МАТЕРИАЛЫ ЗАДАНИЙ

Всероссийской междисциплинарной олимпиады школьников
«Национальная технологическая олимпиада»

по профилю
«Искусственный интеллект»

2023/24 учебный год

http://ntcontest.ru

УДК 373.5.016:004.8
ББК 74.263.2
И86

Авторы:
М.В. Бабушкин, Е.Н. Горечин, К.В. Карпов, К.Д. Кириченко, В.В. Королев, И.Б. Мамай, Е.В. Милованович, А.А. Митрофанов, Н.А. Серебрянская, Т.С. Юрова

И86 Всероссийская междисциплинарная олимпиада школьников 8-11 класса «Национальная технологическая олимпиада». Учебно-методическое пособие
Том 13 **Искусственный интеллект**
—М.: ООО «ВАШ ФОРМАТ», 2024. — 134 с.

ISBN 978-5-00147-603-0

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
10.00.00 Информационная безопасность

ISBN 978-5-00147-603-0

УДК 373.5.016:004.8
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.

Искусственный интеллект

Цель профиля «Искусственный интеллект» — развитие у школьников прикладных навыков в сфере искусственного интеллекта. Профиль знакомит школьников с методами машинного и глубоко обучения для решения актуальных для науки и бизнеса задач.

Соревнования по профилю способствуют повышению уровня обеспечения российского рынка технологий искусственного интеллекта квалифицированными кадрами за счет повышения привлекательности конкурсов и олимпиад, направленных на развитие интеллектуальных и творческих способностей обучающихся, в соответствии с Национальной стратегией развития искусственного интеллекта Российской Федерации, а также вовлекает школьников в сферу искусственного интеллекта и ориентирует на профессиональное развитие в ней.

В 2023/24 учебном году участникам Национальной технологической олимпиады по профилю «Искусственный интеллект» было предложено погрузиться в задачу создания алгоритма, который найдет молекулы, потенциально способные стать лекарствами, использующими радиоактивность для диагностики и терапии онкозаболеваний.

Основная задача отборочных этапов олимпиады — выявить наиболее способных к решению подобных задач школьников и через образовательную составляющую развить у школьников прикладные навыке в области машинного обучения.

Первый отборочный (индивидуальный) дистанционный этап состоял из предметного тура по математике и информатике, а также инженерного тура, в рамках которого участникам было предложено решить базовую задачу машинного обучения по определению наименования упражнения для оценки постурального и кинетического тремора у пациентов с диагностированной болезнью Паркинсона. Диагностика болезни Паркинсона требует от врачей знаний и опыта, при этом используется шкала MDS UPDRS, не исключающая субъективность врача. Объективное оценивание двигательных нарушений при диагнозе и корректировке лечения важно, для чего разрабатывается платформа PN expert, которая анализирует видео с упражнениями пациентов, помогая объективно оценить их состояние. Исследователи призывают разработать алгоритмы для PN expert, которые могут точно измерять тремор и классифицировать выполненные упражнения. Таким образом, задача имеет целью проверить навыки работы участников с алгоритмами классификации для разработки модели, способной автоматически классифицировать упражнения пациентов на видеозаписях. Задача отборочного этапа проверяла компетенции участников в обработке данных и применении алгоритмов машинного обучения, а также способность погружаться в предметную сферу и применять формулы и классические алгоритмы в решении задач по искусственному интеллекту. Задача помогала оценить участников по их знаниям и навыкам в области программирования, медицины, обработки табличных данных, математической статистики и применении технических решений для решения реальных проблем.

Предметный тур определял уровень подготовки школьников по предметам: математика и информатика (программирование). Задачи по информатике относятся к разделам: алгоритмы, программирование и методы оптимизации. Здесь школьники

должны были продемонстрировать простейшие навыки составления и отладки программ, обрабатывающих массивы данных, и понимание таких тем, как комбинаторика, операции со строками, матричный анализ, теория графов. Задачи по математике проверяли у участников знания по алгебре, комбинаторике, теории вероятности и математической статистике. Таким образом, задачи предметного тура выявляли у участников знания, необходимые для решения задач следующего (второго) и заключительного этапов.

Дополнительно для участников был проведен онлайн-интенсив AI-ARROW. Участие в интенсиве подразумевало быстрое погружение в сферу ИИ, начиная с основ программирования на Python с использованием профильных библиотек и заканчивая подходами к разработке нейросетей с использованием технологий компьютерного зрения и обработки естественного языка.

В рамках второго отборочного этапа участникам предстояло решить индивидуальную и командную задачи. Индивидуальная задача второго отборочного этапа была направлена на предсказание свойств малых органических молекул. Используя предоставленное текстовое представление молекул (`SMILES`), участникам необходимо было предсказать их липофильность. Исходные данные были представлены в табличном виде, где первым столбцом были представлены строковые представления молекул (например: «CC(=O)OC$_1$=CC=CC=C$_1$C(=O)O») а вторым — соответствующие безразмерные значения липофильности (например, «1.19»). В процессе тестирования, код участников должен был обрабатывать аналогичные табличные данные, не содержащие, однако, столбца с правильными ответами. Командная задача второго отборочного этапа была посвящена направленному дизайну молекул. Участникам предлагалось решить задачу направленного дизайна новых органических соединений. Для решения необходимо реализовать функцию, принимающую на вход список `SMILES` строк, и возвращающую список той же длины, отличающийся от исходного на один элемент, то есть простейший эволюционный алгоритм предполагает модификацию `SMILES` строк из исходного списка путем замены, добавления и удаления отдельных символов. Сгенерированные таким образом `SMILES` строки проходят формальную проверку на соответствие реальной химической структуре, затем оценивается соответствующее значение целевой функции (липофильности) на предмет нахождения в заданном диапазоне.

В целях подготовки финалистов к заключительному этапу дополнительно был проведен образовательный хакатон, в рамках которого участникам предоставляется возможность погрузиться в увлекательный процесс разработки проекта для интерпретации химических формул. Задача состояла в том, чтобы создать систему, способную распознавать изображения органических молекул и переводить их в формат `SMILES` — уникальный язык, преобразующий сложные химические соединения в легко читаемые текстовые строки. В рамках хакатона финалисты получили возможность отточить необходимые для решения задачи заключительного этапа компетенции, а участники, не прошедшие в заключительный этап, — попробовать себя в решении задач более высокого уровня.

Заключительный этап НТО по профилю «Искусственный интеллект» состоял из командного инженерного и индивидуального предметного туров.

Предметный тур проводился по двум предметам: информатика и математика, и проверял предметные знания, необходимые в решении задач на анализ данных. Баллы, набранные в индивидуальном предметном туре, то есть знания и умения решать олимпиадные задачи в важных для направления разделах математики и

информатики, влияют на индивидуальный результат участников.

Задача командного инженерного тура заключительного этапа предполагала поиск новых радиофармпрепаратов — лекарств, использующих радиоактивность для диагностики и терапии онкозаболеваний. Для успешной работы эти лекарства должны доставлять медицинский радионуклид к раковой опухоли, и финалистам предстояло найти именно те молекулы, которые смогут сделать это наиболее эффективно. Лекарство не должно оказаться опаснее болезни, поэтому на молекулы, которые будут предлагать участники, будут накладываться дополнительные ограничения. В оригинале задача относится к области фармацевтики и связана со знаниями биохимии поведения раковой опухоли. Однако, разбиение ее на отдельные этапы приводит к химической задаче связывания медицинского радионуклида. При этом исключительно химическое ее решение является очень ресурсозатратным, и использование технологий искусственного интеллекта помогает найти потенциальные лекарства значительно быстрее. Наилучшие решения способна дать интеграция знания химии и биологии с одной стороны и технологий искусственного интеллекта с другой.

По сравнению с предыдущими этапами, участники должны были продемонстрировать не только компетенции в области машинного обучения и искусственного интеллекта, но и использовать полученные ранее знания в области химии, а также наработки с предыдущих этапов, для наиболее эффективного решения поставленной задачи, что предполагало активную командную работу.

Таким образом, учащиеся 8–11 классов, прошедшие все этапы НТО, демонстрируют понимание основных операций, необходимых для построения модели машинного обучения: подготовка и анализ данных с целью выявления признаков, необходимых для решения поставленной задачи, построение самой модели и интерпретация результатов ее работы для оценки качества в рамках специализированной доменной области (химия, поиск лекарств).

Для того чтобы сделать это возможным в ходе олимпиады проводился цикл образовательных и отборочных мероприятий.

С момента регистрации для подготовки участникам были доступны:

- курс по машинному обучению (основной уровень) от Академии ИИ: https://stepik.org/course/125587/promo;
- курс по машинному обучению (продвинутый уровень) от Академии ИИ: https://stepik.org/course/134942/promo;
- материалы Академии искусственного интеллекта для школьников: https://ai-academy.ru/.

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 программирование

Первая волна. Задачи 8–11 класса

Задача П.1.1.1. Поздравление в конверте (10 баллов)

Темы: задачи для начинающих.

Условие

Алиса хочет поздравить Боба с днем рождения. Она взяла прямоугольный лист бумаги размера $a \times b$ и написала на нем поздравление в стихах. У Алисы есть красивый конверт тоже прямоугольной формы размером u на v. Алиса хочет положить свое поздравление в этот конверт. Однако лист может не войти в конверт. В этом случае Алиса готова сложить лист пополам вдоль одной из сторон, чтобы поместить его в конверт. Обратите внимание, что Алиса может сделать не более одного сгиба. Лист можно поворачивать, но одна из сторон листа должна быть параллельной одной из сторон конверта.

Напишите программу, которая определит, сможет ли Алиса уложить лист в конверт по указанным правилам.

Мы будем считать, что лист входит в конверт, если сторона листа будет строго меньше соответствующей стороны конверта.

Формат входных данных

На вход в первой строке подается два натуральных числа a и b — длины сторон листа. Во второй строке на вход подаются натуральные числа u и v — размеры конверта. Все числа не превосходят 1000.

В языке Python прочитать два целых числа, записанных в одной строке можно, используя следующий код.

```
a, b = map(int, input().split())
```

Формат выходных данных

Если поздравление можно вложить в конверт без сгиба, то следует вывести число 0. Иначе, если поздравление можно вложить в конверт сделав один сгиб, то следует вывести число 1. В остальных случаях следует вывести число -1.

Методика проверки

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

Примеры

Пример №1

Стандартный ввод
120 200 130 250
Стандартный вывод
0

Пример №2

Стандартный ввод
120 200 110 130
Стандартный вывод
1

Пример №3

Стандартный ввод
400 100 200 150
Стандартный вывод
-1

Решение

В этой задаче требуется аккуратно написать требуемые по условию логические выражения. Их запись существенно упростится, если упорядочить длины так, чтобы всегда имел место инвариант $a \leqslant b$ и $u \leqslant v$.

Тогда для проверки возможности вложения листа в конверт меньшую сторону листа следует всегда сравнивать с меньшей стороной конверта.

Для проверки возможности вложения листа в конверт после сгиба надо поочередно поделить меньшую и большую сторону на два и использовать такую же проверку. Следует не забыть, что после деления длины большей стороны на два, она может стать меньше, чем меньшая сторона.

Пример программы-решения

Ниже представлено решение на языке Python 3.

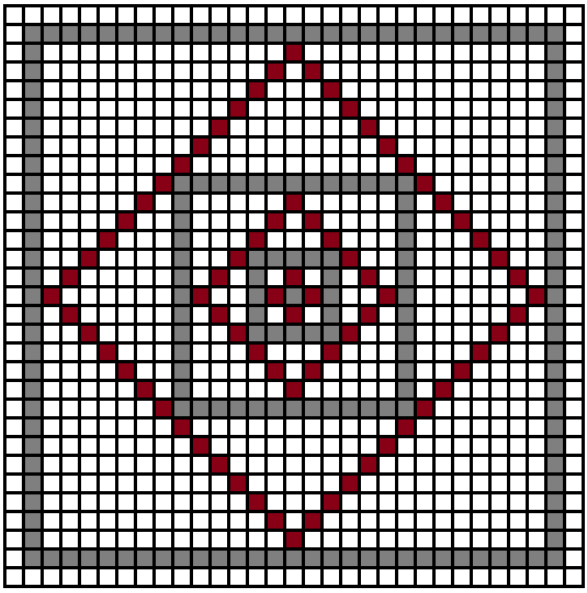
```python
a, b = sorted(list(map(int, input().split())))
u, v = sorted(list(map(int, input().split())))
if a<u and b<v:
    print(0)
elif a/2<u and b<v or a<u and b/2<v or b/2<u and a<v:
    print(1)
else:
    print(-1)
```

Задача II.1.1.2. Квадраты (15 баллов)

Темы: задачи для начинающих, комбинаторика.

Условие

У Алисы и Боба есть прямоугольный лист бумаги в клеточку. Они по очереди рисуют квадраты, закрашивая некоторые из клеточек. Алиса рисует квадраты, ориентируя их вдоль сторон листа, а Боб — под углом в 45 °. При этом Алиса рисует первый квадрат из одной клеточки, а каждый новый квадрат описывается вокруг предыдущего. Для лучшего понимания смотрите рисунок. Серым цветом на нем нарисовано четыре квадрата Алисы, а коричневым нарисовано три квадрата Боба. Всего семь квадратов.



Алиса и Боб вместе нарисовали n квадратов. Напишите программу, которая определит, сколько клеточек на листе бумаги будет закрашено.

Формат входных данных

На вход подается единственное натуральное число n — количество квадратов, $1 \leqslant n \leqslant 100$.

Формат выходных данных

Выведите одно натуральное число — суммарное количество закрашенных клеточек.

Методика проверки

Программа проверяется на 15-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
7
Стандартный вывод
253

Пример №2

Стандартный ввод
2
Стандартный вывод
5

Решение

Заметим, что по условию задачи количество квадратов не превышает 100, поэтому посчитаем, из скольких клеточек состоит каждый квадрат, и просуммируем полученные значения в цикле.

Обратим внимание на количество клеточек на стороне квадрата. Легко заметить и доказать, что если предыдущий квадрат был серый, и его сторона содержала k клеточек, то сторона следующего за ним коричневого квадрата будет содержать $k+1$ клеточку. Если же предыдущий квадрат был коричневым, и его сторона содержала k клеточек, то сторона следующего серого квадрата будет содержать $2k+1$ клеточку.

В приведенной ниже программе в переменной `ln` хранится количество клеточек, из которых состоит одна сторона текущего квадрата. В переменной `ans` накапливается сумма.

Пример программы-решения

Ниже представлено решение на языке Python 3.

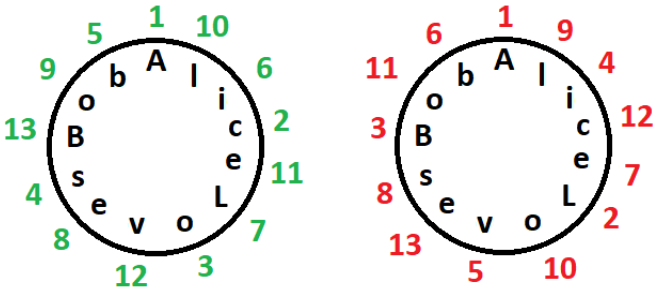
```
n = int(input())
ans = 1
ln = 1
for i in range(n):
    ans += (ln - 1) * 4
    if i%2==0:
        ln += 1
    else:
        ln = 2 * ln + 1
print(ans)
```

Задача П.1.1.3. Две строки (15 баллов)

Темы: строки, структуры данных.

Условие

У Алисы и Боба есть секретная информация, которая записана в виде строки s. Чтобы сохранить секрет Алиса сделала перестановку символов в строке по следующему правилу. Она записала все символы строки по кругу, потом записала в ответ первый символ и далее стала выписывать символы из кольца через два.



Рассмотрим пример. Пусть секретная строка — *AliceLovesBob*. Алиса запишет эту строку, как показано на рисунке. Далее она выпишет первую букву строки A, пропустит два следующих символа, напишет букву c, пропустит еще два символа, напишет букву o и так далее по кругу. В результате у нее будет записана строка *AcosbiLeolevB*. Номера на рисунке слева соответствуют последовательности перечисления букв Алисой.

Боб шифрует эту же строку таким же алгоритмом, однако, в отличие от Алисы, он пропускает по четыре буквы, а не по две. Номера на рисунке справа соответствуют последовательности перечисления букв Бобом. Таким образом Боб получит строку *ALBivbeslooce*.

Боб был небрежен и потерял зашифрованную строку, однако у него есть строка, зашифрованная Алисой. Напишете программу, которая по зашифрованной строке Алисы найдет зашифрованную строку Боба.

Формат входных данных

На вход подается одна непустая строка — шифр Алисы. Строка состоит только из строчных и заглавных символов латиницы. Длина строки не превосходит 1000 и не кратна трем и пяти.

Формат выходных данных

Выведите одну строку — шифр Боба.

Методика проверки

Программа проверяется на 15-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
`AcosbiLeolevB`
Стандартный вывод
`ALBivbeslooce`

Решение

Решение задачи состоит из двух частей. В первой части требуется восстановить исходную строку по коду Алисы. Для этого можно сделать список из символов нужной длины и каждый i-тый символ из кодовой строки записывать в позицию $3i \mod n$, где n — длина строки. Операция взятия остатка от деления здесь используется для движения по кольцу.

