



НТО

МАТЕРИАЛЫ ЗАДАНИЙ

Всероссийской междисциплинарной олимпиады школьников

«Национальная технологическая олимпиада»

по профилю

«Интеллектуальные энергетические системы»

2021/22 учебный год

<http://ntcontest.ru>

Оглавление

1 Профиль «Интеллектуальные энергетические системы»	5
I Первый отборочный этап	9
I.1 Задачи первого этапа. Информатика	9
I.1.1 Первая попытка. Задачи 8–11 класса	9
I.1.2 Вторая попытка. Задачи 8–11 класса	18
I.1.3 Третья попытка. Задачи 8–11 класса	25
I.1.4 Четвертая попытка. Задачи 8–11 класса	34
I.2 Задачи первого этапа. Математика	45
I.2.1 Первая попытка. Задачи 8–9 класса	45
I.2.2 Первая попытка. Задачи 10–11 класса	47
I.2.3 Вторая попытка. Задачи 8–9 класса	51
I.2.4 Вторая попытка. Задачи 10–11 класса	53
I.2.5 Третья попытка. Задачи 8–9 класса	56
I.2.6 Третья попытка. Задачи 10–11 класса	59
I.2.7 Четвертая попытка. Задачи 8–9 класса	64
I.2.8 Четвертая попытка. Задачи 10–11 класса	66
II Второй отборочный этап	71
II.1 Индивидуальная часть	71
II.2 Командная часть	80
III Заключительный этап	95
III.1 Индивидуальный предметный тур	95
III.1.1 Информатика. 8–11 класс	95
III.1.2 Математика. 8–9 классы	108
III.1.3 Математика. 10–11 классы	112
III.2 Командный практический тур	120

III.2.1 Требования к команде	120
III.2.2 Оборудование и программное обеспечение	121
III.2.3 Описание задачи	124
IV Критерии определения победителей и призеров	145

Профиль «Интеллектуальные энергетические системы»

Профиль «Интеллектуальные энергетические системы» посвящен моделированию энергетических систем ближайшего будущего, в основе которых лежат технологии, существующие в том или ином виде, но еще не адаптированные и не интегрированные в единый комплекс. Реализация энергосети в архитектуре интернета энергии требует проектирования и строительства многочисленных надежных гибких энергосистем, эффективно распределяющих электроэнергию в том числе за счет использования альтернативных источников и взаимодействия с внешним рынком мощностей. Важное место в профиле занимают экономические модели, которые в настоящее время не распространены — в частности, биржа экономических микроконтрактов, являющаяся одним из главных компонентов технологии Smart Grid. Успешная работа с такой биржей предполагает применение автоматизации, создание оптимальных стратегий и алгоритмов анализа параметров энергосети, в том числе с применением информационных технологий.

Представленные в профиле задачи энергетики требуют знания математики школьного уровня, а именно теории вероятностей, геометрии и основ анализа. Помимо школьных знаний и навыков, требуется самостоятельное освоение таких тем, как теория игр, теория аукционов, программирование на языке Python, основы численных методов в решении математических задач. Большинство задач профиля требует реализации их решения в программном виде, будь то компонент существующей системы (управляющий скрипт энергосистемы) или самостоятельная программа (инструменты для принятия решений).

Отборочные этапы и подготовительные мероприятия по профилю проектируются таким образом, чтобы в течение года плавно ввести учащихся в контекст и культуру профиля, а также подготовить их к финальной задаче.

Первый отборочный этап (дистанционный индивидуальный) состоит из решения предметных задач по информатике и математике. Этот этап решает несколько задач: отбор участников, уверенно владеющих школьной программой; предоставление возможности сильнейшим участникам из разных городов найти друг друга и объединиться в команды; актуализация знаний именно по тем предметным областям, которые понадобятся в дальнейшем.

На втором отборочном этапе (дистанционном командном) участники формируют команды и загружают общее командное решение. Функций у этого этапа несколько: отбор команд из трех–пяти участников, способных решать сложные задачи; ознакомление участников с математическими и физическими концепциями, на которых будет построена задача заключительного этапа; предъявление командам требований к уровню программирования, идентичных тем, что будут предъявлены им на заключительном этапе. Во время второго отборочного этапа участники решают задачи, которые отражают часть большой командной задачи заключительного этапа, и знакомятся с базовыми концепциями — работают с вероятностными прогнозами, графами, аукционами, задачами на оптимизацию.

Основная задача заключительного этапа профиля — разработка стратегии опти-

мального построения умной энергосети и управления ею в условиях локальной конкуренции. В распоряжении участников находится модель поселения с потребителями разного уровня (жилые дома, больницы, заводы) и электростанциями на альтернативных источниках энергии. Для процесса моделирования используется специальный стенд, также воссоздающий погодные условия (ветер и освещенность) и рынок электроэнергии (многоуровневая биржа микроконтрактов). На сегодня это необходимый навык специалистов будущего, который лучше закладывать в школьном возрасте. Задача командного практического тура заключительного этапа представляет собой комплексную интегральную задачу, полное оптимальное решение которой чрезвычайно сложно. Однако полное приближенное решение способна найти любая команда. Качество и сложность используемых приближений отражает глубину понимания, знаний и уровень способностей участников. Командная задача заключается в экономическом и энергетическом моделировании энергосистемы и конкуренции предложенных участниками решений. Командам необходимо проанализировать прогнозы погоды и поведения потребителей, распределить объекты потребления и генерации на аукционе, собрать из них энергосеть, настроить автоматизированную систему управления (путем написания скриптов на языке Python) и испытать ее в ходе моделирования энергосистемы. Конечной метрикой эффективности разработанной стратегии является прибыль, полученная в результате штатного функционирования энергосистемы. Система оценки полностью автоматизирована и спроектирована таким образом, чтобы однозначно и численно оценить качество найденных и реализованных участниками решений.

Помимо командной задачи (практический тур) на заключительном этапе участникам необходимо решить индивидуальные задачи по предметам информатика и математика (предметный тур). Задачи этого этапа преследуют следующие цели: объективно проверить индивидуальные знания участников; косвенно оценить индивидуальный вклад участников в результат командной работы. Вес этих задач в финальной оценке участников составлял 40%.

Связь между этапами

Структура профиля выстроена таким образом, чтобы не просто провести соревнования, а выстроить годичную образовательную программу для участников профиля. Для задачи второго отборочного и заключительного этапов связаны между собой. Второй этап позволяет очертить область предметных знаний, необходимую для участия в заключительном этапе профиля. В текущем году второй отборочный этап включает задачи индивидуального модуля и задачи командного модуля, для решения которых достаточно школьных знаний и умений, навыков использовать школьные знания и информационный поиск для решения новых задач. Целью задач второго отборочного этапа является подготовка к практическому командному туру заключительного этапа.



Рис. 1.1. На схеме представлена взаимосвязь задач заключительного этапа с темами задач второго тура

Теория игр

На заключительном этапе участникам предстоит на одном поле столкнуться с другими командами, и понимание основ теории игр позволит объективнее оценивать игровую ситуацию в условиях конкуренции за ресурсы.

Математические модели

Работа со сложными математическими моделями и системами — фундаментальный навык, в полной мере раскрывающийся при работе с задачей заключительного этапа.

Теория вероятностей

Мир сложен и неустойчив, и задача заключительного этапа моделирует это в полной мере. Лучше заранее подготовиться и научиться работать с вероятностями, для чего второй отборочный этап содержит достаточно много задач по теории вероятностей. При этом для полноценной работы не потребуется погружаться в нее с головой — достаточно знания основ матстатистики и распределений случайных величин, но даже это даст преимущество в работе над задачей заключительного этапа.

Физика

В работе с энергетическими системами необходимо знать электротехнику в частности и физику в плане построения и работы с физическими моделями.

Алгоритмы

Задача заключительного этапа предполагает написание управляющего скрипта, и здесь важную роль играет навык разработки алгоритмов, равно как и поиска подходящих типовых. На проработку этих навыков и рассчитаны задачи раздела «Алгоритмы». При работе с ними важно делать акцент на информационном поиске и умении выявить типовую подзадачу.

Графы

Энергосети — это графы, и с ними нужно уметь работать. В работе с задачами этого раздела в первую очередь важно овладеть основным арсеналом — программными представлениями графов и базовыми алгоритмами. Это может понадобиться при написании управляющего скрипта на заключительном этапе, не говоря уже о фундаментальном понимании сетей.

Аукционы

Каждая игровая сессия начинается с аукциона, знания о котором лежат в пересечении теории игр, экономики и психологии. Несмотря на то, что этот этап в финале динамичен и иногда непредсказуем, его результаты влияют на всю остальную игру. Команда, обладающая основами теории аукционов, имеет значительное и глобальное преимущество перед другими.

В решении задач заключительного этапа участникам требуются навыки и знания из школьной программы, а также самостоятельно приобретенные на хакатонах и при решении задач второго этапа:

- Умение программировать на языке Python.
- Умение работать со случайными величинами в программных вычислениях.
- Умение решать базовые оптимизационные задачи.
- Навыки реализации решений математических задач в виде программ.
- Понимание правил и решения закрытого аукциона первой цены.
- Навыки командной работы над программным кодом.
- Навыки работы с большими рядами данных в математических задачах.
- Навыки работы с рядами данных в алгоритмах.
- Навыки работы с рекурсивными структурами в алгоритмах.
- Понимание принципов работы и характеристик ветрогенераторов.
- Теория вероятностей.

