



НТО

МАТЕРИАЛЫ ЗАДАНИЙ

Всероссийской междисциплинарной олимпиады школьников

«Национальная технологическая олимпиада»

по профилю

«Большие данные и машинное обучение»

2023/24 учебный год

<http://ntcontest.ru>

УДК 373.5.016:004.65
ББК 74.263.02
Б79

Авторы:

М.В. Бабушкин, А.А. Бойцев, Л.В. Борель, Н.В. Ведерников, А.С. Забашта, К.Д. Кириченко, М.В. Лукина, Е.В. Милованович, Р.А. Попков, Н.А. Серебрянская, В.Ю. Тертычный, Т.С. Юрова

Б79 Всероссийская междисциплинарная олимпиада школьников 8-11 класса «Национальная технологическая олимпиада». Учебно-методическое пособие
Том 6 **Большие данные и машинное обучение**
—М.: ООО «ВАШ ФОРМАТ», 2024. —130 с.

ISBN 978-5-00147-596-5

Данное пособие разработано коллективом авторов на основе опыта проведения всероссийской междисциплинарной олимпиады школьников 8-11 класса «Национальная технологическая олимпиада» в 2023/24 учебном году, а также многолетнего опыта проведения инженерных соревнований для школьников. В пособии собраны основные материалы, необходимые как для подготовки к олимпиаде так и для углубления знаний и приобретения навыков решения инженерных задач.

В издании приведены варианты заданий по профилю Национальной технологической олимпиады за 2023/24 учебный год с ответами, подробными решениями и комментариями. Пособие адресовано учащимся 8–11 классов, абитуриентам, школьным учителям, наставникам и преподавателям учреждений дополнительного образования, центров молодежного и инновационного творчества и детских технопарков.

Методические материалы также могут быть полезны студентам и преподавателям направлений, относящихся к группам:

01.00.00 Математика и механика

02.00.00 Компьютерные и информационные науки

09.00.00 Информатика и вычислительная техника

10.00.00 Информационная безопасность

11.00.00 Электроника, радиотехника и системы связи

12.00.00 Фотоника, приборостроение, оптические и биотехнические системы и технологии

13.00.00 Электро- и теплоэнергетика

15.00.00 Машиностроение

24.00.00 Авиационная и ракетно-космическая техника

27.00.00 Управление в технических системах

45.00.00 Языкознание и литературоведение

ISBN 978-5-00147-596-5

УДК 373.5.016:004.65

ББК 74.263.02



9 785001 475965 >

Оглавление

1 Введение	5
2 Большие данные и машинное обучение	17
I Работа наставника НТО на первом отборочном этапе	20
II Первый отборочный этап	21
II.1 Предметный тур. Информатика и программирование	21
II.1.1 Первая волна. Задачи 8–11 класса	21
II.1.2 Вторая волна. Задачи 8–11 класса	32
II.1.3 Третья волна. Задачи 8–11 класса	42
II.2 Предметный тур. Математика	52
II.2.1 Первая волна. Задачи 8–9 класса	52
II.2.2 Первая волна. Задачи 10–11 класса	56
II.2.3 Вторая волна. Задачи 8–9 класса	63
II.2.4 Вторая волна. Задачи 10–11 класса	68
II.2.5 Третья волна. Задачи 8–9 класса	74
II.2.6 Третья волна. Задачи 10–11 класса	79
II.3 Инженерный тур	85
III Работа наставника НТО на втором отборочном этапе	88
IV Второй отборочный этап	89
IV.1 Алгоритмические задачи	89
IV.2 Задачи по машинному обучению	96

V	Работа наставника НТО при подготовке к заключительному этапу	100
VI	Заключительный этап	101
VI.1	Предметный тур	101
VI.1.1	Информатика и программирование. 8–11 классы	101
VI.1.2	Математика. 8–9 классы	111
VI.1.3	Математика. 10–11 классы	115
VI.2	Инженерный тур	121
VI.2.1	Общая информация	121
VI.2.2	Легенда задачи	121
VI.2.3	Требования к команде и компетенциям участников	121
VI.2.4	Оборудование и программное обеспечение	121
VI.2.5	Описание задачи	121
VI.2.6	Система оценивания	122
VI.2.7	Решение задачи	123
VI.2.8	Материалы и курсы для подготовки	126
VII	Критерии определения победителей и призеров	127
VIII	Работа наставника после НТО	129

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 сообществе 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. Что бы 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 делает 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
- Летящая 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/bd1v/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.
- Летящая робототехника.



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



Большие данные и машинное обучение

Логика выстраивания олимпиадных заданий в рамках Национальной технологической олимпиады по профилю «Большие данные и машинное обучение» является ключевым элементом, определяющим качество и сложность задач, предлагаемых участникам на этапах соревнований. Цикл заданий направлен на развитие навыков в области анализа данных, работы с большими объемами информации и применения методов машинного обучения для решения разнообразных задач. Направление «Большие данные и машинное обучение» фокусируется на решении задач, связанных с анализом данных. В современном мире, благодаря социально-экономическим и технологическим достижениям, мы ежедневно получаем огромное количество информации о деятельности человека. Эти данные можно обрабатывать, анализировать и упорядочивать.

Основная цель профиля — не только проверить знания участников, но и развить их умение логически мыслить, принимать решения на основе данных, анализировать информацию и применять полученные знания на практике. Задания олимпиады часто строятся таким образом, чтобы проверить не только теоретическое понимание материала, но и способность участников к применению полученных знаний в реальных ситуациях.

Знакомство с профилем начинается с «Урока НТО» по профилю «Большие данные и машинное обучение», который проводится в общеобразовательных учреждениях. Материалы для проведения урока находятся на сайте <https://nti-lesson.ru/> и доступны после регистрации на платформе «Талант». Урок погружает участников в выполнение реальных задач, связанных с анализом больших объемов данных, а также знакомит учащихся с такими понятиями как большие данные, машинное обучение и предлагает решить сложную задачу художественного переноса стиля на языке программирования Python с использованием инструмента визуализации Jupyter Notebook.

Для подготовки участникам предоставлены материалы по программированию на Python и использованию основных библиотек для анализа данных, основам машинного обучения, теории вероятностей, практикумы и сборники задач с предыдущих соревнований. Материалы есть на странице профиля и на сайте проекта Академия искусственного интеллекта для школьников.

Ключевые аспекты логики выстраивания заданий включают в себя постепенное усложнение задач, от базового уровня к более сложным, поэтапное развитие навыков участников и предоставление возможности проявить свои умения в различных областях анализа данных и машинного обучения. Кроме того, задания могут быть построены таким образом, чтобы требовать нестандартного подхода к их решению, стимулируя творческое мышление и инновационные подходы.

В рамках первого отборочного этапа участникам необходимо решить задачи по математике и информатике на предметном туре, освоить теорию машинного обучения через образовательный блок и развить свои компетенции в анализе данных на инженерном туре.

Задачи второго этапа готовят участников к задачам заключительного этапа. Участники знакомятся с платформой all cups, на которой размещено задания, которые представляют из себя задачи по олимпиадному программированию и задачи по машинному обучению.

На заключительном этапе участники соревнуются в построении прогнозной модели. Задачи заключительного этапа состоит в том, чтобы спрогнозировать, понравится ли предложенная видеозапись пользователю социальной сети «Одноклассники» по имеющимся данным о пользователях. Пользователи ежедневно просматривают ленту с записями и каким-либо образом взаимодействуют с записями. Для построения системы рекомендации требуется предсказать тип взаимодействия.

Участникам предоставляется доступ к виртуальному серверу для проведения вычислений. На сервере установлен Python 3.9 с Jupyter Notebook, SciPy, NumPy, Scikit Learn и Pandas. Участники могут доустановить необходимые им библиотеки и языки.

Также участникам предоставляется набор данных, подготовленный партнером профиля — компанией VK. Участникам выдается набор данных, описание данных и задание, которое включает описание проверочных данных, критерий качества решения и описание базового решения.

Участникам требуется по выданному набору данных аппроксимировать скрытую зависимость в нем и построить предсказание целевой метки.

Участникам требуется загрузить полученное решение в виде текстового файла с ответами в тестирующую систему, которая оценит его согласно используемому критерию качества. Во время работы над задачей участникам доступен только результат оценивания их решений на небольшом подмножестве проверочных данных.

Для решения задачи требуется написать код для автоматической обработки данных. Код требуется загрузить вместе с соответствующим текстовым решением.

Для подготовки к заключительному этапу участникам предоставляется подборка материалов по машинному обучению и построению алгоритмов рекомендательных систем, на которые нацелена заключительная задача. Для организации эффективной подготовки к олимпиаде профиль проводит хакатон по тематике профиля и разбор заданий второго этапа.

Компетенции, приобретаемые благодаря участию в профиле олимпиады, помогают участникам развить как hard skills: навыки программирования, умение строить алгоритмы и практические знания методов статистического анализа больших данных. Знание как минимум языков программирования Python или R, основных библиотек, алгоритмов, их ограничений, базовые компетенции в теории вероятностей, статистике, математическом анализе и линейной алгебре, так и soft skills: умение работать в команде, эмоциональный интеллект, самоорганизация, тайм менеджмент, проявление лидерских качеств, принятие решений, самостоятельная работа с учебными материалами.

Все эти навыки и компетенции в равной степени помогают участнику на пути к становлению гармоничной и развитой личности.

Победители и призеры профиля «Большие данные и машинное обучение» поступают в ведущие вузы России на специальности, связанные с информационными технологиями и принимают участие в научно-технологических проектных школах. Участие школьников в данном профиле позволяет повысить популярность и осознанность выбора профессии в области IT-технологий. Собственный реальный опыт

в этой области дает возможность уже в школьном возрасте понять свое отношение и выбрать целевой профильный вуз, а значит определить эффективную образовательную траекторию.

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 II.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 $a \leq b$ и $u \leq v$.

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.

```

1 a, b = sorted(list(map(int, input().split())))
2 u, v = sorted(list(map(int, input().split())))
3 if a<u and b<v:
4     print(0)
5 elif a/2<u and b<v or a<u and b/2<v or b/2<u and a<v:
6     print(1)
7 else:
8     print(-1)

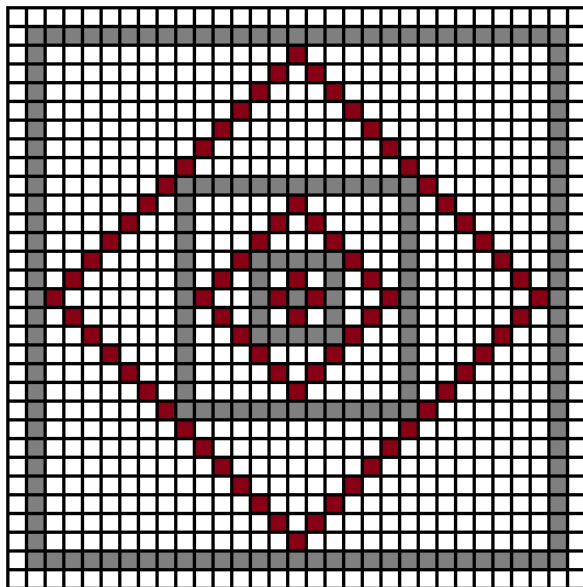
```

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, $1 \leq n \leq 100$.

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.

```

1  n = int(input())
2  ans = 1
3  ln = 1
4  for i in range(n):
5      ans += (ln - 1) * 4
6      if i%2==0:
7          ln += 1
8      else:
9          ln = 2 * ln + 1
10 print(ans)

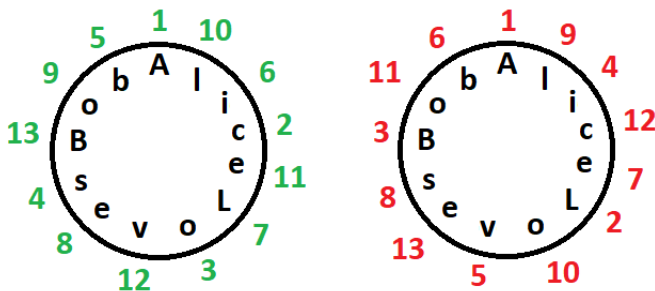
```

Задача II.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 s , 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. Напишите 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 $3i \bmod n$, 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.

```

1 s = input()
2 n = len(s)
3 tmp = [''] * n
4 ans = ''
5 for i in range(n):
6     tmp[(i * 3) % n] = s[i]
7 for i in range(n):
8     ans += tmp[(i * 5) % n]
9 print(ans)

```

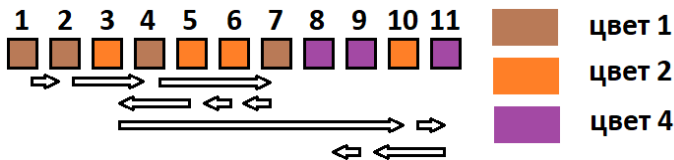
Задача П.1.1.4. Покраска кубиков (30 баллов)

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 меняет 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, $1 \leq n, k \leq 100000$. Во второй строке на вход подается n натуральных чисел c_1, c_2, \dots, c_n , где c_i — требуемый цвет i -го кубика, $1 \leq c_i \leq k$.