Во второй части при помощи аналогичного приема полученная строка кодируется по обратной формуле с множителем 5.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
s = input()
n = len(s)
tmp = [''] * n
ans = ''
for i in range(n):
    tmp[(i * 3) % n] = s[i]
for i in range(n):
    ans += tmp[(i * 5) % n]
print(ans)
```

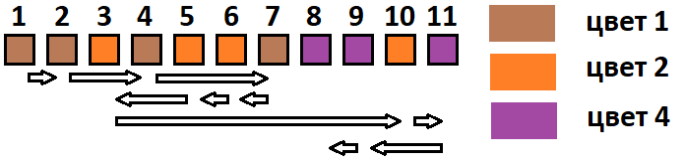
Задача П.1.1.4. Покраска кубиков (30 баллов)

Темы: реализация, сортировки, структуры данных, динамическое программирование, комбинаторика.

Условие

На ленте в один ряд расставлено n кубиков. Каждый кубик необходимо покрасить в определенный цвет. Все цвета пронумерованы числами от 1 до k. Покраска выполняется роботом, который может перемещаться от одного кубика к другому и красить один выбранный кубик в определенный цвет. Конструктивно робот устроен так, что он может сначала красить кубики в цвет номер 1, затем в цвет номер 2 и так далее в порядке возрастания. Вернуться к цвету с меньшим номером после того, как был выбран цвет с большим номером, нельзя.

Среди всего прочего робот тратит время на перемещение между кубиками. Будем считать, что перемещение между двумя соседними кубиками занимает ровно одну с. Требуется составить последовательность действий для робота, в которой время, затраченное на перемещение между кубиками, будет минимально возможным.



Рассмотрим пример на схеме. Имеется 11 кубиков, которые надо покрасить в три цвета с номерами $1, 2, 4$. Робот начнет движение от кубика номер 1 направо к кубику номер 2 (1 с), далее к кубику с номером 4 (2 с), и наконец к кубику номер 7 (3 с). После этого робот поменяет цвет. Будет выгоднее, если робот начнет сначала двигаться влево. Он пройдет к кубику номер 6 (1 с), далее к кубику номер 5 (1 с), далее к кубику номер 3 (2 с). После этого он развернется и пойдет к кубику номер 10 (7 с). Далее робот сменит цвет на 4, так как нет кубиков, которые требуется красить в цвет 3, и пойдет направо к кубику номер 11 (1 с). После этого он развернется и пойдет к кубику номер 9 (2 с) и наконец к кубику номер 8 (1 с). В этот момент робот остановит работу, затратив на перемещения суммарно 21 с.

Обратите внимание, что по условию задачи робот может выбрать цвет 4 только после цвета 2. Существуют другие возможные маршруты движения робота, но они займут больше времени.

Напишите программу, которая найдет минимально возможное суммарное время перемещения робота между кубиками до момента пока все они не будут покрашены. Изначально робот находится у кубика номер 1.

Формат входных данных

На вход в первой строке подается два натуральных числа n и k — количество кубиков и количество цветов, $1 \leqslant n, k \leqslant 100000$. Во второй строке на вход подается n натуральных чисел $c_1, c_2, \ldots, c_n$, где c_i — требуемый цвет i-того кубика, $1 \leqslant c_i \leqslant k$.

Формат выходных данных

Выведите одно число — минимально возможное время перемещения между кубиками.

Методика проверки

Программа проверяется на 60-ти тестах. Прохождение каждого теста оценивается в 0,5 балла. Тест из условия задачи при проверке не используется. Ниже в таблице приведены возможные тестовые случаи.

Тестовый случай	**Номера тестов**
$n = k$; $n \leqslant 1000$; все c_i различны.	1–8
$n = k$; $n \leqslant 100000$; все c_i различны.	9–20
$k = 3$; $n \leqslant 1000$	21–24
$k = 4$; $n \leqslant 1000$	25–30
$k = 5$; $n \leqslant 1000$	31–40
$n \leqslant 1000$	41–50
$n \leqslant 100000$	51–60

Примеры

Пример №1

Стандартный ввод
`11 4` `1 1 2 1 2 2 1 4 4 2 4`
Стандартный вывод
`21`

Решение

Данная задача может быть решена методом динамического программирования. Пусть, начиная покраску кубиков в цвет i, робот находится в некоторой точке x_i, причем самый левый из кубиков этого цвета находится в точке l_i, а самый правый — в r_i. Тогда оптимальным будет один из двух вариантов: из точки x_i пойти в r_i, а потом в l_i или сначала пойти в l_i, а потом в r_i. Таким образом, робот закончит покраску кубиков определенного цвета либо в точке l_i, либо в r_i, причем в первом случае время перемещения робота увеличится на $|x_i - r_i| + r_i - l_i$ с, а во втором — на $|x_i - l_i| + r_i - l_i$ с.

Обозначим за $f_l(i)$ и $f_r(i)$ — оптимальное время покраски кубиков в первые i цветов при условии, что робот остановится в точке l_i и r_i соответственно. Тогда можно определить следующие формулы:

$$
\begin{aligned}
&f_l(0) = 0;\\
&f_r(0) = 0;\\
&f_l(i) = r_i - l_i + \min(f_l(i-1) + |l_{i-1} - r_i|, f_r(i-1) + |r_{i-1} - r_i|);\\
&f_r(i) = r_i - l_i + \min(f_l(i-1) + |l_{i-1} - l_i|, f_r(i-1) + |r_{i-1} - l_i|).
\end{aligned}
$$

В данных формулах находится время для двух вариантов перемещения: от самого левого и от самого правого кубика предыдущего цвета, после чего из полученных значений выбирается минимальное.

Программа будет содержать два цикла. В первом цикле для каждого цвета определяется местоположение самого левого и самого правого кубика, а во втором — выполняются вычисления по формулам.

При реализации программы надо учесть, что некоторые цвета могут отсутствовать. Поэтому в переменные `l` и `r` записываются координаты самого левого и правого кубика для предыдущего цвета.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n, k = map(int, input().split())
left = [n + 1] * (k + 1)
right = [0] * (k + 1)
i = 1
for col in map(int, input().split()):
    left[col] = min(left[col], i)
    right[col] = max(right[col], i)
    i += 1
l, r, fl, fr = 1, 1, 0, 0
for i in range(1, k + 1):
    if right[i] > 0:
        dist = right[i] - left[i]
        tl = dist + min(fl + abs(l - right[i]),
                        fr + abs(r - right[i]))
        tr = dist + min(fl + abs(l - left[i]),
                        fr + abs(r - left[i]))
        l, r, fl, fr = left[i], right[i], tl, tr
print(min(fl, fr))
```

Задача II.1.1.5. Обработка запросов (30 баллов)

Темы: реализация, структуры данных, два указателя, двоичный поиск.

Условие

Алиса проектирует вычислительную систему, предназначенную для обработки большого числа однотипных запросов. Проектируемая система будет содержать некоторое количество одинаковых процессоров. В каждый момент времени каждый процессор может быть либо свободен, либо занят обработкой ровно одного запроса. Продолжительность обработки является одинаковой для всех запросов и составляет s мс. Система должна работать в режиме реального времени, то есть каждый поступивший запрос должен незамедлительно передаваться на обработку любому свободному процессору.

Алиса хочет понять, сколько процессоров должна содержать вычислительная система. Для этого она собрала статистические данные о работе подобных систем в прошлом. Каждый набор данных содержит n чисел $t_1, t_2, \ldots, t_n$, где t_i — момент

поступления запроса с номером i. Числа в наборе могут повторяться, однако они упорядочены по неубыванию. Если некоторый процессор приступил к обработке некоторого запроса в момент t_i, то в момент $t_i + s$ он сможет начать обрабатывать новый запрос.

Набор может содержать достаточно большой объем данных, поэтому от вас требуется написать программу, которая определит, какое минимальное число процессоров должно быть в вычислительной системе, чтобы все запросы были обработаны в момент их поступления.

Формат входных данных

На вход в первой строке подается два натуральных числа n и s — количество запросов и время обработки одного запроса, $1 \leqslant n \leqslant 200000$, $1 \leqslant s \leqslant 10^9$. Во второй строке записаны целые неотрицательные числа t_1, t_2, …, t_n, задающие моменты времени поступления запросов, $0 \leqslant t_1 \leqslant t_2 \leqslant \cdots \leqslant t_n \leqslant 10^9$.

Формат выходных данных

Вывести одно число — минимальное количество процессоров, которое позволит обработать все запросы в момент их поступления.

Методика проверки

Программа проверяется на 30-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тесты из условия задачи при проверке не используются. Ниже в таблице приведены возможные тестовые случаи.

Тестовый случай	**Номера тестов**
$n \leqslant 1000$; для всех t_i выполняется одно из двух условий: либо $t_i = t_{i+1}$, либо $t_i + s \leqslant t_{i+1}$.	1–5
$n \leqslant 1000$	6–15
$n \leqslant 200000$	16–30

Примеры

Пример №1

Стандартный ввод
`9 30` `90 90 90 120 120 120 120 200 200`

Стандартный вывод
`4`

Пример №2

Стандартный ввод
10 30 0 25 110 125 125 130 140 140 140 155
Стандартный вывод
6

Пояснения к примерам

Первый пример соответствует первому тестовому случаю. В момент времени 120 приходит сразу четыре запроса, для обработки которых потребуется четыре процессора.

Расписание выполнения запросов во втором примере можно представить в следующей таблице.

Время поступления запроса	**Время завершения обработки запроса**	**Номер процессора**
0	30	1
25	55	2
110	140	1
125	155	2
125	155	3
130	160	4
140	170	1
140	170	5
140	170	6
155	175	2

Пять процессоров для своевременной обработки всех запросов будет уже недостаточно.

В задаче требуется найти такое число k, что для всех $i \leqslant n - k$ выполняется неравенство $t_{i+k} - t_i \leqslant s$. Действительно, пусть это утверждение имеет место. Тогда процессор, выполнявший задание с номером i, всегда сможет выполнить задание с номером $i+k$, и все задания можно выполнить своевременно, выдавая их процессорам по кругу. С другой стороны, пусть это утверждение не выполняется, то есть найдется такой номер j, что $t_{j+k} - t_j < s$. Тогда в момент времени t_{j+k} все k процессоров будут заняты исполнением запросов с номерами от j до $j + k - 1$, и все запросы не смогут быть выполнены своевременно.

Найти число k, для которого выполняется указанное утверждение можно при помощи двоичного поиска или метода двух указателей. Вариант решения с использованием двух вложенных циклов наберет лишь часть баллов, так как будет превышать ограничение по времени работы.

Метод двух указателей основан на использовании двух переменных `left` и `right`, которые используются в качестве индексов в массиве. Цикл строится таким образом, чтобы для каждого значения `left` находить минимальное значение `right` при котором `t[right]-t[left]>=s`.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n, s = map(int, input().split())
t = list(map(int, input().split()))
ans = 1
left = 0
right = 0
while right < n:
    if t[right] - t[left] >= s:
        left += 1
    else:
        right += 1
    ans = max(ans, right - left)
print(ans)
```

Вторая волна. Задачи 8–11 класса

Задача II.1.2.1. Три мешка конфет (10 баллов)

Темы: задачи для начинающих, реализация.

Условие

Алиса и Боб получили в подарок три мешка конфет и они хотят поделить их поровну. Для этого Алиса возьмет некоторое количество конфет из каждого мешка, а остальные конфеты отдаст Бобу. Возможно, что из некоторого мешка Алиса возьмет все конфеты или не возьмет ни одной. Известно, что суммарное количество конфет является четным числом.

Напишите программу, которая определит, сколько конфет Алиса должна взять из каждого мешка, чтобы у нее оказалось ровно половина всех конфет. Программа может вывести любой правильный ответ.

Формат входных данных

На вход в первой строке подается три натуральных числа a, b и c — количество конфет в каждой кучке, $1 \leqslant a, b, c \leqslant 1000$.

В языке Python прочитать три целых числа, записанных в одной строке можно, используя следующий код.

```
a, b, c = map(int, input().split())
```

Формат выходных данных

Выведите в одной строке через пробел три целых неотрицательных числа — количество конфет, которое возьмет Алиса из каждого мешка.

Методика проверки

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

Примеры

Пример №1

Стандартный ввод
10 5 5
Стандартный вывод
7 3 3

Пояснения к примерам

Ответ 7 3 0 удовлетворяет всем требованиям. Но существует и множество других вариантов, например, 7 2 1 или 0 5 5.

Решение

Существует много способов составить требуемый набор чисел. Например, можно заметить, что если сумма трех чисел четная, то хотя бы одно из слагаемых тоже обязательно четное. Тогда это слагаемое можно поделить на два, еще одно поделить на два с округлением вниз, а последнее — с округлением вверх.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
a, b, c = map(int, input().split())
if a % 2 == 0:
    print(a // 2, b // 2, (c + 1) // 2)
else:
    print((a + 1) // 2, b // 2, c // 2)
```

Задача II.1.2.2. Трехцветная сортировка (15 баллов)

Темы: задачи для начинающих, структуры данных.

Условие

У Алисы есть упорядоченный набор карточек, каждая из которых раскрашена в один из трех цветов: красный, зеленый, синий. Кроме того, на каждой карточке записано некоторое натуральное число. Алиса хочет выполнить сортировку чисел, чтобы сначала шли все числа на красных карточках, далее — на зеленых и наконец — на синих. При этом взаимное расположение карточек одного цвета не должно измениться. Например, если в исходном наборе было две красных карточки с числами 20

и 10, причем карточка с числом 20 располагалась раньше, чем карточка с числом 10, то после упорядочивания 20 по-прежнему должна находиться раньше, чем 10.

Напишите программу, которая отсортирует карточки в требуемом порядке.

Формат входных данных

На вход в первой строке подается последовательность символов $c_1, c_2, \cdots, c_n$, где c_i задает цвет i-той карточки и может принимать одно из трех значений r, g или b. Каждый из символов обозначает определенный цвет: r — красный, g — зеленый, b — синий. Символы записаны без пробелов и других разделителей, $1 \leqslant n \leqslant 1000$.

Во второй строке записана последовательность натуральных чисел $a_1, a_2, \cdots, a_n$, где a_i задает число, записанное на i-той карточке. Все числа различны и не превосходят n.

Формат выходных данных

В одной строке через пробел вывести требуемую последовательность чисел после сортировки.

В языке Python для вывода чисел в цикле на одной строке через пробел можно использовать следующую команду.

```
print(x, end=' ')
```

Методика проверки

Программа проверяется на 15-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
bbb 3 1 2
Стандартный вывод
3 1 2

Пример №2

Стандартный ввод
rgrg 4 1 2 3
Стандартный вывод
4 2 1 3

Пример №3

Стандартный ввод
brrg 1 2 3 4
Стандартный вывод
2 3 4 1

Пояснения к примерам

В первом примере все карточки одного цвета, поэтому упорядочивать нечего.

Во втором примере карточки 4 и 2 красного цвета, поэтому они окажутся в начале, сохранив взаимное расположение. Карточки 1 и 3 зеленого цвета, поэтому они сдвинутся в конец, также сохранив взаимное расположение.

В третьем примере в начале последовательности будут красные карточки 2 и 3, далее зеленая 4, далее синяя 1.

Решение

Для решения этой задачи достаточно сохранить цвета и номера карточек в списке. Далее в трех циклах вывести сначала номера красных карточек, потом — зеленых, и наконец, — синих.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
s = input()
p = list(map(int, input().split()))
for a in 'rgb':
    for i in range(len(s)):
        if s[i] == a:
            print(p[i], end = ' ')
```

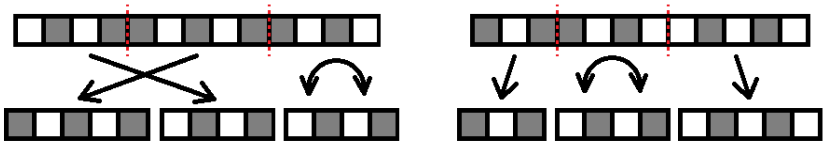
Задача II.1.2.3. Черно-белая полоска (15 баллов)

Темы: задачи для начинающих, реализация, строки.

Условие

У Алисы есть полоска бумаги, расчерченная на клеточки. Полоска имеет ширину в одну клеточку и длину в n клеточек. Алиса хотела раскрасить каждую клеточку в белый или черный цвет так, чтобы клеточки разных цветов чередовались. Но после того, как вся полоска была раскрашена, выяснилось, что Алиса ошиблась, и существует ровно две непересекающихся пары соседних клеточек, раскрашенных в один цвет. Отметим, что *три и более клеток подряд не могут иметь один цвет*. Чтобы исправить свои ошибки, Алиса решила разрезать полоску в двух местах, переставить

и, возможно, развернуть полученные три части, а затем склеить их. На рисунке ниже показаны два примера разрезания и склейки полоски.



В примере на картинке слева исходная полоска разрезается на три части и склеивается в следующем порядке. Кусочек из середины от с номерами клеток из диапазона $[5; 9]$ становится самым левым. Далее к нему пристыковывается кусочек с номерами клеток $[1; 4]$. И, наконец, справа пристыковывается кусочек с номерами клеток $[10; 13]$, который при этом разворачивается на 180°. В результате будет получена полоска из клеток с чередующимися цветами.

В примере на картинке справа кусочки полоски остаются на своих местах, но средняя полоска с номерами клеток $[4; 7]$ разворачивается на 180°.

Напишите программу, которая определит, сможет ли Алиса указанным способом сделать полоску из клеточек чередующихся цветов и, если это возможно, то составит схему разрезания существующей полоски на три кусочка и склейки этих кусочков. Полученная полоска может начинаться как с клетки белого, так и черного цвета. Если требуемую полоску можно получить различными способами, то в качестве ответа можно взять любой из них.

Формат входных данных

На вход подается одна строка, описывающая вид исходной полоски. Строка состоит из символов w и b, обозначающих клетку белого и черного цвета соответственно. Длина строки не превосходит 1000. Гарантируется, что строка имеет вид, описанный в условии задачи.

Формат выходных данных

Если составить полоску из клеток чередующихся цветов невозможно, то программа должна вывести единственное слово *no*. В противном случае вывод должен содержать ровно три строки, каждая из которых описывает кусочек исходной ленты в виде трех чисел. Первое и второе число задают номера начальной и конечной клетки кусочка соответственно. Третье число может иметь одно из двух значений — 0 или 180 в зависимости от того, поворачивается кусочек на 180° или нет. Строки должны следовать в порядке склейки кусочков слева направо.

Методика проверки

Программа проверяется на 15-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
wbwbbwbwbbwbw
Стандартный вывод
5 9 0 1 4 0 10 13 180

Пример №2

Стандартный ввод
bwbbwbwwbwbw
Стандартный вывод
1 3 0 8 12 180 4 7 0

Пример №3

Стандартный ввод
bbwbbw
Стандартный вывод
no

Пример программы-решения

Ниже представлено решение на языке Python 3.

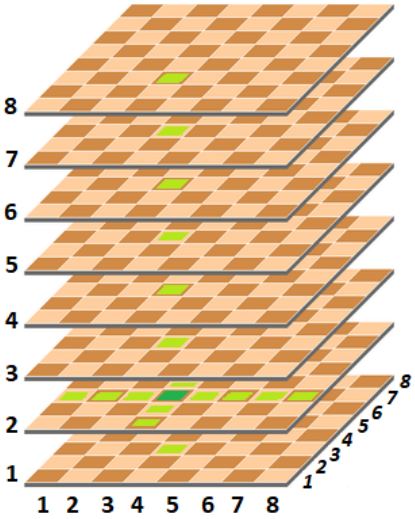
```
s = input()
n = len(s)
x = []
for i in range(1, n):
    if s[i] == s[i-1]:
        x.append(i)
if s[x[0]] != s[x[1]]:
    print(1, x[0], 0)
    print(x[0] + 1, x[1], 180)
    print(x[1] + 1, n, 0)
elif s[x[0]] == s[0] or s[x[0]] == s[-1]:
    print('no')
else:
    print(x[0] + 1, x[1], 0)
    print(1, x[0], 0)
    print(x[1] + 1, n, 180)
```

Задача II.1.2.4. Расстановка ладей на трехмерной шахматной доске (30 баллов)

Темы: реализация, сортировки, структуры данных, динамическое программирование, комбинаторика.

Условие

Алиса и Боб учатся пространственному воображению и решают для этого математические головоломки на трехмерной шахматной доске размера n. Такая доска состоит из n двумерных квадратных досок, расположенных друг над другом. На рисунке изображена трехмерная шахматная доска размера 8.



Каждую клетку трехмерной доски можно задать тремя целыми числами из диапазона $[1; n]$: порядковым номером двумерной доски, номером вертикали на двумерной доске и номером горизонтали. Например, клетка, выделенная на рисунке темно-зеленым цветом, задается тройкой чисел $(2, 4, 3)$.

Трехмерная шахматная ладья ходит по двумерной доске по стандартным правилам, то есть за один ход может переместиться на любую клетку в той же вертикали или горизонтали, где она находится. Вместе с тем трехмерная ладья может за один перейти на любую другую доску в клетку с такой же двумерной координатой. На рисунке светло-зеленым цветом показаны клетки в которые может перейти ладья из клетки с координатами $(2, 4, 3)$. В этом случае говорят, что ладья бьет эти клетки.

Алиса и Боб уверены, что на трехмерной шахматной доске размера n можно расставить n^2 ладей так, что они не будут бить друг друга, но никак не могут понять принцип расстановки.

Напишите программу, которая найдет любую допустимую расстановку ладей на трехмерной шахматной доске размера n так, чтобы они не били друг друга.

Формат входных данных

На вход подается единственное натуральное число n — размер доски, $1 \leqslant n \leqslant 30$.

Формат выходных данных

Выведите координаты n^2 клеток, на которых будут расположены ладьи. Три координаты каждой клетки выводятся через пробел в отдельной строке. Порядок перечисления клеток может быть произвольным.

Методика проверки

Программа проверяется на 30-ти тестах. Номер теста совпадает с числом n.

Примеры

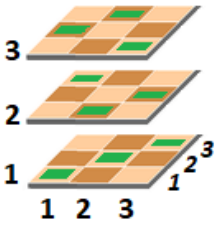
Пример №1

Стандартный ввод
3

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

Пояснения к примеру

Расстановка ладей из примера показана на рисунке ниже.



Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n = int(input())
for i in range(n):
    for j in range(n):
        print(i + 1, j + 1, (i + j) % n + 1)
```