Решение задач командного заключительного этапа предполагает освоение и развитие следующих навыков и умений:

- Умение работы в команде.
- Умение самостоятельно выделять и формулировать подзадачи.
- Понимание принципов оценки риска.
- Навыки программирования, в том числе на языке Python.
- Навыков работы с рядами данных, как аналитической, так и алгоритмической.
- Навыки работы с системами с инерцией.
- Практического использования теории вероятностей, в том числе в программных вычислениях.

Методические материалы для самостоятельного освоения предоставляются участникам во время подготовительных мероприятий второго отборочного этапа, проходящих в виде еженедельных вебинаров.

Первый отборочный этап

Задачи первого этапа. Информатика

Первая попытка. Задачи 8–11 класса

Задача I.1.1.1. Расчет скидки (20 баллов)

Темы: задачи для начинающих, простая математика.

Одна продуктовая сеть в рамках акции выдает скидочные купоны двух видов. По первому купону можно получить скидку в 8% от стоимости покупки, но не более 100 рублей. По второму купону можно получить скидку в 5% от стоимости покупки без других ограничений. Предъявлять можно только один купон, разделять покупку на части нельзя. Покупатель делает покупку на p рублей. У него есть оба купона. Напишите программу, которая вычислит максимальный размер скидки, которую покупатель сможет получить.

Формат входных данных

На вход подается одно целое число — размер покупки в рублях. Число не превосходит 10000.

Формат выходных данных

Вывести одно число — размер скидки в рублях. Ответ может оказаться не целым.

Методика проверки

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
810
Стандартный вывод
64.8

Пример №2

Стандартный ввод
1530
Стандартный вывод
100

Пример №3

Стандартный ввод
10000
Стандартный вывод
500

Решение

В этой задаче требуется рассмотреть два варианта. Обозначим сумму покупки за p .

При использовании купона со скидкой в 8% сумма скидки является минимумом из 100 и $0,08p$. При использовании купона со скидкой в 5% сумма скидки будет равна $0,05p$.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
1 p=int(input())
2 print(max(p*0.05,min(p*0.08,100)))
```

Задача I.1.1.2. Произведение многочленов (20 баллов)

Темы: реализация, простая математика.

Многочлены — это одни из самых распространенных математических объектов, которые используются практически во всех прикладных областях. Задан многочлен $a_n x^n + a_{n-1} x^{n-1} + \dots + a_2 x^2 + a_1 x + a_0$. От вас требуется написать программу, которая найдет произведение этого многочлена на $x + 1$. Многочлен задан своими коэффициентами $a_n, a_{n-1}, \dots, a_2, a_1, a_0$. Обратите внимание, что многочлен степени n состоит из $n + 1$ одночлена. Некоторые из одночленов могут отсутствовать. В этом случае соответствующий коэффициент считается равным нулю.

Например, многочлен $2x^3 + 3x^2 + 1$ будет задан набором коэффициентов 2 3 0 1. Результатом умножения будет многочлен четвертой степени с набором коэффициентов 2 5 3 1 1, что можно проверить, раскрыв скобки.

$$(2x^3 + 3x^2 + 1)(x + 1) = 2x^4 + 3x^3 + x + 2x^3 + 3x^2 + 1 = 2x^4 + 5x^3 + 3x^2 + x + 1$$

Формат входных данных

На вход программы в первой строке подается одно натуральное число n — степень многочлена. $1 \leq n \leq 100$. Далее во второй строке через пробел подается $n + 1$ целое число — коэффициенты многочлена $a_n, a_{n-1}, \dots, a_2, a_1, a_0$. Каждый из коэффициентов не превосходит 1000 по абсолютной величине. $a_n \neq 0$.

Формат выходных данных

Требуется вывести через пробел $n + 2$ коэффициента полученного многочлена.

Если вы программируете на Python, то убрать перенос строки в функции `print` можно при помощи именованного параметра `end`, например, `print(a, end=' ')`.

Методика проверки

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
3
2 3 0 1
Стандартный вывод
2 5 3 1 1

Решение

Найдем произведение многочлена $a_n x^n + a_{n-1} x^{n-1} + \dots + a_2 x^2 + a_1 x + a_0$ и $x + 1$.

$$\begin{aligned}
 & (a_n x^n + a_{n-1} x^{n-1} + \dots + a_2 x^2 + a_1 x + a_0)(x + 1) = \\
 & = (a_n x^{n+1} + a_{n-1} x^n + \dots + a_2 x^3 + a_1 x^2 + a_0 x) + \\
 & \quad + (a_n x^n + a_{n-1} x^{n-1} + \dots + a_2 x^2 + a_1 x + a_0) = \\
 & = a_n x^{n+1} + (a_{n-1} + a_n) x^n + \dots + (a_1 + a_2) x^2 + (a_0 + a_1) x + a_0
 \end{aligned}$$

Таким образом каждый коэффициент, кроме первого и последнего, является суммой двух соседних элементов исходного списка.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 input()
2 x=map(int,input().split())
3 prev=int(next(iter(x)))
4 print(prev,end=' ')

```

```
5 for t in x:
6     print(t+prev,end=' ')
7     prev=t;
8 print(prev)
```

Ниже представлено решение в функциональном стиле на языке Python 3.

```
1 x=list(map(int,input().split()))
2 print(*[a+b for (a,b) in zip([0]+x,x+[0])])
```

Задача I.1.1.3. Очередь (20 баллов)

Темы: структуры данных.

Студенческая группа сдает зачет преподавателю. Он расположил студентов по некоторому неизвестному порядку и сообщил, кто после кого сдает зачет. Теперь студенты хотят выяснить, в каком порядке им приходиться, чтобы не стоять всей группой за дверью.

Формат входных данных

На вход программы в первой строке подается одно натуральное число n — количество студентов в группе. $2 \leq n \leq 30$. Далее в $n - 1$ строке через пробел подается по два имени: имя студента, сдающего зачет, и имя того студента, который будет сдавать перед ним. Имена не содержат пробелов и состоят только из строчных и прописных символов латиницы. Гарантируется, что очередь задана корректно, имена студентов в группе не повторяются.

Формат выходных данных

Требуется вывести имена всех студентов группы в том порядке, в котором они сдают зачет. Каждое имя выводится в отдельной строке.

Методика проверки и пояснение к тесту

В приведенном примере в группе 5 студентов. Первой сдает зачет Лиза, поскольку только для нее не указано, кто приходит раньше. После Лизы приходит Иван, далее Мария, затем Петр и последним сдает Игорь.

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
5 Petr Mariya Ivan Liza Mariya Ivan Igor Petr
Стандартный вывод
Liza Ivan Mariya Petr Igor

Решение

В этой задаче требуется продемонстрировать умение работать со структурами данных.

Наиболее простым способом решения будет создание ассоциативного массива (словаря), в котором для каждого студента будет указано имя следующего в очереди.

Дополнительной сложностью является нахождение имени первого студента. Для этого надо найти имя, которое есть среди ключей, но которого нет среди значений словаря. Такая операция может быть записана как разность множества ключей и множества значений словаря. По условию такая разность будет содержать только один элемент, который и будет искомым именем.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1  n=int(input())
2  nxt=dict()
3  for i in range(n-1):
4      val,key=input().split()
5      nxt[key]=val
6  cur=next(iter(nxt.keys()-nxt.values()))
7  print(cur)
8  for i in range(n-1):
9      cur=nxt[cur]
10     print(cur)

```

Задача I.1.1.4. Четырехугольник (20 баллов)

Темы: геометрия, неравенство треугольника, перебор, реализация.

Сергею на уроке геометрии задали следующее задание. Даны 5 чисел. Требуется нарисовать произвольный четырехугольник с одной диагональю так, чтобы длины

сторон и этой диагонали равнялись заданным числам. Сергею надо выбрать длину диагонали и каждую из сторон так, чтобы было возможно нарисовать требуемую фигуру. Если вариантов решения задачи несколько, можно выбрать любой. Нарисованная диагональ не должна лежать на одной из сторон. Возможно, что нарисовать требуемый четырехугольник не получится. В этом случае надо будет вывести ноль.

Формат входных данных

На вход через пробел подаются 5 натуральных чисел от 1 до 1000.

Формат выходных данных

Требуется вывести ответ в следующем порядке. В первой строке вывести число — длину диагонали четырехугольника. Во второй строке 2 числа — длины отрезков, лежащих с одной стороны от диагонали. В третьей строке еще 2 числа — длины отрезков, лежащих с другой стороны от диагонали. Если построить четырехугольник невозможно, то вывести 0.

В этой задаче можно выводить любой правильный ответ. В частности, можно переставить числа в одной строке или поменять местами вторую и третью строку.

Методика проверки и пояснение к тестам

В первом тесте в качестве диагонали можно взять отрезок длины 7. Тогда с одной стороны от диагонали будут стороны с длинами 3 и 5, а с другой — 9 и 3. Вторую и третью строку, а также числа в этих строках можно вывести в любом порядке. Также возможно нарисовать четырехугольник с диагональю 5 и длинами сторон 9, 7 и 3, 3. Кроме того, возможен вариант с диагональю 3 и длинами сторон 5, 3 и 7, 9. Любой из этих вариантов будет считаться верным.

Во втором тесте нарисовать четырехугольник невозможно. Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
3 9 5 3 7
Стандартный вывод
7
3 5
9 3

Пример №2

Стандартный ввод
3 9 5 1 7
Стандартный вывод
0

Решение

Для решения этой задачи требуется знать неравенство треугольника, которое говорит о том, что в невырожденном треугольнике сумма длин двух любых сторон больше, чем длина третьей. Четырехугольник с одной диагональю можно представить как два треугольника, смежных по одной стороне.

Решение должно заключаться в переборе нескольких вариантов. Этот перебор можно организовать разными способами. Один из способов заключается в использовании функций для генерации перестановок. В Python это функция `permutations` из модуля `itertools`. Можно получить все перестановки пяти заданных чисел и проверить их по неравенству треугольника.

Чтобы получить другой способ решения, можно заметить, что две самых длинных стороны в любом случае выгоднее использовать при построении одного треугольника. Тогда количество вариантов сокращается до трех.

Пример программы-решения

Ниже представлено решение на языке Python 3 с полным перебором.