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
$n = k; n \leq 1000$; все c_i различны.	1–8
$n = k; n \leq 100000$; все c_i различны.	9–20
$k = 3; n \leq 1000$	21–24
$k = 4; n \leq 1000$	25–30
$k = 5; n \leq 1000$	31–40
$n \leq 1000$	41–50
$n \leq 100000$	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.

```

1 n, k = map(int, input().split())
2 left = [n + 1] * (k + 1)
3 right = [0] * (k + 1)
4 i = 1
5 for col in map(int, input().split()):
6     left[col] = min(left[col], i)
7     right[col] = max(right[col], i)
8     i += 1
9 l, r, fl, fr = 1, 1, 0, 0
10 for i in range(1, k + 1):
11     if right[i] > 0:
12         dist = right[i] - left[i]
13         tl = dist + min(fl + abs(l - right[i]),
14                       fr + abs(r - right[i]))
15         tr = dist + min(fl + abs(l - left[i]),
16                       fr + abs(r - left[i]))
17         l, r, fl, fr = left[i], right[i], tl, tr
18 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, \dots, 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, $1 \leq n \leq 200000$, $1 \leq s \leq 10^9$. Во второй строке записаны целые неотрицательные числа t_1, t_2, \dots, t_n , задающие моменты времени поступления запросов, $0 \leq t_1 \leq t_2 \leq \dots \leq t_n \leq 10^9$.

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
$n \leq 1000$; для всех t_i выполняется одно из двух условий: либо $t_i = t_{i+1}$, либо $t_i + s \leq t_{i+1}$.	1–5
$n \leq 1000$	6–15
$n \leq 200000$	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 $i \leq n - k$ выполняется неравенство $t_{i+k} - t_i \leq s$. 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.

```

1 n, s = map(int, input().split())
2 t = list(map(int, input().split()))
3 ans = 1
4 left = 0
5 right = 0
6 while right < n:
7     if t[right] - t[left] >= s:
8         left += 1
9     else:
10        right += 1
11    ans = max(ans, right - left)
12 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, $1 \leq a, b, c \leq 1000$.

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.

```

1 a, b, c = map(int, input().split())
2 if a % 2 == 0:
3     print(a // 2, b // 2, (c + 1) // 2)
4 else:
5     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, \dots, 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, $1 \leq n \leq 1000$.

Во второй строке записана последовательность натуральных чисел a_1, a_2, \dots, 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.

```

1 s = input()
2 p = list(map(int, input().split()))
3 for a in 'rgb':
4     for i in range(len(s)):
5         if s[i] == a:
6             print(p[i], end = ' ')

```

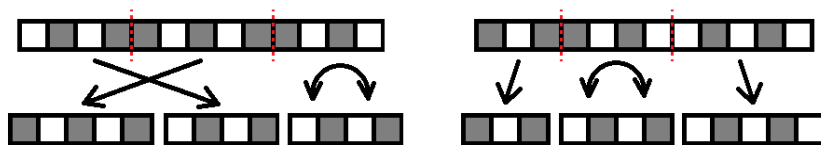
Задача II.1.2.3. Черно-белая полоска (15 баллов)

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
wbwbwbwbwbwbw
Стандартный вывод
5 9 0
1 4 0
10 13 180

Пример №2

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

Пример №3

Стандартный ввод
bbwbw
Стандартный вывод
no

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 s = input()
2 n = len(s)
3 x = []
4 for i in range(1, n):
5     if s[i] == s[i-1]:
6         x.append(i)
7 if s[x[0]] != s[x[1]]:
8     print(1, x[0], 0)
9     print(x[0] + 1, x[1], 180)
10    print(x[1] + 1, n, 0)
11 elif s[x[0]] == s[0] or s[x[0]] == s[-1]:
12    print('no')
13 else:
14    print(x[0] + 1, x[1], 0)
15    print(1, x[0], 0)
16    print(x[1] + 1, n, 180)

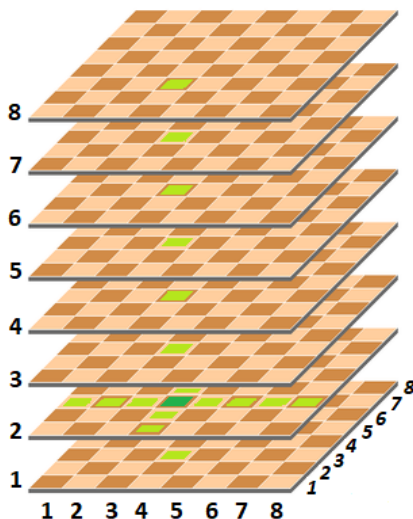
```

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 переход 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, $1 \leq n \leq 30$.

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 .

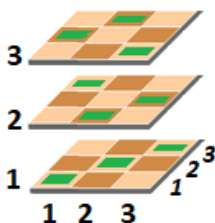
Примеры

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.

```

1 n = int(input())
2 for i in range(n):
3     for j in range(n):
4         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, состоящее 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, $1 \leq n \leq 63$.

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.

```

1 Names = ['Anna', 'Boris', 'Cary', 'David', 'Eva', 'Fiona']
2 n = int(input())
3 for i in range(6):
4     if n % 2 == 1:
5         print(Names[i])
6     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, $1 \leq n \leq 100$. 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.

```

1 n = int(input())
2 ans = 0
3 for i in range(n):
4     s = input()
5     for j in range(1, len(s)):
6         if ord(s[j]) - ord(s[j - 1]) != 1:
7             break

```

```
8     else:
9         ans += len(s) ** 2
10    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 позаботиться 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 называть 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, третьей — 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, $1 \leq n \leq 1000$. 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.

```
1 n = int(input())
2 x = list(map(int, input().split()))
3 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 в 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:

- $x_i \geq a_i$;
- $y_i \geq b_i$;
- $x_i + y_i \leq c_i$;
- $\sum_{i=1}^n x_i = m$;
- $\sum_{i=1}^n y_i = k$;
- $\sum_{i=1}^n |x_i - y_i| \rightarrow \min$.

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; $1 \leq n \leq 100$; $0 \leq m, k \leq 10000$. Во второй строке через пробел записаны числа a_1, a_2, \dots, a_n , задающие количество активных детей, изначально находящихся в каждом из автобусов. В третьей строке аналогично записаны числа b_1, b_2, \dots, b_n , задающие количество спокойных детей, изначально находящихся в каждом из автобусов. Наконец, в четвертой строке записаны натуральные четные числа c_1, c_2, \dots, c_n , задающие вместимость каждого из автобусов; $2 \leq c_i \leq 100$. 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 $x_i \geq a_i$ и $y_i \geq b_i$. 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.

```
1 n, m, k = map(int, input().split())
2 a = list(map(int, input().split()))
3 b = list(map(int, input().split()))
4 c = list(map(int, input().split()))
5 m -= sum(a)
6 k -= sum(b)
7 for i in range(n):
8     if a[i] > b[i]:
9         v = min(k, a[i] - b[i], c[i] - a[i] - b[i])
10        b[i] += v
11        k -= v
12    else:
13        v = min(m, b[i] - a[i], c[i] - a[i] - b[i])
14        a[i] += v
15        m -= v
16 for i in range(n):
17    v = min(m, k, (c[i] - a[i] - b[i]) // 2)
18    a[i] += v
19    b[i] += v
20    m -= v
21    k -= v
22 for i in range(n):
23    v = min(m, c[i] - a[i] - b[i])
24    a[i] += v
25    m -= v
26    v = min(k, c[i] - a[i] - b[i])
27    b[i] += v
28    k -= v
29 print(*a)
30 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; $1 \leq n \leq 100000$; $1 \leq m \leq 100000$; $0 \leq k \leq 100000$.

Во второй строке через пробел записаны k натуральных чисел d_1, d_2, \dots, d_k — номера дней, в которые произойдут события с известной датой; $1 \leq d_i \leq n$. 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, \dots, 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.

