НТО

МАТЕРИАЛЫ ЗАДАНИЙ

Всероссийской междисциплинарной олимпиады школьников
«Национальная технологическая олимпиада»

по профилю
«Технологии компьютерного зрения и цифровые сервисы»

2023/24 учебный год

http://ntcontest.ru

УДК 373.5.016:004.9
ББК 74.263.2
Т38

Авторы:
И.А. Андриенко, М.В. Бабушкин, А.Г. Балахчи, Д.А. Казаков, И.А. Кобец, Д.И. Лукашевский, Е.А. Лутковская, Е.В. Милованович, П.А. Паткина, Ю.В. Пестова, И.С. Петрушин, Н.А. Серебрянская, Я.А. Угорская, Т.С. Юрова

Т38 Всероссийская междисциплинарная олимпиада школьников 8-11 класса «Национальная технологическая олимпиада». Учебно-методическое пособие
Том 34 **Технологии компьютерного зрения и цифровые сервисы**
—М.: ООО «ВАШ ФОРМАТ», 2024. — 184 с.

ISBN 978-5-00147-624-5

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:
01.00.00 Математика и механика
02.00.00 Компьютерные и информационные науки
09.00.00 Информатика и вычислительная техника

ISBN 978-5-00147-624-5

УДК 373.5.016:004.9
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 компьютерного зрения и цифровые сервисы

Цель профиля «Технологии компьютерного зрения и цифровые сервисы» состоит в исследовании, разработке и применении методов компьютерного зрения для создания инновационных цифровых сервисов и продуктов.

Задачи профиля «Технологии компьютерного зрения и цифровые сервисы»:

1. Разработка алгоритмов компьютерного зрения: создание новых алгоритмов и методов обработки изображений для распознавания объектов, классификации изображений, сегментации изображений и других задач.
2. Применение машинного обучения: использование методов машинного обучения, таких как нейронные сети, для улучшения производительности систем компьютерного зрения, а также для обучения моделей на больших объемах данных.
3. Разработка цифровых сервисов: создание программных продуктов и сервисов, использующих технологии компьютерного зрения для решения конкретных задач, например, системы видеонаблюдения, автоматическое распознавание лиц, медицинские диагностические системы и многое другое.
4. Исследование новых применений: исследование и разработка новых областей применения технологий компьютерного зрения, таких как виртуальная и дополненная реальность, автономные транспортные средства, медицинская диагностика и т. д.
5. Мотивация интереса к поиску нового контекста в разработке и применении технологии.

Профиль «Технологии компьютерного зрения и цифровые сервисы» объединяет экспертизу в области компьютерного зрения, машинного обучения, обработки изображений, а также в различных областях применения, таких как медицина, автоматизация производства, биометрия, робототехника, умный город и другие.

В 2023–2024 году профиль был посвящен разработке цифрового сервиса на основе библиотеки `OpenCV`, представляющего собой интерактивную демонстрационно-игровую проекционную систему. Интерактивная демонстрационно-игровая проекционная система на основе библиотеки `OpenCV` является увлекательным и инновационным способом использования технологий компьютерного зрения для создания интерактивного опыта для пользователей. Участники финала реализовывали игровой комплекс. В игре «Мячик-метатель» игрок становится участником захватывающего испытания, бросая мячик по виртуальным объектам, проецируемым на стену. Цель игры состоит в том, чтобы попасть мячиком в объекты, например, воздушные шары, что приводит к их исчезновению и зарабатыванию очков. Участникам финала же предстояло отслеживатель, как цели-проекции, так и коллизию реальный мяч-проекция.

Вторая задача состояла в разработке игры, в которой игрокам предстоит загнать как можно больше виртуальных мячиков в подвижную проекцию корзины. Для достижения этой цели они могут использовать возможность рисовать линии на

проецируемой поверхности. Виртуальные мячи, выпущенные игроками, взаимодействуют с этими маркерными линиями: скатываются по ним и отскакивают, изменяя свое направление. Игрокам предстоит стратегически распределить линии таким образом, чтобы максимальное количество мячиков попало в корзину, требуя от них не только точности и меткости, но и умения строить эффективные траектории. Успех в этой задаче зависит от быстроты реакции и умения предвидеть движение мячиков, создавая захватывающий и динамичный игровой опыт. В данной задаче участники финала не только использовали инструменты компьютерного зрения, но моделировали поведение проекций виртуальных объектов при их коллизии с маркерными линиями.

Первый отборочный дистанционный этап (индивидуальный) определяет общий уровень подготовки школьников по предметам математика и информатика. На этом этапе участникам также предоставляется возможность продемонстрировать свои знания по основам технологии, решая задачи инженерного тура. Здесь предлагаются индивидуальные задания: тест на основные понятия по технологии, создание трехмерной модели по заданным параметрам, задачи на алгоритмы компьютерного зрения.

Задачи второго отборочного этапа представлены комплексным командным заданием, которое направлено на ознакомление участников с технологиями компьютерного зрения, как составляющими основу дополненной реальности в целом, так и являющимися основной технологией в проекте 2023–2024 года, направленном на создание цифрового сервиса, представляющего собой интерактивную демонстрационно-игровую проекционную систему.

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.

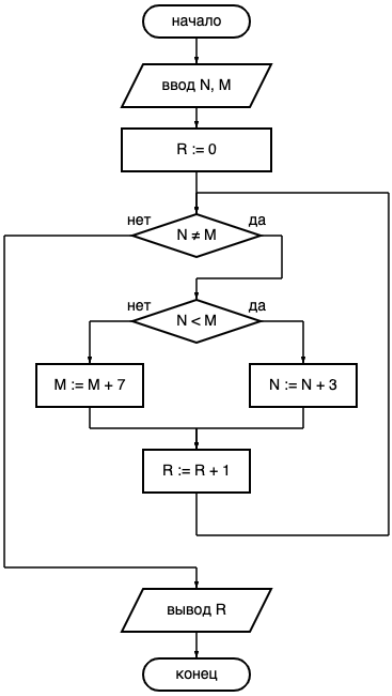
Ответ: 186.

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 $M = 57$.

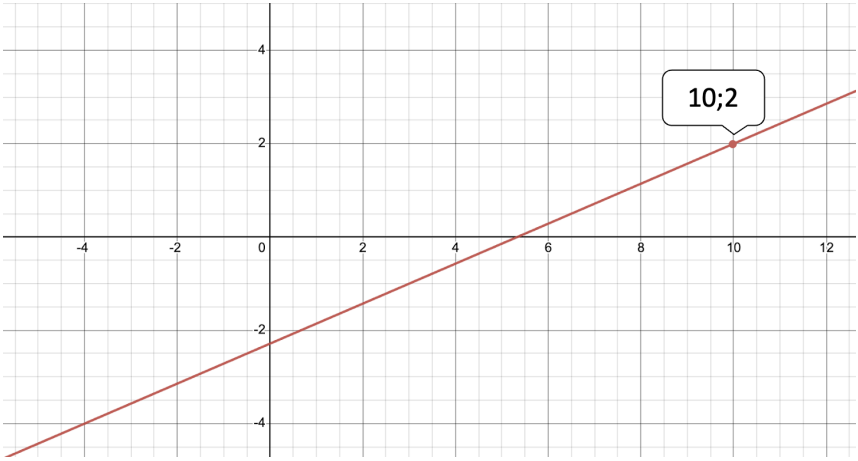


Решение

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

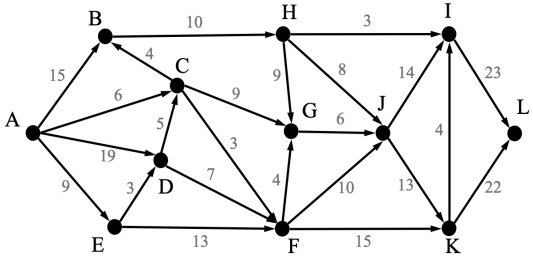
На рисунке приведена схема района «Северный», где каждая вершина графа, показанная латинскими буквами от A до L, обозначают объекты его инфраструктуры, а ребра — дороги между ними.

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.

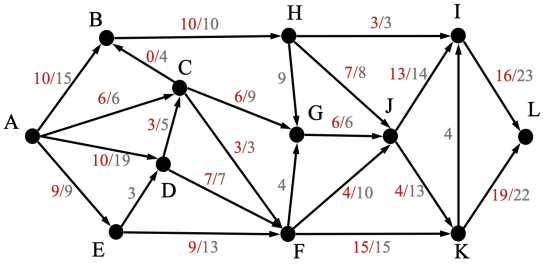


Решение

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

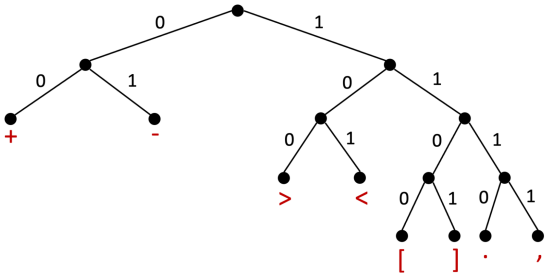
Даня и Ваня на уроке информатики получили очень странное задание. В нем им необходимо определить, какое минимальное количество информации будет содержать сообщение о способе перемещения случайного прохожего. Для этого они

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.

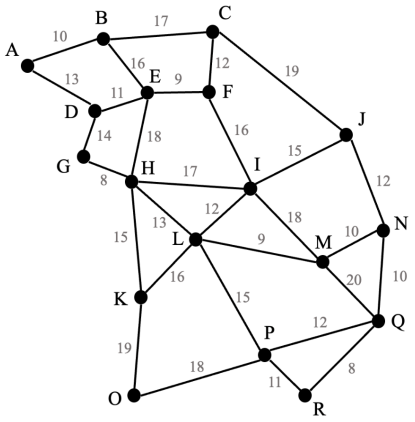
Ответ: 13.

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.



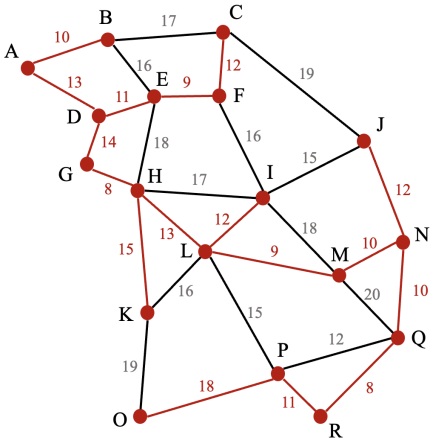
Главным условием является то, что до каждого из зданий должна быть проложена хотя бы одна дорога.

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.

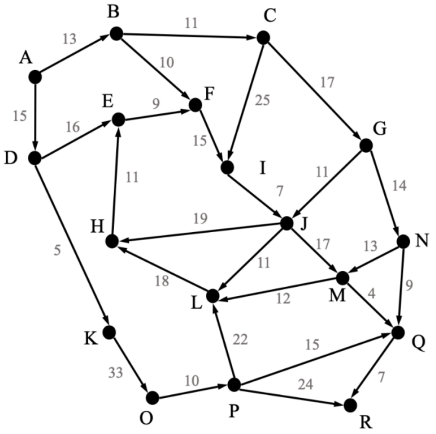
Ответ: 1723.

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.

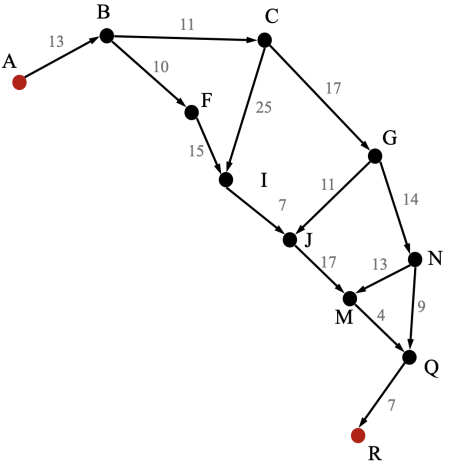


Стартом заезда считается пункт A, а финишем пункт Q. Они замерили максимальное время, которое им удается продержаться на каждом дорожном участке. После замеров Даня и Ваня просят у вас помощи. Найдите максимально возможное время заезда при условии, что через каждый пункт можно проезжать только один раз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

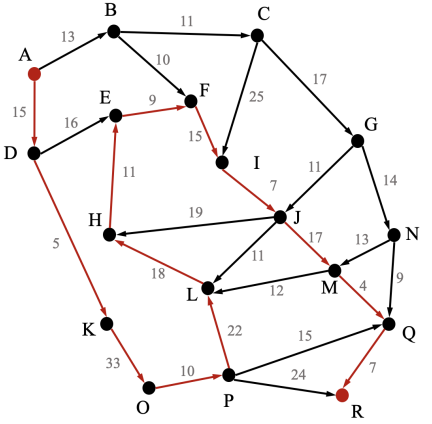
Решение

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)

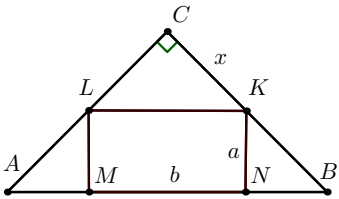
Темы: планиметрия.

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

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

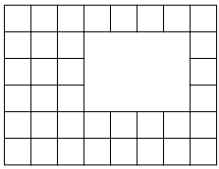
Ответ: $1 + \dfrac{p(n-1)}{100(2n-1)}$.

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.

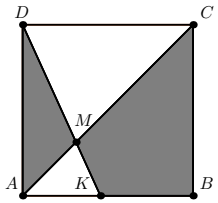
Условие

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)

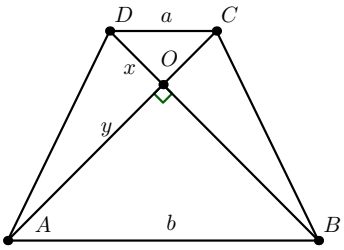
Темы: планиметрия, трапеция.

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

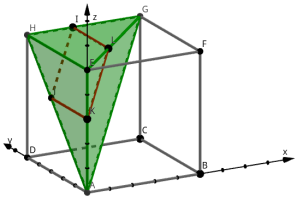
Выбирая три числа из $[0, a]$, назовем x — наименьшее из них, z — наибольшее, а y — лежащее между x и z. Выбор чисел x, y, z равносилен выбору точки $\alpha(x, y, z)$ в пространстве, удовлетворяющей условиям:

- 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

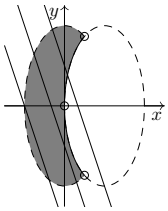
$$\log_{x^2+\frac{y^2}{a^2}} 2x \geqslant 1, \tag{II.2.1}$$

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.

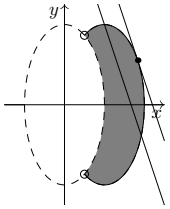


Рассматривая всевозможные прямые $y = m - bx$, проходящие через данную область (некоторые из этих прямых изображены на рисунке), видим, что наибольшего значения m не существует (можно сколь угодно близко приближать прямую к границе области).

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)

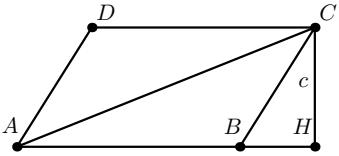
Темы: планиметрия, параллелограмм.

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

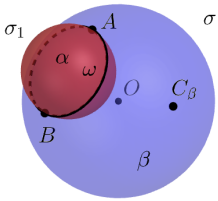
Точки A и B лежат на сфере с центром O так, что угол AOB равен α°. Случайно на сфере выбирается еще одна точка C (любой выбор равновозможен). Определите вероятность того, что угол ACB окажется острым.

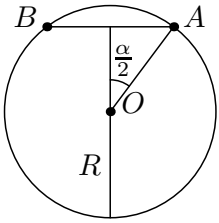
Запишите ответ в процентах с точностью до 0,01.

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

Задача II.3.1. (5 баллов)

Темы: задача классификации изображений, предобученная модель машинного обучения, загрузка модели с платформы hugging face.

Условие

В объектив фотографа–путешественника попали разнообразные животные. Создать приложение, определяющее присутствие на фотографии животных, заданных классов или констатирующее факт, что таких животных на фотографии нет.

Для создания приложения использовать предобученную модель AliGhiasvand86/10-animals-classification на платформе hugging face (https://huggingface.co/).

Датасет для тестирования модели можно скачать по ссылке ссылка на (https://drive.google.com/file/d/1AVzltSVseiKRexrmyJNdRY6SDkDL5Faj/view?usp=drive_link).

Найдите и скачайте нужную модель с платформы hugging face.

Ответьте на следующий вопрос: какие классы животных умеет определять модель?

Выберите из списка те классы, которые описаны в модели:

1. tiger;
2. giant panda;
3. cat;
4. butterfly;
5. spider;
6. dog;
7. horse;
8. lion;
9. eagle;
10. pinguin;
11. sheep;
12. monkey;
13. dolphin;
14. cow;
15. fox.

Ответ: 2, 3, 5, 6, 7, 9, 11, 12, 13, 14.

Задача II.3.2. (5 баллов)

Темы: задача классификации изображений, предобученная модель машинного обучения, тестирование модели.

Условие

В объектив фотографа–путешественника попали разнообразные животные. Создать приложение, определяющее присутствие на фотографии животных, заданных классов или констатирующее факт, что таких животных на фотографии нет.

Для создания приложения использовать предобученную модель AliGhiasvand86/10-animals-classification на платформе hugging face (https://huggingface.co/).

Датасет для тестирования модели можно скачать по ссылке ссылка на (https://drive.google.com/file/d/1AVzltSVseiKRexrmyJNdRY6SDkDL5Faj/view?usp=drive_link).

Загрузите проверочный датасет на свой компьютер.

Сколько файлов содержит проверочный датасет?

Ответ: 111.

Задача II.3.3. (2 балла)

Темы: задача классификации изображений, предобученная модель машинного обучения, тестирование модели.

Условие

Создайте программу на python, подключающую обученную модель AliGhiasvand86/10-animals-classification. На вход программе подается путь к каталогу, содержащему тестовые изображения (тестовые изображения скачиваются по ссылке). На выходе программа выдает вероятность принадлежности ко всем классам животных, которые умеет определять модель.

Чему равна вероятность принадлежности изображения на объекте классу «cow»?

Введите ответ в формате десятичной дроби округленной до тысячных. Пример: 0,988.

Ответ: 0,791.

Задача II.3.4. (2 балла)

Темы: задача классификации изображений, предобученная модель машинного обучения, тестирование модели.

Условие

Создайте программу на python, подключающую обученную модель AliGhiasvand86/10-animals-classification. На вход программе подается путь к каталогу, содержащему тестовые изображения (тестовые изображения скачиваются по ссылке). На выходе программа выдает вероятность принадлежности ко всем классам животных, которые умеет определять модель.

Чему равна вероятность принадлежности изображения на объекте классу «cow»?

Введите ответ в формате десятичной дроби округленной до тысячных. Пример: 0,988.

Ответ: 0,475.

Задача II.3.5. (2 балла)

Темы: задача классификации изображений, предобученная модель машинного обучения, тестирование модели.

Условие

Создайте программу на python, подключающую обученную модель AliGhiasvand86/10-animals-classification. На вход программе подается путь к каталогу, содержащему тестовые изображения (тестовые изображения скачиваются по ссылке). На выходе программа выдает вероятность принадлежности ко всем классам животных, которые умеет определять модель.

Чему равна вероятность принадлежности изображения на объекте классу «horse»?

Введите ответ в формате десятичной дроби округленной до тысячных. Пример: 0,988.

Ответ: 0,943.

Задача II.3.6. (2 балла)

Темы: задача классификации изображений, предобученная модель машинного обучения, тестирование модели.

Условие

Создайте программу на python, подключающую обученную модель AliGhiasvand86/10-animals-classification. На вход программе подается путь к каталогу, содержащему тестовые изображения (тестовые изображения скачиваются по ссылке). На выходе программа выдает вероятность принадлежности ко всем классам животных, которые умеет определять модель.

Чему равна вероятность принадлежности изображения на объекте классу «stool»?

Введите ответ в формате десятичной дроби округленной до тысячных. Пример: 0,988.

Ответ: 0,345.

Задача II.3.7. (10 баллов)

Темы: задача классификации изображений, предобученная модель машинного обучения, тестирование модели.

Условие

Скачайте на свой компьютер проверочный датасет `test_data.zip`.

Протестируйте работу модели на проверочном датасете и вычислите точность исследуемой модели на основании метрики accuracy:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}.$$

Какова точность исследуемой модели классификации, на основании метрики *accuracy*?

Ответ: 0,98.

Задача II.3.8. (10 баллов)

Темы: задача классификации изображений, обучение модели машинного обучения, загрузка модели.

Условие

В объектив фотографа–путешественника попали разнообразные животные. Создать приложение, определяющее присутствие на фотографии животных, заданных классов или констатирующее факт, что таких животных на фотографии нет.

Для создания приложения использовать модель машинного обучения из библиотеки TensorFlow TensorFlow 2 Object Detection (http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz) и предложенный дата сет для обучения и тестирования модели (https://drive.google.com/file/d/17J-daR8GbPAF2CPYeUATyUWYvlaL1Fx-/view?usp=drive_link).

Загрузите на свой компьютер датасет. Произведите его разметку. Проанализируйте размеченный дата сет.

Какое количество классов можно выделить в предложенном дата сете?

Ответ: 5.

Задача II.3.9. (10 баллов)

Темы: задача классификации изображений, обучение модели машинного обучения, загрузка и анализ датасета.

Условие

В объектив фотографа–путешественника попали разнообразные животные. Создать приложение, определяющее присутствие на фотографии животных, заданных классов или констатирующее факт, что таких животных на фотографии нет.

Для создания приложения использовать модель машинного обучения из библиотеки TensorFlow TensorFlow 2 Object Detection (http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz) и предложенный дата сет для обучения и тестирования модели (https://drive.google.com/file/d/17J-daR8GbPAF2CPYeUATyUWYvlaL1Fx-/view?usp=drive_link).

Загрузите на свой компьютер датасет. Произведите его разметку. Проанализируйте размеченный дата сет.

