)^2(a + b).$$

Далее площадь основания тетраэдра $KLEI$

$$S_{LEI} = S_{EGH} - S_{LGI} - S_{EIH} = \frac{1}{2}a^2 - \frac{1}{2}b^2 - \frac{1}{2}(a - b)a = \frac{1}{2}(a - b)b.$$

Тогда

$$V_{KLEI} = \frac{1}{3}KE \cdot S_{LEI} = \frac{1}{3}(a - b) \cdot \frac{1}{2}(a - b)b = \frac{1}{6}(a - b)^2 b.$$

Значит, искомая вероятность

$$p = \frac{V_{IHEKJ} + V_{KLEI}}{V_{AGEH}} = \frac{\frac{1}{6}(a - b)^2(a + b) + \frac{1}{6}(a - b)^2 b}{\frac{1}{6}a^3} = \frac{(a - b)^2(2b + a)}{a^3}.$$

Для получения ответа в процентах умножим полученное выражение на 100.

Погрешность 0,01.

Варианты

$a = 8, 9, \ldots, 16$; $b = 1, 2, \ldots, 7$.

Ответ: $\dfrac{100(a-b)^2(2b+a)}{a^3}$.

Задача II.2.4.5. (25 баллов)

Темы: алгебра, неравенства, экстремальные значения.

Условие

Найдите наибольшее значение выражения $y + bx$ при условии

$$\log_{x^2+\frac{y^2}{a^2}} 2x \geqslant 1.$$

Формат ответа: приближенный с точностью до $0,01$.

Решение

Положим $m = y + bx$. Тогда $y = m - bx$ — уравнение прямой с угловым коэффициентом $-b$, которая отсекает отрезок m на оси Oy. Для любой точки (x_0, y_0) этой прямой получается одно и то же значение $m = y_0 + bx_0$. Таким образом, задача сводится к поиску такой точки, удовлетворяющей неравенству

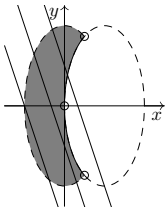
$$\log_{x^2+\frac{y^2}{a^2}} 2x \geqslant 1, \tag{II.2.1}$$

которая лежала бы на прямой с наибольшим параметром m. В этой точке и будет достигаться наибольшее значение выражения $y + bx$.

Рассмотрим два случая. Первый: $0 < x^2+\frac{y^2}{a^2} < 1$. Это неравенство задает внутренность эллипса с центром в начале координат и полуосями 1 (по оси x) и a (по оси y), центр эллипса и его граница исключаются. На этом множестве неравенство (II.2.1) равносильно

$$\log_{x^2+\frac{y^2}{a^2}} 2x \geqslant \log_{x^2+\frac{y^2}{a^2}} \left(x^2 + \frac{y^2}{a^2}\right) \iff 2x \leqslant x^2 + \frac{y^2}{a^2} \iff (x-1)^2 + \frac{y^2}{a^2} \geqslant 1.$$

Полученное неравенство задает границу и внешнюю часть эллипса с центром в точке $(1, 0)$ и полуосями 1 (вдоль оси x) и a (вдоль оси y). Пересечение этих областей показано на рисунке.

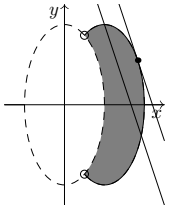


Рассматривая всевозможные прямые $y = m - bx$, проходящие через данную область (некоторые из этих прямых изображены на рисунке), видим, что наибольшего значения m не существует (можно сколь угодно близко приближать прямую к границе области).

Второй случай: $x^2 + \frac{y^2}{a^2} > 1$. Это неравенство задает внешность эллипса с центром в начале координат и полуосями 1 (по оси x) и a (по оси y), граница исключается. На этом множестве неравенство (II.2.1) равносильно

$$\log_{x^2+\frac{y^2}{a^2}} 2x \geqslant \log_{x^2+\frac{y^2}{a^2}} \left(x^2 + \frac{y^2}{a^2}\right) \iff 2x \geqslant x^2 + \frac{y^2}{a^2} \iff (x-1)^2 + \frac{y^2}{a^2} \leqslant 1.$$

Полученное неравенство задает границу и внутреннюю часть эллипса с центром в точке $(1, 0)$ и полуосями 1 (вдоль оси x) и a (вдоль оси y). Пересечение этих областей показано на рисунке.



В этом случае наибольшее значение m соответствует прямой $y = m - bx$, касающейся эллипса в верхней его части. Найдем точку касания.

$$\begin{cases} (x-1)^2 + \frac{y^2}{a^2} = 1, \\ y = m - bx. \end{cases}$$

Подставляя значение y из второго уравнения в первое, имеем

$$x^2 - 2x + \frac{(m-bx)^2}{a^2} = 0 \iff x^2\left(1 + \frac{b^2}{a^2}\right) - 2x\left(1 + \frac{mb}{a^2}\right) + \frac{m^2}{a^2} = 0.$$

Уравнение должно иметь единственное решение, значит, дискриминант, деленный на 4, равен нулю:

$$\frac{D}{4} = \left(1 + \frac{mb}{a^2}\right)^2 - \frac{m^2}{a^2}\left(1 + \frac{b^2}{a^2}\right) = 0.$$

Отсюда

$$m^2 - 2bm - a^2 = 0,$$

то есть $m = b \pm \sqrt{a^2 + b^2}$. Знак «$-$» перед корнем соответствует нижней точке касания, а знак «$+$» — верхней. Поэтому интересующее значение $m = b + \sqrt{a^2 + b^2}$.

Погрешность 0,01.

Варианты

$a = 1, 2, \ldots, 10$; $b = 1, 2, \ldots, 10$.

Ответ: $b + \sqrt{a^2 + b^2}$.

Третья волна. Задачи 8–9 класса

Задача II.2.5.1. (15 баллов)

Темы: алгебра, квадратный корень.

Условие

Решите уравнение

$$x^4 \cdot \sqrt{\frac{1}{x^2} - \frac{1}{x^3}} - x \cdot \sqrt{1 - \frac{1}{x}} = r\sqrt{-x}\sqrt{1-x}.$$

Запишите ответ с точностью до 0,01 (если корней несколько, то запишите в ответе наибольший из них).

Решение

Так как в уравнении присутствуют выражения $\sqrt{-x}$ и $\frac{1}{x}$, то решениями могут быть только отрицательные значения x. Учитывая, что $x < 0$, имеем

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

Далее

$$x^4\sqrt{\frac{1}{x^2} - \frac{1}{x^3}} = x^2 \cdot x^2\sqrt{\frac{1}{x^2} - \frac{1}{x^3}} = x^2\sqrt{x^2 - x}.$$

По свойству корня будет

$$\sqrt{-x}\sqrt{1-x} = \sqrt{-x(1-x)} = \sqrt{x^2 - x}.$$

Учитывая все сказанное, получаем равносильное исходному уравнение

$$x^2\sqrt{x^2 - x} + \sqrt{x^2 - x} = r\sqrt{x^2 - x}.$$

Так как $x < 0$, то $x^2 - x > 0$, поэтому деление уравнения на $\sqrt{x^2 - x}$ не приведет к потере корней. Имеем

$$x^2 = r - 1.$$

Снова учитывая, что $x < 0$, находим единственный корень

$$x = -\sqrt{r-1}.$$

Погрешность 0,01.

Варианты

$r = 3, 4, \ldots, 50$.

Ответ: $-\sqrt{r-1}$.

Задача II.2.5.2. (20 баллов)

Темы: комбинаторика.

Условие

В свой день рождения Алина решила приготовить фруктовый шашлык. Кусочки фруктов насаживаются на деревянную шпажку в следующих количествах: a кружков банана, b кубиков киви, c брусочков ананаса, и d долек мандарина. Сколько у Алины есть способов расположить фрукты на шпажке, если кусочки одного фрукта неотличимы, а шашлыки, получающиеся друг из друга переворотом шпажки, считаются одинаковыми?

Решение

Подсчитаем общее число шашлыков сначала без учета переворота шпажки. Если различать все кусочки фруктов (можно каждому назначить номер), то всего существует $(a+b+c+d)!$ способов расположить их. Однако перестановка фруктов одного вида не изменяет шашлык. Поэтому общее число способов

$$N = \frac{(a+b+c+d)!}{a!b!c!d!}.$$

Теперь подсчитаем количество шашлыков, которые не меняются при перевороте шпажки, то есть симметричных относительно середины. Поскольку число d нечетно, а числа a, b, c четны, то симметричные шашлыки существуют, в их середине располагается долька мандарина, а по разные стороны от середины располагаются все фрукты в равных количествах. Тогда общее число симметричных шашлыков

$$S = \frac{\left(\frac{a+b+c+(d-1)}{2}\right)!}{\frac{a}{2}!\frac{b}{2}!\frac{c}{2}!\frac{d-1}{2}!}.$$

Теперь будем считать одинаковыми шашлыки с точностью до переворота. Тогда симметричные шашлыки ранее были учтены один раз, а несимметричные — два раза. Поэтому количество несимметричных шашлыков с точностью до переворота равно $\frac{N-S}{2}$.

Добавляя к этому количеству число симметричных шашлыков, получаем, что всего у Алины способов

$$\frac{N-S}{2} + S = \frac{N+S}{2} = \frac{1}{2}\left(\frac{(a+b+c+d)!}{a!b!c!d!} + \frac{\left(\frac{a+b+c+(d-1)}{2}\right)!}{\frac{a}{2}!\frac{b}{2}!\frac{c}{2}!\frac{d-1}{2}!}\right).$$

Погрешность 0.

Варианты

$a = 2, 4, 6$; $b = 2, 4, 6$; $c = 2, 4, 6$; $d = 3, 5, 7$.

Ответ: $\dfrac{1}{2}\left(\dfrac{(a+b+c+d)!}{a!b!c!d!} + \dfrac{\left(\frac{a+b+c+(d-1)}{2}\right)!}{\frac{a}{2}!\frac{b}{2}!\frac{c}{2}!\frac{d-1}{2}!}\right)$.

Задача II.2.5.3. (20 баллов)

Темы: вероятность, схема Бернулли.

Условие

На Объединенной физико-математической олимпиаде участникам предлагается a задачи по математике и b задачи по физике. Михаил решает задачу по математике с вероятностью $P\%$, а задачу по физике — с вероятностью $Q\%$. С какой вероятностью Михаил решит на олимпиаде не менее двух задач?

Ответ дайте в процентах с точностью до 0,01.

Решение

Вычислим вероятность дополнительного события: Михаил решит на олимпиаде менее двух задач, то есть либо ни одной, либо ровно одну задачу. Далее используем обозначения $p = \frac{P}{100}$, $q = \frac{Q}{100}$.

Вероятность не решить ни одну задачу

$$P_0 = (1-p)^a(1-q)^b.$$

Вероятность решить ровно одну задачу

$$P_1 = ap(1-p)^{a-1}(1-q)^b + (1-p)^a bq(1-q)^{b-1}$$

(первое слагаемое — вероятность решить ровно одну задачу по математике, второе — ровно одну по физике).

Тогда искомая вероятность

$$P = 1 - (P_0 + P_1) = 1 - (1-p)^a(1-q)^b - ap(1-p)^{a-1}(1-q)^b - (1-p)^a bq(1-q)^{b-1}.$$

Для получения ответа в процентах умножим это выражение на 100.

Погрешность 0,01.

Варианты

$a = 2, 3$; $b = 2, 3$; $P = 10, 15, \ldots, 60$; $Q = 10, 15, \ldots, 60$.

Ответ: $100(1-(1-\frac{P}{100})^a(1-\frac{Q}{100})^b - a\frac{P}{100}(1-\frac{P}{100})^{a-1}(1-\frac{Q}{100})^b - (1-\frac{P}{100})^a b\frac{Q}{100}(1-\frac{Q}{100})^{b-1})$.

Задача II.2.5.4. (20 баллов)

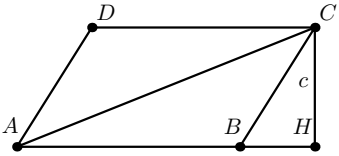
Темы: планиметрия, параллелограмм.

Условие

Сумма длин смежных сторон параллелограмма равна p, а его высоты равны c и d. Найдите расстояние от вершины тупого угла параллелограмма до его большей диагонали.

Формат ответа: приближенный с точностью до 0,01.

Решение



Назовем вершины параллелограмма буквами A, B, C, D так, чтобы углы B и D были тупыми, а $AB > BC$. Искомое расстояние от вершины D до диагонали AC равно высоте h треугольника ACD, которую будем искать, используя формулу для площади

$$S_{ACD} = \frac{1}{2} AC \cdot h.$$

Так как площадь S_{ACD} в два раза меньше площади S параллелограмма, то

$$h = \frac{S}{AC}. \tag{II.2.2}$$

Поскольку $S = c \cdot AB = d \cdot BC$, то $AB = \frac{d}{c} \cdot BC$. Но $AB + BC = p$, поэтому $\frac{d}{c} \cdot BC + BC = p$, откуда

$$BC = \frac{pc}{d+c}, \quad AB = \frac{pd}{d+c}.$$

Таким образом, площадь параллелограмма равна

$$S = c \cdot AB = \frac{pcd}{d+c}.$$

Теперь найдем длину диагонали AC. Проведем высоту CH параллелограмма. По теореме Пифагора для треугольника BHC имеем

$$BH = \sqrt{BC^2 - c^2}.$$

По теореме Пифагора для треугольника AHC имеем

$$AC = \sqrt{(AB + BH)^2 + c^2} = \sqrt{\left(\frac{pd}{d+c} + \sqrt{\left(\frac{pc}{d+c} \right)^2 - c^2} \right)^2 + c^2}.$$

По формуле (II.2.2) окончательно получаем

$$h = \frac{pcd}{d+c} \cdot \frac{1}{AC} = \frac{pcd}{\sqrt{(pd + c\sqrt{p^2 - (d+c)^2})^2 + c^2(d+c)^2}}.$$

Погрешность 0,01.

Варианты

$c = 2, 3, \ldots, 6$; $d = 7, 8, \ldots, 11$; $p = 18, 19, \ldots, 25$.

Ответ: $\dfrac{pcd}{\sqrt{\left(pd + c\sqrt{p^2 - (d+c)^2}\right)^2 + c^2(d+c)^2}}$.

Задача II.2.5.5. (25 баллов)

Темы: теория чисел, делимость, остатки.

Условие

Завод производит N холодильников в день. Каждый день нанимается одна из компаний-перевозчиков для развоза техники по торговым точкам. Первая компания перевозит всю технику, загрузив в каждый автомобиль по 5 холодильников. Автомобили второй загружаются по 7 холодильников, кроме последнего, перевозящего a штук. Третья загружает в автомобили по 8 холодильников, но для последней машины остается только b штук. Определите наименьшее возможное значение N, если известно, что $N \geqslant 280$.

Решение

По условию задачи составим систему уравнений

$$\begin{cases} N = 5k, \quad k \in \mathbb{Z}, \\ N = 7l + a, \quad l \in \mathbb{Z}, \\ N = 8m + b, \quad m \in \mathbb{Z}. \end{cases}$$

Из первого и второго уравнения системы следует

$$5k = 7l + a.$$

Чтобы определить значения k, удовлетворяющие этому уравнению, рассмотрим семь случаев, соответствующих возможным остаткам от деления на 7 числа k.

1. Если $k = 7x$, $x \in \mathbb{Z}$, то $5 \cdot 7x = 7l + a$. Поскольку a не делится на 7, то этот случай не дает решений.
2. Если $k = 7x+1$, $x \in \mathbb{Z}$, то $5 \cdot 7x+5 = 7l+a$. Если $a = 5$, то подходящие значения k имеют вид $7x + 1$, иначе этот случай не дает решений.
3. Если $k = 7x + 2$, $x \in \mathbb{Z}$, то $5 \cdot 7x + 10 = 7l + a$. Если $a - 10$ делится на 7, то есть $a = 3$, то подходящие k имеют вид $7x + 2$, иначе этот случай не дает решений.

Аналогично рассматриваются оставшиеся четыре случая. Подходящие значения k имеют вид $k = 7x + r$, где число $r \in [1, 6]$ определится однозначно.

Воспользуемся первым и третьим уравнением системы. Имеем

$$5k = 8m + b.$$

Учитывая найденный вид числа k, получаем

$$5(7x+r) = 8m + b \iff 35x = 8m + b - 5r.$$

Значения x подберем, перебирая возможные остатки от деления x на 8.

1. Если $x = 8y$, $y \in \mathbb{Z}$, то $35 \cdot 8y = 8m + b - 5r$. Если $b - 5r$ делится на 8, то подходящие значения x имеют вид $8y$, иначе этот случай не дает решений.
2. Если $x = 8y + 1$, $y \in \mathbb{Z}$, то $35 \cdot 8y + 35 = 8m + b - 5r$. Если $b - 5r - 35$ делится на 8, то подходящие значения x имеют вид $8y + 1$, иначе этот случай не дает решений.

Аналогично рассматриваются оставшиеся шесть случаев. Подходящие значения x имеют вид $x = 8y + q$, где число $q \in [0, 7]$ определится однозначно.

Таким образом,

$$N = 5k = 5(7x + r) = 5(7(8y + q) + r) = 280y + 35q + 5r.$$

Так как по условию $N \geqslant 280$, то наименьшее значение $N = 280 + 35q + 5r$ получается при $y = 1$.

Погрешность 0.

Варианты

$a = 1, 2, \ldots, 6$; $b = 1, 2, \ldots, 7$.

Ответ: $(105b + 120a)\%280 + 280$, где $\alpha\%\beta$ —остаток от деления α на β.

Примечание: формула для ответа получена из общей теории систем линейных сравнений. Предполагается, что участники будут решать задачу методом перебора, как было описано выше, а не выводить данную формулу.

Третья волна. Задачи 10–11 класса

Задача II.2.6.1. (15 баллов)

Темы: теория чисел, комбинаторика.

Условие

Число n в b-ичной системе счисления записывается как 1000. Выписали все натуральные числа от 1 до n в той же системе счисления. Сколько среди выписанных чисел таких, в записи которых используется ровно две различные цифры?

Решение

Всего двузначных чисел, в записи которых ровно две различные цифры, равно $(b-1)^2$ (на первом месте может быть любая цифра, кроме 0, а на втором — любая, кроме той, что на первом месте).

Множество нужных трехзначных чисел разобьем на 2 группы: в первой группе первые две цифры одинаковые, а во второй — разные. В первой группе чисел столько же, сколько двузначных чисел с двумя различными цифрами, то есть $(b-1)^2$. Для подсчета чисел во второй группе учтем, что первые две цифры можно выбрать $(b-1)^2$ способами, а третью цифру — двумя способами. Значит, всего во второй группе $2(b-1)^2$ чисел. А всего интересующих трехзначных будет $(b-1)^2+2(b-1)^2 = 3(b-1)^2$.

Также единственное выписанное в условии четырехзначное число использует в своей записи две различных цифры.

Итого имеется

$$(b-1)^2 + 3(b-1)^2 + 1 = 4(b-1)^2 + 1$$

интересующих нас чисел.

Погрешность 0.

Варианты

$b = 20, 21, \ldots, 50$.

Ответ: $4(b-1)^2 + 1$.

Задача II.2.6.2. (20 баллов)

Темы: текстовая задача, логика.

Условие

Домашние часы со стрелками и цифровые часы синхронизованно показывают верное время. Ровно в полночь батарейка в часах со стрелками разрядилась до критического значения: раз в минуту скорость их хода стала меняться в $1-\frac{1}{k}$ раз (первый раз стрелки замедлились, когда цифровые часы показали 00:00, затем 00:01 и т. д.; в течение каждой минуты скорость стрелок постоянна). Сколько минут будут показывать цифровые часы в момент, когда стрелочные часы вновь покажут верное время?

Решение

Пусть t — время в минутах, прошедшее с того момента, как стрелочные часы замедлились в первый раз, до момента, когда они вновь показали верное время. Пусть $n = \lfloor t \rfloor$ (здесь $\lfloor \cdot \rfloor$ — округление вниз до ближайшего целого).

Положим $q = 1 - \frac{1}{k}$. За n минут конец минутной стрелки преодолеет количество минутных долей циферблата, равное

$$M = q + q^2 + q^3 + \ldots + q^n = \frac{q(1-q^n)}{1-q}.$$

То есть в момент n минутная стрелка находится между делениями, отвечающими $\lfloor M \rfloor$ и $\lfloor M \rfloor + 1$ минутам.

Так как $q = 1 - \frac{1}{k} = \frac{k-1}{k}$, имеем

$$M = \frac{\frac{k-1}{k}\left(1 - \left(\frac{k-1}{k}\right)^n\right)}{1/k} = k - 1 - (k-1)\left(\frac{k-1}{k}\right)^n.$$

Поскольку стрелочные часы покажут верное время не ранее, чем через 12 часов, то $n \geqslant 12 \cdot 60 = 720$. А так как $k < 100$, то

$$(k-1)\left(\frac{k-1}{k}\right)^n < 100\left(\frac{99}{100}\right)^n \leqslant 100\left(\frac{99}{100}\right)^{720} < 0,1.$$

Поэтому

$$\lfloor M \rfloor = \left\lfloor k - 1 - (k-1)\left(\frac{k-1}{k}\right)^n \right\rfloor = k - 2.$$

Аналогично, к моменту $n+1$ минутная стрелка пройдет

$$\frac{q(1-q^{n+1})}{1-q} = k - 1 - (k-1)\left(\frac{k-1}{k}\right)^{n+1}$$

долей циферблата. Это число меньше $k-1$. А так как $t \in [n, n+1)$, то в момент времени t минутная стрелка будет между делениями, отвечающими $k-2$ и $k-1$ минутам. Следовательно, в момент t цифровые часы будут показывать количество минут, равное остатку от деления $k-2$ на 60.

Погрешность 0.

Варианты

$k = 65, 66, \ldots, 99$.

Ответ: $(k-2)\%60$, где $x\%y$ — остаток от деления x на y.

Задача II.2.6.3. (20 баллов)

Темы: стереометрия, геометрическая вероятность.

Условие

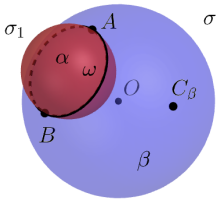
Точки A и B лежат на сфере с центром O так, что угол AOB равен α°. Случайно на сфере выбирается еще одна точка C (любой выбор равновозможен). Определите вероятность того, что угол ACB окажется острым.

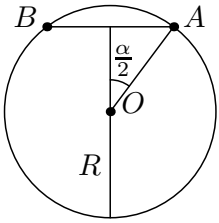
Запишите ответ в процентах с точностью до 0,01.

Решение

Назовем данную сферу σ. Построим сферу σ_1, для которой точки A и B диаметрально противоположны. При пересечении сфер σ и σ_1 получается окружность ω. Для произвольной точки C_ω, взятой на окружности ω, угол $AC_\omega B$ — прямой, поскольку он опирается на диаметр.

Окружность ω разбивает все множество точек на сфере на два множества, для которых ω — граница. Назовем α меньшую из них по площади, а β — большую. Границу ω в эти множества не включаем.





Рассмотрим произвольную точку C_α из множества α. Проведем плоскость через точки A, B и C_α. Эта плоскость пересечет сферу σ_1 по окружности, внутри которой лежит точка C_α. Это значит, что угол $AC_\alpha B$ — тупой.

Аналогично рассуждая, обнаружим, что для произвольной точки C_β из множества β угол $AC_\beta B$ — острый.