```

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

```

```
7     ans.extend([n] * max(m - len(ans), 0))
8 else:
9     p = 0
10    segs = []
11    for i in range(1, n + 1):
12        if d[i]:
13            if p == 0:
14                plen = i - 1
15            elif i - p > 2:
16                segs.append((p + 2, i))
17            p = i
18    segs.sort(key = lambda x: x[1] - x[0] + ((x[1] - x[0]) % 2) * 100000)
19    ans = []
20    for (a, b) in segs:
21        ans.extend(range(a, b, 2))
22    if n - p > 1:
23        ans.extend(range(p + 2, n + 1, 2))
24    if plen > 1:
25        ans.extend(range(plen - 1, 0, -2))
26    if (n - p) % 2 == 1:
27        ans.append(n)
28    ans = ans[: min(m, len(ans))]
29    ans.extend([1] * (m - len(ans)))
30    for i in ans:
31        d[i] = True
32    s = 0
33    for i in range(1, n + 1):
34        if d[i] or d[i - 1] and d[i + 1]:
35            s += 1
36    print(s)
37    print(*ans)
```

Предметный тур. Математика

Первая волна. Задачи 8–9 класса

Задача П.2.1.1. (15 баллов)

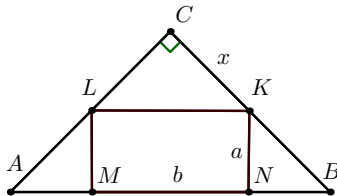
Темы: планиметрия.

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 $\angle AML = 90^\circ$. Следовательно, треугольник AML — прямоугольный равнобедренный. Пусть $AM = ML = a$. Тогда

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

Отсюда $a = \sqrt{2S_1}$.

Так как $\angle CLK = 180^\circ - \angle MLA - \angle MLK = 180^\circ - 45^\circ - 90^\circ = 45^\circ$, 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

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

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

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

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

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

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

Задача II.2.1.3. (20 баллов)

Темы: теория множеств, логика.

Условие

Чтобы обсудить последние новости, n друзей 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 друзей вкусы совпали хотя бы для одного бокала, а самый популярный вкус выбрали ровно k друзей. Определите наименьшее возможное значение k .

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

Решение

Оценка. Докажем, что самый популярный вкус выбрали не менее $\lceil \frac{2}{3}n \rceil$ друзей (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 друг 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 друг выбрал хотя бы один из оставшихся вкусов. Пусть, например, он выбрал B . Если первый друг заказал только один бокал лимонада, то по условию все остальные друзья тоже выбрали бокал лимонада вкуса B . Но тогда получается, что вкус B популярнее вкуса A . Это противоречие показывает, что первый друг заказал лимонады обоих вкусов, B и C .

Поскольку у первого друга 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

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

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

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

Ответ: $\frac{300}{(n-1)(n-3)}$.

Задача II.2.1.5. (25 баллов)

Темы: алгебра, неравенства.

Условие

При каком наибольшем значении параметра a неравенство:

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

верно для всех вещественных значений x и y ?

Решение

Выделим полные квадраты:

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

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

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

Имеем $a = 23\beta$.

Погрешность 0.

Варианты

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

Ответ: 23β .

Первая волна. Задачи 10–11 класса**Задача П.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

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

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 — $x(t - \frac{t}{k-1})$, третий — $x(t - \frac{2t}{k-1})$ и т. д. Поэтому

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

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

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

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

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

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

Используя формулу для суммы арифметической прогрессии, получаем

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

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

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

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

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

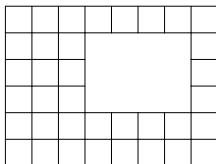
Ответ: $1 + \frac{p(n-1)}{100(2n-1)}$.

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

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

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

Ответ: $kn - lm + 1$.

Задача II.2.2.4. (20 баллов)

Темы: геометрическая вероятность, выпуклый четырехугольник.

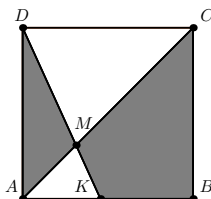
Условие

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 $\frac{a\sqrt{2}}{2} = a$. 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

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

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

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

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

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

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

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)

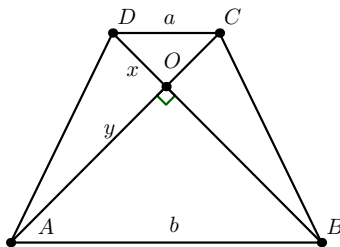
Темы: планиметрия, трапеция.

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,

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

Переносим 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

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

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

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

Погрешность 0.

Варианты

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

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

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

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

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

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

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

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

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 разбилась 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 $V - P + G = 2$, где V — число вершин, P — число ребер, G — число граней (грань — участок пастбища либо внешняя территория).

Число вершин известно: $V = n + 5$. Число участков, на которые разбито пастбище, равно $G - 1$.

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 $5G = 2P$ записей. Поэтому $P = 5G/2$.

Возвращаясь к формуле Эйлера, находим

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

Отсюда

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

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

Погрешность 0.

Варианты

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

Ответ: $\frac{2(n+3)}{3} - 1$.

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

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

Ответ: $\frac{N!C_{N-1}^{k-1}}{k!}$.

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

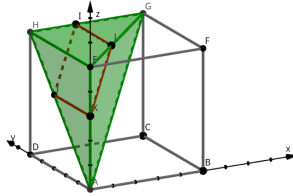
Выбирая три числа из $[0, a]$, назовем x — наименьшее из них, z — наибольшее, а y — лежащее между x и z . Выбор чисел x, y, z равносильно выбору точки $\alpha(x, y, z)$ в пространстве, удовлетворяющей условиям:

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

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

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 $KLEI$

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

Тогда

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

Значит, искомая вероятность

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

Для получения ответа в процентах умножим полученное выражение на 100.

Погрешность 0,01.

Варианты

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

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

Задача II.2.4.5. (25 баллов)

Темы: алгебра, неравенства, экстремальные значения.

Условие

Найдите наибольшее значение выражения $y + bx$ при условии

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

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

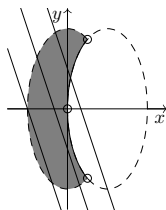
$$\log_{x^2 + \frac{y^2}{a^2}} 2x \geq 1, \quad (\text{II.2.1})$$

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

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

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.

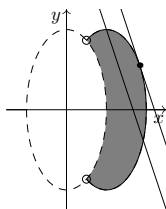


Рассматривая всевозможные прямые $y = m - bx$, проходящие через данную область (некоторые из этих прямых изображены на рисунке), видим, что наибольшего значения m не существует (можно сколь угодно близко приближать прямую к границе области).

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

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

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. Найдём 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 интересное значение $m = b + \sqrt{a^2 + b^2}$.

Погрешность 0,01.

Варианты

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

Ответ: $b + \sqrt{a^2 + b^2}$.

Третья волна. Задачи 8–9 класса

Задача П.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

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

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

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

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

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

Погрешность 0.

Варианты

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

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

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

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

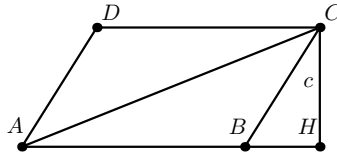
Ответ: $100(1 - (1 - \frac{P}{100})^a (1 - \frac{Q}{100})^b - a \frac{P}{100} (1 - \frac{P}{100})^{a-1} (1 - \frac{Q}{100})^b - (1 - \frac{P}{100})^a b \frac{Q}{100} (1 - \frac{Q}{100})^{b-1})$.

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

$$h = \frac{S}{AC}. \quad (\text{II.2.2})$$

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

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

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

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 $N \geq 280$.

Решение

По условию задачи составим систему уравнений

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

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 $N \geq 280$, то наименьшее значение $N = 280 + 35q + 5r$ получается при $y = 1$.

Погрешность 0.

Варианты

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

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

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

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 преодолет количество минутных долей циферблата, равное

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

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 $n \geq 12 \cdot 60 = 720$. А так как $k < 100$, то

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

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

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

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

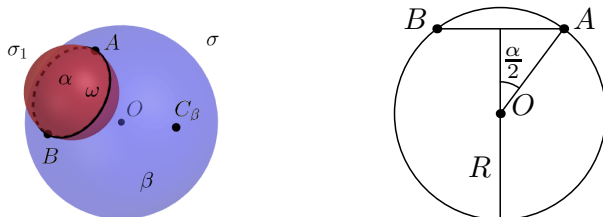
Точки A и B лежат на сфере с центром O так, что угол AOB равен α° . Случайно на сфере выбирается еще одна точка C (любой выбор равновозможен). Определите вероятность того, что угол ACB окажется острым.

Защитите 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

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

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 $2 \leq x \leq m-1$ (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 $2 \leq y \leq m-1$, то $y = 1 + \frac{m}{2}$.

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

Погрешность 0.

Варианты

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

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 $k \leq n/2$, 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

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

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

Инженерный тур

Задача П.3.1. Один из них (100 баллов)

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 256 Мбайт.

Условие

Александр работает на таможне в космопорте. Однажды туда прибыл космический корабль после экспедиции на Марс. Сканер в порту забил тревогу, на корабле находится ровно один марсианин, который замаскировался под члена экипажа. К сожалению, сканер не способен показать, кто конкретно является марсианином.

Тогда Александр решил запросить протокол общения всех членов экипажа. Известно, что в языке реальных членов экипажа вероятность появления каждого символа зависит от предыдущего. Однако марсианин этого не знал, и пытался пародировать язык реальных членов экипажа, выбирая буквы независимо от предыдущих.

Помогите Александру определить, кто из членов экипажа на самом деле является марсианином. Гарантируется, что с имеющимся протоколом общения это реально сделать.

Формат входных данных

Первая строка содержит натуральное число N ($3 \leq N \leq 10$) — число членов экипажа на корабле (включая замаскированного марсианина).

Следующие N строк содержат текст высказывания соответствующих членов экипажа.

Каждая строка состоит только из маленьких латинских букв и пробелов. Сумма длин всех строк не превышает 10^4 .

Формат выходных данных

Выведите одно натуральное число — номер строки, которая соответствует тексту высказывания замаскированного марсианина.

Примеры

Пример №1

Стандартный ввод
3 aba caba aba daba baa cdaa
Стандартный вывод
3

Решение

Построим матрицу p вероятности встречи символа y после символа x . Сперва эта матрица будет состоять из нулей. Затем переберем тексты высказываний всех членов экипажа, и для каждого текста переберем все пары соседних символов в нем. Для каждой пары добавим в соответствующую ячейку матрицы единицу, деленную на длину соответствующего текста. Делить на длину необходимо, чтобы все тексты вносили одинаковый вклад. Иначе замаскированного марсианина нельзя было бы обнаружить в случаях, когда длина его высказывания превышает сумму длин высказываний остальных членов экипажа. Наконец, элементы итоговой матрицы должны быть нормированы, чтобы она содержала корректные распределения вероятностей.

После построения матрицы p . Заново переберем все тексты, и для каждого текста вычислим вероятность его возникновения. Для этого вычислим сумму логарифмов вероятностей для каждой пары символов в тексте. Сумма логарифмов нужна, чтобы не вычислять произведения большого числа малых величин. Затем эту сумму логарифмов нужно поделить на длину текста. Иначе замаскированного марсианина нельзя было бы обнаружить в случаях, когда длина его высказывания сильно меньше высказываний остальных членов экипажа.

Ответом будет являться номер высказывания, которое получило наименьшую вероятность.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 import math
2
3 k = 128
4 n = int(input())
5 texts = []
6
7 for i in range(n):
8     texts.append(list(map(ord, input())))
9
10 p = [[0.0] * k for _ in range(k)]
11
12 for i in range(n):
13     text = texts[i]
```

```
14     m = len(text)
15
16     for j in range(1, m):
17         p[text[j - 1]][text[j]] += 1.0 / (m - 1)
18
19     for x in range(k):
20         r = p[x]
21         s = 1e-12 + sum(r)
22
23         for y in range(k):
24             r[y] /= s
25
26     log_prob = [0.0] * n
27     for i in range(n):
28         text = texts[i]
29         m = len(text)
30
31         for j in range(1, m):
32             log_prob[i] += math.log(p[text[j - 1]][text[j]]) / (m - 1)
33
34     ans = 0
35     for i in range(1, n):
36         if log_prob[i] < log_prob[ans]:
37             ans = i
38
39     print(ans + 1)
```

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 этапа направлены на тестирование знания алгоритмов и задач машинного обучения. Для работы с большими объемами данных требуется уметь реализовывать эффективные алгоритмы, а умение решать нестандартные задачи полезно для улучшения метрик машинного обучения. В финальной задаче участники работают с большим набором данных, а также решают задачу, рекомендации к которой можно свести к стандартной задаче классификации или регрессии.

Участники команды решают задачи по отдельности, либо объединяются, распределяя роли аналитика, генератора идей и программиста.

Алгоритмические задачи

Задача IV.1.1. k -ближайших рептилий (100 баллов)

Тема: умение реализовывать алгоритмы машинного обучения.

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 2 с.

Ограничение по памяти: 256 Мбайт.

Условие

Рептилии решили предпринять очередную попытку атаковать богатырей. Для этого они выстроились в одну длинную прямую шеренгу. У каждой рептилии есть характеристика — живучесть.

Богатыри решили спланировать оборону и оценить различные расстановки. Для этого им необходимо знать значение суммарной живучести у k -ближайших рептилий к некоторой точке. Все запросы независимы.

К сожалению, богатыри используют счет древних русов и им не просто произвести необходимые вычисления. Помогите им решить задачу.

Формат входных данных

Первая строка содержит одно целое положительное число N ($1 \leq N \leq 10^5$) — число рептилий в шеренге.

Следующие N строк содержат описание рептилий. Каждая из этих строк содержит два разделенных пробелом целых числа: X_i ($|X_i| \leq 10^9$) и P_i ($1 \leq P_i \leq 10^9$) — положение в шеренге и живучесть соответствующей рептилии. Гарантируется, что все X_i различны.

Далее следует строка с одним целым положительным числом M ($1 \leq M \leq \leq 2 \cdot 10^4$) — число запросов оценок расстановок.

Следующие M строк содержат описание запросов. Каждая из этих строк содержит два разделенных пробелом целых числа: X_q ($|X_q| \leq 10^9$) и K_q ($1 \leq K_q \leq N$) — положение в шеренге и число ближайших рептилий к нему.

Формат выходных данных

Для каждого запроса выведите одно целое число — суммарную живучесть k -ближайших рептилий. Если нельзя однозначно выбрать k -ближайших рептилий, то выведите 0.

Примеры

Пример №1

Стандартный ввод
5
1 8
5 1
3 4
7 2
9 8
4
2 1
6 2
5 3
8 4
Стандартный вывод
0
3
7
15

Решение

Для начала отсортируем рептилий (x_i, y_i) по возрастанию положения x_i и вычислим префиксные суммы живучести $s_i = \sum_{j=1}^i y_j = s_{i-1} + y_i$.

Теперь для каждого запроса (x_q, k_q) будем двоичным поиском искать наибольшую длину отрезка с центром в x_q , на котором находятся не более k_q рептилий. Для этого запустим еще два двоичных поиска. Они будут искать наименьший l и наибольший r индекс рептилий, которые все еще расположены на проверяемом отрезке.

Как только искомый отрезок будет найден, необходимо проверить, что на нем расположены ровно k_q рептилий. Для этого необходимо вычислить разницу между r и $l - 1$. Если эта разница равна k_q , то необходимо вывести разницу соответствующих префиксных сумм s_r и s_{l-1} , а иначе вывести ноль.