```

1  from itertools import permutations
2  def triangle(a,b,c):
3      return a<b+c and b<a+c and c<a+b
4  x=map(int,input().split())
5  for t in list(permutations(x)):
6      if triangle(t[0],t[1],t[2]) and triangle(t[0],t[3],t[4]):
7          print(t[0])
8          print(t[1],t[2])
9          print(t[3],t[4])
10         break
11  else:
12         print(0)

```

Ниже представлено решение на языке Python 3 с сортировкой.

```

1  x=sorted(list(map(int,input().split())),reverse=True)
2  if x[0]<x[1]+x[2] and x[2]<x[3]+x[4]:
3      print(x[2])
4      print(x[0],x[1])
5      print(x[3],x[4])
6  elif x[0]<x[1]+x[3] and x[1]<x[2]+x[4]:
7      print(x[1])
8      print(x[0],x[3])
9      print(x[2],x[4])
10 elif x[0]<x[1]+x[4] and x[1]<x[2]+x[3]:
11     print(x[1])
12     print(x[0],x[4])
13     print(x[2],x[3])
14 else:
15     print(0)

```

Задача I.1.1.5. Рыцари (20 баллов)

Темы: структуры данных, реализация, алгоритмическая сложность.

В королевстве Логрес за круглым столом собираются рыцари. Каждый рыцарь гордится своими победами, поэтому делает на своих доспехах царاپины по количеству побежденных противников. Ранг рыцаря определяется по количеству царापин. Рыцари весьма горды и тот, чей ранг ниже, должен оказывать почтение тому, чей ранг выше. Если вдруг возникает ситуация, что подряд сидят рыцари с одинаковым рангом, то они устраивают между собой турнир, по итогам которого определяется один победитель. Он ставит на доспехи новые царاپины (если в турнире участвовало k рыцарей, то победитель поставит $k - 1$ новую царापину) и возвращается за стол на свое место. Остальные участники турнира уходят залечивать раны и уязвленное самолюбие. Если после этого вновь возникает аналогичная ситуация, то проводится новый турнир, и так далее.

За круглым столом собралось n рыцарей, и они предусмотрительно расселись так, чтобы ранги соседей были различными. Но опоздавший Галахад все испортил. Он сел на случайно выбранное место, и карусель турниров снова закрутилась.

Известны ранги всех n рыцарей, изначально сидевших за столом. Также известен ранг Галахада и место, на которое он сел. Напишите программу для определения количества рыцарей, которые останутся за столом после того, как все турниры завершатся.

Формат входных данных

В первой строке на вход подается число n — количество рыцарей без учета Галахада. $1 \leq n \leq 300000$. Во второй строке через пробел записаны n натуральных чисел r_1, \dots, r_n — ранги всех рыцарей. $r_i \leq 10^6$; $r_i \neq r_{i+1}$; $r_1 \neq r_n$. В третьей строке через пробел записаны два натуральных числа p и t . $1 \leq p \leq n$; $t \leq 10^6$. Число p задает место, на которое сел Галахад и означает, что он оказался между рыцарями с номерами p и $p + 1$. Если $p = n$, то Галахад находится между первым и последним рыцарем. Число t задает исходный ранг Галахада.

Формат выходных данных

Требуется вывести одно число — ответ к задаче.

Методика проверки и пояснение к тесту

После того, как Галахад сядет за стол после второго рыцаря, ранги рыцарей за столом будут (7 5 5 6 8 9 10 12 10 8). Два рыцаря с рангом 5 проведут турнир, останется один из них, ранг которого повысится до 6 (7 6 6 8 9 10 12 10 8). Из двух рыцарей с рангом 6 вновь останется 1 с рангом 7 (7 7 8 9 10 12 10 8). После очередного турнира получим (8 8 9 10 12 10 8). Помня, что стол круглый, видим 3 рыцарей с рангом 8, сидящих подряд. Из них останется один с рангом 10 (10 9 10 12 10). Будет проведен еще один турнир, после которого оставшиеся 4 рыцаря не будут иметь соседей с одинаковым рангом (11 9 10 12).

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
9 7 5 6 8 9 10 12 10 8 2 5
Стандартный вывод
4

Решение

В этой задаче требовалось придумать достаточно простой в реализации способ решения, который при этом удовлетворял бы требованию по времени работы программы. Для этого необходимо, чтобы элементы не удалялись из массива, так как эта операция достаточно затратная по времени. Дополнительной сложностью является то, что последовательность элементов должна быть циклической.

Заметим, что все изменения могут происходить вокруг позиции Галахада, поэтому будет удобно, если его ранг не будет храниться в последовательности, а номера рыцарей слева от него будут начинаться с нуля. Тогда справа от Галахада будет последний элемент последовательности. Такого состояния можно добиться при помощи циклического сдвига всей последовательности.

Далее будем использовать метод двух указателей. Номер рыцаря слева от Галахада будем хранить в переменной i , а номер правого рыцаря — в переменной j . Вместо удаления элементов будем просто смещать значения этих переменных.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1  n=int(input())
2  lst=list(map(int,input().split()))
3  p,r=map(int,input().split())
4  lst=lst[p:]+lst[:p]
5  i=0
6  j=n-1
7  while i<j:
8      if lst[i]==lst[j]==r:
9          r+=2
10         i+=1
11         j-=1
12     elif lst[i]==r:
13         r+=1
14         i+=1
15     elif lst[j]==r:
16         r+=1
17         j-=1
18     else:
19         break
20 if i==j and lst[i]==r:
21     i+=1
22 print(j-i+2)

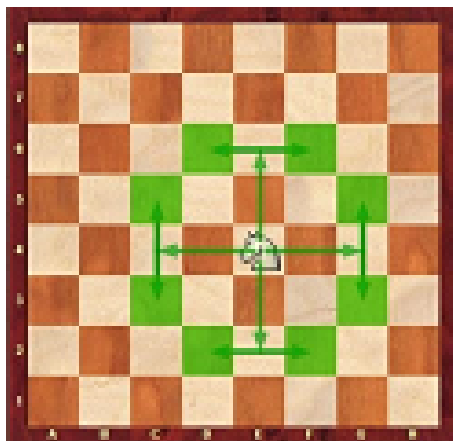
```

Вторая попытка. Задачи 8–11 класса

Задача I.1.2.1. Конь (20 баллов)

Темы: задачи для начинающих, перебор.

Шахматный конь стоит на доске размером 8×8 в i -той строке и j -том столбце. Напишите программу, которая определит, сколько ходов он может сделать.



Конь ходит, как показано на рисунке. Из центральной части доски он может сделать 8 ходов, но если конь находится ближе к краю доски, то количество ходов уменьшится, так как он не может выйти за ее границы.

Формат входных данных

На вход подается два натуральных числа в диапазоне от 1 до 8 — номер клетки, в которой находится конь, по горизонтали и вертикали. Каждое число записано в отдельной строке.

Формат выходных данных

Вывести одно число — количество возможных ходов коня.

Методика проверки

Программа проверяется на 40 тестах. Прохождение каждого теста оценивается в 0,5 балла. Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
5
3
Стандартный вывод
8

Решение

В предложенном решении этой задачи перебираются все клетки шахматной доски, и выполняется проверка того, что конь может сходить в эту клетку. Возможны и другие решения.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1  a=int(input())
2  b=int(input())
3  ans=0
4  for i in range(1,9):
5      for j in range(1,9):
6          if a!=i and b!=j and abs(a-i)+abs(b-j)==3:
7              ans+=1
8  print(ans)

```

Задача I.1.2.2. Королевство чисел (20 баллов)

Темы: задачи для начинающих, простая математика.

Алиса и Боб стали королями в королевствах на множестве натуральных чисел. Подданными Алисы являются все натуральные числа, которые делятся на 3 без остатка, а все остальные числа стали подданными Боба. Алиса дружит с Бобом, и они хотят, чтобы их подданные тоже дружили между собой. Они разбили все числа на пары, причем i -тое по порядку число из королевства Алисы будет дружить с i -тым по порядку числом из королевства Боба. Вам задан набор из n чисел. Напишите программу для нахождения друга каждого из чисел.

Первые 10 чисел из королевства Алисы — это $\{3, 6, 9, 12, 15, 18, 21, 24, 27, 30, \dots\}$. Первые 10 чисел из королевства Боба — это $\{1, 2, 4, 5, 7, 8, 10, 11, 13, 14, \dots\}$. Таким образом, парами друзей являются $(3, 1)$ $(6, 2)$ $(9, 4)$ и так далее.

Формат входных данных

На вход в первой строке подается натуральное число n — количество чисел в наборе. $1 \leq n \leq 10^5$. Во второй строке через пробел подается n натуральных чисел a_1, a_2, \dots, a_n . Числа не превосходят 10^{18} . Обратите внимание, что для хранения таких чисел в программе на C++ вам потребуется тип `long long`. В программе на PascalABC такой тип называется `Int64`.

Формат выходных данных

Программа должна вывести через пробел n натуральных чисел b_1, b_2, \dots, b_n . Число b_i должно быть другом числа a_i .

Если вы программируете на Python, то заменить перенос строки на пробел в функции `print` можно при помощи именованного параметра `end`, например `print(a, end=' ')`.

Методика проверки

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется. В первых 5 тестах $n \leq 10$, $a_i \leq 1000$. В следующих пяти тестах $n \leq 10^5$, $a_i \leq 10^6$. В последних 10 тестах $a_i \leq 10^{18}$.

Примеры

Пример №1

Стандартный ввод
10
1 2 3 4 5 6 7 8 9 10
Стандартный вывод
3 6 1 9 12 2 15 18 4 21

Решение

Можно заметить, что соответствующие числа в различных множествах различаются примерно в два раза. Поэтому требуемые формулы можно получить, умножая или деля заданное число на 2. В зависимости от четности к результату надо будет прибавить некоторую константу. С использованием свойств целочисленной арифметики программу можно упростить.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 n=int(input())
2 for x in map(int,input().split()):
3     if x%3!=0:
4         print(3*(x-x//3),end=' ')
5     else:
6         print((x-1)//2,end=' ')

```

Задача I.1.2.3. Алфавит (20 баллов)

Темы: строки, символы, структуры данных.