Таким образом, вероятность того, что угол ACB — острый, равна отношению площадей множеств σ и β. Пусть R — радиус сферы σ. Имеем

$$S_\sigma = 4\pi R^2.$$

Величину S_β найдем по формуле для площади сферической поверхности шарового сегмента:

$$S_\beta = 2\pi RH,$$

где H — высота шарового сегмента. Имеем

$$H = R + R\cos\frac{\alpha}{2} = R\left(1 + \cos\frac{\alpha}{2}\right).$$

Таким образом, искомая вероятность равна

$$\frac{S_\beta}{S_\sigma} = \frac{2\pi R^2\left(1+\cos\frac{\alpha}{2}\right)}{4\pi R^2} = \frac{1+\cos\frac{\alpha}{2}}{2}.$$

Для получения значения в процентах, домножим полученный результат на 100.

Погрешность 0,01.

Варианты

$\alpha = 5, 10, \ldots, 175$.

Ответ: $50\left(1 + \cos\frac{\alpha}{2}\right)$.

Задача II.2.6.4. (20 баллов)

Темы: теория чисел.

Условие

Саша придумал алгоритм шифрования пары целых чисел: первое заменяется на остаток от деления на m их суммы, а второе заменяется на остаток от деления на m их произведения. Саша выбрал два числа из промежутка $[2, m-1]$ и зашифровал их. Далее он изменил исходную пару, уменьшив на единицу второе число. Оказалось, что шифр новой пары отличается от шифра прежней перестановкой чисел. Определите числа, которые изначально выбрал Саша.

Запишите в ответ эти числа подряд без разделяющих символов. Например, если первое число 872, а второе число 43, то ответ должен быть 87243.

Решение

Пусть x, y — исходная пара чисел. Из условия задачи получаем систему

$$\begin{cases} x + y \equiv \alpha \pmod m, \\ xy \equiv \beta \pmod m, \\ x + y - 1 \equiv \beta \pmod m, \\ x(y-1) \equiv \alpha \pmod m. \end{cases}$$

Вычитая из первого уравнения третье, получаем

$$1 \equiv \alpha - \beta \pmod m.$$

Вычитая из второго уравнения четвертое и используя полученное выше соотношение, находим

$$x \equiv \beta - \alpha \equiv -1 \pmod m.$$

Значит, $x = -1 + km$, где $k \in \mathbb{Z}$. Так как $2 \leqslant x \leqslant m-1$ (по условию), то $x = m - 1$.

Подставим полученное значение x в первое и второе уравнения:

$$\begin{cases} m - 1 + y \equiv \alpha \pmod m, \\ (m-1)y \equiv \beta \pmod m. \end{cases} \iff \begin{cases} y - 1 \equiv \alpha \pmod m, \\ -y \equiv \beta \pmod m. \end{cases}$$

Вычитая эти два уравнения, получаем

$$2y - 1 \equiv \alpha - \beta \equiv 1 \pmod m.$$

То есть $2y \equiv 2 \pmod m$. Отсюда $y = 1 + \frac{qm}{2}$, где $q \in \mathbb{Z}$. Так как $2 \leqslant y \leqslant m-1$, то $y = 1 + \frac{m}{2}$.

Таким образом, $x = m - 1$, $y = 1 + \frac{m}{2}$.

Погрешность 0.

Варианты

$m = 10^6, 10^6 + 10^5, \ldots, 10^7$.

Ответ: $(m-1) \cdot 10^{\lfloor \log_{10}(1+m/2)+1 \rfloor} + 1 + \frac{m}{2}$.

Задача II.2.6.5. (20 баллов)

Темы: графы, комбинаторика.

Условие

В распоряжении парфюмера имеется n основных ароматов, среди которых k пар несовместимых. Какое наименьшее количество различных духов, составленных из трех ароматов, сможет создать парфюмер?

Решение

Оценка. Рассмотрим граф, в котором вершинами являются ароматы, а ребра соединяют совместимые ароматы. Представим сначала, что граф полный, а затем будем последовательно удалять ребра, соответствующие несовместимым ароматам.

Рассмотрим две произвольные вершины. Их можно соединить с третьей вершиной $n-2$ способами. Поэтому удаление одного ребра приводит к запрету не более чем $n-2$ видов духов. Поскольку всего k пар несовместимых духов, то, последовательно удаляя соответствующие ребра, мы запретим не более, чем $k(n-2)$ видов духов.

Всего троек ароматов C_n^3. Поэтому возможных видов духов не менее

$$C_n^3 - k(n-2) = \frac{n(n-1)(n-2)}{6} - k(n-2) = (n-2)\left(\frac{n(n-1)}{6} - k\right).$$

Пример. Допустим, что каждый аромат, имеющийся в списке пар несовместимых, встречается только в одной такой паре (так как $k \leqslant n/2$, то такая ситуация возможна). Рассмотрим вершины A и B графа, соответствующие несовместимым ароматам. Вершина A соединена со всеми вершинами, кроме B, а вершина B соединена со всеми вершинами, кроме A. Тогда удаление ребра AB приводит к запрету ровно $n-2$ видов духов. Значит, последовательно удаляя из полного графа ребра, соответствующие несовместимым ароматам, мы запретим ровно $k(n-2)$ видов духов, а возможных духов будет ровно $(n-2)\left(\frac{n(n-1)}{6} - k\right)$.

Погрешность 0.

Варианты

$n = 16, 17, \ldots, 30$; $k = 4, 5, \ldots, 8$.

Ответ: $(n-2)\left(\dfrac{n(n-1)}{6} - k\right)$.

Инженерный тур

В первую очередь в пакет задач первого отборочного этапа включены задачи для проверки базовых знаний по геометрии, по алгоритмам и структурам данных.

Задача II.3.1. Объем пентаэдра (100 баллов)

Темы: геометрия, программирование.

Условие

Даны три точки в пространстве $(x_i,\ y_i,\ z_i)$. Необходимо найти объем выпуклого пятигранника (пентаэдра), образуемого следующими вершинами: $(x_1,\ y_1,\ 0)$, $(x_2,\ y_2,\ 0)$, $(x_3,\ y_3,\ 0)$, $(x_1,\ y_1,\ z_1)$, $(x_2,\ y_2,\ z_2)$, $(x_3,\ y_3,\ z_3)$.

Формат входных данных

В i-ой строке содержатся вещественные числа — координаты заданных точек в пространстве.

Формат выходных данных

Требуется вывести одно число — объем пентаэдра, округленный до третьего знака после запятой.

Примеры

Пример №1

Стандартный ввод
0 0 1 1 0 1 0 1 1
Стандартный вывод
0.500

Тесты

https://disk.yandex.ru/d/HhX3TlsCkgJoSQ.

Пример программы-решения

Ниже представлено решение на языке C++.

```
#include <iomanip>
#include <iostream>

using namespace std;

struct Vector3 {
   double x, y, z;
};

int main() {
   Vector3 points[3];

   for (int i = 0; i < 3; i++) {
      cin >> points[i].x >> points[i].y >> points[i].z;
   }

   for (int i = 0; i < 3; i++) {
      for (int ii = 0; ii < 2 - i; ii++) {
         if (points[ii].z > points[ii + 1].z) {
            Vector3 temp = points[ii];
            points[ii] = points[ii + 1];
            points[ii + 1] = temp;
         }
      }
   }

   double sides[3];
   double p = 0;

   for (int i = 0; i < 3; i++) {
      sides[i] = sqrt(pow((points[i].x - points[(i + 1) % 3].x), 2) +
      ↪  pow((points[i].y - points[(i + 1) % 3].y), 2));
      p += sides[i];
   }
   p /= 2;
   double square = sqrt(p * (p - sides[0]) * (p - sides[1]) * (p - sides[2]));
```

```
    double res = 0;
    res += square * points[0].z;
    res += 1.0 / 3 * square * (points[1].z - points[0].z);
    Vector3 newPoint = points[2];
    newPoint.z = points[0].z;

    p = 0;
    for (int i = 0; i < 3; i++) {
       sides[i] = sqrt(pow((points[i].x - points[(i + 1) % 3].x), 2) +
       ↪  pow((points[i].y - points[(i + 1) % 3].y), 2) + pow((points[i].z -
       ↪  points[(i + 1) % 3].z), 2));
       p += sides[i];
    }
    p /= 2;
    square = sqrt(p * (p - sides[0]) * (p - sides[1]) * (p - sides[2]));

    double A = 0, B = 0, C = 0, D = 0;
    A = (points[1].y - points[0].y) * (points[2].z - points[0].z) - (points[2].y -
    ↪  points[0].y) * (points[1].z - points[0].z);
    B = -((points[1].x - points[0].x) * (points[2].z - points[0].z) - (points[2].x
    ↪  - points[0].x) * (points[1].z - points[0].z));
    C = (points[1].x - points[0].x) * (points[2].y - points[0].y) - (points[2].x -
    ↪  points[0].x) * (points[1].y - points[0].y);
    D = -points[0].x * A - points[0].y * B - points[0].z * C;

    double h = abs(A * newPoint.x + B * newPoint.y + C * newPoint.z + D) / sqrt(A *
    ↪  A + B * B + C * C);

    res += 1.0 / 3 * h * square;
    cout << fixed << setprecision(3) << res;
}
```

Задача II.3.2. Маляры (100 баллов)

Темы: алгоритмы, программирование.

Условие

В бригаде n маляров. Каждый маляр имеет свое время, за которое он красит один квадратный метр стены, кто-то быстрее, а кто-то медленнее. Всего стен также n, на каждую из них назначается по одному маляру. Расставьте маляров так, чтобы работа была закончена за минимальное количество минут.

Все маляры стартуют красить стены одновременно. Если вдруг какой-то маляр закончил красить свою стену, он отправляется пить чай и больше не работает.

Формат входных данных

В первой строке на вход программе подается целое число n ($1 \leqslant n \leqslant 10^5$) — количество маляров и стен. В следующей строке записано n целых чисел $1 \leqslant v_i \leqslant \leqslant 2 \cdot 10^4$ — скорость окраски маляром квадратных метров в минуту. Далее в n строках записано по два целых числа h_i и w_i ($1 \leqslant h_i, w_i \leqslant 10^4$) — высота и ширина стены соответственно.

Формат выходных данных

Выведите одно целое число — количество минут, за которое маляры выполнят работу. Если время выполнения работы не получается целым числом, округлите его в большую сторону.

Примеры

Пример №1

Стандартный ввод
4 5 12 20 30 5 10 2 3 10 9 7 8

Стандартный вывод
5

Тесты

https://disk.yandex.ru/d/EaaG5xwnd3FMTA.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
def solve() -> str:
    n = int(input())
    workers = list(map(int, input().split()))
    stones = []
    for i in range(n):
        a, b = map(int, input().split())
        stones.append(a * b)
        workers.sort()
        stones.sort()
        res = 0
        for i in range(n):
            worktime = stones[i] // workers[i]
            if stones[i] % workers[i]:
                worktime += 1
            res = max(worktime, res)
        return str(res)
    print(solve())
```

Задача II.3.3. Доставка грузов (100 баллов)

Темы: программирование, структуры данных.

Условие

Мальчик Ваня работает в службе доставки грузов. На сегодня у него есть n домов, в которые надо доставить некоторые вещи. Машина у Вани маленькая, поэтому взять он может максимум один груз за раз, то есть после доставки ему надо вернуться на склад. В начальный момент времени Ваня находится на складе.

Мэр города перфекционист, поэтому все дороги в городе односторонние и имеют одинаковую длину, равную d км.

Помогите Ване узнать, какое минимальное количество км ему надо проехать, чтобы доставить все грузы. Учтите, что при последнем грузе возвращаться на склад не обязательно.

Формат входных данных

В первой строке входных данных находится два целых числа n ($1 \leqslant n \leqslant 2000$) и d ($1 \leqslant d \leqslant 200$) — количество домов, в которые надо доставить груз, и длина каждой дороги соответственно. В следующих $n + 1$ строках располагается число a_i ($0 \leqslant a_i \leqslant n$) — количество дорог, выходящих из дома с номером i. Далее в этой же строке следует a_i чисел b_j ($1 \leqslant b_j \leqslant n$) — номера домов, в которые можно прийти из дома с номером i. Если среди этих чисел есть $n + 1$, это означает, что из этого дома можно попасть на склад. В $n + 1$ строке описано, в какие дома можно попасть из склада.

Формат выходных данных

Выведете одно целое число — минимальное количество километров, которые необходимо проехать Ване, чтобы доставить все грузы. Если это невозможно, выведите -1.

Примеры

Пример №1

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

Стандартный вывод
9

Тесты

https://disk.yandex.ru/d/7G4eROGpCTjZ5A.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
from queue import Queue
n, d = map(int, input().split())
graph = [[] for i in range(n + 1)]
reversed_graph = [[] for i in range(n + 1)]

for i in range(n + 1):
    data = list(map(int, input().split()))
    for j in range(1, data[0] + 1):
        graph[i].append(data[j] - 1)
        reversed_graph[data[j] - 1].append(i)

queue_to_base = Queue()
distance_to_base = [10 ** 20] * (n + 1)
distance_to_base[n] = 0
queue_to_base.put(n)

while not queue_to_base.empty():
    vertex = queue_to_base.get()
    for to in reversed_graph[vertex]:
        if distance_to_base[to] == 10 ** 20:
            distance_to_base[to] = distance_to_base[vertex] + d
            queue_to_base.put(to)

queue_from_base = Queue()
distance_from_base = [10 ** 20] * (n + 1)
distance_from_base[n] = 0
queue_from_base.put(n)

while not queue_from_base.empty():
    vertex = queue_from_base.get()
    for to in graph[vertex]:
        if distance_from_base[to] == 10 ** 20:
            distance_from_base[to] = distance_from_base[vertex] + d
            queue_from_base.put(to)

if max(distance_from_base) == 10 ** 20:
    print(-1)
elif distance_to_base.count(10 ** 20) >= 2:
    print(-1)
else:
    res = 0
    max_return_distance = max(distance_to_base)
    for i in range(n):
        res += distance_from_base[i]
        if max_return_distance != distance_to_base[i]:
            res += distance_to_base[i]
print(res)
```

Задача II.3.4. Модный шоппинг (100 баллов)

Темы: программирование, два указателя.

Условие

Даша пошла в магазин за покупками, где увидела акцию «3 = 2».

Условия акции таковы: «Купите три вещи, и самая дешевая из них будет бесплатной».

У Даши не так много денег, всего k рублей, поэтому ей эта акция очень понравилась, и она решает выбрать ровно три вещи. Но при этом она одевается модно, поэтому минимум две из этих трех вещей должны быть одного цвета.

Помогите Даше выбрать три вещи так, чтобы они в сумме имели максимальную стоимость, но после применения акции у Даши хватило денег на покупку.

Формат входных данных

В первой строке располагаются два целых числа $3 \leqslant n \leqslant 10^3$ и $0 \leqslant k \leqslant 10^9$ — количество вещей в магазине и сумма Дашиных денег.

В следующих n строчках описываются вещи двумя целыми числами $1 \leqslant c_i \leqslant 10^9$ и $1 \leqslant a_i \leqslant 10^9$ — стоимостью и номером цвета соответственно.

Формат выходных данных

Выведите одно число — полную стоимость 3 вещей, которые лучше купить Даше. Если такое невозможно, выведите: -1.

Примеры

Пример №1

Стандартный ввод
5 10 1 1 2 1 4 3 5 3 6 3
Стандартный вывод
12

Тесты

https://disk.yandex.ru/d/-QXaZnJ98lgzrg.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```python
def solve():
    n, k = map(int, input().split())
    elements = [] for i in range(n):
        c_i, a_i = map(int, input().split())
        elements.append((c_i, a_i))
    elements.sort()

    res = -1
    for i in range(n):
        for j in range(i + 1, n):
            if elements[i][1] != elements[j][1]:
                continue
            left, r = 0, n
            now = elements[i][0] + elements[j][0]
            while r - left > 1:
                m = (r + left) // 2
                if m == i or m == j:
                    m += 1
                    if m == r or m == i or m == j:
                        m -= 2
                    if m == left or m == i or m == j:
                        r -= 1
                        continue

                if elements[m][0] > k - now + min(elements[i][0], elements[j][0],
                ↪  elements[m][0]):
                    r = m
                else:
                    left = m

                if elements[left][0] > k - now + min(elements[i][0],
                ↪  elements[j][0], elements[left][0]):
                    continue
                now += elements[left][0]

                if now > res:
                    res = now
    return str(res)

print(solve())
```

Работа наставника НТО на втором отборочном этапе

На втором отборочном этапе участникам предлагаются индивидуальные и командные задачи в рамках выбранных профилей. Для подготовки к нему наставник может использовать следующие рекомендуемые форматы и мероприятия:

- Подготовка по образовательным программам НТО по ряду технологических направлений.
- Разбор задач второго отборочного этапа НТО прошлых лет.
- Прохождение онлайн-курсов по разбору задач НТО прошлых лет.
- Прохождение онлайн-курсов, рекомендованных разработчиками профилей.
- Разбор материалов для подготовки к профилям.
- Практикумы. Для организации практикумов возможно использовать разные подходы или их комбинации:
  - Проведение практикумов по описаниям на страницах профилей и материалов для подготовки.
  - Декомпозиция задач заключительных этапов прошлых лет для выделения наиболее актуальных элементов и их изучения.
  - Анализ технических знаний и навыков (hard skills), требуемых для конкретного профиля, и самостоятельная разработка или поиск занятия для развития наиболее актуальных из них.
  - Посещение практикумов на площадках подготовки и онлайн-мероприятий от разработчиков профилей. Объявления о таких мероприятиях публикуются в группах НТО в VK и в телеграм-канале для наставников НТО (https://t.me/kruzhok_association).

Второй отборочный этап

Во второй отборочный этап традиционно включаются алгебраические и геометрические задания по программированию, а также характерные для профиля Технологии виртуальной реальности задачи с использованием игровых движков. Благодаря этому производится отбор лучших участников, кроме того, решение задач позволяет им подготовиться к предстоящему финальному этапу.

Тема, которая раскрывается на финальных соревнованиях, посвящена космосу и космическим явлениям, поэтому участникам предлагается разработка космического симулятора.

Задача IV.1. Навигационные спутники Васи (20 баллов)

Темы: программирование, алгебра и геометрия.

Условие

Юный программист Вася, вдохновившись принципом работы глобальных навигационных спутниковых систем, решил спроектировать собственную модель движения спутников вокруг Земли, которые в будущем могли бы определять его местоположение.

Для ведения своих расчетов Вася использует сферическую систему координат (r, φ, ψ), где r — расстояние от точки до начала координат, $\varphi \in [-90, 90]$ — зенитный угол, $\psi \in [0, 360]$ — азимутный угол. Направления изменения углов показаны на рис. 1 и рис. 2.

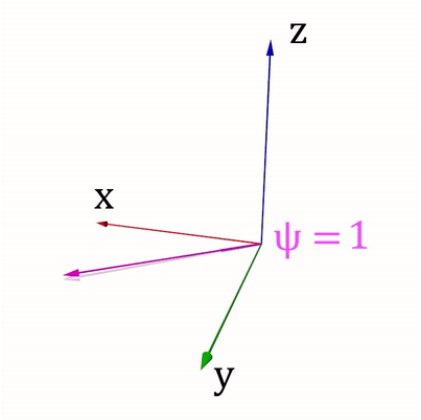


Рис. 1

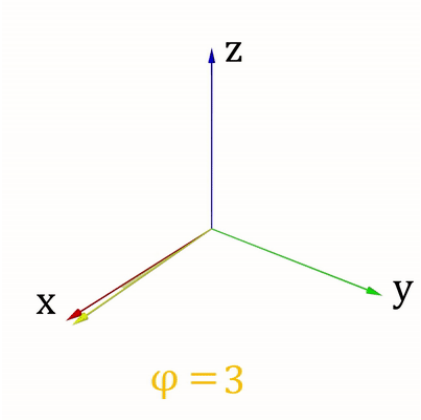


Рис. 2

В качестве модели Земли Вася взял идеальную сферу с уравнением $r = 100$ в

сферических координатах. Первая координата зафиксирована, поэтому каждая точка на поверхности будет описываться парой (φ, ψ).

Спутники Васи располагаются на k орбитах. Каждая i-я орбита представляет собой пространственную окружность радиуса R_i с центром в начале координат, лежащую в плоскости, заданной нормальным вектором $n = (r_n, \varphi_n, \psi_n)$. На каждой орбите находится m_i точечных спутников, равноудаленных друг от друга. В начальный момент времени $t_o = 0$ первый спутник расположен в координатах $(R_i, \varphi_{i1}, \psi_{i1})$; все спутники вращаются против часовой стрелки, если посмотреть на плоскость орбиты со стороны нормали $\underline{n}$, и пронумерованы последовательно от первого спутника, начиная с 1. Каждый спутник на орбите совершает полный оборот за время t_i.

Вася хочет написать программу, которая смогла бы оценить удачность выбранной системы спутников и расположения орбит. Для этого он выбирает целевую точку на поверхности сферы с координатами (φ_p, ψ_p) на сфере и пытается выяснить, сколько спутников в момент времени t будут находится на связи с целевой точкой. Будем говорить, что некоторый спутник находится на связи с целевой точкой, если отрезок, соединяющий спутник и целевую точку не пересекает сферу. Помогите Васе написать такую программу.

Формат входных данных

- В первой строке находится два числа: φ_p и ψ_p — координаты целевой точки на поверхности сферы.
- Во второй строке находится число t — время для изучения положения спутников.
- В третьей строке находится число k — количество орбит.

Орбиты пронумерованы, начиная с первой, в порядке их появления. Для каждой i-ой орбиты даны следующие данные:

- число m_i — количество спутников на i-ой орбите;
- числа r_n, φ_n, ψ_n — координаты вектора нормали плоскости вращения орбиты;
- числа R_i, φ_{i1}, ψ_{i1} — координаты первого спутника на орбите;
- число t_i — время полного оборота каждого спутника орбиты.

Формат выходных данных

Выходной файл содержит число k_v — число спутников, находящихся на связи с целевой точкой в момент времени t. Далее следуют k_v строк, содержащих два числа — номера орбит и порядковых номеров спутников на этих орбитах, находящихся на связи с целевой точкой в порядке возрастания обоих значений. Если на связи нет ни одного спутника — выведите 0.

Примеры

Пример №1

Стандартный ввод
90 0 0 1 2 1 0 0 150 90 0 1
Стандартный вывод
1 1 1

Тесты

Ссылка на архив с тестовыми наборами данных: https://disk.yandex.ru/d/kdcqXQK2pvZrrA/A_tests.zip.

Пример программы-решения

Ниже представлено решение на языке C++.