Полученное решение будет работать за

$$\mathcal{O}(N \log(N) + M \log(\max(x_i) - \min(x_i)) \log(N)).$$

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1  def findL(n, x, ql):
2      l, r = 0, n - 1
3      if ql <= x[l]:
4          return l
5      while r - l > 1:
6          m = (r + l) // 2
7          if x[m] < ql:
8              l = m
9          else:
10             r = m
11     return r
12
13  def findR(n, x, qr):
14     l, r = 0, n - 1
15     if x[r] <= qr:
16         return n
17     while r - l > 1:
18         m = (r + l) // 2
19         if x[m] <= qr:
20             l = m
21         else:
22             r = m
23     return r
24
25  def solve(n, x, s, k, q):
26     if q <= x[0]:
27         return s[k]
28     if q >= x[n - 1]:
29         return s[n] - s[n - k]
30     if n == k:
31         return s[n]
32     l, r = 0, 2_000_000_000
33     while r - l > 1:
34         m = (l + r) // 2
35         f = findL(n, x, q - m)
36         t = findR(n, x, q + m)
37         if t - f <= k:
38             l = m
39         else:
40             r = m
41     f = findL(n, x, q - 1)
42     t = findR(n, x, q + 1)
43     if t - f == k:
44         return s[t] - s[f]
45     else:
46         return 0
47
48  def prefSum(n, y):
49     s = [0] * (n + 1)
50     for i in range(n):
51         s[i + 1] = s[i] + y[i]
52     return s
53
54  def sort(n, x, y):
55     order = [i for i in range(n)]
56     cx, cy = x[:,], y[:,]

```

```

57     order.sort(key=lambda i: cx[i])
58     for i in range(n):
59         x[i] = cx[order[i]]
60         y[i] = cy[order[i]]
61
62     def main():
63         n = int(input())
64         x, y = [], []
65         for _ in range(n):
66             xi, yi = map(int, input().split())
67             x.append(xi)
68             y.append(yi)
69         sort(n, x, y)
70         s = prefSum(n, y)
71         m = int(input())
72         for _ in range(m):
73             q, k = map(int, input().split())
74             print(solve(n, x, s, k, q))
75
76     if __name__ == "__main__":
77         main()

```

Задача IV.1.2. Квадрат через квадрат (100 баллов)

Тема: умение реализовывать алгоритмы машинного обучения.

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 256 Мбайт.

Условие

Однажды Максим решил заняться супрематическим творчеством. Он захотел рисовать картины на уровне Малевича, а, может, даже выше. Поэтому Максим решил закрашивать картину не одним цветом, а ориентироваться на заполненную числами опорную квадратную матрицу.

Чтобы картины не казались случайно окрашенными, Максим придумал хитрый способ генерации опорной матрицы. Он брал две заполненные числами квадратные матрицы разных размеров: первая матрица всегда была больше второй. Затем Максим брал вторую матрицу и накладывал ее на первую всеми возможными способами так, чтобы вторая матрица всегда оставалась внутри первой. Каждый раз, когда ячейки матриц сопоставлялись, Максим выписывал сумму произведений сопоставленных ячеек в соответствующую сдвигу ячейку третьей матрицы. Эту третью матрицу он и использовал как опорную. Пример построения такой матрицы представлен на рисунке ниже.

1	2	3
4	5	6
7	8	9

Первая матрица

4	3
2	1

Вторая матрица

23	33
53	63

Третья матрица

<table border="1"> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>4</td><td>5</td><td>6</td></tr> <tr><td>7</td><td>8</td><td>9</td></tr> </table> $1 \cdot 4 + 2 \cdot 3 + 4 \cdot 2 + 5 \cdot 1 = 23$	1	2	3	4	5	6	7	8	9	<table border="1"> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>4</td><td>5</td><td>6</td></tr> <tr><td>7</td><td>8</td><td>9</td></tr> </table> $2 \cdot 4 + 3 \cdot 3 + 5 \cdot 2 + 6 \cdot 1 = 33$	1	2	3	4	5	6	7	8	9
1	2	3																	
4	5	6																	
7	8	9																	
1	2	3																	
4	5	6																	
7	8	9																	
<table border="1"> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>4</td><td>5</td><td>6</td></tr> <tr><td>7</td><td>8</td><td>9</td></tr> </table> $4 \cdot 4 + 5 \cdot 3 + 7 \cdot 2 + 8 \cdot 1 = 53$	1	2	3	4	5	6	7	8	9	<table border="1"> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>4</td><td>5</td><td>6</td></tr> <tr><td>7</td><td>8</td><td>9</td></tr> </table> $5 \cdot 4 + 6 \cdot 3 + 8 \cdot 2 + 9 \cdot 1 = 63$	1	2	3	4	5	6	7	8	9
1	2	3																	
4	5	6																	
7	8	9																	
1	2	3																	
4	5	6																	
7	8	9																	

Нарисовав несколько картин таким способом, Максим показал их своему другу, но тот усомнился, что цвета на картине расположены не случайно. Чтобы доказать обратное, Максим должен предъявить первую и вторую матрицу для каждой картины. Однако он смог найти только первую матрицу для каждой картины и восстановить третью по самой картине. Помогите Максиму по первой и третьей матрицам найти вторую.

Формат входных данных

Первая строка содержит разделенных пробелом два натуральных числа N и M ($1 < M < N \leq 10$) — размеры первой и третьей матриц соответственно.

Далее следует N строк с N разделенными пробелом целыми числами в каждой — значения ячеек первой матрицы. Все числа находятся в диапазоне $[0; 10]$.

Затем следует M строк с M разделенными пробелом целыми числами в каждой — значения ячеек третьей матрицы. Все числа находятся в диапазоне $[0; 255]$.

Формат выходных данных

Выведите значения ячеек второй матрицы. Числа не обязательно должны быть целыми. Если решений несколько, разрешается вывести любое.

Ответ засчитывается, если абсолютная погрешность полученной в результате третьей матрицы не превышает 10^{-6} .

Примеры

Пример №1

Стандартный ввод	
3	2
1	2 3
4	5 6
7	8 9
23	33
53	63
Стандартный вывод	
4.0	3.0
2.0	1.0

Решение

Заметим, что получение значения в каждой из ячеек третьей матрицы можно представить как скалярное произведение значений из первой матрицы на набор неизвестных коэффициентов из второй. Поэтому задачу можно свести к решению системы линейных уравнений. Например, для первого теста получится следующая система:

$$\begin{cases} 1 \cdot x_{1,1} + 2 \cdot x_{1,2} + 4 \cdot x_{2,1} + 5 \cdot x_{2,2} = 23, \\ 2 \cdot x_{1,1} + 3 \cdot x_{1,2} + 5 \cdot x_{2,1} + 6 \cdot x_{2,2} = 33, \\ 4 \cdot x_{1,1} + 5 \cdot x_{1,2} + 7 \cdot x_{2,1} + 8 \cdot x_{2,2} = 53, \\ 5 \cdot x_{1,1} + 6 \cdot x_{1,2} + 8 \cdot x_{2,1} + 9 \cdot x_{2,2} = 63. \end{cases}$$

В общем случае число уравнений в системе не обязательно будет равно числу неизвестных коэффициентов, поэтому для решения необходимо использовать псевдообратную матрицу, как это делается в задаче линейной регрессии. Также неизвестные коэффициенты можно найти при помощи градиентного спуска.

Пример программы-решения

Пример №1

Ниже представлено решение на языке Python 3.

```

1 n, m = map(int, input().split())
2 a = []
3 b = []
4 for i in range(n):
5     a.append(list(map(int, input().split())))
6 for i in range(m):
7     b.extend(list(map(int, input().split())))
8
9 k= n-m+1
10 b = 0

```

```

11 coef = []
12 for i in range(k-1, n):
13     for j in range(k-1, n):
14         coef.append([])
15         for u in range(k-1,-1,-1):
16             for w in range(k-1,-1,-1):
17                 coef[-1].append(a[i-u][j-w])
18
19 import numpy as np
20
21 coef = np.array(coef)
22 b = np.array(b)
23
24 result = np.linalg.pinv(coef.astype(float)) @ b
25 for i in range(k,len(result)+1, k):
26     print(*result[i-k:i])

```

Пример №2

Ниже представлено решение на языке Python 3.

```

1 def solve(n, m, k, a, b, lr=0.005):
2     lr /= (n * m)
3     c = [[0.0 for _ in range(k)] for _ in range(k)]
4     tau = 1e-9
5     dc = [[0.0 for _ in range(k)] for _ in range(k)]
6
7     for _ in range(10000):
8         for x in range(k):
9             for y in range(k):
10                dc[x][y] = tau * c[x][y]
11
12        for i in range(m):
13            for j in range(m):
14                res = 0
15                for x in range(k):
16                    for y in range(k):
17                        res += a[i + x][j + y] * c[x][y]
18                diff = res - b[i][j]
19                for x in range(k):
20                    for y in range(k):
21                        dc[x][y] += diff * a[i + x][j + y]
22
23        for i in range(k):
24            for j in range(k):
25                c[i][j] -= (lr * dc[i][j])
26
27    return c
28
29 n = int(input())
30 m = int(input())
31 a = []
32 b = []
33
34 for i in range(n):
35     a.append(list(map(float, input().split())))
36
37 for i in range(m):
38     b.append(list(map(float, input().split())))
39

```

```

40 c = solve(n, m, n - m + 1, a, b)
41
42 for row in c:
43     for val in row:
44         print(val, end=' ')
45     print()

```

Задачи по машинному обучению

Задача IV.2.1. Стас вычисляет цены (100 баллов)

Темы: умение решать задачи машинного обучения.

Условие

Однажды Стасу на склад магазина поступила партия товаров. А вместе с ней информация о характеристиках и ценах. Стасу поручили внести ее в базу. Что-то пошло не так, и часть информации потерялась. Он пришел в недоумение от этого, но решил восстановить потерянную информацию. Он смог восстановить сведения о характеристиках, так как соответствующие товары были на складе. Помогите теперь Стасу восстановить цены хотя бы у части товаров.

Файл `train.csv` содержит описание товаров с известными ценами:

- `id` — артикул (идентификатор) товара,
- `f1 ... f40` — числовые характеристики товара,
- `target` — цена товара.

Файл `test.csv` содержит описание товаров в том же формате, но с неизвестными ценами.

Вы должны загрузить в тестирующую систему `csv`-файл со столбцами `id` и `target`, в котором будут предсказания цен для подмножества товаров из файла `test.csv`, допускается от 30000 до 50000 объектов. Все цены должны быть неотрицательны.

Помимо файла с предсказанием вы должны загрузить архив с кодом программы, которая это предсказание построило, а также необходимыми инструкциями по запуску.

Критерии оценивания

Решение будет оцениваться функцией качества:

$$(1 - SMAPE) \cdot 100,$$

где $SMAPE$ — это средняя величина $abs(y_{true} - y_{pred}) / (y_{true} + y_{pred})$, вычисленная на присланном подмножестве объектов.

Обратите внимание, что в ходе соревнования вы будете видеть оценку только на 20% объектах, а после на оставшихся 80%, что и будет финальной оценкой решения. Также учитываться будет только оценка выбранного решения.

Баллы, которые вы получите в результате, будут вычислены по формуле:

$$\max\left(0; 100 \frac{S - B}{M - B}\right),$$

где S — качество решения задачи командой,

B — качество базового решения,

M — качество наилучшего среди всех команд решения.

Решение

Точного решения у задачи нет. Помимо стандартного подхода к работе с данными и решения задачи регрессии предполагалось, что участники воспользуются тем, что предсказание можно строить не для всех объектов из тестового множества. Для этого требуется дополнительно решать задачу по оценке уверенности или правильности ответа обученной модели. Такую оценку можно получить, если для предсказания обучить ансамбль из нескольких моделей и измерять, насколько их предсказания отличаются между собой.

Базовое решение

Ниже представлено решение на языке Python 3.

```

1 import numpy as np
2 import pandas as pd
3 from sklearn import tree
4
5 train = pd.read_csv("train.csv")
6
7 test = pd.read_csv("test.csv")
8
9 X_train = train.drop(columns=['target', 'id'], axis=1).to_numpy()
10 y_train = train['target']
11 X_test = test.drop(columns=['id'], axis=1).to_numpy()
12
13 dt = tree.DecisionTreeRegressor(max_depth=12)
14 dt.fit(X_train, y_train)
15
16 submission = pd.DataFrame(test['id'], columns=["id"])
17 submission['target'] = dt.predict(X_test)
18 submission = submission.sample(30000)
19
20 submission.to_csv("submission.csv", index=False)

```

Задача IV.2.2. Дизайнерская раскраска (100 баллов)

Тема: умение решать задачи машинного обучения.

Условие

Дед Мороз решил сделать заказ на изготовление игрушек к Новому году. Для этого ему необходимо выбрать цвет каждой игрушки. Он попросил своего знакомого

дизайнера Артема сделать это. Артем согласился помочь и приступил к подбору цветов для каждой игрушки. Перебрав сотню-другую игрушек, Артем устал и отказался от этой затеи.

Дед Мороз решил разобраться, по какому принципу были выбраны цвета для игрушек, но Артем не смог сформулировать свое дизайнерское чутье. Помогите Деду Морозу завершить список цветов игрушек.

Файл `data.csv` содержит описание игрушек:

- `id` — артикул (индекс) игрушки,
- `f1 ... f45` — числовые характеристики игрушки,
- `target` — цвет игрушки (символ «?» кодирует пропуск).

К сожалению, Дед Мороз тоже ударился в абстракцию, поэтому информации о том, что именно измеряют числовые характеристики, нет.

Вы должны загрузить в тестирующую систему `csv`-файл со столбцами `id` и `target`, в котором будут предсказания цветов для всех 150000 игрушек из файла `data.csv`.

Помимо файла с предсказанием вы должны загрузить архив с кодом программы, которая это предсказание построило, а также необходимыми инструкциями по запуску.

Критерии оценивания

Решение будет оцениваться функцией качества: точность.

Обратите внимание, что в ходе соревнования вы будете видеть оценку только на 20% объектах, а после на оставшихся 80%, что и будет финальной оценкой решения. Также учитываться будет только оценка выбранного решения.

Баллы, которые вы получите в результате, будут вычислены по формуле:

$$\max\left(0; 100 \frac{S - B}{M - B}\right),$$

где S — качество решения задачи командой,

B — качество базового решения,

M — качество наилучшего среди всех команд решения.

Решение

Точного решения у задачи нет. Помимо стандартного подхода к работе с данными и решения задачи классификации предполагалось, что участники воспользуются тем, что тестовые объекты перемешаны с тренировочными. Эти объекты можно было использовать для обучения модели извлечения признаков, а затем полученное множество признаков использовалось бы для построения модели классификации.

Базовое решение

Ниже представлено решение на языке Python 3.