Задача II.1.2.5. Удаление скобок (30 баллов)

Темы: математика, строки, рекурсивные алгоритмы.

Условие

Боб любит формализм во всем, включая запись математических выражений, поэтому при их записи он ставит скобки так, чтобы каждая операция выделялась своей парой скобок. Например, выражение $(a + b + c) * d$ он запишет как $(((a + b) + c) * d)$ или как $((a + (b + c)) * d)$, а выражение $a * b + c * d$ как $((a * b) + (c * d))$. Таким образом, количество пар скобок в выражениях Боба всегда равно количеству операций, а для операции, которая выполняется последней, пара скобок всегда ограничивает все выражение.

Алиса считает такой перфекционизм избыточным и старается ставить скобки только там, где они нужны для правильных вычислений. Например, в выражении $(a + b + c) * d$ скобки убрать уже нельзя, поскольку выражение $a + b + c * d$ по математическим правилам задает другой порядок применения операций, и результаты вычисления этих двух выражений могут различаться. А вот в выражении $a + (b + c)$ скобки убрать уже можно, поскольку для сложения имеет место сочетательное свойство или, как говорят математики, аксиома ассоциативности. Также можно убрать скобки и в выражении $(a * b) + (c * d)$, поскольку по договоренностям умножение выполняется раньше, чем сложение.

Напишите программу, которая перепишет выражение, записанное Бобом, в тот вид, который нравится Алисе. Обратите внимание, что программа должна просто убрать все избыточные скобки. Другие преобразования делать нельзя. Полученное выражение должно иметь результат вычислений, совпадающий с результатом исходного выражения, при любых значениях параметров.

Формат входных данных

На вход в единственной строке поступает правильно записанное арифметическое выражение, состоящие из имен параметров, скобок и операций «+» и «*». Каждый параметр записывается в виде одной строчной буквы латиницы. Имена параметров не повторяются и встречаются в алфавитном порядке, таким образом, количество операций не превосходит 25. Выражение содержит как минимум одну операцию. Каждая операция в выражении выделяется своей парой скобок, как записано в условии задачи.

Формат выходных данных

Программа должна вывести исходное выражение без избыточных скобок. Порядок следования параметров в ответе должен совпадать с порядком в исходном выражении. В частности, это означает что для теста $(a + b)$ ответ $b + a$ будет считаться ошибочным.

Методика проверки

Программа проверяется на 30-ти тестах. Прохождение каждого теста оценивается в 1 балл. Выражения в первых девяти тестах содержат не более двух операций. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
`(a+b)`
Стандартный вывод
`a+b`

Пример №2

Стандартный ввод
`((a+(b+c))*d)`
Стандартный вывод
`(a+b+c)*d`

Пример №3

Стандартный ввод
`((a*b)+(c*d))`
Стандартный вывод
`a*b+c*d`

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
1  def parse(s,i):
2      if s[i] != '(':
3          return (s[i], i + 1, s[i])
4      else:
5          left, i, lop = parse(s, i + 1)
6          op = s[i]
7          right, i, rop = parse(s, i + 1)
8          if op == '*' and lop == '+':
9              left = '(' + left + ')'
10         if op == '*' and rop == '+':
11             right = '(' + right + ')'
12         return (left + op + right, i + 1, op)
13 print(parse(input(), 0)[0])
```

Третья волна. Задачи 8–11 класса

Задача II.1.3.1. Кодировка подмножеств (10 баллов)

Темы: задачи для начинающих, математика, реализация.

Условие

Недавно Алиса узнала об одном способе закодировать одним целым числом любое подмножество некоторого заданного конечного множества. Для этого необходимо сопоставить каждому элементу множества число, равное некоторой степени двойки. Теперь в качестве кода произвольного подмножества можно взять сумму чисел, соответствующих элементам этого подмножества.

Алиса составила множество из шести своих друзей и поставила им в соответствие последовательные степени двойки:

- 1 — *Anna*;
- 2 — *Boris*;
- 4 — *Cary*;
- 8 — *David*;
- 16 — *Eva*;
- 32 — *Fiona*.

Например, подмножество {*Anna, Cary, Eva, Fiona*} будет закодировано числом 53. $(1 + 4 + 16 + 32 = 53)$.

Алиса тренируется быстро декодировать подмножество по его коду. Напишите программу, которая позволит проверить ее навыки. Программа должна получать на вход некоторый код и выводить имена друзей, входящих в подмножество с этим кодом.

Формат входных данных

На вход подается единственное натуральное число n — код подмножества, $1 \leqslant n \leqslant 63$.

Формат выходных данных

Выведите имена друзей Алисы, которые входят в закодированное подмножество. Каждое имя следует выводить в отдельной строке. Порядок имен может быть произвольным.

Ниже приведен фрагмент программы на языке Python в котором создается список с правильным написанием слов.

```
Names = ['Anna', 'Boris', 'Cary', 'David', 'Eva', 'Fiona']
```

Методика проверки

Программа проверяется на 20-ти тестах. Прохождение каждого теста оценивается в 0,5 балла. Тест из условия задачи при проверке не используется. Первые шесть тестов — это последовательные степени двойки от 1 до 32.

Примеры

Пример №1

Стандартный ввод
53

Стандартный вывод
Anna Cary Eva Fiona

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
Names = ['Anna', 'Boris', 'Cary', 'David', 'Eva', 'Fiona']
n = int(input())
for i in range(6):
    if n % 2 == 1:
        print(Names[i])
    n //= 2
```

Задача II.1.3.2. Алфавитные подстроки (15 баллов)

Темы: задачи для начинающих, строки, реализация.

Условие

Алиса разрабатывает обучающую игру для младших школьников. В ней игроку дается строка из строчных символов латиницы, а он должен разбить ее на подстроки из последовательных символов алфавита. Такие подстроки далее будем называть правильными. В правильной подстроке после буквы a должна идти буква b, после b — c и так далее. При этом правильная подстрока может начинаться с любого символа. Например, строка *bcdefaabcef* должна быть разбита на $bcdef + a + abc + ef$. Обратите внимание, что подстрока может состоять и из одного символа.

Конечно, игрок может ошибиться и разбить строку неправильным или неоптимальным способом. Например, игрок может разбить строку *bcdefaabcef* на $bcd + ef + aabc + ef$. Чтобы учесть такую возможность Алиса считает очки за найденное разбиение. Если подстрока является правильной, то игроку добавляется количество очков, равное квадрату длины подстроки. Неправильные подстроки не учитываются. Например, за разбиение $bcdef + a + abc + ef$ игрок получит $5^2 + 1^2 +$

$+3^2 + 2^2 = 39$ очков, а за разбиение $bcd{+}ef{+}aabc{+}ef$ лишь $3^2 + 2^2 + 2^2 = 17$ очков.

Напишите программу, которая посчитает количество очков, полученных игроком за сделанное разбиение произвольной строки.

Формат входных данных

На вход в первой строке подается одно натуральное число n — количество фрагментов в разбиении, $1 \leqslant n \leqslant 100$. Далее записаны сами фрагменты разбиения. Каждый фрагмент записан в отдельной строке и состоит только из строчных символов латиницы. Длина каждого фрагмента не превосходит 26.

Формат выходных данных

Выведите одно целое число — количество очков, которое получит игрок за сделанное разбиение.

Методика проверки

Программа проверяется на 15-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется. В первых четырех тестах разбиение состоит из одного фрагмента. В следующих четырех тестах каждый фрагмент содержит не более двух символов.

Примеры

Пример №1

Стандартный ввод
4 bcd ef aabc ef
Стандартный вывод
17

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n = int(input())
ans = 0
for i in range(n):
    s = input()
    for j in range(1, len(s)):
        if ord(s[j]) - ord(s[j - 1]) != 1:
            break
```

```
    else:
        ans += len(s) ** 2
print(ans)
```

Задача II.1.3.3. Пат и Паташон (15 баллов)

Темы: жадные алгоритмы, реализация.

Условие

Боб придумал логическую игру «Пат и Паташон». В этой игре на виртуальной сцене находится n персонажей различного роста, которые выстроены в один ряд. Игрок может удалить часть персонажей со сцены, при этом оставшиеся персонажи смыкаются, не изменяя своего взаимного расположения. После этого персонажи на сцене разбиваются на пары: первый со вторым, третий с четвертым, пятый с шестым и так далее.

В момент удаления игрок должен позаботится о том, чтобы количество оставшихся персонажей стало четным. Первого персонажа в паре (с нечетным номером) будем называть Патом, а второго (с четным номером) — Паташоном. Эффектностью пары будем назвать разность роста Пата и Паташона. Эффектность может быть отрицательной, если окажется, что Пат ниже, чем Паташон. За один раунд игрок получает количество очков, равное сумме эффектностей всех пар. Игрок может удалить со сцены всех персонажей. В этом случае он получит ноль очков.

Рассмотрим пример. Пусть на сцене изначально находилось девять персонажей, рост которых задается массивом чисел $(120, 160, 180, 160, 120, 110, 150, 170, 100)$. Игрок удалил со сцены первого второго и седьмого персонажа. На сцене осталось шесть персонажей с ростом $(180, 160, 120, 110, 170, 100)$. Они разбились на три пары $(180, 160), (120, 110), (170, 100)$, при этом эффектность первой пары — 20, второй — 10, третей — 70. Таким образом, за такое разбиение на пары игрок получит 100 очков. Однако, если игрок оставит на сцене четырех персонажей с ростом $(180, 110, 170, 100)$, то он получит 140 очков.

Напишите программу, которая посчитает максимальное количество очков, которые может получить игрок, для заданной последовательности персонажей.

Формат входных данных

На вход в первой строке подается одно натуральное число n — количество персонажей на сцене в начале игры, $1 \leqslant n \leqslant 1000$. Далее в одной строке через пробел записана последовательность из n натуральных чисел, которые задают рост персонажей. Числа не превосходят 1000.

Формат выходных данных

Выведите одно целое число — максимальное количество очков, которое может получить игрок для заданной последовательности персонажей.

Методика проверки

Программа проверяется на 15-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется. В первых пяти тестах на сцене изначально находится ровно четыре персонажа.

Примеры

Пример №1

Стандартный ввод
9 120 160 180 160 120 110 150 170 100
Стандартный вывод
140

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n = int(input())
x = list(map(int,input().split()))
print(sum([max(x[i]-x[i+1],0) for i in range(n-1)]))
```

Задача II.1.3.4. Выезд на экскурсию (30 баллов)

Темы: математика, структуры данных, реализация.

Условие

Администрация школы организовала автобусную экскурсию для своих учеников. Всего было заказано n автобусов разной вместимости. Обозначим за c_i количество детей, которые могут находиться в i-том автобусе. Все c_i являются *четными* числами. Учителя неформально делят всех учеников на активных и спокойных. Всего на экскурсию поедет m активных и k спокойных детей. Учителя хотели бы распределить детей по автобусам так, чтобы количество спокойных и активных детей в каждом автобусе отличалось как можно меньше. Формально это означает следующее. Обозначим за x_i и y_i количество активных и спокойных детей соответственно в i-том автобусе. Вычислим модули разности количества активных и спокойных детей а каждом автобусе и просуммируем полученные числа. Полученная величина $\sum_{i=1}^{n} |x_i - y_i|$ должна оказаться минимально возможной.

Но когда автобусы подъехали, все пошло не по плану. Часть детей выбежали из школы и расселись по автобусам произвольно. После подсчетов выяснилось, что в i-том автобусе уже находится a_i активных детей и b_i спокойных. Чтобы не увеличивать неразбериху, было решено оставить их на своих местах и постараться рассадить оставшихся детей в соответствии с изначально выбранным принципом.

Напишите программу, которая найдет значения x_i и y_i с учетом всех требований, а именно:

- $x_i \geqslant a_i$;
- $y_i \geqslant b_i$;
- $x_i + y_i \leqslant c_i$;
- $\sum_{i=1}^{n} x_i = m$;
- $\sum_{i=1}^{n} y_i = k$;
- $\sum_{i=1}^{n} |x_i - y_i| \to \min$.

Если допустимых ответов несколько, то можно вывести любой.

Формат входных данных

На вход в первой строке через пробел подается три целых числа n, m и k — количество автобусов, количество активных и количество спокойных детей соответственно; $1 \leqslant n \leqslant 100$; $0 \leqslant m, k \leqslant 10000$. Во второй строке через пробел записаны числа $a_1, a_2, \cdots, a_n$, задающие количество активных детей, изначально находящихся в каждом из автобусов. В третьей строке аналогично записаны числа $b_1, b_2, \cdots, b_n$, задающие количество спокойных детей, изначально находящихся в каждом из автобусов. Наконец, в четвертой строке записаны натуральные четные числа $c_1, c_2, \cdots, c_n$, задающие вместимость каждого из автобусов; $2 \leqslant c_i \leqslant 100$. Все входные значения заданы корректно в соответствии с условием задачи. В том числе гарантируется, что общее количество школьников не превосходит суммарной вместимости всех автобусов.

Формат выходных данных

Вывод должен состоять из двух строк. В первой строке через пробел следует вывести значения x_i — количество активных детей в каждом из автобусов. Во второй строке аналогично вывести значения y_i — количество спокойных детей в каждом из автобусов.

Методика проверки

Программа проверяется на 30-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется. В первых пяти тестах количество автобусов равно двум. В следующих пяти тестах суммарное количество школьников равно суммарной вместимости автобусов.

Примеры

Пример №1

Стандартный ввод

`4 50 80`
`10 20 0 0`
`25 5 20 0`
`40 30 40 30`

Стандартный вывод
`10 20 10 10` `25 10 25 20`

Пояснения к примеру

Ответ удовлетворяет всем ограничениям. Сумма всех x_i равна 50. Сумма всех y_i равна 80. В первом и третьем автобусе едет по 35 детей, а во втором и четвертом — по 30. Эти значения не превосходят вместимости соответствующих автобусов. Также для всех i выполняются неравенства $x_i \geqslant a_i$ и $y_i \geqslant b_i$. Значение выражения $\sum_{i=1}^{n} |x_i - y_i|$ равно 50. Можно доказать, что другие допустимые варианты распределения не дадут меньшей величины.