Какое количество изображений в каждом классе? В ответе указажите совокупное количество объектов в каждом классе тренировочного и тестового дата сетов; ответ упорядочить через «, » по убыванию.

Пример: 7, 5, 3, 1.

Ответ: 135, 135, 135, 134, 133.

Задача II.3.10. (15 баллов)

Темы: задача классификации изображений, предобученная модель машинного обучения, тестирование модели.

Условие

Создайте программу на python, решающую следующие задачи:

- программа загружает и устанавливает библиотеку TensorFlow и TensorFlow Object Detection API;
- программа подключает модель ssd Mobilenet v2 320x320 и задает нужные параметры конфигурации для обучения модели.

Рекомендуемые параметры конфигурации следующие:

- общее количество шагов, используемых для обучения модели. Рекомендуемое значение параметра 40000 шагов;
- количество изображений, используемых на каждом этапе обучения. Для обучения модели с использованием среды Google Collab рекомендуемое значение параметра 16.

После завершения обучения модели проведите тестирование натренированной модели с использованием тестировочного дата сета.

Какое количество правильно соотнесенных с классом «deer» изображений Вы получили?

Ответ: 21.

Задача II.3.11. (20 баллов)

Темы: задача классификации изображений, предобученная модель машинного обучения, тестирование модели.

Условие

Создайте программу на python, решающую следующие задачи:

- программа загружает и устанавливает библиотеку TensorFlow и TensorFlow Object Detection API;
- программа подключает модель ssd Mobilenet v2 320x320 и задает нужные параметры конфигурации для обучения модели.

Рекомендуемые параметры конфигурации следующие:

- общее количество шагов, используемых для обучения модели. Рекомендуемое значение параметра 40000 шагов;
- количество изображений, используемых на каждом этапе обучения. Для обучения модели с использованием среды Google Collab рекомендуемое значение параметра 16.

После завершения обучения модели проведите тестирование натренированной модели с использованием тестировочного дата сета.

Какова точность исследуемой модели классификации, на основании метрики *accuracy*?

Ответ: 0,64.

Задача II.3.12. (15 баллов)

Темы: задача классификации изображений, обучение модели машинного обучения, валидация модели.

Условие

Проведите валидацию модели на валидационном дата сете. Какое количество объектов на валидационных изображения было классифицировано правильно?

Ответ: 36.

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

Технологии компьютерного зрения являются драйвером развития во многих областях деятельности человека. Они играют ключевую роль в исследованиях, дающих новые знания, с их помощью создаются новые инструменты, усовершенствующие производственные процессы. Новые продукты, базирующиеся на технологиях компьютерного зрения, оказывают огромное влияние на жизнь и деятельность людей. Используясь во множестве отраслей, она по праву входит в перечень сквозных технологий.

В этом году задача финала профиля посвящена разработке приложения на основе технологии дополненной реальности.

Дополненная реальность (augmented reality, AR) — это технология, позволяющая переносить виртуальные изображения на объекты реального мира. Технология позволяет расширить ощущения человека, получить новый пользовательский опыт.

Дополненная реальность неразрывно связана с компьютерным зрением, благодаря которому и определяются объекты в реальном мире. Они являются триггерами для отображения оверлея (дополнительный контент, накладывающийся на экран пользователя). Но и данная технология не стоит на месте, и вместо обычных экранов уже внедряются проекционные системы, благодаря которым можно размещать контент на различных объектах и поверхностях и расширять границы реальности для пользователя.

Необходимые компетенции и роли в команде в текущем году

Количество участников в команде: 4–5 человек.

Специалист по компьютерному зрению (2–3 человека: программист + DevOps-инженер):

- Понимание принципов работы компьютерного зрения, а также умение разрабатывать и применять алгоритмы обработки изображений.
- Навыки программирования. Владение языками программирования, используемыми в области компьютерного зрения, такими как Python, C++, а также знание библиотек и фреймворков, таких как OpenCV, TensorFlow, PyTorch.
- Умение разрабатывать алгоритмы, позволяющие распознавать и отслеживать объекты в реальном времени.
- Понимание принципов работы аппаратуры дополненной реальности и умение интегрировать ее с компьютерным зрением.

Программисты (2–3 человека: программист и специалист по использованию нейросетей в КЗ):

- Владение языками программирования, используемыми в разработке ПО для управления проекционной аппаратурой.
- Умение создавать интерактивные объекты и элементы игры, которые взаимодействуют с объектами в реальном мире.
- Умение создавать нейросети.

Состав задач отборочного этапа

Задания второго отборочного этапа разработаны с учетом требований к компетенциям участников. Представлены два модуля: командный и индивидуальный. Индивидуальные задания разделены на две основные части: задачи на нахождение объектов в пространстве и задачи на взаимодействие между найденными объектами. Эти задачи оцениваются автоматически, благодаря настройке соответствующих компонентов платформы Stepik. Командный модуль состоит из комплексной задачи. Задача оценивается экспертами команды разработчиков вручную.

Индивидуальные задачи

Компьютерное зрение — одна из технологий, которая находит свое применение в самых различных сферах — от медицины и промышленности до развлечений и образования. Одной из трендовых технологий, полностью основанных на компьютерном зрении, является дополненная реальность. Технология дополненной реальности открывает портал из цифрового мира в реальный. Именно ей посвящены задачи профиля в этом году.

В рамках индивидуальных задач второго этапа вам предстоит переместиться в фиджитал-мир и стать экспериментаторами, которые будут разрабатывать инструменты, позволяющие распознавать цифровые сущности, отслеживать их действия и помогать им взаимодействовать друг с другом.

Давайте приступим! Впереди нас ждет страна цифровых чудес, где нам предстоит столкнуться с самыми необычными существами и вполне обычным их поведением в реальности!

Индивидуальные задания разделены на две основные части: задачи на нахождение объектов в пространстве и задачи на взаимодействие между найденными объектами. Эти задачи оцениваются автоматически, благодаря настройке соответствующих компонентов платформы Stepik.

Каждая индивидуальная задача оценивается в 10 баллов и для решения выделено по 5 попыток.

Часть 1. Нахождение объектов в пространстве

Задача IV.1.1.1. Как найти Шарокота? (10 баллов)

Темы: компьютерное зрение.

Условие

Чтобы обнаружить объект на изображении, применяются различные методы компьютерного зрения — те самые, которые составляют основу оптической дополненной реальности.

Сейчас ведется наблюдение за Шаракотом (объектом круглой формы) в ограниченном пространстве размера 64×64 пкс. Необходимо определить положение и

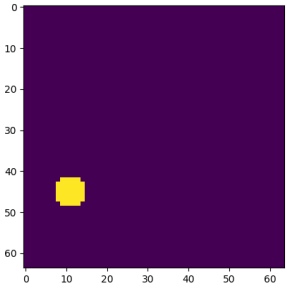
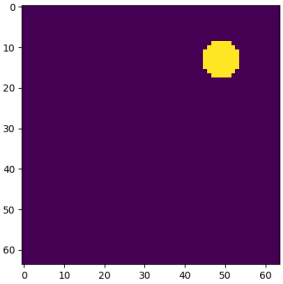
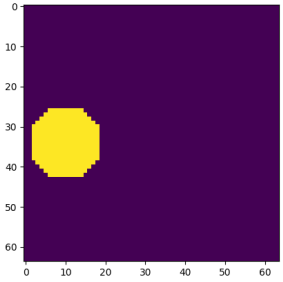
размеры объекта относительно этого пространства.

Формат входных данных

Изображение, с полем размера 64×64 и Шаракотом внутри.

Формат выходных данных

Список массивов для каждого изображения
`[координата_центра_y, координата_центра_x, размер радиуса]`.

Входные данные	Выходные данные
	`45 11 4`
 	`[13, 49, 5], [34, 10, 9]`

Примеры

Пример №1

Стандартный ввод

https://disk.yandex.ru/d/n9_wpL7dNDUDWg.

Стандартный вывод

```
[17, 28, 10], [23, 27, 7], [10, 44, 9], [37, 27, 5], [47, 23, 9], [28, 14, 9],
[41, 13, 6], [47, 17, 7], [18, 18, 8], [44, 55, 8], [44, 53, 10], [13, 22, 8],
[23, 42, 4], [17, 33, 9], [9, 18, 8], [43, 54, 9], [27, 20, 6], [57, 13, 4],
[44, 50, 7], [53, 12, 7], [26, 31, 9], [57, 45, 4], [50, 52, 5], [43, 53, 7],
[38, 53, 9], [53, 27, 6], [54, 42, 8], [49, 39, 5], [37, 41, 9], [51, 48, 6]
```

Внимание! Это задача на данные, скидывать решение (код) не нужно. В блоке ответа нужно ввести массивы с позициями и радиусом Шарокота для ваших входных данных.

Примечание: переносы строк не влияют на правильность ответа.

Генерация исходного изображения

```python
import random
import numpy as np
import matplotlib.pyplot as plt

# задаем параметры холста
H = 64
W = 64

start_img = np.zeros((H,W))

# генерируем данные для круга
R = random.randint(4,10)
x, y = (random.randint(R,H-R), random.randint(R,W-R))

cur_parametrs = [x,y,R]
X = np.arange(W).reshape(W, 1)
Y = np.arange(H).reshape(1, H)
mask = ((X - x) ** 2 + (Y - y) ** 2) < (R) ** 2
start_img[mask] = 1

plt.imshow(start_img)
plt.savefig('img.png', transparent=True)
```

Пример программы-решения

Ниже представлено решение на языке Python 3.

```python
import math
import cv2

img= cv2.imread("img.png", 1)
rr=img[59:427, 144:513]
```

```
6
7  low_red = (30,220,220)
8  high_red = (40,255,255)
9  only_r = cv2.inRange(rr, low_red, high_red)
10 x_coor =  np.mean(np.where(only_r==255)[0])
11 y_coor =  np.mean(np.where(only_r==255)[1])
12
13 k1=np.where(only_r==255)[0][0]
14 k2=np.where(only_r==255)[0][-1]
15 #коэффициент преобразования пикселей 5.78125
16 answ = [round(x_coor/5.78125), round(y_coor/5.78125),
   ↪  math.ceil((k2-k1)/5.78125/2)]
```

Задача IV.1.1.2. Траектория движения Шарокота (10 баллов)

Темы: компьютерное зрение.

Условие

Неожиданно отслеживаемый Шаракот начал перемещаться в пространстве, исследователи записали его движение на видео. Необходимо определить траекторию его движения. Известно, что объект может смещаться либо влево на 1 пкс, либо вправо на 1 пкс, либо вверх на 1 пкс, либо вниз на 1 пкс, а также не меняет своих первоначальных размеров.

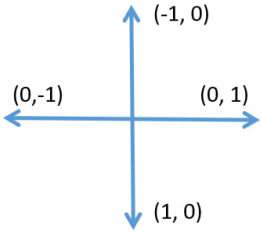
Формат входных данных

Видео в формате `mp4` с N кадрами.

Формат выходных данных

Массив размера $N-1$ — траектория движения заданная векторами $(0,\,1)$, $(0,\,-1)$, $(1,\,0)$, $(-1,\,0)$.

Направления векторов представлены ниже.



Примеры

Пример №1

Стандартный ввод

https://disk.yandex.ru/i/EVm3kxdt0AKbaA.

Стандартный вывод

```
(-1, 0), (-1, 0), (0, -1), (-1, 0), (-1, 0), (0, 1), (0, -1), (1, 0), (0, 1),
(0, -1), (1, 0),  (0, 1), (0, 1), (1, 0), (0, 1), (0, 1), (-1, 0), (0, 1),
(1, 0),  (0, 1), (1, 0), (0, -1), (0, 1), (0, -1), (1, 0), (0, 1), (0, 1),
(0, 1), (-1, 0), (0, 1), (0, 1), (-1, 0), (-1, 0), (1, 0), (0, 1), (0, 1),
(0, 1), (-1, 0), (0, -1), (0, -1), (1, 0), (1, 0), (1, 0), (1, 0), (-1, 0),
(0, -1), (1, 0), (0, 1), (0, -1).
```

Внимание! Это задача на данные, скидывать решение (код) не нужно. В блоке ответа нужно ввести массивы с позициями и радиусом Шарокота для ваших входных данных.

Примечание: переносы строк не влияют на правильность ответа.

Генерация видео

```python
import random

def generate_trajectory(R=5,N=10, H=64, W=64):
    directions = [(0,1), (0, -1), (1, 0), (-1,0)]

    trajectory = []
    images = []

    start_img = np.zeros((H,W))
    coords = (random.randint(R,H-R), random.randint(R,W-R))
    mask = draw_circle(start_img, R, coord=coords)
    start_img[mask] = 1

    images.append(start_img)

    for i in range(N-1):
        dir_id = random.randint(0,3)
        new_direction = directions[dir_id]
        new_circle = np.zeros((H,W))

        if not (R < coords[0] + new_direction[0] < H-R):
        new_direction = (-new_direction[0],new_direction[1])

        if not (R < coords[1] + new_direction[1] < W-R):
        new_direction = (new_direction[0],-new_direction[1])

        coords = (coords[0] + new_direction[0], coords[1] + new_direction[1])

        mask = draw_circle(new_circle, R, coord=coords)
        new_circle[mask] = 1
        images.append(new_circle)
        trajectory.append(new_direction)

```

```
    return images, trajectory

def draw_circle(img, R=5, coord=(10, 9)):
    width, height = img.shape[:2]
    x, y = coord
    X = np.arange(width).reshape(width, 1)
    Y = np.arange(height).reshape(1, height)
    mask = ((X - x) ** 2 + (Y - y) ** 2) < (R) ** 2
    return mask

circles, traj = generate_trajectory(R=5, N=1000)

import matplotlib.pyplot as plt
import matplotlib.cm as cm
import matplotlib.animation as animation

img = [] # some array of images
frames = [] # for storing the generated images
fig = plt.figure()
for i in range(len(circles)):
    frames.append([plt.imshow(circles[i], animated=True)])


ani = animation.ArtistAnimation(fig, frames, interval=50, blit=True,
                                repeat_delay=1000)
ani.save('movie.mp4')
```

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image, ImageChops

def find_point(diff):
    low_red = (30,220,220)
    high_red = (40,255,255)
    only_r = cv2.inRange(diff, low_red, high_red)
    plt.imshow(only_r)
    x_coor =  np.mean(np.where(only_r==255)[0])
    y_coor =  np.mean(np.where(only_r==255)[1])
    return np.array([x_coor, y_coor])

def find_coor(p1, p):
    aa = np.rint((p-p1)/6).astype("int")
    return (aa[0],aa[1])

answ_arr=[]

cap = cv2.VideoCapture('movie1.mp4')
ret, frame1 = cap.read()
point1 = find_point(frame1)


while True:
    ret, frame = cap.read()
    if not ret:
```

```
        break
    point = find_point(frame)
    answ = find_coor(point1,point)
    answ_arr.append(answ)

    point1 = point
    frame1 = frame

print(answ_arr)
cap.release()
cv2.destroyAllWindows()
```

Проверка

```
if (np.array(traj) != np.array(answ_arr)).sum() == 0:
    print("Верная траектория")
else:
    print("Пока неправильно, попробуйте еще раз")
```

Задача IV.1.1.3. Где прячутся Кубопсы? (10 баллов)

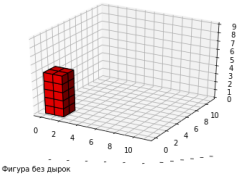
Темы: компьютерное зрение.

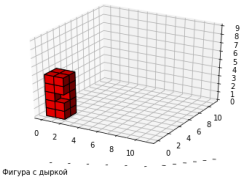
Условие

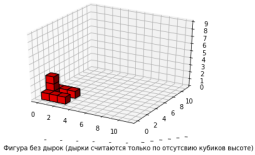
Кубопсы прячут найденные в фиджитал-мире ценные вещи в дырки, которые находят в оцифрованных объектах.

Имеется скан некоторого трехмерного объекта. Объект задан в трехмерной пространстве (9, 11, 11), где 1 соответствует детали объекта. Вам необходимо найти дырку в объекте, и в ответе указать координаты дырки.

Дырками объекта считаются пропуски деталей по координате z. Объясним проще: фигура строится из вертикальных столбиков и кубиков. Если в столбике есть промежуток, через который кубики в столбике появляются снова (кубики пропущены в вертикальном ряду), то эта дырка.







На вход вам будут даны N отсканированных объектов в виде массивов. Необходимо обнаружить, есть ли в них дырки:

- если дырок не обнаружено — вывести $[0]$;
- если обнаружены — вывести их координаты в виде массива $[z\ x\ y]$.

Ответы в виде массивов необходимо вписать в порядке обнаружения через запятую.

Если дырок несколько, отсортируйте их в порядке вывода осей и возрастания значений внутри осей.

Пример визуализации фигур и дырок в них представлены ниже.

Красным обозначены фигуры, которые даны на входе, голубым изображены дырки в фигурах.

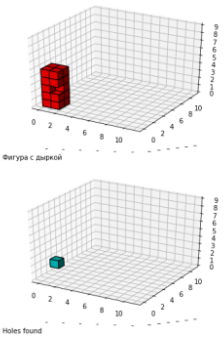


Рис. IV.1.1. а)

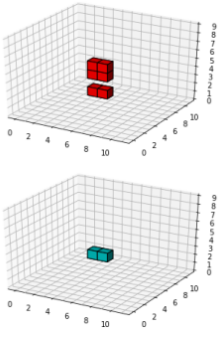


Рис. IV.1.2. б)

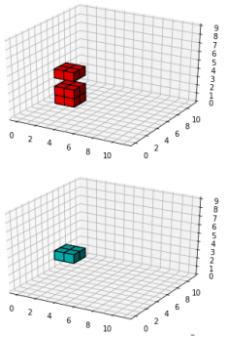


Рис. IV.1.3. в)

Ответ: а) `[[2 1 0]]`, б) `[[4 6 2], [4 7 2]]`, в) `[[4 3 1], [4 3 2], [4 4 1], [4 4 2]]`.

Внимание! Это задача на данные, скидывать решение (код) не нужно. В блоке ответа нужно ввести массивы с позициями и радиусом Шарокота для ваших входных данных.

Примечание: переносы строк не влияют на правильность ответа.

Генерация данных

```python
#генерируем N простых фигур, и N таких же фигур с дырками
def generate_shapes(N, dims):
    z, x, y = dims

    shapes = []

    for i in range(N):
        shape = np.zeros(shape=(z, x, y))

        z_up = random.randint(4,z-1)
        z_l = random.randint(0,z_up-4)

        x_up = random.randint(1,x-1)
        x_l = random.randint(0,x_up-1)

        y_up = random.randint(1,y-1)
        y_l = random.randint(0,y_up-1)

        shape[z_l:z_up, x_l:x_up, y_l:y_up] = 1

        shapes.append(shape)
```

```
22
23          shape_holes = shape.copy()
24
25          hole_z_up = random.randint(z_l+2, z_up-1)
26          hole_z_l = random.randint(z_l+1, hole_z_up-1)
27          hole_x_up = random.randint(x_l+1, x_up)
28          hole_x_l = random.randint(x_l, hole_x_up-1)
29          hole_y_up = random.randint(y_l+1, y_up)
30          hole_y_l = random.randint(y_l, hole_y_up-1)
31
32          shape_holes[hole_z_l:hole_z_up, hole_x_l:hole_x_up+1,
            ↪  hole_y_l:hole_y_up+1] = 0
33
34          shapes.append(shape_holes)
35
36      return shapes
37
38  def main():
39      z, x, y = (9, 11, 11)
40      shapes = generate_shapes(5,(z, x, y))
41      return shapes
```

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
1   # Получаем залитую фигуру(фигуру у которой нет дырок)
2   def get_filled_shape(shape):
3       lower_ends = np.argmax(shape,axis=0)
4       upper_ends = (shape.shape[0] - np.argmax(np.flip(shape,axis=0),axis=0)) %
        ↪  shape.shape[0]
5
6       filled_shape = np.zeros(shape.shape)
7
8       for x,y in zip(np.where(upper_ends)[0],np.where(upper_ends)[1]):
9           upper_b = upper_ends[x,y]
10          lower_b = lower_ends[x,y]
11          filled_shape[lower_b:upper_b,x,y] = 1
12
13      return filled_shape
14
15  # проверяем есть ли дырки, сравнивая с залитой фигурой
16  def check_holes(shape):
17      filled_shape = get_filled_shape(shape)
18      holes = filled_shape != shape
19
20      if (holes.sum()>0):
21          print("Holes found")
22          print(np.argwhere(holes))
23      else:
24          print("No holes found")
25          print("0")
26
27
28  for s in shapes:
29      check_holes(s)
```

Часть 2. Взаимодействие объектов

Вторая часть индивидуальных задач нацелена на определение взаимодействия объектов между собой и программирование их дальнейшего поведения. Эти задачи оцениваются автоматически, благодаря настройке соответствующих компонентов платформы Stepik.

Задача IV.1.2.1. Расчет угла наклона плоскости (10 баллов)

Темы: физика мяча, моделирование движения мяча.

Условие

Шарокот съедает все круглые объекты, которые будут находиться рядом. Чтобы поглощать объекты наиболее эффективно, он решил найти угол, под которым объекты будут скатываться к нему в рот с оптимальной скоростью, при которой он успеет прожевать предыдущий круглый объект. Он решил провести несколько опытов и выяснить при заданных массе и ускорении угол, под которым скатился к нему объект.

Напишите функцию, которая будет вычислять угол наклонной поверхности, по которой скатывается шар. Трение не учитывается.

На вход ваша функция должна принимать массу и ускорение шара, на выход возвращать угол в градусах, под которым расположена наклонная поверхность. В случае, если значение превышает 90°, вернуть сообщение `Error`.

Ускорение свободного падения считать равным 9,8 м/с^2.

Формат входных данных

Два числа (разделенные пробелом): целое положительное число массы m, вещественное положительное число ускорения a.

Формат выходных данных

Целое число α — угол наклона в градусах.