```
1 import numpy as np
2 import pandas as pd
3 from sklearn import tree
4
5 df = pd.read_csv("data.csv")
6 df.fillna(0, inplace=True)
7
8 train = df[df['target'] != '?']
9 trainX = train.drop(['id', 'target'], axis=1).to_numpy()
10 trainY = train['target']
11
12 dt = tree.DecisionTreeClassifier()
13 dt.fit(trainX, trainY)
14
15 testX = df.drop(['id', 'target'], axis=1).to_numpy()
16 pred = dt.predict(testX)
17
18 sdf = df[['id', 'target']]
19 sdf.loc['target'] = pred
20
21 sdf.to_csv("submission.csv", index=False)
```

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 особое 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 волонтеры, 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/1M3eqWMBGXecMA>.

Задача VI.1.1.1. Площадь с осями, параллельными осям координат (15 баллов)

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 256 Мбайт.

Условие

У Алексы есть четыре палочки с положительными целыми длинами a_1 , a_2 , a_3 и a_4 ($a_1 \leq a_2 \leq a_3 \leq a_4$).

Она хочет разместить их на плоскости таким образом, чтобы каждая палочка была параллельна одной из двух координатных осей, и площадь, ограниченная этими палочками, была как можно больше.

Найдите эту максимальную ограниченную площадь.

Формат входных данных

На вход подаются четыре положительных целых числа a_1 , a_2 , a_3 и a_4 , каждое на отдельной строке, обозначающие длины палочек в неубывающем порядке ($1 \leq a_1 \leq a_2 \leq a_3 \leq a_4 \leq 100$).

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 .

Примеры*Пример №1*

Стандартный ввод
2
2
4
7
Стандартный вывод
8

Пример №2

Стандартный ввод
10
10
10
10
Стандартный вывод
100

Решение

Единственная область на плоскости, которую можно ограничить с помощью четырех отрезков, параллельных осям координат — это прямоугольник.

Если два отрезка образуют противоположные стороны прямоугольника, то реальной длиной стороны прямоугольника сможет стать только меньший из этих двух отрезков (а у большего останется лишняя длина).

Значит, нужно разбить заданные четыре отрезка на две пары, чтобы произведение минимумов в этих парах (которое и будет равно площади прямоугольника) было как можно больше.

Можно заметить, что всегда оптимально в одну пару отнести два наименьших отрезка (a_1 и a_2), а во вторую — два наибольших (a_3 и a_4). При этом минимумы в этих парах мы уже знаем: $a_1 \leq a_2$ и $a_3 \leq a_4$ по условию.

Следовательно, ответом на задачу является произведение $a_1 \cdot a_3$.

Пример программы-решения

Ниже представлено решение на языке C++.

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  int main() {
6      vector<int> a(4);
7      for (int i = 0; i < 4; i++) {
```

```
8     cin >> a[i];
9     }
10    sort(a.begin(), a.end());
11    cout << a[0] * a[2] << '\n';
12    return 0;
13 }
```

Задача VI.1.1.2. Пропущенные гласные (18 баллов)

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 256 Мбайт.

Условие

Существует множество способов записи слов на бумаге. Например, в некоторых письменностях, таких как арабская и иврит, большинство гласных опускается, хотя некоторые из них записываются.

В этой задаче мы будем рассматривать только строки, состоящие из английских букв и дефисов. Буквы *a*, *e*, *i*, *o*, *u*, и *y* считаются гласными, в то время как дефисы и все остальные буквы считаются согласными. Все сравнения нечувствительны к регистру: верхний и нижний регистр одной и той же буквы считаются равными.

Вам даны две строки *s* и *f*, называемые *коротким* именем и *полным* именем соответственно. Ваша задача — проверить, можно ли получить короткое имя *s* из полного имени *f*, опустив некоторые гласные (возможно, ни одной).

Формат входных данных

Первая строка содержит одну строку *s*, обозначающую короткое имя.

Вторая строка содержит одну строку *f*, обозначающую полное имя.

Каждая строка непуста и состоит максимум из 1000 английских букв и дефисов.

Формат выходных данных

Выведите **Same**, если короткое имя *s* можно получить из длинного имени *f*, опустив некоторые гласные, и **Different**, в противном случае.

Примеры

Пример №1

Стандартный ввод
Shrm-el-Shikh Sharm-el-Sheikh
Стандартный вывод
Same

Пример №2

Стандартный ввод
Eilot Eilat
Стандартный вывод
Different

Пример №3

Стандартный ввод
Saint-Petersburg Saint-Petersburg
Стандартный вывод
Same

Решение

Задача требует дополнить строку s гласными до строки f , если это возможно. Условие немного напоминает классические задачи на динамическое программирование, но в данной задаче достаточно жадного алгоритма. А именно, пройдем параллельно по строкам f и s , и по каждой паре текущих символов $\langle f_i, s_j \rangle$ выберем один из трех вариантов поведения (конец строки для простоты также принят за особый согласный символ):

1. сдвигаемся вперед в обеих строках, если $f_i = s_j$; если одновременно был достигнут конец обеих строк, то останавливаемся с ответом **Same**;
2. сдвигаемся вперед в строке f , если f_i — гласный и $f_i \neq s_j$;
3. останавливаемся с ответом **Different**, если f_i — согласный (включая символ конца строки) и $f_i \neq s_j$.

Почему жадный алгоритм работает? Примем инвариант: на каждом шаге сопоставления подстрока $s_{1\dots j}$ может быть дополнена до подстроки $f_{1\dots i}$. Параллельное сопоставление почти во всех случаях при продвижении по строке f предполагает единственный вариант продвижения по строке s , сохраняющий инвариант, за исключением случая совпадения гласных символов ($f_i = s_j$).

При $f_j = s_i$ возможно как (а) продвижение по строке s , так и (б) добавление копии f_i перед s_j , без продвижения по s . Однако вариант (б) в смысле сопоставления строк беднее: он накладывает дополнительное ограничение (необходимо найти

место для s_j в строке f после символа f_i), не давая возможностей (ведь вставить еще одну копию гласного s_j можно в любой момент). Поэтому вариант (а) всегда предпочтительнее.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 def isvowel(c):
2     return "AEIOUY".find(c) != -1
3
4 s = input().upper()
5 f = input().upper()
6 dp = [[0 for j in range(len(f) + 1)] for i in range(len(s) + 1)]
7 dp[0][0] = 1
8 for i in range(len(s) + 1):
9     for j in range(len(f) + 1):
10        if i < len(s) and j < len(f) and s[i] == f[j] and dp[i][j] == 1:
11            dp[i + 1][j + 1] = 1
12        if dp[i][j] == 1 and j < len(f) and isvowel(f[j]) and dp[i][j] == 1:
13            dp[i][j + 1] = 1
14 print("Same" if dp[len(s)][len(f)] > 0 else "Different")

```

Задача VI.1.1.3. Разбиение на простые слагаемые (20 баллов)

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 256 Мбайт.

Условие

Вы программист в крупной компании. И вам поручили написать функцию, которая считает количество разбиений числа на простые слагаемые.

Вам дается число n , и ваша задача — написать функцию, которая возвращает количество разбиений числа n . Простое число — это число, которое имеет ровно два делителя: 1 и само число.

Формат входных данных

Дано одно число n ($1 \leq n \leq 1000$).

Формат выходных данных

Вывести количество разбиений на простые слагаемые числа n .

*Примеры**Пример №1*

Стандартный ввод
2
Стандартный вывод
1

Пример №2

Стандартный ввод
6
Стандартный вывод
2

Пример №3

Стандартный ввод
8
Стандартный вывод
3

Решение

Для подсчета количества разбиений на простые слагаемые числа n с использованием динамического программирования можно воспользоваться следующим алгоритмом:

1. Создайте массив dp размером $(n + 1)$ и инициализируйте его нулями. Этот массив будет служить для хранения количества разбиений на простые слагаемые для каждого числа от 0 до n .
2. Инициализируйте $dp[0] = 1$, так как одна пустая последовательность является разбиением для любого числа.
3. Проходите по всем простым числам p от 2 до n :
 - для каждого числа i от p до n , увеличивайте $dp[i]$ на $dp[i - p]$. Это происходит потому, что разбиение числа i на простые слагаемые можно получить путем добавления p к каждому разбиению числа $i - p$.
4. В конце выполнения алгоритма значение $dp[n]$ будет содержать количество разбиений на простые слагаемые числа n .

Вот пример решения на языке Python, реализующий данный алгоритм.

```

1 def count_partitions(n):
2     dp = [0] * (n + 1)
3     dp[0] = 1
4
5     primes = [True] * (n + 1)

```

```

6     primes[0] = primes[1] = False
7
8     for p in range(2, n + 1):
9         if primes[p]:
10            for i in range(p, n + 1):
11                dp[i] += dp[i - p]
12            for i in range(2 * p, n + 1, p):
13                primes[i] = False
14
15     return dp[n]
16
17 print(count_partitions(int(input())))

```

Вы можете вызвать функцию `count_partitions(n)` для подсчета количества разбиений на простые слагаемые числа n и получить результат.

Альтернативным вариантом решения задачи является написание рекурсивного перебора с запоминанием состояния.

Задача VI.1.1.4. Болото (22 баллов)

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 256 Мбайт.

Условие

Вы находитесь на болоте размером n на m и окруженном непроходимым лесом. Вы стоите в точке $(0, 0)$ и хотите добраться до точки выхода (n, m) . На болоте есть k опасных трясин. Если вы попадете в трясину, то будете затянуты и не сможете выбраться. Каждая трясина представляет собой круг.

Ваша задача состоит в том, чтобы определить, сможете ли вы выбраться из болота или нет.

Формат входных данных

В первой строке даны три числа n, m, k ($10 \leq n, m \leq 10\,000, 1 \leq k \leq 1\,000$) — размеры болота и количество трясин.

Следующие k содержат по три целых числа x_i, y_i, r_i ($0 < x_i < n, 0 < y_i < m, 0 < r_i < 10\,000$) — координаты центра i -й трясины и радиус.

Формат выходных данных

Вывести сообщение YES, если сможете выбраться из болота, иначе вывести сообщение NO.

*Примеры**Пример №1*

Стандартный ввод
10 22 2 6 16 5 4 6 5
Стандартный вывод
YES

Пример №2

Стандартный ввод
10 10 2 5 4 4 3 7 4
Стандартный вывод
NO

Примечание: первая группа тестов состоит из тестов, для которых выполняется ограничение $k = 1$. Баллы за эту группу начисляются только при прохождении всех тестов группы. Стоимость группы составляет 4 балла.

Вторая группа тестов состоит из тестов, для которых выполняется ограничение $1 < k \leq 50$. Баллы за эту группу начисляются только при прохождении всех тестов группы. Стоимость группы составляет 12 баллов.

Третья группа тестов состоит из тестов, для которых выполняются полные ограничения. Баллы за эту группу начисляются только при прохождении всех тестов группы. Стоимость группы составляет 6 баллов.

Решение

Факт №1. Между двумя трясинами нельзя пройти, если расстояние между центрами меньше суммарной длины радиусов.

Факт №2. Между границей леса и трясинной нельзя пройти, если расстояние между ними меньше радиуса.

Давайте построим граф, где вершины — это трясина, добавим еще четыре вершины — это границы леса. Построим ребра между вершинами, если между ними нельзя пройти.

Тогда пройти из координаты $(0, 0)$ нельзя дойти до (n, m) , если существует путь из левой границы леса до нижней или правой, или существует путь из верхней границы до нижней или правой.

Существование такого пути можно проверить обычным dfs.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 import sys
2
3 sys.setrecursionlimit(10**9)
4
5 def dfs(v):
6     if flg[v]:
7         return
8     flg[v] = True
9     for j in range(1, k + 1):
10        r2 = (x[v] - x[j]) ** 2 + (y[v] - y[j]) ** 2
11        if r2 <= (d[j] + d[v]) ** 2:
12            dfs(j)
13
14 m, n, k = map(int, input().split())
15 x = [0] * 1001
16 y = [0] * 1001
17 d = [0] * 1001
18 flg = [False] * 1001
19
20 for i in range(1, k + 1):
21     x[i], y[i], d[i] = map(int, input().split())
22     flg[i] = False
23
24 for i in range(1, k + 1):
25     if y[i] <= d[i]:
26         dfs(i)
27
28 for i in range(1, k + 1):
29     if (m - x[i]) <= d[i]:
30         dfs(i)
31
32 for i in range(1, k + 1):
33     if (n - y[i]) <= d[i] and flg[i]:
34         print('NO')
35         exit(0)
36
37 for i in range(1, k + 1):
38     if x[i] <= d[i] and flg[i]:
39         print('NO')
40         exit(0)
41
42 print('YES')

```

Задача VI.1.1.5. Суперсчастливые числа (25 баллов)

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 10 с.