Панграммой называется строка, в которой присутствуют все буквы алфавита. Однако при этом строка может содержать пробелы. У Алисы есть строка, состоящая из строчных и заглавных символов латиницы. Она хочет убедиться, что эта строка является панграммой. Алиса действует следующим образом. Для начала она удаляет из строки все пробелы, а также заменяет все заглавные буквы на строчные. Далее она вырезает из строки символы и составляет из них вторую строку по такому алгоритму. Она перебирает последовательно все буквы алфавита от «a» до «z» и пытается найти самое первое вхождение этой буквы в строке. Если вхождение нашлось, то Алиса вырезает из строки эту букву и добавляет ее справа ко второй строке и переходит к следующей букве.

Например, из строки «**A Bra Kada Bra**» будет получена строка «**abrakadabra**», далее из нее последовательно будут вырезаны самые первые вхождения букв **a**, **b**, **d**, **k**, **r**, после чего она превратится в строку **aaabra**.

Вы должны написать программу, которая определит содержимое обеих строк после преобразований Алисы.

Формат входных данных

На вход подается одна строка, содержащая не более 200 строчных и заглавных символов латиницы и пробелов. Гарантируется, что хотя бы один из символов латиницы входит в строку несколько раз.

Формат выходных данных

Программа должна вывести обе полученные строки без пробелов.

Методика проверки

Программа проверяется на 10 тестах. Прохождение каждого теста оценивается в 2 балла. Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
A Bra Kada Bra
Стандартный вывод
aaabra
abdkr

Решение

В этой задаче проверяется знание некоторых функций Python для работы со строками. В частности, используются: метод `replace()` для удаления всех пробелов, метод `lower()` для смены регистра, метод `join()` для объединения символов в строку. Для хранения уникальных символов в приведенном решении используются множества.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 unique=set()
2 other=''
3 for x in input().replace(' ','').lower():
4     if x in unique:
5         other+=x

```

```
6     else:
7         unique.add(x)
8     print(other)
9     print(''.join(sorted(list(unique))))
```

Задача I.1.2.4. Велосипедисты (20 баллов)

Темы: математика, уравнения, двоичный поиск.

Два велосипедиста выехали одновременно из пункта А по одной дороге с различными скоростями u и v метров в секунду. Через t секунд им вдогонку выехал электромобиль и через некоторое время обогнал одного, а затем и другого велосипедиста. При этом интервал между моментами обгона составил d секунд.

Вы должны написать программу, которая вычислит скорость движения электромобиля.

Формат входных данных

На вход через пробел подаются четыре натуральных числа: u , v , t , d . При этом $u \neq v$; $u, v \leq 50$; $t, d \leq 10000$. Гарантируется, что введенные данные будут таковы, что ответ не превысит 200.

Формат выходных данных

Программа должна вывести одно вещественное число — скорость электромобиля.

Методика проверки и пояснение к тесту

Ответ участника считается верным, если он отличается от ответа жюри не более чем на 10^{-8} .

Программа проверяется на 10 тестах. Прохождение каждого теста оценивается в 2 балла. При этом в первых 5 тестах ответ обязательно будет целым числом. Тест из условия задачи при проверке не используется.

Рассмотрим тест из примера. Утверждается, что для заданных параметров ответом является 12. Проверим это. Можно вычислить, что электромобиль, двигаясь со скоростью 12 м/с, обгонит более медленного велосипедиста на расстоянии 480 метров, а более быстрого на расстоянии в 1200 метров. Действительно, электромобиль преодолет 480 и 1200 метров за 40 и 100 секунд соответственно. Таким образом, интервал между моментами обгона действительно равен 60. Велосипедисты до моментов обгона будут двигаться на 20 секунд дольше, по 60 и 120 секунд соответственно. И, проверив пройденное расстояние $60 \cdot 8 = 480$ и $120 \cdot 10 = 1200$, убедимся, что ответ верен. **Обратите внимание, что это пояснение лишь показывает, как проверить правильность ответа, но не является алгоритмом решения.**

Примеры

Пример №1

Стандартный ввод
10 8 20 60
Стандартный вывод
12.0

Решение

Обозначим скорость электромобиля за x . Тогда до старта электромобиля велосипедист проехал tu метров. Электромобиль догнал велосипедиста через $\frac{tu}{x-u}$ секунд. Предполагая, что второй велосипедист двигался быстрее чем первый, получим уравнение:

$$\frac{tv}{x-v} - \frac{tu}{x-u} = d$$

Уравнение можно решить аналитически, а также тернарным или бинарным поиском, учитывая, что с ростом x интервал между моментами обгона будет сокращаться.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 u,v,t,d=map(int,input().split())
2 if u>v:
3     u,v=v,u
4 d/=t
5 b=(d+1)*v+(d-1)*u
6 print((b+(b*b-4*d*d*u*v)**0.5)/(2*d))

```

Ниже представлено решение бинарным поиском на языке Python 3.

```

1 u,v,t,d=map(int,input().split())
2 if u>v:
3     u,v=v,u
4 left=max(u,v)
5 right=200
6 for i in range(100):
7     mid=(left+right)/2
8     if t*v/(mid-v)-t*u/(mid-u)>d:
9         left=mid
10    else:
11        right=mid
12 print(right)

```

Задача I.1.2.5. Много единиц (20 баллов)

Темы: системы счисления, битовые операции, реализация.

Алиса учится работать с двоичными числами. Она уже поняла, что число в двоичной записи получается в несколько раз длиннее, чем в десятичной. А еще она

поняла, что нули писать дольше, чем единицы. И теперь ее любимые числа — это те, двоичная запись которых содержит как можно больше единиц. Алисе дали задание — выбрать одно произвольное число из заданного закрытого интервала $[a; b]$ и перевести его в двоичную запись. И теперь Алиса просит, чтобы вы написали программу, которая найдет в этом интервале число, двоичная запись которого содержит наибольшее количество единиц. Если таких чисел будет несколько, то Алиса будет рада любому из них.

Формат входных данных

На вход через пробел подаются два натуральных числа a и b . При этом $1 \leq a \leq b \leq 10^{18}$. Обратите внимание, что для хранения таких чисел в программе на C++ вам потребуется тип **long long**. В программе на PascalABC такой тип называется **Int64**.

Формат выходных данных

Программа должна вывести одно целое число из заданного диапазона, двоичная запись которого содержит наибольшее количество единиц. Само число следует выводить в десятичной записи.

Методика проверки и пояснение к тесту

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. При этом в первых пяти тестах $1 \leq a \leq b \leq 1000$. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
150 200
Стандартный вывод
191

Пример №2

Стандартный ввод
1 255
Стандартный вывод
255

Пример №3

Стандартный ввод
127 200
Стандартный вывод
127

Решение

Рассмотрим некоторое число в двоичной записи, которое содержит k единиц. Чтобы получить ближайшее сверху число в записи которого $k + 1$ единица, требуется заменить на единицу самый правый ноль. Таким образом решение задачи состоит в том, чтобы взять число из начала интервала и заменять самые правые нули до тех пор, пока результат не превысит правую границу интервала.

Реализовать такой алгоритм можно, представляя число в виде строки, а также при помощи арифметических или битовых операций.

Пример программы-решения

Ниже представлено поиском на языке Python 3 с использованием строк.

```

1 a,b = map(int,input().split())
2 a=list(bin(a)[2:])
3 b=list(bin(b)[2:])
4 while len(a)<len(b):
5     a=['0']+a
6 for i in range(len(a)-1,-1,-1):
7     a[i]='1'
8     if a>b:
9         a[i]='0'
10        break
11 print(int(''.join(a),2))

```

Ниже представлено поиском на языке Python 3 с использованием арифметических операций.

```

1 a,b = map(int,input().split())
2 p=1
3 while a+p<=b:
4     if (a//p)%2==0:
5         a+=p
6     p*=2
7 print(a)

```

Ниже представлено поиском на языке Python 3 с использованием битовых операций.

```

1 a,b = map(int,input().split())
2 while a | (a+1) <= b:
3     a = a | (a+1)
4 print(a)

```

Третья попытка. Задачи 8–11 класса

Задача I.1.3.1. Урожай (20 баллов)

Темы: задачи для начинающих.

Дядя Саша с сыном Колей копают картошку. Урожай выдался, как всегда, отменным, и они накопили n мешков. Дядя Саша пригнал грузовичок, в который может

поместиться не более a мешков картошки, а в Колин грузовичок поместится не более b мешков. Урожай они хотят поделить поровну. Если количество мешков не будет делиться на 2, то лишний мешок на правах старшего заберет дядя Саша. Вместе с тем, никто не сможет забрать мешков больше, чем поместится в его грузовик. И, конечно же, они не оставят ни одного мешка на поле.