Возможны и другие правильные ответы, например, следующий.

```
15 20 15 0
25 10 20 25
```

Для этого ответа также выполнены все ограничения, а сумма $\sum_{i=1}^{n} |x_i - y_i|$ равна 50.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```python
n, m, k = map(int, input().split())
a = list(map(int, input().split()))
b = list(map(int, input().split()))
c = list(map(int, input().split()))
m -= sum(a)
k -= sum(b)
for i in range(n):
    if a[i] > b[i]:
        v = min(k, a[i] - b[i], c[i] - a[i] - b[i])
        b[i] += v
        k -= v
    else:
        v = min(m, b[i] - a[i], c[i] - a[i] - b[i])
        a[i] += v
        m -= v
for i in range(n):
    v = min(m, k, (c[i] - a[i] - b[i]) // 2)
    a[i] += v
    b[i] += v
    m -= v
    k -= v
for i in range(n):
    v = min(m, c[i] - a[i] - b[i])
    a[i] += v
    m -= v
    v = min(k, c[i] - a[i] - b[i])
    b[i] += v
    k -= v
print(*a)
print(*b)
```

Задача II.1.3.5. Нескучные каникулы (30 баллов)

Темы: сортировки, структуры данных, реализация.

Условие

У Алисы закончился очередной учебный год, и она составляет расписание на каникулы. Алиса планирует, что в ее каникулы состоится некоторое число событий, таких как посещение концертов, празднование дней рождений и так далее. Алиса называет i-тый день каникул нескучным, если для него выполняется хотя бы одно из двух условий:

- в i-тый день состоится хотя бы одно событие;
- хотя бы одно событие состоится в день с номером $i-1$ и в день с номером $i+1$.

Рассмотрим пример. Пусть в каникулах 10 дней и некоторые события произойдут в дни с номерами $2, 3, 5, 9, 10$. Тогда нескучными будут все эти дни, а также день с номером 4, поскольку некоторые события произойдут в два соседних с ним дня.

При составлении расписания Алиса учитывает, что для некоторых событий заранее известна дата, а для других она сама может подобрать подходящий день. Алиса хочет расставить события с открытой датой так, чтобы каникулы получились наиболее нескучными, то есть чтобы количество нескучных дней в каникулах было максимальным.

Напишите программу, которая подберет дни для событий с открытой датой так, чтобы каникулы получились наиболее нескучными.

Формат входных данных

На вход в первой строке через пробел подается три целых числа n, m и k — продолжительность каникул в днях, количество событий с открытой датой и количество событий с заданной датой соответственно; $1 \leqslant n \leqslant 100000$; $1 \leqslant m \leqslant 100000$; $0 \leqslant k \leqslant 100000$.

Во второй строке через пробел записаны k натуральных чисел $d_1, d_2, \cdots, d_k$ — номера дней, в которые произойдут события с известной датой; $1 \leqslant d_i \leqslant n$. Числа могут повторяться и следовать в произвольном порядке. Если k будет равно нулю, то вторая строка будет пустой.

Формат выходных данных

В первой строке выведите одно натуральное число s — количество нескучных дней в каникулах. Во второй строке через пробел выведите m натуральных чисел $t_1, \cdots, t_m$ — номера дней, в которые Алиса должна запланировать события с открытой датой. Если допустимых ответов будет несколько, то можно вывести любой. Числа могут повторяться и следовать в произвольном порядке.

Методика проверки

Программа проверяется на 30-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тесты из условия задачи при проверке не используются. В трех первых тестах $k = 0$. В следующих трех тестах $k = 1$. В первых 15-ти тестах n, m и k не превосходят 100.

Примеры

Пример №1

Стандартный ввод
11 5 6 1 3 5 7 9 11
Стандартный вывод
11 1 1 1 1 1

Пример №2

Стандартный ввод
11 2 0
Стандартный вывод
3 2 4

Пример №3

Стандартный ввод
15 2 5 1 2 8 12 14
Стандартный вывод
11 4 6

Пояснения к примеру

В первом примере все дни каникул являются нескучными из-за событий с известной датой, поэтому пять событий с открытой датой можно расставить произвольно.

В ответе ко второму примеру нескучными будут дни с номерами 2, 3, 4. Улучшить ответ нельзя.

В ответе к третьему примеру нескучными будут 11 дней с номерами 1, 2, 3, 4, 5, 6, 7, 8, 12, 13, 14. Улучшить этот ответ нельзя, хотя набор дней может быть другим, например, 4, 10 или 6, 10.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n, m, k = map(int, input().split())
d = [False] * (n + 2)
for x in map(int, input().split()):
    d[x] = True
if k == 0:
    ans = list(range(1, min(n, 2 * m), 2))
```

```
    ans.extend([n] * max(m - len(ans), 0))
else:
    p = 0
    segs = []
    for i in range(1,n + 1):
        if d[i]:
            if p == 0:
                plen = i - 1
            elif  i - p > 2:
                segs.append((p + 2, i))
            p = i
    segs.sort(key = lambda x: x[1] - x[0] + ((x[1] - x[0]) % 2) * 100000)
    ans = []
    for (a, b) in segs:
        ans.extend(range(a, b, 2))
    if n - p > 1:
        ans.extend(range(p + 2, n + 1, 2))
    if plen > 1:
        ans.extend(range(plen - 1, 0, -2))
    if (n - p) % 2 == 1:
        ans.append(n)
    ans = ans[: min(m, len(ans))]
    ans.extend([1] * (m - len(ans)))
for i in ans:
    d[i] = True
s = 0
for i in range(1, n + 1):
    if d[i] or d[i - 1] and d[i + 1]:
        s += 1
print(s)
print(*ans)
```

Предметный тур. Математика

Первая волна. Задачи 8–9 класса

Задача II.2.1.1. (15 баллов)

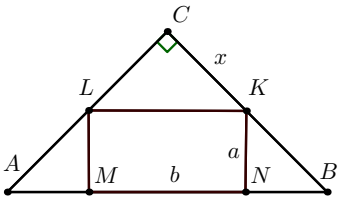
Темы: планиметрия.

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

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

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

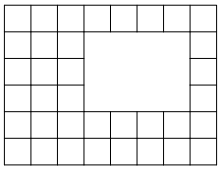
Ответ: $1 + \dfrac{p(n-1)}{100(2n-1)}$.

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.

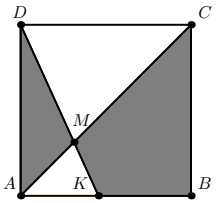
Условие

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)

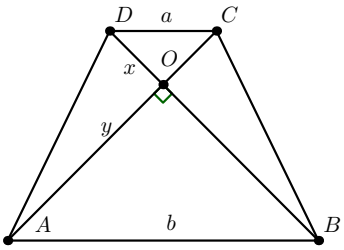
Темы: планиметрия, трапеция.

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

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

Наконец, для пересечения всех трех множеств получаем

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

Таким образом, в четвертой группе соревнуются команды в количестве

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

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

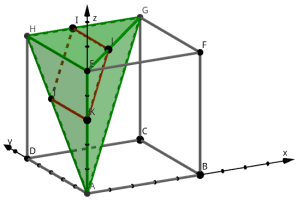
Выбирая три числа из $[0, a]$, назовем x — наименьшее из них, z — наибольшее, а y — лежащее между x и z. Выбор чисел x, y, z равносилен выбору точки $\alpha(x, y, z)$ в пространстве, удовлетворяющей условиям:

- 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

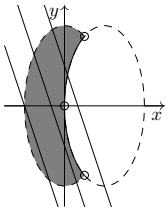
$$\log_{x^2+\frac{y^2}{a^2}} 2x \geqslant 1, \tag{II.2.1}$$

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.

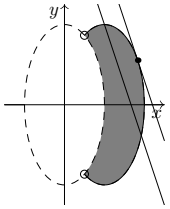


Рассматривая всевозможные прямые $y = m - bx$, проходящие через данную область (некоторые из этих прямых изображены на рисунке), видим, что наибольшего значения m не существует (можно сколь угодно близко приближать прямую к границе области).

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)

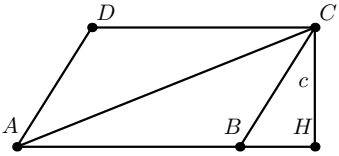
Темы: планиметрия, параллелограмм.

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

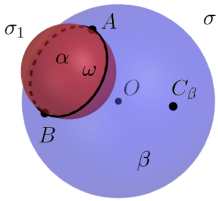
Точки A и B лежат на сфере с центром O так, что угол AOB равен α°. Случайно на сфере выбирается еще одна точка C (любой выбор равновозможен). Определите вероятность того, что угол ACB окажется острым.

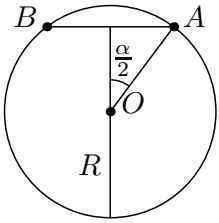
Запишите ответ в процентах с точностью до 0,01.

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 имеет целью проверить навыки работы участников с алгоритмами классификации для разработки модели, способной автоматически классифицировать упражнения пациентов на видеозаписях. Она проверяет компетенции участников в обработке данных и применении алгоритмов машинного обучения, а также способность погружаться в предметную сферу и применять формулы и классические алгоритмы в решении задач по искусственному интеллекту. Задача помогает оценить участников по их знаниям и навыкам в области программирования, медицины, обработки табличных данных, математической статистики и применении технических решений для решения реальных проблем.

Задача II.3.1. PN-expert (100 баллов)

Темы: программирование, исследование данных, машинное обучение, математика.

Условие

В первом отборочном этапе участникам предлагается решить простую задачу по определению наименования упражнения для оценки постурального и кинетического тремора у пациентов с диагностированной болезнью Паркинсона.

Почему это важно?

Диагностирование и постановка болезни Паркинсона (далее — БП) требует от врача наличие достаточного опыта и понимания требований клинических рекомендаций, утвержденных Минздравом. Наиболее распространенным диагностическим инструментом БП является единая шкала оценки MDS UPDRS: https://sudact.ru/law/klinicheskie-rekomendatsii-bolezn-parkinsona-vtorichnyi-parkinsonizm-i/prilozhenie-g1-gn/prilozhenie-g1/.

Таким образом, на постановку клинически достоверного диагноза влияет субъективный фактор, характеризующийся исключительно восприятием врачом движений пациента в процессе осмотра. То есть сейчас эта оценка массово субъективна!

Объективная оценка двигательных нарушений является приоритетом для клинического наблюдения за пациентами при постановке диагноза, назначении и корректировке лечения, особенно проявляющаяся при мониторинге и контроле реакций пациента на изменение лекарственной терапии (смена дозировки и/или препарата). Созданием такого объективного инструмента занимаются исследователи, разрабатывающие PN-expert: https://inscience.news/ru/article/nti/8982.

Эта платформа позволяет отслеживать состояние пациентов благодаря анализу видеозаписей, где люди выполняют специальные упражнения.

Для того чтобы диагностический инструмент работал корректно, алгоритмам важно не только успешно измерять тремор, но и автоматически разграничивать упражнения, которые пациент выполняет на видео. В нашей задаче мы предлагаем вам поработать со множеством данных из наблюдений за пациентами, тестирующими PN-expert, и создать модель машинного обучения, помогающую классифицировать

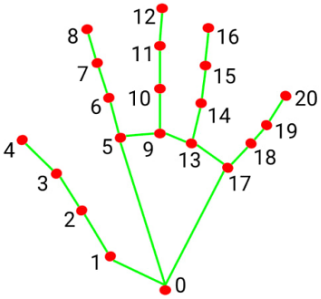
упражнения.

Ссылка на оригинальное соревнование: https://cups.online/ru/contests/first_round_nto_ai.

Описание данных

- Имя файла данных — название файла, где хранятся полные записи о результатах прохождения тестов конкретного пациента.
- Folder Path — наша целевая переменная. Класс упражнения, которое выполнил конкретный пациент.
- Пол — пол конкретного пациента.
- Полных лет — количество полных лет конкретного пациента.
- Пациент off/on — работа нейростимулятора у конкретного пациента.
- Определенный диагноз врачом (0–5) — субъективная оценка состояния конкретного пациента врачом по шкале `MDS UPDRS`.

Архив с полными записями можно скачать по этой ссылке: https://disk.yandex.ru/d/3sQ-vEewhCh2aw/data.zip.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Критерии оценивания

Accuracy по приватной части датасета.

Ссылка на описание кода метрики: https://disk.yandex.ru/d/3sQ-vEewhCh2aw/scoring.ipynb.

Решение

Требуется разработать алгоритм для классификации типа движения и представить его результат работы в виде `CSV` файла с колонками `path` и `pred`.

Ответ: решением задачи будет файл с предсказанием, загружаемый на платформу проведения соревнования для оценки качества решения. Пример загружаемого файла с решением https://disk.yandex.ru/d/3sQ-vEewhCh2aw/sample_submit.csv.

Ссылка на эталонное экспертное решение: https://disk.yandex.ru/d/3sQ-vEewhCh2aw/reference_solution.

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

Задачи второго отборочного этапа сосредоточены на предсказании свойств и дизайне малых органических молекул. Эти задачи высоко релевантны в современной химии и биологии, так как играют ключевую роль в разработке новых лекарств, материалов и технологических процессов.

Индивидуальная задача

Задача IV.1.1. (100 баллов)

Темы: анализ данных, методы QSAR, химические базы данных.

Условие

Участникам необходимо разработать алгоритм, способный прогнозировать липофильность малых органических молекул. Данный параметр по сути представляет коэффициент распределения между неполярной и полярной фазой — соотношение растворимости соединения в октаноле-1 и воде. Липофильность — важный параметр, расчет которого является неотъемлемой частью разработки новых лекарственных средств.

Формат входных данных

Файл, содержащий строковое представление (`Simplified Molecular Input Line Entry System, SMILES`).

Формат выходных данных

Файл с расширением `CSV`, содержащий предсказанные значения липофильности молекул (каждое значение — на новой строке).

Пример входных данных (фрагмент файла)

```
CCOc1ccccc1O
Cc1cccc(C)n1
CC(C)C#N
```

Результат работы программы (фрагмент файла)

```
1.73
1.44
0.23
```

Критерии оценивания

Для оценки решений будет использована стандартная метрика для регрессионных задач, корень из средней квадратичной ошибки (root mean square error, RMSE).

Ссылка на описание метрики: https://scikit-learn.org/stable/modules/model_evaluation.html#mean-squared-error.

Решение

Участникам доступно базовое решение от разработчиков задачи с использованием библиотек `scikit-learn` и `RDKit`.

Предполагаемый пайплайн работы модели включает:

- генерацию представления молекул, пригодного для использования в качестве входных данных для предсказательных моделей;
- предсказание целевого параметра исходя из сгенерированного представления.

Пример решения (с комментариями) доступен по ссылке: https://disk.yandex.ru/d/unyk1lr18_zxiw.

Командная задача

Задача IV.2.1. (100 баллов)

Темы: эволюционные алгоритмы, методы представления молекулярной структуры, генеративные модели, интерпретация молекулярных данных, командная работа.

Условие

Участникам необходимо разработать алгоритм, предназначенный для направленного поиска молекул с заданными свойствами. Следуя практике научных исследований, оценка качества генерации будет проводиться для представительной выборки, а не отдельных соединений. Разнообразие, валидность, уникальность и соответствие заданному диапазону физико-химического свойства — целевые показатели для массива новых молекул, к максимизации которых следует стремиться.

Формат входных данных

Файл с расширением CSV (пример: https://disk.yandex.ru/d/L---JfJV1EOAxQ), содержащий список `SMILES` последовательностей (по одной на строку) — это набор молекул, служащих в качестве стартовой выборки для генерации новых соединений.

Формат выходных данных

Файл с расширением `CSV`, содержащий список `SMILES` последовательностей (по одной на строку) — это результат работы предложенного участниками решения; набор молекул, соответствующих заданным критериями решения задачи.

Пример входных данных (фрагмент файла)

```
ClC(Cl)(Cl)C(Cl)(Cl)Cl
Cc1c(Cl)cccc1Cl
CCCCCI
```

Результат работы программы (фрагмент файла)

```
CC(CSC(=O)c1ccccc1)C(=O)N1CC(Sc2ccccc2)CC1C(=O)O
CCC/C(=N\OCC)C1C(=O)CC2(CCC(C)CC2)OC1=O
COc1ccc(C2Cc3cc(OC(F)F)ccc3N(CCN(C)C)C(=O)C2OC(C)=O)cc1
```

Критерии оценивания

Оценка качества решения будет проводиться на основании комплексной величины, представляющей собой произведение четырех следующих метрик:

- Доля молекул в сгенерированной выборке, для которых значение целевого свойства (липофильности) входит в заданный диапазон $(2,0 < \log P < 3,0)$. Оценка величины будет производиться с помощью модели, недоступной участникам.
- Валидность (`validity`). Доля `SMILES` строк в сгенерированной выборке, соответствующих химически валидным молекулам.
- Новизна (`novelty`). Доля `SMILES` строк в сгенерированной выборке, не входящих в исходную выборку, предоставленную участникам.
- Разнообразие выборки (`internal diversity`).

Решение

Участникам доступно базовое решение от организаторов. Предполагаемый пайплайн работы включает в себя:

- Файл `generate.py` должен содержать функцию `search_step()`, которая принимает на вход список `SMILES` строк и возвращает список той же длины, отличающийся от исходного не более чем на один элемент.
- Файл модели для оценки липофильности. Одна из составляющих метрики предполагает оценку данного параметра. Участники могут воспользоваться собственной моделью, обученной на индивидуальном этапе, или любой иной.

Стартовая выборка молекул, содержащаяся в файле `check.csv` (https://disk.yandex.ru/d/L---JfJV1EOAxQ) подвергается многократному обновлению посредством вызова функции `search_step()`. Итоговая выборка оценивается в точки зрения описанных метрик.

Пример решения (с комментариями) доступен по ссылке: https://disk.yandex.ru/d/_-AqM2pRFCIs8Q.

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 программирование. 8–11 классы

Тестовые наборы для задач представлены по ссылке — https://disk.yandex.ru/d/w8wseOPVsnyiYg.

Задача VI.1.1.1. ИИ максимизирует сумму (10 баллов)

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

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

Ограничение по времени выполнения программы: 3 с.

Ограничение по памяти: 256 Мбайт.

Условие

Для обучения искусственного интеллекта (ИИ) разработчик использует следующую задачу. На вход подается массив размера $3n$, состоящий из целых чисел $a_1, a_2, \ldots, a_{3n}$. После этого ИИ должен сгенерировать массив $b_1, b_2, \ldots, b_{3n}$, в котором ровно n элементов равны числу 1, ровно n элементов равны числу 2 и ровно n элементов равны числу 3. Массив b должен быть сгенерирован таким образом, чтобы сумма $a_1 \cdot b_1 + a_2 \cdot b_2 + \ldots + a_{3n} \cdot b_{3n}$ принимала максимально возможное значение.