Ограничение по памяти: 1024 Мбайт.

Условие

В далекой стране существует особое племя, которое верит в суперсчастливые числа. Они считают, что если число удовлетворяет определенным условиям, то оно приносит удачу и счастье своему обладателю. Чтобы стать частью этого племени,

необходимо уметь находить *суперсчастливые числа*.

Назовем *суперсчастливым числом* длины $2n$, если сумма первых n цифр равна сумме последних n цифр, а также произведение цифр первой половины числа и произведение цифр второй половины числа совпадают и больше нуля.

Напишите программу, которая принимает на вход число n и выводит количество суперсчастливых чисел длины $2n$.

Формат входных данных

Дано одно целое число n ($1 \leq n \leq 27$).

Формат выходных данных

Вывести количество суперсчастливых чисел длины $2n$.

Примеры

Пример №1

Стандартный ввод
1
Стандартный вывод
9

Пример №2

Стандартный ввод
2
Стандартный вывод
153

Решение

Подсчитаем для половины длины n сколько пар с суммой s и p . Пусть пар (s, p) — cnt , тогда в ответ надо добавить cnt^2 (так как любая пара может быть в первой половине суперсчастливого числа, так и во второй).

Осталось научиться считать количество таких пар. Для этого воспользуемся динамическим программированием.

Пусть у нас для длины i подсчитана пара (s, p) . Тогда на $i+1$ мы можем поставить цифру d (от 1 до 9) и тогда для всех пар $(s+k, p+k)$ мы должны добавить ответ для пары (s, p) .

Стоит обратить внимание, что произведение и ответ может выйти за тип данных, и поэтому надо воспользоваться длинной арифметикой или написать решение на языках, в которых она встроена, например, на языке Python.

В зависимости от эффективности реализации можно набрать различное число баллов. Но все ответы можно подсчитать у себя на компьютере и отправить программу, которая будет выводить ответы с уже подсчитанными ответами.

Математика. 8–9 классы

Задача VI.1.2.1. (15 баллов)

Темы: текстовая задача, задача на движение.

Условие

Две блохи Бло и Хи решили устроить соревнования по бегу по кругу (бег в одну и ту же сторону). В начальный момент времени обе блохи располагались в точке старта. Известно, что Бло пробегает полный круг за 25 с, а Хи — за 30. Через какое количество с они в первый раз будут находиться на наибольшем расстоянии друг от друга?

Решение

Снова в точку старта обе блохи попадут через $\text{НОК}(25, 30) = 150$ с. За это время Бло пробежит 6 кругов, а Хи — 5 кругов. Рассмотрим систему отсчета, в которой Хи неподвижна. Тогда Бло пробежит 1 круг. Значит, половину круга Бло пробежит за $150/2 = 75$ с.

Ответ: 75 с.

Задача VI.1.2.2. (20 баллов)

Темы: графы.

Условие

Матрицей смежности графа называют таблицу, в которой записаны нули и единицы: на пересечении строки i и столбца j стоит единица тогда и только тогда, когда вершины графа i и j соединены дугой. Из трех представленных таблиц (матриц смежности) графов выберите все те таблицы, каждая из которых представляет граф, который можно нарисовать, не отрывая карандаш от бумаги. При рисовании можно в вершины графа входить несколько раз, а дуги повторять нельзя. Нарисуйте графы, соответствующие выбранным таблицам, и укажите порядок обхода вершин.

	1	2	3	4	5	6	7
1	0	1	0	0	0	0	0
2	1	0	1	1	1	0	0
3	0	1	0	0	1	0	0
4	0	1	0	0	1	1	0
5	0	1	1	1	0	0	1
6	0	0	0	1	0	0	1
7	0	0	0	0	1	1	0

Таблица 1

	1	2	3	4	5	6	7
1	0	0	1	0	0	0	0
2	0	0	1	1	1	0	0
3	1	1	0	0	1	0	0
4	0	1	0	0	1	1	1
5	0	1	1	1	0	0	1
6	0	0	0	1	0	0	1
7	0	0	0	1	1	1	0

Таблица 2

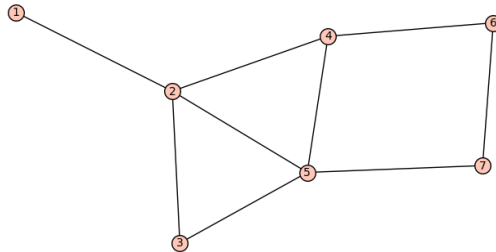
	1	2	3	4	5	6	7
1	0	0	1	0	0	0	0
2	0	0	1	1	1	0	0
3	1	1	0	0	1	0	0
4	0	1	0	0	1	1	0
5	0	1	1	1	0	0	1
6	0	0	0	1	0	0	1
7	0	0	0	0	1	1	0

Таблица 3

Решение

Граф может быть нарисован одним росчерком тогда и только тогда, когда он имеет эйлеров путь. В свою очередь, эйлеров путь, как известно, существует тогда и только тогда, когда число вершин с нечетной степенью равно либо нулю, либо двум. Таким образом, можно подсчитать в каждой строке количество единиц (это степени вершин), и если окажется, что в графе более двух нечетных вершин, значит, свойство «рисуемости одним росчерком» ему не присуще.

Таблицы 2 и 3 не подходят (вершины 1, 2 и 3 имеют нечетную степень). Таблица 1 подходит. Соответствующий рисунок.



Пример порядка обхода вершин для рисования одним росчерком:

1-2-3-5-2-4-5-7-6-4.

Ответ: таблица 1.

Задача VI.1.2.3. (20 баллов)

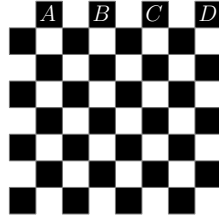
Темы: комбинаторика, правила сложения и умножения.

Условие

Сколько существует способов расставить на шахматной доске три шашки — одну белую дамку и две черных — так, чтобы белая шашка, находясь на крайней горизонтали, могла бить хотя бы одну из черных? Напомним, что шашки расставляются на черных клетках; белая дамка может бить черную шашку, если они находятся на

одной диагонали; белая дамка бьет черную пешку перепрыгивая через нее на одну из свободных клеток на той же диагонали (если таких свободных клеток нет, то бить нельзя), при этом нельзя перепрыгивать через две пешки, расположенные в соседних клетках.

Решение



Разобьем множество всех требуемых расстановок пешек на 4 случая: белая дамка стоит в позиции A , B , C или D (см. рисунок).

Эти случаи разбираются по одинаковой схеме. Рассматриваем два подслучая:

1. на общих диагоналях с дамкой стоит ровно одна черная пешка;
2. на этих диагоналях стоит две черных пешки.

В остальных случаях белая пешка не может бить ни одну из черных.

Случай $A.1$. Имеется 5 подходящих положений для черной пешки на общих с белой диагоналях и $32 - 8 = 24$ положений для другой черной пешки, то есть всего $24 \cdot 5 = 120$ способов.

Случай $A.2$. Всего $C_7^2 = 21$ способ расставить две черных пешки на общих диагоналях, но из них надо исключить 5 способов, когда черные пешки стоят рядом, и 1 случай, когда они стоят в крайних клетках. Итого $21 - 5 - 1 = 15$ способов.

В случаях B и C получаем такие же количества.

Случай $D.1$. Имеется 6 положений для черной пешки на одной с белой дамкой диагонали и $32 - 8 = 24$ положений для другой черной пешки. То есть всего $24 \cdot 6 = 144$ способа.

Случай $D.2$. Имеется $C_7^2 = 21$ способ расставить две черные пешки на диагонали, но из них надо исключить 6 способов, когда черные пешки стоят рядом. Итого $21 - 6 = 15$ способов.

Складывая все полученные количества способов, находим

$$(120 + 15) \cdot 3 + 144 + 15 = 564$$

способа расставить три пешки.

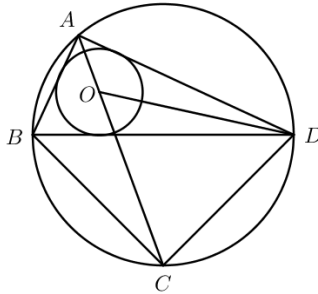
Ответ: 564.

Задача VI.1.2.4. (20 баллов)

Темы: планиметрия, геометрическая вероятность.

Условие

Вокруг четырехугольника $ABCD$ описана окружность с диаметром BD , причем $BC = CD = 8\sqrt{2}$, а $AB = 8$. В треугольник ABD вписана окружность с центром в точке O . Найти вероятность попадания в треугольник COD случайно брошенной в $ABCD$ точки.

Решение

Вычислим площади треугольника COD и четырехугольника $ABCD$.

Треугольники BAD и BCD — прямоугольные (опираются на диаметр описанной окружности), треугольник BCD — равнобедренный (по условию). Поэтому $BD = BC \cdot \sqrt{2} = 16$.

В треугольнике BAD катет $AB = 8$ равен половине гипотенузы, значит, $\angle ADB = 30^\circ$, а $\angle ABD = 60^\circ$.

Имеем: $\angle ACD = \angle ABD = 60^\circ$ (опираются на одну дугу);

$\angle ODB = \frac{1}{2}\angle ADB = 15^\circ$ (так как OD — биссектриса). Тогда

$$\angle ODC = \angle ODB + \angle BDC = 45^\circ + 15^\circ = 60^\circ.$$

Треугольник COD — равносторонний. Поэтому

$$S_{COD} = \frac{1}{2}(8\sqrt{2})^2 \cdot \sin 60^\circ = 32\sqrt{3}.$$

Далее, $S_{BCD} = \frac{1}{2}(8\sqrt{2})^2 = 64$.

Из прямоугольного треугольника BAD находим $AD = \sqrt{16^2 - 8^2} = 8\sqrt{3}$. Тогда $S_{BAD} = \frac{1}{2} \cdot 8 \cdot 8\sqrt{3} = 32\sqrt{3}$.

Получаем

$$S_{ABCD} = 64 + 32\sqrt{3} = 32(2 + \sqrt{3}).$$

Следовательно, искомая вероятность

$$\frac{S_{COD}}{S_{ABCD}} = \frac{32\sqrt{3}}{32(2 + \sqrt{3})} = 2\sqrt{3} - 3.$$

Ответ: $2\sqrt{3} - 3$.

Задача VI.1.2.5. (25 баллов)*Темы: теория чисел, неравенства.***Условие**

Найдите все тройки различных простых чисел p, q, r , для которых верно неравенство $pqr < pq + qr + pr$.

Решение

Не умаляя общности будем считать, что $p < q < r$. Тогда $p \geq 2, q \geq 3, r \geq 5$.

Переносим слагаемое pq в левую часть, имеем неравенство

$$pq(r-1) < r(p+q).$$

Разделим обе части на число $r-1$. Получаем

$$pq < \left(1 + \frac{1}{r-1}\right)(p+q).$$

Поскольку $r \geq 5$, то $1 + \frac{1}{r-1} \leq \frac{5}{4}$. Тогда необходимо

$$pq < \frac{5}{4}(p+q). \quad (\text{VI.1.1})$$

Отсюда, разделив обе части на $\frac{5}{4}pq$, находим

$$\frac{4}{5} < \frac{1}{p} + \frac{1}{q}.$$

Поскольку $p < q$, то

$$\frac{4}{5} < \frac{2}{p},$$

то есть $p < 2,5$. Следовательно, $p = 2$.

Тогда из неравенства (VI.1.1) следует, что $q < \frac{10}{3}$, то есть $q = 3$.

Возвращаясь к неравенству из условия задачи, при $p = 2, q = 3$ получаем

$$6r < 6 + 3r + 2r \iff r < 6.$$

Следовательно, $r = 5$.