Примеры

Пример №1

Стандартный ввод
`8 11.2`

Стандартный вывод
`Error`

Пример №2

Стандартный ввод
5 4.0
Стандартный вывод
24

Пример №3

Стандартный ввод
5 75.0
Стандартный вывод
Error

Пример №4

Стандартный ввод
87 6.0
Стандартный вывод
37

Пример №5

Стандартный ввод
90 5.0
Стандартный вывод
30

Пример №6

Стандартный ввод
5 90.0
Стандартный вывод
Error

Пример №7

Стандартный ввод
1 1.0
Стандартный вывод
5

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
import math
def angle(mass, acceleration):
    g = 9.8
    return int(math.degrees(math.asin(acceleration/g))) if abs(acceleration) <=
    ↪  9.8 else 'Error'
```

Задача IV.1.2.2. Достижение цели (10 баллов)

Темы: физика мяча, моделирование движения мяча.

Условие

В сторону Кубопса был отправлен подарок (в форме шара), однако из-за неуклюжести Кубопса шар не всегда достигает цели.

Определите, поглотит ли Купопес шар, если известно, что он может открывать рот для поглощения шара раз в секунду. Если шар оказался у Купопса в момент, когда у того был закрыт рот, то считается, что Кубопес не смог его съесть. При решении задач рекомендуется использовать библиотеку `math`. Значения координат при расчетах в рамках задачи округляются (по математическим правилам) до двух знаков после запятой.

Для данной задачи считаем, что находимся в среде с отсутствием влияния внешних сил.

Кроме того, для упрощения задачи считается:

- первое открытие рта осуществляется в нулевую секунду;
- открытие рта не занимает времени;
- радиус шара всегда 1;
- соотношение объем шара и рта не влияет на его поглощение.

Формат входных данных

- Строка, содержащая начальные координаты шара, в формате x_0 y_0.
- Строка, содержащая координаты расположения Кубопса, x_1 y_1 (нижняя левая) и x_2 y_2 (верхняя правая точка прямоугольника).
- Вещественное положительное число v — начальная скорость движения шара.
- Вещественное отрицательное число a — ускорение шара.
- Целочисленное число α от 0 до 360 — угол направления движения шара (угол задается относительно единичной тригонометрической окружности, то есть ноль справа).

Формат выходных данных

Строка `ДА` если Кубопес съест шар, и `НЕТ` в противном случае.

Примеры

Пример №1

Стандартный ввод
5 10 -1 6 0 11 9 -30 180
Стандартный вывод
НЕТ

Пример №2

Стандартный ввод
1 2 5 6 7 8 3 -0.5 60
Стандартный вывод
НЕТ

Пример №3

Стандартный ввод
0 0 4 4 6 6 3 -0.5 45
Стандартный вывод
ДА

Пример №4

Стандартный ввод
0 0 -1 6 0 11 3 -0.1 64
Стандартный вывод
НЕТ

Пример №5

Стандартный ввод
1 2 0 1 4 5 2 -0.15 73
Стандартный вывод
ДА

Пример №6

Стандартный ввод
1 2 5 6 7 8 3 -0.5 45
Стандартный вывод
ДА

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
import math

def check(x1, y1, x2, y2, x0, y0):
    if x0 >= x1 and x0 <= x2 and y0 >= y1 and y0 <= y2:
        return True
    else:
        return False

def angle_correction(alpha):
    if alpha > 90 and alpha < 180:
        return -1, 1, 180 - alpha, alpha - 90
    if alpha > 180 and alpha < 270:
        return -1, -1, alpha - 180, 270 - alpha
    if alpha > 270:
        return 1, -1, 360 - alpha, alpha - 270
    return 1, 1, alpha, 90 - alpha

def reach_destination(v, a, alpha, x1, y1, x2, y2, x0, y0):
    xflag, yflag, alpha, beta = angle_correction(alpha)
    while v > 0:
        r = v + a/2
        x0 += round(xflag*r*math.cos(math.radians(alpha)),2)
        y0 += round(yflag*r*math.cos(math.radians(beta)),2)
        if check(x1, y1, x2, y2, x0, y0):
            return 'ДА'
        v += a
    return 'НЕТ'

```

```
x0, y0 = list(map(int, input().split()))
x1, y1, x2, y2 = list(map(int, input().split()))
v = float(input())
a = float(input())
alpha = int(input())
print(reach_destination(v, a, alpha, x1, y1, x2, y2, x0, y0))
```

Задача IV.1.2.3. Столкновение шаров (10 баллов)

Темы: физика мяча, моделирование движения мяча.

Условие

Шарокот гонялся за своим обедом, и раскидал шарики в разные стороны. Теперь он пытается отследить, куда шарики раскатились. Помогите Шарокоту понять, куда откинуло шары после через 10 с после столкновения.

Для данной задачи считаем, что шары перемещаются в среде с отсутствием влияния других сил и двигаются всегда с одной и той же скоростью. Вычисление положений шаров производится 1 раз в 1 с.

Формат входных данных

- Строка, содержащая начальные координаты расположения первого шара x_1 y_1.
- Строка, содержащая начальные координаты расположения второго шара x_2 y_2.
- Пара вещественных положительных чисел v_1 — начальная скорость движения первого шара, в формате (X, Y).
- Пара вещественных положительных чисел v_2 — начальная скорость движения второго шара, в формате (X, Y).

Формат выходных данных

2 вектора координат шаров r_1, r_2 через 10 с. Радиус обоих шаров всегда равен 1.

Полученные значения векторов указываются в `float` формате с округлением до десятых.

Например:

- 5.0;
- 4.1.

Примеры

Пример №1

Стандартный ввод
-5.5 0 5.5 0 1.0 1.0 -1.0 1.0

Стандартный вывод
[-5.5, 10.0], [5.5, 10.0]

Пример №2

Стандартный ввод
3.2 2.0 6.4 1.0 1.0 1.4 0.8 -1.0
Стандартный вывод
[13.2, 16.0], [14.4, -9.0]

Пример №3

Стандартный ввод
11.4 4.8 7 4.8 -0.3 0.0 1.0 0.0
Стандартный вывод
[18.8, 4.8], [6.6, 4.8]

Пример №4

Стандартный ввод
2 -2 8 4 0.0 1.0 -1.5 0.0
Стандартный вывод
[-0.9, 9.9], [-4.1, 2.1]

Пример №5

Стандартный ввод
0 17 12 5 1.0 -0.5 -0.2 0.7
Стандартный вывод
[8.8, 13.2], [11.2, 10.8]

Пример программы-решения

Ниже представлено решение на языке Python 3.

```python
def ifcollision(radius, v1, v2, r1, r2):
    if d <= 2 * radius:
        vx1 = (((v1[0] - v2[0]) * (r1[0] - r2[0])) / ((r1[0] - r2[0])**2)) *
        ↪  (r1[0] - r2[0])
        vy1 = (((v1[1] - v2[1]) * (r1[1] - r2[1])) / ((r1[1] - r2[1])**2)) *
        ↪  (r1[1] - r2[1])
        vx2 = (((v1[0] - v2[0]) * (r1[0] - r2[0])) / ((r2[0] - r1[0])**2)) *
        ↪  (r2[0] - r1[0])
        vy2 = (((v1[1] - v2[1]) * (r1[1] - r2[1])) / ((r2[1] - r1[1])**2)) *
        ↪  (r2[1] - r1[1])
        v1= [vx1, vy1]
        v2 = [vx2, vy2]
    return v1, v2

def new_coordinates(r1, r2, v1, v2):
    r1[0] = round(r1[0] + v1[0],2)
    r1[1] = round(r1[0] + v1[1],2)
    r2[0] = round(r1[0] + v2[0],2)
    r2[1] = round(r1[0] + v2[1],2)
    return r1, r2

#pos b1 b2
x1, y1, x2, y2 = list(map(int, input().split()))
v1 = list(map(float, input().split()))
v2 = list(map(float, input().split()))
radius = float(input())
r1 = [x1, y1]
r2 = [x2, y2]

lst_x1 = [x1]
lst_y1 = [y1]
lst_x2 = [x2]
lst_y2 = [y2]

for i in range(10):
    dx = r2[0] - r1[0]
    dy = r2[1] - r1[1]
    d = round((dx**2 + dy**2)**0.5,2)
    v1, v2 = ifcollision(radius, v1, v2, r1, r2)
    r1, r2 = new_coordinates(r1, r2, v1, v2)
```

Командные задачи

Игропрактика с проекционной дополненной реальностью

Командная задача разделена на четыре задания.

Задание 1. Необходимо научиться распознавать цвет и количество объектов, демонстрируемых в видеопоток камеры устройства.

Задание 2. Фиксировать траекторию движения, показываемого объекта. Траекторию движения отображать на экране монитора линией соответствующей цвету и форме объекта.

Задание 3. Реализовать падение объектов на экране монитора. Объекты накладываются пользователем в определенную область.

Задание 4. Используя ранее разработанные элементы программного кода, приду-

мать и реализовать мини-игру в проекционной дополненной реальности с собственной легендой и дизайном.

Все задачи оцениваются экспертами по следующим критериям.

Критерий	**Балл**
Задача 1	**12**
Различается 5 различных цветов (желтый, красный, зеленый, синий, + дополнительный любой)	3
Различаются дополнительные три цвета	1
Определяется форма круга	2
Выводится поверх видеоряда сообщение с цветом и количеством кругов	1
Сообщение с инструкцией, если нет кругов	1
Сданные видеоматериалы полностью демонстрируют реализованную часть задачи	2
Код решения содержит комментарии или прилагаемое описание работы кода	2
Задача 2	**12**
Отрисовывается траектория демонстрируемого круга (любым образом)	4
Траектория не отслеживается, если кругов в кадре несколько	1
Траектория отрисовывается размером и цветом круга	1
Траектория остается на экране, если круг исчезает	2
Сданные видеоматериалы полностью демонстрируют реализованную часть задачи	2
Код решения содержит комментарии или прилагаемое описание работы кода	2
Задача 3	**11**
Круги генерируются вне области видимости проекции и начинают падение в рандомном порядке	2
Корзина размечается (в реальности — 3, в проекции — 1)	3
Считываются все круги положенные в корзину и генерируются для падения в том же размере и цвете	2
Сданные видеоматериалы полностью демонстрируют реализованную часть задачи	2
Код решения содержит комментарии или прилагаемое описание работы кода	2
Задача 4	**15**
Мини-игра от команды должна быть реализована в дополненной реальности. Предположительно и желательно реализовывать проект для проекционной дополненной реальности. *Но так как у некоторых команд нет доступа к проектору, то на текущий момент глобального упора на проекции нет.* *Но тем не менее, взоимодействие в игре идет через реальный мир и оверлей должен отрисовываться поверх видеоряда на устройстве пользователя.*	
Легенда игры	1
Концепция дизайна (отрисованы все элементы игры в одной стилистике и привязкой к легенде)	2
Механики (до трех — 1, более трех — 2)	2

Критерий	Балл
Готовый продукт (написанный код к придуманной игре, с учетом разных сценариев использования и внедренного описанного дизайна и есть погружение пользователя в легенду; игра работает от одного запуска, а не отдельными кусочками)	5
Сданные видеоматериалы полностью демонстрируют реализованную часть задачи	2
Код решения содержит комментарии или прилагаемое описание работы кода	3

Условие каждой задачи описаны в блоках. Задачи разбиваются на основную и условно названную дополнительную части для удобства решения командой. В дополнительной части содержатся элементы из предыдущего задания или функциональные добавления, которые может выполнять другой участник команды.

Предоставление результатов

В качестве результата необходимо предоставить ссылку на архив, в котором будут:

- текстовый файл: название команды, состав участников;
- код реализации: для каждой задачи отдельная папка с соответствующим названием (`ex1`, `ex2`...), внутри которой файлы с кодом;
- видеопримеры испытаний работоспособности реализованного ПО для каждой выполненной задачи (внутри соответствующих папок).

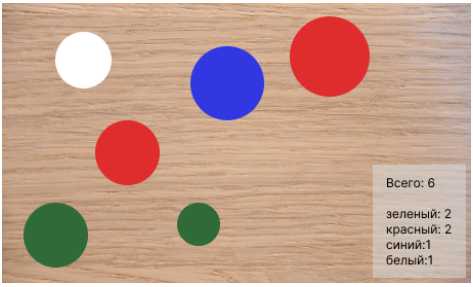
Задача IV.2.1. Обнаружение цвета (12 баллов)

Темы: компьютерное зрение.

Условие

В камеру демонстрируются объекты круглой формы. Предлагаем вам вырезать их из цветного картона (чтобы не гнулись) или размещать на столе.

Вам необходимо написать программу, которая позволяет распознавать цвет демонстрируемых в камеру объектов и их количество. Необходимо отобразить поверх видеоряда текущее значение: количество демонстрируемых объектов и их цвет.



Дополнительно: чтобы пользователь знал, что именно демонстрировать, отобразите ему сообщение на экране (поверх видеоряда) с просьбой начать демонстрирование шаров.

Решение

Примерный алгоритм решения:

1. Инициализировать видеопоток с веб-камеры.
2. Запросить пользователя демонстрировать перед камерой круглые плоские объекты и объемные мячи разного цвета.
3. Для каждого объекта определить его цвет с помощью анализа изображения с камеры.
4. Вывести количество демонстрируемых объектов и их цвета.

Примечание: можно использовать функции OpenCV для обработки изображений, такие как `cv2.inRange()` для определения цветового диапазона и `cv2.findContours()` для обнаружения объектов.

Пример программы-решения

Ниже представлено решение на языке Python 3.

Вариант 1 (для плоских объектов)

```
import cv2
import numpy as np

# Инициализация видеопотока с веб-камеры
cap = cv2.VideoCapture(0)

def get_color():
    ret, frame = cap.read()
    return np.mean(frame, axis=(0,1))

try:
    # Запросить пользователя демонстрировать плоский объект перед камерой
    print("Пожалуйста, демонстрируйте плоские объекты разного цвета перед
    ↪  камерой.")
    target_color = np.round(get_color())
    print(f"Цвет объекта: {target_color}")

    while True:
        # Захватить кадр
        ret, frame = cap.read()

        # Определить бинарное изображение, содержащее только цвет объекта
        lower_bound = np.array(target_color - 30, dtype=np.uint8)
        upper_bound = np.array(target_color + 30, dtype=np.uint8)
        mask = cv2.inRange(frame, lower_bound, upper_bound)

        # Поиск контуров объекта
        contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL,
        ↪  cv2.CHAIN_APPROX_SIMPLE)

        # Отрисовка контура объекта на кадре
```

```
        cv2.drawContours(frame, contours, -1, (0, 255, 0), 2)

        # Отображение кадра с результатами
        cv2.imshow("Frame", frame)

        # Обработка нажатия клавиши 'q' для выхода из программы
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

except Exception as e:
    print(f"Произошла ошибка: {e}")

finally:
    # Закрыть видеопоток и окна OpenCV
    cap.release()
    cv2.destroyAllWindows()
```

Вариант 2 (для объемных объектов)

```
import cv2
import random
import time

def find_color(hsv,lower,upper):
    mask=cv2.inRange(hsv,lower,upper)
    mask=cv2.erode(mask,None,iterations=5)
    mask=cv2.dilate(mask,None,iterations=5)
    contours=cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
    ↪  cv2.CHAIN_APPROX_SIMPLE)[0]
    if len(contours)>0:
        contour=max(contours,key=cv2.contourArea)
        (x,y),radius=cv2.minEnclosingCircle(contour)
        return True, x
    else:
        return False,-1

colors={"orange":[10,30],"green":[50,70],"blue":[90,110]}
sequence=list(colors)
# random.shuffle(sequence) //для рандома последовательности

cap = cv2.VideoCapture("http://192.168.1.139:8080/video")
cv2.namedWindow("Camera", cv2.WINDOW_NORMAL)

latest = time.perf_counter()

while cap.isOpened():
    ret,frame=cap.read()
    blurred=cv2.GaussianBlur(frame,(11,11),0)
    hsv=cv2.cvtColor(blurred,cv2.COLOR_BGR2HSV)

    count=0
    result={}
    for color in colors:
        result[color]=find_color(hsv, (colors[color][0],100,100),
        ↪  (colors[color][1],255,255))
        count+=result[color][0]
    if count==len(colors):
        cseq=[[key,result[key][1]] for key in result]
        # cseq=sorted(cseq,key=lambda item: item[1])[::-1] //для вебки
        cseq=sorted(cseq,key=lambda item: item[1])
        cseq=[item[0] for item in cseq]
```

```
            if sequence==cseq:
                cv2.putText(frame,"You win!", (10,200), cv2.FONT_HERSHEY_SIMPLEX, 2.7,
                ↪  (0,0,255), 3)
            else:
                cv2.putText(frame,"You're wrong!", (10,200), cv2.FONT_HERSHEY_SIMPLEX,
                ↪  2.7, (0,0,255), 3)
        elif count>0 and count<len(colors) or count>len(colors):
            cv2.putText(frame,"You're wrong!", (10,200), cv2.FONT_HERSHEY_SIMPLEX,
            ↪  2.7, (0,0,255), 3)

        cv2.imshow("Camera", frame)
        cv2.resizeWindow("Camera", 600, 400)

        key=cv2.waitKey(1)
        if key==ord("q"):
            break

cap.release()
cv2.destroyAllWindows()
```

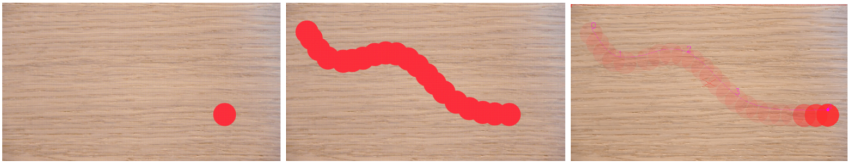
Задача IV.2.2. Траектория движения цветного объекта (12 баллов)

Темы: компьютерное зрение.

Условие

В видеопоток камеры демонстрируется объект (предлагаем использовать тот же картонный круг, что и для первой задачи).

В этой задаче пользователь начинает медленно двигать объект перед камерой. Вам необходимо написать программное обеспечение, которое определит траекторию ее движения. Траектория движения отрисовывается на экране монитора, и при устранении объекта из поля зрения камеры траектория остается. Таким образом, пользователь как бы рисует картинку, которая остается на экране.



Поясняющие условия:

- для решения данной задачи предусматривается демонстрирование только одного объекта в камеру;
- камера должна быть зафиксирована.

Дополнительно: траектория движения отрисовывается на экране монитора соответствующим демонстрируемому объекту цветом и формой.

Решение

Примерный алгоритм решения:

1. Инициализировать видеопоток с веб-камеры.
2. Запросить пользователя начать медленно двигать объект перед камерой.
3. Определить цвет демонстрируемого объекта с помощью анализа кадров с камеры.
4. Отследить движение объекта, сохраняя координаты его центра на каждом кадре.
5. Отрисовать траекторию движения объекта на экране монитора с соответствующим цветом.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
import cv2
import numpy as np

# Инициализация видеопотока с веб-камеры
cap = cv2.VideoCapture(0)

# Запросить пользователя начать двигать объект
input("Нажмите Enter, чтобы начать двигать объект перед камерой...")

# Инициализировать переменные для отслеживания траектории
trajectory = []
color = None

try:
    while True:
        # Захватить кадр
        ret, frame = cap.read()

        # Определить цвет объекта
        if color is None:
            color = np.mean(frame, axis=(0, 1))

        # Определить бинарное изображение, содержащее только цвет объекта
        lower_bound = np.array(color - 30, dtype=np.uint8)
        upper_bound = np.array(color + 30, dtype=np.uint8)
        mask = cv2.inRange(frame, lower_bound, upper_bound)

        # Поиск контуров объекта
        contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL,
        ↪  cv2.CHAIN_APPROX_SIMPLE)

        if contours:
            # Найти центр объекта
            M = cv2.moments(contours[0])
            cX = int(M["m10"] / M["m00"])
            cY = int(M["m01"] / M["m00"])

            # Добавить координаты центра в траекторию
            trajectory.append((cX, cY))
```

```
39
40              # Отрисовать траекторию
41              if len(trajectory) > 1:
42                  for i in range(1, len(trajectory)):
43                      cv2.line(frame, trajectory[i-1], trajectory[i],
                        ↪  color.tolist(), 2)
44
45          # Отображение кадра с результатами
46          cv2.imshow("Frame", frame)
47
48          # Обработка нажатия клавиши 'q' для выхода из программы
49          if cv2.waitKey(1) & 0xFF == ord('q'):
50              break
51
52  except Exception as e:
53      print(f"Произошла ошибка: {e}")
54
55  finally:
56      # Закрыть видеопоток и окна OpenCV
57      cap.release()
58      cv2.destroyAllWindows()
```

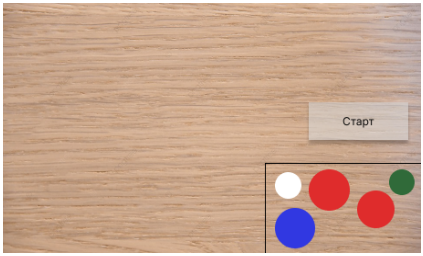
Задача IV.2.3. Цифровые объекты на цветной траектории (11 баллов)

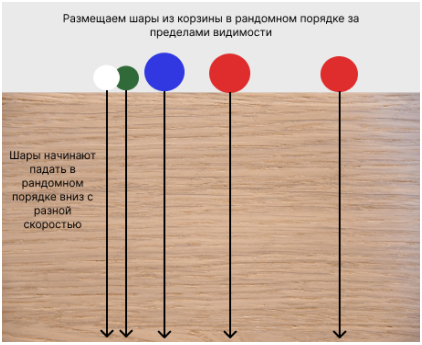
Темы: компьютерное зрение.

Условие

В рабочую область, на которую направлена камера, размещается корзина (например, квадратной формы). В нее пользователь может положить N объектов круглой формы разного цвета (предлагаем использовать тот же картонный круг, что и для других задач). Пользователь кладет круги так, чтобы каждый был виден, без наложения друг на друга.

Кроме корзины на экране пользователя размещена кнопка «Пуск», при нажатии которой сверху в рандомных точках по горизонтали начинают падать шары. Шаров должно быть ровно столько, сколько было в корзине на момент нажатия кнопки, и они должны быть окрашены в соответствующие цвета. Порядок падения шаров неважен.





Упрощенная форма реализации: предзаданное программно количество шаров разного цвета.

Решение

Примерный алгоритм решения:

1. Инициализировать видеопоток с веб-камеры.
2. Определить цвет движущегося объекта 1.
3. Подсчитать количество виртуальных объектов круглой формы (n) на видео.
4. Задать цвета виртуальных объектов круглой формы (для примера, сгенерировать случайные цвета).
5. Отследить движение объекта 1 и его траекторию.
6. Отследить положение статичных объектов 2 и объектов n.
7. Реализовать логику отскоков виртуальных объектов от траектории объекта 1.
8. Отобразить все объекты и траекторию на кадре.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
import cv2
import numpy as np

# Инициализация видеопотока с веб-камеры
cap = cv2.VideoCapture(0)

# Определить цвет движущегося объекта 1
def get_color():
    ret, frame = cap.read()
    return np.mean(frame, axis=(0,1))

try:
    # Определить цвет движущегося объекта 1
    print("Нажмите Enter, когда будете готовы определить цвет объекта 1...")
    input()
    target_color = np.round(get_color())
    print(f"Цвет объекта 1: {target_color}")

    while True:
        # Захватить кадр
        ret, frame = cap.read()

        # Подсчитать количество виртуальных объектов круглой формы (n)
        n = 5 # Замените это на действительное количество объектов

        # Задать цвета виртуальных объектов круглой формы (в данном случае,
        ↪  случайные цвета)
        virtual_object_colors = [np.random.randint(0, 255, size=3) for _ in
        ↪  range(n)]

        # Отображение всех объектов

        # 1. Отобразить движущийся объект 1 и его траекторию
```

```
        # TODO: Ваш код для отображения объекта 1 и траектории

        # 2. Отобразить статичные объекты 2
        # TODO: Ваш код для отображения объектов 2

        # 3. Отобразить виртуальные объекты n
        for i, color in enumerate(virtual_object_colors):
            cv2.circle(frame, (50 * (i+1), 50), 20, color.tolist(), -1)

        # Отображение кадра с результатами
        cv2.imshow("Frame", frame)

        # Обработка нажатия клавиши 'q' для выхода из программы
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

except Exception as e:
    print(f"Произошла ошибка: {e}")

finally:
    # Закрыть видеопоток и окна OpenCV
    cap.release()
    cv2.destroyAllWindows()
```

Задача IV.2.4. Мини-игры (15 баллов)

Темы: компьютерное зрение.