От вас требуется проверить работу ИИ. Для этого необходимо вычислить максимально возможное значение суммы, которую должен получить ИИ.

Формат входных данных

В первой строке дано целое число n, $1 \leqslant n \leqslant 30000$.

Во второй строке через пробел заданы $3n$ целых чисел $1 \leqslant a_1, a_2, \ldots, a_{3n} \leqslant 1000$.

Формат выходных данных

Выведите единственное целое число — максимально возможное значение суммы, которую должен получить ИИ.

Примеры

Пример №1

Стандартный ввод
1 1 1 1
Стандартный вывод
6

Пример программы-решения

Ниже представлено решение на языке C++.

```cpp
#include <bits/stdc++.h>
using namespace std;

int n;
vector<int> a;

int main() {
    ios_base::sync_with_stdio(false); cin.tie(0);
    cin >> n;
    a.resize(3 * n);
    for (auto& x : a)
        cin >> x;
    sort(a.begin(), a.end());
    int res = 0;
    for (int i = 0; i < 3 * n; ++i)
        res += (i / n + 1) * a[i];
    cout << res << endl;

    return 0;
}
```

Задача VI.1.1.2. Поиск геометрической прогрессии (15 баллов)

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

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

Ограничение по времени выполнения программы: 3 с.

Ограничение по памяти: 256 Мбайт.

Условие

Вас попросили доработать специальный модуль для умного помощника. Этот модуль должен находить геометрические прогрессии в числовых последовательностях.

На вход подается массив из n целых положительных чисел $a_1, a_2, \ldots, a_n$. Требуется найти количество пар $1 \leqslant l < r \leqslant n$, таких что набор чисел $a_i, \ldots, a_r$ после сортировки образует геометрическую прогрессию.

Рассмотрим в качестве примера массив 1, 1, 4, 2, 3.

Для $l = 1$, $r = 2$ набор 1, 1 образует геометрическую прогрессию со знаменателем 1.

Для $l = 2$, $r = 3$ набор 1, 4 образует геометрическую прогрессию со знаменателем 4.

Для $l = 3$, $r = 4$ набор 4, 2 после сортировки превращается в набор 2,4 и образует геометрическую прогрессию со знаменателем 2.

Для $l = 4$, $r = 5$ набор 2, 3 образует геометрическую прогрессию со знаменателем 3/2.

Для $l = 2$, $r = 4$ набор 1, 4, 2 после сортировки превращается в набор 1, 2, 4 и образует геометрическую прогрессию со знаменателем 2.

Других подходящих пар для данного массива нет, поэтому ответ равен пяти.

Формат входных данных

В первой строке дано целое число n — размер массива, $2 \leqslant n \leqslant 10^4$.

Во второй строке через пробел заданы n целых чисел $1 \leqslant a_1, a_2, \ldots, a_n \leqslant 10^6$.

Формат выходных данных

Выведите единственное целое число — ответ на задачу.

Примеры

Пример №1

Стандартный ввод
5 1 1 4 2 3
Стандартный вывод
5

Пример программы-решения

Ниже представлено решение на языке C++.

```
#include <bits/stdc++.h>
using namespace std;

int n;
vector<long long> a, b, v;

bool ok() {
    int m = v.size();
    for (int i = 0; i + 2 < m; ++i)
        if (v[i + 1] * v[i + 1] != v[i] * v[i + 2])
            return false;
    return true;
}

int main() {
```

```
16      ios_base::sync_with_stdio(false); cin.tie(0);
17      cin >> n;
18      a.resize(n);
19      for (auto& x : a)
20          cin >> x;
21      int res = n - 1;
22      b.assign(n, 1);
23      for (int i = n - 2; i >= 0; --i) {
24          if (a[i] == a[i + 1])
25              b[i] = b[i + 1] + 1;
26          if (b[i] > 2)
27              res += b[i] - 2;
28      }
29      for (int l = 0; l + 1 < n; ++l) {
30          if (a[l] == a[l + 1]) continue;
31          v.clear();
32          v.push_back(a[l]);
33          v.push_back(a[l + 1]);
34          for (int r = l + 2; r < l + 20 && r < n; ++r) {
35              v.push_back(a[r]);
36              sort(v.begin(), v.end());
37              if (ok()) ++res;
38          }
39      }
40      cout << res << endl;
41
42      return 0;
43  }
```

Задача VI.1.1.3. Кольцевой маршрут (20 баллов)

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

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

Ограничение по времени выполнения программы: 3 с.

Ограничение по памяти: 256 Мбайт.

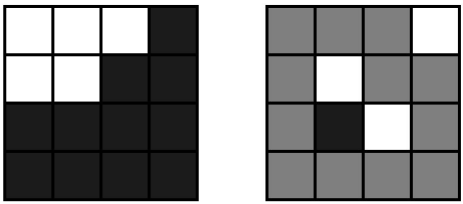
Условие

Для соревнования роботов из квадратных клеток была собрана квадратная площадка размера n×n клеток. На площадке есть свободные клетки, на которые может заезжать робот, и клетки, на которых располагается преграда, и на такую клетку робот заехать не может. Робот может перемещаться с одной свободной клетки на другую свободную клетку, если эти клетки являются соседними по стороне.

Василий привез своего робота на соревнования и хочет его потренировать на площадке. Самый лучший вариант для тренировки — это запустить робота по «кольцевому маршруту». «Кольцевой маршрут» по определению Василия — это такая последовательность свободных клеток $k_1, k_2, \ldots, k_s$, что каждая клетка встречается в этой последовательности ровно один раз, клетки k_1 и k_2, k_2 и $k_3, \ldots, k_{s-1}$ и k_s, k_s и k_1 являются соседними по стороне, а также внутренняя область ограниченная маршрутом должна содержать хотя бы одну клетку с преградой.

На рисунке приведены две площадки, на которых свободные клетки покрашены в черный цвет, а клетки с преградой покрашены в белый цвет. На первой площадке

нет ни одного «кольцевого маршрута». На второй площадке «кольцевой маршрут» присутствует и для наглядности клетки одного из «кольцевых маршрутов» покрашены в серый цвет.



В этой задаче нужно определить есть ли «кольцевой маршрут» на заданной площадке или нет.

Формат входных данных

В первой строке дано целое число t — количество тестов в задаче, $1 \leqslant t \leqslant 20$.

Далее идет описание тестов. Каждый тест начинается со строки, в которой дано целое число n — размер площадки, $3 \leqslant n \leqslant 30$.

В следующих n строках даны строки длины n, которые задают площадку для данного теста. Если символ строки равен 1, то соответствующая клетка площадки является свободной, а если символ равен 0, то соответствующая клетка содержит преграду.

Формат выходных данных

Выведите t строк — ответы для каждого из тестов. Если в соответствующем тесте дана площадка, содержащая «кольцевой маршрут», то выведите слово YES. В противном случае выведите слово NO.

Примеры

Пример №1

Стандартный ввод
2 4 0001 0011 1111 1111 4 1110 1011 1101 1111

Стандартный вывод
NO YES

Решение

Пример программы-решения

Ниже представлено решение на языке C++.

```
#include <bits/stdc++.h>
using namespace std;

int n, imi, ima, jmi, jma;
vector<vector<int> > u;
vector<string> a;

void dfs(int i, int j) {
    u[i][j] = 1;
    imi = min(imi, i);
    ima = max(ima, i);
    jmi = min(jmi, j);
    jma = max(jma, j);
    for (int di = -1; di < 2; ++di) {
        for (int dj = -1; dj < 2; ++dj) {
            int I = i + di;
            int J = j + dj;
            if (I < 0 || I >= n) continue;
            if (J < 0 || J >= n) continue;
            if (a[I][J] == '1') continue;
            if (!u[I][J])
                dfs(I, J);
        }
    }
}

bool f() {
    cin >> n;
    a.resize(n);
    for (int i = 0; i < n; ++i)
        cin >> a[i];
    u.assign(n, vector<int>(n, 0));
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            if (u[i][j]) continue;
            if (a[i][j] == '1') continue;
            imi = ima = i;
            jmi = jma = j;
            dfs(i, j);
            if (imi > 0 && ima + 1 < n && jmi > 0 && jma + 1 < n)
                return true;
        }
    }
    return false;
}

int main() {
    ios_base::sync_with_stdio(false); cin.tie(0);
```

```
49      int t; cin >> t;
50      while (t--)
51          cout << (f() ? "YES\n" : "NO\n");
52
53      return 0;
54  }
```

Задача VI.1.1.4. Связи в нейронной сети (20 баллов)

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

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

Ограничение по времени выполнения программы: 3 с.

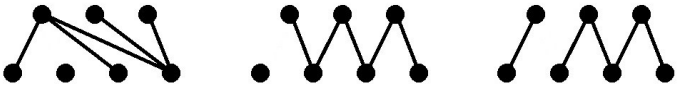
Ограничение по памяти: 256 Мбайт.

Условие

При разработке планарной нейронной сети возникла задача, с которой вам придется справиться. Для начала рассмотрим два соседних слоя планарной нейронной сети. Пусть в одном слое есть n узлов, а в другом — m узлов. Узлы каждого из слоев располагаются вдоль прямой, причем разные слои лежат на разных прямых параллельных друг другу. Связи проводятся в виде отрезков между узлами соседних слоев, причем в планарной нейронной сети эти отрезки не могут пересекаться между собой.

Полной системой связей между двумя соседними слоями с n и m узлами в планарной нейронной сети является такая система связей, при которой нельзя провести ни одной дополнительной связи. На рисунке ниже приведены два примера полных систем связей при $n = 3$, $m = 4$.

Почти полной системой связей между двумя соседними слоями с n и m узлами в планарной нейронной сети является такая система связей, при которой можно провести еще одну связь между этими слоями, после чего данная система связей станет полной системой связей. На рисунке ниже приведены три примера почти полных систем связей при $n = 3$, $m = 4$.



Ваша задача заключается в том, чтобы находить количество различных полных или почти полных систем связей для двух соседних слоев с n и m узлами в планарной нейронной сети. Две системы связей считаются различными, если найдутся две вершины из соседних слоев, которые в одной системе связей соединены отрезком, а в другой системе связей — не соединены.

Формат входных данных

В первой строке через пробел даны два целых числа n, m — количества узлов в двух соседних слоях, $1 \leqslant n, m \leqslant 20$.

Во второй строке дано целое число k, которое задает тип интересующей нейронной сети, $1 \leqslant k \leqslant 2$. Если $k = 1$, то необходимо вычислить количество полных систем связей. Если $k = 2$, то необходимо вычислить количество почти полных систем связей.

В данной задаче тестов с $k = 1$ и тестов с $k = 2$ равное количество.

Формат выходных данных

В единственной строке выведите единственное целое число — количество полных или почти полных (в зависимости от значения k) систем связей для двух соседних слоев с n и m узлами в планарной нейронной сети.

Примеры

Пример №1

Стандартный ввод
1 3 1
Стандартный вывод
1

Пример №2

Стандартный ввод
1 3 2
Стандартный вывод
3

Пример программы-решения

Ниже представлено решение на языке C++.

```
#include <bits/stdc++.h>
using namespace std;

vector<vector<long long> > dp, dd;
int h = 20;
int n, m, k;

int main() {
    ios_base::sync_with_stdio(false); cin.tie(0);
    dp.assign(h + 1, vector<long long>(h + 1, 0));
```

```
    dd.assign(h + 1, vector<long long>(h + 1, 0));
    for (int e = 1; e < h + 1; ++e)
        dp[e][1] = dp[1][e] = 1;
    for (int i = 2; i < h + 1; ++i)
        for (int j = 2; j < h + 1; ++j)
            dp[i][j] = dp[i - 1][j] + dp[i][j - 1];

    for (int i = 1; i < h + 1; ++i) {
        for (int j = 1; j < h + 1; ++j) {
            dd[i][j] = (i + j > 2 ? 2 : 1) * dp[i][j];
            for (int a = 1; a < i; ++a)
                for (int b = 1; b < j; ++b)
                    dd[i][j] += dp[a][b] * dp[i - a][j - b];
            for (int a = 1; a < i + 1; ++a)
                for (int b = 1; b + 1 < j; ++b)
                    dd[i][j] += dp[a][b] * dp[i + 1 - a][j - 1 - b];
            for (int a = 1; a + 1 < i; ++a)
                for (int b = 1; b < j + 1; ++b)
                    dd[i][j] += dp[a][b] * dp[i - 1 - a][j + 1 - b];
        }
    }
    cin >> n >> m >> k;
    cout << (k == 1 ? dp[n][m] : dd[n][m]) << endl;

    return 0;
}
```

Задача VI.1.1.5. Одинаковые квадраты (30 баллов)

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

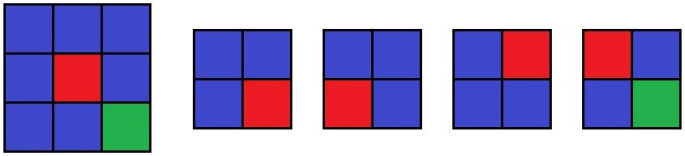
Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 3 с.

Ограничение по памяти: 256 Мбайт.

Условие

В этой задаче нам предстоит проверить искусственный интеллект (ИИ), который был создан для распознавания изображений. Для проверки используется квадратная таблица размера $n \times m$, каждая клетка которой покрашена в один из трех цветов — красный, зеленый или синий. ИИ должен рассмотреть все квадратные подтаблицы данной таблицы и вычислить количество одинаковых пар среди этих подтаблиц. ИИ считает две квадратные подтаблицы одинаковыми, если их размеры совпадают и полностью совпадают цвета соответствующих клеток в этих подтаблицах. Но есть еще одна особенность — если цвета двух подтаблиц не совпали, то ИИ будет поворачивать одну из этих таблиц и снова проводить сравнение. Поэтому если после поворотов одна из таблиц оказывается такой же, как другая, то ИИ считает их одинаковыми. Обратите внимание, что ИИ делает только повороты и не может делать отражений.



На рисунке приведена таблица размера 3×3. У этой таблицы есть девять квадратных подтаблиц размера 1×1 — семь синего цвета и по одной красного и зеленого цвета, что дает 21 пару одинаковых синих таблиц размера 1×1. Еще у этой таблицы есть четыре квадратных подтаблицы размера 2×2, три из которых одинаковы и дают еще 3 пары одинаковых подтаблиц. Квадратная подтаблица размера 3×3 только одна — сама исходная таблица, поэтому больше пар одинаковых подтаблиц нет. Итого имеем 24 пары одинаковых подтаблиц для данной таблицы.

Формат входных данных

В первой строке дано целое число n — размер данной таблицы, $2 \leqslant n \leqslant 70$.

В следующих n строках задана сама таблица. В каждой из них дается строка длины n, состоящая из символов «1», «2» и «3», которые обозначают красный, зеленый и синий цвет соответственно.

Формат выходных данных

Выведите единственное целое число — количество пар одинаковых квадратных подтаблиц данной таблицы, с учетом того, что подтаблицы при сравнении можно поворачивать.

Примеры

Пример №1

Стандартный ввод
3 333 313 332

Стандартный вывод
24

Пример программы-решения

Ниже представлено решение на языке C++.

```
#include<bits/stdc++.h>
using namespace std;
#define ll  long long
#define ull unsigned ll
```

```
5
6   ll  n;
7   ull p = 17, q = 19;
8   vector<string> a;
9   map<vector<ull>, ll> M;
10  vector<vector<ull> > h[4];
11  vector<ull> pp, qq;
12
13  ull hh(int i, int I, int j, int J, int c) {
14      return (h[c][I][J] - h[c][i][J] - h[c][I][j] + h[c][i][j]) * pp[n - i] * qq[n
        ↪  - j];
15  }
16
17  vector<ull> ha(int i, int I, int j, int J) {
18      vector<ull> v;
19      for (int c = 0; c < 4; ++c) {
20          v.push_back(hh(i, I + 1, j, J + 1, c));
21          i = n - 1 - i; swap(i, j);
22          I = n - 1 - I; swap(I, J);
23          swap(j, J);
24      }
25      sort(v.begin(), v.end());
26      return v;
27  }
28
29  void rotation() {
30      auto b = a;
31      for (int i = 0; i < n; ++i)
32          for (int j = 0; j < n; ++j)
33              a[j][n - 1 - i] = b[i][j];
34  }
35
36  int main() {
37      ios_base::sync_with_stdio(false); cin.tie(0);
38      cin >> n;
39      a.resize(n);
40      for (int i = 0; i < n; ++i)
41          cin >> a[i];
42      pp.assign(n + 5, 1);
43      qq.assign(n + 5, 1);
44      for (int i = 1; i < n + 5; ++i) {
45          pp[i] = pp[i - 1] * p;
46          qq[i] = qq[i - 1] * q;
47      }
48      for (int c = 0; c < 4; ++c) {
49          h[c].assign(n + 1, vector<ull>(n + 1, 0));
50          for (int i = 0; i < n; ++i)
51              for (int j = 0; j < n; ++j)
52                  h[c][i + 1][j + 1] = (a[i][j] - '0') * pp[i] * qq[j];
53          for (int i = 0; i < n + 1; ++i)
54              for (int j = 1; j < n + 1; ++j)
55                  h[c][i][j] += h[c][i][j - 1];
56          for (int j = 0; j < n + 1; ++j)
57              for (int i = 1; i < n + 1; ++i)
58                  h[c][i][j] += h[c][i - 1][j];
59          rotation();
60      }
61      ll res = 0;
62      for (int m = 1; m < n; ++m) {
63          M.clear();
```

```
64          for (int i = 0; i + m - 1 < n; ++i)
65              for (int j = 0; j + m - 1 < n; ++j)
66                  ++M[ha(i, i + m - 1, j, j + m - 1)];
67          for (auto el : M) {
68              ll num = el.second;
69              res += (num * (num - 1)) / 2;
70          }
71      }
72      cout << res << endl;
73
74      return 0;
75  }
```

Математика. 8–9 классы

Задача VI.1.2.1. (15 баллов)

Темы: текстовая задача, делимость.