Ответ: $(p, q, r) = (2, 3, 5)$, а также всевозможные перестановки этой тройки.

Математика. 10–11 классы**Задача VI.1.3.1. (15 баллов)***Темы: графы.*

Условие

В студенческой группе 10 юношей и 10 девушек. Александр, Алексей и Антон дружат со всеми одноклассниками, Владимир и Владислав с 17, Иван с 8, Тимур и Тарас с 5, Максим и Мирон с 3. Со сколькими девушками дружит Иван?

Решение

Представим граф, в котором вершины обозначают студентов группы, а ребра — дружеские отношения. В изначальном графе вершины Александр, Алексей, Антон имеют степень 19, Владимир и Владислав — 17, Иван — 8, Тимур и Тарас — 5, Максим и Мирон — 3.

После удаления из графа вершин Александр, Алексей, Антон у всех остальных степень понижается на 3. Максима и Мирона также можно исключить, так как они изолированы. Осталось 15 вершин. Теперь Владимир и Владислав имеют степень 14 (то есть дружат со всеми оставшимися), Иван — 5, Тимур и Тарас — 2 (то есть дружат только с Владимиром и Владиславом).

Исключив по тому же принципу Владимира, Владислава, Тимура и Тараса, получаем, что оставшийся Иван имеет степень 3, то есть дружит с тремя девушками.

Ответ: 3.

Задача VI.1.3.2. (20 баллов)

Темы: теория чисел, делимость.

Условие

Найдите такое наименьшее число, что произведение его целой части на дробную равно 2024.

Решение

Пусть α — искомое число, $\alpha = a + b$, где $a \in \mathbb{N}$ — целая, а $b \in (0, 1)$ — дробная часть числа α . Имеем

$$ab = 2024.$$

Тогда

$$\alpha = \frac{2024}{b} + b.$$

Заметим, что функция $\frac{2024}{b} + b$ убывает при $b \in (0, 1)$ (что можно проверить с помощью производной).

Из равенства $ab = 2024$ и включения $a \in \mathbb{N}$ следует, что b — рациональное число. Пусть $b = \frac{p}{q}$, где p и q взаимно просты. Так как число $\frac{2024}{b} = \frac{2024q}{p}$ целое, то p следует искать среди делителей 2024. Поскольку нас интересует число b , наиболее близкое к единице, то следует взять $q = p + 1$ и самое большое p среди возможных, то есть $p = 2024$.

Таким образом,

$$\alpha = \frac{2024 \cdot 2025}{2024} + \frac{2024}{2025} = 2025 \frac{2024}{2025}.$$

Ответ: $2025 \frac{2024}{2025}$.

Задача VI.1.3.3. (20 баллов)

Темы: текстовая задача, системы уравнений.

Условие

В булочной продается два вида пирожных: с кремом и со взбитыми сливками, причем пирожных со взбитыми сливками больше, чем пирожных с кремом. Пекарь заметил, что если увеличить количество пирожных с кремом в некоторое, вообще говоря, нецелое количество раз, а количество пирожных со взбитыми сливками оставить прежним, то на продажу будет выставлено 100 пирожных. Если наоборот, в то же число раз увеличить количество пирожных со сливками, но не менять количество пирожных с кремом, то на продажу будет выставлено 101 пирожное. Сколько пирожных с кремом и со взбитыми сливками мог изначально печь пекарь (перечислите все возможные варианты)?

Решение

Пусть x пирожных с кремом и y — со взбитыми сливками; k — коэффициент увеличения количества пирожных. Тогда

$$\begin{cases} kx + y = 100, \\ x + ky = 101. \end{cases}$$

Вычитая и складывая уравнения системы, получаем

$$\begin{cases} (k-1)(y-x) = 1, \\ (k+1)(y+x) = 201. \end{cases}$$

Исключая k , находим

$$\frac{201}{x+y} - \frac{1}{y-x} = 2.$$

Отсюда

$$201(y-x) - (x+y) = 2(x+y)(y-x).$$

Пусть $u = y - x$, $v = x + y$. Тогда $201u - v = 2uv$, следовательно,

$$v(2u+1) - 201u = 0.$$

Умножим обе части полученного равенства на 2 и прибавим к обеим частям -201 . После разложения на множители, имеем

$$(201 - 2v)(2u + 1) = 3 \cdot 67.$$

Рассмотрим два случая:

1. $201 - 2v = 3, 2u + 1 = 67$. Отсюда $x = 33, y = 66$.

2. $201 - 2v = 67, 2u + 1 = 3$. Отсюда $x = 33, y = 34$.

Остальные случаи не подходят по условию задачи.

Ответ: 33 пирожных с кремом и 66 или 34 со взбитыми сливками.

Задача VI.1.3.4. (20 баллов)

Темы: теория вероятностей, апостериорные вероятности.

Условие

Паша и Андрей работают над одним проектом на GitHub. У каждого есть свой набор задач, на начальном этапе друг от друга независимых. Для каждого написанного класса и каждой функции делается коммит. Однажды Андрей, ввиду неизвестных конфликтов в коде, откатил проект назад на два коммита. Какова вероятность, что эти два коммита были Пашиными, если известно, что это были две функции, а на момент отката Пашей было написано 5 классов и 6 функций, а Андреем — 4 класса и 8 функций?

Решение

Рассмотрим возможные ситуации (гипотезы):

H_1 — оба последних коммита Пашины;

H_2 — первый коммит Пашин, второй — Андрея;

H_3 — первый коммит Андрея, второй — Пашин;

H_4 — оба коммита Андрея.

Обозначим за A событие, что оба последних коммита — это функции.

H_i	$P(H_i)$	$P(A/H_i)$	$P(H_i) \cdot P(A/H_i)$
H_1	$\frac{11}{23} \cdot \frac{10}{22}$	$\frac{6}{11} \cdot \frac{5}{10}$	$\frac{30}{23 \cdot 22}$
H_2	$\frac{11}{23} \cdot \frac{12}{22}$	$\frac{6}{11} \cdot \frac{8}{12}$	$\frac{48}{23 \cdot 22}$
H_3	$\frac{12}{23} \cdot \frac{11}{22}$	$\frac{8}{12} \cdot \frac{6}{11}$	$\frac{48}{23 \cdot 22}$
H_4	$\frac{12}{23} \cdot \frac{11}{22}$	$\frac{8}{12} \cdot \frac{7}{11}$	$\frac{56}{23 \cdot 22}$

Далее вычисляем искомую вероятность по формуле Байеса:

$$P(H_1/A) = \frac{P(H_1) \cdot P(A/H_1)}{\sum_{i=1}^4 P(H_i) \cdot P(A/H_i)} = \frac{30}{23 \cdot 22} : \frac{182}{23 \cdot 22} = \frac{15}{91}.$$

Ответ: $\frac{15}{91}$.

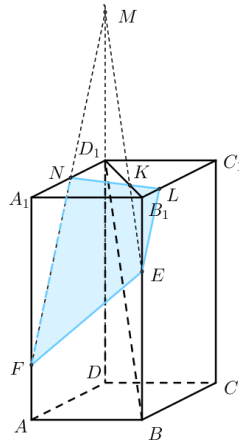
Задача VI.1.3.5. (25 баллов)

Темы: стереометрия.

Условие

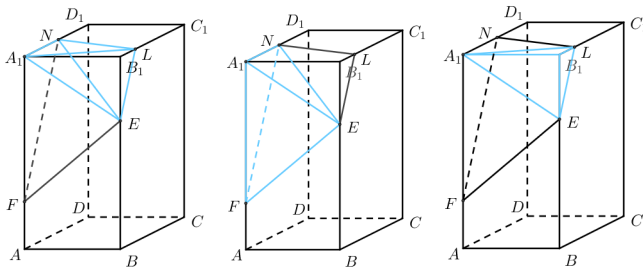
В основании прямоугольного параллелепипеда лежит квадрат, высота параллелепипеда в 2 раза больше стороны основания. На боковом ребре AA_1 отмечена точка F , так что $AF : FA_1 = 1 : 3$, а на ребре BB_1 — точка E , так что $BE : EB_1 = 2 : 1$. Через эти точки проведена плоскость параллельная прямой BD_1 . Найдите отношение объемов параллелепипеда и его большей части, отсеченной плоскостью.

Решение



Пусть точка M — пересечение прямой DD_1 и прямой, проходящей через точку E параллельно BD_1 . Положим $K = EM \cap B_1D_1$, $N = FM \cap A_1D_1$, $L = NK \cap B_1C_1$. Тогда $FNLE$ — плоскость, проведенная в условии задачи.

Пусть сторона основания равна a , тогда высота равна $2a$, а объем параллелепипеда $V = 2a^3$. Вычислим объем отсеченной части A_1FNLE . Для этого разобьем ее на три пирамиды.



1. Пирамида A_1NLE имеет объем $V_{A_1NLE} = \frac{1}{3}S_{A_1NL} \cdot EB_1$. Найдем A_1N . Так как BD_1ME — параллелограмм, то

$$D_1M = BE = \frac{4}{3}a.$$

Треугольники $\triangle FA_1N$ и $\triangle MD_1N$ подобны, поэтому

$$\frac{A_1N}{ND_1} = \frac{A_1F}{D_1M} = \frac{\frac{3}{2}a}{\frac{4}{3}a} = \frac{9}{8}.$$

Отсюда

$$A_1N = \frac{9}{17}A_1D_1 = \frac{9}{17}a, \quad ND_1 = \frac{8}{17}a.$$

Тогда

$$V_{A_1NLE} = \frac{1}{6} \cdot \frac{9}{17}a^2 \cdot \frac{2}{3}a = \frac{a^3}{17}.$$

2. Пирамида A_1NEF имеет объем

$$V_{A_1NEF} = \frac{1}{3}S_{FA_1N} \cdot A_1B_1 = \frac{1}{6} \cdot \frac{9}{17}a \cdot \frac{3}{2}a \cdot a = \frac{9}{68}a^3.$$

3. Пирамида EA_1B_1L имеет объем $V_{EA_1B_1L} = \frac{1}{3}S_{EB_1L} \cdot A_1B_1$. Имеем

$$\frac{D_1K}{KB_1} = \frac{BE}{EB_1} = \frac{2}{1}.$$

Треугольники $\triangle B_1KL$ и $\triangle D_1KN$ подобны, поэтому

$$\frac{B_1L}{ND_1} = \frac{KB_1}{D_1K} = \frac{1}{2}.$$

Отсюда $B_1L = \frac{1}{2}ND_1 = \frac{4}{17}a$. Тогда

$$V_{EA_1B_1L} = \frac{1}{6} \cdot \frac{2}{3}a \cdot \frac{4}{17}a \cdot a = \frac{4}{153}a^3.$$

Получаем

$$V_{A_1FNLE} = V_{A_1NLE} + V_{A_1NEF} + V_{EA_1B_1L} = \frac{1}{17}a^3 + \frac{9}{68}a^3 + \frac{4}{153}a^3 = \frac{133}{612}a^3.$$

Следовательно, искомое отношение объемов равно

$$1 - \frac{V_{A_1FNLE}}{V} = 1 - \frac{133}{612 \cdot 2} = 1 - \frac{133}{1224} = \frac{1091}{1224}.$$

Ответ: $\frac{1091}{1224}$.

Инженерный тур

Общая информация

Участникам необходимо построить рекомендательную систему видеозаписей.

Легенда задачи

Есть видеосервис в социальной сети Одноклассники. Пользователи видеосервиса взаимодействуют с видеозаписями: просматривают и ставят отметки «нравится». Была собрана история таких взаимодействий и разбита на две части: прошлое — для тренировки моделей и будущее — для тестирования. Помимо этого была извлечена информация о пользователях, видеозаписях и каналах, на которые они загружены. Необходимо каждому пользователю порекомендовать десять видеозаписей.

Требования к команде и компетенциям участников

Число участников в команде: 3.

Компетенции, которыми должны обладать члены команды:

1. Аналитик — анализирует данные, выявляет закономерности, предлагает решения.
2. Программист — реализует решения, оптимизирует код.
3. Тестировщик — ищет ошибки в коде, настраивает гиперпараметры.

Оборудование и программное обеспечение

Наименование	Описание
Виртуальный сервер	Все расчеты участники производили на выделенном каждой команде виртуальном сервере. Взаимодействие с сервером осуществлялось через веб-интерфейс интерактивной среды разработки Jupyter Notebook.
Тестирующая система https://cups.online/contests/nto_big_data	Участники загружали свои решения в тестирующую систему для оценки результата.

Описание задачи

Требуется построить систему рекомендации видеозаписей на основе собранной информации, которая лежит в CSV-файлах.