Напишите программу, которая определит, сколько мешков увезет дядя Саша, а сколько Коля.

Формат входных данных

На вход подаются натуральные числа n , a и b по одному числу в строке. Числа не превосходят 1000. Гарантируется, что $n \leq a + b$.

Формат выходных данных

Программа должна вывести в одной строке через пробел два числа — количество мешков, которое увезут дядя Саша и Коля на своих грузовичках.

Методика проверки и пояснение к тестам

Программа проверяется на 10 тестах. Прохождение каждого теста оценивается в 2 балла. Тесты из условия задачи при проверке не используются.

В первом тесте 59 мешков будут разделены почти поровну — 30 мешков дяде Саше и 29 — Коле. Во втором тесте Коля не сможет увезти причитающиеся ему 29 мешков и отдаст лишнее дяде Саше.

Примеры

Пример №1

Стандартный ввод
59
35
40
Стандартный вывод
30 29

Пример №2

Стандартный ввод
59
41
25
Стандартный вывод
34 25

Решение

Данную задачу можно решать различными способами. В предлагаемом варианте решения мешки сначала делятся поровну, а далее, в случае необходимости, перемещаются из одного грузовика в другой.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
1 n=int(input())
2 a=int(input())
3 b=int(input())
4 x, y = (n+1)//2, n//2
5 if x>a:
6     y+=x-a
7     x=a
8 if y>b:
9     x+=y-b
10    y=b
11 print(x,y)
```

Задача I.1.3.2. Скайраннинг (20 баллов)

Темы: задачи для начинающих.

Скайраннингом называется бег в горной местности по неподготовленным трассам, которые обязательно проходят через одну или несколько вершин. Важной характеристикой маршрута в скайраннинге является набор высоты, который равен сумме перепада высот на всех участках подъема. Например, если на маршруте имеется три участка подъема, причем конечная точка каждого участка выше начальной на 200 метров, то набор высоты на маршруте будет равен 600 метров. Кроме того, весь маршрут может быть поделен на высотные зоны. В нашей задаче мы будем рассматривать две высотные зоны: ниже 2000 метров и выше 2000 метров. В этом случае все параметры, в том числе и набор высоты, рассчитываются для разных высотных зон.

Рассмотрим такой пример. Пусть профиль трассы содержит семь точек с высотами 1200, 2300, 2100, 2900, 3100, 1000, 1800 метров. На этой трассе есть четыре участка подъема: (1200; 2300), (2100; 2900), (2900; 3100), (1000;1800). На первом участке подъема 800 метров набирается в первой высотной зоне и 300 метров во второй. На втором и третьем участках набирается по 800 и 200 метров соответственно во второй высотной зоне. На четвертом участке набирается 800 метров в первой зоне. Таким образом в первой высотной зоне набирается 1600 метров, а во второй — 1300 метров.

Ваша задача — написать программу, которая по заданному профилю трассы найдет набор высоты для двух высотных зон: ниже 2000 и выше 2000 метров.

Формат входных данных

На вход в первой строке подается одно натуральное число n — количество точек в профиле высоты. $2 \leq n \leq 100$. Во второй строке через пробел записаны n целых чисел a_1, \dots, a_n , задающих высоту каждой точки. $-416 \leq a_i \leq 8848$.

Формат выходных данных

Программа должна вывести в одной строке через пробел два числа — набор высоты для первой и второй высотной зоны.

Методика проверки

Программа проверяется на 10 тестах. Прохождение каждого теста оценивается в 2 балла. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
7 1200 2300 2100 2900 3100 1000 1800
Стандартный вывод
1600 1300

Пример №2

Стандартный ввод
5 2000 2675 2675 1215 -416
Стандартный вывод
0 675

Решение

В этой задаче необходимо перебрать все участки трассы, на которых происходит подъем, и рассмотреть три случая: весь участок ниже 2000 метров, весь участок выше 2000 метров, участок проходит через высоту в 2000 метров.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 n = int(input())
2 h = list(map(int, input().split()))
3 a1, a2 = 0, 0
4 for i in range(1, n):
5     if h[i] > h[i-1]:
6         if h[i-1] >= 2000:
7             a2 += h[i] - h[i-1]
8         elif h[i] <= 2000:
9             a1 += h[i] - h[i-1]
10        else:
11            a1 += 2000 - h[i-1]
12            a2 += h[i] - 2000
13 print(a1, a2)

```

Задача I.1.3.3. Шоколадка (20 баллов)

Темы: перебор, сортировки, структуры данных.

У Аленки есть шоколадка прямоугольной формы, размером $n \times m$ долек. Аленка разламывает ее на две части по вертикали или по горизонтали и съедает одну из двух частей. Если Аленка чувствует, что не наелась, то она снова может разломить оставшийся кусок на две части и съесть одну из них, и так далее. Всю шоколадку Аленка есть не будет, а оставит про запас, как минимум, одну дольку.

Производителям стало известно о такой привычке девочки и они захотели узнать, какое количество долек может быть съедено при таком алгоритме поедания шоколада. Они хотят, чтобы вы написали программу, которая по известному размеру шоколадки найдет все возможные количества съеденных долек и выведет их в порядке возрастания.

Рассмотрим такой пример. Шоколадка имеет размер 3×3 . Тогда Аленка может разломить ее на две части 3×1 и 3×2 . Таким образом, она сможет съесть 3 или 6 долек и остановиться. Но также Аленка сможет съесть кусок из 6 долек, а от оставшегося отломить 1 или 2 дольки и съесть еще и их. Таким образом, она сможет съесть 7 или 8 долек. Наконец, она сможет съесть кусок 3×1 , а от оставшегося куска 3×2 отломить и съесть еще 2 дольки, тогда количество съеденных долек будет равно 5. Очевидно, что съесть 1, 2 или 4 дольки Аленка не сможет.

Формат входных данных

На вход в одной строке подается два натуральных числа n и m — размеры шоколадки. $1 \leq n, m \leq 100$; $n + m \geq 3$.

Формат выходных данных

Программа должна вывести в одной строке через пробел все числа, являющиеся ответами к задаче. Числа должны выводиться в порядке возрастания без повторений.

Если вы программируете на Python, то убрать перенос строки в функции `print` можно при помощи именованного параметра `end`, например, `print(a, end='')`.

Методика проверки

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
3 3
Стандартный вывод
3 5 6 7 8

Решение

Заметим, что у Алёнки всегда остается один кусок шоколадки прямоугольной формы. Поэтому можно перебрать все прямоугольные области меньшего размера и найти разность между количеством долек во всей шоколадке и в прямоугольниках. Полученные значения могут повторяться, поэтому потребуется отбросить одинаковые числа. Для этого можно использовать множество или булевский массив.

Пример программы-решения

Ниже представлено решение на языке Python 3.

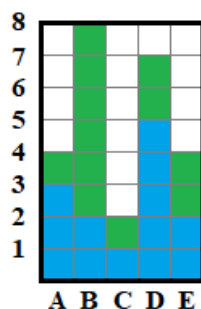
```

1 n, m = map(int, input().split())
2 k=[False]*(n*m)
3 for i in range(1,n+1):
4     for j in range(1,m+1):
5         k[n*m-i*j]=True
6 for i in range(1,n*m):
7     if k[i]:
8         print(i, end=' ')

```

Задача I.1.3.4. Стекочная гистограмма (20 баллов)

Темы: реализация, структуры данных.



По определению Википедии, стекочная гистограмма отображает зоны, представляющие свойства одной категории, друг поверх друга. Имеется n категорий с двумя свойствами. Требуется нарисовать стекочную гистограмму без подписей в консоли с использованием символов «.*#», как показано в примере.

Формат входных данных

В первой строке на вход подается одно натуральное число n — количество категорий в гистограмме. $2 \leq n \leq 50$. Во второй строке через пробел записано n натуральных чисел a_1, \dots, a_n — значения первого свойства для каждой категории. В третьей строке аналогично записаны натуральные числа b_1, \dots, b_n — значения второго свойства. $1 \leq a_i, b_i \leq 20$.

Формат выходных данных

Требуется вывести в консоль текстовое представление гистограммы. Текст должен содержать ровно $\max(a_i + b_i)$ строк, каждая строка должна содержать ровно n символов и завершаться переводом строки. Если рассматривать текст по столбцам, то i -тый столбец должен содержать снизу ровно a_i решеток, далее b_i звездочек, а еще выше — точки.

Методика проверки и пояснение к тесту

Программа проверяется на 5 тестах. Прохождение каждого теста оценивается в 4 балла. Тест из условия задачи при проверке не используется.

На картинке изображена гистограмма из примера. Синие квадратики соответствуют решеткам, зеленые — звездочкам, белые — точкам. Количество строк равно высоте самого высокого столбика.

Примеры

Пример №1

Стандартный ввод
5
3 2 1 5 2
1 6 1 2 2
Стандартный вывод
.*. . .
.*. *.
.*. *.
.*. #.
** . #*
#* . #*
####
#####

Решение

Для решения этой задачи можно создать двумерный символьный массив и заполнить его требуемыми символами. Строки надо будет выводить на экран в обратном порядке. Можно обойтись и без двумерного массива. Для вывода можно использовать два вложенных цикла и определять вид символа по номеру строки и столбца. Чтобы определить количество строк, потребуется найти максимум из сумм значений по категориям.