```
#include <fstream>
#define _USE_MATH_DEFINES
#include <math.h>
#include <vector>

using namespace std;

double planetRadius = 100;
int orbitNumber;
double timeSinceLaunch;
double eps = 1e-6;

class Vector3 {
public:
   double x;
   double y;
   double z;
   Vector3() {
      x = y = z = 0;
   }
   Vector3(double x1, double y1, double z1) {
      x = x1;
      y = y1;
      z = z1;
   }
   Vector3 operator+ (Vector3& v) {
      Vector3 res(x + v.x, y + v.y, z + v.z);
      return res;
```

```
29      }
30      Vector3 operator- (Vector3& v) {
31         Vector3 res(x - v.x, y - v.y, z - v.z);
32         return res;
33      }
34      Vector3 operator=(Vector3 v) {
35         x = v.x;
36         y = v.y;
37         z = v.z;
38         return *this;
39      }
40      Vector3 normalize() {
41         double xt = x; double yt = y; double zt = z;
42         x /= sqrt(xt * xt + yt * yt + zt * zt);
43         y /= sqrt(xt * xt + yt * yt + zt * zt);
44         z /= sqrt(xt * xt + yt * yt + zt * zt);
45         return *this;
46      }
47      double length() {
48         return sqrt(x * x + y * y + z * z);
49      }
50  };
51
52  double ToRad(double degrees) {
53      return degrees * M_PI / 180;
54  }
55
56  Vector3 RotateAroundAxis(Vector3 axis, float angle, Vector3 targetVector)
57  {
58      axis.normalize();
59      if ((axis.x == 0 && axis.y == 0 && axis.z == 0) || angle == 0)
60      {
61         return targetVector;
62      }
63
64      Vector3 res;
65      res.x = (cos(angle) + (1 - cos(angle)) * axis.x * axis.x) * targetVector.x
66         + ((1 - cos(angle)) * axis.x * axis.y - sin(angle) * axis.z) *
           ↪  targetVector.y
67         + ((1 - cos(angle)) * axis.x * axis.z + sin(angle) * axis.y) *
           ↪  targetVector.z;
68      res.y = ((1 - cos(angle)) * axis.y * axis.x + sin(angle) * axis.z) *
        ↪  targetVector.x
69         + (cos(angle) + (1 - cos(angle)) * axis.y * axis.y) * targetVector.y
70         + ((1 - cos(angle)) * axis.y * axis.z - sin(angle) * axis.x) *
           ↪  targetVector.z;
71      res.z = ((1 - cos(angle)) * axis.z * axis.x - sin(angle) * axis.y) *
        ↪  targetVector.x
72         + ((1 - cos(angle)) * axis.y * axis.z + sin(angle) * axis.x) *
           ↪  targetVector.y
73         + (cos(angle) + (1 - cos(angle)) * axis.z * axis.z) * targetVector.z;
74      return res;
75  }
76
77  struct Orbit
78  {
79      int numberOfSatellites;
80      Vector3 orbitNormalVector;
81      Vector3 firstSatellite;
82      double timeForFullRotation;
```

```
};

int main() {
   ifstream fin("input.txt");
   ofstream fout("output.txt");

   double sphereCoords[2];

   fin >> sphereCoords[0] >> sphereCoords[1];
   Vector3 targetPosition;
   targetPosition.x = planetRadius * cos(ToRad(sphereCoords[0])) *
    ↪  cos(ToRad(sphereCoords[1]));
   targetPosition.y = planetRadius * cos(ToRad(sphereCoords[0])) *
    ↪  sin(ToRad(sphereCoords[1]));
   targetPosition.z = planetRadius * sin(ToRad(sphereCoords[0]));

   if (abs(targetPosition.x) < eps) {
      targetPosition.x = 0;
   }
   if (abs(targetPosition.y) < eps) {
      targetPosition.y = 0;
   }
   if (abs(targetPosition.z) < eps) {
      targetPosition.z = 0;
   }

   fin >> timeSinceLaunch >> orbitNumber;

   vector<Orbit>orbits(orbitNumber);

   for (int i = 0; i < orbitNumber; i++) {
      Orbit o;
      Vector3 normal;
      fin >> o.numberOfSatellites >> normal.x >> normal.y >> normal.z >>
         o.firstSatellite.x >> o.firstSatellite.y >> o.firstSatellite.z >>
          ↪  o.timeForFullRotation;
      o.orbitNormalVector.x = normal.x * cos(ToRad(normal.y)) *
       ↪  cos(ToRad(normal.z));
      o.orbitNormalVector.y = normal.x * cos(ToRad(normal.y)) *
       ↪  sin(ToRad(normal.z));
      o.orbitNormalVector.z = normal.x * sin(ToRad(normal.y));

      orbits[i] = o;
   }


   int result = 0;

   vector<int>resOrbit;
   vector<int>resSatellite;

   for (int i = 0; i < orbitNumber; i++) {
      for (int ii = 0; ii < orbits[i].numberOfSatellites; ii++) {

         float x = timeSinceLaunch * 2 * M_PI / orbits[i].timeForFullRotation + ii
          ↪  * 2 * M_PI / orbits[i].numberOfSatellites;

         Vector3 pos(orbits[i].firstSatellite.x *
          ↪  cos(ToRad(orbits[i].firstSatellite.y)) *
          ↪  cos(ToRad(orbits[i].firstSatellite.z)),
```

```
135             orbits[i].firstSatellite.x * cos(ToRad(orbits[i].firstSatellite.y)) *
                ↪  sin(ToRad(orbits[i].firstSatellite.z)),
136             orbits[i].firstSatellite.x * sin(ToRad(orbits[i].firstSatellite.y)));
137          pos = RotateAroundAxis(orbits[i].orbitNormalVector, x, pos);
138
139          if (abs(pos.x) < eps) {
140             pos.x = 0;
141          }
142          if (abs(pos.y) < eps) {
143             pos.y = 0;
144          }
145          if (abs(pos.z) < eps) {
146             pos.z = 0;
147          }
148          Vector3 a = pos - targetPosition;
149          if (a.x != 0) {
150             double A = (1 + a.y * a.y / (a.x * a.x) + a.z * a.z / (a.x * a.x));
151             double B = 2 * (a.y / a.x * (targetPosition.y - a.y / a.x *
                ↪  targetPosition.x) + a.z / a.x * (targetPosition.z - a.z / a.x *
                ↪  targetPosition.x));
152             double C = pow((targetPosition.y - a.y / a.x * targetPosition.x), 2) +
                ↪  pow((targetPosition.z - a.z / a.x * targetPosition.x), 2) -
                ↪  planetRadius * planetRadius;
153             if (B * B - 4 * A * C <= 0) {
154                result++;
155                resOrbit.push_back(i);
156                resSatellite.push_back(ii);
157             }
158             else {
159                Vector3 res[2];
160                res[0].x = (-B - sqrt(B * B - 4 * A * C)) / (2 * A);
161                res[1].x = (-B + sqrt(B * B - 4 * A * C)) / (2 * A);
162                res[0].y = a.y / a.x * (res[0].x - targetPosition.x) +
                   ↪  targetPosition.y;
163                res[1].y = a.y / a.x * (res[1].x - targetPosition.x) +
                   ↪  targetPosition.y;
164                res[0].z = a.z / a.x * (res[0].x - targetPosition.x) +
                   ↪  targetPosition.z;
165                res[1].z = a.z / a.x * (res[1].x - targetPosition.x) +
                   ↪  targetPosition.z;
166
167                Vector3 temp[2];
168                temp[0] = res[0] - targetPosition;
169                temp[1] = res[1] - targetPosition;
170
171                Vector3 solution;
172                for (int i = 0; i < 2; i++) {
173                   if (temp[i].length() > eps) {
174                      solution = res[i];
175                   }
176                }
177                if (a.length() < (pos - solution).length()) {
178                   result++;
179                   resOrbit.push_back(i);
180                   resSatellite.push_back(ii);
181                }
182             }
183          }
184          else if (a.y != 0)
185          {
```

```
186             double A = (1 + a.z * a.z / (a.y * a.y));
187             double B = 2 * a.z / a.y * (targetPosition.z - a.z / a.y *
                ↪  targetPosition.y);
188             double C = pow((targetPosition.z - a.z / a.y * targetPosition.y), 2) +
                ↪  targetPosition.x * targetPosition.x - planetRadius * planetRadius;
189             if (B * B - 4 * A * C <= 0) {
190                result++;
191                resOrbit.push_back(i);
192                resSatellite.push_back(ii);
193             }
194             else {
195                Vector3 res[2];
196                res[0].x = targetPosition.x;
197                res[1].x = targetPosition.x;
198                res[0].y = (-B - sqrt(B * B - 4 * A * C)) / (2 * A);
199                res[1].y = (-B + sqrt(B * B - 4 * A * C)) / (2 * A);
200                res[0].z = a.z / a.y * (res[0].y - targetPosition.y) +
                   ↪  targetPosition.z;
201                res[1].z = a.z / a.y * (res[1].y - targetPosition.y) +
                   ↪  targetPosition.z;
202
203                Vector3 temp[2];
204                temp[0] = res[0] - targetPosition;
205                temp[1] = res[1] - targetPosition;
206
207                Vector3 solution;
208                for (int i = 0; i < 2; i++) {
209                   if (temp[i].length() > eps) {
210                      solution = res[i];
211                   }
212                }
213                if (a.length() < (pos - solution).length()) {
214                   result++;
215                   resOrbit.push_back(i);
216                   resSatellite.push_back(ii);
217                }
218             }
219          }
220          else if (a.z != 0) {
221             double A = (1 + a.y * a.y / (a.z * a.z));
222             double B = 2 * a.y / a.z * (targetPosition.y - a.y / a.z *
                ↪  targetPosition.z);
223             double C = pow((targetPosition.y - a.y / a.z * targetPosition.z), 2) +
                ↪  targetPosition.x * targetPosition.x - planetRadius * planetRadius;
224             if (B * B - 4 * A * C <= 0) {
225                result++;
226                resOrbit.push_back(i);
227                resSatellite.push_back(ii);
228             }
229             else {
230                Vector3 res[2];
231                res[0].x = targetPosition.x;
232                res[1].x = targetPosition.x;
233                res[0].z = (-B - sqrt(B * B - 4 * A * C)) / (2 * A);
234                res[1].z = (-B + sqrt(B * B - 4 * A * C)) / (2 * A);
235                res[0].y = a.y / a.z * (res[0].z - targetPosition.z) +
                   ↪  targetPosition.y;
236                res[1].y = a.y / a.z * (res[1].z - targetPosition.z) +
                   ↪  targetPosition.y;
237
```

```
238                 Vector3 temp[2];
239                 temp[0] = res[0] - targetPosition;
240                 temp[1] = res[1] - targetPosition;
241
242                 Vector3 solution;
243                 for (int i = 0; i < 2; i++) {
244                    if (temp[i].length() > eps) {
245                       solution = res[i];
246                    }
247                 }
248                 if (a.length() < (pos - solution).length()) {
249                    result++;
250                    resOrbit.push_back(i);
251                    resSatellite.push_back(ii);
252                 }
253              }
254           }
255        }
256     }
257     fout << result << endl;
258     for (int i = 0; i < result; i++) {
259        fout << resOrbit[i] + 1 << " " << resSatellite[i] + 1 << endl;
260     }
261  }
```

Задача IV.2. Траектория космического аппарата (30 баллов)

Темы: геометрия, численный метод.

Условие

Из начала координат запускают исследовательский аппарат, который должен достичь небесного тела, движущегося по циркулярной орбите вокруг некоторого неподвижного центра. Из-за больших масштабов размеры аппарата и целевого объекта можно считать незначительными.

В этом случае будем рассматривать их как две материальные точки. Для простоты будем считать, что гравитационными эффектами здесь также можно пренебречь. Предполагается, что аппарат будет запущен по прямой с некоторой постоянной скоростью V.

Траектория небесного тела задается вектором, ортогональным к плоскости эклиптики $\underline{N} = (N_x, N_y, N_z)$, координатами неподвижного центра (C_x, C_y, C_z), начальной точкой (P_x, P_y, P_z) и угловой скоростью Ω. Движение происходит против часовой стрелки в правой системе координат, порожденной вектором $\underline{N}$. Известно, что до момента посадки аппарат не способен маневрировать, а также менять свою скорость. Таким образом, чтобы достичь назначенной цели, ему необходимо попасть ровно в ту точку орбиты, в которой на тот момент будет находиться небесное тело.

Требуется определить направление, в котором должен быть запущен аппарат, чтобы достичь своей цели за минимальное время. Задачу следует решить для серии запросов из различных небесных тел.

Формат входных данных

В начале входного файла хранится целое число n, за которым следует набор из n запросов, каждый из которых задается набором вещественных чисел: N_x, N_y, N_z, C_x, C_y, C_z, P_x, P_y, P_z, Ω, V.

Формат выходных данных

Выходной файл должен содержать ответы на каждый входной запрос, включающий в себя координаты точки орбиты (D_x, D_y, D_Z), в направлении которой должен быть запущен аппарат, указанные с точностью до пятого знака после запятой.

Ограничения

- $-10 \leqslant (N_x, N_y, N_z) \leqslant 10, -10 \leqslant (C_x, C_y, C_z) \leqslant 10$,
- $1 \leqslant |P - C| \leqslant 10, 10^{-1} \leqslant (\Omega, V) \leqslant 10$,
- $1 \leqslant n \leqslant 10^4$.

Гарантируется, что в пределах заданной точности ответ может быть получен однозначно.

Примеры

Пример №1

Стандартный ввод
1 1.00000 1.00000 1.00000 2.00000 -1.00000 0.00000 2.00000 0.00000 -1.00000 0.50000 2.00000
Стандартный вывод
1.52932 0.14849 -0.67781

Тесты

Ссылка на архив с тестовыми наборами данных: https://disk.yandex.ru/d/kdcqXQK2pvZrrA/B_tests.zip.

Решение

Словесное описание алгоритма решения. Дополним систему координат плоскости, в которой лежит орбита небесного тела, векторами:

$$A = P - C, B = N \times A.$$

Выполним их нормировку, чтобы получить ортонормированный базис.

Переведем стартовую точку в новую систему координат:

$$Q = (-C \cdot A, -C \cdot B, -C \cdot N).$$

Расстояние от точки Q до каждой из точек, лежащих на орбите, определяется функцией:

$$d(\varphi) = \sqrt{(r \cdot cos(\varphi) - Q_x)^2 + (r \cdot sin(\varphi) - Q_y)^2} + Q_z^2,$$

где $r = |P - C|$ — радиус окружности,

φ — полярный угол, отсчитываемый от точки P.

Очевидно, что данная функция является периодической по параметру φ с периодом 2π.

Для начала найдем точки ее экстремальных значений (минимума и максимума).

Точки экстремума функции $d(\varphi)$ совпадают с экстремумами функции $q(\varphi) = d^2(\varphi)$.

Чтобы найти их приравняем к нулю ее производную:

$$q'(\varphi) = 2 \cdot r \cdot (Q_x \cdot sin(\varphi) - Q_y \cdot cos(\varphi)) = 0.$$

Множество решений указанного уравнения имеет вид:

$$\varphi(k) = \arctan\left(\frac{Q_y}{Q_x}\right) + k \cdot \pi,$$

либо (в случае, если $Q_x = 0$):

$$\varphi(k) = \frac{\pi}{2} + k,$$

где k — произвольное целое число.

Выберем из них пару значений φ_1 и φ_2, попадающих в интервал $[0, 2\pi]$.

Найдем значения $d_{min} = \min\{d(\varphi_1), d(\varphi_2)\}$, $d_{max} = \max\{d(\varphi_1), d(\varphi_2)\}$.

Особый случай имеет место, если оба значения Q_x и Q_y равны нулю. Это значит, что $d(\varphi)$ имеет постоянное значение на всем интервале.

Время подлета корабля до каждой из точек орбиты определяется функцией: $t(\varphi) - d(\varphi)/V$.

Ее минимальное и максимальное значения имеют вид:

$$t_{min} = \frac{d_{min}}{V}, t_{max} = \frac{d_{max}}{V}.$$

Время пролета небесного тела через точку орбиты определяется функцией: $f(\varphi) = \varphi/\Omega$.

Таким образом, нам требуется найти корни уравнения: $g(\varphi) = t(\varphi) - f(\varphi) = 0$.

Прежде чем искать решение уравнения, выполним отделение его корней. Как утверждалось ранее, функция $t(\varphi)$ периодична по φ с периодом 2π, в то время как $f(\varphi)$ монотонно растет с постоянной скоростью. Следовательно, можно определить интервал $[\varphi_{min}, \varphi_{max}]$, в пределах которого их разность принимает нулевые значения.

Для этого воспользуемся следующими неравенствами:

$$t_{min} - f(\varphi) \leqslant 0, t_{max} - f(\varphi) \geqslant 0.$$

Откуда получим:

$$\varphi_{min} = \Omega \cdot t_{min}, \varphi_{max} = \Omega \cdot t_{max}.$$

Разобьем интервал $[\varphi_{min}, \varphi_{max}]$ выделенными ранее точками $\varphi(k)$, в которых находятся экстремумы функции $t(\varphi)$.

В результате получим набор подынтервалов, на каждом из которых функция $t(\varphi)$ сохраняет монотонный вид.

По очереди перебирая такие интервалы, будем искать точки, в которых $g(\varphi)$ принимает нулевые значения.

На тех интервалах, где функция $t(\varphi)$ монотонно убывает, функция $g(\varphi)$ также будет вести себя монотонно. Тогда для решения уравнения можно воспользоваться двоичным поиском.

На интервалах, где $t(\varphi)$ возрастает, данное правило может нарушаться.

При этом можно заметить, что функция $t(\varphi)$ ведет себя симметрично относительно середины интервала, ускоряясь к середине и замедляясь на концах. В таком случае функция $g(\varphi)$ может иметь не более двух экстремумов, расположенных в разных половинах интервала.

Запустив троичный поиск на каждой из них, найдем точки экстремумов, получив тем самым подынтервалы, на которых $g(\varphi)$ ведет себя монотонно.

...

Полученное решение φ^* определяет точку окружности, в направлении которой должен быть запущен аппарат.

Пример программы-решения

Ниже представлено решение на языке C++.

```
#include <algorithm>
#include <numeric>
#include <limits>
#include <cmath>
#include <array>

typedef double t_scalar; typedef std::array<t_scalar, 3> t_vector;

constexpr t_scalar inf = std::numeric_limits<t_scalar>::infinity();
constexpr t_scalar eps = 1.e-14;

t_vector operator%(const t_vector &lhs, const t_vector &rhs) { return
↪  t_vector{lhs[1] * rhs[2] - lhs[2] * rhs[1], lhs[2] * rhs[0] - lhs[0] * rhs[2],
↪  lhs[0] * rhs[1] - lhs[1] * rhs[0]}; }

t_vector operator-(const t_vector &lhs, const t_vector &rhs) { return
↪  t_vector{lhs[0] - rhs[0], lhs[1] - rhs[1], lhs[2] - rhs[2]}; }

t_vector operator+(const t_vector &lhs, const t_vector &rhs) { return
↪  t_vector{lhs[0] + rhs[0], lhs[1] + rhs[1], lhs[2] + rhs[2]}; }
```

```
17
18  t_scalar operator*(const t_vector &lhs, const t_vector &rhs) { return (lhs[0] *
    ↪  rhs[0] + lhs[1] * rhs[1] + lhs[2] * rhs[2]); }
19
20  t_vector operator*(const t_vector &vec, t_scalar val) { return t_vector{vec[0] *
    ↪  val, vec[1] * val, vec[2] * val}; }
21
22  t_vector operator*(t_scalar val, const t_vector &vec) { return vec * val; }
23
24  t_vector operator/(const t_vector &vec, t_scalar val) { return t_vector{vec[0] /
    ↪  val, vec[1] / val, vec[2] / val}; }
25
26  t_vector operator-(const t_vector &vec) { return -1.0 * vec; }
27
28  t_scalar len(const t_vector &vec) { return sqrt(vec * vec); }
29
30  struct t_scalar_function {
31
32          virtual t_scalar evaluate(t_scalar) const = 0;
33          virtual t_scalar derivate(t_scalar) const = 0;
34
35          virtual ~t_scalar_function() = default;
36  };
37
38  //Функция, задающая время подлета объекта к точке орбиты с заданным значением
    ↪  угла:
39  struct t_object_function:
40  public t_scalar_function {
41
42          t_object_function(t_scalar qx, t_scalar qy, t_scalar qz,
43                            t_scalar v, t_scalar r):
44             _qx(qx), _qy(qy), _qz(qz), _v(v), _r(r) {}
45
46          virtual t_scalar evaluate(t_scalar p) const override {
47             t_scalar rqx = _r * std::cos(p) - _qx;
48             t_scalar rqy = _r * std::sin(p) - _qy;
49             t_scalar q = rqx * rqx + rqy * rqy + _qz * _qz;
50             return std::sqrt(q) / _v;
51          }
52          virtual t_scalar derivate(t_scalar p) const override {
53             t_scalar rqx = _r * std::cos(p) - _qx;
54             t_scalar rqy = _r * std::sin(p) - _qy;
55             t_scalar s = _qx * std::sin(p) -
56                          _qy * std::cos(p);
57             t_scalar q = rqx * rqx + rqy * rqy + _qz * _qz;
58             return (_r * s) /
59                std::sqrt(q) / _v;
60          }
61
62  private:
63          t_scalar _qx, _qy, _qz;
64          t_scalar _v;
65          t_scalar _r;
66  };
67
68  //Функция разности между временем полета объекта и временем оборота небесного
    ↪  тела:
69  struct t_search_function:
70  public t_object_function {
71
```

```
72          t_search_function(t_scalar qx, t_scalar qy, t_scalar qz,
73                            t_scalar v, t_scalar w, t_scalar r):
74          t_object_function(qx, qy, qz, v, r), _w(w) {}
75
76          virtual t_scalar evaluate(t_scalar p) const override {
77                  return t_object_function::evaluate(p) - p / _w;
78          }
79          virtual t_scalar derivate(t_scalar p) const override {
80                  return t_object_function::derivate(p) - 1 / _w;
81          }
82  private:
83          t_scalar _w;
84  };
85
86  #include <iostream>
87  #include <fstream>
88  #include <iomanip>
89  #include <cassert>
90
91  t_scalar find2(const t_scalar_function &F,
92                   t_scalar a, t_scalar b, t_scalar e = eps) {
93
94          t_scalar p1 = a, f1 = F.evaluate(p1);
95          t_scalar p2 = b, f2 = F.evaluate(p2);
96          if (f1 * f2 > 0) {
97             return std::numeric_limits<t_scalar>::quiet_NaN();
98          }
99          const int s = (f1 < 0)? -1: +1;
100         while (std::abs(f1) > e) {
101            t_scalar p = (p1 + p2) / 2., f = F.evaluate(p);
102            if (s * f < 0) {
103               p2 = p; f2 = f;
104            }
105            else {
106               p1 = p; f1 = f;
107            }
108         }
109         return p1;
110 }
111
112 t_scalar find3(const t_scalar_function &F,
113                  t_scalar a, t_scalar b, t_scalar e = eps) {
114
115         const int s = (F.derivate(a) < 0)? -1: +1;
116         t_scalar d;
117         while ((d = (b - a)) > e) {
118            d = d / 3;
119            auto pa = a + d, fa = F.evaluate(pa);
120            auto pb = b - d, fb = F.evaluate(pb);
121            if ((fa - fb) * s < 0)
122               a = pa;
123            else {
124               b = pb;
125            }
126         }
127         return a;
128 }
129
130 int main() {
131
```

```
132         const t_scalar PI2 = 2.0 * std::acos(-1.0);
133         const t_scalar PI = std::acos(-1.0);
134
135         std::ofstream fout("output.txt");
136         fout << std::setprecision(5) << std::fixed;
137
138         std::ifstream finp("input.txt");
139         size_t num; finp >> num;
140
141         for (int i = 0; i < num; ++ i) {
142                 //Считываем параметры траектории орбиты:
143                 t_vector C, N, P;
144                 t_scalar W, V;
145                 finp >> N[0] >> N[1] >> N[2];
146                 finp >> C[0] >> C[1] >> C[2];
147                 finp >> P[0] >> P[1] >> P[2];
148                 finp >> W;
149                 finp >> V;
150                 //Check:
151                 assert(
152                 std::abs(N * (P - C)) < eps
153                 );
154                 //Формируем локальную систему координат:
155                 t_vector A = P - C;
156                 t_scalar R = len(A);
157                 N = N / len(N);
158                 A = A / R;
159                 t_vector B = N % A;
160                 t_vector Q = {
161                    - C * A,
162                    - C * B,
163                    - C * N
164                 };
165                 //Находим точки экстремумов:
166                 t_search_function FS(Q[0], Q[1], Q[2], V, W, R);
167                 t_object_function FO(Q[0], Q[1], Q[2], V, R);
168                 t_scalar t1 = 0., t2 = 0.;
169                 if (std::abs(Q[0]) > eps) {
170                    t1 = std::atan(Q[1] / Q[0]);
171                    if (t1 < - eps) t1 += PI;
172                 }
173                 else {
174                    t1 = PI / 2.;
175                 }
176                 t2 = t1 + PI;
177                 t_scalar F1 = FO.evaluate(t1);
178                 t_scalar F2 = FO.evaluate(t2);
179                 if (F1 > F2) std::swap(F1,F2);
180                 //...
181                 t_scalar s1 = find3(
182                 FS, t1, t1 + PI / 2., 1.e-8
183                 );
184                 t_scalar s2 = find3(
185                 FS, t2 - PI / 2., t2, 1.e-8
186                 );
187                 //Сдвигаем диапазон поиска:
188                 t_scalar p1 = F1 * W;
189                 t_scalar p2 = F2 * W;
190                 //Определяем интервалы, на которых функция сохраняет монотонный
                    ↪  вид:
```