Условие

Пусть n — 2024-значное натуральное число, состоящее только из шестерок, семерок и восьмерок (других цифр нет). Количество шестерок на 515 больше количества восьмерок. Докажите, что число n делится на 9.

Критерии оценивания

1. Из текста задачи верно получена система уравнений (VI.1.1), но дальнейших продвижений нет — 5 баллов.
2. В ходе вычислений допущена арифметическая ошибка — штраф 2 балла.
3. Приведено полное доказательство — 15 баллов.

Доказательство

Пусть количество шестерок m, семерок l, восьмерок k. Тогда

$$\begin{cases} m + l + k = 2024, \\ m = k + 515, \end{cases} \tag{VI.1.1}$$

или

$$\begin{cases} m + l + k = 2024, \\ l + 2k = 1509. \end{cases}$$

Найдем сумму цифр числа n:

$$S = 6m + 7l + 8k = 6(m + l + k) + (l + 2k) = 6 \cdot 2024 + 1509 = 13653 \vdots 9.$$

Сумма цифр числа n делится на 9, следовательно, и само число n кратно 9.

Задача VI.1.2.2. (20 баллов)

Темы: планиметрия.

Условие

В тупоугольном треугольнике ABC, $\angle B = 110^\circ$, проведена медиана BM. На стороне BC нашлась такая точка K, что $BK = AB$ и $\angle BMK = 90^\circ$. Найдите угол MBC.

Критерии оценивания

1. Построена точка D — 5 баллов.
2. Доказано, что треугольник BDK равнобедренный — 7 баллов.
3. Верно найден $\angle MBC$ — 8 баллов (баллы суммируются).

Решение

Продлим луч BM и на его продолжении отметим точку D так, что $MD = BM$. Тогда отрезки AC и BD точкой M делятся пополам, следовательно, $ABCD$ — параллелограмм, $\angle BCD = 70^\circ$ и $AB = DC$.

Заметим, что точка K лежит на серединном перпендикуляре к отрезку BD. Следовательно, она равноудалена от его концов, т. е. $BK = DK$. По условию задачи $BK = AB$, тогда $DK = DC$ и треугольник DKC — равнобедренный, $\angle DKC = \angle DCK = 70^\circ$. По теореме о внешнем угле

$$\angle DKC = \angle BDK + \angle DBK = 2\angle DBK = 2\angle MBC.$$

Тем самым получаем, что $\angle MBC = 35^\circ$.

Ответ: 35°.

Задача VI.1.2.3. (20 баллов)

Темы: классическая вероятность.

Условие

Для участия в математических боях преподаватель составляет команду четырех девятиклассников из группы 10 человек, занимающихся у него в кружке. Все дети достаточно сильные, каждый имеет как свои сильные, так и слабые стороны. Какова вероятность, что два друга Илья и Миша попадут в число избранных?

Критерии оценивания

1. Найдено число A — 5 баллов.
2. Найдено число B — 5 баллов.
3. Получен правильный ответ — 5 баллов.
4. За арифметические ошибки, не влияющие на логику рассуждений, — штраф до 5 баллов.

Решение

Пусть A — количество способов составить команду в составе четырех человек из 10 имеющихся; B — количество способов составить команду, в составе которой уже имеются Илья и Миша. Тогда требуемая вероятность будет равна отношению количества команд, в составе которых присутствуют Илья и Миша, к количеству

всех способов составления команд, т. е. $p = A/B$. Для выборки команд важен только их состав (члены команды не отличаются по ролям друг от друга). Следовательно, выборки — это сочетания по m элементов из n элементов. Тогда

$$A = C_{10}^4 = \frac{10!}{4! \cdot 6!} = 210,$$

$$B = C_8^2 = \frac{8!}{2! \cdot 6!} = 28.$$

Тогда

$$p = \frac{B}{A} = \frac{28}{210} = \frac{2}{15}.$$

Ответ: $p = \dfrac{2}{15}$.

Задача VI.1.2.4. (20 баллов)

Темы: квадратный трехчлен.

Условие

Решить уравнение $f(x) = g(x)$, где $f(x)$ и $g(x)$ — различные квадратные трехчлены со старшим коэффициентом, равным 1, для которых выполняется равенство

$$f(2022) + f(2023) + f(2024) = g(2022) + g(2023) + g(2024).$$

Критерии оценивания

1. Получено равенство (VI.1.2) — 5 баллов.
2. Получено равенство (VI.1.3) — 5 баллов.
3. Сделано замечание, что $p_1 - p_2 \neq 0$ и $q_1 - q_2 \neq 0$ — 5 баллов.
4. Из равенств (VI.1.2) и (VI.1.3) сделан вывод, что $x = 2023$ — 5 баллов (баллы суммируются).

Решение

Пусть $f(x) = x^2 + p_1 x + q_1$ и $g(x) = x^2 + p_2 x + q_2$. Тогда уравнение $f(x) = g(x)$ можно переписать в виде $f(x) - g(x) = 0$, или

$$x(p_1 - p_2) + (q_1 - q_2) = 0. \tag{VI.1.2}$$

Распишем заданное соотношение в условии задачи

$$f(2022) + f(2023) + f(2024) = g(2022) + g(2023) + g(2024),$$

$$2022^2 + 2023^2 + 2024^2 + (2022 + 2023 + 2024)p_1 + 3q_1 =$$

$$= 2022^2 + 2023^2 + 2024^2 + (2022 + 2023 + 2024)p_2 + 3q_2,$$

$$6069p_1 + 3q_1 = 6069p_2 + 3q_2.$$

$$2023(p_1 - p_2) + (q_1 - q_2) = 0. \qquad \text{(VI.1.3)}$$

Так как $f(x)$ и $g(x)$ — различные квадратные трехчлены, то $p_1 - p_2 \neq 0$ и $q_1 - q_2 \neq 0$.

Иначе, если бы $p_1 - p_2 = 0$, то из равенства (VI.1.2) следовало бы, что и $q_1 - q_2 = 0$, если же и $q_1 - q_2 = 0$, то из равенства (VI.1.3) следовало бы, что $p_1 - p_2 = 0$. В обоих случаях многочлены $f(x)$ и $g(x)$ бы совпали, и это противоречит условию задачи. Тогда равенства (VI.1.2) и (VI.1.3) совпадают при $x = 2023$, т. е. $f(2023) = g(2023)$. Тем самым уравнение $f(x) = g(x)$ имеет единственное решение $x = 2023$.

Ответ: 2023.

Задача VI.1.2.5. (25 баллов)

Условие

Найдите все пары простых чисел (p, q), для которых число $p^2 + 5pq + 4q^2$ является полным квадратом.

Критерии оценивания

1. Получено уравнение (VI.1.4) — 8 баллов.
2. Получена система уравнений (VI.1.5) — 6 баллов.
3. Получено равенство (VI.1.6) — 5 баллов.
4. Из равенства (VI.1.6) найдены все пары простых чисел (p, q) — 6 баллов.
5. За потерю какой-либо пары чисел — штраф 2 балла. Приведено полное решение задачи — 25 баллов (как сумма баллов всех пунктов).

Решение

Если такие пары (p, q) существуют, то справедливо равенство

$$p^2 + 5pq + 4q^2 = k^2, \text{ где } k \in N.$$

Тогда

$$k^2 - p^2 - 4pq - 4q^2 = pq,$$
$$k^2 - (p + 2q)^2 = pq,$$
$$(k - p - 2q)(k + p + 2q) = pq. \qquad \text{(VI.1.4)}$$

Левая часть равенства — произведение двух сомножителей, а правая — произведение двух простых чисел, значит, для левой части возможны следующие варианты: $1 \cdot pq$, $p \cdot q$, $q \cdot p$, $pq \cdot 1$. Но $k + p + 2q$ заведомо больше и p, и q, следовательно, возможен только первый вариант. То есть

$$\begin{cases} k - p - 2q = 1, \\ k + p + 2q = pq. \end{cases} \qquad \text{(VI.1.5)}$$

Вычитая из второго уравнения первое, получим

$$2p + 4q = pq - 1 \text{ или } (p - 4)(q - 2) = 9. \qquad \text{(VI.1.6)}$$

Для левой части возможны варианты: $9 \cdot 1$, $1 \cdot 9$, $3 \cdot 3$. Соответственно, $(p, q) \in \{(13, 3), (5, 11), (7, 5)\}$, все значения удовлетворяют условию.

Ответ: $(p, q) \in \{(13, 3), (5, 11), (7, 5)\}$.

Математика. 10–11 классы

Задача VI.1.3.1. (15 баллов)

Темы: векторы.

Условие

В пространстве заданы 9 точек A_1, A_2, A_3, …, A_9 с целочисленными координатами. Докажите, что найдется вектор $\overrightarrow{A_iA_j}, i, j \in 1, 2, \ldots, 9$, координаты которого — четные числа.

Критерии оценивания

1. Рассмотрены частные случаи координат точек — 0 баллов.
2. Перечислены в зависимости от четности все возможные наборы целочисленных координат точек в пространстве, дальнейших продвижений нет, — 5 баллов.
3. Согласно принципу Дирихле, из предыдущего пункта сделан вывод о существовании вектора с четными координатами — 10 баллов (баллы суммируются).

Доказательство

Каждая целочисленная координата точки может быть четной или нечетной. Тогда запишем все возможные варианты координат точек в пространстве в зависимости от четности (ч, ч, ч), (ч, ч, н), (ч, н, ч), (ч, н, н), (н, ч, ч), (н, ч, н), (н, н, ч), (н, н, н).

Всего 8 вариантов, а точек задано 9. Тогда согласно принципу Дирихле по меньшей мере две точки из 9 будут иметь одинаковую «сигнатуру». Следовательно, координаты вектора с началом и концом в этих точках – четные.

Задача VI.1.3.2. (20 баллов)

Темы: условная вероятность.

Условие

При подготовке к зачету по теории вероятностей Петя выучил 12 вопросов из 16. Преподаватель последовательно задает ему несколько вопросов. Зачет считается сданным, если Петя правильно ответит на два заданных ему вопроса. Оказалось, что после первых двух заданных преподавателем вопроса Петя еще не получил зачет. Какова вероятность, что зачет будет получен после третьего вопроса?

Критерии оценивания

1. За арифметические ошибки, не влияющие на логику рассуждений, — штраф до 5 баллов.

Решение

Пусть событие A — «Петя не получил зачет после первых двух заданных ему вопроса». Его вероятность

$$P(A) = 1 - \frac{12}{16} \cdot \frac{11}{15} = \frac{9}{20}.$$

Событие B — «Петя получил зачет после третьего вопроса». В этом случае его ответы на вопросы могут распределяться как $\{+; -; +\}$ или $\{-; +; +\}$, где «+» означает, что Петя ответил на вопрос правильно, а «−» — соответственно неправильно. Вероятность данного события равна

$$P(B \cap A) = \frac{12}{16} \cdot \frac{4}{15} \cdot \frac{11}{14} + \frac{4}{16} \cdot \frac{12}{15} \cdot \frac{11}{14} = \frac{11}{35}.$$

Тогда условная вероятность события

$$P(B|A) = \frac{P(B \cap A)}{P(A)} = \frac{11}{35} : \frac{9}{20} = \frac{44}{63}.$$

Ответ: $\frac{44}{63}$.

Задача VI.1.3.3. (20 баллов)

Темы: алгебра.

Условие

Пусть a_1, a_2, …, a_{2024} — члены арифметической прогрессии, ни один из которых не равен нулю. Найдите значение x, при котором будет выполняться равенство

$$\frac{1}{a_1 a_2} + \frac{1}{a_2 a_3} + \cdots + \frac{1}{a_{2023} a_{2024}} = \frac{x}{a_1 a_{2024}}.$$

Критерии оценивания

1. Каждое слагаемое исходной суммы в левой части представлено в виде $\frac{1}{a_k a_{k+1}} = \frac{1}{d}\left(\frac{1}{a_k} - \frac{1}{a_{k+1}}\right)$ — 7 баллов.
2. Найдена сумма в левой части уравнения — 7 баллов.
3. Найдено решение преобразованного уравнения — 6 баллов (баллы всех пунктов суммируются).

Решение

Пусть d — разность арифметической прогрессии. Преобразуем дроби в левой части равенства

$$\frac{1}{a_1 a_2} = \frac{1}{d}\left(\frac{d}{a_1 a_2}\right) = \frac{1}{d}\cdot\frac{a_2 - a_1}{a_1 a_2} = \frac{1}{d}\left(\frac{1}{a_1} - \frac{1}{a_2}\right).$$

Аналогично,

$$\frac{1}{a_2 a_3} = \frac{1}{d}\left(\frac{1}{a_2} - \frac{1}{a_3}\right).$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$\frac{1}{a_{2023} a_{2024}} = \frac{1}{d}\left(\frac{1}{a_{2023}} - \frac{1}{a_{2024}}\right).$$

Складывая данные равенства, получаем

$$\frac{1}{a_1 a_2} + \frac{1}{a_2 a_3} + \ldots + \frac{1}{a_{2023} a_{2024}} = \frac{1}{d}\left(\frac{1}{a_1} - \frac{1}{a_2} + \frac{1}{a_2} - \frac{1}{a_3} + \ldots + \frac{1}{a_{2023}} - \frac{1}{a_{2024}}\right) =$$

$$= \frac{1}{d}\left(\frac{1}{a_1} - \frac{1}{a_{2024}}\right) = \frac{1}{d}\frac{a_{2024} - a_1}{a_1 a_{2024}} = \frac{1}{d}\cdot\frac{2023d}{a_1 a_{2024}} = \frac{2023}{a_1 a_{2024}}.$$

Сравнивая полученный результат с правой частью исходного равенства, получаем, что $x = 2023$.

Ответ: 2023.

Задача VI.1.3.4. (20 баллов)

Темы: планиметрия, геометрическая вероятность.

Условие

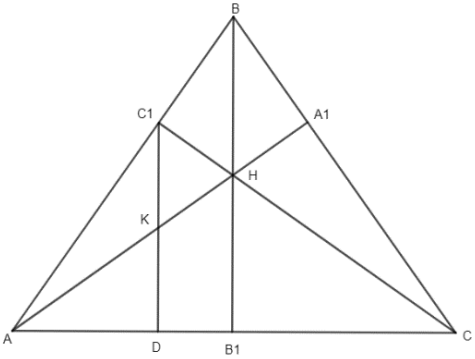
Через основание C_1 высоты CC_1 треугольника ABC проведена прямая, параллельная высоте BB_1, которая пересекает высоту AA_1 в точке K. Найдите вероятность того, что случайно брошенная точка в треугольник ABC, попадет в треугольник C_1HK, где H — ортоцентр треугольника ABC и $AB = 5$, $BC = 6$, $AC = \sqrt{41}$.

Критерии оценивания

1. Высказана идея, что искомая вероятность равна отношению площадей треугольников C_1HK и ABC — 3 балла.
2. Доказано, что треугольники C_1HK и ABC подобны — 7 баллов.
3. Найден коэффициент подобия треугольников C_1HK и ABC — 10 баллов.
4. В ходе верных рассуждений получена вычислительная ошибка, что привело к неверному ответу — штраф 2 балла.
5. Получено полное решение исходной задачи — 20 баллов (как сумма баллов всех пунктов).

Решение

Согласно определению геометрической вероятности, вероятность попадания в треугольник C_1HK случайно брошенной в треугольник ABC точки равна отношению площадей данных треугольников. Тогда задачу можно свести к поиску данного отношения.



Заметим, что

$$\angle C_1HK = 180^\circ - \angle C_1HA_1 = 180^\circ - (360^\circ - \angle HC_1B - HA_1B - \angle ABC) = \angle ABC.$$

$$\angle KC_1H = \angle DC_1C = 90^\circ - \angle DCC_1 = 90^\circ - \angle ACC_1 = \angle C_1AC = \angle BAC.$$

Таким образом, треугольники C_1HK и ABC подобны по двум углам. Тогда отношение площадей данных двух треугольников равно квадрату их коэффициента подобия, т. е.

$$k^2 = \left(\frac{C_1H}{AB}\right)^2 = \left(\frac{C_1H}{5}\right)^2 = \frac{C_1H^2}{25}.$$

Длину отрезка C_1H найдем из треугольника AC_1H:

$$C_1H = AC_1 \cdot \operatorname{ctg}(\angle C_1HA) = AC_1 \cdot \operatorname{ctg}(\angle ABC).$$

По теореме косинусов из треугольника ABC найдем $\cos(\angle ABC)$:

$$AC^2 = AB^2 + BC^2 - 2 \cdot AB \cdot BC \cdot \cos(\angle ABC),$$

$$41 = 25 + 36 - 60\cos(\angle ABC),$$

$$\cos(\angle ABC) = \frac{1}{3}.$$

Тогда

$$\sin(\angle ABC) = \sqrt{1 - \cos^2(\angle ABC)} = \frac{2\sqrt{2}}{3},$$

$$\operatorname{ctg}(\angle ABC) = \frac{\cos(\angle ABC)}{\sin(\angle ABC)} = \frac{1}{3} : \frac{2\sqrt{2}}{3} = \frac{1}{2\sqrt{2}} = \frac{\sqrt{2}}{4}.$$

Из треугольника CC_1B:

$$C_1B = BC \cdot \cos(\angle C_1BC) = BC \cdot \cos(\angle ABC) = 6 \cdot \frac{1}{2} = 3,$$

$$AC_1 = AB - C_1B = 5 - 3 = 2.$$

Тогда $C_1H = \dfrac{2\sqrt{2}}{4} = \dfrac{\sqrt{2}}{2}$ и $k^2 = \left(\dfrac{\sqrt{2}}{2}\right)^2 \cdot \dfrac{1}{25} = \dfrac{1}{50} = 0,02.$

Таким образом, вероятность попадания в треугольник C_1KH случайно брошенной в треугольник ABC точки равна 0,02.

Ответ: 0,02.

Задача VI.1.3.5. (25 баллов)

Темы: задача оптимизации.

Условие

Два пешехода, находящиеся в начальный момент времени на расстоянии 20 км друг от друга, движутся с постоянными скоростями в своих фиксированных прямолинейных направлениях. Через сколько минут от начального момента времени расстояние между ними будет минимально, если известно, что через 35 мин от старта оно было 15 км, а еще через 20 мин — 13 км? Найдите это расстояние.