Пример программы-решения

Ниже представлено решение на языке Python 3 с двумерным массивом.

```
1 n=int(input())
2 a=list(map(int,input().split()))
```

```

3 b=list(map(int,input().split()))
4 h=max([a[i]+b[i] for i in range(n)])
5 s=[['.']*n for i in range(h)]
6 for i in range(n):
7     for j in range(a[i]):
8         s[j][i]='#'
9     for j in range(a[i],a[i]+b[i]):
10        s[j][i]='*'
11 for x in reversed(s):
12     print(''.join(x))

```

Ниже представлено решение на языке Python 3 без двумерного массива.

```

1 n=int(input())
2 a=list(map(int,input().split()))
3 b=list(map(int,input().split()))
4 h=max([a[i]+b[i] for i in range(n)])
5 for j in range(h):
6     for i in range(n):
7         if h-j <= a[i]:
8             print('#',end='')
9         elif h-j <= a[i]+b[i]:
10            print('*',end='')
11        else:
12            print('.',end='')
13    print()

```

Задача 1.1.3.5. Префиксный код (20 баллов)

Темы: математика, строки, перебор, структуры данных.

Префиксное кодирование является очень распространенным способом сжатия информации для ее хранения и передачи. Каждый символ кодируется некоторым кодовым словом — строкой из нулей и единиц. Кодовые слова могут иметь различную длину, но при этом должно выполняться следующее свойство: ни одно кодовое слово не начинается с другого кодового слова. Например, множество кодовых слов $\{00, 011, 1000, 11\}$ подходит для префиксного кодирования, а $\{00, 101, 11, 1010\}$ — нет, так как 101 является началом 1010.

От вас требуется написать программу, которая найдет, какое максимальное количество кодовых слов заданной длины m можно добавить к некоторому уже построенному множеству слов для префиксного кодирования. Например $m = 4$, и уже построено множество $\{10, 001, 000, 0111, 11111, 11000, 111101\}$. Без нарушения условия префиксного кодирования можно добавить следующие слова длины 4: $\{0100, 0101, 0110, 1101, 1110\}$. Таким образом, ответ равен 5.

Формат входных данных

В первой строке на вход подается два натуральных числа n и m — количество уже известных кодовых слов и длина новых кодовых слов. $m \leq 60$. Во второй строке через пробел записано n кодовых слов. Все слова состоят из нулей и единиц и удовлетворяют условию префиксного кодирования. Суммарная длина всех слов не превосходит 10^6 .

Формат выходных данных

Требуется вывести одно целое число — ответ к задаче. Ответ может быть равным нулю.

Методика проверки

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. При этом в 3 тестах $m \leq 10$ и $n \leq 100$, еще в 3 тестах длина всех заданных слов не превосходит m , еще в 4 тестах длина всех заданных слов не меньше чем m . Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
7 4 10 001 000 0111 11111 11000 111101
Стандартный вывод
5

Решение

Сразу отметим, что существует 2^m различных кодовых слов длины m . Каждое кодовое слово, ранее добавленное в код, не позволит использовать некоторое количество возможных слов длины m . Так, если некоторое уже добавленное слово a имеет длину k , меньшую чем m , то любое слово, полученное приписыванием к a справа $m - k$ символов, будет недоступно. Существует 2^{m-k} способов дописать справа к a некоторую последовательность из нулей и единиц, поэтому такую величину надо будет вычесть из ответа.

Если кодовое слово имеет длину большую m , то его префикс длины m также нельзя использовать. Однако различные слова могут иметь одинаковые префиксы, поэтому среди них надо найти все различные, например, при помощи множеств.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 n,m=map(int,input().split())
2 ans=2**m
3 wset=set()
4 for x in input().split():
5     if len(x)>m:
6         wset.add(x[:m])
7     else:
8         ans-=2**(m-len(x))
9 print(ans-len(wset))

```

Четвертая попытка. Задачи 8–11 класса

Задача I.1.4.1. Кросс нации (20 баллов)

Темы: математика, задачи для начинающих.

Некоторые измеряют скорость в километрах, пройденных за час. А вот люди, занимающиеся бегом, измеряют темп бега в секундах на километр, то есть указывают количество минут и секунд, за которые они пробегают ровно 1 километр. Например, скорость в 12,5 км/ч соответствует темпу бега в 4 минуты 48 секунд.

Ваша задача — написать программу, которая определит темп бега по скорости.

Формат входных данных

На вход подается единственное число u — скорость в километрах за час. Скорость является вещественным положительным числом не более чем с двумя знаками после точки. $5 \leq u \leq 50$.

Формат выходных данных

Вывести через пробел два целых числа — время в минутах и секундах, за которое бегун пробегает 1 километр. Секунды должны быть округлены до ближайшего целого числа по стандартным правилам.

Методика проверки

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
12.5
Стандартный вывод
4 48

Пример №2

Стандартный ввод
15.03
Стандартный вывод
4 0

Пример №3

Стандартный ввод
15.04
Стандартный вывод
3 59

Решение

В одном часе 3600 секунд. Поэтому, если бегун пробегает за 3600 секунд a километров, то один километр он пробежит за $\frac{3600}{a}$ секунд. Это число надо округлить и привести к целому типу, далее выделить из него минуты и секунды, используя деление и остаток от деления на 60.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
1 a=float(input())
2 a=int(round(3600/a))
3 print(a//60,s%60)
```

Задача I.1.4.2. Говорящий конь (20 баллов)

Темы: задачи для начинающих, реализация.

Говорящий конь Юлий шел по дороге и встретил трех богатырей.

- *Здравствуй, богатырь номер 1,— сказал говорящий конь Юлий.*
- *Здравствуй, говорящий конь Юлий,— сказал богатырь номер 1.*
- *Здравствуй, богатырь номер 2,— сказал говорящий конь Юлий.*
- *Здравствуй, говорящий конь Юлий,— сказал богатырь номер 2.*
- *Здравствуй, богатырь номер 3,— сказал говорящий конь Юлий.*
- *Здравствуй, говорящий конь Юлий,— сказал богатырь номер 3.*
- *Здравствуй, лошадь богатыря номер 1,— сказал говорящий конь Юлий.*
- *Здравствуй, говорящий конь Юлий,— сказала лошадь богатыря номер 1.*
- *Здравствуй, лошадь богатыря номер 2,— сказал говорящий конь Юлий.*
- *Здравствуй, говорящий конь Юлий,— сказала лошадь богатыря номер 2.*
- *Здравствуй, лошадь богатыря номер 3,— сказал говорящий конь Юлий.*
- *Здравствуй, говорящий конь Юлий,— сказала лошадь богатыря номер 3.*
- *До свиданья, богатырь номер 1,— сказал говорящий конь Юлий.*
- *До свиданья, говорящий конь Юлий,— сказал богатырь номер 1.*
- *До свиданья, богатырь номер 2,— сказал говорящий конь Юлий.*
- *До свиданья, говорящий конь Юлий,— сказал богатырь номер 2.*
- *До свиданья, богатырь номер 3,— сказал говорящий конь Юлий.*

- До свиданья, говорящий конь Юлий, – сказал богатырь номер 3.
- До свиданья, лошадь богатыря номер 1, – сказал говорящий конь Юлий.
- До свиданья, говорящий конь Юлий, – сказала лошадь богатыря номер 1.
- До свиданья, лошадь богатыря номер 2, – сказал говорящий конь Юлий.
- До свиданья, говорящий конь Юлий, – сказала лошадь богатыря номер 2.
- До свиданья, лошадь богатыря номер 3, – сказал говорящий конь Юлий.
- До свиданья, говорящий конь Юлий, – сказала лошадь богатыря номер 3.

И говорящий конь Юлий пошел по дороге дальше.

Маленькому Ванечке очень нравится эта сказка и он любит слушать ее в разных вариантах. Ему особенно нравится вариант, в котором говорящий конь Юлий встречает на дороге 40 разбойников, но почему-то усталые взрослые не хотят ее рассказывать.

Напишите программу, которая по номеру предложения сказки о встрече говорящего коня Юлия с 40 разбойниками будет выводить это предложение. Поскольку родители считают, что Ванечка с самого юного возраста должен изучать иностранные языки, от вас требуется вывести ответ по-английски.

Формат входных данных

Одно натуральное число n — номер предложения сказки. $1 \leq n \leq 322$.

Формат выходных данных

Вывести требуемое предложение сказки в одной строке. Слова должны быть разделены ровно одним пробелом. Перед знаками препинания пробел не ставится. Перевод всех предложений и формат вывода смотрите в примерах.

Методика проверки

Программа проверяется на 50 тестах. Прохождение каждого теста оценивается в 0,4 балла. **Первые 10 контрольных тестов совпадают с тестами из условия задачи.**

Примеры

Пример №1

Стандартный ввод
1
Стандартный вывод
Talking horse Julius was walking along the road and met 40 robbers.

Пример №2

Стандартный ввод
2
Стандартный вывод
- Hello, robber number 1,- said talking horse Julius.

Пример №3

Стандартный ввод
3
Стандартный вывод
- Hello, talking horse Julius,- said robber number 1.

Пример №4

Стандартный ввод
82
Стандартный вывод
- Hello, horse of robber number 1,- said talking horse Julius.

Пример №5

Стандартный ввод
83
Стандартный вывод
- Hello, talking horse Julius,- said horse of robber number 1.

Пример №6

Стандартный ввод
162
Стандартный вывод
- Goodbye, robber number 1,- said talking horse Julius.

Пример №7

Стандартный ввод
163
Стандартный вывод
- Goodbye, talking horse Julius,- said robber number 1.

Пример №8

Стандартный ввод
242
Стандартный вывод
- Goodbye, horse of robber number 1,- said talking horse Julius.