```
191                 std::array<t_scalar, 4> s = {0, s1 - t1, s2 - t1, t2 - t1};
192                 int t = int((p1 - t1) / PI);
193                 if (t1 + t * PI > p1) -- t;
194                 if (std::abs(p1 - p2) > eps) {
195                    p1 = t1 + t * PI; ++ t;
196                 }
197                 //Поочередно запускаем поиск корней на каждом из интервалов:
198                 t_scalar p0 = p1;
199                 while (p1 < p2) {
200                    for (int k = 0; k < 3; ++ k) {
201                       p0 = find2(
202                       FS, p1 + s[k], p1 + s[k + 1], 1.e-8
203                       );
204                       if (!std::isnan(p0)) break;
205                    }
206                    if (!std::isnan(p0)) break;
207                    p1 = t1 + t * PI;
208                    ++ t;
209                 }
210                 assert(!std::isnan(p0));
211                 t_vector D = R * (
212                 std::cos(p0) * A +
213                 std::sin(p0) * B
214                 ) + C;
215                 fout << D[0] << " ";
216                 fout << D[1] << " ";
217                 fout << D[2];
218                 fout << "\n";
219                 //Check:
220                 t1 = len(D) / V;
221                 t2 = p0 / W;
222                 assert(
223                 std::abs(t1 - t2) <
224                 1.e-7
225                 );
226           }
227
228     return 0;
229  }
```

Задача IV.3. Тор с подвижными частицами (30 баллов)

Темы: геометрия, численный метод.

Условие

В одной исследовательской лаборатории собираются провести очередной эксперимент с подвижными частицами, в рамках которого им придется взаимодействовать с некоторым объектом, имеющим форму тора.

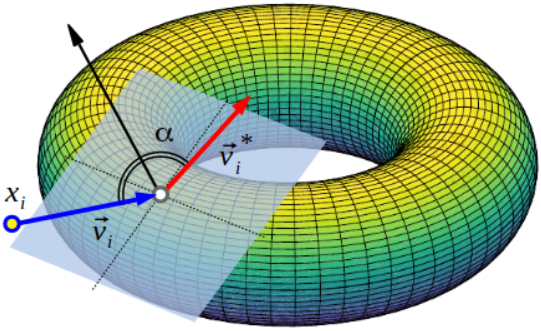
Известно, что каждая такая частица в течение всей своей жизни совершает непрерывное прямолинейное движение с некоторой постоянной скоростью. При столкновении с каким-либо препятствием она отклоняется от исходной траектории, а затем продолжает свое движение как ни в чем не бывало. В очень редких случаях также могут встречаться абсолютно неподвижные частицы, скорости которых равны нулю.

Будем полагать, что тор задается радиусами направляющей R_1 и образующей

R_2 окружностей. Его ось вращения совпадает с осью O_z, а центр лежит в начале координат. Первоначально все исходные частицы можно условно разбить на две группы: лежащие внутри и снаружи тора. Важным условием здесь является то, что в процессе движения ни одна из частиц не должна покидать область, которой она принадлежала в начальный момент времени.

Рассмотрим поведение указанных частиц на временном интервале от 0 до T.

В процессе своего движения каждая отдельно взятая частица может то и дело сталкиваться с границей тора. При столкновении с тором частица отражается от вектора нормали в соответствующей точке (см. рисунок). Модуль ее скорости при этом сохраняется.



Требуется написать программу, которая по известному начальному положению частиц, исходным компонентам скорости и параметрам тора производит расчет их траекторий в течение заданного промежутка времени.

В качестве ответа выведите координаты положения частиц в равноотстоящие моменты времени: $t_k = k \cdot T/m$, где $k = 1, 2, \ldots, m$.

Формат входных данных

В первой строке входного файла содержатся радиусы направляющей R_1 и образующей R_2 окружностей тора. Во второй строке указано значение T, задающее длину временного интервала, и число его подынтервалов m. Далее следует натуральное число n и последовательность из $6 \times n$ вещественных чисел, задающих начальные координаты и скорости каждой из частиц: X_i, Y_i, Z_i, U_i, V_i, W_i.

Формат выходных данных

Выходной файл должен содержать координаты положения частиц в моменты времени t_k, расположенные в том же порядке, что и во входном файле. Все значения должны быть указаны с точностью до пятого знака после запятой.

Ограничения

Предполагается, что начальное расстояние от каждой частицы до поверхности тора $\geqslant 10^{-2}$.

Гарантируется, что для каждой отдельной частицы число столкновений с тором на заданном интервале не превосходит 20.

- $10 \geqslant (R_1 - 1) \geqslant R_2 \geqslant 1, 0 < T \leqslant 10, 0 < m \leqslant 10,$
- $-10 \leqslant (X_i, Y_i, Z_i, U_i, V_i, W_i) \leqslant 10,$
- $0 < n \leqslant 4000$..

Примеры

Пример №1

Стандартный ввод
5.00000 2.00000 1.00000 4 3 -4.50000 2.00000 0.00000 -3.50000 -4.25000 1.50000 -1.50000 2.00000 2.00000 -0.50000 -3.00000 -4.00000 1.00000 1.00000 1.00000 3.00000 3.00000 0.80000

Стандартный вывод
-5.37500 0.93750 0.37500 -1.62500 1.25000 1.00000 1.75000 1.75000 1.20000 -6.25000 -0.12500 0.75000 -1.75000 0.50000 0.00000 2.50000 2.50000 1.40000 -6.28766 -1.10048 0.66906 -1.87500 -0.25000 -1.00000 2.66100 2.66100 2.42614 -5.34270 -1.97383 0.05307 -2.00000 -1.00000 -2.00000 2.79624 2.79624 3.48841

Тесты

Ссылка на архив с тестовыми наборами данных: https://disk.yandex.ru/d/kdcqXQK2pvZrrA/TOR.zip.

Решение

Словесное описание алгоритма решения.

Будем последовательно отслеживать изменение траектории каждой отдельно взятой частицы на заданном интервале $[0, T]$.

Поиск ближайшего столкновения

Для каждой i-й частицы траектория движения представляет собой прямую, которая может быть описана следующим параметрическим уравнением:

$$x(t) = x_0 + t \cdot u,\ y(t) = y_0 + t \cdot v,\ z(t) = z_0 + t \cdot w.$$

В свою очередь, каноническое уравнение тора имеет следующий вид:

$$(x^2 + y^2 + z^2 + R_1^2 - R_2^2)^2 - 4 \cdot R_1^2 \cdot (x^2 + y^2) = 0.$$

Подставив в него параметрические выражения для $x(t)$, $y(t)$ и $z(t)$, получим алгебраическое уравнение четвертой степени относительно t.

$$F(t) = A \cdot t^4 + B \cdot t^3 + C \cdot t^2 + D \cdot t + E = 0.$$

Существующие аналитические методы решения подобных уравнений (такие, как метод Феррари) на практике могут приводить к накоплению ошибок вычислений. По этой причине для решения уравнений высоких степеней, как правило, прибегают к использованию численных методов.

Однако для их применения необходимо определить подынтервалы, на которых функция $F(t)$ имеет не более одного корня.

Рассмотрим метод отделения корней для многочлена степени n. Известно, что число действительных корней не превосходит n. При этом искать их следует между точками экстремумов, где производная обращается в ноль: $F(t) = 0$. Т. к. производная многочлена сама является многочленом меньшей степени, получим аналогичную задачу для $n - 1$, где, очевидно, нерекурсивным случаем будет $n \leqslant 2$.

После того как найдены точки экстремумов, переходим к поиску корней. Для этого можно воспользоваться любым доступным методом (например, двоичным поиском). Из множества найденных корней выбираем наименьший, попадающий в расчетный интервал.

Корректировка траектории

После того как был найден ближайший момент столкновения и соответствующая ему точка, следует получить вектор нормали к поверхности тора. Для этого возьмем радиус-вектор, связанный с найденной точкой $\underline{X} = (x, y, z)$ и спроецируем его на плоскость XY, в которой лежит направляющая окружность. Иначе говоря, занулим z-компоненту $\underline{X^*} = (x, y, 0)$.

Нормируем его и умножим на R_1, получив вектор $\underline{P} = R_1 \cdot \dfrac{\underline{X^*}}{|\underline{X^*}|}$.

Вектор $\underline{P}$ связан с ближайшей к нам точкой направляющей окружности.

Вектор нормали в точке $\underline{X}$ можно получить как $\underline{N} = \underline{X} - \underline{P}$.

Дополнительно выполним для него нормировку. Остается отразить вектор скорости $\underline{V}$ от вектора нормали:

$$\underline{V^*} = \underline{V} - 2(\underline{N} \cdot \underline{V}) \cdot \underline{N}.$$

Трекинг частиц

Описанная выше схема позволяет найти столкновение и изменить направление движения частицы.

Однако нам нужно научиться определять положения частиц в отдельно взятые моменты времени t_k.

Сделать это несложно. Если в процессе поиска столкновений мы достигли момента времени $t^* \geqslant t_k$, это значит, что прежде чем менять траекторию нужно вывести накопленный ранее результат. Для этого можно запоминать момент последнего обработанного столкновения, а также связанные с ним координаты частицы и компоненты скорости.

Особое внимание здесь следует уделить тому, чтобы после пересчета траектории не засчитать повторно момент предыдущего столкновения. По этой причине при отборе возможных столкновений следует тестировать их на предмет выхода частицы за пределы содержащей ее области. Иначе говоря, проекция отраженного вектора скорости на вектор нормали должна сохранять знак на всем временном интервале.

Пример программы-решения

Ниже представлено решение на языке C++.

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <cassert>
#include <algorithm>
#include <numeric>
#include <limits>
#include <cmath>
#include <vector>
#include <array>

typedef double t_scalar; typedef std::array<t_scalar, 3> t_vector;

constexpr t_scalar inf = std::numeric_limits<t_scalar>::infinity();
constexpr t_scalar eps = 1.e-14;

t_vector operator%(const t_vector &lhs, const t_vector &rhs) { return
↪  t_vector{lhs[1] * rhs[2] - lhs[2] * rhs[1], lhs[2] * rhs[0] - lhs[0] * rhs[2],
↪  lhs[0] * rhs[1] - lhs[1] * rhs[0]}; }

t_vector operator-(const t_vector &lhs, const t_vector &rhs) { return
↪  t_vector{lhs[0] - rhs[0], lhs[1] - rhs[1], lhs[2] - rhs[2]}; }

t_vector operator+(const t_vector &lhs, const t_vector &rhs) { return
↪  t_vector{lhs[0] + rhs[0], lhs[1] + rhs[1], lhs[2] + rhs[2]}; }

t_scalar operator*(const t_vector &lhs, const t_vector &rhs) { return (lhs[0] *
↪  rhs[0] + lhs[1] * rhs[1] + lhs[2] * rhs[2]); }

t_vector operator*(const t_vector &vec, t_scalar val) { return t_vector{vec[0] *
↪  val, vec[1] * val, vec[2] * val}; }

t_vector operator*(t_scalar val, const t_vector &vec) { return vec * val; }

t_vector operator/(const t_vector &vec, t_scalar val) { return t_vector{vec[0] /
↪  val, vec[1] / val, vec[2] / val}; }

t_vector operator-(const t_vector &vec) { return -1.0 * vec; }

```

```
33 t_scalar len(const t_vector &vec) { return sqrt(vec * vec); }
34
35 std::ostream& operator<<(std::ostream &out, const t_vector &vec) {
36    return out << vec[0] << " " << vec[1] << " " << vec[2];
37 }
38 std::istream& operator>>(std::istream &inp, t_vector &vec) {
39    return inp >> vec[0] >> vec[1] >> vec[2];
40 }
41
42 struct t_range {
43
44    bool contain(t_scalar c) const { return (_data[0] <= c) && (c <= _data[1]); }
45
46    auto operator[](int i) const { return _data[i]; }
47
48    t_range(t_scalar l, t_scalar r): _data{std::min(l, r), std::max(l, r)} {}
49
50 private:
51    std::array<t_scalar, 2> _data;
52 };
53
54 struct t_func {
55
56    virtual t_scalar evaluate(t_scalar p) const = 0;
57
58    virtual ~t_func() = default;
59 };
60
61 template <unsigned N>
62 struct t_poly:
63 public t_func {
64
65    t_poly(const std::array<t_scalar, N + 1> &fact): _fact(fact) {}
66
67    virtual t_scalar evaluate(t_scalar p) const override {
68       t_scalar s = 0.;
69       for (int i = N; i >= 0; -- i) s = s * p + _fact[i];
70       return s;
71    }
72    t_scalar operator[](int i) const { return _fact[i]; }
73
74    t_poly<N - 1> derivate() const {
75       std::array<t_scalar, N> f;
76       for (int i = 1; i <= N; ++ i)
77       f[i - 1] = _fact[i] * i;
78       return f;
79    }
80
81 private:
82    std::array<t_scalar, N + 1> _fact;
83 };
84
85 //Двоичный поиск нулей монотонной функции на заданом интервале
86 t_scalar binSearch(const t_func &F, const t_range &R, t_scalar e = eps) {
87
88    t_scalar p1 = R[0], f1 = F.evaluate(p1);
89    t_scalar p2 = R[1], f2 = F.evaluate(p2);
90    if (f1 * f2 > 0) {
91       return std::numeric_limits<t_scalar>::quiet_NaN();
92    }
```

```
93      const int s = (f1 < 0)? -1: +1;
94      while (std::abs(f1) > e) {
95         t_scalar p = (p1 + p2) / 2., f = F.evaluate(p);
96         if (s * f <= 0) {
97            p2 = p; f2 = f;
98         }
99         else {
100           p1 = p; f1 = f;
101        }
102     }
103     return p1;
104  }
105
106  //Поиск корней полиномов на заданном интервале
107
108  std::vector<t_scalar> solve(const t_poly<1> &P, const t_range& R, t_scalar e =
     ↪  eps) {
109
110     t_scalar c = - P[0] / P[1]; if (R.contain(c)) return {c};
111     return {};
112  }
113
114  /*std::vector<t_scalar> solve(const t_poly<2> &P, const t_range& R, t_scalar e =
     ↪  eps) {
115
116     if (std::abs(P[2]) < eps) {
117        return solve(t_poly<1>({P[0], P[1]}), R, e);
118     }
119     t_scalar d = P[1] * P[1] - 4. * P[2] * P[0];
120
121     std::vector<t_scalar> t; t.reserve(2);
122     if (d >= 0) {
123        t_scalar c = (- P[1] - std::sqrt(d)) / P[2] / 2;
124        if (R.contain(c)) t.push_back(c);
125        c = (- P[1] + std::sqrt(d)) / P[2] / 2;
126        if (R.contain(c)) t.push_back(c);
127     }
128     if ((t.size() > 1) && (t[0] > t[1])) {
129        std::swap(t[0], t[1]);
130     }
131     return t;
132  }*/
133
134  template <unsigned N>
135  std::vector<t_scalar> solve(const t_poly<N> &P, const t_range& R, t_scalar e =
     ↪  eps) {
136
137     std::vector<t_scalar> z; z.reserve(N + 2);
138     std::vector<t_scalar> t; t.reserve(N);
139
140     z.push_back(R[0]);
141     for (auto p: solve(P.derivate(), R, e))
142        z.push_back(p);
143     z.push_back(R[1]);
144
145     for (int i = 1; i < z.size(); ++ i) {
146        auto p = binSearch(
147        P, {z[i - 1], z[i]}, e
148        );
149        if (!std::isnan(p)) {
```

```
150             t.push_back(p);
151          }
152       }
153       return t;
154    }
155
156    int main() {
157
158       std::ofstream fout("output.txt");
159       fout << std::setprecision(5) << std::fixed;
160
161       std::ifstream finp("input.txt");
162       t_scalar R1, R2; finp >> R1 >> R2;
163       t_scalar max_t; finp >> max_t;
164       size_t num_t; finp >> num_t;
165       auto len_t = max_t / num_t;
166       size_t num_p; finp >> num_p;
167
168       std::vector<t_vector> X1(num_p), V1(num_p), X2(num_p), V2(num_p);
169       std::vector<t_scalar> T1(num_p);   //Момент последнего зафиксированного
          ↪  столкновения
170       std::vector<t_scalar> T2(num_p);   //Момент следующего зафиксированного
          ↪  столкновения
171       std::vector<int> S(num_p);   //+1 -- для внеших частиц, -1 -- для внутренних
172
173       //Считываем координаты частиц и компоненты скоростей
174       for (int i = 0; i < num_p; ++ i) {
175          finp >> X2[i] >> V2[i];
176          //Проверяем начальное положение частицы:
177          t_vector C = {X2[i][0], X2[i][1], 0.};
178          t_scalar d = len(C);
179          if (d > eps) {
180             C = R1 * C / d;
181             if (len(X2[i] - C) < R2) {
182                S[i] = -1;
183             }
184             else {
185                S[i] = 1;
186             }
187          }
188          else {
189             S[i] = +1;
190          }
191       }
192
193       //Запускам процедуру трекинга:
194       for (int k = 1; k <= num_t; ++ k) for (int i = 0; i < num_p; ++ i) {
195
196          t_scalar cur_t = k * len_t;
197
198          while (T2[i] < cur_t) {
199
200             X1[i] = X2[i]; V1[i] = V2[i]; T1[i] = T2[i];
201
202             //Заполняем коэффициенты уравнения:
203             const t_scalar x = X2[i][0];
204             const t_scalar y = X2[i][1];
205             const t_scalar z = X2[i][2];
206             const t_scalar u = V2[i][0];
207             const t_scalar v = V2[i][1];
```

```
208             const t_scalar w = V2[i][2];
209             const t_scalar uuvv = u * u + v * v;
210             const t_scalar xuyv = x * u + y * v;
211             const t_scalar xxyy = x * x + y * y;
212             const t_scalar c = (xxyy + z * z) +
213                       (R1 * R1 - R2 * R2);
214             const t_scalar b = 2 * (xuyv + z * w);
215             const t_scalar a = (uuvv + w * w);
216             const t_scalar q = 4 * R1 * R1;
217             const t_poly<4> F({
218                (c * c) - q * xxyy,
219             2 * (b * c - q * xuyv),
220             2 * (c * a) +
221                 b * b - q * uuvv,
222             2 * (a * b),
223                (a * a)
224             });
225
226             //Запускаем поиск корней на интервале:
227             auto tt = solve(F, t_range{0., max_t}, 1.e-9);
228             //Выбираем ближайший подходящий момент столкновения:
229             bool is = false;
230             for (auto t: tt) {
231                //Пересчитываем положение частицы:
232                t_vector XX = X2[i] + t * V2[i];
233                //Ближайшая точка на окружности:
234                t_vector C = {XX[0], XX[1], 0.};
235                C =  (R1 / len(C)) * C;
236                //Нормаль в точке столкновения:
237                t_vector N = XX - C; N = S[i] * N / len(N);
238                //Пересчитываем скорости:
239                t_vector VV =
240                V2[i] - 2 * (V2[i] * N) * N;
241                //Проверяем попадание в допустимую область!
242                if (N * VV < 0) continue;
243                //Фиксируем результаты:
244                T2[i] = T2[i] + t;
245                X2[i] = XX;
246                V2[i] = VV;
247                is = true; break;
248             }
249             if (!is) {
250                T2[i] = max_t;
251                break;
252             }
253          }
254          //Выводим текущее положение частицы:
255          t_scalar t = cur_t - T1[i];
256          t_vector C = X1[i] +
257                 t * V1[i];
258          fout << C << "\n";
259       }
260
261       return 0;
262    }
```

Задача IV.4. Earth VR (20 баллов)

Темы: Unity, алгебра, программирование C#.

Условие

Юный программист Вася работает над своим новым проектом в виртуальной реальности на Unity — он создает виртуальную копию нашей планеты в масштабе 1 : 1. Пока готово несколько небольших отдельных локаций, на которых он разметил схематичные здания, дороги, а также спроецировал изображение со спутника на поверхность. Поверхность каждой локации плоская, и направление севера совпадает с направлением оси Z в координатах Unity.

Вася планирует сделать проект многопользовательским и уже тестирует систему подключения игроков к виртуальному миру. Он хочет, чтобы пользователь сначала выбрал на карте свое местоположение, которое соответствует географическим координатам (φ_1, ψ_1), после чего тот появится на игровой сцене в координатах $(x, 0, z)$, которые соответствуют выбранному местоположению.

Допустим, первый игрок уже подключился и находится на сцене в координатах $(x_1, 0, z_1)$, Положение первого игрока на сцене соответствует географическим координатам (φ_1, ψ_1). Второй пользователь выбрал на карте географические координаты (φ_2, ψ_2). Помогите Васе разместить игрока на позиции в локации, которая бы соответствовала его географическим координатам.

Вася подготовил сцену и написал несколько вспомогательных скриптов, вам осталось лишь реализовать функцию `SpawnCoordinates()` типа `Vector2` в классе `Solution`. Эта функция будет вызываться на старте программы, возвращать значение (x_2, z_2), и с ее помощью Вася присвоит модели второго игрока его игровые координаты $(x_2, 0, z_2)$.

Весь ваш код должен находится в классе `Solution`. Другие изменения в сцене учитываться не будут. Объект manager предоставляет следующие функции для работы с входными данными:

- `manager.GetFirstPlayerPosition()` — возвращает координаты первого игрока на сцене Unity в виде `Vector2 (`x_1`,`z_1`)`, где x_1,z_2 — компоненты позиции игрока;
- `manager.GetEarthCoordsOfFirstPlayer()` — возвращает географические координаты первого игрока в виде массива `double[2]`, где первый элемент — широта (в градусах), второй — долгота (в градусах);
- `manager.GetEarthCoordsOfSecondPlayer()` — возвращает географические координаты второго игрока в виде массива `double[2]`, где первый элемент — широта (в градусах), второй — долгота (в градусах).

Проект доступен по этой ссылке: https://github.com/ntomaterials/Earth-VR.

Вам осталось реализовать класс `Solution`. Доступны две локации для отладки вашего решения – кампус ДВФУ на острове Русский и Красная площадь в Москве. Модели локаций были получены с помощью OpenStreetMap. Решение будет проверяться на бóльшем количестве локаций, расположенных в разных частях мира.

Для тестирования своего решения вы можете изменять данные, находящиеся в файле `input.txt` в корне папки с проектом.

Процедура тестирования осуществляется следующим образом:

- Выберите сцену с одной из двух локаций и откройте ее.
- Разместите объект `Test Player` в произвольной точке на игровой модели, и запишите в `input.txt` два числа через пробел x_1, z_1 — компоненты позиции игрока.

- С помощью любого картографического сервиса найдите выбранную вами точку на карте и скопируйте ее географические координаты — запишите их во вторую строку через пробел без запятой (сначала широту, затем долготу).
- Выберите на карте любую другую точку и запишите ее координаты на третьей строке в том же формате.
- Запустите сцену в Unity и найдите объект Second Player.
- Оцените его расположение в игре с той точкой, которую вы отметили на карте — он должен попасть в радиус 5 м от нее.

Примечания:

- вам не нужно самостоятельно реализовывать считывание из файла `input.txt`; вместо этого используйте соответствующие функции в объекте `manager`;
- вы не можете программно получать информацию о любых объектах, находящихся на сцене;
- в зависимости от выбранного картографического сервиса погрешность решения может варьироваться;
- для большей надежности проверяйте решение на сервисе, спутниковое изображение на котором похоже на предоставленное в локации.