Условие

Используя разработанные элементы программного кода из заданий отборочного этапа, предложите мини-игру в проекционной дополненной реальности с собственной легендой и дизайном. Реализуйте ее код.

Легенду и дизайн нужно оформить в текстовый документ.

Критерии оценки:

- легенда(история) — 1 балл;
- концепция дизайна — 2 балла;
- механика — 2 балла;
- готовый продукт — 5 баллов.

Решение

Используя разработанные элементы программного кода из заданий отборочного этапа, предложите мини-игру в проекционной дополненной реальности с собственной легендой и дизайном. Реализуйте ее код.

Название игры: «Цветные пузырьки»

Легенда

Вам предстоит погрузиться в захватывающий мир цветных пузырьков! В этой проекционной дополненной реальности на экране вы увидите движущийся объект, который задает траекторию движения цветных пузырьков. Ваша задача — ловить

эти пузырьки, соответствующие цветам других объектов, отскакивающих от траектории. Каждый пойманный пузырек приносит вам очки, но будьте осторожны, не пропустите ни одного!

Правила игры

1. Вам предстоит поймать как можно больше пузырьков, соответствующих цветам виртуальных объектов, отскакивающих от траектории объекта 1.
2. У вас есть ограниченное время для пойманных пузырьков.
3. За каждый пойманный пузырек вы получаете очки.
4. Пропущенные пузырьки уменьшают ваш счет.
5. Игра заканчивается, когда заканчивается отведенное время.

Элементы игры

1. Движущийся объект 1, задающий траекторию.
2. Статичные объекты 2 и n, задающие цвета пузырьков.
3. Пузырьки разных цветов, отскакивающие от траектории объекта 1.
4. Счетчик времени и счетчик очков.

Алгоритм решения

Для создания этой мини-игры нужно интегрировать коды, представленные в предыдущих задачах; добавить логику для отслеживания столкновений между пузырьками и траекторией, а также для подсчета очков. Кроме того, потребуется добавить интерфейс с отображением счетчика времени и счетчика очков на экране.

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/_XV2M_1vP_rSRg.

Задача VI.1.1.1. Поделить пирог (20 баллов)

Условие

Двое делят между собой круглый пирог из N кусков-секторов (см. рисунок). Каждый может взять один или два **соседних** (прилегающих друг к другу) куска за раз. После взятия оставшиеся куски не сдвигаются и между ними остается пустое пространство.

Выигрывает тот, что заберет последний кусок. Считая, что каждый из игроков действует, исходя из оптимальной для него стратегии, определите, кто выиграет в каждом случае при заданном числе кусков в пироге от 1 до 8. В качестве ответа введите число из 8 цифр (без пробелов), каждый разряд в котором означает выигравшего игрока: цифра 1 — выиграет первый игрок, 2 — второй, 0 — нет однозначного ответа.

Решение

Выигрывает тот, кто забирает последний кусок. Инициатива почти всегда в руках второго игрока.

При 1–2 кусках выигрывает первый игрок, так как второй не сможет ходить после первого хода. Игра сводится к тому, чтобы привести расположение к ситуации, когда

остается 1–2 куска (расположенных вместе или раздельно), при этом второй игрок при числе 3 и более кусков имеет преимущество. При 3 выигрывает второй игрок, так как любой ход первого игрока сводит к ситуации обратной выше. Это оптимальная стратегия.

Ответ: 11222222.

Задача VI.1.1.2. День рождения программиста (15 баллов)

Условие

Программист Петя родился в день программиста 13.09.2009. Он решил проверить, сколько раз нужно прибавить 256 дней к этой дате, чтобы опять получился день программиста, то есть 256-й день в году. В каком году это случится? Учитывайте, что в високосном году день программиста 12 сентября. При расчете используйте каноническое определение високосного года в григорианском календаре: он делится на 4, но не делится на 100, кроме тех случаев, когда делится на 400.

Решение

Решать задачу можно несколькими способами. Удобно использовать языки программирования, где доступен календарь. Например, в Windows Powershell такой расчет выполняется простым циклом (для високосного года нужно провести аналогичные вычисления и определить, какая дата наступит раньше).

```
$startDate = Get-Date "13.09.2009"
$targetDate = $startDate.AddDays(256)
while($targetDate.Month -ne 9 -or $targetDate.Day -ne 13) {
     $targetDate = $targetDate.AddDays(256)
}
Write-Output "Programmer's birthday $targetDate"
```

Более общий вариант с вычислением дня, с учетом високосного года на Python без использования календаря.

```
year = 2009; day_of_year = 256

while True:
  day_of_year += 256
  year_days = 365 + ((year % 4 == 0 and year % 100 != 0) or (year % 400 == 0))
  if (day_of_year >= year_days):
    year += 1
    day_of_year %= year_days
  if day_of_year == 256:
    print(year)
    break
```

Если доступны электронные таблицы (например, Excel), можно формулами вычислить на много лет вперед вычислить даты с шагом 256 дней с последующим поиском нужной даты (12 или 13 сентября).

Ответ: 2241.

Задача VI.1.1.3. Команда разработчиков (20 баллов)

Условие

Группа поклонников настольных игр собирает команду для разработки мобильной игры. Сразу изъявили желание участвовать N товарищей, а руководитель проекта оценил сложность работы для каждого из них в виде коэффициента C_i (целое число). Считая общий объем работ по разработке равным единице, доля проекта выполненная каждым участником будет равна $1/C_i$. Для успешной разработки игры в отведенное время сумма по $1/C_i$ для всех участников должна быть $\geqslant 1$.

Оценив свои возможности, участники выяснили, что их не хватит для завершения работы в срок, и запросили помощи у другой команды. Сложность работы для каждого участника которой обозначим D_j.

Вам требуется выбрать из этой дополнительной команды минимальное число участников, чтобы уложиться вовремя. Число баллов зависит от оптимальности решения (решения в вещественных или целых числах).

Формат входных данных

В первой строке записано два целых числа N и M ($1 \leqslant N, M \leqslant 10$) — количество членов основной и дополнительной команды.

Во второй строке N целых чисел C_i ($1 \leqslant C_i \leqslant 20$). В третьей строке записаны M чисел D_j ($1 \leqslant D_j \leqslant 20$).

Формат выходных данных

Если команда справится с проектом вовремя и без дополнительных участников, выведите 0.

Если даже после найма всей дополнительной команды времени не хватит, выведите -1.

Иначе выведите минимальное количество участников дополнительной команды, необходимых для завершения разработки в срок.

Примеры

Пример №1

Стандартный ввод
5 6 11 11 8 12 11 11 9 11 10 10 11

Стандартный вывод
6

Пример №2

Стандартный ввод
2 4 3 3 4 4 3 5

Стандартный вывод
1

Решение

Сначала рассчитаем имеющуюся рабочую силу, но так как у вещественных чисел есть погрешность, лучше завести две длинных целых (`long long int`) переменных, одна из которых будет отвечать за числитель дроби, вторая — за знаменатель. Просуммируйте в них всю силу основной команды, и если ее хватает, то ответ 0. После каждой операции сложения дробей сокращайте ее с помощью функции быстрого нахождения наибольшего общего делителя. Если рабочей силы не хватает, отсортируйте массив силы дополнительных работников по возрастанию (так как это знаменатели, то сильнейший работник окажется с наименьшим числом) и добирайте по одному, пока силы не хватит, либо пока не закончатся работники. Если дополнительные работники исчерпаны, а итоговая дробь меньше 1, верните результат -1.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
def gcd(a, b):
    if b == 0:
        return a
    return gcd(b, a % b)

def add_ratio(numer, denom, denom_new):
    numer = numer * denom_new + denom
    denom = denom * denom_new
    common = gcd(numer, denom)
    numer //= common
    denom //= common
    return numer, denom

def solve():
    C, D = map(int, input().split())
    c = list(map(int, input().split()))
    d = list(map(int, input().split()))

    numer = 1
    denom = c[0]

    for i in range(1, C):
        numer, denom = add_ratio(numer, denom, c[i])

    if numer >= denom:
        print(0)
        return
```

```
28
29      d.sort()
30      for i in range(D):
31          numer, denom = add_ratio(numer, denom, d[i])
32          if numer >= denom:
33              print(i + 1)
34              return
35
36      print(-1)
37
38  solve()
```

Ниже представлено решение на языке C++.

```
1   #include <bits/stdc++.h>
2
3   typedef long long int ll;
4
5   using namespace std;
6
7   ll gcd(ll x, ll y) {
8     if (y == 0) return x;
9     return gcd(y, x % y);
10  }
11
12  void simplify(ll& numer, ll& denom) {
13    ll tmp = gcd(numer, denom);
14    numer /= tmp;
15    denom /= tmp;
16  }
17
18  int main()
19  {
20    ios_base::sync_with_stdio(0);
21    cin.tie(0);
22    cout.tie(0);
23
24    int n, m;
25    cin >> n >> m;
26    vector <ll> c(n);
27    vector <ll> d(m);
28    for (int i = 0; i < n; i++) {
29      cin >> c[i];
30    }
31    for (int i = 0; i < m; i++) {
32      cin >> d[i];
33    }
34    sort(d.begin(), d.end());
35
36    ll numer = 1;
37    ll denom = c[0];
38    for (int i = 1; i < n; i++) {
39      numer = numer * c[i] + denom;
40      denom *= c[i];
41      simplify(numer, denom);
42    }
43    if (numer >= denom) {
44      cout << "0\n";
45      return 0;
46    }
```

```
47
48    for (int i = 0; i < m; i++) {
49      numer = numer * d[i] + denom;
50      denom *= d[i];
51      if (numer >= denom) {
52        cout << (i + 1) << "\n";
53        return 0;
54      }
55    }
56
57    cout << "-1\n";
58    return 0;
59  }
```

Задача VI.1.1.4. Стрельба по мишеням (20 баллов)

Условие

Группа спортсменов-биатлонистов стреляет по мишеням. Как известно, для стрельбы стоя диаметр мишени равен 115 мм, а для стрельбы лежа — 45 мм, пуля имеет круглое сечение диаметром 5,6 мм. Центр мишени находится в точке $(0, 0)$. Вам даны координаты X, Y центров попадания пуль относительно центра мишени: 5 для положения стоя и 5 для положения лежа.

Напишите программу для определения числа промахов для каждого положения спортсмена. Попадание засчитывается только в том случае, когда круг пули полностью находится внутри круга мишени. Число баллов зависит от оптимальности решения по памяти и времени выполнения.

Формат входных данных

На вход подается 10 строк: первые 5 строк описывают попадание пули в стенд с мишенью при стрельбе лежа, а последующие 5 — при стрельбе стоя.

В каждой строке записано по два неотрицательных целых числа — смещение пули в мкм (микрометрах) по горизонтали и вертикали от центра мишени соответственно. Все числа целые и не превосходят 10^6.

Формат выходных данных

Выведите два целых числа в отдельных строках. В первой количество промахов при стрельбе лежа, а во второй — при стрельбе стоя.

Примеры

Пример №1

Стандартный ввод
0 0 123 123 39400 0 0 39401 30000 30000 30000 30000 0 39401 87654 12345 123123 321312 109400 1
Стандартный вывод
3 3

Пример №2

Стандартный ввод
0 0 123 123 19700 0 0 19701 15000 15000 30000 30000 0 39401 54700 0 123123 321312 54700 1
Стандартный вывод
2 2

Решение

Задача решается по теореме Пифагора — считается гипотенуза от точки попадания до центра окружности (нуля), и если она меньше радиуса мишени, то пуля попала в цель. Но перед расчетами нужно вычесть из радиуса мишени радиус сечения пули. Вместо операции извлечения корня по формуле, рекомендуется возводить в квадрат противоположную часть уравнения (радиус получившейся мишени), чтобы избежать использования вещественных чисел. Особое внимание стоит уделить пограничным значениям при тестировании решения.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```python
import math
n = 5 # number of lines
radius_small = float( 45 * 1000)/2
radius_big = float(115 * 1000)/2
radius_bullet = float(5600)/2

misses_small = 0
for i in range (n):
  (x, y) = list(map(int, input().strip().split()))
  if (math.hypot(x,y) + radius_bullet > radius_small):
    misses_small += 1

misses_big = 0
for i in range (n):
  (x, y) = list(map(int, input().strip().split()))
  if (math.hypot(x,y) + radius_bullet > radius_big):
    misses_big += 1

print(misses_small)
print(misses_big)
```

Задача VI.1.1.5. Телефонные разговоры (25 баллов)

Условие

На автоматической телефонной станции абоненты постоянно звонят друг другу и для каждого разговора известно время начала и окончания (в секундах). Ваша программа должная выяснить, в какой момент было наибольшее число одновременных разговоров. Если таких моментов несколько — сообщите самый ранний из них. Гарантируется, в момент завершения разговора не начинается другой разговор. Число баллов зависит от оптимальности решения по памяти и времени выполнения.

Формат входных данных

Первая строка содержит число разговоров (целое $\leqslant 10000$), в последующих строках сообщены время начала и окончания разговора (целые числа, разделенные пробелом). Числа по модулю не превосходят 10^9.

Формат выходных данных

Выведите целое число — искомый момент времени

Примеры

Пример №1

Стандартный ввод
3 3 4 1 6 0 7

Стандартный вывод
3

Пример №2

Стандартный ввод
5 4 5 0 3 1 9 7 8 2 6

Стандартный вывод
2

Решение

Задача подразумевает более чем одно решение, но одним из оптимальных вариантов будет ведение массива пар длиной 2^N. Каждая такая пара будет хранить значение времени и информацию о том, начало это звонка, или конец. Массив сортируется по возрастанию, после чего по нему можно сделать один обход, чтобы понять, в какой момент времени была наибольшая нагрузка — увеличивайте переменную каждый раз, когда попадаете на начало звонка, и уменьшайте, когда на конец звонка. Отдельно храните индекс предыдущего наибольшего по нагрузке момента (достав момент времени из просматриваемой пары). Сложность такого решения оценивается как $O(N \cdot \log(N))$, она также не будет зависеть от разницы по времени между самими звонками.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
from functools import cmp_to_key

def compare(pair1, pair2):
    if pair1[0] == pair2[0]:
        # if time is equal
        # return CALL_END first (which is 'False')
        # to avoid overlap
        if pair1[1] == pair2[1]:
            return 0
        if pair1[1] <  pair2[1]:
            return -1
        return 1
    # if time is not equal, return earlier first
    if pair1[0] <  pair2[0]:
        return -1
    return 1

n = int(input())
# memory reserve for list of 2N pairs
#   pair structure:
```

```
#       [TIME, IS_THIS_A_CALL_START]
#   for example, if input was '4 5',
#   following pairs will be inserted:
#       [4, True]
#       [5, False]
ar = [[None, None]] * (n * 2)
for i in range(n):
    a, b = map(int, input().split())
    ar[i * 2    ] = [a, True ]
    ar[i * 2 + 1] = [b, False]
ar = sorted(ar, key=cmp_to_key(compare))