Критерии оценивания

1. Записана функция изменения расстояния между пешеходами, дальнейших продвижений нет — 5 баллов.
2. Найдена точка минимума данной функции в зависимости от параметров — 5 баллов.
3. Исходя из условий задачи найдены параметры, которые определяют движение пешеходов — 10 баллов.
4. Получены числовые значения минимума функции и ее наименьшее значение — 5 баллов.
5. При правильной логике размышлений получены вычислительные ошибки — штраф до 5 баллов.

Решение

Введем в начальный момент времени прямоугольную декартову систему координат, в начале которой располагается первый пешеход, а на оси абсцисс в точке с координатами $(20; 0)$ находится второй. Будем рассматривать каждую координату пешехода как некоторую линейную функцию от времени. Тогда движение пешеходов может быть описано следующими системами условий.

$$\text{I: } \begin{cases} x_1(t) = a_1 t, \\ y_1(t) = b_1 t, \end{cases} \quad \text{и II: } \begin{cases} x_2(t) = 20 + a_2 t, \\ y_2(t) = b_2 t. \end{cases}$$

В этом случае расстояние между пешеходами может быть задано формулой

$$f(t) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

Подставляя координаты пешеходов, получим

$$f(t) = \sqrt{(20 + (a_2 - a_1)t)^2 + ((b_2 - b_1)t)^2}.$$

Полученную функцию необходимо минимизировать. Точку минимума функции можно найти из уравнения $\dfrac{df(t)}{dt} = 0$. Найдем производную функции $f(t)$ и приравняем ее к нулю:

$$\frac{df(t)}{dt} = \frac{(20 + (a_2 - a_1)t)(a_2 - a_1) + (b_2 - b_1)^2 t}{\sqrt{(20 + (a_2 - a_1)t)^2 + ((b_2 - b_1)t)^2}} = 0.$$

Так как в числителе данной дроби стоит линейное выражение относительно t, старший коэффициент которого $(a_2 - a_1)^2 + (b_2 - b_1)^2 > 0$ (условия $a_2 = a_1$ и $b_2 = b_1$ не могут выполняться одновременно, иначе функция $f(t) = 20$ для любого t), а знаменатель дроби всегда принимает положительное значение, то корень данного уравнения является точкой минимума функции $f(t)$ (левее этой точки производная принимает отрицательные значения, а правее — положительные). Тогда минимум функции достигается в точке

$$t_0 = -\frac{20(a_2 - a_1)}{(a_2 - a_1)^2 + (b_2 - b_1)^2}.$$

Для окончательного определения значения t_0 найдем значения выражений $(a_2 - a_1)$ и $(a_2 - a_1)^2 + (b_2 - b_1)^2$.

Из условия задачи следует, что $f(35) = 15$ и $f(55) = 13$. Составим соответствующую систему уравнений

$$\begin{cases} (20 + 35(a_2 - a_1))^2 + (35(b_2 - b_1))^2 = 15^2, \\ (20 + 55(a_2 - a_1))^2 + (55(b_2 - b_1))^2 = 13^2. \end{cases}$$

Введем в данной системе замену переменных

$$\begin{cases} a_2 - a_1 = u, \\ b_2 - b_1 = v. \end{cases}$$

Тогда она примет вид

$$\begin{cases} (20 + 35u)^2 + 7^2 \cdot 5^2 \cdot v^2 = 15^2, \\ (20 + 55u)^2 + 11^2 \cdot 5^2 \cdot v^2 = 13^2. \end{cases}$$

Умножим первое уравнение на 11^2, второе — на 7^2, и вычтем из первого второе. В результате получим

$$11^2 \cdot (20 + 35u)^2 - 7^2 \cdot (20 + 55u)^2 = 15^2 \cdot 11^2 - 13^2 \cdot 7^2,$$

$$(220 + 385u - 140 - 385u)(220 + 385u + 140 + 385u) = (165 - 91)(165 + 91),$$

$$80(360 + 770u) = 74 \cdot 256,$$

$$u = -\frac{4}{25}.$$

Подставим найденное значение u в первое уравнение и найдем v^2:

$$\left(\frac{72}{5}\right)^2 + 35^2 \cdot v^2 = 15^2,$$

$$v^2 = \frac{9}{625}.$$

Тогда

$$u^2 + v^2 = 16/625 + \frac{9}{625} = \frac{1}{25}.$$

Возвращаемся обратно к замене переменных и получаем:

$$a_2 - a_1 = -\frac{4}{25},$$

$$(a_2 - a_1)^2 + (b_2 - b_1)^2 = \frac{1}{25}.$$

Подставляем полученные значения в выражение для точки минимума функции $f(t)$:

$$t_0 = -\frac{20(a_2 - a_1)}{(a_2 - a_1)^2 + (b_2 - b_1)^2} = -\frac{20 \cdot \left(-\frac{4}{25}\right)}{\frac{1}{25}} = 80.$$

Находим минимальное расстояние, которое будет между пешеходами через 80 минут:

$$f(80) = \sqrt{\left(20 + 80 \cdot \left(-\frac{4}{25}\right)\right)^2 + 80^2 \cdot \frac{9}{625}} = \sqrt{\left(\frac{180}{25}\right)^2 + \left(\frac{240}{25}\right)^2} = 12.$$

Тем самым получаем ответ на задачу: минимальное расстояние между пешеходами будет составлять 12 км через 80 мин от начального момента времени.

Ответ: минимальное расстояние между пешеходами будет составлять 12 км через 80 мин от начального момента времени.

Инженерный тур

Общая информация

Финалистам предстоит поучаствовать в процессе поиска новых радиофармпрепаратов — лекарств, использующих радиоактивность для диагностики и терапии онкозаболеваний. Для успешной работы эти лекарства должны доставлять медицинский радионуклид к раковой опухоли, и финалистам предстоит найти именно те молекулы, которые смогут сделать это наиболее эффективно.

Лекарство не должно оказаться опаснее болезни, поэтому на молекулы, которые будут предлагать участники, будут накладываться дополнительные ограничения.

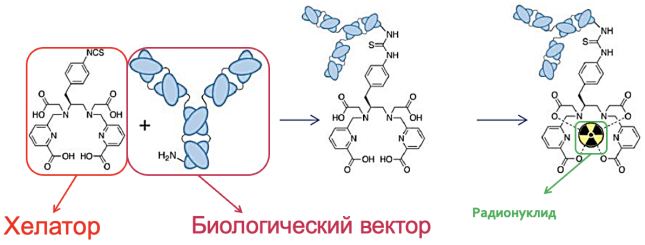
Легенда задачи

В оригинале задача относится к области фармацевтики и связана со знаниями биохимии поведения раковой опухоли. Однако разбиение ее на отдельные этапы приводит нас к химической задаче связывания медицинского радионуклида. При этом исключительно химическое ее решение является очень ресурснозатратным, и использование технологий искусственного интеллекта помогает найти потенциальные лекарства значительно быстрее. Наилучшие решения способна дать интеграция знания химии и биологии с одной стороны и технологий искусственного интеллекта — с другой.

Радиофармпрепараты состоят из трех частей:

- биологического вектора, отвечающего за доставку лекарства;
- радионуклида, реализующего саму терапию;
- хелатора, связывающего эти две части.

В задаче участникам предстоит найти такие хелаторы, которые могли бы наиболее прочно связывать медицинские радионуклиды



Требования к команде и компетенциям участников

Количество участников в команде: 2 человека.

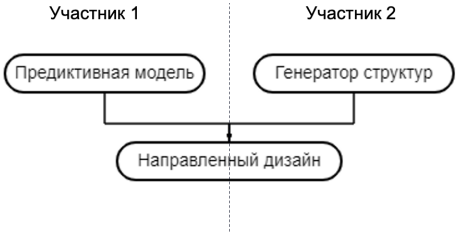
Компетенции, которыми должны обладать члены команды:

- анализ данных;
- методы QSAR;
- химические базы данных.

Необходимо уметь работать:

- с молекулярными данными (например `rdkit`);
- с методами машинного обучения (например, `scikit-learn`), в т. ч. с нейронными сетями (например, `pytorch`).

Роли, которые должны быть представлены в команде.



Оборудование и программное обеспечение

Участникам предоставляются ноутбуки:

- оперативная память — 16 Гб;
- хранилище SSD — 256 Гб;
- процессор — Intel Core i5.

Предустановленное ПО:

- Браузер Chrome;
- PyCharm:
- Jupyter Lab;
- Python 3.5;
- GIT;
- MS Office 2016.

Описание задачи

Задача

Финалистам предстоит поучаствовать в процессе поиска новых радиофармпрепаратов — лекарств, использующих радиоактивность для диагностики и терапии онкозаболеваний. Для успешной работы эти лекарства должны доставлять медицинский радионуклид к раковой опухоли, и финалистам предстоит найти именно те молекулы, которые смогут сделать это наиболее эффективно. Кроме того, не стоит забывать, что лекарство не должно оказаться опаснее болезни, а значит, на молекулы, которые будут предлагать участники, будут накладываться дополнительные ограничения.

Предоставляемые данные

Набор молекул в текстовом представлении (`SMILES`) и соответствующих значений эффективности связывания медицинского радионуклида.

Цель

Генерация выборки из 100 молекул с максимальным значением целевого свойства.

Дополнительные условия

Выборка молекул должна удовлетворять следующим критериям:

- Каждая молекула должна включать только элементы из списка: C, H, O, N, P, S.
- Каждая молекула должна включать не менее трех разных элементов из списка выше.
- Каждая молекула должна включать суммарно не более 12 атомов следующих элементов: O, N, P, S.
- Молекулы не должны встречаться в тренировочном наборе.
- Молекулярная масса каждой молекулы $\leqslant 500$ Да.
- Индекс синтетической доступности каждой молекулы< 5 (рассчитывается согласно инструкции https://greglandrum.github.io/rdkit-blog/posts/2023-12-01-using_sascore_and_npscore.html).
- Средняя попарная схожесть молекул в наборе $< 0,5$ (рассчитывается согласно метрике Танимото со значениями параметров «по умолчанию», как описано в инструкции https://www.rdkit.org/docs/GettingStartedInPython.html).

В рамках решения задачи выделяются два промежуточных этапа, которые могут выполняться двумя членами команды параллельно для достижения наибольшей эффективности.

Этап 1

Один из участников может заниматься построением предсказательной модели, обученной на выданных участникам данных. Наилучшая предсказательная модель обеспечивает максимальное качество отбора молекул в итоговое решение. При этом участники уже имеют опыт построения предсказательных моделей на подготовительных этапах, однако задача финала осложняется нехваткой данных, и участникам необходимо выбрать метод решения этой проблемы.

Команда, успешно справившаяся только с этим этапом, может в дальнейшем использовать эту модель для скрининга открытых баз данных, что потенциально позволяет получить решение всей задачи. Однако качество решения будет заметно меньше, чем у участников, решивших оба этапа.

Команда, не справившаяся с построением предсказательной модели, может рассчитывать только на ненаправленный (случайный) поиск решения или на построение оценщика сгенерированных молекул на основе детерминированных правил, что, даже в случае знаний химии, заметно превышающих требуемые для решения задачи, все равно не позволяет получить решение, способное войти в призовые.

Этап 2

Вторым этапом решения задачи является построение генеративной модели, способной вести направленный поиск молекул, отвечающих заданным условиям. Более простая версия подобной задачи ранее предлагалась участникам, однако задача финала осложняется необходимостью создания модели, способной учитывать сразу несколько ограничений на генерируемые объекты, при этом различных типов: континуальные или категориальные.

Участники, успешно справившиеся с построением подобной модели, могут вести направленный поиск решений даже с предсказательной моделью (полученной на первом этапе) не лучшего качества, хотя, конечно, проиграли бы участниками, успешно решившим оба этапа. Отсутствие такой модели ограничивает выбор молекул скринингом существующих баз данных и случайной или ручной генерацией новых молекул, что, особенно в условиях ограниченного числа попыток, с очень низкой вероятностью может привести к конкурентоспособному решению.

Система оценивания

Проверка предложенных молекул происходит на соответствие заявленным условиям, оценка целевого свойства на подвыборке из предложенной выборки. Существует закрытая от участников модель, способная проводить оценку целевого свойства молекул, предложенных участниками. Наборы молекул будут анализироваться этой моделью, и рейтингом участников будет являться среднее значение свойства на лучшей половине сгенерированных молекул. В качестве метрики используется среднее значение эффективности связывания медицинского радионуклида, оцененное по выборке данных из набора, загруженного участниками. Для расчета этого значения использовалось закрытое от участников программное обеспечение.

Решение задачи

В качестве примера предлагается решение команды-победителя. Было использовано представление молекул в машинно-читаемом виде на базе расчетных физико-химических дескрипторах, полученных из библиотеки `rdkit`, а также метод градиентного бустинга для построения предсказательной модели и вариационный автокодировщик для построения генеративной модели.

Из-за большого объема написанного участниками кода и наличия дополнительных файлов, неконвертируемых в текстовый формат, в данном разделе приводятся основные элементы решения, демонстрирующие подход, предложенный участниками. Полное решение в оригинальном виде доступно по ссылке: https://disk.yandex.ru/d/dFaLso3lZKax2g.