Формат входных данных

В первой строке находится 2 числа x_1 и z_1 — координаты первого игрока на сцене в Unity. Во второй строке находится 2 числа φ_1 и ψ_1 — географические координаты первого игрока. В третьей строке находится 2 числа φ_2 и ψ_2 — географические координаты второго игрока.

Формат выходных данных

Выходной файл содержит два числа x_2 и z_2 — координаты второго игрока на сцене в Unity. Допускается любой ответ с погрешностью не более 5 м. Реализовывать вывод в файл `output.txt` не нужно.

Ограничения

$$-90 \leqslant \varphi_i \leqslant 90, -180 \leqslant \psi_i \leqslant 180, i \in 1, 2.$$

Примеры

Пример №1

Стандартный ввод

`-587.68 135.34`
`43.03001773493322 131.89306628571381`
`43.029822179162174 131.8905516024877`

Стандартный вывод
`-793.07 113.5914`

Тесты

Ссылка на архив с тестовыми наборами данных: https://disk.yandex.ru/d/kdcqXQK2pvZrrA/A_EarthVR.zip.

Пример программы-решения

Ниже представлено решение на языке C#.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System;

public class Solution : MonoBehaviour
{
    private SceneManager manager;
    private double EarthRadius = 6371210;
    private double eps = 1;

    public Solution(SceneManager manager)
    {
        this.manager = manager;
    }

    public Vector2 SpawnCoordinates()
    {
        double[] firstPlayerWorldPos = manager.GetEarthCoordsOfFirstPlayer();
        double firstPlayerLatitude = firstPlayerWorldPos[0];
        double firstPlayerLongitude = firstPlayerWorldPos[1];
        double northLatitude = 90;
        double[] secondPlayerWorldPos = manager.GetEarthCoordsOfSecondPlayer();
        double secondPlayerLatitude = secondPlayerWorldPos[0];
        double secondPlayerLongitude = secondPlayerWorldPos[1];

        if (Math.Abs(firstPlayerLatitude - secondPlayerLatitude) < 0.00000001 &&
        ↪  Math.Abs(firstPlayerLongitude - secondPlayerLongitude) < 0.00000001)
        {
            return new Vector2(manager.GetFirstPlayerPosition().x,
            ↪  manager.GetFirstPlayerPosition().y);
        }
        double northLongitude = secondPlayerLongitude;
        double eastLatitude = secondPlayerLatitude;
        double eastLongitude = secondPlayerLongitude + 1;
        double distFirstToSecond = Math.Acos(Math.Sin(firstPlayerLatitude * Math.PI
        ↪  / 180) * Math.Sin(secondPlayerLatitude * Math.PI / 180) +
        ↪  Math.Cos(firstPlayerLatitude * Math.PI / 180) *
        ↪  Math.Cos(secondPlayerLatitude * Math.PI / 180) *
        ↪  Math.Cos((firstPlayerLongitude - secondPlayerLongitude) * Math.PI /
        ↪  180)) * EarthRadius;

        double distNorthToSecond = Math.Acos(Math.Sin(northLatitude * Math.PI / 180)
        ↪  * Math.Sin(secondPlayerLatitude * Math.PI / 180) +
        ↪  Math.Cos(northLatitude * Math.PI / 180) * Math.Cos(secondPlayerLatitude
        ↪  * Math.PI / 180) * Math.Cos((northLongitude - secondPlayerLongitude) *
        ↪  Math.PI / 180)) * EarthRadius;

```

```
38          double distFirstToNorth = Math.Acos(Math.Sin(northLatitude * Math.PI / 180)
            ↪  * Math.Sin(firstPlayerLatitude * Math.PI / 180) + Math.Cos(northLatitude
            ↪  * Math.PI / 180) * Math.Cos(firstPlayerLatitude * Math.PI / 180) *
            ↪  Math.Cos((northLongitude - firstPlayerLongitude) * Math.PI / 180)) *
            ↪  EarthRadius;
39
40          double distEastToSecond = Math.Acos(Math.Sin(eastLatitude * Math.PI / 180) *
            ↪  Math.Sin(secondPlayerLatitude * Math.PI / 180) + Math.Cos(eastLatitude *
            ↪  Math.PI / 180) * Math.Cos(secondPlayerLatitude * Math.PI / 180) *
            ↪  Math.Cos((eastLongitude - secondPlayerLongitude) * Math.PI / 180)) *
            ↪  EarthRadius;
41
42          double distFirstToEast = Math.Acos(Math.Sin(eastLatitude * Math.PI / 180) *
            ↪  Math.Sin(firstPlayerLatitude * Math.PI / 180) + Math.Cos(eastLatitude *
            ↪  Math.PI / 180) * Math.Cos(firstPlayerLatitude * Math.PI / 180) *
            ↪  Math.Cos((eastLongitude - firstPlayerLongitude) * Math.PI / 180)) *
            ↪  EarthRadius;
43
44          double angDistFirstToSecond = distFirstToSecond / EarthRadius;
45          double angDistNorthToSecond = distNorthToSecond / EarthRadius;
46          double angDistFirstToNorth = distFirstToNorth / EarthRadius;
47
48          double angDistEastToSecond = distEastToSecond / EarthRadius;
49          double angDistFirstToEast = distFirstToEast / EarthRadius;
50          double angFirstToSecond_NorthToSecond = (Math.Cos(angDistFirstToNorth) -
            ↪  Math.Cos(angDistFirstToSecond) * Math.Cos(angDistNorthToSecond)) /
            ↪  (Math.Sin(angDistFirstToSecond) * Math.Sin(angDistNorthToSecond));
51
52          if (Math.Abs(angFirstToSecond_NorthToSecond) > 1.0f)
53          {
54              angFirstToSecond_NorthToSecond =
                ↪  Math.Sign(angFirstToSecond_NorthToSecond);
55          }
56          double azimuthNorth = Math.Acos(angFirstToSecond_NorthToSecond);
57
58          azimuthNorth *= 180 / Math.PI;
59
60          double azimuthEast = Math.Acos((Math.Cos(angDistFirstToEast) -
            ↪  Math.Cos(angDistFirstToSecond) * Math.Cos(angDistEastToSecond)) /
            ↪  (Math.Sin(angDistFirstToSecond) * Math.Sin(angDistEastToSecond)));
61          azimuthEast *= 180 / Math.PI;
62
63          if (90 - eps <= azimuthEast - azimuthNorth && azimuthEast - azimuthNorth <=
            ↪  90 + eps || 270 - eps <= azimuthEast + azimuthNorth && azimuthEast +
            ↪  azimuthNorth <= 270 + eps)
64          {
65              azimuthNorth = 360 - azimuthNorth;
66          }
67          Vector2 firstPlayerPos = manager.GetFirstPlayerPosition();
68          Vector3 secondPlayerPos = Quaternion.Euler(0, ((float)azimuthNorth - 180),
            ↪  0) * new Vector3(0, 0, (float)distFirstToSecond);
69          Vector2 result = new Vector2(secondPlayerPos.x, secondPlayerPos.z) + new
            ↪  Vector2(firstPlayerPos.x, firstPlayerPos.y);
70          return result;
71      }
72  }
```

Работа наставника НТО при подготовке к заключительному 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. Для подготовки рекомендуется использовать сборники задач прошлых лет. Кроме того, наставнику важно изучить организационные особенности заключительного этапа, чтобы помочь ученикам разобраться в формальных особенностях его проведения.

Наставник НТО также может познакомиться с разработчиками профилей для получения консультации о подготовке к заключительному этапу, дополнительных материалах и способах поддержки высокой мотивации участников.

При работе с командой участников рекомендуется уделить внимание следующим вопросам:

- Сплочение команды. Наставнику необходимо уделить этому особенное внимание, если участники команды находятся в разных городах и не имеют возможности встретиться в очном формате. Регулярные встречи, в том числе в дистанционном формате, помогут поддержать эффективную и позитивную коммуникацию внутри команды.
- Анализ состава команды. Необходимо обсудить роли участников в команде и задачи, которые им предстоит решать в рамках выбранных ролей. Кроме того, нужно обсудить взаимозаменяемость ролей.
- Анализ знаний и компетенций участников. Необходимо убедиться, что участники обладают нужными навыками и компетенциями и продумать план по формированию и развитию недостающих навыков и компетенций.
- Составление плана подготовки. График занятий строится, исходя из даты начала заключительного этапа.
- Участие в подготовительных мероприятиях от разработчиков профилей. Перед заключительным этапом проводятся установочные вебинары, разборы задач прошлых лет, практикумы, хакатоны, мастер-классы для финалистов. Информация о таких мероприятиях публикуется в группе НТО в VK и в чатах профилей в Telegram.
- Проведение практикумов или хакатонов. Для этого наставники могут использовать материалы для подготовки к соответствующему профилю и сборники задач прошлых лет. Практикумы и хакатоны могут проводиться дистанционно, рекомендации для этого формата приведены в сборниках 2020–22 гг.

Во время заключительного этапа участников сопровождают модераторы или вожатые, разработчики профиля и организаторы НТО. Внешнее вмешательство в ход соревнований запрещено. Участники, получившие во время проведения НТО стороннюю помощь, могут быть дисквалифицированы.

Заключительный этап

Предметный тур

Информатика и информационные технологии. 8–11 классы

Тестовые наборы для задач представлены по ссылке — https://disk.yandex.ru/d/zY6GsraK2tRVnw.

Задача VI.1.1.1. Кодирование набора чисел (100 баллов)

Условие

Пусть имеется последовательность целых десятичных чисел строго больших нуля: 4 3 6 7 10 26 100 40 21 14 12 22 148 29 18 81 28 31 27 20.

Требуется закодировать данную последовательность следующим образом:

1. каждое такое число необходимо перевести в двоичную систему счисления, задействовав минимально возможное число бит, в которое оно может уместиться;
2. выполнить конкатенацию (слияние, склеивание) всех полученных кодов в порядке исходной последовательности;
3. полученное двоичное число следует представить в 16-ричной системе счисления, задействовав цифры от 0 до 9 и строчные буквы латинского алфавита от от a до f.

Решение

Для получения ответа следует выполнить пошагово следующее:

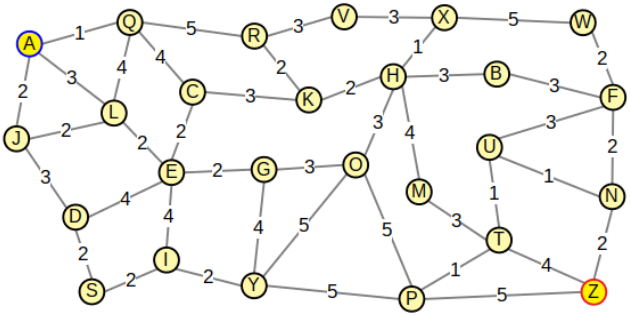
1. Каждое число из последовательности необходимо представить в двоичной системе $4 = 100_2$, $3 = 11_2$,$6 = 110_2$, $7 = 111_2$, $10 = 1010_2$, $26 = 11010_2$ и т. д.
2. Конкатенация: 100 11 110 111 1010 11010 и т. д. получаем последовательность 1001 1110 1111 0101 1010.
3. При переводе полученной последовательности получается `0x9ef5ac9457b2d29d951e7f74`.
   В условии сказано, что в качестве ответа необходимо отправлять решение, состоящее из последовательности цифр 0 и 9 и строчных букв от a до f, что однозначно описывает формат ответа

Ответ: `9ef5ac9457b2d29d951e7f74`.

Задача VI.1.1.2. Поиск кратчайшего пути (100 баллов)

Условие

Пусть имеется граф из 26 вершин, промаркированных заглавными буквами латинского алфавита от A до Z (см. рисунок).



Каждому ребру такого графа приписан некоторый целочисленный вес, обозначающий время, за которое его можно пройти.

Требуется определить кратчайший маршрут, ведущий из вершины A в вершину Z и проходящий через как можно большее число вершин.

В качестве ответа требуется вывести строку, составленную из маркеров вершин, через которые проходит такой маршрут, в порядке их обхода (например, AZ).

Решение

Задачу можно решить вручную, перерисовав граф на листочек, и посчитать вручную или написать программу алгоритмом Дейкстры. В качестве решения нужно будет отправлять полученный кратчайший путь.

Ответ: $ALEGOPTUNZ$.

Задача VI.1.1.3. Космический побег (100 баллов)

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 2 с.

Ограничение по памяти: 256 Мбайт.

Условие

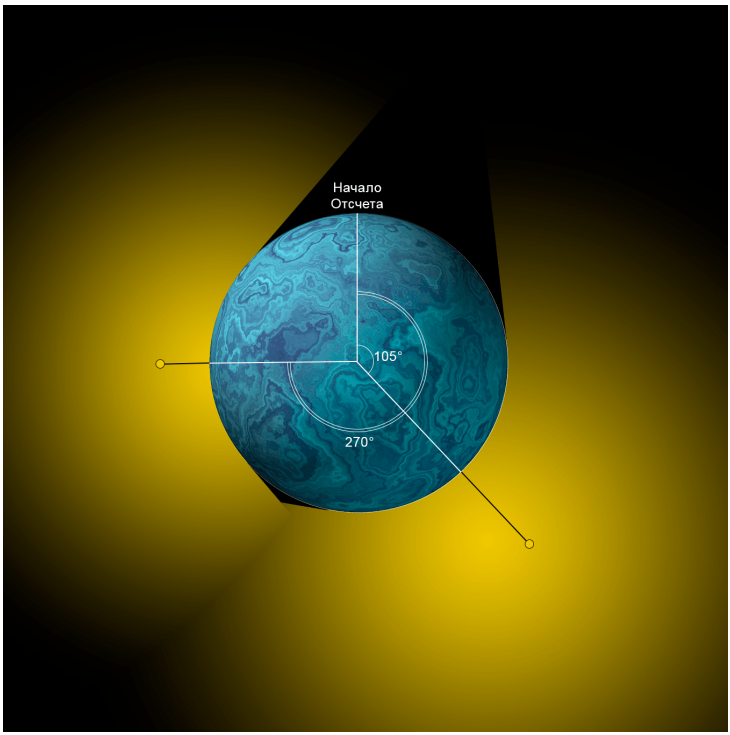
Инопланетянин по имени Скр живет в плоском мире. Скр снова не доел мамин суп, поэтому теперь он в бегах... Летя на своем корабле он увидел перед собой ма-

ленькую планету — круг радиуса r. На планете нет ничего, кроме ярких фонарей. Эти фонари настолько яркие, что освещают на сколько угодно большое расстояние, однако сквозь поверхность планеты они просветить не могут. Всего n фонарей, i-ый фонарь представляет собой тонкий столб высоты h_i, на верхушке которого располагается светящийся элемент, освещающий все в зоне своей видимости. Укажем точку отчета — произвольную точку на поверхности планеты. И для каждого фонаря известно вещественное число a_i — угол в радианах по часовой стрелки от точки отсчета до основания фонаря (см. иллюстрацию).

Скр пытается уйти от погони, поэтому хочет спрятаться в тени, то есть в такой точки планеты, которую не освещает ни один фонарь.

Инопланетянин очень не хочет попасться маме, поэтому просит вас о помощи. Он хочет узнать, есть ли на этой планете место, в котором мог бы спрятаться.

Фонари располагаются перпендикулярно поверхности планеты.



Формат входных данных

В первой строке входных данных располагается целое число n — количество фонарей на планете.

Во второй строке одно вещественное число r — радиус планеты.

В следующих n строках располагаются по два вещественных числа — a_i и h_i — угол в радианах от начальной точки до основания фонаря и высота фонаря.

Формат выходных данных

Выведите в единственной строке `YES`, если существует место, в котором Скр может спрятаться, и `NO` в ином случае.

Ограничения

$$1 \leqslant n \leqslant 10^5.$$

$$1 \leqslant r \leqslant 100.$$

$$1 \leqslant a_i \leqslant 2 \cdot \pi.$$

$$0 < h_i < 100.$$

Система оценки и описание подзадач

Баллы за подзадачи начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	**Баллы**	**Дополнительные ограничения**	**Необходимые подзадачи**	**Информация о проверке**
		n		
1	15	$1 \leqslant n \leqslant 2$		полная
2	55	$1 \leqslant n \leqslant 10^3$	1	полная
3	30	Без ограничений	1, 2	полная

Примеры

Пример №1

Стандартный ввод
3 4 1.047197551 1.000000000 3.141592653 2.000000000 5.235987755 0.500000000

Стандартный вывод
YES

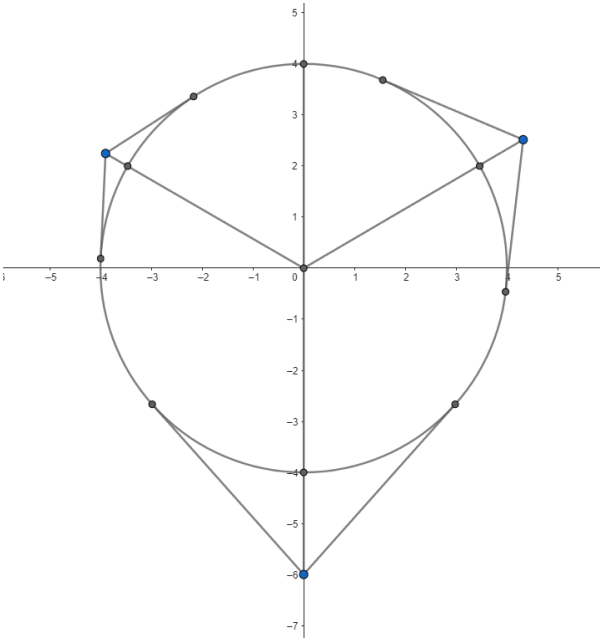
Пример №2

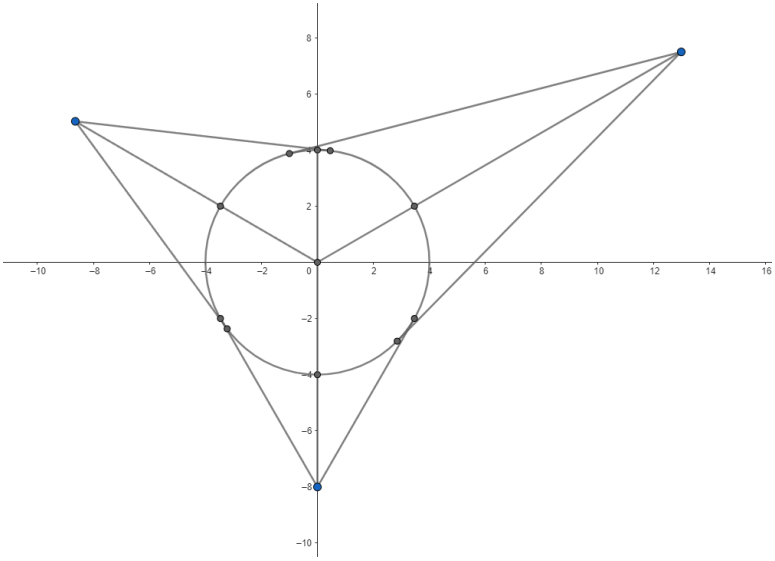
Стандартный ввод
3 4 1.047197551 11.000000000 3.141592653 4.000000000 5.235987755 6.000000000
Стандартный вывод
NO

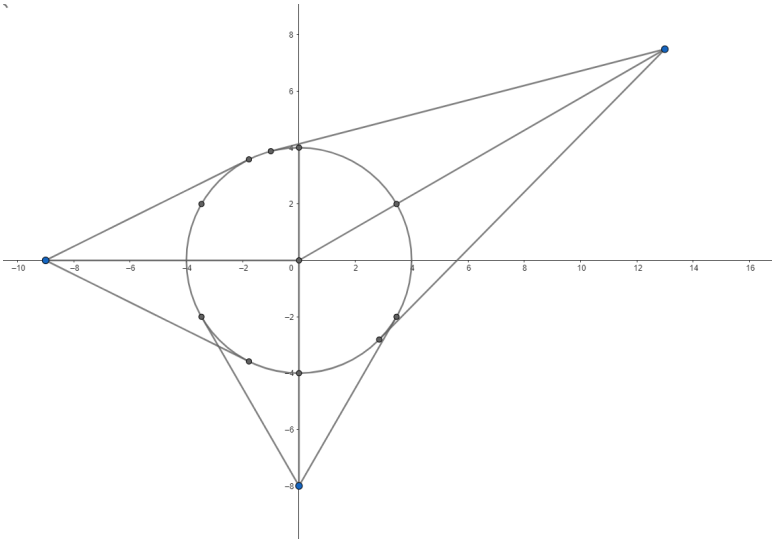
Пример №1

Стандартный ввод
3 4 1.047197551 11.000000000 3.141592653 4.000000000 4.712388980 5.000000000
Стандартный вывод
YES

Иллюстрации к примерам







В иллюстрациях начало отсчета располагается в верхней точке.

Решение

Задачу надо свести к задаче об пересечении отрезков (это известная задача, о ней можно почитать в открытых источниках).

Для каждого фонаря находим две точки — его точки касания, между которыми область, которую этот фонарь освещает. Ту точку, которая по углу ближе к начальной точке, нужно сделать точкой начала отрезка, а вторую — точкой его конца.

Сначала надо посчитать, сколько отрезков покрывает начальную точку. Далее все точки отсортировать по углу до начальной точки и обойти их по часовой стрелке.

Когда встречаем точку начала отрезка, к счетчику прибавляем единицу, а когда встречаем конец, отнимаем единицу. Если в какой-то момент пересекает на счетчике стоит ноль, это означает, что мы нашли темное место.

Пример программы-решения

Ниже представлено решение на языке C++.