calls_maximum = 0
calls_currently = 0
peak_time = -2000000001
for i in range(2 * n):
    if ar[i][1]:
        calls_currently += 1
    else:
        calls_currently -= 1
    if calls_currently > calls_maximum:
        calls_currently = calls_maximum
        peak_time = ar[i][0]
print(peak_time)
```

Математика. 8–9 классы

Задача VI.1.2.1. Про испорченные ячейки памяти (10 баллов)

Условие

Тестировщик памяти простейшего устройства запрограммирован таким образом, что он считывает с каждой ячейки памяти присвоенный ей порядковый номер, и в конце работы выдает полученную сумму.

Пусть ячейки памяти пронумерованы последовательно от 1 до N, и если ячейка памяти испорчена, то тестировщик не сможет считать ее номер, следовательно, не прибавит его в общую сумму.

Известно, что в результате механического повреждения были выведены из строя две последовательные ячейки памяти. Нужно узнать, какие, если тестировщик выдал сумму, равную 15000.

Критерии оценивания

- Верно найдено одно из решений задачи — 5 баллов.
- Верно найдены оба решения задачи — 10 баллов.

Решение

Сумма целых чисел от 1 до N равна $\dfrac{N(N+1)}{2}$.

Пусть повреждены были ячейки с номерами x и $x-1$.

Имеем уравнение с двумя неизвестными:

$$\frac{N(N+1)}{2} - x - (x-1) = 15000.$$

Так как в общей сумме число $x + x - 1$ мало, решим вначале уравнение

$$\frac{N(N+1)}{2} \approx 15000.$$

Тогда $N(N+1) = N^2 + N \approx 30000$.

Из этих двух слагаемых большая часть представлена N^2. Поэтому решим уравнение $N^2 \approx 30000$. Ближайшее целое решение этого уравнения с округлением вниз, это 173.

Действительно, $173^2 = 29929$. Тогда

$$N^2 + N = 30102,$$

$$\frac{N(N+1)}{2} = 15051,$$

следовательно, $15051 - x - (x-1) = 15000$.

Отсюда $2x - 1 = 51$, $x = 26, x - 1 = 25$. То есть могут быть уничтожены ячейки памяти с номерами 25 и 26.

Однако если взять ближайшее целое решение уравнения $N^2 = 30000$ с округлением вверх, это 174.

Действительно, $174^2 = 30276$.

Тогда

$$N^2 + N = 30450,$$

$$\frac{N(N+1)}{2} = 15225,$$

следовательно, $15225 - x - (x - 1) = 15000$.

Отсюда $2x - 1 = 225$, $x = 113, x - 1 = 112$.

То есть могут быть уничтожены ячейки памяти с номерами 112 и 113.

Другие варианты невозможны.

Действительно, если $N = 175$, то $175^2 = 30625$.

Тогда

$$N^2 + N = 30800,$$

$$\frac{N(N+1)}{2} = 15400,$$

следовательно, $15400 - x - (x - 1) = 15000$.

Отсюда $2x - 1 = 400$, $x = 200,5$, что, мало того, что число дробное, так еще и больше 175 — максимального номера ячейки.

А если $N = 172$, то $172^2 = 29584$. Тогда

$$N^2 + N = 29756,$$

$$\frac{N(N+1)}{2} = 14878,$$

следовательно, $14878 - x - (x - 1) = 15000$.

Отсюда $2x - 1 = -122$, что не может быть, т. к. номер ячейки не может быть отрицательным.

Ответ: могут быть уничтожены ячейки памяти с номерами 25 и 26, если всего 173 ячейки памяти, или 112 и 113, если всего 174 ячейки памяти.

Задача VI.1.2.2. Про робота и дворецкого (20 баллов)

Условие

Человекоподобный робот Алекс был взят в качестве помощника дворецкого Александра. Однажды робот заметил, как дворецкий разбавляет хозяйский гранатовый сок в бочке: он вылил из бочки 3 стакана сока и добавил 3 стакана воды. Даже со своим супермощным компьютерным зрением робот Алекс не может на глаз определить, каково теперь процентное содержание сока в бочке, но алгоритмы, встроенные в его нейросеть, позволяют ему это рассчитать.

Во второй день дворецкий проделал с разбавленным соком то же самое. И на третий день — то же. После чего, по подсчетам Алекса, в бочке осталось 50% сока и 50% воды. И когда хозяин за обедом сказал, пригубляя гранатовый сок: «Стаканчик этого сока точно поправит мое здоровье», робот, запрограммированный на то, чтобы сообщать хозяину о вещах, угрожающих его здоровью, не выдержал и сказал: «Должен информировать вас, господин, о том, что в вашем стакане только 50% сока».

Роботу Алексу это известно, а вы можете рассчитать, сколько стаканов сока было в бочке изначально?

Примечание: количество стаканов не обязательно было целое.

Критерии оценивания

- Верно составлено уравнение для нахождения решения, но решено неверно — 10 баллов.
- Приведен правильный ответ, который полностью математически обоснован — 20 баллов.

Решение

Обозначим за x — первоначальное количество стаканов сока в бочке и составим таблицу, отражающую количество стаканов сока и его концентрацию в бочке на первый, второй и третий день махинаций с бочонком гранатового сока нечестного дворецкого.

День	**Сок (ст.)**	**Концентрация сока (доля ст. сока в смеси)**
0	x	1=100%
1	$x-3$	$\frac{x-3}{x}=1-\frac{3}{x}$
2	$x-3-3(1-3/x)=$ $=x-6+9/x$	$\frac{x-6+9/x}{x}=(1-\frac{3}{x})^2$
3	$x-6+9/x-3(1-3/x)^2$	$(1-\frac{3}{x})^3$

Теперь приравниваем долю стаканов сока в смеси на третий день к $1/2$, и находим первоначальное количество стаканов сока:

$$(1-\frac{3}{x})^3=\frac{1}{2};\ 1-\frac{3}{x}=\frac{1}{\sqrt[3]{2}};\ x=\frac{3\sqrt[3]{2}}{\sqrt[3]{2}-1}\approx 14,54.$$

Ответ: 14,54 стакана.

Задача VI.1.2.3. Про покемонов (30 баллов)

Условие

В игре *PokemonGo* есть три разных покемона, каждый из которых появляется через определенный промежуток времени: A — каждые 2 часа, B — каждые 4 часа, а C — каждые 6 часов, но какова разница между появлениями этих персонажей — не известно.

Найти, какой из покемонов вероятнее всего появится первым после того, как вы вошли в игру в случайный момент времени, и вероятность наступления этого события в течение первых двух часов вашего участия в игре.

Критерии оценивания

- Верно описаны четыре возможных случая, но вероятности интересующих нас событий в этих случаях, и вероятности самих случаев, найдены не верно или не найдены — 5 баллов.
- Верно описаны четыре возможных случая, верно найдены либо вероятности интересующих нас событий в этих случаях, либо вероятности самих случаев — 15 баллов.
- Верно описаны четыре возможных случая, верно найдены вероятности интересующих нас событий в этих случаях и вероятности самих случаев — 25 баллов.
- Приведен правильный ответ, который полностью математически обоснован — 30 баллов.

Решение

Вероятность появления персонажа A в течение двух часов подряд равна 1, так как он появляется 1 раз в два часа, соответственно, вероятность появления персонажа B в течение двух часов подряд — 1/2, а персонажа C — 1/3.

В таблицу запишем четыре случая, которые могут произойти в течение ближайших двух часов.

Рассчитаем вероятность возникновения каждого случая. Найдем условную вероятность того, что мы увидим первым покемона A в этот период, при условии, что возникнет данный случай. Эта вероятность обозначается как $P(A/I)$, если возникнет первый случай, $P(A/II)$, если возникнет второй случай, $P(A/III)$, если возникнет третий случай, и $P(A/IV)$, если возникнет четвертый случай. Аналогично обозначаются условные вероятности появления первыми покемонов B и C.

Сл	Событие	Случая	$P_{\frac{A}{I-IV}}$	$P_{\frac{B}{I-IV}}$	$P_{\frac{C}{I-IV}}$
I	Появятся и A, и B, и C.	$1 \cdot \frac{1}{2} \cdot \frac{1}{3} = \frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
II	Появятся A и B, но не C.	$1 \cdot \frac{1}{2} \cdot \frac{2}{3} = \frac{1}{3}$	$\frac{1}{2}$	$\frac{1}{2}$	0
III	Появятся A и C, но не B.	$1 \cdot \frac{1}{2} \cdot \frac{1}{3} = \frac{1}{6}$	$\frac{1}{2}$	0	$\frac{1}{2}$
IV	Появится только A, B и C —нет.	$1 \cdot \frac{1}{2} \cdot \frac{2}{3} = \frac{1}{3}$	1	0	0

Таким образом, полная вероятность события A — что мы увидим покемона A

первым в течение двух часов

$$P(A) = P(I) \cdot P(A/I) + P(II) \cdot P(A/II) + P(III) \cdot P(A/III) + P(IV) \cdot P(A/IV) =$$

$$= \frac{1}{6} \cdot \frac{1}{3} + \frac{1}{3} \cdot \frac{1}{2} + \frac{1}{6} \cdot \frac{1}{2} + \frac{1}{3} \cdot 1 = \frac{23}{36} \approx 63,9\%.$$

Аналогично, полная вероятность события B — что мы увидим покемона B первым в течение двух часов

$$P(B) = P(I) \cdot P(B/I) + P(II) \cdot P(B/II) + P(III) \cdot P(B/III) + P(IV) \cdot P(B/IV) =$$

$$= \frac{1}{6} \cdot \frac{1}{3} + \frac{1}{3} \cdot \frac{1}{2} + \frac{1}{6} \cdot 0 + \frac{1}{3} \cdot 0 = \frac{2}{9} \approx 22,2\%.$$

И, наконец, полная вероятность события C — что мы увидим покемона C первым в течение двух часов

$$P(C) = P(I) \cdot P(C/I) + P(II) \cdot P(C/II) + P(III) \cdot P(C/III) + P(IV) \cdot P(C/IV) =$$

$$= \frac{1}{6} \cdot \frac{1}{3} + \frac{1}{3} \cdot 0 + \frac{1}{6} \cdot \frac{1}{2} + \frac{1}{3} \cdot 0 = \frac{5}{36} \approx 13,9\%.$$

Ответ: вероятнее всего первым мы увидим покемона A, вероятность этого события $\frac{23}{36} \approx 63,9\%$.

Задача VI.1.2.4. Про логотип компании (40 баллов)

Условие

Логотип компании «Дополненная реальность» состоит из двух квадратов со стороной $a = 12$. Один из квадратов со сторонами, параллельными и перпендикулярными горизонту, изображает объективную реальность. А второй квадрат имеет с первым общую точку (левый нижний угол), но повернут относительно линии горизонта на 30° влево. Он изображает дополненную реальность.

Найти:

1. их общую площадь;
2. площадь всей фигуры без общей площади;
3. геометрическое место точек, описываемых нижним правым углом квадрата со стороной a при всех возможных углах поворота этого квадрата относительно линии горизонта.

Критерии оценивания

- Верно сделан чертеж — 5 баллов.
- Приведен правильный ответ пункта 1, который не полностью математически обоснован, либо математические рассуждения, которые привели к ответу на пункт 1 верны, но в силу арифметической ошибки ответ неверен — 15 баллов.
- Приведен правильный ответ пункта 1, который полностью математически обоснован — 30 баллов.

- Приведен правильный ответ пунктов 1, 2 которые полностью математически обоснованы — 35 баллов.
- Приведен правильный ответ пунктов 1, 2, 3, которые полностью математически обоснованы — 40 баллов.

Решение

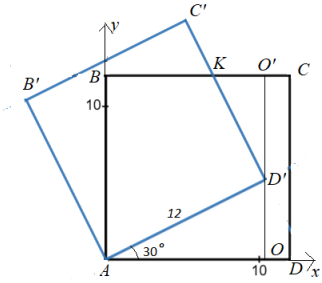


Рис. VI.1.1. Обозначения и дополнительные построения

Поместим квадраты в декартову систему координат так, чтобы общая точка совпала с началом координат, и первый квадрат лежал сторонами на осях (см. рис. VI.1.1).

Тогда координаты угловых точек этого квадрата будут соответственно $A(0,0)$, $B(0,12)$, $D(12,0)$, $C(12,12)$.

Найдем координаты точки D'.

Из прямоугольного треугольника AOD' имеем: $\sin 30^\circ = \frac{D'O}{AD'}$, $\cos 30^\circ = \frac{AO}{AD'}$, следовательно, $D'O = 12\sin 30^\circ = 6$, $AO = 12\cos 30^\circ = 6\sqrt{3}$.

Зафиксируем координаты точки $D'(6\sqrt{3}, 6)$.

Построим уравнение прямой AD'. Эта прямая проходит через начало координат, и тангенс угла ее наклона к положительному направлению оси x : $\operatorname{tg} 30^\circ = \frac{1}{\sqrt{3}}$.

Таким образом, уравнение прямой $AD' : y = \frac{1}{\sqrt{3}}x$.

Построим уравнение прямой $D'C'$.

Эта прямая проходит через точку D', и она перпендикулярна прямой AD', т. е. тангенс угла ее наклона к положительному направлению оси x равен $-\frac{1}{\operatorname{tg} 30^\circ} = -\sqrt{3}$.

Таким образом, уравнение прямой $D'C'$: $y-6 = -\sqrt{3}(x-6\sqrt{3})$ или $y = -\sqrt{3}x+24$.

Найдем точку пересечения прямых BC и $D'C'$ как решение системы уравнений:

$$\begin{cases} y = 12, \\ y = -\sqrt{3}x + 24. \end{cases}$$

Получим, что координаты точки $K(\frac{12}{\sqrt{3}}, 12)$.

Теперь найдем искомые площади:

1. общая площадь

$$S_{ABKD'} = S_{ABO_1O} - S_{\triangle AD'O} - S_{\triangle O'KD'} = 12 \cdot 6\sqrt{3} - \frac{1}{2} \cdot 6 \cdot 6\sqrt{3} - \frac{1}{2} \cdot 6 \cdot 2\sqrt{3} = 48\sqrt{3};$$

2. площадь всей фигуры без общей площади

$$12 \cdot 12 + 12 \cdot 12 - 48\sqrt{3} = 288 - 48\sqrt{3};$$

3. геометрическое место точек, описываемых точкой D' при разных углах наклона стороны AD' к положительному направлению оси x, — это окружность радиуса $a = 12$. Это следует из координат точки $D'(a\cos\alpha, a\sin\alpha)$. Если мы подставим эту точку в уравнение окружности радиуса a с центром в точке $(0, 0)$: $x^2 + y^2 = a^2$, то получим тождество.

Ответ:

1. $48\sqrt{3}$;
2. $288 - 48\sqrt{3}$;
3. окружность радиуса a.

Математика. 10–11 классы

Задача VI.1.3.1. Про волшебный кувшин гнома (10 баллов)

Условие

Для того чтобы перейти на следующий уровень в игре, нужно отгадать загадку гнома. У гнома есть волшебный кувшин, который увеличивает количество золота, положенного в него, но увеличивает неравномерно. Чем больше в него положишь, тем больше возьмешь. Также известно, что если положить сколько-то монет в пустой кувшин, потом вынуть все деньги, которые «даст» кувшинчик, положить их все туда снова и опять вынуть все монеты из сосуда, то первоначальная сумма утроится. Гном на ваших глазах положил в пустой кувшин 13 монет. Если всю сумму, которую он вынет в первый раз, положить в кувшинчик снова, то получим 39 монет. Сколько монет он вынет в первый раз?

Критерии оценивания

- Найдено верное решение, но оно не полностью обосновано, или нащупан алгоритм для получения результатов работы кувшинчика, но ответ подсчитан неверно — 5 баллов.
- Найдено верное решение, и оно полностью обосновано — 10 баллов.

Решение

Представим действие кувшинчика как функцию f.

Пусть $f(13) = x$, тогда $f(x) = 3 \cdot 13 = 39$. При этом известно, что $f(x)$ — возрастающая функция, т. е. если $x_2 > x_1$, то $f(x_2) > f(x_1)$, и аргументы и значения этой функции всегда натуральные числа.

Попробуем найти некоторые значения функции. Пусть $f(1) = p$ нам пока неизвестно, зато известно, что $p > 1$ и $f(p) = 3 \cdot 1 = 3 > p$. Но между 1 и 3 только одно целое значение — это 2. Следовательно, $f(1) = 2$ и $f(2) = 3$. Идем дальше: пусть $f(3) = q > 3$, и $f(q) = 9$, а $f(9) > 9$. Если $f(8) = 9$, то $f(3) = 8$, и мы не сможем заполнить таблицу значений для $f(4), f(5), f(6), f(7)$ (см. таблицу VI.1.1).

Таблица VI.1.1: Неправильные значения функции

Значения функции	**Значения функции**
$f(1) = 2$	$f(6) =$
$f(2) = 3$	$f(7) =$
$f(3) = 8$	$f(8) = 9$
$f(4) =$	$f(9) =$
$f(5) =$	$f(10) =$

Аналогично, если $f(7) = 9$, то $f(3) = 7$, и мы снова не сможем заполнить таблицу значений для $f(4), f(5), f(6)$ разными целыми числами по возрастанию, так как между 7 и 9 только одно целое число — 8. А вот случай $f(6) = 9$ и $f(3) = 6$ подходит. Тогда $f(4) = 7, f(5) = 8$ (см. таблицу 2). Тогда становится известно, что $f(7) = 3 \cdot 4 = 12, f(8) = 3 \cdot 5 = 15, f(9) = 3 \cdot 6 = 18$. После чего становится ясно, что $f(12) = 3 \cdot 7 = 21, f(15) = 3 \cdot 8 = 24$.

Таблица VI.1.2: Правильные значения функции

Значения функции	**Значения функции**	**Значения функции**
$f(1) = 2$	$f(6) = 9$	$f(11) =$
$f(2) = 3$	$f(7) = 12$	$f(12) = 21$
$f(3) = 6$	$f(8) = 15$	$f(13) =$
$f(4) = 7$	$f(9) = 18$	$f(14) =$
$f(5) = 8$	$f(10) =$	$f(15) = 24$

Посмотрев в таблицу VI.1.2, мы теперь видим, что между $f(12) = 3 \cdot 7 = 21$ и $f(15) = 3 \cdot 8 = 24$ есть только два целых значения для функции: 22 и 23, следовательно, $f(13) = 22$, а $f(14) = 23$.

Аналогично получим, что $f(10) = 19$, а $f(11) = 20$.

Ответ: 22 монеты.

Задача VI.1.3.2. Про кибербаскетбол (20 баллов)

Условие

Для попадания в команду олимпиады по киберспорту игроку в баскетбол необходимо не только хорошо играть на реальном поле, но и уметь завоевывать очки в киберигре. Пусть в начале сезона тренировок процент удачных трехочковых бросков в кибербаскетболе у игрока был ниже 75%, а к концу сезона стал выше 75%. Был ли период в этом сезоне, когда у игрока процент удачных бросков был ровно 75%? То есть вопрос состоит в том, могло ли быть, что такого периода не было?

Например, если в начале сезона из двух сделанных бросков игрок попал один раз, то его процент удачных попаданий на тот момент составлял $1/2 = 50\%$. Допустим, в следующий бросок он попал, т. е. его процент попаданий вырос до $2/3 = 67\%$. Но $50 < 60 < 67$, то есть в этом случае у игрока не было периода, когда его процент попаданий был ровно 60%. Могло ли это случиться с 75%-ым барьером?

И второй вопрос: для каких процентов попаданий в общем невозможно пропустить момент, когда таких попаданий было ровно столько процентов?

Критерии оценивания

- Доказано, что невозможно пропустить 75%-й барьер — 10 баллов.
- Доказано, что невозможно пропустить 75%-й барьер. Найдены еще значения, которые невозможно пропустить, но это не доказано — 15 баллов.
- Доказано, что невозможно пропустить 75%-й барьер. Найдены все остальные значения, которые невозможно пропустить, и это доказано — 20 баллов.

Решение

1. Докажем, что нельзя пропустить 75%-й барьер, то есть если в начале сезона тренировок процент удачных трехочковых бросков в кибербаскетболе у игрока был ниже 75%, а к концу сезона он стал выше 75%, то обязательно в какой-то момент времени у игрока было ровно 75% попаданий.

Проведем доказательство от противного.

Допустим, что после n бросков процент удачных трехочковых бросков в кибербаскетболе у игрока был ниже 75%, а после $(n+1)$-го броска он стал выше 75%.

Допустим, что после n бросков игрок попал $S(n)$ раз. Тогда его процент попаданий составил $\frac{S(n)}{n}$. Пусть после следующего броска процент его попаданий вырос и составил $\frac{S(n+1)}{n+1}$, но так как процент попаданий вырос, то $S(n+1) = S(n) + 1$.

Получаем два неравенства:

$$\frac{S(n)}{n} < \frac{3}{4},$$
$$\frac{S(n+1)}{n+1} > \frac{3}{4}.$$

Второе неравенство перепишем как $\frac{S(n)+1}{n+1} > \frac{3}{4}$.

Решая неравенства, получим $4S(n) < 3n$ и $4S(n) + 4 > 3n + 3$, что означает, что $4S(n) + 1 > 3n$.

Получим двойное неравенство: $4S(n) < 3n < 4S(n) + 1$, которое означает, что между двумя последовательными целыми числами находится целое число $3n$, что невозможно. Полученное противоречие доказывает, что обязательно в какой-то момент времени у игрока было ровно 75% попаданий.

2. Докажем также, что в общем случае невозможно пропустить моменты, когда процент попаданий будет $\frac{k-1}{k}$.

Пусть

$$\frac{S(n)}{n} < \frac{k-1}{k} < \frac{S(n)+1}{n+1}.$$

Получим двойное неравенство: $kS(n) < (k-1)n < kS(n)+1$, которое означает, что между двумя последовательными целыми числами находится целое число $(k-1)n$, что невозможно. Полученное противоречие доказывает, что обязательно в какой-то момент времени у игрока было ровно $\frac{k-1}{k}\%$ попаданий.

Целые проценты получатся для следующих из этих значений:

$$\frac{1}{2} = 50\%,\ \frac{3}{4} = 75\%,\ \frac{4}{5} = 80\%,\ \frac{9}{10} = 90\%,$$

$$\frac{19}{20} = 95\%,\ \frac{24}{25} = 96\%,\ \frac{49}{50} = 98\%,\ \frac{99}{100} = 99\%.$$

Ответ:

1. Нельзя пропустить 75%-ый барьер.
2. Невозможно пропустить моменты, когда процент попаданий будет $\frac{k-1}{k}\%$.

Задача VI.1.3.3. Про логотип компании (30 баллов)

Условие

Логотип компании «Компьютерное зрение» состоит из стилизованного глаза, верхнее веко которого описывается колоколообразной кривой $y = e^{-x^2}$, нижнее — ее отражением относительно оси x : $y = -e^{-x^2}$. А радужкой глаза является вписанный в эти веки круг с центром в начале координат (см. рис. VI.1.2). Найти радиус «радужки глаза».

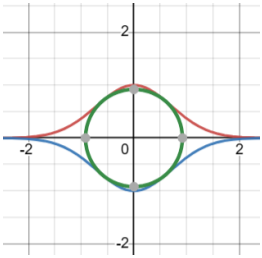


Рис. VI.1.2. Рисунок к задаче VI.1.3.3

Критерии оценивания

- Привели верное уравнение для нахождения радиуса, но ошиблись в расчетах — 10 баллов.
- Верно найден радиус, но не до конца обосновано решение, например, не обосновано, что он является минимумом функции расстояния между точками кривой и центром круга — 15 баллов.
- Решение найдено верно и полностью обосновано — 30 баллов.

Решение

Максимальное значение кривой $y = e^{-x^2}$ достигается при $x = 0$ и равно 1. Следовательно, радиус круга $0 < R < 1$.

Найдем его как наименьшее расстояние между точками на кривой $y = e^{-x^2}$ или $(y = -e^{-x^2})$ и центром окружности (0,0):

$$R = \sqrt{(x-0)^2 + (\pm e^{-x^2} - 0)^2} = \sqrt{x^2 + e^{-2x^2}} \to \min .$$

Так как расстояние больше нуля, корень можно убрать.

Найдем минимум функции $f(x) = x^2 + e^{-2x^2}$ при $0 < x < 1$.

Для этого найдем производную и приравняем ее к нулю: $f'(x) = 2x - 4xe^{-2x^2} = 0$, или $2x(1 - 2e^{-2x^2}) = 0$.

Отсюда либо $x = 0$, что не подходит, либо $e^{-2x^2} = 1/2$. Потенцируем:

$$ln(e^{-2x^2}) = ln(1/2).$$

Отсюда $-2x^2 = ln1 - ln2 = -ln2$, и $x^2 = \frac{ln2}{2}$. Получаем, что $x = \sqrt{ln\sqrt{2}}$.

Проверим, действительно ли в этой точке достигается минимум функции, проследив изменение знака производной на числовой оси (см. рис. VI.1.3),

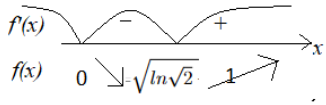


Рис. VI.1.3

Для примера подставим в производную значение $x = 1 : 2(1 - \frac{2}{e^2}) > 0$. Получили, что при переходе через точку $x = \sqrt{ln\sqrt{2}}$ производная функции меняет знак с минуса на плюс, следовательно, сама функция имеет в этой точке минимум.

Подставим так найденное значение x^2 в формулу для R :

$$R = \sqrt{\frac{ln2}{2} + e^{-ln2}} = \sqrt{\frac{ln2}{2} + e^{ln(1/2)}} = \sqrt{\frac{ln2}{2} + \frac{1}{2}} \approx 0,92.$$

Ответ: $R = \sqrt{\frac{ln2}{2} + \frac{1}{2}} = \sqrt{ln\sqrt{2} + 0,5} \approx 0,92.$

Задача VI.1.3.4. Про точки на окружности и сфере (40 баллов)

Условие

В компьютерной игре на первом уровне игры планета представляет собой круглый плоский диск, и обитаем только край этого диска. Игроки выбирают для обитания три точки произвольным образом, и они соединяются в треугольник. Найти вероятность того, что этот треугольник содержит центр диска.

На втором уровне этой игры планета представляет собой уже шар, и обитаема его поверхность. Игроки выбирают четыре точки на поверхности планеты произвольным образом, и они соединяются в тетраэдр. Найти вероятность, что этот тетраэдр содержит в себе центр планеты.

Критерии оценивания

- Сделан верный чертеж двумерного случая — 5 баллов.
- Сделан верный чертеж для двумерного случая, найдена верная вероятность для двумерного случая, но не обоснована — 10 баллов.
- Сделан верный чертеж двумерного случая, найдена верная вероятность для двумерного случая, но не до конца обоснована — 15 баллов.
- Двумерный случай полностью разобран, решение найдено верно и обосновано — 20 баллов.