Пример №9

Стандартный ввод
243
Стандартный вывод
- Goodbye, talking horse Julius,- said horse of robber number 1.

Пример №10

Стандартный ввод
322
Стандартный вывод
And talking horse Julius went on along the road.

Решение

Текст сказки состоит из 10 различных предложений. Каждое предложение встречается в определенном диапазоне номеров на четных или нечетных позициях. Исходя из этого, можно записать программу через одну инструкцию ветвления с 10 вариантами работы.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1  n=int(input())
2  if n==1:
3      print('Talking horse Julius was walking along the road and met 40 robbers.')
4  elif n<82 and n%2==0:
5      print('- Hello, robber number '+str(n//2)+',- said talking horse Julius.')
6  elif n<82:
7      print('- Hello, talking horse Julius,- said robber number '+str(n//2)+'.')
8  elif n<162 and n%2==0:
9      print('- Hello, horse of robber number '+str((n-80)//2)+',- said talking horse
   ↪ Julius.')
10 elif n<162:
11     print('- Hello, talking horse Julius,- said horse of robber number
   ↪ '+str((n-80)//2)+'.')
12 elif n<242 and n%2==0:
13     print('- Goodbye, robber number '+str((n-160)//2)+',- said talking horse
   ↪ Julius.')
14 elif n<242:
15     print('- Goodbye, talking horse Julius,- said robber number
   ↪ '+str((n-160)//2)+'.')
16 elif n<322 and n%2==0:
17     print('- Goodbye, horse of robber number '+str((n-240)//2)+',- said talking
   ↪ horse Julius.')
18 elif n<322:
19     print('- Goodbye, talking horse Julius,- said horse of robber number
   ↪ '+str((n-240)//2)+'.')
20 else:
21     print('And talking horse Julius went on along the road.')
```

Задача I.1.4.3. Нумерация (20 баллов)

Темы: задачи для начинающих, реализация, структуры данных.

Власти города Байтленда решили построить новый микрорайон, который будет состоять из одной длинной улицы. Процесс постройки домов будет выглядеть следующим образом. Сначала будет построен самый первый дом, который получит номер 0. (Разумеется, в Байтленде все нумерации начинаются с нуля.) Далее все дома будут пристраиваться слева или справа от существующей застройки. Дома будут получать номера в порядке их ввода в эксплуатацию.

Рассмотрим пример. Пусть дом номер 1 был построен слева от дома номер 0. Тогда нумерация домов слева направо будет выглядеть как 1 0. Далее был построен дом номер 2 слева от существующих. Теперь дома на улице будут иметь номера 2 1 0. Далее был построен дом номер 3 справа от существующих. Теперь на улице будут стоять дома с номерами 2 1 0 3. Наконец, если дом номер 4 будет построен слева, а дома с номерами 5 и 6 справа, то последовательность номеров домов превратится в 4 2 1 0 3 5 6.

На улице построили n домов с номерами от 0 до $n - 1$. Для каждого дома с ненулевым номером нам стало известно с какой стороны от существующих он был построен. Теперь ваша задача — написать программу, которая перечислит номера домов в порядке движения по улице слева направо.

Формат входных данных

В первой строке на вход подается число n — количество построенных домов. $2 \leq n \leq 400000$. Во второй строке записана последовательность из $n - 1$ символов «L» или «R» (без кавычек). Символ «L» в i -той позиции означает, что дом с номером i был построен слева от предыдущих, а символ «R» — справа.

Формат выходных данных

Вывести через пробел в одной строке последовательность из n чисел — нумерацию домов после завершения строительства.

Методика проверки и описание тестов

Программа проверяется на 10 тестах. Прохождение каждого теста оценивается в 2 балла. Тест из условия задачи при проверке не используется.

В первых пяти тестах $n \leq 100$.

Примеры

Пример №1

Стандартный ввод
7 LLRLRR
Стандартный вывод
4 2 1 0 3 5 6

Решение

По условию задачи требуется составить последовательность чисел, добавляя их слева или справа к существующим. Для решения можно использовать структуру данных дек или изменить порядок выполнения команд добавления элементов. Требуемую последовательность можно получить, выполнив два прохода по строке, задающей команды. Во время первого прохода будем добавлять числа слева от нуля. Чтобы получить нужный порядок, по строке потребуется пройти справа налево. Далее необходимо вывести ноль и сделать второй проход для вывода чисел справа от нуля.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1  n=int(input())
2  x=input()
3  ans=''
4  for i in range(n-1,0,-1):
5      if x[i-1]=='L':
6          print(i,end=' ')
7  print(0,end=' ')
8  for i in range(1,n):
9      if x[i-1]=='R':
10         print(i,end=' ')

```

Задача I.1.4.4. Космическая связь (20 баллов)

Темы: реализация.

В этой задаче от вас потребуется написать программу для составления расписания сеансов связи со спутником. Каждый сеанс заключается в обмене короткими сообщениями, поэтому мы считаем, что он происходит мгновенно. Поскольку ресурсы оборудования ограничены, следует стремиться к минимизации количества сеансов. Вместе с тем, интервал времени между сеансами не должен превышать d миллисекунд. Кроме того, существуют промежутки времени, в течении которых связь невозможна. При этом на границах промежутка мгновенный сеанс связи возможен. Расписание составляется на t миллисекунд. Первый сеанс должен обязательно состояться в момент 0, а последний — в момент t .

Рассмотрим пример. Пусть $t = 100$, $d = 20$ и задано 3 промежутка недоступности связи: (5;25), (27;40), (75;90). Тогда потребуется восемь сеансов связи, которые можно провести в моменты времени 0, 5, 25, 45, 65, 75, 90, 100. Конкретное расписание может быть другим, но в любом случае количество сеансов не может быть меньше 8.

Ваша программа должна по имеющейся информации найти минимальное возможное количество сеансов связи.

Формат входных данных

В строке 1 через пробел записаны 3 натуральных числа n , d и t — количество интервалов недоступности связи, максимальный интервал между сеансами

и время, на которое составляется расписание. $n \leq 200000$, $d, t \leq 10^9$. Далее в n строках заданы по два целых неотрицательных числа a_i и b_i — начало и конец каждого интервала недоступности связи. $b_i - a_i \leq d$. Интервалы недоступности связи не пересекаются, каждый следующий интервал начинается строго после окончания предыдущего. $0 \leq a_1 < b_1 < a_2 < b_2 < \dots < a_n < b_n \leq t$.

Формат выходных данных

Вывести число — количество сеансов связи в графике.

Методика проверки

Программа проверяется на 25 тестах. Прохождение каждого теста оценивается в 0,8 балла. Тест из условия задачи при проверке не используется.

В первых 10 тестах $t \leq 1000$. В следующих 10 тестах $t \leq 10^6$.

Примеры

Пример №1

Стандартный ввод
3 20 100
5 25
27 40
75 90
Стандартный вывод
8

Решение

Для решения этой задачи можно использовать идею жадного алгоритма. Каждый следующий сеанс связи будем проводить как можно позже. Если время предыдущего сеанса было равно t , то следующий сеанс будем проводить в момент $t + d$. Однако, если $t + d$ попадет внутрь некоторого недоступного интервала $[a; b]$, то время сеанса потребуется сместить до левой границы интервала.

Для получения полных баллов за решение требуется составить алгоритм линейной сложности относительно количества недоступных для связи интервалов. Таким образом, на каждом шаге цикла будет рассматриваться один интервал, который будем называть текущим.