```
#include <iostream>
#include <vector>
#include <cmath>
#include <fstream>
#include "algorithm"

using namespace std;

const double Pi = 3.141592653589793;
int main() {
    cin.tie(0);
    cout.tie(0);
    cin.sync_with_stdio(0);
    cout.sync_with_stdio(0);
    int n;
    double r;
    cin >> n >> r;
    vector<pair<double, bool>> a;
    double fi, h;
    int start_ans = 0;
    for (int i = 0; i < n; ++i) {
        cin >> fi >> h;
        double alpha = acos(r / (r + h));
        double fi1 = fi - alpha, fi2 = fi + alpha;
        if (fi1 < 0) {
            start_ans += 1;
            fi1 += 2 * Pi;
        }
        if (fi2 > 2 * Pi) {
            start_ans += 1;
            fi2 -= 2 * Pi;
        }
        a.emplace_back(fi1, true);
        a.emplace_back(fi2, false);
    }
    std::sort(a.begin(), a.end());
    bool ans = start_ans == 0;
    for (auto i : a) {
        if (i.second) {
            ++start_ans;
        } else {
            --start_ans;
        }
```

```
44          if (!start_ans) {
45              ans = true;
46          }
47      }
48      if (ans) {
49          cout << "YES\n";
50      } else {
51          cout << "NO\n";
52      }
53  }
```

Задача VI.1.1.4. Саша и кофе (100 баллов)

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 1,5 с.

Ограничение по памяти: 256 Мбайт.

Условие

Программист Саша пьет кофе всегда из одной и той же кружки, которую он никогда не моет.

Когда кофе находится в кружке достаточно долго, он оставляет на ней кофейное кольцо.

Иногда Саша подливает кофе в кружку, от чего уровень кофе поднимается, и все кольца, которые оказываются ниже уровня кофе, растворяются.

Саша может выполнить 3 действия:

1. Налить в кружку $coffee_i$ мл кофе.
2. Выпить $coffee_i$ мл кофе.
3. Определить, сколько кофейных колец сейчас на кружке.

Считаем, что после каждого действия Саши кофе в кружке стоит достаточно долго, чтобы кольцо успело образоваться.

Формат входных данных

Первая строка входных данных содержит число n — количество действий, которые сделал Саша.

В следующих n строках идет:

- $+\ coffee_i$ — Саша доливает в кружку $coffee_i$ мл кофе.
- $-\ coffee_i$ — Саша выливает из кружки $coffee_i$ мл кофе.

Гарантируется, что Саша не пытается выпить больше кофе, чем есть в кружке. Изначально в кружке нет ни 1 мл кофе.

Формат выходных данных

Требуется вывести количество кофейных колец, которое было на кружке после каждого действия Саши. Каждое в новой строке.

Ограничения

$$0 < n \leqslant 10^7.$$

$$0 < c_i \leqslant 10^8.$$

Гарантируется, что объем кофе в стакане никогда не достигнет 2^{32}.

Примеры

Пример №1

Стандартный ввод
14 + 108 + 105 + 110 + 101 - 32 + 32 + 32 + 32 + 32 - 103 - 114 - 101 - 101 - 100

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

Пример №2

Стандартный ввод
9 + 1 - 1 + 1 + 1 + 2 + 3 - 4 + 1 + 7

Стандартный вывод
1 1 1 1 1 1 2 2 1

Решение

Так как команды подаются раздельно друг от друга, то алгоритм решения не может быть быстрее, чем $O(n)$.

Кольцо может образоваться только при изменении уровня кофе в кружке, и притом только 1 кольцо.

Давайте хранить кофейные кольца упорядочено по уровню.

Тогда, когда Саша доливает кофе, нужно лишь перебрать кольца с наименьшим уровнем и стереть их.

А когда Саша выпивает кофе, мы добавляем новое кольцо.

Однако нужно отдельно обработать момент, когда кофе закончился, так как тогда кольцо не образуется. И если Саша ничего не долил или ничего не выпил, тогда новое кольцо тоже не может появится.

Описание реализации.

Нам нужно упорядоченно хранить кольца, для этого могут подойти: *queue*, *set*, *stack*. Но так как кольцо появляются последовательно, и новое кольцо всегда меньше, чем предыдущее, нам хватит и *stack*.

$rings$ — стэк, хранящий все кофейные кольца, которые есть сейчас. Кольцо определяется уровнем кофе, на котором оно появилось.

На дне *stack* будут кольца с большим уровнем, а наверху — с наименьшим.

Когда приходит команда на изменение объема кофе, мы изменяем переменную с текущим уровнем кофе, назовем ее $coffee$.

Если мы долили кофе, то нужно выкинуть сверху *rigns* все кольца с объемом, меньшим или равным текущему уровню. А в конце добавить кольцо текущего уровня.

Если же Саша выпил кофе, то добавляем новое кольцо.

Отдельно рассмотреть ситуации, про которые говорилось ранее.

Чтобы посчитать количество колец, нам нужно лишь вернуть размер *rings*, это работает за $O(1)$.

Пример программы-решения

Ниже представлено решение на языке C++.

```
#include <iostream>
#include <stack>

int main() {
    std::cin.tie(0);
    std::cout.tie(0);
    std::ios_base::sync_with_stdio(0);
    int n;
    std::cin >> n;

    char command;
    unsigned int now = 0, temp;
    std::stack<unsigned int> rings;
    for (int _ = 0; _ < n; ++_) {
        std::cin >> command >> temp;

        if (!temp) {
            continue;
        }

        if (command == '+') {
            now += temp;
            while (!rings.empty() && rings.top() <= now) {
                rings.pop();
            }
            rings.emplace(now);
        } else {
            now -= temp;
            if (now) {
                rings.emplace(now);
            }
        }
        std::cout << rings.size() << '\n';
    }
}
```

Задача VI.1.1.5. Стильная гирлянда (100 баллов)

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 3 с.

Ограничение по памяти: 256 Мбайт.

Условие

В преддверии нового года Дед Мороз наколдовал гирлянду, которая представляет собой последовательность разноцветных лампочек.

Однако полученная гирлянда не соответствует стилю 2024 года, из-за чего Дед Мороз поручил своим эльфам вырвать из гирлянды наименьшее количество лампочек так, чтобы полученная гирлянда соответствовала стилю 2024 года.

Изначальная гирлянда задается строкой s. Гирлянда является стильной, если она состоит из последовательно соединенных строк k.

Более формально: если строка $k = k_1, k_2, \ldots, k_n$, то стильная гирлянда будет представлять собой последовательность следующего вида:

$$k_{1_1}, k_{2_1} \ldots k_{n_1}, k_{1_2} \ldots k_{n_2}, \ldots k_{n_m}.$$

Где $k_{i_j} == k_i$.

Формат входных данных

Первая строка входных данных — s.

Вторая строка — k.

Гарантируется, что обе строки состоят только из латинских букв (строчных и прописных).

Формат выходных данных

Выходные данные должны содержать одно целое число, максимальная длина стильной гирлянды, которая может получиться выдергиванием лампочек из изначальной гирлянды.

Ограничения

$$0 < length(s) \leqslant 10^7.$$

Примеры

Пример №1

Стандартный ввод
acbacabba abac

Стандартный вывод
4

Пример №2

Стандартный ввод
abac acbacabba
Стандартный вывод
0

Решение

Во-первых нужно понять, что если уже есть какая-то красивая гирлянда и вы пытаетесь удлинить ее, то всегда нужно брать ближайшую подходящую лампочку.

Брать более далекую лампочку бессмыслено, так как из-за этого мы не получим гирлянду короче, только длиннее или такую же.

А значит можно последовательно пробежаться по исходной гирлянде, если встречается подходящая лампочка, то ее берем, если нет — идем дальше.

В ответ выводим длину строки из полных строк k.

Описание реализации:

Пусть *count* длина стильной гирлянды, которую мы уже собрали.

Циклом проходимся по строке s, если текущий символ совпадает с текущим символом строки k, то увеличиваем *count* на 1 и берем следующий символ строки k.

В конце выводим *count*.

Пример программы-решения

Ниже представлено решение на языке C++.

```
#include <iostream>
#include <string>

int main() {
    std::cin.tie(nullptr);
    std::cout.tie(nullptr);
    std::ios_base::sync_with_stdio(false);
    std::string lights, beautiful;
    std::cin >> lights >> beautiful;