- Двумерный случай полностью разобран, решение найдено верно и обосновано. Сделан верный чертеж трехмерного случая — 25 баллов.
- Двумерный случай полностью разобран, решение найдено верно и обосновано. Сделан верный чертеж трехмерного случая, найдена верная вероятность для трехмерного случая, но не до конца обоснована — 35 баллов.
- Найдено верное и полностью обоснованное решение двумерного и трехмерного случая — 40 баллов.

Решение

Зафиксируем две точки на окружности P_1 и P_2, и проведем от них прямые через центр окружности.

Пусть на противоположной стороне окружности эти прямые заканчиваются точками P_1' и P_2', соответственно (см. рис. VI.1.4).

Тогда $\triangle P_1P_2P_3$ будет содержать центр круга тогда и только тогда, когда точка P_3 будет находиться на дуге $P_1'P_2'$.

Длина этой дуги зависит от длины дуги P_1P_2. Если P_1 и P_2 удалены друг от друга на четверть длины окружности, то вероятность формирования такого треугольника равна $1/4$. Если P_1 и P_2 удалены друг от друга на половину длины окружности, то вероятность формирования такого треугольника равна $1/2$. Если P_1 и P_2 находятся очень близко друг к другу, то вероятность формирования такого треугольника стремится к нулю. Так как любое взаимное расположение этих точек одинаково возможно, то есть P_1 и P_2 могут быть удалены друг от друга на любое расстояние от 0 до половины длины окружности, то искомая вероятность равна среднему из всех возможных значений от 0 до $1/2$, то есть $1/4$.

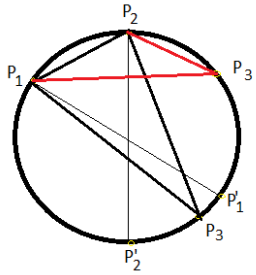


Рис. VI.1.4

Попробуем немного другой подход, чтобы перейти к трехмерному случаю.

Вместо того чтобы выбирать местоположение трех точек на окружности, сформулируем проблему так. Зафиксируем точку P_3 на окружности. Выберем отрезки P_1P_1' и P_2P_2', проходящие через центр круга, и чьи концы находятся на окружности. Нужно найти, при каком наименовании концов этих отрезков, $\triangle P_1P_2P_3$ будет содержать центр круга.

Очевидно, что это будет тогда, когда обе точки P_1 и P_2 будут находится на противоположной половине круга от точки P_3.

Например, на рис. VI.1.5 в трех случаях из четырех треугольники не содержат точку центра круга: $\triangle P_1'P_2P_3$, $\triangle P_1P_2'P_3$, $\triangle P_1'P_2'P_3$, и только $\triangle P_1P_2P_3$ содержит.

Искомая вероятность равна $1/4$.

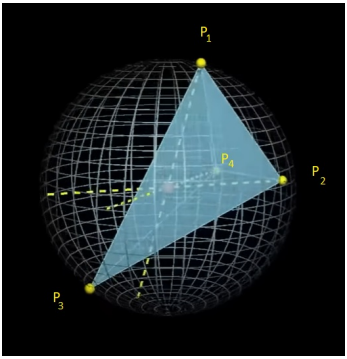


Рис. VI.1.5

Перейдем на трехмерный случай (см. рис. VI.1.6).

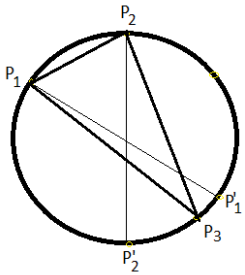


Рис. VI.1.6

Выберем одну точку на сфере: P_3. Выберем отрезки P_2P_2', P_1P_1', и P_4P_4', проходящие через центр шара, и чьи концы находятся на сфере.

Нужно найти, при каком наименовании концов этих отрезков, тетраэдр $P_1P_2P_3P_4$ будет содержать центр шара.

Очевидно, что это будет тогда, когда все три точки: P_2, P_1 и P_4, будут находится на противоположной половине шара от точки P_3.

Так как вариантов различного переименования концов у каждого отрезка 2, а отрезков — три, всех возможных комбинаций наименования будет $2^3 = 8$. Из них только одна комбинация подходящая.

Следовательно, искомая вероятность равна $1/8$.

Ответ: для первого уровня — $1/4 = 0,25$; для второго уровня — $1/8 = 0,125$.

Инженерный тур

Общая информация

В 2024 году впервые прошли соревнования по профилю Технологии компьютерного зрения и цифровые сервисы. Появление профиля НТО, посвященного компьютерному зрению, на площадке НИУ ВШЭ Нижний Новгород является неслучайным. В 2023 году, когда профиль был заявлен, исполнилось 25 лет всемирно известной библиотеке компьютерного зрения `OpenCV`, которая зародилась в Нижнем Новгороде. Участники разработки инициировали реализацию профиля как эстафету передачи знаний и навыков в исследовании и разработке инструментов компьютерного зрения молодому поколению.

В 2023–2024 году профиль посвящен разработке на основе библиотеки `OpenCV` цифрового сервиса, представляющего собой интерактивную демонстрационно-игровую проекционную систему. Эта система на основе библиотеки является увлекательным и инновационным способом использования технологий компьютерного зрения для создания интерактивного опыта для пользователей. Участники финала реализовывали игровой комплекс. В игре «Мячик-метатель» игрок становится участником захватывающего испытания, бросая мячик по виртуальным объектам, проецируемым на стену. Цель игры состоит в том, чтобы попасть мячиком в объекты, например, воздушные шары, что приводит к их исчезновению и начислению очков. Участники финала отслеживают как цели-проекции, так и коллизию — реальный мяч-проекцию.

Вторая задача состоит в разработке игры, в которой игрокам предстоит загнать как можно больше виртуальных мячиков в подвижную проекцию корзины. Для достижения этой цели они могут использовать возможность рисовать линии на проецируемой поверхности. Виртуальные мячи, выпущенные игроками, взаимодействуют с этими маркерными линиями: скатываются по ним и отскакивают, изменяя свое направление. Игроки распределяют линии таким образом, чтобы максимальное количество мячиков попало в корзину, требуя от них не только точности и меткости, но и умения строить эффективные траектории. Успех зависит от быстроты реакции и умения предвидеть движение мячиков, создавая захватывающий и динамичный игровой опыт. В данной задаче участники финала не только используют инструменты компьютерного зрения, но моделируют поведение проекций виртуальных объектов при их коллизии с маркерными линиями.

Игры, подобные описанным, могут быть удивительно полезны в образовательных целях:

- Развитие логического мышления — выстраивание стратегии и необходимость находить оптимальные методы способствуют развитию логического мышления и умению принимать решения.
- Улучшение координации — взаимодействие с виртуальными объектами, требующее точности и меткости, помогает улучшить координацию движений.
- Практика математики и физики — при вычислении траектории мячей и прогнозировании их движения можно внедрить математические и физические концепции, такие как геометрия, кинематика и законы сохранения энергии.

- Развитие командной работы — многие игры могут быть сыграны в группах, поощряя командную работу, взаимодействие и коммуникацию между участниками.
- Стимулирование творческого мышления — создание стратегий для достижения цели в игре требует творческого мышления и поощряет экспериментирование с различными подходами.
- Практика пространственного мышления — взаимодействие с виртуальными объектами на трехмерной проекции способствует развитию пространственного мышления и пониманию трехмерных пространств.

Требования к команде и компетенциям участников

Количество участников в команде: 3–5 человек.

Компетенции, которыми должны обладать члены команды:

- Ответственный за калибровку — исправляет полученное с камеры изображение с учетом искажений, работает с маркерами для задания границ изображения.
- Ответственный за нахождение и отслеживание объектов — реализует функции нахождения необходимых объектов и их отслеживания.
- Ответственный за улучшение изображения — повышает качество изображения: устраняет шумы и осуществляет цветовую коррекцию.
- Специалист по механикам — задает логику генерации объектов, реализовывает механики, объединяет все модули в игровой сервис.

Оборудование и программное обеспечение

Наименование	**Описание**
Название и ссылка (если есть).	Для чего используется в задаче.
Компьютер или ноутбук (базовые характеристики: объем RAM — 8 ГБ, процессор — i3-7020U @ 2.30 GHz × 4, SSD — 256 ГБ).	Для разработки программных компонентов, проведения вычислений.
Веб-камера (базовые характеристики:FPS — 30, разрешение — 720p).	Для получения взаимодействий с проецируемым изображением.
Проектор (базовые характеристики: разрешение — 1920 × 1080, яркость — 1200 лм).	Для проецирования изображения с компьютера на поверхность.
Редактор кода (например, VSCode (https://code.visualstudio.com/), Sublime text (https://www.sublimetext.com/) или другие аналоги).	Для разработки программных компонентов.
OpenCV (https://opencv.org/).	Python и C++ — библиотека с реализованными методами компьютерного зрения.
NumPy (https://numpy.org/).	Python-библиотека для более быстрых и удобных вычислений (по сравнению со встроенными средствами языка).
Pygame (https://www.pygame.org/).	Python-библиотека для разработки игр.
Pymunk (https://www.Pymunk.org/).	Python-библиотека для расчета физики, результаты которой можно использовать в Pygame.

Описание задачи

Инженерный тур разбит на два этапа, каждый из которых представляет собой отдельную независимую игру (некоторые наработки, например, обработка видеопотока, можно перенести из первой во вторую):

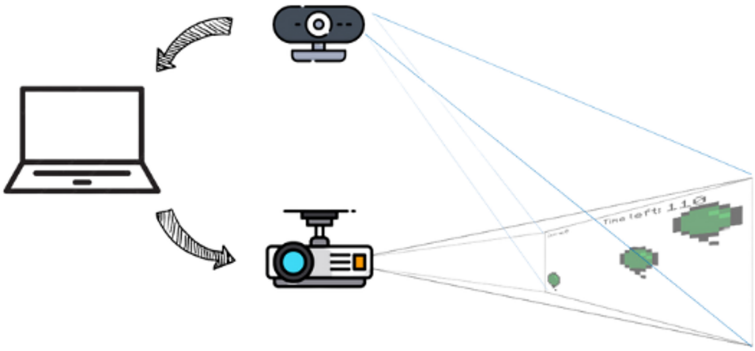
1. Охота на объекты в мирах — игровой сервис на основе механики метания пластмассовыми шариками. Например, шутер, где в роли патронов выступают выдаваемые пластмассовые шарики, а цели представляют собой летящие по экрану объекты.
2. Распознание сущностей — игровой сервис на основе механики рисования на доске. Пользователь рисует маркером фигуры, остальные объекты на них реагируют. Реализация на основе библиотеки `Pymunk`.

На выходе каждого из этапов участники должны получить конечный продукт, работающий в разных условиях освещенности, расположения камеры, а также корректно определяющий необходимые объекты.

Подробнее о каждом этапе ниже.

Этап 1. Игра «Охота на объекты в мирах»

На данном этапе участники решают задачу реализации игры «Охота на объекты в мирах», в рамках которой игроки должны с помощью реальных объектов (пластмассовых шариков четырех цветов) попадать по проецируемым ектобъам. После соударения (коллизии) реального объекта с проекцией последняя распадается на части, и объект возвращается в цифровой мир.



Например, шутер, где в роли снарядов выступают пластмассовые шарики, а цели — летящие по экрану объекты, объекты — это воздушные шары (на картинке это воздушные шары).

Рис. VI.2.1. Общая схема задачи первого этапа

Для достижения цели участникам необходимо решить следующие подзадачи:

1. Генерация шариков. Может быть как случайной, так и зависящей от действий игрока, наличии в каком-то месте других шариков и т. д.
2. Поведение шариков. Они могут как просто появляться в различных частях экрана, так и двигаться в одном направлении, менять его в зависимости от различных условий и т. д.
3. Нахождение пластмассовых шариков. В этот пункт входит также и отделение реальных объектов от проецируемых, если в этом есть необходимость.
4. Разделение пластмассовых шариков по четырем цветам: зеленому, синему, желтому и красному. Данный пункт и предыдущий могут идти в любом порядке, однако выполнение одного из них не означает выполнение другого.
5. Калибровка изображения. Считанные с камеры объекты должны корректно переводиться в нужные координаты вне зависимости от расположения камеры.
6. Создание начального экрана и экрана окончания игры. На этих экранах допускается взаимодействие с игрой посредством клавиатуры и мыши (не через проекцию).
7. Объединение различных модулей в одну программу. Программа не должна требовать запуска дополнительных файлов пользователем в течение работы.
8. Разработка игровых механик. Они могут быть связаны с подсчетом очков, использованием различных цветов мячей, дополнительными игровыми объектами и т. д.
9. Улучшение изображения. Как уже было сказано ранее, сюда входят устранение шумов, цветокоррекция и работа с бликами.
10. Определение момента столкновения шарика с проекцией. Это нужно, чтобы пролетающий мимо проекции мячик не попадал по всем шарикам на своем пути.

Дополнительные подзадачи:

1. Разработка собственных маркеров для взаимодействия с игрой. Это могут быть как и определенные объекты, на которые программа реагирует особым образом, так и собственные ArUco-маркеры.
2. Создание таблицы рекордов. Данные должны передаваться между запусками

игры через текстовый файл или базу данных.

3. Динамическое изменение размеров поля во время игры. С помощью передвижения маркеров меняется размер игровой области.

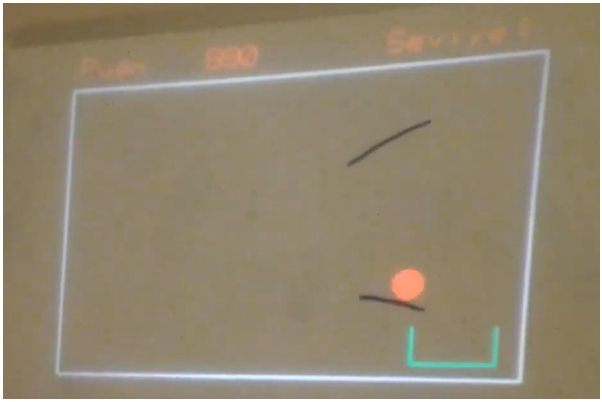
Возможные проблемы:

- Слишком узкий/широкий диапазон цветов — нужные цвета не считываются/считываются ненужные объекты.
- Круги вообще не распознаются или большинство объектов распознается как круги.
- В распознавание включаются нарисованные на фоне объекты.
- Проблемы в коммуникации между программными модулями: разницы в записи координат и сложности в объединении программных компонентов, если они разрабатывались независимо.

Данная задача не связана со следующей, однако такие моменты, как коррекция изображения, генерация объектов и таблица рекордов могут быть использованы на следующем этапе.

Этап 2 Игра «Распознавание сущностей»

Предложенная в рамках второго этапа инженерного тура для финалистов задача «Распознавание сущностей» подразумевает разработку игры, в ходе которой игроки должны направлять проецируемые мячики в корзину посредством рисования на площади отображения проекции маркерами направляющих линий. Для рисования используются маркеры. Проецируемые мячики отталкиваются или скатываются по линиям. Задача пользователя: используя такой инструмент взаимодействия с проекцией, загнать мячик в корзину (корзина может быть подвижной).



Для упрощения реализации физических законов участникам предлагалось использовать библиотеку `Pymunk`.

Рис. VI.2.2. Общая схема задачи второго этапа

На данном этапе можно выделить следующие подзадачи:

1. Генерация мячей. Может быть случайной или зависящей от действий игрока.
2. Программирование поведения корзины. Она может быть зафиксирована в одной точке, может передвигаться или менять местоположение на случайное в зависимости от различных условий.
3. Реализация физики мяча (отскоки, гравитация). Принимается как собственная, так и реализованная на основе `Pymunk`.
4. Регистрация линий. Программа должна корректно считывать нарисованные маркером линии и преобразовывать их в игровые объекты.
5. Реализация системы очков. В решение данной подзадачи входит описание логики присуждения очков и создание таблицы рекордов.
6. Создание начального экрана и экрана окончания игры. На экранах допускается взаимодействие с игрой посредством клавиатуры и мыши (не через проекцию).
7. Объединение различных модулей в одну программу. Программа не должна требовать запуска дополнительных файлов пользователем в течение работы.
8. Улучшение изображения: устранение шумов, цветокоррекция и работа с бликами.
9. Калибровка изображения. Считанные с камеры объекты должны корректно переводиться в нужные координаты вне зависимости от расположения камеры.

Дополнительные подзадачи:

1. Учет перекрытия проекции. Например, прошедший между проекцией и камерой человек не должен распознаваться как набор линий или любой другой игровой объект.
2. Разработка дополнительных игровых механик и интеграция дополнительных способов взаимодействия. Механики могут быть связаны с использованием различных цветов маркера, дополнительными игровыми объектами и т. д. Дополнительные способы взаимодействия подразумевает использование новых маркеров реального мира в ходе игрового процесса, а также управление жестами.

Возможные проблемы:

- Некорректное считывание нарисованных линий.
- В распознавание включаются нарисованные на фоне объекты.
- Проблемы в коммуникации между программными модулями: разница в записи координат и сложность в объединении программных компонентов, если они разрабатывались независимо.

Система оценивания

Критерии оценки задачи делятся на три составляющие:

- оценка этапа 1;
- оценка этапа 2;
- оценка презентации.

Критерии первых двух пунктов также делятся на функциональные критерии (60% баллов), критерии к оформлению и сданным материалам (10% баллов) и критерии усложненных функций (30% баллов). Они соответствуют подзадачам или включают в себя несколько подзадач, необходимых для решения задачи этапа (см. этап 1 и этап 2).

Таблица VI.2.1: Критерии этапа 1

Блок	**Название**	**Пояснения**	**Макс. балл**
Функциональные критерии	Динамическая проекция объектов.	1 балл — случайная генерация и фиксированное движение; 3 балла — созависимости траекторий и действий пользователя.	3
	Объект взаимодействия с проекцией: определение формы и цвета.	+5 баллов — определяется шар;* +1 балл — за каждый дополнительный цвет (максимум 4).	9
	Метод фиксации объекта с проекцией	2 балла — при пересечениях проекции с шаром; 5 баллов — в момент столкновения со стеной.	5
	Предобработка видеопотока	+4 балла — распознавание поверхности и угла наклона; +2 балла — в условиях разной освещенности (блики); +2 балла — цветокоррекция изображения.	8
	Количество механик для полноценной геймификации сервиса		2
	Полнота решения: запуск с одной кнопки и наличие старта, процесса игры, и завершение игрового процесса	+1 балл — наличие начального экрана; +3 балла — наличие игры; +1 балл — игра конечна и/или имеет экран конца игры.	5
Критерии к оформлению и сданным материалам	Код решения содержит комментарии или прилагаемое описание работы кода.	1 балл — комментарии описывают меньшую часть неочевидных решений, собственных функций и смысловых блоков кода; 2 балла — комментарии описывают большую часть неочевидных решений, собственных функций и смысловых блоков кода.	2

Таблица VI.2.1: Критерии этапа 1

Блок	**Название**	**Пояснения**	**Макс. балл**
	Сданные видеоматериалы полностью демонстрируют реализованную часть задачи.	1 балл — частично демонстрируют; 2 балла — полностью демонстрируют.	2
	Документация по использованному стеку технологий с объяснением каждого элемента.	1 балл — частичное описание используемых библиотек и кода; 2 балла — описаны для чего используется та или иная библиотека и ее место в решении.	2
Критерии усложненных функций	Разработка собственных маркеров для определения игровой области	2 балла — использование упрощенного решения (цветные точки, собственные ArUco-маркеры и т. п.); 5 баллов — обучение нейросетью на новые изображения.	5
	Создание рейтинговой таблицы	1 балл — вывод пользователю счета и фиксация в переменной кода/ввод имени пользователя; 2 балла — сохранение счета в файл; 3 балла — подключение базы данных.	3
	Динамическое масштабирование игрового поля в процессе игры	5 баллов — масштабирование в любой момент; 3 балла — при старте.	3

** В силу ограничений методов, предлагаемых классическим компьютерным зрением (без использования машинного обучения) и невысокого количества кадров в секунду от используемой веб-камеры, распознавание кругов на больших объектах в кадре (таких, как проходящий человек) не считается ошибкой.*

Таблица VI.2.2: Критерии этапа 2

Блок	**Название**	**Пояснения**	**Макс. балл**
Функциональные критерии	Реализация корзины.	1 балл — корзина статичная; 2 балла — корзина меняет местоположение на случайное по условию; 3 балла — корзина движется.	3
	Наличие реализации физики и отскоков мяча.	1 балл — мяч отскакивает в неверном направлении; 2 балла — физика внедрена корректно.	2
	Регистрация линий.	3 балла — обнаруживаются только прямые линии 5 баллов — обнаруживаются линии любой формы.	5

Таблица VI.2.2: Критерии этапа 2

Блок	**Название**	**Пояснения**	**Макс. балл**
	Базовая механика.	+1 балл — фиксируется попадание мяча в корзину; +1 балл — наличие счета; +2 балла — наличие таблицы рекордов.	4
	Предобработка видеопотока.	+3 балла — распознавание поверхности и угла наклона; +1 балл — в условиях разной освещенности (блики); +1 балл — цветокоррекция изображения.	5
	Полнота решения: запуск с одной кнопки и наличие старта, процесса игры и завершение игрового процесса.	+1 балл — наличие начального экрана; +3 балла — наличие игры; +1 балл — игра конечна и/или имеет экран конца игры.	5
Критерии к оформлению и сданным материалам	Сданные видеоматериалы полностью демонстрируют реализованную часть задачи.	1 балл — частично демонстрируют; 2 балла — полностью демонстрируют.	2
	Документация по использованному стеку технологий с объяснением каждого элемента.	1 балл — частичное описание используемых библиотек и кода; 2 балла — описано, для чего используется та или иная библиотека и ее место в решении.	2
Критерии усложненных функций	Учет перекрытия проекции человеком.		3
	Новые механики: новые способы взаимодействия — не только линии, например, дополнительные цвета или учет распознавания форм.		7

Некоторые повторяющиеся с первого этапа критерии были уменьшены в баллах, поскольку участники имели возможность переиспользовать код или получили навыки, значительно упрощающие реализацию.

Таблица VI.2.3: Критерии оценки презентации

Название	**Пояснения**	**Макс. балл**
В докладе представлены алгоритмы решения (для каждой из задач).	3 балла — за все подзадачи; 1–2 балла — частично.	6
В докладе рассказано о вкладе каждого участника в реализованный проект.		3

Таблица VI.2.3: Критерии оценки презентации

Название	**Пояснения**	**Макс. балл**
Техническая оригинальность решения.		2
Презентация имеет брендированный под команду стиль.		2
Ответы на вопросы	2 балла — команда отвечает уверенно; 1 балл — команда владеет информацией слабо.	2
Соблюдение регламента	Регламент — 7 мин на защиту +3 мин на демонстрационные видео.	1
Общие впечатления		1

Решение задачи

Этап 1. Охота на объекты в мирах

Программный цикл выглядит следующим образом:

1. Генерация изображения (итерация игрового цикла).
2. Вывод изображения на поверхность.
3. Получение с камеры видеоряда, содержащего проекцию.
4. Поиск на кадре нужных объектов (пластмассовых шариков).
5. Изменение состояния игровых объектов в соответствии с новой информацией.

Таким образом, итерация игрового цикла происходит внутри итерации программного.

1.1.Генерация шариков

Для определения поведения шариков был реализован класс `Balloon`, наследующий класс `pygame.sprite.Sprite`.

В инициализации определяем необходимые для отрисовки параметры (изображение — `self.surf` и его прямоугольник — `self.rect`), а также размер, скорость и изначальное положение шарика.