Большая часть решения выполнялась на предоставленных ноутбуках, для выполнения наиболее ресурсоемкой части участники использовали вычислительные ресурсы, предоставляемые в открытом доступе.

```
# импорт необходимых библиотек:

import pandas as pd
import numpy as np
from xgboost import XGBRegressor
from sklearn.model_selection import train_test_split
from rdkit import Chem
from rdkit.Chem import PandasTools, Descriptors
from rdkit.ML.Descriptors.MoleculeDescriptors import MolecularDescriptorCalculator
from joblib import load, dump
import seaborn as sns
import matplotlib.pyplot as plt

# отбор дескрипторов для описания молекулярных данных в формате, пригодном для
↪  тренировки моделей машинного обучения
st = [
 'MaxAbsEStateIndex',
 'MinAbsEStateIndex',
 'MinEStateIndex',
 'qed',
 'MolWt',
 'NumRadicalElectrons',
 'MaxPartialCharge',
 'MinPartialCharge',
 'MaxAbsPartialCharge',
 'FpDensityMorgan1',
 'FpDensityMorgan3',
 'BCUT2D_MWHI',
 'BCUT2D_MWLOW',
 'BCUT2D_CHGHI',
 'BCUT2D_LOGPLOW',
 'BCUT2D_MRLOW',
 'AvgIpc',
 'BalabanJ',
 'Chi4n',
 'PEOE_VSA1',
 'PEOE_VSA12',
 'PEOE_VSA13',
 'PEOE_VSA2',
 'PEOE_VSA3',
```

```
    'PEOE_VSA5',
    'PEOE_VSA6',
    'PEOE_VSA7',
    'SMR_VSA1',
    'SMR_VSA10',
    'SMR_VSA3',
    'SMR_VSA5',
    'SMR_VSA7',
    'SMR_VSA8',
    'SMR_VSA9',
    'SlogP_VSA1',
    'SlogP_VSA11',
    'SlogP_VSA2',
    'SlogP_VSA7',
    'SlogP_VSA8',
    'SlogP_VSA9',
    'TPSA',
    'EState_VSA10',
    'EState_VSA3',
    'EState_VSA4',
    'EState_VSA5',
    'EState_VSA6',
    'EState_VSA7',
    'EState_VSA8',
    'EState_VSA9',
    'VSA_EState2',
    'VSA_EState3',
    'VSA_EState4',
    'VSA_EState5',
    'VSA_EState7',
    'VSA_EState8',
    'FractionCSP3',
    'NHOHCount',
    'NumAliphaticCarbocycles',
    'NumAromaticHeterocycles',
    'NumAromaticRings',
    'NumRotatableBonds',
    'RingCount',
    'fr_Al_COO',
    'fr_Al_OH',
    'fr_ArN',
    'fr_Ar_NH',
    'fr_NHO',
    'fr_NH1',
    'fr_NH2',
    'fr_alkyl_halide',
    'fr_allylic_oxid',
    'fr_amide',
    'fr_azide',
    'fr_bicyclic',
    'fr_diazo',
    'fr_ester',
    'fr_ether',
    'fr_halogen',
    'fr_imidazole',
    'fr_isocyan',
    'fr_para_hydroxylation',
    'fr_piperdine',
    'fr_priamide',
    'fr_prisulfonamd',
```

```
  'fr_quatN',
  'fr_sulfonamd',
  'fr_unbrch_alkane']

def my_mol2vec(smiles):
    global st
    mol = Chem.MolFromSmiles(smiles)
    chosen_descriptors = st
    mol_descriptor_calculator = MolecularDescriptorCalculator(chosen_descriptors)
    list_of_descriptor_vals = list(mol_descriptor_calculator.CalcDescriptors(mol))
    return list_of_descriptor_vals

#обучение предсказательной модели:

def eval_model(model_, X_test, y_test):
    pred = model_.predict(X_test)
    summ = 0
    for i in range(len(pred)):
        summ += (pred[i] - y_test.to_numpy()[i])**2
    return (summ/len(y_test))**0.5
def save_model(name, model):
    dump(model, f'{name}.joblib')
def my_fit(X_train, X_test, y_train, y_test, rs=123):
    model = XGBRegressor(n_estimators=500, max_depth=2, learning_rate=0.03,
    ↪  subsample=0.5, seed=rs, num_parallel_tree=16)#, early_stopping_rounds=10)

    model.fit(X_train, y_train,
                early_stopping_rounds=100,
                eval_set=[(X_test, y_test)], verbose=100)
    return model, eval_model(model, X_test, y_test)

def my_fit2(X, y):
    model = XGBRegressor(n_estimators=2000, max_depth=2, learning_rate=0.03,
    ↪  subsample=0.5, seed=123, num_parallel_tree=16)#, early_stopping_rounds=10)

    model.fit(X, y)
    return model

model, eval_ = my_fit(X_train, X_test, y_train, y_test)
save_model('lgK_hope_to_ans', model)

# Проверка выполнения условий задачи

def only_permitted_elements(smiles):
  mol = Chem.MolFromSmiles(smiles)
  for atom in mol.GetAtoms():
    if not (atom.GetSymbol() in ['C', 'H', 'O', 'N', 'P', 'S']):
      return False

  return True

def only_permitted_elements_count_min(smiles):
  mol = Chem.MolFromSmiles(smiles)
  unique_el = []

  for atom in mol.GetAtoms():
    if not (atom.GetSymbol() in unique_el):
      unique_el.append(atom.GetSymbol())

  return len(unique_el) >= 3
```

```
158
159 def only_permitted_elements_count_max(smiles):
160   mol = Chem.MolFromSmiles(smiles)
161   d = {'O': 0, 'N': 0, 'P': 0, 'S': 0}
162
163   for atom in mol.GetAtoms():
164     if atom.GetSymbol() in d.keys():
165       d[atom.GetSymbol()] += 1
166
167   sum_el = sum(d.values())
168   return sum_el <= 12
169
170 def MW(smiles):
171   mol = Chem.MolFromSmiles(smiles)
172   MW = Descriptors.MolWt(mol)
173
174   return MW <= 500
175
176 def check_sascore(smiles):
177   mol = Chem.MolFromSmiles(smiles)
178   try:
179     sascore = sascorer.calculateScore(mol)
180   except:
181     return False
182
183   return sascore < 5
184
185 def average_similarity(smiles):
186   simis =[]
187   for i in range(len(smiles)):
188     for j in range(i+1, len(smiles)):
189       smi1 = smiles[i]
190       smi2 = smiles[j]
191
192       # create molecule using given smiles
193       mol1 = Chem.MolFromSmiles(smi1)
194       mol2 = Chem.MolFromSmiles(smi2)
195
196       # generate fingerprints
197       fpgen = GetRDKitFPGenerator()
198       fp1 = fpgen.GetFingerprint(mol1)
199       fp2 = fpgen.GetFingerprint(mol2)
200
201       # compute Tanimoto Similarity using fingerprints
202       simi = DataStructs.TanimotoSimilarity(fp1, fp2)
203
204       simis.append(simi)
205
206   return (sum(simis)/len(simis))
207
208 def all_check(smiles):
209   flags = [only_permitted_elements(smiles),
      ↪  only_permitted_elements_count_min(smiles),
      ↪  only_permitted_elements_count_max(smiles), MW(smiles),
      ↪  check_sascore(smiles)]
210
211   return all(flags)
212
213 # создание вариационного автокодировщика для использования в качестве генеративной
    ↪  модели
```

```
214
215 # -*- coding: utf-8 -*-
216 import torch
217 import torch.nn as nn
218 import torch.nn.functional as F
219
220 q_bidir = True
221 q_d_h = 256
222 q_n_layers = 1
223 q_dropout = 0.5
224 d_n_layers = 3
225 d_dropout = 0
226 d_z = 128
227 d_d_h = 512
228
229 class VAE(nn.Module):
230   def __init__(self,vocab,vector):
231     super().__init__()
232     self.vocabulary = vocab
233     self.vector = vector
234
235     n_vocab, d_emb = len(vocab), vector.size(1)
236     self.x_emb = nn.Embedding(n_vocab, d_emb, c2i['<pad>'])
237     self.x_emb.weight.data.copy_(vector)
238
239     #ENCODER
240     self.encoder_rnn =
      ↪  nn.GRU(d_emb,q_d_h,num_layers=q_n_layers,batch_first=True,dropout=q_dropout
      ↪  if q_n_layers > 1 else 0,bidirectional=q_bidir)
241     q_d_last = q_d_h * (2 if q_bidir else 1)
242     self.q_mu = nn.Linear(q_d_last, d_z)
243     self.q_logvar = nn.Linear(q_d_last, d_z)
244
245     # Decoder
246     self.decoder_rnn = nn.GRU(d_emb +
      ↪  d_z,d_d_h,num_layers=d_n_layers,batch_first=True,dropout=d_dropout if
      ↪  d_n_layers > 1 else 0)
247     self.decoder_latent = nn.Linear(d_z, d_d_h)
248     self.decoder_fullyc = nn.Linear(d_d_h, n_vocab)
249
250     # Grouping the model's parameters
251     self.encoder = nn.ModuleList([self.encoder_rnn,self.q_mu,self.q_logvar])
252     self.decoder =
      ↪  nn.ModuleList([self.decoder_rnn,self.decoder_latent,self.decoder_fullyc])
253     self.vae = nn.ModuleList([self.x_emb,self.encoder,self.decoder])
254
255   @property
256   def device(self):
257     return next(self.parameters()).device
258
259   def string2tensor(self, string, device='model'):
260     ids = string2ids(string, add_bos=True, add_eos=True)
261     tensor = torch.tensor(ids, dtype=torch.long,device=self.device if device ==
      ↪  'model' else device)
262     return tensor
263
264   def tensor2string(self, tensor):
265     ids = tensor.tolist()
266     string = ids2string(ids, rem_bos=True, rem_eos=True)
267     return string
```

```
268
269     def forward(self,x):
270       global count_epoch
271       z, kl_loss = self.forward_encoder(x)
272       recon_loss = self.forward_decoder(x, z)
273       count_epoch += 1
274       if count_epoch % 100 == 0:
275         print(count_epoch)
276       return kl_loss, recon_loss
277
278     def forward_encoder(self,x):
279       x = [self.x_emb(i_x) for i_x in x]
280       x = nn.utils.rnn.pack_sequence(x)
281       _, h = self.encoder_rnn(x, None)
282       h = h[-(1 + int(self.encoder_rnn.bidirectional)):]
283       h = torch.cat(h.split(1), dim=-1).squeeze(0)
284       mu, logvar = self.q_mu(h), self.q_logvar(h)
285       eps = torch.randn_like(mu)
286       z = mu + (logvar / 2).exp() * eps
287       kl_loss = 0.5 * (logvar.exp() + mu ** 2 - 1 - logvar).sum(1).mean()
288       return z, kl_loss
289
290     def forward_decoder(self,x, z):
291       lengths = [len(i_x) for i_x in x]
292       x = nn.utils.rnn.pad_sequence(x, batch_first=True, padding_value=
          ↪  c2i['<pad>'])
293       x_emb = self.x_emb(x)
294       z_0 = z.unsqueeze(1).repeat(1, x_emb.size(1), 1)
295       x_input = torch.cat([x_emb, z_0], dim=-1)
296       x_input = nn.utils.rnn.pack_padded_sequence(x_input, lengths,
          ↪  batch_first=True)
297       h_0 = self.decoder_latent(z)
298       h_0 = h_0.unsqueeze(0).repeat(self.decoder_rnn.num_layers, 1, 1)
299       output, _ = self.decoder_rnn(x_input, h_0)
300       output, _ = nn.utils.rnn.pad_packed_sequence(output, batch_first=True)
301       y = self.decoder_fullyc(output)
302
303       recon_loss = F.cross_entropy(y[:, :-1].contiguous().view(-1, y.size(-1)),x[:,
          ↪  1:].contiguous().view(-1),ignore_index= c2i['<pad>'])
304       return recon_loss
305
306     def sample_z_prior(self,n_batch):
307       return torch.randn(n_batch,self.q_mu.out_features,device=
          ↪  self.x_emb.weight.device)
308     def sample(self,n_batch, max_len=100, z=None, temp=1.0):
309       with torch.no_grad():
310         if z is None:
311           z = self.sample_z_prior(n_batch)
312           z = z.to(self.device)
313           z_0 = z.unsqueeze(1)
314           h = self.decoder_latent(z)
315           h = h.unsqueeze(0).repeat(self.decoder_rnn.num_layers, 1, 1)
316           w = torch.tensor(c2i['<bos>'], device=self.device).repeat(n_batch)
317           x = torch.tensor([c2i['<pad>']], device=device).repeat(n_batch, max_len)
318           x[:, 0] = c2i['<bos>']
319           end_pads = torch.tensor([max_len], device=self.device).repeat(n_batch)
320           eos_mask = torch.zeros(n_batch, dtype=torch.bool, device=self.device)
321
322           for i in range(1, max_len):
323             x_emb = self.x_emb(w).unsqueeze(1)
```

```
            x_input = torch.cat([x_emb, z_0], dim=-1)

            o, h = self.decoder_rnn(x_input, h)
            y = self.decoder_fullyc(o.squeeze(1))
            y = F.softmax(y / temp, dim=-1)

            w = torch.multinomial(y, 1)[:, 0]
            x[~eos_mask, i] = w[~eos_mask]
            i_eos_mask = ~eos_mask & (w == c2i['<eos>'])

            end_pads[i_eos_mask] = i + 1
            eos_mask = eos_mask | i_eos_mask


            new_x = []
            for i in range(x.size(0)):
              new_x.append(x[i, :end_pads[i]])

        return [self.tensor2string(i_x) for i_x in new_x]

    from torch.optim.lr_scheduler import _LRScheduler
    import torch.optim as optim
    from torch.utils.data import DataLoader
    from torch.nn.utils import clip_grad_norm_
    import math
    import numpy as np
    from collections import UserList, defaultdict
    n_last = 1000
    n_batch = 32
    kl_start = 0
    kl_w_start = 0.0
    kl_w_end = 1.0
    n_epoch = 200
    n_workers = 0

    clip_grad = 50
    lr_start = 0.003
    lr_n_period = 10
    lr_n_mult = 1
    lr_end = 3 * 1e-4
    lr_n_restarts = 6

    def _n_epoch():
        return sum(lr_n_period * (lr_n_mult ** i) for i in range(lr_n_restarts))

    def _train_epoch(model, epoch, train_loader, kl_weight, optimizer=None):
        if optimizer is None:
            model.eval()
        else:
            model.train()

        kl_loss_values = CircularBuffer(n_last)
        recon_loss_values = CircularBuffer(n_last)
        loss_values = CircularBuffer(n_last)
        for i, input_batch in enumerate(train_loader):
            input_batch = tuple(data.to(device) for data in input_batch)

        #forward
            kl_loss, recon_loss = model(input_batch)
            loss = kl_weight * kl_loss + recon_loss
```

```
384         #backward
385             if optimizer is not None:
386                 optimizer.zero_grad()
387                 loss.backward()
388                 clip_grad_norm_(get_optim_params(model),clip_grad)
389                 optimizer.step()
390 
391             kl_loss_values.add(kl_loss.item())
392             recon_loss_values.add(recon_loss.item())
393             loss_values.add(loss.item())
394             lr = (optimizer.param_groups[0]['lr'] if optimizer is not None else None)
395 
396         #update train_loader
397             kl_loss_value = kl_loss_values.mean()
398             recon_loss_value = recon_loss_values.mean()
399             loss_value = loss_values.mean()
400             postfix = [f'loss={loss_value:.5f}', f'(kl={kl_loss_value:.5f}',
                ↪  f'recon={recon_loss_value:.5f})', f'klw={kl_weight:.5f} lr={lr:.5f}']
401         postfix = {'epoch': epoch,'kl_weight': kl_weight,'lr': lr,'kl_loss':
            ↪  kl_loss_value,'recon_loss': recon_loss_value,'loss': loss_value,'mode':
            ↪  'Eval' if optimizer is None else 'Train'}
402         return postfix
403 
404     def _train(model, train_loader, val_loader=None, logger=None):
405         optimizer = optim.Adam(get_optim_params(model),lr= lr_start)
406 
407         lr_annealer = CosineAnnealingLRWithRestart(optimizer)
408 
409         model.zero_grad()
410         for epoch in range(n_epoch):
411 
412             kl_annealer = KLAnnealer(n_epoch)
413             kl_weight = kl_annealer(epoch)
414             postfix = _train_epoch(model, epoch,train_loader, kl_weight, optimizer)
415             lr_annealer.step()
416     def fit(model, train_data, val_data=None):
417         logger = Logger() if False is not None else None
418         train_loader = get_dataloader(model,train_data,shuffle=True)
419 
420         val_loader = None if val_data is None else get_dataloader(model, val_data,
            ↪  shuffle=False)
421         _train(model, train_loader, val_loader, logger)
422         return model
423     def get_collate_device(model):
424         return model.device
425     def get_dataloader(model, train_data, collate_fn=None, shuffle=True):
426         if collate_fn is None:
427             collate_fn = get_collate_fn(model)
428             print(collate_fn)
429         return DataLoader(train_data, batch_size=n_batch, shuffle=shuffle,
            ↪  num_workers=n_workers, collate_fn=collate_fn)
430 
431     def get_collate_fn(model):
432         device = get_collate_device(model)
433 
434         def collate(train_data):
435             train_data.sort(key=len, reverse=True)
436             tensors = [string2tensor(string, device=device) for string in train_data]
437             return tensors
438 
```

```
439     return collate
440
441 def get_optim_params(model):
442     return (p for p in model.parameters() if p.requires_grad)
443
444 class KLAnnealer:
445     def __init__(self,n_epoch):
446         self.i_start = kl_start
447         self.w_start = kl_w_start
448         self.w_max = kl_w_end
449         self.n_epoch = n_epoch
450
451
452         self.inc = (self.w_max - self.w_start) / (self.n_epoch - self.i_start)
453
454     def __call__(self, i):
455         k = (i - self.i_start) if i >= self.i_start else 0
456         return self.w_start + k * self.inc
457
458 class CosineAnnealingLRWithRestart(_LRScheduler):
459     def __init__(self , optimizer):
460         self.n_period = lr_n_period
461         self.n_mult = lr_n_mult
462         self.lr_end = lr_end
463
464         self.current_epoch = 0
465         self.t_end = self.n_period
466
467         # Also calls first epoch
468         super().__init__(optimizer, -1)
469
470     def get_lr(self):
471         return [self.lr_end + (base_lr - self.lr_end) *
472                 (1 + math.cos(math.pi * self.current_epoch / self.t_end)) / 2
473                 for base_lr in self.base_lrs]
474
475     def step(self, epoch=None):
476         if epoch is None:
477             epoch = self.last_epoch + 1
478         self.last_epoch = epoch
479         self.current_epoch += 1
480
481         for param_group, lr in zip(self.optimizer.param_groups, self.get_lr()):
482             param_group['lr'] = lr
483
484         if self.current_epoch == self.t_end:
485             self.current_epoch = 0
486             self.t_end = self.n_mult * self.t_end
487
488 class CircularBuffer:
489     def __init__(self, size):
490         self.max_size = size
491         self.data = np.zeros(self.max_size)
492         self.size = 0
493         self.pointer = -1
494
495     def add(self, element):
496         self.size = min(self.size + 1, self.max_size)
497         self.pointer = (self.pointer + 1) % self.max_size
498         self.data[self.pointer] = element
```

```
499             return element
500
501         def last(self):
502             assert self.pointer != -1, "Can't get an element from an empty buffer!"
503             return self.data[self.pointer]
504
505         def mean(self):
506             return self.data.mean()
507
508     class Logger(UserList):
509         def __init__(self, data=None):
510             super().__init__()
511             self.sdata = defaultdict(list)
512             for step in (data or []):
513                 self.append(step)
514
515         def __getitem__(self, key):
516             if isinstance(key, int):
517                 return self.data[key]
518             elif isinstance(key, slice):
519                 return Logger(self.data[key])
520             else:
521                 ldata = self.sdata[key]
522                 if isinstance(ldata[0], dict):
523                     return Logger(ldata)
524                 else:
525                     return ldata
526
527         def append(self, step_dict):
528             super().append(step_dict)
529             for k, v in step_dict.items():
530                 self.sdata[k].append(v)
```

Материалы для подготовки

До финала был проведен хакатон «Молекула», в рамках которого участникам предоставлялась возможность погрузиться в увлекательный процесс разработки проекта для интерпретации химических формул. Задача — создать систему, способную распознавать изображения органических молекул и переводить их в формат SMILES — уникальный язык, преобразующий сложные химические соединения в легко читаемые текстовые строки, — также отработать все навыки в сфере машинного обучения, которые нужны для успешного решения задачи финала.

Тренировочные задачи: https://ai-academy.ru/training/training-tasks/.

Видеоуроки:

1. Уроки Академии ИИ (плейлист): https://www.youtube.com/playlist?list=PLxF_rYtB5vBC6MIx4Y3VX_O67jX0iG6U-.
2. Лекции по ИИ для школьников от Microsoft (плейлист): https://www.youtube.com/playlist?list=PL6XUtJhtlpPOJu1OTwZ4I7xmnoyqQ_BFR.
3. Курс по машинному обучению (основной уровень) от Академии ИИ: https://stepik.org/course/125587/promo.
4. Курс по машинному обучению (продвинутый уровень) от Академии ИИ: https://stepik.org/course/134942/promo.

5. Курс Docker и Git от Академии ИИ: https://ai-academy.ru/training/courses/docker-git/.
6. Курс «Введение в LinuxЮЮ от Академии ИИ: https://ai-academy.ru/training/courses/vvedenie-v-linux/promo/.
7. Курс «Математика для Data Science» от Академии ИИ: https://ai-academy.ru/training/courses/matematika-dlya-data-science/promo/.
8. «Готовый стекинг за вас»: https://www.kaggle.com/c/allstate-claims-severity/discussion/25743.
9. «Строим ансамбли с умом»: https://mlwave.com/kaggle-ensembling-guide/.
10. «Как правильно «фармить» Kaggle»: https://habr.com/ru/company/ods/blog/426227/.
11. «Как выиграть соревнование CatBoost'ом единым»: https://habr.com/ru/company/ods/blog/475182/.
12. 10 статей «Открытого курса ODS по машинному обучению»: https://habr.com/en/company/ods/blog/322626/.
13. Beginner's guide to create a Time Series Forecast: https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/.
14. CatBoost vs. Light GBM vs. XGBoos: t https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db.
15. Рекомендательные системы:
введение: https://habr.com/ru/post/476222/;
лайфхаки: https://habr.com/ru/post/476224/.

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 50 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 56,6 баллов, независимо от уровня.

Заключительный этап

Индивидуальный предметный тур

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

Командный инженерный тур

Команды заключительного этапа получали за командный инженерный тур от 0 до 100 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 $K_1 = 0,15$ каждый предмет и баллы за командное решение задач инженерного тура с весом $K_2 = 0,7$.

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 8 победителей и 18 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	67,04 и выше
Призеры	От 56,05 до 65,23

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».