    int count = 0;
    for (char& el : lights) {
        if (el == beautiful[count % beautiful.size()]) {
            ++count;
        }
    }
    std::cout << count - (count % beautiful.size());
}
```

Математика. 8–9 классы

Задача VI.1.2.1. (15 баллов)

Темы: числовые множества, оценка + пример, комбинаторика.

Условие

Пусть A — такое подмножество $\{1, 2, ..., 9\}$, что все суммы, образованные сложением двух чисел из A, различны. Например, подмножество $\{1, 2, 3, 4, 5\}$ не такое, поскольку пары $1, 4$ и $2, 3$ имеют одинаковую сумму. Какое максимальное количество элементов может содержать A?

Решение

Можно проверить, что все суммы пар множества $\{1, 2, 3, 5, 8\}$ различны. Предположим, что A — подмножество $\{1, ..., 9\}$, содержащее 6 элементов и обладающее нужным свойством. Наименьшая возможная сумма двух чисел из A равна $1 + 2 = 3$, а наибольшая возможная сумма $8 + 9 = 17$. Это дает 15 возможных сумм: 3,..., 17. Заметим, что $C_6^2 = 15$, и поэтому каждое из чисел 3,..., 17 есть сумма ровно одной пары из A. Единственная пара из $\{1, 2, ..., 9\}$, которая дает 3, — это $\{1, 2\}$, а 17 — $\{8, 9\}$. Таким образом, 1, 2, 8, 9 принадлежат A. Но тогда $1+9 = 2+8$, что дает противоречие. Следовательно, максимальное число элементов, которые может содержать A, равно 5.

Критерии оценивания

- Найдено нужное подмножество из 5 элементов — 3 балла.
- Доказано, что каждое из чисел 3,...,17 есть сумма ровно одной пары из A — 10 баллов.

Ответ: 5.

Задача VI.1.2.2. (20 баллов)

Темы: текстовая задача, целочисленные уравнения.

Условие

На точных двенадцатичасовых часах часовая и минутная стрелки движутся непрерывно. Ровно через m мин ($1 \leqslant m \leqslant 720$, m — целое) после 12:00 угол между часовой и минутной стрелками равен ровно 1°. Чему равно m?

Решение

Минутная стрелка совершает полный оборот на 360 градусов каждые 60 минут, поэтому через m мин она поворачивается на $m(360/60) = 6m^\circ$. Часовая стрелка

делает полный оборот каждые 12 ч (720 мин), значит, через m мин она повернется на $m(360/720) = m/2°$. Поскольку в 12:00 обе стрелки находились в одном и том же положении, угол между ними равен 1°, если $6m - m/2 = \pm 1 + 360k$ для некоторого целого числа k. Тогда

$$m = \frac{720k \pm 2}{11} = 65k + \frac{5k \pm 2}{11}.$$

Поскольку $1 \leqslant m \leqslant 720$, то $1 \leqslant k \leqslant 11$ и $5k \pm 2$ должно быть кратным 11, $5k \pm 2 = 11q$.

Значит,

$$k = 2q + \frac{q \pm 2}{5}.$$

Теперь ясно, что только $q = 2$ и $q = 3$ удовлетворяют всем условиям.

Таким образом, $k = 4$ или $k = 7$ и возможные значения $m = 262$ и 458.

Критерии оценивания

Составлено верное уравнение для m — 3 балла.

Ответ: 262 или 458.

Задача VI.1.2.3. (20 баллов)

Темы: планиметрия, геометрическая вероятность.

Условие

В выпуклом четыреугольнике $ABCD$ наибольшая сторона AB. На сторонах AB и BC выбраны точки M и N так, что каждый из отрезков AN и CM делит четырехугольник на две части равной площади. Через точку D проведена прямая параллельная AC, которая пересекается с прямыми AB и BC в точках P и Q соответственно. Найдите вероятность того, что точка, случайным образом брошенная в тругольник BPQ, попадет в треугольник BMN.

Решение

Из равенств $S_{MADC} = \frac{1}{2}S_{ABCD} = S_{NADC}$ следует, что $S_{ANC} = S_{AMC}$, поэтому MN параллельна AC.

Тогда

$$S_{MPC} = S_{MAC} + S_{CAP} = S_{MAC} + S_{CAD} = S_{MADC} = S_{BMC}.$$

Следовательно, $BM = MP$. Аналогично $BN = NQ$ и тогда MN есть средняя линия треугольника BPQ. Поэтому площадь треугольника BMN в четыре раза меньше площади тругольника BPQ.

Критерии оценивания

- Доказано, что MN параллельна AC — 5 баллов.
- Доказано, что $S_{MPC} = S_{BMC}$ — 10 баллов.

Ответ: 0,25.

Задача VI.1.2.4. (20 баллов)

Темы: числовые последовательности, функциональные уравнения.

Условие

Нейросеть составила последовательность натуральных чисел $a_1, a_2, a_3, \ldots$ в соответствии с правилом: для любых $n, m \in \mathbb{N}$

$$a_n \cdot a_{m+1} = n(a_m + 1).$$

Чему равен 2024-й член последовательности и что это за последовательность?

Решение

Положим $m = 1$.

Тогда $a_n = xn$, где $x = (1 + a_1)/a_2$.

Далее, полагая $n = 1$ и учитывая, что $a_1 = x, a_m = xm, a_{m+1} = x(m+1)$, получаем

$$x^2(m + 1) = xm + 1 = x(m + 1) - (x - 1) \Leftrightarrow (x(m + 1) + 1)(x - 1) = 0.$$

Поэтому $x = 1$, $a_n = n$.

Критерии оценивания

Найдено, что a_n пропорционально n — 5 баллов.

Ответ: $a_{2024} = 2024$, последовательность — натуральный ряд.

Задача VI.1.2.5. (25 баллов)

Темы: целые числа, принцип Дирихле.

Условие

Пусть $M = \{a_1, a_2, \ldots, a_{2024}\}$ — набор целых чисел (числа могут повторяться) из промежутка $[-1012, 1012]$, сумма которых равна 1.

Докажите, что всегда можно выбрать числа из M, сумма которых равна 0, либо среди этих чисел есть 0.

Решение

Пусть все числа из M ненулевые. Составим из M новую последовательность чисел $S = \{b_1, b_2, \ldots, b_{2024}\}$ следующим образом.

Пусть $b_1 > 0, b_2 < 0$. Такие всегда будут в M, т. к. сумма всех чисел равна 1.

Далее, для каждого $i = 2, 3, ..., 2024$ выбираем число b_i из M, знак которого противоположен знаку величины

$$s_{i-1} = b_1 + b_2 + \ldots + b_{i-1}.$$

Заметим, что мы можем считать, что $s_{i-1} \neq 0$, поскольку иначе все доказано. Кроме того, на каждом шаге нужный выбор b_i гарантирован, т. к. условие $a_1 + a_2 + + \ldots + a_{2024} = 1$ означает, что сумма чисел, не входящих в сумму s_{i-1}, либо равна 0, либо имеет знак, противоположный s_{i-1}.

Таким образом, получили набор ненулевых целых чисел $s_1, s_2, \ldots, s_{2024}$, причем по построению $s_i = s_{i-1} + b_i$, $s_1 = b_1 \in (0, 1012]$ и знаки s_{i-1}, b_i разные.

Поэтому $s_i \in [-1011, 1012]$.

На указанном интервале 2023 ненулевых целых и, следовательно, $s_j = s_k$ для некоторых $1 \leqslant j < k \leqslant 2024$. Тогда $b_{j+1} + b_{j+2} + \ldots + b_k = 0$, что и требовалось доказать.

Критерии оценивания

- Предложен метод упорядочить числа из M — 5 баллов.
- Предложен способ составления частичных сумм — 15 баллов.

Математика. 10–11 классы

Задача VI.1.3.1. (15 баллов)

Темы: текстовые задачи, уравнения в целых числах.

Условие

У профессора математики время на проведение экзамена зависит только от количества выставленных двоек и обратно пропорционально их числу. На первый экзамен ушло на 30 мин меньше времени, чем на второй. А если бы на первом экзамене он поставил на 5 двоек больше, то он затратил бы на первый экзамен на 2 ч 30 мин меньше времени, чем на второй. Сколько двоек он поставил на каждом экзамене?

Решение

Пусть x — количество двоек на первом экзамене, y — на втором.

Тогда

$$\frac{k}{x} = \frac{k}{y} - \frac{1}{2}, \ \frac{k}{x+5} = \frac{k}{y} - \frac{5}{2}.$$

Исключим k и выразим y через x,

$$y = \frac{4x(x+5)}{4x+25} = x - \frac{5x}{4x+25}.$$

Целочисленность x, y дает решение $x = 25$, $y = 24$.

Критерии оценивания

Составлено уравнение для числа двоек на экзаменах — 5 баллов.

Ответ: 25 на первом экзамене, 24 на втором экзамене.

Задача VI.1.3.2. (20 баллов)

Темы: планиметрия, выпуклые множества.

Условие

Робот Саша научился работать с точками на плоскости. Для набора из любых пяти точек $A = \{A_1, A_2, A_3, A_4, A_5\}$, никакие три из которых не лежат на одной прямой, он может вычислить величину $f(A)$, равную минимальному значению среди углов $\angle A_i A_j A_k$, где i, j, k различные целые числа от 1 до 5.

Какое максимальное значение $f(A)$ может вычислить Саша?

Решение

Заметим, что если данные пяти точек образуют правильный пятиугольник, то минимум углов, образуемых любой тройкой из пяти вершин, равен 36°. Следовательно, $\max f(A) \geqslant 36$.

Покажем, что для любого набора $A = \{A_1, A_2, A_3, A_4, A_5\}$ из пяти точек, удовлетворяющего условию задачи, должен существовать угол, меньший или равный 36°, образованный тремя точками из набора. Рассмотрим наименьшее выпуклое множество, назовем его Ω, на плоскости, содержащее $A = \{A_1, A_2, A_3, A_4, A_5\}$. Поскольку это выпуклое множество Ω должно быть либо треугольником, либо четырехугольником или пятиугольником, то этот многоугольник должен иметь внутренний угол 108° или меньше.

Пусть, без ограничения общности, этот угол $\angle A_1 A_2 A_3$. Из построения Ω следует, что оставшиеся 2 точки A_4, A_5 лежат внутри угловой области $\angle A_1 A_2 A_3$ и, следовательно, должен быть угол меньше или равный $108/3 = 36$, который образован тройкой, выбранной из данных 5 точек.

Критерии оценивания

- Рассмотрен пятиугольник — 5 баллов.
- Построена выпуклая оболочка — 10 баллов.

Ответ: 36.

Задача VI.1.3.3. (20 баллов)

Темы: многочлены, разложение на множители.

Условие

Многочлен

$$P(x) = x^4 - 2x^3 + \frac{3}{2}x^2 + ax + b$$

разложили на множители $P(x) = (x - x_1)(x - x_2)(x - x_3)(x - x_4)$ и оказалось, что все x_k, $k = 1, 2, 3, 4$ положительны.

Какие значения может принимать параметр b?

Решение

В равенстве

$$x^4 - 2x^3 + \frac{3}{2}x^2 + ax + b = (x - x_1)(x - x_2)(x - x_3)(x - x_4)$$

раскроем скобки в правой части и приравняем коэффициенты при x^3 и x^2.

Тогда

$$x_1 + x_2 + x_3 + x_4 = 2, \quad \sum_{1 \leqslant k < j \leqslant 4} x_k x_j = \frac{3}{2}.$$

Из тождества

$$\left(\sum_{k=1}^{4} x_k\right)^2 = \sum_{k=1}^{4} x_k^2 + 2 \sum_{1 \leqslant k < j \leqslant 4} x_k x_j$$

следует, что $\sum_{k=1}^{4} x_k^2 = 1$.

Далее заметим, что

$$\sum_{k=1}^{4} (2x_k - 1)^2 = 4 \sum_{k=1}^{4} x_k^2 - 4 \sum_{k=1}^{4} x_k + 4 = 0.$$

Поэтому $x_1 = x_2 = x_3 = x_4 = 1/2$.

Заметим, что $b = P(0) = x_1 x_2 x_3 x_4 = \frac{1}{16}$.

Критерии оценивания

- Найдено, что $= \sum_{k=1}^{4} x_k = 2$, $\sum_{1 \leqslant k < j \leqslant 4} x_k x_j = 3/2$ — 5 баллов.
- Найдено, что $\sum_{k=1}^{4} x_k^2 = 1$ — 10 баллов.

Ответ: $\frac{1}{16}$.

Задача VI.1.3.4. (20 баллов)

Темы: планиметрия, площадь треугольника.

Условие

В треугольнике ABC выбрали точку P и обозначили через D, E, F точку пересечения прямой AP со стороной BC, прямой BP со стороной CA и прямой CP со стороной AB соответственно. Площади треугольников PFA, PDB и PEC равны 1.

Найти площадь треугольника ABC.

Решение

Будем обозначать площадь треугольника XYZ через $[XYZ]$.

Пусть $x = [PAB]$, $y = [PBC]$, $z = [PCA]$.

Треугольники PBC и PCA имеют общую сторону и значит, отношение их площадей равно отношению высот, опущенных из точек B и A на прямую FC, которое равно, в свою очередь, $BF : AF$,

$$\frac{y}{z} = \frac{BF}{AF} = \frac{[BPF]}{[APF]} = \frac{x-1}{1}.$$

Следовательно,

$$y = z(x-1), \quad zx = y + z, \quad x(z+1) = x + y + z.$$

Рассуждая аналогично, получим

$$y(x+1) = x + y + z, \quad z(y+1) = x + y + z. \tag{VI.1.1}$$

Тогда

$$x(z+1) = y(x+1) = z(y+1). \tag{VI.1.2}$$

Далее, не нарушая общности, будем считать, что $x \leqslant y$, $x \leqslant z$.

Докажем, что $y = z$.

Действительно, если $y > z$, то

$$(y+1)z > (z+1)z \geqslant (z+1)x,$$

что противоречит (VI.1.2).

Если $y < z$, то

$$(x+1)y < xy + z \leqslant zy + z = z(y+1),$$

что также противоречит (VI.1.2).

Поэтому $y = z$ и тогда из (VI.1.2) следует, что $x = y = z$, а из (VI.1.1) получаем $x = y = z = 2$.

Площадь треугольника ABC равна $x + y + z = 6$.

Критерии оценивания

- Найдено, что $\frac{y}{z} = \frac{BF}{AF}$ — 5 баллов.
- Найдено, что $\frac{y}{z} = \frac{x-1}{1}$ — 10 баллов.

Ответ: 6.

Задача VI.1.3.5. (25 баллов)

Темы: оценка + пример.

Условие

Найти максимальное значение суммы квадратов $x_1^2 + x_2^2 + \ldots + x_{2024}^2$, если

$$x_1 \geqslant x_2 \geqslant \ldots \geqslant x_{2024} \geqslant 0, \quad x_1 + x_2 \leqslant 2024, \quad x_3 + x_4 + \ldots + x_{2024} \leqslant 2024.$$

Решение

Пусть $n = 2024$.

Складывая два последних неравенства в условии, получаем

$$x_1 + x_2 + \ldots + x_n \leqslant 2n.$$

Оценим сумму квадратов сверху.

$$x_1^2 + x_2^2 + \ldots + x_n^2 \leqslant (n - x_2)^2 + x_2^2 + \ldots + x_n^2 = n^2 - 2nx_2 + 2x_2^2 + x_3^2 + \ldots + x_n^2 \leqslant$$

$$\leqslant n^2 - (x_1 + x_2 + \ldots + x_n)x_2 + 2x_2^2 + x_3^2 + \ldots + x_n^2 =$$

$$= n^2 + (x_2^2 - x_1x_2) + (x_3^2 - x_3x_2) + \ldots + (x_n^2 - x_nx_2).$$

Каждое слагаемое в скобках неположительно и поэтому

$$x_1^2 + x_2^2 + \ldots + x_n^2 \leqslant n^2.$$

Максимальное значение суммы квадратов не может быть больше n^2.

Нетрудно видеть, что указанная величина достигается, если $x_1 = n$, $x_2 = 0$, $x_3 = 0$, …, $x_n = 0$.

Критерии оценивания

- Найдено, что $x_1 + x_2 + \ldots + x_n \leqslant 2n$ — 3 балла.
- Найден, пример $x_1 = n, x_2 = 0, x_3 = 0, \ldots, x_n = 0$ — 1 балл.

Ответ: 2024^2.

Инженерный тур

Общая информация

Участники разрабатывают космический VR-симулятор, на котором реализуется сценарий совершения мягкой посадки на спутник Сатурна — «Титан», а также соответствующие механики для маневрирования космическим кораблем во время посадки.

Легенда задачи

В 2022 году участники финала Олимпиады НТО разработали VR-симулятор космического корабля, на котором игроку необходимо было преодолеть пояс астероидов.

В этом году участникам необходимо разработать продолжение VR-симулятора космического корабля: этап совершения посадки в исследовательских целях на самый большой спутник Сатурна — «Титан», обладающий атмосферой. Основная цель игрока — совершить мягкую посадку для сохранения возможности дальнейшего функционирования космического корабля.

Требования к команде и компетенциям участников

Количество участников в команде: 3–4 человека.

Компетенции, которыми должны обладать члены команды:

- тимлид;
- программист;
- 3D-художник;
- UX/UI-художник;
- геймдизайнер.

Участник может совмещать несколько ролей.

Оборудование и программное обеспечение

Рабочая станция:

- компьютер;
- монитор;
- клавиатура;
- мышь (следует заранее убедиться, что на столах можно работать с мышью без коврика).

Минимальные требования:

- ОС: Windows 7 SP1, Windows 8.1 (или выше), Windows 10.
- Процессор: Intel Core i5-4590/AMD FX 8350 (эквивалент или лучше).
- Оперативная память: ОЗУ 4 ГБ или больше.
- Видеокарта: NVIDIA GeForce GTX 970, AMD Radeon R9 290 (эквивалент или лучше).

Рекомендовано:

- ОС: Windows 10, 11.
- Процессор: Intel Core i5-4590/AMD FX 8350 (эквивалент или лучше).
- Оперативная память: ОЗУ 4 ГБ или больше.
- Видеокарта: NVIDIA GeForce GTX 1070/Quadro P5000 (эквивалент или лучше), AMD Radeon Vega 56 (эквивалент или лучше).

Обязательные требования:

- DisplayPort 1.2 или более новая модель (если шлем HTC Vive).
- Шлем HTC Vive Cosmos.

Наименование	**Описание**
Рабочая станция для VR (характеристики см. выше)	Для работы, отладки и тестирования ПО на целевом устройстве
Шлем HTC Vive Cosmos	Шлем виртуальной реальности — целевое устройство, на котором должно работать VR-приложение

Описание задачи. Симулятор космического корабля: Мягкая посадка

Задача состоит из пяти миссий:

1. Выведение космического корабля на орбиту.
2. Поворот.
3. Атмосфера.
4. Пилотирование в условиях шторма.
5. Раскрытие парашюта.

Миссии зависимы друг от друга, для реализации функционала каждой последующей миссии должны быть реализованы сценарии предыдущих миссий.

Подготовительный этап

Требуется сделать `sample` проект с VR и залить его на gitlab.

В качестве git-клиента можно использовать: https://git-scm.com/download/gui/windows.

Необходимо зарегистрироваться на gitlab. Для каждой команды создан приватный репозиторий в организации https://gitlab.com/nto-vr/2024.

- В каждый репозиторий добавить файл `.gitignore`:
  - Unity: https://github.com/github/gitignore/blob/main/Unity.gitignore;
  - UE: https://github.com/github/gitignore/blob/main/UnrealEngine.gitignore;
  - Godot: https://github.com/github/gitignore/blob/main/Godot.gitignore.
- Добавить в репозиторий `README.md` с названием шлема, на котором вы запускаете свой проект.
- Основная работа в ветке `main/master`, для сдачи работы необходимо будет делать. Ее будут проверять эксперты.

Общее описание

Предыстория

В 2022 году участники финала Олимпиады НТО разработали VR-симулятор космического корабля, на котором игрок преодолевал пояс астероидов. В текущем году участникам необходимо разработать продолжение VR-симулятора космического корабля: этап совершения посадки в исследовательских целях на самый большой спутник Сатурна — «Титан», обладающий атмосферой. Основная цель игрока — совершить мягкую посадку для сохранения возможности дальнейшего функционирования космического корабля.

Игровой процесс

Прохождение игры должно обеспечиваться посредством использования VR-гарнитуры и контроллеров. Перемещение должно осуществляться за счет физического перемещения в реальном мире и возможности телепортации по виртуальному космическому кораблю с использованием контроллеров. Игрок не должен падать сквозь стены и пол виртуального космического корабля.

Первоначально игрок находится у пульта управления космического корабля, наблюдая за его показателями. Космический корабль находится в космосе за гравитационным полем спутника «Титана». В окне иллюминатора можно наблюдать спутник, на который сейчас начнется посадка. Игра должна иметь условия для победы и поражения. Условие победы — совершение мягкой посадки. Условие поражения — критические повреждения космического корабля или игрока.

На проверку работ выделено 20 мин, поэтому рекомендуется разработать приложение таким образом, чтобы игровой процесс занимал не более 20 мин.

Требования к графике

1. В качестве космического корабля необходимо использовать 3D-модель, расположенную по ссылке: https://disk.yandex.ru/d/oxnMVbrnn_5Rhg/spaceship.zip.
2. Для визуализации требуется добавить `skybox`, ститистически отражающий космическое пространство вблизи Сатурна и его спутника «Титан».

3. Требуется самостоятельно разработать приспособления для совершения механик:
   3.1. Два контроллера для управления космическим кораблем (как показано на рис. VI.2.1). Левый и правый контроллеры должны быть визуально различимы. У каждого контроллера — подвижный стик (джойстик). Контроллеры должны позволять управлять (как показано на рис. VI.2.2) креном, рысканием и тангажом космического корабля (см. рис. VI.2.3). После нажатия на курок (кнопка триггера под указательным пальцем) на правом контроллере должно происходить включение двигателя, а уровень нажатия должен позволять регулировать тягу двигателя.
   3.2. Необходимо разработать кнопку с фиксацией режима управления (автопилот или ручное управление). Такой механизм должен позволять игроку удерживать кнопку в нажатом положении без постоянного удерживания ее рукой. Когда кнопка нажата и зафиксирована, должен быть включен режим автопилота до тех пор, пока игрок не нажмет на нее снова для переключения на ручное управление.
   3.3. Для выпуска парашютной системы требуется разработать «ручной трос», который должен крепиться к потолку. Для активации выпускания (выброса) парашюта игрок должен иметь возможность потянуть за «ручной трос».
   3.4. Требуется разработать отсек и капсулу для парашюта. Для того чтобы парашют можно было раскрыть, необходимо добавить возможность перемещения капсулы с парашютом в отсек для него.
   3.5. У игрока должна быть возможность посмотреть, успешно ли раскрылся парашют. Также необходимо добавить анимацию на парашют.
4. Следует визуализировать орбиту спутника.
5. Требуется визуализировать траекторию, которой следует придерживаться для совершения мягкой посадки.
6. Необходимо визуализировать разные слои атмосферы.
7. Требуется визуализировать шторм.

Рис. VI.2.1. Референс контроллера для управления космическим кораблем

Рис. VI.2.2. Один из вариантов управления космическим кораблем

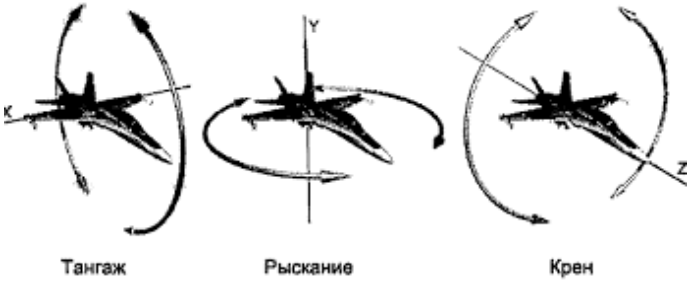


Рис. VI.2.3. Терминология

При разработке собственных 3D-моделей рекомендуется придерживаться стилистики космического корабля. При оценивании графики учитывается качество реализации и соответствие стилистике, заданной космическим кораблем.

Этапы проверки

Для защиты работоспособности проекта необходимо продемонстрировать выполнение миссий. На демонстрацию всех миссий командам доступно 20 мин.

С запуском игры игрок должен появляется в центре космического корабля, космический корабль находится за орбитой спутника «Титан», на который планируется совершить посадку.

1. Миссия: выведение космического аппарата на орбиту.
2. Миссия: поворот космического аппарата.
3. Миссия: погружение в атмосферу.
4. Миссия: пилотирование в условиях шторма.
5. Миссия: раскрытие парашюта и посадка.

В каждом блоке требований начисляются дополнительные баллы за самостоятельную реализацию, которую участник должен быть способен обосновать. За реализацию неуказанного в ТЗ функционала могут начисляться дополнительные баллы.

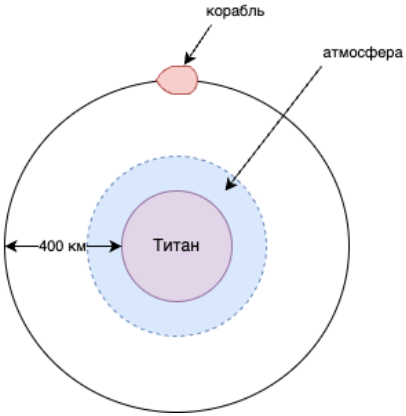
Должна быть возможность начать выполнять с каждую миссию самостоятельно.

Миссия 1: Выведение космического корабля на орбиту

Краткое описание

В начале игрового процесса космический корабль находится за орбитой спутника, на который планируется совершить посадку. Прежде чем совершать мягкую посадку космического корабля, игроку необходимо вывести космический корабль на орбиту спутника.

Орбита спутника должна находиться на высоте $h = 400$ км, атмосфера для физических расчетов начинается с высоты 140 км. Чтобы космический корабль оставался на орбите, ему нужно развить первую космическую скорость. Первая космическая скорость, которой необходимо достигнуть космическому кораблю, чтобы оставаться на орбите равна $1,73 \cdot 10^3$ м/с.



Навигационная панель

Игроку должна быть доступна навигационная панель, на которой должна быть отражена/визуализирована следующая информация:

- Орбита спутника, на которую следует вывести космический аппарат (должна быть видна через окна космического корабля и на навигационной панели).
- Траектория курса, которого следует придерживаться для совершения посадки.
- Расстояние в км до поверхности спутника.
- Скорость космического аппарата в км/ч.
- Текущее время и коэффициент ускорения времени при необходимости (x_2, $x_{1,5}$).

Впоследствии в задании указаны описание сил. В случае, если игрок сходит с курса, система должна сигнализировать о рисках (показатели скорости выше нормы в км/ч или в м/с).

Панель управления

Для корректировки курса игроку должна быть доступна панель управления космическим кораблем. Требуется спроектировать удобное управление космическим аппаратом с использованием контроллеров (см. рис. VI.2.2).

Для изменения режима управления игроку должна быть доступна кнопка с фиксацией состояния:

- режим автопилота;
- режим ручного управления (специальный контроллер, разработанный по референсу, см. рис. VI.2.1).

Уточнение

После выведения космического корабля на орбиту следует развернуть космический корабль, чтобы в следующей миссии двигаться вперед соплом космического корабля.

Миссия 2: Поворот

Краткое описание

После выведения космического аппарата на орбиту космический корабль должен двигаться по орбите вперед соплом двигателя.

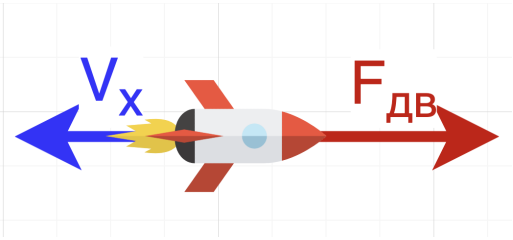


Рис. VI.2.4. Направление скорости движения по орбите и силы двигателя

Игрок должен постепенно совершать поворот космического аппарата, как показано на схеме посадки космического корабля.

Масса космического аппарата с учетом топлива 1200 т. Космический корабль должен обладать первой космической скоростью. Как только игрок готов совершать поворот космического аппарата, ему следует включить двигатель, что позволит притормозить движение космического корабля по орбите V_x, так как на корабль начнет действовать сила двигателя $F_{\text{дв}}$, направленная в противоположную сторону. Постепенно на космический корабль должна начать влиять сила тяжести $F_{\text{тяж}}$, которая всегда направлена вниз. Требуется смоделировать поворот космического корабля с учетом сил $F_{\text{тяж}}$, $F_{\text{дв}}$, V_i.

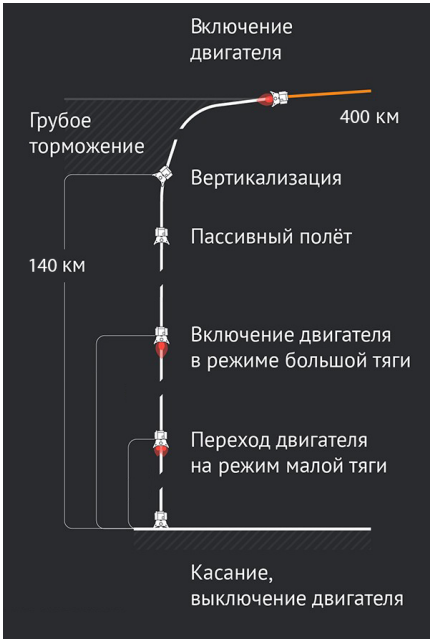


Рис. VI.2.5. Схема посадки космического корабля

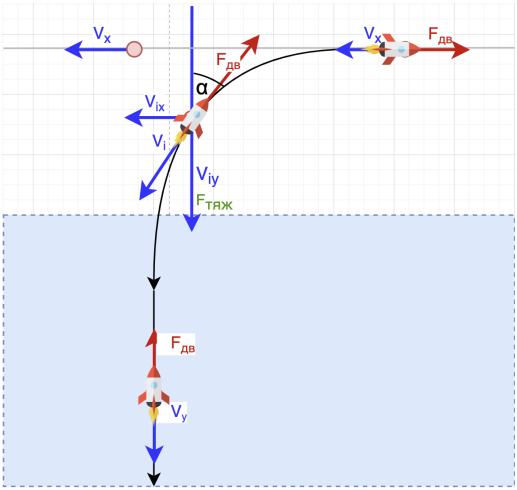


Рис. VI.2.6. Схема поворота и посадки космического корабля с указанием направлений сил

V_i — скорость космического аппарата.

Пусть $\Delta t = 10^{-3}$ с.

При t_0: $V_{0x} = V_{1k}$, $V_{0y} = 0$ корабль находится в горизонтальном положении.

При t_n: $V_{nx} = 0$, $V_{ny} = V_n$ корабль находится в вертикальном положении.

$F_{\text{тяж}} = mg$ — сила тяжести, где m — масса, g — ускорение свободного падения;

$F_{\text{дв}} = ma_{\text{дв}}$ — сила космического корабля, где m — масса, $a_{\text{дв}}$ —ускорение двигателя;

$a'_{\text{дв}} = a_{\text{дв}}\Delta t$ — изменение скорости от работы двигателя за Δt;

$g = \dfrac{GM}{(R+h)^2}$ — зависимость g от высоты h;

$g' = g\Delta t$ — изменение скорости от работы двигателя за Δt;

$a = \text{arctg}\left(\dfrac{V_{ix}}{V_{iy}}\right)$ — отклонение от оси OY;

$V_i = \sqrt{V_{ix}^2 + V_{iy}^2}$ — зависимость скорости космического корабля от проекций;

$t = 0 \to$ вкл. двиг. 1 мс > расчет g_1, V_{1x}, V_{1y}, a_1 1 мс > расчет g_2, V_{2x}, V_{2y}, a_2 ... мс > ... ... мс > расчет $g_{(n-1)}, V_{(n-1)x}, V_{(n-1)y}, a_{(n-1)}$ 1 мс > расчет g_n, V_{nx}, V_{ny}, a_n — логика расчета.

t, мс	i	V_{ix}	V_{iy}
0	0	V_{1x}	0
1	1	$V_{0x} - a'_{\text{дв}}\sin(\alpha)$	$V_{0y} + g'_1 - a'_{\text{дв}}\cos(\alpha)$
2	2	$V_{1x} - a'_{\text{дв}}\sin(\alpha)$	$V_{1y} + g'_2 - a'_{\text{дв}}\cos(\alpha)$
.	.	.	.
.	.	.	.
$n-1$	$n-1$	$V_{(n-2)x} - a'_{\text{дв}}\sin(\alpha)$	$V_{(n-2)y} + g'_{(n-1)} - a'_{\text{дв}}\cos(\alpha)$
n	n	$V_{(n-1)x} - a'_{\text{дв}}\sin(\alpha)$	$V_{(n-1)y} + g'_n - a'_{\text{дв}}\cos(\alpha)$

При $t = n$ корабль должен стать полностью вертикально.

Поворот с учетом атмосферы (дополнительные баллы)

В качестве задания на дополнительные баллы смоделируйте посадку космического корабля, при которой он влетает в атмосферу не в вертикальном состоянии. Следует считать, что α изменяется на $-2 \cdot 10^{-3\circ}$ каждую 1 мс (миллисекунду).

Другой расчет выполняется аналогично расчетам предыдущей таблицы, с учетом трения — $a_{\text{тр}}$.

t, мс	i	V_{ix}	V_{iy}
0	0	V_{1x}	0
1	1	$V_{0x} - a'_{\text{дв}}\sin(\alpha) - a_{\text{тр}}\sin(\alpha)$	$V_{0y} + g'_1 - a'_{\text{дв}}\cos(\alpha) - a_{\text{тр}}\cos(\alpha)$
2	2	$V_{1x} - a'_{\text{дв}}\sin(\alpha) - a_{\text{тр}}\sin(\alpha)$	$V_{1y} + g'_2 - a'_{\text{дв}}\cos(\alpha) - a_{\text{тр}}\cos(\alpha)$
.	.	.	.
.	.	.	.
$n-1$	$n-1$	$V_{(n-2)x} - a'_{\text{дв}}\sin(\alpha) - a_{\text{тр}}\sin(\alpha)$	$V_{(n-2)y} + g'_{(n-1)} - a'_{\text{дв}}\cos(\alpha) - a_{\text{тр}}\cos(\alpha)$
n	n	$V_{(n-1)x} - a'_{\text{дв}}\sin(\alpha) - a_{\text{тр}}\sin(\alpha)$	$V_{(n-1)y} + g'_n - a'_{\text{дв}}\cos(\alpha) - a_{\text{тр}}\cos(\alpha)$

Минимальное значение угла $\alpha = 0$. Если $\alpha = 0$, то корабль полностью повернулся на 90°.

Сила трения

В расчетах требуется учитывать $F_{\text{тр}} = ma_{\text{тр}}$ (сила трения об атмосферу). Сила трения при посадке космического корабля — это сумма сил трений при посадке.

$$F_{\text{тр}} = F_{\text{тр корабля}} = \frac{1}{2} \cdot pv^2 C_{d\ \text{корабля}} S_{\text{корабля}},$$

где p — плотность атмосферы, зависит от высоты;

v — скорость корабля;

$C_{dk} = 0,84$ (коэффициент сопротивления);

$S_{\text{корабля}}$ — площадь корабля.

Миссия 3. Атмосфера

Требуется визуально отразить слои атмосферы, которые должно быть видно из иллюминаторов космического аппарата:

- $>$ 300 км — внешний слой (см. рис. VI.2.7);
- 250–300 км — слой поглощающего ультрафиолетовое излучение тумана («фотохимический смог»);
- 100–200 км — непрерывно идущие дожди.

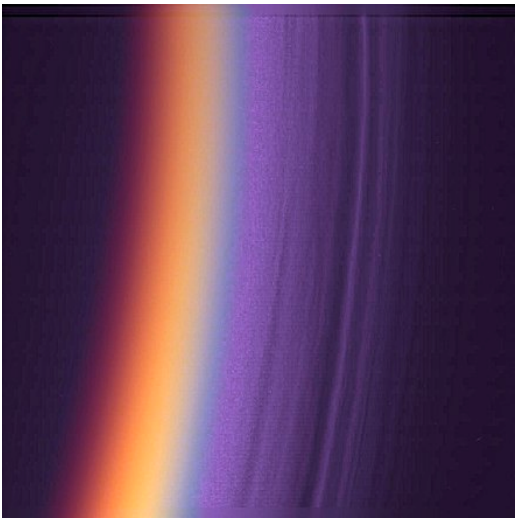


Рис. VI.2.7. Слоистое строение атмосферы. «Кассини», 2004 год. Изображение раскрашено в естественные цвета

В зависимости от слоя атмосферы космический аппарат должен испытывать на себе различные эффекты, которые необходимо отражать игроку, например:

- в области туманов и облаков должен появляться визуальный эффект тумана и будет снижаться видимость;

- во время дождя на иллюминаторах космического корабля должны появляться капли жидкости (предполагается реализация при помощи шейдеров).

С погружением в атмосферу на навигационной панели необходимо начать отображать плотность атмосферы. Для расчета плотности атмосферы используйте формулу ниже. Данные о температуре и давлении — см. рис. VI.2.8. Для расчетов следует использовать приблизительные значения по рисунку, важно продемонстрировать зависимость одной величины от другой.

Каждые 20 км необходимо обновлять данные о плотности атмосферы на навигационной панели.

$$p = \frac{p \cdot M}{R \cdot T},$$

где p — давление (зависит от h);

T — температура,

M — молярная масса $27,8 \cdot 10^{-3}$ кг/моль,

R — универсальная газовая постоянная (8,31 Дж/(К·моль)).

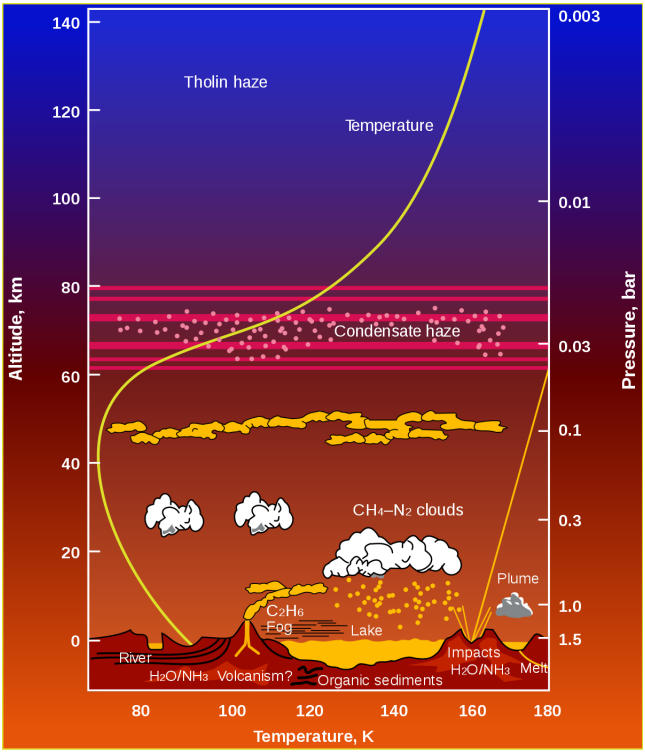


Рис. VI.2.8. Зависимость температуры и давления от высоты

Миссия 4. Пилотирование в условиях шторма

С помощью снимков орбитальной станции «Кассини», которая вошла в систему Сатурна в июле 2004 года, международная группа ученых из Франции, США, Европейского космического агентства, NASA и других университетов и институтов обнаружила на «Титане» наличие пылевых штормов (источник: https://www.jpl.nasa.gov/news/dust-storms-on-titan-spotted-for-the-first-time).

Рис. VI.2.9. Пылевая буря на «Титане» в представлении художника. Источник: IPGP/Labex UnivEarthS/University Paris Diderot — C. Epitalon & S. Rodriguez

Вам необходимо смоделировать модель такого шторма и научиться пилотировать корабль в таких условиях.

Сценарий

Во время снижения корабля, через некоторое время нахождения в атмосфере «Титана», на пути следования вашего корабля должен оказаться пылевой шторм. В этот момент на корабле запускается сигнализация о штормовом предупреждении.

Игроку необходимо подойти к пульту управления, отключить сигнализацию и запустить режим корректировки курса. Навигатор проложит новую траекторию, которая позволит преодолеть шторм проще всего. После того как корабль окажется внутри шторма, видимость из кабины пропадает. Игроку нужно вернуть управление в автоматический режим, покинуть штурвал и подойти к управлению стабилизацией корабля. В момент нахождения внутри пылевого шторма вихри и порывы будут периодически пытаться закрутить корабль, чему игрок должен препятствовать, поворачивая корабль в противоположное направление. После выхода из шторма видимость возвращается к нормальной, в этот момент игрок завершает режим стабилизации, и задача по преодолению шторма считается выполненной.

Механики

- Сигнализация. За 1 мин до попадания внутрь шторма на корабле должно произойти звуковое и визуальное оповещение. Внутренняя подсветка корабля должна смениться на сигнальный цвет предупреждения об опасности и плавно мигать. Звуковое оповещение должно содержать звук сирены с последующим уведомлением о штормовом предупреждении. Сигнализация должна продолжаться до тех пор, пока игрок не подойдет к панели управления и не отключит ее.
- Навигатор. На навигаторе должен отобразиться шторм и скорректированная траектория корабля для захода в него. Эта траектория должна существенно отличается от вашего текущего курса, которым следует корабль в режиме автопилота, для того чтобы игрок мог перейти в режим ручного управления и следовать ей.
- Ручное управление. С помощью двух контроллеров игрок должен направлять корабль согласно траектории, предложенной навигатором. Навигатор должен реагировать на действия игрока и показывать текущее положение корабля относительно курса. Задача игрока — вывести корабль на заданный курс.
- Заход в шторм. По истечению времени навигационная панель должна стать недоступной, поскольку внутри шторма затруднена связь. Игрок должен перевести корабль в режим автоматического управления и оперативно подойти к панели для осуществления стабилизации корабля.
- Управление стабилизацией (в режиме автопилота). На приборной панели игрок должен взять виртуальные пульты управления в каждую руку и, ориентируясь на показания приборов, корректировать вращение корабля. Управление должно осуществляться посредством ввода с джойстиков VR-контроллеров, пока в руке у игрока находятся виртуальные пульты. Правый пульт отвечает за крен корабля (ось X джойстика) и тангаж (ось Y джойстика) корабля. Левый пульт отвечает за рыскание (ось X джойстика).
- Корабль будет совершать периодические случайные вращения по всем трем осям, задача игрока — вовремя реагировать на изменение угла и корректировать его.
- Механика сложности игры. В зависимости от того, насколько сильно игрок отклонился от построенного курса при заходе в шторм, в режиме управления стабилизацией игроку будет сложнее удерживать корабль. Вам необходимо самостоятельно придумать метрику качества захода корабля по заданному курсу с условиями, что если отклонение минимально, то корабль будет совершать редкие, небольшие, медленные повороты, и время до вылета из шторма будет также минимальным; если же игрок вообще не прикасался к ручному управлению, то повороты будут максимально частыми, в больших количествах и быстрые, время прохождения шторма будет также максимальным.
- Выход из шторма. После завершения шторма видимость из кабины должна вернуться к прежней. В этот момент игрок может положить пульты управления стабилизацией на место.

Требования к приспособлениям

- Панель управления. Состоит из навигационного модуля, текущего состояния, а также виртуальных кнопок для смены режимов управления и отключения сигнализации. Имеется текстовое поле, информирующее о текущем режиме.
- Пульты управления стабилизацией. Представляют собой два виртуальных контроллера, каждый из которых помещаются в одну руку. Различаются по внешнему виду, чтобы не путать левый и правый пульты управлений.

Требования к визуальному оформлению

- Пылевой шторм должен быть визуально похож на объемное, непрозрачное, быстро движущееся облако пыли. При попадании в центр шторма вокруг корабля должны пролетать частицы пыли, видимость сквозь него изнутри должна быть почти нулевой.
- Приборы, отражающие текущее положение корабля в режиме управления стабилизации, могут быть выполнены в произвольной стилистике — требуется лишь удобство ориентирования и контроля корабля по показаниям этих приборов.

Условия победы в модуле

Игрок прошел все описанные стадии и успешно преодолел шторм.

Условия поражения в модуле

Во время управления стабилизацией отклонение от начального положения по любой из осей по модулю превышает 180°.

Миссия 5. Раскрытие парашюта

Метановые облака

Метановые облака необходимо визуально исполнить, пройдя которые нужно готовиться к раскрытию парашюта.

Раскрытие парашюта

Парашют следует раскрывать на высоте 50 км, что позволяет увеличить лобовое сопротивление. Выброс парашюта должно вызывать торможение космического аппарата. Чтобы раскрыть парашют, игроку необходимо дернуть за специальный ручной трос под потолком.

Сила трения

В расчетах требуется учитывать силу трения при раскрытии парашюта. Сила трения при посадке космического корабля — это сумма сил трений при посадке.

Силу трения следует учитывать на высоте до 140 км.

$$F_{\text{тр}} = F_{\text{тр корабля}} + F_{\text{тр парашюта}} =$$

$$= \frac{1}{2} \cdot pv^2 C_{d\ \text{корабля}} S_{\text{корабля}} + \frac{1}{2} \cdot pv^2 C_{d\ \text{парашюта}} S_{\text{парашюта}} =$$

$$= \frac{1}{2} pv^2 (C_{dk} S_k + C_{d\text{п}} S_n),$$

где p — плотность атмосферы, зависит от высоты;

$C_{dk} = 0,84$, $C_{d\text{п}} = 1,6$ (коэффициенты сопротивления);

v — скорость корабля;

$S_{\text{корабля}}$, $S_{\text{парашюта}}$ — площади сечения корабля и парашюта соответственно, которые рассчитываются вами самостоятельно.

Сборка и интеграция с LIV

После завершения работы над проектом вам необходимо выполнить его сборку в исполняемый файл (`.exe`) для того, чтобы иметь возможность запускать его на любом устройстве.

Если ваш проект не зависит от VR-гарнитуры, которая использовалась при его разработке, и может запускаться с любыми совместимы с PC-устройствами, вы также можете получить дополнительный балл.

LIV

В этом году для демонстраций работ участников будет использоваться захват изображения с участником в шлеме на хромакее и последующим наложением игрового окружения.

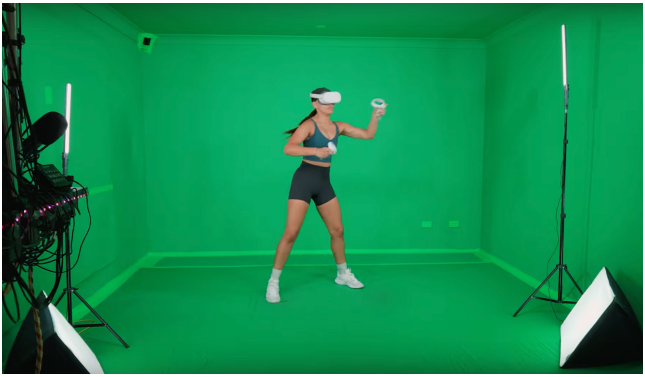


Рис. VI.2.10. Пример применения техники наложения изображения в игре Beat Saber

Для того чтобы вы могли демонстрировать свой проект в таком режиме, необходимо добавить и настроить в вашем проекте пакет LIV. LIV SDK поддерживает только Unity (кроме HDRP) и Unreal Engine.

Инструкция для Unity

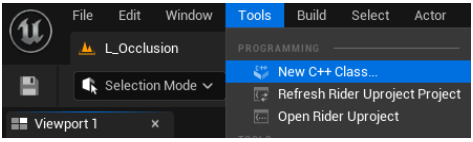
1. Скачайте файл unitypackage по ссылке: https://drive.google.com/file/d/1gL_iHc31pONK3jQhilx4FWyPQJId_vhF/view?usp=sharing.
2. Сделайте резервную копию вашего проекта (сделайте коммит, если используете систему контроля версий).
3. Добавьте пакет в ваш проект.
4. Добавьте в каждую сцену компонент LIV.
5. В поле **HMD Camera** добавьте камеру игрового персонажа.
6. В поле **Stage** добавьте родительский объект игрового персонажа.

Если вы используете `Universal Render Pipeline`, то дополнительно выполните следующее:

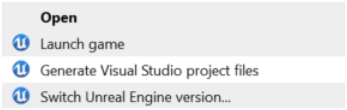
1. Найдите `File` → `Build Settings` → `Player Settings` → `Other Settings` → `Scripting Define Symbols`.
2. Введите: `LIV_UNIVERSAL_RENDER`.
3. Найдите ваши `Forward Renderer Assets` в Unity проекте (примечание: в вашем проекте могут быть несколько таких ассетов; убедитесь в том, что вы выполняете это для всех используемых ассетов, которые участвуют в сценах с LIV).
4. Добавьте SDK Universal Render Feature после Renderer Features.
5. На добавленных вами на сцены компонентах LIV включите `Fix Post-Effects alpha channel`.

Инструкция для Unreal Engine

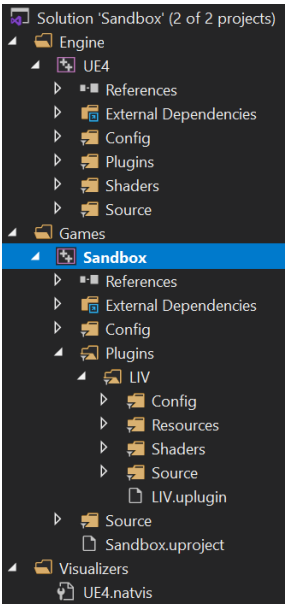
1. Скачайте архив `zip` по ссылке: https://drive.google.com/file/d/1OVCDsRAFQyT5mM8AeymAuhGBQ2STqg1o/view?usp=drive_link и распакуйте его.
2. Скопируйте папку LIV в папку `Plugins` в директории вашего проекта. Создайте эту папку, если ее еще не существует.
3. Если ваш проект до сих пор не поддерживает C++, добавьте эту возможность, выбрать «`Tools`» New C++ Class.



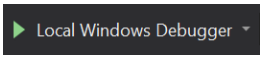
4. Перегенерируйте решение Visual Studio для вашего проекта. Чтобы это сделать, нажмите ПКМ по вашему `.uproject` файлу, затем выберите `Generate Visual Studio project files`.



5. Откройте решение Visual Studio (`.sln`) и убедитесь, что LIV загружен в обозреватель решения.



6. Запустите `Unreal Editor` для вашего проекта через Visual Studio, выбрав `Build and Run` вашего проекта для запуска редактора. Вы можете нажать F6 или кнопку, изображенную ниже.



7. Убедитесь, что плагин LIV загружен в окне `Plugins` (`Edit > Plugins`).

Система оценивания

Задача оценивается независимо разными экспертами. Участники получают усредненный балл по каждому критерию.

- **Миссия 1: Выведение космического аппарата на орбиту**
  - Навигационная панель: визуализация орбиты и траектории курса: 3 балла.
  - Навигационная панель: предупреждение о рисках: 2 балла.
  - Навигационная панель: расстояние до поверхности, скорость космического аппарата: 2 балла.
  - Возможность менять коэффициент ускорения времени: 2 балла.
  - Панель управления: режим автопилота: 3 балла.
  - Панель управления: режим ручного управления: 1 балл.
  - Корабль движется по орбите (аппарат обладает первой космической скоростью, балл за физику): 3 балла.
  - Контроллеры: корректно реализованы крен, рыскание, тангаж: 3 балла.
  - Включение двигателя, регулирование тяги: 1 балл.
  - Визуализация траектории для мягкой посадки: 3 балла.
  - В качестве космического корабля используется упомянутая в ТЗ модель (2 балла): 2 балла.
  - Модель переключателя режимов управления: 2 балла.
  - Модель ручного троса: 2 балла.
  - Модели контроллеров для управления космическим кораблем: 3 балла.
- **Миссия 2: Поворот**

  - Моделирование посадки с учетом сил тяжести, силы двигателя: 3 балла.
  - Поворот с учетом атмосферы, силы трения (доп баллы): 2 балла.
  - Физические расчеты по формулам: 2 балла.
- **Миссия 3: Атмосфера**
  - Визуальной реализации внешнего слоя атмосферы и «Фотохимический смога»: 2 балла.
  - Визуальной реализации непрерывно идущих дождей: 2 балла.
  - Визуальной реализации метанового тумана и облаков: 2 балла.
  - Реализовано снижение видимости в тумане и облаках: 1 балл.
  - Реализации появления капель жидкости во время дождя при помощи самостоятельно написанных шейдеров: 5 баллов.
  - На навигационной панели отображается корректно рассчитанное давление: 1 балл.
  - Физические расчеты по формулам: 2 балла.
- **Миссия 4: Пилотирование в условиях шторма**
  - Пылевой шторм реализован визуально: 1 балл.
  - Присутствует штормовое предупреждение: 1 балл.
  - Снижается видимость внутри шторма: 1 балл.
  - Шторм пытается снести корабль: 2 балла.
  - Игрок может стабилизировать корабль во время шторма: 2 балла.
  - Реализован навигатор, отображающий местоположение шторма, траектории, корабля: 2 балла.
  - Недоступность навигационной панели во время шторма: 1 балл.
  - Реализована механика сложности игры (трудное маневрирование): 2 балла.
- **Миссия 5: Раскрытие парашюта и посадка**
  - Механика ручного троса для раскрытия парашюта: 2 балла.
  - Выше 50 км парашют не действует: 1 балл.
  - Парашют замедляет движение: 1 балл.
  - Физические расчеты: сопротивление (парашюта — 1, корабля — 1): 2 балла.
  - Перемещение капсулы с парашютом в отсек для парашюта, для раскрытия: 1 балл.
- **3D графика**
  - Модель отсека для парашюта: 1 балл.
  - Модель капсулы для парашюта: 1 балл.
  - Модель кнопки для режима автопилота: 2 балла.
  - Модель троса для раскрытия парашюта: 2 балла.
  - Модель парашюта: 1 балл.
  - Анимация раскрытия парашюта: 1 балл.
- **Общее**
  - Качество Skybox: 1 балл.
  - Материалы, развертки, текстуры на самостоятельно разработанных 3D-моделях: 5 баллов.

- Стилистическое единство: 2 балла.
- Геймдизайн игрового процесса: 1 балл.
- Наличие звуковых эффектов и соответствие их окружению: 2 балла.
- Механика проигрыша (жесткая посадка, крах): 2 балла.
- Механика выигрыша (мягкая посадка): 2 балла.
- Возможность начать с определенной миссии: 1 балл.
- Сохранение и загрузка: 3 балла.
- Качество репозитория: 3 балла.
- Непредусмотренные ТЗ фичи: 3 балла.
- Дополнительные баллы: 7 баллов.
- Критические проблемы: −10 балла.

Всего: до 103 баллов.

Решение задачи

Решение команды победителя In time. Репозиторий проекта находится по ссылке: https://gitlab.com/nto-vr/2024/in-time.

Скрипт `ShipPhysicMission1.cs` содержит реализацию физических расчетов, необходимых для совершения посадки с учетом влияния физических сил.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ShipPhysicMission1 : MonoBehaviour
{
    public float G;
    //in physical universe every body would be both attractor and attractee
    public Rigidbody ship;
    public Rigidbody planet;
    public bool isSimulatingLive = true;
    private void Start()
    {
        ship.AddForce(ship.transform.forward * 10 * ship.mass);
    }
    void FixedUpdate()
    {
        if (isSimulatingLive)//PathHandler changes this
            SimulateGravities();
    }
    public void SimulateGravities()
    {
        AddGravityForce(planet, ship);
    }

    public void AddGravityForce(Rigidbody attractor, Rigidbody target)
    {
        float massProduct = attractor.mass * target.mass * G;

        //You could also do
        //float distance = Vector3.Distance(attractor.position,target.position.
        Vector3 difference = attractor.position - target.position;
```

```
        float distance = difference.magnitude; // r = Mathf.Sqrt((x*x)+(y*y))

        //F = G * ((m1*m2)/r^2)
        float unScaledforceMagnitude = massProduct / Mathf.Pow(distance, 2);
        float forceMagnitude = G * unScaledforceMagnitude;

        Vector3 forceDirection = difference.normalized;

        Vector3 forceVector = forceDirection * forceMagnitude;

        target.AddForce(forceVector);
    }
}
```