```
class Balloon(pygame.sprite.Sprite):
def __init__(self, x, y, size):
    super(Balloon, self).__init__()
    self.surf = pygame.image.load('assets/balloon.png').convert_alpha()
    self.surf = pygame.transform.scale(
        self.surf, (size * 2, size * 2 + 0.8 * size))
    self.rect = self.surf.get_rect()
    self.velocity = pygame.Vector2(0, -0.1)
    self.pos = pygame.Vector2(x, y)
    self.size = size
    self.rect.update(self.pos.x - self.size,
```

```
                    self.pos.y - self.size, self.size * 2, self.size * 2)
# ...
```

Случайная генерация является статическим методом и включает в себя случайную генерацию позиции, реализованную через генератор, который передается в функцию `generate(gen)` в качестве параметра.

```
class Balloon(pygame.sprite.Sprite):
# ...
    def generate(gen):
        size = 50
        x, y = next(gen)
        return Balloon(x, y, size)
# ...
```

Функция генератора: алгоритм заключается в том, что все поле разбивается на столбцы, в рамках которых в случайном порядке происходит генерация (случайный порядок реализован через функцию `_randomSequence(maxNum)`) шариков за нижней границей экрана. Чтобы такая логика не была слишком явной, в самих столбцах также выбирается случайный x.

```
class Balloon(pygame.sprite.Sprite):
# ...
def getNextPos(screenHeight, screenWidth, size):
    offset = 15
    cols = int(screenWidth / (size * 2 + offset))
    randXs = Balloon._randomSequence(cols)
    fieldWidth = int(screenWidth / cols)
    while True:
        if len(randXs) == 0:
            randXs = Balloon._randomSequence(cols)
        x = randXs.pop()
        randFieldX = (fieldWidth * x + size,
                      fieldWidth * (x + 1) - size)
        outX = random.randint(randFieldX[0], randFieldX[1])
        outY = screenHeight + size * 2 + 30
        yield outX, outY
def _randomSequence(maxNum):
    randList = [*range(maxNum)]
    random.shuffle(randList)
    return randList
```

Функция генерации вызывается из кода игры по кастомному событию и при запуске программы.

1.2 Поведение шариков

Позиция шарика должна высчитываться в каждом кадре на основе скорости, заданной в свойстве объекта. Для этого используется 2D-вектор, класс, предоставляемый библиотекой `Pygame`. В нашем случае скорость шарика в горизонтальной оси равна 0, а в вертикальной — $-0,1$, т. е. шарик движется строго вверх.

Функции обновления положения и рисования разделены, поскольку первая — более трудозатратная.

Переменная `dpos` — результат нормализации скорости. Таким образом, мы гарантируем, что при любом количестве кадров в секунду за одно и то же время объект преодолеет равное расстояние.

```
class Balloon(pygame.sprite.Sprite):
    # ...
    def update(self, dt):
        dpos = self.velocity * dt
        self.pos += self.dpos
        self.rect.move_ip(dpos.x, dpos.y)
    def draw(self, surf):
        surf.blit(self.surf, self.rect)
    # ...
```

1.3 Нахождение пластмассовых шариков

Для начала найдем все присутствующие в кадре `frame` круги. Переведем изображение в оттенки серого, наложим медианный фильтр для избавления от шумов и воспользуемся методом `cv2.HoughCircles(image, method, dp, minDist, circles, param1, param2, minRadius, maxRadius)`.

Он работает следующим образом:

1. с помощью оператора Кэнни на изображении в оттенках серого image выделяются контуры с параметрами `param1` — для верхнего порогового значения, `param1 / 2` — для нижнего;
2. для каждой точки полученных контуров выводятся окружности с радиусом, входящим в заданные границы (`minRadius` и `maxRadius`);
3. при наличии круга построенные окружности одного радиуса будут пересекаться в одной точке, т. е. чем больше пересечений — тем больше форма похожа на окружность — пороговое значение окружности определяется в `param2`;
4. близко лежащие точки отсеиваются на основании `minDist`;
5. найденные круги передаются на выход функции и в переменную `circles` координатами центра и радиуса.

Что касается нерассмотренных аргументов:

1. `method` — метод поиска окружностей, в `OpenCV` на данный момент реализован только `cv2.HOUGH_GRADIENT`;
2. `dp` — отношение разрешения аккумулятора к разрешению изображения; проще всего в качестве значения передавать 1. Параметры для трансформации Хафа подобраны эмпирическим путем и могут изменяться в зависимости от освещения.

```
def detectCircles(frame, minRadius=0, maxRadius=0):
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    gray = cv2.medianBlur(gray, 5)
    circles = cv2.HoughCircles(gray, cv2.HOUGH_GRADIENT, 1, 20,
        param1=50, param2=25, minRadius=minRadius, maxRadius=maxRadius)
```

```
    if circles is None:
        return

    circles = np.uint16(np.around(circles)) # Необходимо для дальнейшей работы
    return circles
```

Далее полученные круги передаем в функцию `separateCircles`, в которой сравниваются позиции и размеры найденных кругов с имеющимися игровыми (переменной `offset` допускаем небольшую погрешность).

```
def separateCircles(circles, game, frame):
    offset = 30
    deletedIdx = []
    for i, c in enumerate(circles[0, :]):
        for b in game.baloons:
            xInRange = c[0] > b.pos.x - offset and c[0] < b.pos.x + offset
            yInRange = c[1] > b.pos.y - offset and c[1] < b.pos.y + offset
            rInRange = c[2] > b.size - 5 and c[2] < b.size + 5

            if xInRange and yInRange and rInRange:
                deletedIdx.append(i)
                continue

    background = [[b.pos.x, b.pos.y, b.size] for b in game.baloons]
    foreground = np.delete(circles[0], deletedIdx, axis=0)
    # Обводим круги разными цветами
    for c in foreground:
        cv2.circle(frame, (c[0], c[1]), c[2], (225, 0, 0), 2)
    for c in background:
        cv2.circle(frame, (int(c[0]), int(c[1])), c[2], (0, 255, 0), 2)
    return foreground, background
```

1.4 Разделение пластмассовых шариков по четырем цветам

Для разделения цветов имеющийся кадр сначала преобразовывается из BGR (RGB в формате `OpenCV`) в `HSV`, после чего по граничным значениям цвета находятся на изображении. HSV используется по причине более простого определения границ цвета: непосредственно оттенок задается первым параметром, второй влияет на насыщенность (чем меньше, тем ближе цвет к белому), третий — на яркость (чем меньше — тем ближе цвет к черному). Таким образом, при использовании цветов с далекими друг от друга оттенками достаточно будет сопоставить границы оттенка с имеющимся цветом. Для минимальных значений каналов S и V использовать близкие к нулю, а для максимальных — 255.

Границы цветов могут различаться в зависимости от особенностей камеры и освещения.

```
def getColorMask(frame):
    yellow = [[20, 30, 30], [35, 255, 255]]
    green = [[40, 30, 30], [75, 255, 255]]

    # Красный присутствует на обеих границах
```

```
    red_low = [[0, 30, 30], [15, 255, 255]]
    red_high = [[160, 30, 30], [179, 255, 255]]

    blue = [[80, 30, 30], [130, 255, 255]]

    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    mask_yellow = cv2.inRange(hsv, yellow[0], yellow[1])
    mask_red = cv2.bitwise_or(cv2.inRange(
        hsv, red_low[0], red_low[1]), cv2.inRange(hsv, red_high[0],
        red_high[1]))
    mask_blue = cv2.inRange(hsv, blue[0], blue[1])
    mask_green = cv2.inRange(hsv, green[0], green[1])

    return mask_yellow, mask_red, mask_blue, mask_gree
```

1.5. Калибровка изображения

Алгоритм калибровки поверхности выглядит следующим образом:

1. При запуске приложения в окне, в котором впоследствии будет отображаться игра, выводятся четыре ArUco-маркера. Каждый находится в одном из углов окна и повернут так, чтобы его изначально левый верхний угол находился в ближайшем углу окна.
2. Камера фиксируется, и на полученном с нее изображении выполняется поиск маркеров.
3. Как только будут найдены четыре маркера, выполняется их сортировка в порядке выведения (по часовой стрелке, начиная с левого верхнего угла) и записывается положение их изначально левых верхних углов.
4. Используя операцию `cv2.getPerspectiveTransform(corners, out)`, где `out` — углы окна со сгенерированными маркерами, проводится соответствие между двумя системами координат. На выход получаем матрицу трансформации перспективы. Запускается игра.
5. На каждом кадре проводим `cv2.warpPerspective` с полученной матрицей, здесь же указываем необходимое разрешение.

Необходимые в решении методы были вынесены в класс `ArucoUtils`.

```
class ArucoUtils:
    def __init__(self, size: int,
    markersDictionary: dict = cv2.aruco.DICT_4X4_50):
        self.size = size
        self.availableMarkers = markersDictionary
        self.markers = []
        self.ids = [4, 7, 13, 24]
        self.detectedMarkers = {"Ids": np.array([]), "Corners": np.array([])}
        self.offset = 3

        # self.detectedCorners = np.array([])
        # self.detectedIds = np.array([])

    def _generateMarker(self, idx: int):
        marker = np.zeros((self.size, self.size, 1), dtype="uint8")
```

```
        markersDict = cv2.aruco.getPredefinedDictionary(self.availableMarkers)
        markersDict.generateImageMarker(idx, self.size, marker)
        return marker

    def generateMarkers(self):
        for i in self.ids:
            self.markers.append(self._generateMarker(i))

    def placeMarkers(self, image):
        size = self.size + self.offset
        image[self.offset:size, self.offset:size] = self.markers[0]
        image[-size:-self.offset, -size:-
              self.offset] = self.markers[2]
        image[self.offset:size, -size:-
              self.offset] = self.markers[1]
        image[-size:-self.offset, self.offset:size] = self.markers[3]

    def detectCorners(self, image):
        self._detectMarkers(image)
        if type(self.detectedMarkers["Ids"]) == np.ndarray and
        len(self.detectedMarkers["Ids"]) == len(self.ids):
            allCorners = self.detectedMarkers["Corners"]
            return np.array([allCorners[0][0][0], allCorners[3][0][3],
            allCorners[2][0][2], allCorners[1][0][1]])
        return None

    def _detectMarkers(self, image):
        greyImg = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        detectorParams = cv2.aruco.DetectorParameters()
        markersDict = cv2.aruco.getPredefinedDictionary(self.availableMarkers)
        detector = cv2.aruco.ArucoDetector(markersDict, detectorParams)
        rejectedImgPoints = np.array([])
        self.detectedMarkers["Corners"], self.detectedMarkers["Ids"],
        rejectedImgPoints = detector.detectMarkers(greyImg)
        self.detectedMarkers["Corners"] = np.array(
            self.detectedMarkers["Corners"])
        if type(self.detectedMarkers["Ids"]) == np.ndarray:
            self._sortMarkers()
        return rejectedImgPoints

    def outlineMarkers(self, image):
        cv2.aruco.drawDetectedMarkers(
            image, self.detectedMarkers["Corners"], self.detectedMarkers["Ids"])

    def _sortMarkers(self):
        sortedIds = self.detectedMarkers["Ids"].argsort(0)[
            :, 0]
        self.detectedMarkers["Ids"] = self.detectedMarkers["Ids"][sortedIds]
        self.detectedMarkers["Corners"] =
        self.detectedMarkers["Corners"][sortedIds]
```

Функция создания окна с маркерами.

```
def getTestAruco():
```

```
    _arucoUtils = aruco.ArucoUtils(100)
    _arucoUtils.generateMarkers()
    _image = np.empty((480, 640, 3), dtype='uint8')
    _image.fill(255)
    _arucoUtils.placeMarkers(_image)
    cv2.imshow("Game", _image)
    return _arucoUtils, _image
```

Функция получения матрицы.

```
def getWarpMatrix(frame, markersImage):
    corners = arucoUtils.detectCorners(frame)
    arucoUtils.outlineMarkers(frame)
    warpMatrix = None

    cv2.imshow('Frame', frame)

    if type(corners) == np.ndarray:
        # x, y format
        out = np.float32([[0, 0],
                          [0, markersImage.shape[0] - 1],
                          [markersImage.shape[1] - 1,
                           markersImage.shape[0] - 1],
                          [markersImage.shape[1] - 1, 0]])
        # print(corners, out)
        warpMatrix = cv2.getPerspectiveTransform(corners, out)
    return warpMatrix
```

1.6. Создание начального экрана и экрана конца игры

Начальным экраном данного решения выступает калибровочное окно, представленное на предыдущем шаге. Для последующих изменений игровых состояний был реализован enum-класс `GameStates`.

```
class GameStates(Enum):
    RUNNING = 0
    GAME_OVER = 1
    QUIT = 2

class BalloonsGame:
    # ...
    def run(self):
        for event in pygame.event.get():
            if event.type == self.timerEvent and self.timer > 0:
                self.timer -= 1
                if self.timer <= 0:
                    self.state = GameStates.GAME_OVER
    # ...
```

Условием завершения игры (перехода в состояние `GAME_OVER`) является окончание игрового таймера. При этом прекращается расчет игровой логики и отрисовка объектов, вместо этого выводится надпись Game over и итоговый счет.

```
class BalloonsGame:
    # ...
    def _writeScore(self):
        myfont = pygame.font.Font('./assets/Minecraft Seven.ttf', 30)
        label = myfont.render("Score: " + str(self.score), 1, (0, 0, 0))
        self.screen.blit(label, (20, 10))

    def _writeGameOver(self):
        myfont = pygame.font.Font('./assets/Minecraft Seven.ttf', 50)

        label = myfont.render("GAME OVER", 1, (0, 0, 0))
        label_rect = label.get_rect(
        center=(self.screen.get_width()/2, self.screen.get_height()/2))
        self.screen.blit(label, label_rect)
    # ...
```

1.7. Объединение различных модулей в одну программу

В решении используется два вышеуказанных модуля: `aruco` (собственный) и `game`, каждый из которых с другим напрямую не взаимодействует, однако результаты выполнения их методов объединяются в `main`, который также содержит большинство функций для работы с компьютерным зрением. Запуск происходит через единственный файл — `main.py`.

1.8. Разработка игровых механик

Для примера в решении представлена самая базовая возможная механика: попадание по мячам за очки. Попадание засчитывается при расстоянии между центрами проецируемого шарика и пластмассового меньшем, чем радиус проецируемого.

```
class BalloonsGame:
    # ...
    def popBalloon(self, pos):
        for b in self.Balloons:
            if math.sqrt((pos[0] - b.pos.x) ** 2 + (pos[1] - b.pos.y) ** 2)
            < b.size:
                b.pop(pos)
                Timer(3, self.removeBalloon, [b]).start()
                self.score = self.score + 1

    def removeBalloon(self, item):
        self.Balloons.remove(item)

    def _writeScore(self):
        myfont = pygame.font.Font('./assets/Minecraft Seven.ttf', 30)
        label = myfont.render("Score: " + str(self.score), 1, (0, 0, 0))
        self.screen.blit(label, (20, 10))
    # ...
```

1.9. Улучшение изображения

Перед тем как изображение подается основной программе, оно проходит следующую обработку по каждому из RGB-каналов.

```
def preprocess(frame):
    rgb_planes = cv2.split(frame)

    result_norm_planes = []
    for plane in rgb_planes:
```

Первые два пункта позволяют сгладить разницу в освещении на изображении.

- Все имеющиеся пиксели сильно увеличиваются в размерах (ранее рассмотренная морфологическая операция `cv2.dilate`) и размываются с помощью медианного фильтра (то есть не остается почти ничего, кроме градиента теней).

```
dilated_img = cv2.dilate(plane, np.ones((7,7), np.uint8))
bg_img = cv2.medianBlur(dilated_img, 21)
```

- Находится разница между исходным изображением и полученным, после чего она инвертируется (пиксели с одинаковыми значениями становятся черными и наоборот).

```
diff_img = 255 - cv2.absdiff(plane, bg_img)
```

- Изображение нормализуется — выравнивается его гистограмма для лучшей контрастности.

```
norm_img = cv2.normalize(diff_img, None, alpha=0, beta=255,
norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_8UC1)
result_norm_planes.append(norm_img)
```

После нормализации всех каналов происходит их повторное слияние. Получившееся изображение — результат выполнения функции.

```
result_norm = cv2.merge(result_norm_planes)
return result_norm
```

1.10. Столкновение шарика с проекцией

Для определения момента столкновения мяча с проекцией использовалась его скорость. Сначала были собраны данные по скорости мячей в течение игры, что позволило увидеть, какие наименьшие значения фиксируются — это и есть момент его остановки. Далее в примерных найденных границах было определено среднее (медиана, поскольку в данном случае существует вероятность выбросов), которое и использовалось в конечной программе для определения момента остановки.

Для определения пересечения мячика с игровыми мишенями была реализована функция `detectCollision`, которая сначала разделяет проецируемые круги от посторонних с помощью функции `separateCircles`, рассмотренной в пункте 1.3, после чего находит пересечения между представленными в списках кругами.

```
def detectCollision(rawCircles, game, frame):
    foreground, background = separateCircles(rawCircles, game, frame)
    collided = np.empty(shape=(0, 2), dtype=np.int8)
```

```
    for fg in foreground:
        fgX = fg[0]
        fgY = fg[1]
        fgR = fg[2]

        for bg in background:
            bgX = bg[0]
            bgY = bg[1]
            bgR = bg[2]
            dist = sqrt((fgX - bgX) ** 2 + (fgY - bgY) ** 2)
            if dist < (bgR + fgR / 2):
                collided = np.append(collided, [[fgX, fgY]], axis=0)
    return collided
```

Коллизий может быть несколько из-за случайного появления в кадре посторонних объектов, таких как рука.

Далее определяем скорость шарика между кадрами с помощью функции `findMinDist`, которая находит минимальное расстояние между прошлой позицией и настоящей. На выходе получаем непосредственно скорость и соответствующую ей новую точку в пространстве.

```
def findMinDist(collided, prevPos):
    dists = np.linalg.norm(collided - prevPos, axis=1)
    return np.min(dists), np.argmin(dists)
```

Если при сравнении скорости с выбранным пороговым значением (принцип описан в начале пункта) она оказывается меньше, «лопаем» шарик и аннулируем позицию для измерения скорости prevPos. Так или иначе, новая позиция возвращается из функции для ее использования в следующей итерации.

```
def runGame(frame, prevPos):
        # ...
        collided = detectCollision(circles, game, gameImage)
        if np.any(collided):
            minVal, minIdx = findMinDist(collided, prevPos)
            prevPos = collided[minIdx]

            if minVal < 17.5:
                game.popBaloon(prevPos)
                prevPos[:] = 0
            # ...

    return prevPos, minVal
```

Источник

Полный код решения можно найти по ссылке: https://github.com/KindWarlock/Shooter.

Этап 2. Распознавание сущностей

В решении используется предложенная для физических расчетов библиотека `pymunk`.

2.1 Генерация мячей

Создание мяча происходит один раз за игру, в момент ее старта. Он представляет собой `pymunk.Body` — физическое тело, одну из основных сущностей библиотеки `pymunk`. При создании ей задаются три параметра: масса, момент вращения и тип (в нашем случае — `pymunk.Body.DYNAMIC`, означающий, что расчет изменения положения объекта (ускорение и скорость), а также коллизий — соприкосновений между телами — передается библиотеке).

Стоит отметить, что задача `pymunk` — определить поведение объекта, чтобы потом, используя новую позицию, его можно было отрисовать в `pygame`.

В `pymunk` тело состоит из форм — объектов типа `pymunk.Shape`, на основе которых рассчитываются коллизии (если привязанных форм несколько, то поведение каждой по отдельности). Например, если в симуляции падения тело состоит из двух форм разного веса, соединенных веревкой, более легкая будет всегда находится выше, а ее скорость будет во многом зависеть от скорости более тяжелой. В нашей задаче все гораздо проще: тело состоит из одной простой формы — круга, для которого используем `pymunk.Circle`, указывая объект привязки и радиус. Также мы указываем трение (`friction`) и упругость (`elasticity`). Здесь и далее параметры выбираются таким образом, чтобы мяч не менял своей изначальной скорости, только направление.

Позиция появления выбирается случайной точкой на поле, а вектор скорости — одним из диагональных направлений (влево вверх, вправо вверх, вправо вниз и влево вниз), умноженных на скалярную величину скорости.

После определения всех свойств объекта он добавляется в пространство `pymunk.Space` (здесь представлено переменной `space`), в котором должны находится все созданные физические объекты.