1. `train.csv` — основная таблица содержит информацию о взаимодействии пользователей с видеозаписями. Состоит из четырех колонок:
 - `interaction_type` — тип взаимодействия: `like` или `view`;
 - `timestamp` — дата и время взаимодействия в Unix-формате;
 - `user_id` — идентификатор пользователя;
 - `video_id` — идентификатор видеозаписи.
2. `user.csv` — таблица с информацией о пользователях. Состоит из семи колонок:
 - `user_id` — идентификатор пользователя;
 - `gender` — пол пользователя: `M` или `F`;
 - `age` — возраст пользователя;
 - `language` — основной язык пользователя;
 - `city_id` — хэшированный `id` города, в котором живет пользователь;
 - `birth_city_id` — хэшированный `id` города, в котором родился пользователь;
 - `create_date` — дата создания аккаунта пользователя.Колонки `city_id` и `birth_city_id` содержат данные одного и того же рода.
3. `video.csv` — таблица с информацией о видеозаписях. Состоит из семи колонок:
 - `video_id` — идентификатор видео;
 - `owner_id` — идентификатор канала;
 - `duration` — длительность видео, `s`;
 - `upload_timestamp` — дата и время загрузки видео в Unix-формате.
4. `owner.csv` — таблица с информацией о каналах. Состоит из пяти колонок:
 - `owner_id` — идентификатор канала;
 - `subscribers_count` — число подписчиков;
 - `last_active_date` — дата последней активности;
 - `city_id: string` — хэшированный `id` города, в котором работает автор;
 - `create_date` — дата создания канала.

Необходимо загрузить в тестирующую систему CSV-файл с рекомендацией идентификаторов видеозаписей для каждого пользователя. CSV-файл должен состоять из двух колонок:

- `user_id` — идентификатор пользователя;
- `recommendation` — список из ровно 10 различных идентификаторов видеозаписей, разделенных пробелом.

Система оценивания

Рекомендация оценивается на наборе данных с информацией о будущих взаимодействиях.

Для каждого пользователя вычисляется удовлетворенность предсказанием как сумма числа просмотров и удесятеренного числа отметок `like` для рекомендованных видеозаписей в будущем.

Решение оценивается как средняя удовлетворенность предсказанием по всем пользователям.

Предварительно решение будет оценено на 25% пользователей, а финальное тестирование будет на оставшейся части.

Решение задачи

Можно решить задачу аналитически, если предположить, что пользователи в основном смотрят видеозаписи с каналов, на которые они подписаны. Однако эта информация недоступна в явном виде. Поэтому необходимо для каждого пользователя перебрать видеозаписи, с которыми он взаимодействовал, и определить каналы, на которые они были загружены, а затем рекомендовать наиболее популярные видеозаписи из еще не просмотренных.

Также можно решить задачу общим подходом к построению систем рекомендаций. Например, при помощи разложения матриц. Для этого необходимо представить историю взаимодействия пользователей и видеозаписей как разреженную матрицу, в которой необходимо заполнить пропуски.

В качестве примера приведено лучшее решение команды победителей.

```

1 import pandas as pd
2 import numpy as np
3 import random
4 from implicit.evaluation import mean_average_precision_at_k, train_test_split
5 from implicit.approximate_als import FaissAlternatingLeastSquares
6 from implicit.nearest_neighbours import CosineRecommender,
  ↳ BM25Recommender, TFIDFRecommender
7 from implicit.cpu.lmf import LogisticMatrixFactorization
8 import scipy.sparse as sp
9 from implicit.bpr import BayesianPersonalizedRanking
10 from implicit.als import AlternatingLeastSquares
11
12 train_df = pd.read_csv('train.csv', parse_dates=['timestamp'])
13
14 def chrono_split(df: pd.DataFrame, split_by_column: str = 'user_id', ratio: float
  ↳ = 0.7, col_timestamp: str = 'timestamp'):
15     df = df.sort_values([split_by_column, col_timestamp])
16     groups = df.groupby(split_by_column)
17
18     df["count"] = groups[split_by_column].transform("count")
19     df["rank_s"] = groups.cumcount() + 1
20
21     ratio = [ratio, 1 - ratio]
22     splits = []
23     prev_threshold = None
24     for threshold in np.cumsum(ratio):
25         condition = df["rank_s"] <= round(threshold * df["count"])
26         if prev_threshold is not None:
27             condition &= df["rank_s"] > round(prev_threshold * df["count"])
28         splits.append(df[condition].drop(["rank_s", "count"], axis=1))
29         prev_threshold = threshold
30
31     return splits
32
33 def train_val_split(train_df: pd.DataFrame, val_users_n: int = 200_000):
34     user_ids = train_df['user_id'].unique()
35     user_ids_val = random.sample(list(user_ids), val_users_n)

```

```

36     condition = train_df['user_id'].isin(user_ids_val)
37
38     val = train_df[condition]
39     val_no_targets, val_targets = chrono_split(val, ratio=0.7)
40
41     train = pd.concat([train_df[~condition],
42     ↪ val_no_targets]).sort_values('timestamp')
43     return train, val_no_targets, val_targets
44
45 random.seed(56)
46 train, val_no_targets, val_targets = train_val_split(train_df, val_users_n=100_000)
47 idxes = train.index
48 train['user_id'].iloc[0]
49 train['video_id'].value_counts()[:10]
50 val_no_targets[val_no_targets['interaction_type'] != 'view'].shape[0] /
51 ↪ val_no_targets.shape[0]
52
53 def apk(actual, predicted, k=10):
54     if len(predicted)>k:
55         predicted = predicted[:k]
56
57     score = 0.0
58     num_hits = 0.0
59
60     for i,p in enumerate(predicted):
61         if p in actual and p not in predicted[:i]:
62             num_hits += 1.0
63             score += num_hits / (i+1.0)
64     if not actual:
65         return 0.0
66     return score / min(len(actual), k)
67
68 def mapk(actual, predicted, k=10):
69     return np.mean([apk(a,p,k) for a,p in zip(actual, predicted)])
70
71 val_labels = val_targets.groupby('user_id')['video_id'].apply(lambda
72 ↪ x:x.tolist()).tolist()
73
74 train_best_cols = train['video_id'].value_counts().index[:10]
75 mapk(val_labels, [train_best_cols] * len(val_labels), k=10)
76
77 train['is_like'] = train['interaction_type'].apply(lambda x: 1 if x == 'like' else
78 ↪ 0)
79 video_features = train.groupby('video_id')['is_like'].agg(['sum', 'count'])
80
81 video_features['index'] = video_features['sum'] * 10 + video_features['count']
82
83 video_features['video_id'] = video_features.index
84
85 train_best_cols = video_features.sort_values(by='index')[::-1][:10].index.tolist()
86 mapk(val_labels, [train_best_cols] * len(val_labels), k=10)
87
88 video_features[['video_id', 'index']].index = range(video_features.shape[0])
89 video_features['range'] = range(video_features.shape[0])
90 video_features = video_features.set_index('range')
91
92 train = train.merge(video_features[['video_id', 'index']], on='video_id')
93
94 train['index'] = train['index'].map(lambda x: x ** 0.25)
95

```

```

92 users_inv_mapping = dict(enumerate(train_df['user_id'].unique()))
93 users_mapping = {v: k for k, v in users_inv_mapping.items()}
94
95 items_inv_mapping = dict(enumerate(train_df['video_id'].unique()))
96 items_mapping = {v: k for k, v in items_inv_mapping.items()}
97 len(users_mapping), len(items_mapping)
98
99 def get_coo_matrix(df, user_col='user_id', item_col='item_id',
100 ↪ weight_col=None, users_mapping=None, items_mapping=None):
101     if weight_col is None:
102         weights = np.ones(len(df), dtype=np.float32)
103     else:
104         weights = df[weight_col].astype(np.float32)
105
106     interaction_matrix =
107     ↪ sp.coo_matrix((weights, (df[user_col].map(users_mapping.get),
108     ↪ df[item_col].map(items_mapping.get))))
109     return interaction_matrix
110
111 train_mat = get_coo_matrix(df=train, user_col='user_id', item_col='video_id',
112 ↪ weight_col='index', users_mapping=users_mapping,
113 ↪ items_mapping=items_mapping).tocsr()
114
115 model = LightFM(no_components=5)
116 model.fit(train_mat)
117
118 def predict_impl(model, test_users, mat, users_mapping, items_inv_mapping, N=10,
119 ↪ falh=True):
120     recs, scores = [], []
121     for id in (test_users):
122         row_id = users_mapping[id]
123         ranks = model.predict(row_id, mat[row_id])
124         recs += [[items_inv_mapping.get(it) for it in ranks[0]]]
125         scores += [ranks[1]]
126     return recs, scores
127
128 def predict_impl_batched(model, test_users, mat, users_mapping, items_inv_mapping,
129 ↪ batch_size=1024, N=10, falh=True):
130     recs, scores = [], []
131     N = len(test_users)
132     for i in (range(0, N, batch_size)):
133         ids = test_users[i:i+batch_size]
134         row_id = [users_mapping[id] for id in ids]
135         ranks_lst = model.predict(row_id, mat[row_id], N=N)
136         for ranks in ranks_lst:
137             recs += [[items_inv_mapping.get(it) for it in ranks[0]]]
138             scores += [ranks[1]]
139     return recs, scores
140
141 val_group = val_targets.groupby('user_id')
142 act = val_group['video_id'].agg(lambda x: x.tolist()).tolist()
143 val_users = val_group.agg(lambda x: x.tolist()).index.tolist()
144
145 ktop = 100
146 test_preds, test_scores = predict_impl(model, val_users, train_mat, users_mapping,
147 ↪ items_inv_mapping, N=ktop, falh=True)
148 test_preds, test_scores = predict_impl(model, val_users, train_mat, users_mapping,
149 ↪ items_inv_mapping, N=ktop, falh=True)
150
151 mapk(val_labels, test_preds, k=ktop)

```

```

143
144 pd.DataFrame({'preds':test_preds,
↳ 'scores':test_scores}).to_parquet('TFIDF_preds_val100.parquet')
145
146 train_df['is_like'] = train_df['interaction_type'].apply(lambda x: 1 if x ==
↳ 'like' else 0)
147 video_features = train_df.groupby('video_id')['is_like'].agg(['sum', 'count'])
148 video_features['index'] = video_features['sum'] * 10 + video_features['count']
149 video_features['video_id'] = video_features.index
150 video_features[['video_id', 'index']].index = range(video_features.shape[0])
151 video_features['range'] = range(video_features.shape[0])
152 video_features = video_features.set_index('range')
153 train_df = train_df.merge(video_features[['video_id', 'index']], on='video_id')
154 train_df['index'] = train_df['index'].map(lambda x: x ** 0.25)
155
156 train_mat = get_coo_matrix(df=train_df, user_col='user_id', item_col='video_id',
↳ weight_col='index', users_mapping=users_mapping,
↳ items_mapping=items_mapping).tocsr()
157
158 model = TFIDFRecommender(K=ktop)
159 model.fit(train_mat)
160
161 sample_sub = pd.read_csv('sample_submission.csv')
162 test_id = sample_sub['user_id'].tolist()
163
164 test_preds, test_scores = predict_impl(model, test_id, train_mat, users_mapping,
↳ items_inv_mapping, N=ktop, falh=True)
165
166 pd.DataFrame({'preds':test_preds,
↳ 'scores':test_scores}).to_parquet('TFIDF_preds_test100.parquet')

```

Материалы и курсы для подготовки

- Онлайн-учебник по машинному обучению:
<https://education.yandex.ru/handbook/ml>.
- Начало работы с Pandas:
http://pandas.geekwriter.ru/getting_started/index.html.
- Основы анализа данных и Python:
<https://practicum.yandex.ru/data-analysis-basic/>.
- Введение в Data Science и машинное обучение:
<https://stepik.org/course/4852/>.
- Рекомендательные системы на практике:
<https://stepik.org/course/172126/>.

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 125 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 300 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 $K_1 = 0, 2$ каждый предмет и баллы за командное решение задач инженерного тура с весом $K_2 = 0, 6$.

Итоговый балл определяется по формуле:

$$S = K_1 \cdot (S_1 + S_2) + K_2 \cdot S_3,$$

где S_1 — балл первой части заключительного этапа по математике (предметный тур) в стобальной системе ($S_{1 \text{ макс}} = 100$);

S_2 — балл первой части заключительного этапа по информатике (предметный тур) в стобальной системе ($S_{2 \text{ макс}} = 100$);

S_3 — итоговый балл инженерного командного тура в стобальной 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 5 победителей и 12 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	59,15 и выше
Призеры	От 44,99 до 55,83

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».