Скрипт `InfoPanel.cs` содержит в себе реализацию визуализации информационной панели и код изменения давления в зависимости от высоты.

```
using System.Collections;
using System.Collections.Generic;
using TMPro;

using UnityEngine;
using UnityEngine.SceneManagement;


public class InfoPanel : MonoBehaviour
{
    public GameObject ship;
    public Transform planet;
    public TMP_Text speedText;
    public TMP_Text ditanceText;
    public TMP_Text warningText;
    public TMP_Text PresuhereText;
    public float maxSpeed;
    public GameObject warningBar;
    public bool isScene2;
    public float density;
    public float distanceEnd;
    public int scene;

    private float speed;
    private float oldDistance;
    private float distanceToPlanet;
    private Rigidbody shipRB;

    private float molarM = 27.8f * Mathf.Pow(10, -3);
    private float preshure;
    private float constR = 8.31f;
    private float T = 1000;


    public void Start()
    {
        shipRB = ship.GetComponent<Rigidbody>();
        oldDistance = Vector3.Distance(planet.position, ship.transform.position);
    }

    private void FixedUpdate()
    {
        speed = shipRB.velocity.magnitude / 3.6f;
```

```
44          distanceToPlanet = Vector3.Distance(planet.position,
            ↪  ship.transform.position);
45          if(distanceToPlanet <= distanceEnd)
46          {
47              SceneManager.LoadScene(scene);
48          }
49          if (isScene2)
50          {
51
52              if (distanceToPlanet <= oldDistance - 20f)
53              {
54                  UpdatePreshure();
55                  oldDistance -= 20;
56              }
57          }
58
59          UpdateText();
60      }
61
62      void UpdateText()
63
64      {
65          speedText.text = (Mathf.Round(speed*1000f)/400f).ToString() + " * 10^3 /";
66          ditanceText.text = distanceToPlanet.ToString() + " ";
67          if (speed > maxSpeed)
68          {
69              warningText.text = "      !";
70              warningBar.SetActive(true);
71          }
72          else
73          {
74              warningText.text = "";
75              warningBar.SetActive(false);
76          }
77      }
78
79      void UpdatePreshure()
80      {
81          density = (preshure * molarM) / (constR * T);
82          density = Mathf.Round(density*10000000)/100;
83          Debug.Log(density);
84          if (distanceToPlanet < 20)
85          {
86              T = 170f;
87              preshure = 1.0f;
88              PresuhereText.text = ": " + density.ToString() + "/^3";
89          }
90          else if (distanceToPlanet < 40)
91          {
92              T = 140f;
93              preshure = 0.3f;
94              PresuhereText.text = ": " + density.ToString() + "/^3";
95          }
96          else if (distanceToPlanet < 60)
97          {
98              T = 120f;
99              preshure = 0.06f;
100             PresuhereText.text = ": " + density.ToString() + "/^3";
101         }
102         else if (distanceToPlanet < 80)
```

```
103         {
104             T = 100f;
105             preshure = 0.02f;
106             PresuhereText.text = ": " + density.ToString() + "/^3";
107         }
108         else if (distanceToPlanet < 100)
109         {
110             T = 80f;
111             preshure = 0.01f;
112             PresuhereText.text = ": " + density.ToString() + "/^3";
113         }
114         else if (distanceToPlanet < 120)
115         {
116             T = 60f;
117             preshure = 0.006f;
118             PresuhereText.text = ": " + density.ToString() + "/^3";
119         }
120         else if (distanceToPlanet < 140)
121         {
122             T = 40f;
123             preshure = 0.003f;
124             PresuhereText.text = ": " + density.ToString() + "/^3";
125         }
126         else if (distanceToPlanet < 160)
127         {
128             T = 40f;
129             preshure = 0.003f;
130             PresuhereText.text = ": " + density.ToString() + "/^3";
131         }
132         else
133         {
134             PresuhereText.text = "    ";
135         }
136     }
137 }
```

Скрипт `SandStorm.cs` содержит реализацию пылевого шторма и уведомление пользователя различными визуальными сигналами.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using TMPro;
5  using Unity.VisualScripting;
6  using UnityEngine.Rendering.Universal;
7  using UnityEngine.Rendering;
8
9
10 public class SandStorm : MonoBehaviour
11 {
12     public GameObject warningBar1;
13     public GameObject warningBar2;
14     public GameObject warningSignal;
15
16     public TMP_Text warningText1;
17     public TMP_Text warningText2;
18
19     public Color32 start;
20     public Color32 end;
21
```

```
22      public float speed;
23
24      public Light warningLight;
25      private ShipController shipController;
26      private void Start()
27      {
28          shipController =  FindObjectOfType<ShipController>();
29      }
30
31      public void WarningOff()
32      {
33          warningBar1.SetActive(false);
34          warningBar2.SetActive(false);
35          warningSignal.SetActive(false);
36          warningText1.text = "";
37          warningText2.text = "";
38      }
39
40      private void OnTriggerEnter(Collider other)
41      {
42          if (other.tag == "ship")
43          {
44              shipController.isSandStorm = true;
45              warningBar1.SetActive(true);
46              warningBar2.SetActive(true);
47              warningSignal.SetActive(true);
48              warningText1.text = "!  ";
49              warningText2.text = "!  ";
50          }
51      }
52
53      private void OnTriggerStay(Collider other)
54      {
55          if (other.tag == "ship")
56          {
57              float lerp = Mathf.PingPong(Time.time * speed, 1.0f);
58              warningLight.color = Color32.Lerp(start, end, lerp);
59          }
60      }
61
62      private void OnTriggerExit(Collider other)
63      {
64          if (other.tag == "ship")
65          {
66              shipController.isSandStorm = false;
67              warningBar1.SetActive(false);
68              warningBar2.SetActive(false);
69              warningSignal.SetActive(false);
70              warningText1.text = "";
71              warningText2.text = "";
72          }
73      }
74  }
```

Материалы для подготовки

1. Курс линейной алгебры: https://stepik.org/course/2461.

2. Основы C/C++ для спортивного программирования: https://stepik.org/course/80538.
3. Программирование на C#: https://stepik.org/course/4143.
4. VR-интенсив: https://stepik.org/course/4566.
5. Курс Unity: https://unity.com/ru/learn.
6. Курс по основам работы в Unreal Engine: https://ed.vrnti.ru/unrealengine.
7. Unreal Online Learning: https://www.unrealengine.com/en-US/onlinelearning.
8. Программное обеспечение. Unity3D: https://docs.unity3d.com.

Критерии определения победителей и призеров

Первый отборочный этап

В первом отборочном этапе участники решали задачи предметного тура по двум предметам: информатике и математике 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 не менее 60 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 83 балла, независимо от уровня.

Заключительный этап

Индивидуальный предметный тур

- информатика — максимально возможный балл за все задачи — 100 баллов;
- математика — максимально возможный балл за все задачи — 100 баллов.

Командный инженерный тур

Команды заключительного этапа получали за командный инженерный тур от 0 до 113 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 предметам (информатика, математика) с весом $K_1 = 0,2$ каждый предмет и баллы за командное решение задач инженерного тура с весом $K_2 = 0,65$.

Итоговый балл определяется по формуле:

$$S = K_1 \cdot (S_1 + S_2) + K_2 \cdot S_3,$$

где S_1 — балл первой части заключительного этапа по информатике (предметный тур) в стобалльной системе ($S_{1\ \text{макс}} = 100$);

S_2 — балл первой части заключительного этапа по математике (предметный тур) в стобалльной системе ($S_{2\ \text{макс}} = 100$);

S_3 — итоговый балл инженерного командного тура в стобалльной системе ($S_{3\ \text{макс}} = 100$).

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 3 победителя и 7 призеров (первые 25% участников рейтинга становятся победителями или призерами, из них первые 8% становятся победителями, оставшиеся — призерами).

Критерий определения победителей и призеров (независимо от уровня)

Категория	**Количество баллов**
Победители	58,80 и выше
Призеры	От 51,78 до 56,04

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.

Важная задача наставника — превратить неудачу в инструмент будущего успеха. Для этого необходимо вместе с учениками наметить план развития компетенций и подготовки к будущему сезону Олимпиады. Подробные материалы о проведении рефлексии представлены в курсе «Наставник НТО»: https://academy.sk.ru/events/310.

Наставнику важно проинформировать руководство образовательного учреждения, если его учащиеся стали финалистами, призерами и победителями. Публичное признание высоких результатов дополнительно повышает мотивацию.

В процессе рефлексии с учениками, не ставшими призерами или победителями, рекомендуется уделить особое внимание особенностям командной работы: распределению ролей, планированию работы, возникающим проблемам. Для этого могут использоваться опросники для самооценки собственной работы и взаимной оценки участниками других членов команды (P2P). Такие опросники 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.

Еще одним направлением работы наставника после НТО может стать создание кружка по направлению профилей или по формированию необходимых компетенций: программирование, электроника, робототехника, 3D-моделирование и т. п. Формат подобного кружка может быть различным: короткие модули, дополнительные курсы, факультативы, группы дополнительного образования. Для создания кружков можно воспользоваться образовательными программами, опубликованными на сайте НТО: https://ntcontest.ru/mentors/education-programs/.

Важным фактором успешного участия в следующих сезонах НТО может стать поддержка родителей учеников. Знакомство с родителями помогает наставнику продемонстрировать им 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».