```
class Game:
    # ...
    def create_ball(self, space, position):
        body = pymunk.Body(1, 100, body_type=pymunk.Body.DYNAMIC)

        shape = pymunk.Circle(body, 20)
        shape.elasticity = 1
        shape.friction = 0.0
       body.position = random.randint(50, screen_width - 50),
       random.randint(50, screen_height - 50)
        direction = random.choice([(1, 1), (-1, 1), (1, -1), (-1, -1)])
        speed = 300
        body.velocity = direction[0] * speed, direction[1] * speed

        space.add(body, shape)
        return shape
    # ...
```

2.2 Реализация корзины

Корзина в данном решении состоит из трех линий, для создания которых был реализован метод класса `Game` `create_body_from_lines` (универсальный и используется для всех физических линий в игре). В него в качестве параметров передается пространство, список линий, тип коллизии (представляет собой целое число, назначение которого будет рассмотрено в следующем пункте; вопрос о том, является ли тело контуром, решается в дальнейшем). Далее, в зависимости от типа коллизии, создается тело одного из типов, указанных в классе `Enum`.

```
class Game:
    # ...
    class coll_types(Enum):
        BALL = 0
        STATIC = 1
        RING = 2
    # ...
```

В нашем случае это будет `pymunk.Body.KINEMATIC`, означающий, что тело может двигаться только по логике, прописанной нами в коде (указана ниже).

Поскольку тело состоит из линий, в качестве формы для каждой задаем `pymunk.Segment` с координатами относительно тела, используя для перевода из одной системы координат в другую метод `pymunk.Body.world_to_local`.

```
class Game:
    # ...
    def create_body_from_lines(self, space, lines, collision_type,
    is_contour=False):
        if collision_type == self.coll_types.STATIC:
            body_type = pymunk.Body.STATIC
        else:
            body_type = pymunk.Body.KINEMATIC
        body = pymunk.Body(body_type=body_type)
        if is_contour:
  # ...
        else:
            body.position = lines[0][0]
        space.add(body)
        for line in lines:
            shape = pymunk.Segment(body, body.world_to_local(
                line[0]), body.world_to_local(line[1]), 1)
            shape.elasticity = 1  # Эластичность стены
            shape.friction = 0  # Трение стены
            # print(collision_type.value)
            shape.collision_type = collision_type.value
            space.add(shape)
        return body
    # ...
```

Для движения корзины определяем собственную функцию изменения положения. Дополнительно к самому движению корзины с указанной скоростью в ней присутствует проверка соприкосновения с краями экрана: в случае касания она меняет вектор скорости на противоположный.

```
class Game:
    def main(self):
        # ...
        def move(body, dt):
            body.position += body.velocity
            if body.position.x <= 0 or body.position.x + ring_w >= screen_width:
                body.velocity *= -1
                body.position += 2 * body.velocity
        ring = self.create_body_from_lines(
            space, ring_lines, self.coll_types.RING)
        ring.velocity = pymunk.Vec2d(5, 0)
        ring.position_func = move
        # ...
```

Для отрисовки корзины и других сегментированных форм была реализована следующая функция. Поскольку при передвижении тела относительное положение сегментов остается неизменным и меняется только ее собственное, для отрисовки каждой формы к ее точкам следует прибавить текущее положение тела (о вычислениях точек по оси Y речь пойдет ниже).

```
def draw_segment_body(self, screen, body, color, width=1):
        for line in body.shapes:
            moved_a = body.position + line.a
            moved_b = body.position + line.b
            p1 = int(moved_a.x), int(screen_height - moved_a.y)
            p2 = int(moved_b.x), int(screen_height - moved_b.y)
            pygame.draw.lines(screen, color, False, [p1, p2], width=width)
```

2.3 Реализация физики

Как уже было сказано, в примере используется библиотека `pymunk`, которая при каждом вызове функции `pymunk.Space.step(dt)` рассчитывает поведение всех инициализированных объектов. Нами уже были рассмотрены объекты линий и мяча, в данном случае речь пойдет о реализованных способах взаимодействия между ними.

Для увеличения разнообразия были переопределены обработчики коллизий: при столкновении мяча с линией к значению угла его отражения добавляется небольшая погрешность. В зависимости от объекта, к которому линия принадлежит, происходит одно из событий:

- увеличивается счет (для корзины);
- проигрывается звук (для стен или нарисованных линий).

В `pymunk` такое можно сделать следующим образом:

1. Определить новые функции обработки с аргументами (`arbiter`, `space`, `data`), где с помощью `arbiter` можно будет получить параметры столкновения, `space` означает используемое пространство, а `data` — дополнительная информация.
2. Добавить в функции необходимый функционал, вернуть из функции булево значение (`True` при удачной обработке).

```
class Game:
    def main(self):
        # ...
```

```
    def ball_window_collision(arbiter, space, data):
        ball_shape, border_shape = arbiter.shapes
        # Получаем текущее направление движения мяча
        velocity = ball_shape.body.velocity
        # Добавляем случайное отклонение к новому направлению
        velocity = (velocity[0] + random.uniform(-50, 50),
                    velocity[1] + random.uniform(-50, 50))
        # Устанавливаем новое направление движения мяча
        ball_shape.body.velocity = velocity
        # Воспроизведение звука столкновения
        self.collision_sound.play()
        return True
    def ball_ring_collision(arbiter, space, data):
        ball_shape, ring_shape = arbiter.shapes
        velocity = ball_shape.body.velocity
        velocity = (velocity[0] + random.uniform(-50, 50),
                    velocity[1] + random.uniform(-50, 50))
        ball_shape.body.velocity = velocity
        self.score += 1
        return True
```

3. Создать объект обработчика, передав в качестве параметров, между объектами с какими типами коллизий будет происходить столкновение.

```
ring_collision_handler = space.add_collision_handler(
        self.coll_types.BALL.value, self.coll_types.RING.value)

collision_handler = space.add_collision_handler(
    self.coll_types.BALL.value, self.coll_types.STATIC.value)
```

4. Указать созданные в пунктах 1–2 функции для `handler.pre_solve`.

```
ring_collision_handler.pre_solve = ball_ring_collision
    collision_handler.pre_solve = ball_window_collision
    # ....
```

2.4 Регистрация линий

Функция нахождения линий вызывает каждый кадр, а за основу распознавания использует маску по цвету (синему).

```
class Game:
    # ...
    def detect_blue_lines(self, frame, lower_blue, upper_blue, blur_value,
    threshold_value, dilation_value):
        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)  # Определение HSV кода
        # Создание маски на основе
        mask = cv2.inRange(hsv, lower_blue, upper_blue)
```

Далее для избавления от шумов используется размытие по Гауссу. Полученная mask черно-белая, однако после размытия в ней появились оттенки серого, которые отсеиваем с помощью бинаризации (значение интенсивности ниже порогового обращается в 0, остальные — в 1).

```
    blurred = cv2.GaussianBlur(
            mask, (blur_value * 2 + 1, blur_value * 2 + 1), 0)
        _, thresh = cv2.threshold(
            blurred, threshold_value, 255, cv2.THRESH_BINARY)
```

Полученные линии могут быть некорректно тонкими, поэтому сделаем их толще с помощью морфологической операции `cv2.dilate` (к пикселям вокруг каждого пикселя бинарного изображения со значением 1 в соответствии с `kernel` применяется операция бинарного ИЛИ; аргумент `iterations` показывает, сколько раз надо провести данную операцию).

```
 kernel = np.ones(
            (dilation_value * 2 + 1, dilation_value * 2 + 1), np.uint8)
        dilated = cv2.dilate(thresh, kernel, iterations=1)
```

Для поиска контуров используем метод `cv2.findContours`, в который передаем само изображение, метод (в нашем случае `cv2.RETR_LIST`, который ищет и возвращает контуры в самой простой форме) и `cv2.CHAIN_APPROX_SIMPLE`, убирающий ненужные точки.

```
# Определение контуров
 contours, _ = cv2.findContours(
            dilated, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
        return contours
        # ...
```

Полученные контуры добавляются с помощью рассмотренного выше метода с флагом `is_contour=True`. В таком случае логика обработки несколько меняется: сначала убираются лишние оси, затем точки разбиваются по парам и приводятся к списку.

```
class Game:
    # ...
    def pairwise(self, iterable):
        a = iter(iterable)
        return zip(a, a)
    def create_body_from_lines(self, space, lines, collision_type,
    is_contour=False):
        # ...
        if is_contour:
            lines = np.squeeze(lines, axis=1)
            new_lines = []
            for p1, p2 in self.pairwise(lines):
                new_lines.append([list(p1), list(p2)])
            lines = new_lines
            body.position = list(lines[0][0])
        else:
            body.position = lines[0][0]
        # ...
```

Также для поиска контуров можно было использовать оператор Кэнни (подробнее о нем можно почитать в материалах для подготовки).

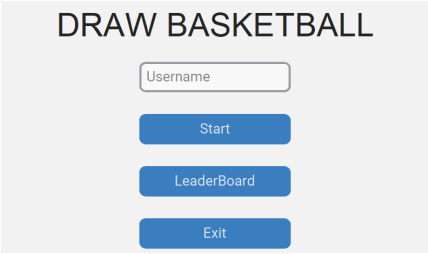
2.5 Создание начального экрана и экрана окончания игры

Интерфейс был реализован с помощью библиотеки `CustomTkinter`, каждый экран представляет собой отдельный класс. Желаемую тему интерфейса записываем в константы, которые используем позже.

```
APPEARANCE_MODE = "light"  # Modes: system (default), light, dark
COLOR_THEME = "dark-blue"  # Themes: blue (default), dark-blue, green
```

На начальном экране предлагается ввести имя игрока, а также имеется три кнопки:

- игра;
- список лидеров;
- выход.




```
class MainMenu:
    usernameInputField = ""
    def __init__(self):
        self.username = None
        self.CameraClass = None
    def on_exit(self):
        self.app.destroy()
```

При старте игры запускаем два потока: с самой игрой и таймером (отдельный класс).

```
class Timer:
    timer_counter = 0
    time_on_end = 60
    isEnded = False
    def main(self):
        while self.timer_counter < self.time_on_end:
            time.sleep(1)
            self.timer_counter += 1
        LooseClass = Loose()
        LooseThread = threading.Thread(target=LooseClass.main())
        LooseThread.start()
class MainMenu:
    # ...
```

```
    def on_start_game(self):
        self.CameraClass = game.Game(username=self.usernameInputField.get())
        Timer_Class = Timer()
        Timer_Thread = threading.Thread(target=Timer_Class.main)
        camera_Thread = threading.Thread(target=self.CameraClass.main)
        Timer_Thread.start()
        camera_Thread.start()
        # ...
```

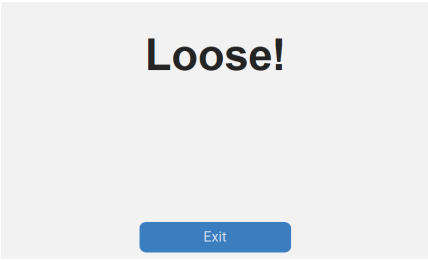
Нажатие на кнопку с таблицей рекордов также создает отдельный поток.

```
@staticmethod
    def on_leader_board():
        LeaderBoardClass = LeaderBoard()
        LeaderBoardThread = threading.Thread(target=LeaderBoardClass.main())
        LeaderBoardThread.start()
```

Метод `main` — основной, и вызывается при запуске приложения. В нем создается весь интерфейс и указываются параметры оформления, после чего запускается основной цикл приложения.

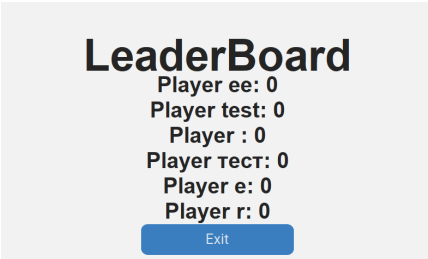
```
def main(self):
        ctk.set_appearance_mode(APPEARANCE_MODE)
        ctk.set_default_color_theme(COLOR_THEME)
        self.app = ctk.CTk()
        self.app.geometry("400x240")
        self.app.title("Main Menu")
        # Use CTkButton instead of tkinter Button
        PerpetuaFont = ctk.CTkFont(family='Perpetua', size=30)
        ctk_label = ctk.CTkLabel(
            master=self.app, text="DRAW BASKETBALL", font=PerpetuaFont)
        button_start = ctk.CTkButton(
            master=self.app, text="Start", command=self.on_start_game)
        button_exit = ctk.CTkButton(
            master=self.app, text="Exit", command=self.on_exit)
        button_leaderboard = ctk.CTkButton(master=self.app, text="LeaderBoard",
                                           command=self.on_leader_board)
        usernameInputField = ctk.CTkEntry(
            self.app, placeholder_text="Username")
        usernameInputField.place(relx=0.5, rely=0.3, anchor=ctk.CENTER)
        self.username = usernameInputField.get()
        self.usernameInputField = usernameInputField
        button_start.place(relx=0.5, rely=0.5, anchor=ctk.CENTER)
        button_leaderboard.place(relx=0.5, rely=0.7, anchor=ctk.CENTER)
        button_exit.place(relx=0.5, rely=0.9, anchor=ctk.CENTER)
        ctk_label.place(relx=0.5, rely=0.1, anchor=ctk.CENTER)

        self.app.mainloop()
```

Экран проигрыша содержит одну кнопку, закрывающую окно.




```
class Loose:
    def onExit(self):
        self.app.destroy()
    def main(self):
        ctk.set_appearance_mode(APPEARANCE_MODE)
        ctk.set_default_color_theme(COLOR_THEME)
        self.app = ctk.CTk()
        self.app.geometry("400x240")
        self.app.title("Loose")
        Perpetua_font_label = ctk.CTkFont(
            family='Perpetua', size=40, weight="bold")
        Perpetua_font_leader = ctk.CTkFont(
            family='Perpetua', size=20, weight="bold")
        button_exit = ctk.CTkButton(
            master=self.app, text="Exit", command=self.onExit)
        ctk_label = ctk.CTkLabel(
            master=self.app, text="Loose!", font=Perpetua_font_label)
        ctk_label.place(relx=0.5, rely=0.2, anchor=ctk.CENTER)
        button_exit.place(relx=0.5, rely=0.9, anchor=ctk.CENTER)
        self.app.mainloop()
```

В таблице рекордов указаны все записанные игроки и их наивысший счет. Все данные занесены в базу данных, обращение к ней происходит с помощью `database.Login().get_users()`. О ней и о других функциях взаимодействия с БД речь пойдет в следующем пункте. Сейчас важно лишь то, что на выходе она возвращает список кортежей (игрок, рекорд).




```
class LeaderBoard:
    def __init__(self):
```

```
        self.leaderText = self.parseLeaders()
    def parseLeaders(self) -> str:
        string_to_return = ""
        lines = database.Login().get_users()
        for line in lines:
            name = line[0]
            score = line[1]
            string_to_return += f"Player {name}: {score}\n"
        string_to_return.lstrip()
        print(string_to_return)
        if string_to_return == "":
            return "Nothing"
        return string_to_return
    def onExit(self):
        self.app.destroy()
    def main(self):
        ctk.set_appearance_mode(APPEARANCE_MODE)
        ctk.set_default_color_theme(COLOR_THEME)
        self.app = ctk.CTk()
        self.app.geometry("400x240")
        self.app.title("LeaderBoard")
        # Use CTkButton instead of tkinter Button
        Perpetua_font_label = ctk.CTkFont(
            family='Perpetua', size=40, weight="bold")
        Perpetua_font_leader = ctk.CTkFont(
            family='Perpetua', size=20, weight="bold")
        ctk_label = ctk.CTkLabel(
            master=self.app, text="LeaderBoard", font=Perpetua_font_label)
        ctk_labelListOfLeaders = ctk.CTkLabel(
            master=self.app, text=self.leaderText, font=Perpetua_font_leader)
        button_exit = ctk.CTkButton(
            master=self.app, text="Exit", command=self.onExit)
        ctk_label.place(relx=0.5, rely=0.2, anchor=ctk.CENTER)
        ctk_labelListOfLeaders.place(relx=0.5, rely=0.6, anchor=ctk.CENTER)
        button_exit.place(relx=0.5, rely=0.9, anchor=ctk.CENTER)
        self.app.mainloop()
```

2.6 Реализация системы очков

В пункте 2.4 при рассмотрении обработчиков коллизий уже было упомянуто, что очки присуждаются в момент столкновении с корзиной. По окончании игры они записываются в базу данных `SQLite`, взаимодействие с которой реализовано в отдельном классе `Login`.

При создании объекта происходит попытка подключения к базе данных. Если она не была найдена, то будет создана в папке с файлом. Далее получаем курсор, с помощью которого будет происходить обращение к базе, и создаем саму таблицу.

```
class Login:
    def __init__(self):
        # Подключаемся к базе данных
        self.conn = sqlite3.connect('mydatabase.db')
        self.cursor = self.conn.cursor()
```

```
        # Создаем таблицу, если она еще не существует
        self.cursor.execute('''
            CREATE TABLE IF NOT EXISTS users (
                username TEXT,
                score INTEGER
            )
        ''')
        self.conn.commit()
```

Запись новых игроков и обновление рекордов старых происходит через одну функцию.

```
 def add_update(self, username, score):
        self.cursor.execute(
            'SELECT * FROM users WHERE username = ?', (username,))
        ex_user = self.cursor.fetchone()
        if ex_user is None:
            self.cursor.execute(
        'INSERT INTO users (username, score) VALUES (?, ?)', (username, score))
        else:
            stored_score = ex_user[1]
            if score > stored_score:
                self.cursor.execute(
            'UPDATE users SET score = ? WHERE username = ?', (score, username))
        # Сохраняем изменения
        self.conn.commit()
```

Для получения всех игроков используется простой select-запрос.

```
def get_users(self):
        self.cursor.execute('SELECT * FROM users')
        return self.cursor.fetchall()
```

2.7 Объединение различных модулей в одну программу

Приложение содержит три модуля и базу данных — ко всем этим компонентам можно так или иначе перейти в ходе работы приложения. Само приложение запускается через файл `User_Interface`, представленный в предыдущем пункте, из которого можно непосредственно перейти в игру (`game`). Взаимодействие с модулем `database` и базой данных возможно в двух случаях: окончание игры и запись счета в таблицу и получение таблицы рекордов.

2.8. Улучшение изображения

Калибровка поверхности аналогична пункту 1.9.

2.9. Калибровка и преобразования координат

Калибровка поверхности аналогична пункту 1.5.

Pymunk использует систему координат, отличающуюся от той, которую использует pygame: в первом случае начало оси Y находится внизу, а во втором — сверху. Кроме того, в Pymunk координаты записаны числами с плавающей точкой, а в pygame позиция определяется целыми числами.

Пример взаимодействия между библиотеками.

```
class Game:
    # ...
    def draw_segment_body(self, screen, body, color, width=1):
        for line in body.shapes:
            moved_a = body.position + line.a
            p1 = int(moved_a.x), int(screen_height - moved_a.y)

            # ...
```

Также в Pymunk имеется отдельный модуль, упрощающий взаимодействие с pygame — Pymunk.pygame_util.

Источник

Полный код решения можно найти по ссылке: https://github.com/KindWarlock/Lines.

Материалы для подготовки

1. Документация OpenCV — https://docs.opencv.org/4.x/index.html.
2. Книга Learning OpenCV — https://www.oreilly.com/library/view/learning-opencv/9780596516130/.
3. Онлайн-курс по основам компьютерного зрения и машинного обучения — https://openedu.ru/course/hse/COMPVISION/?ysclid=liwrmg9rgn307596833/.
4. Онлайн-курс по основам OpenCV — https://stepik.org/course/116539/promo/.
5. Онлайн-курс по решению практических задач в области компьютерного зрения — https://stepik.org/course/50352/promo/.
6. Книга Python Cookbook, 3rd Edition — https://www.oreilly.com/library/view/python-cookbook-3rd/9781449357337/.
7. Онлайн-курс по разработке приложений на Python — https://www.youtube.com/playlist?list=PLQC2_OcDcSKBHamFYA6ncnc_fYuEQUy0s/.
8. Документация pygame — https://www.pygame.org/.
9. Примеры реализации различных задач в pygame — https://github.com/Rabbid76/PyGameExamplesAndAnswers/tree/master/.
10. Онлайн-курс по основам работы в команде —https://proskilling.ru/Effektivnyye_kommunikatsii_i_rabota_v_komande_new_course/.

Критерии определения победителей и призеров

Первый отборочный этап

В первом отборочном этапе участники решали задачи предметного тура по двум предметам: математике и информатике и инженерного тура. В каждом предмете максимально можно было набрать 100 баллов, в инженерном туре 100 баллов. Для того, чтобы пройти во второй этап участники должны были набрать в сумме по обоим предметам не менее 40 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 25 баллов, независимо от уровня.

Заключительный этап

Индивидуальный предметный тур

- математика — максимально возможный балл за все задачи — 62 балла;
- информатика — максимально возможный балл за все задачи — 100 баллов.

Командный инженерный тур

Команды заключительного этапа получали за командный инженерный тур от 0 до 68 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 (математика, информатика) с весом $K_1 = 0,2$ каждый предмет и баллы за командное решение задач инженерного тура с весом $K_2 = 0,6$.

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 математике (предметный тур) в стобалльной системе ($S_{1\ \text{макс}} = 100$);

S_2 — балл первой части заключительного этапа по информатике (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 2 победителя и 6 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	84,60 и выше
Призеры	От 77,01 до 83,87

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».