Решение этой задачи требует аккуратной реализации. Инвариантом предлагаемого решения является утверждение о том, что t никогда не попадет внутрь некоторого недоступного интервала, и $t + d$ будет больше, чем правая граница текущего интервала, что гарантирует правильность работы на следующих итерациях цикла.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
1 n,d,s=map(int,input().split())
2 m=1
3 t=0
4 for i in range(n):
5     a,b=map(int,input().split())
6     k=(a-t)//d
7     t+=d*k
8     m+=k
9     if t+d<b:
10        t=a
11        m+=1
12 print(m+(s-t+d-1)//d)
```

Задача I.1.4.5. Простая задача (20 баллов)

Темы: реализация.

В этой задаче не будет длинного условия и простого решения. Все будет наоборот. Требуется провести непрерывную линию произвольного вида из точки с координатами (x_1, y_1) в точку с координатами (x_2, y_2) так, чтобы минимизировать количество точек на этой линии, в которых хотя бы одна координата является целым числом. Ответом к задаче будет являться количество таких точек. Если начальная или конечная точка линии будет иметь хотя бы одну целочисленную координату, то ее тоже надо учитывать.

Формат входных данных

Каждый тест в этой задаче будет содержать n запросов. $1 \leq n \leq 100$. Натуральное число n будет записано в первой строке. Далее в n строках записаны запросы. Каждый запрос располагается в отдельной строке и состоит из четырех чисел x_1, y_1, x_2, y_2 , которые задают координаты двух точек. Точки не совпадают. Координаты могут быть целыми или вещественными числами не более чем с 2 знаками после точки. Координаты не превосходят 10^9 по абсолютной величине. Если координата является целым числом, то ее запись не содержит десятичной точки.

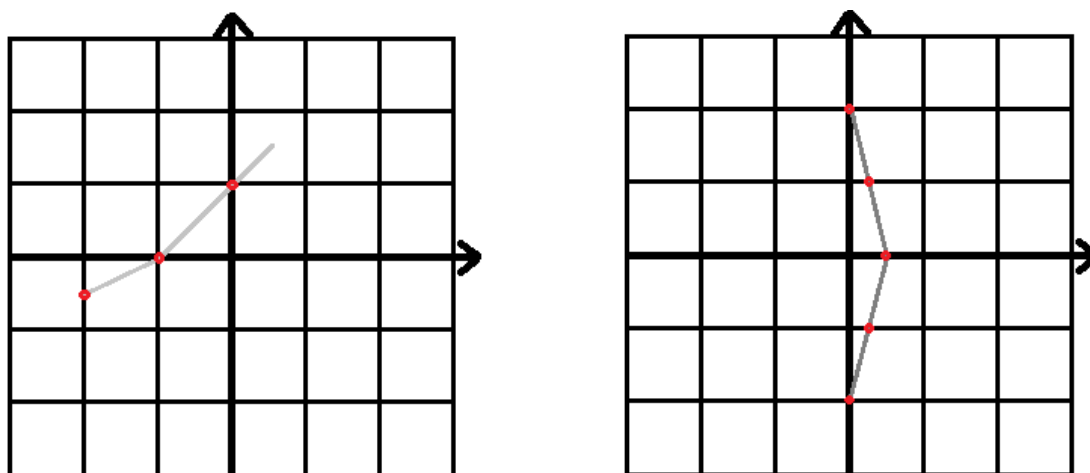
Формат выходных данных

Требуется вывести ответы на запросы по одному ответу в каждой строке.

Методика проверки и пояснение к тесту

Программа проверяется на 10 тестах. Прохождение каждого теста оценивается в 2 балла. Тест из условия задачи при проверке не используется.

Следующие рисунки поясняют ответ к тесту. Обратите внимание, что никакой фрагмент линии не может лежать на сетке, так как в этом случае количество точек с целочисленной координатой будет бесконечно большим.



Примеры

Пример №1

Стандартный ввод
2 -2 -0.5 0.5 1.5 0 -2 0 2
Стандартный вывод
3 5

Решение

Для решения задачи потребуется найти количество горизонтальных и вертикальных прямых целочисленной сетки, которые будут пересекаться нарисованной линией. Поскольку одна вертикальная и одна горизонтальная прямая могут быть пересечены в одной точке, из полученных значений надо будет взять максимальное. Далее надо будет проверить, являются ли координаты концов линии целыми числами, и, в случае необходимости, учесть эти точки в ответе.

Для поиска количества точек пересечения в предлагаемом решении используется функция *bw*. Сначала границы диапазона смещаются на некоторую небольшую величину, чтобы их координаты перестали быть целочисленными. Далее верхняя граница диапазона округляется вниз, а нижняя вверх, после чего вычисляется количество целых чисел в диапазоне.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 import math
2 def bw(a,b):
3     if a>b:
4         a,b=b,a

```

```
5     return max(0,math.floor(b-0.001)-math.ceil(a+0.001)+1)
6
7 n=int(input())
8 for i in range(n):
9     x1,y1,x2,y2=map(float,input().split());
10    print(max(bw(x1,x2),bw(y1,y2))+
11 int((x1==round(x1) or y1==round(y1))+
12 int((x2==round(x2) or y2==round(y2))))))
```

Задачи первого этапа. Математика

Первая попытка. Задачи 8–9 класса

Задача I.2.1.1. (15 баллов)

Темы: числа, сравнения, логика.

Пусть x — некоторое натуральное число. Среди утверждений:

- « $2x$ больше 70»;
- « x меньше 100»;
- « $3x$ больше 25»;
- « x не меньше 10»;
- « x больше 5»;

три верных и два неверных. Чему равно x ?

Решение

Если $x \leq 8$, то как минимум три утверждения (первое, третье и четвертое) неверны, поэтому такие значения x не удовлетворяют условию.

Если $10 \leq x < 100$, то верны как минимум четыре утверждения (все, кроме первого), поэтому такие x также не удовлетворяют условию.

Если $x \geq 100$, то верны все утверждения, кроме второго, и такие x также не подходят.

Остается единственный вариант $x = 9$, при этом верны второе, третье и пятое утверждения и неверны первое и четвертое, т. е. условия задачи выполнены.

Ответ: 9.

Задача I.2.1.2. (15 баллов)

Темы: алгебра, преобразования.

Известно, что $\frac{2x+3y}{2x-3y} + \frac{2x-3y}{2x+3y} = \frac{50}{7}$. Чему равно значение выражения $\frac{x^2+2y^2}{x^2-2y^2}$?

Решение

Приведем исходное уравнение к общему знаменателю:

$$\frac{(2x+3y)^2 + (2x-3y)^2}{(2x-3y)(2x+3y)} = \frac{50}{7},$$

откуда $7(8x^2 + 18y^2) = 50(4x^2 - 9y^2)$, или после упрощения $x^2 = 4y^2$. Подставим в искомое выражение: $\frac{x^2 + 2y^2}{x^2 - 2y^2} = \frac{6y^2}{2y^2} = 3$.

Ответ: 3.

Задача I.2.1.3. (20 баллов)

Темы: геометрия, периметр.

Кукурузное поле имеет форму прямоугольника, разделенного на девять меньших прямоугольников. Агроном Александр измерил периметры некоторых участков (в метрах) и отметил их на плане (см. рисунок). Помогите Александру без дополнительных измерений узнать периметр участка, обозначенный на схеме за x . Ответ укажите в метрах.

	600	1000
700		x
800	500	

Решение

Легко заметить, что периметры участков, отмеченные на схеме **жирным**, равны периметру всего поля, и то же касается периметров, отмеченных курсивом. Из уравнения $700 + 500 + 1000 = 600 + x + 800$ находим $x = 800$.

	<i>600</i>	1000
700		x
<i>800</i>	500	

Ответ: 800.

Задача I.2.1.4. (20 баллов)

Темы: числа, делимость.

Сергей перемножил шесть последовательных натуральных нечетных чисел и записал результат на бумаге, согнул листочек и передал его Саше. Саша долго носил его в кармане, а когда развернул, то увидел число $10530*075$ («*» заменяет одну цифру, стершуюся на сгибе). На какую цифру надо Саше заменить «*», чтобы получился верный результат без повторения вычислений?

Решение

Среди шести последовательных нечетных натуральных чисел есть два числа, кратные 3. Значит, искомое произведение делится на 9, а, значит, и сумма цифр должна делиться на 9. Сумма оставшихся цифр числа равна 21, следовательно, «*» = 6.

Ответ: 6.

Задача I.2.1.5. (30 баллов)

Темы: комбинаторика, процессы.

Ослик Иа собрал кучу из 50 шишек и решил разложить их по кучкам следующим образом. Сначала кучу он произвольно разбивает на две кучи. Затем любую из имеющихся куч он снова разбивает на две кучи, и так далее до тех пор, пока каждая кучка не будет состоять из одной шишки. Чтобы еще заодно тренироваться в арифметике, Иа при каждом разбиении какой-либо кучи на две записывает произведение количеств шишек в двух получившихся кучках. Чему может быть равна сумма всех записанных чисел?

Решение

Изобразим шишки точками и соединим каждую пару точек отрезком. Получим $\frac{50 \cdot (50 - 1)}{2} = 1225$ отрезков. При каждом разбиении одной кучи шишек на две будем стирать все отрезки, соединяющие точки, соответствующие шишкам, оказавшимся в разных кучах. Пусть на некотором шаге мы разбили шишки одной из уже имевшихся куч на две кучки по x и y шишек. Тогда мы стираем xy отрезков. Это же число мы записываем. Таким образом, сумма записанных чисел — это количество всех стертых отрезков. Так как изначально было 1225 отрезков, а в итоге все отрезки стерты, то общее количество стертых отрезков равно 1225.

Ответ: 1225.

Первая попытка. Задачи 10–11 класса**Задача I.2.2.1. (15 баллов)**

Темы: алгебра, преобразования, уравнения.

Известно, что $\frac{7x + 3y}{x + 5y} + \frac{3x - 2y}{2x + y} = 4$. Чему может равняться значение выражения $t = \frac{x^2 + 2y^2}{x^2 - 2y^2}$? Если вариантов несколько, запишите в ответ наименьшее возможное значение t .

Решение

Приведем исходное уравнение к общему знаменателю:

$$\frac{(7x + 3y)(2x + y) + (3x - y)(x + 5y)}{(x + 5y)(2x + y)} = 4,$$

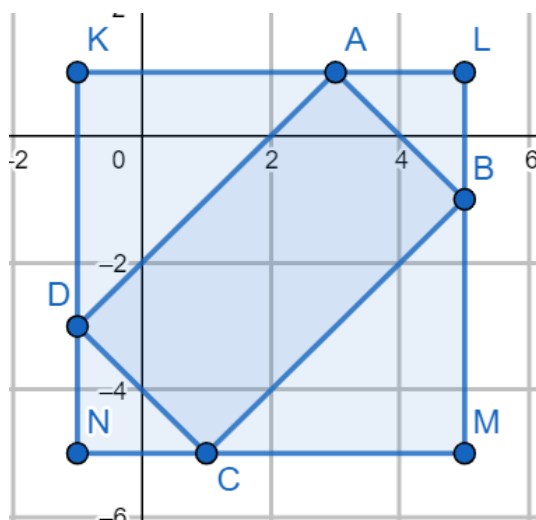
откуда после упрощения $x^2 - 2xy - 3y^2 = 0$. Корнями уравнения являются $x = 3y$ и $x = -y$. В первом случае $t = \frac{11}{7}$, во втором $t = -3$. В ответ записываем наименьшее значение $t = -3$.

Ответ: -3 .

Задача I.2.2.2. (15 баллов)

Темы: диофантовы уравнения, площадь.

Найдите площадь выпуклого многоугольника, координаты $(x; y)$ вершин которого являются целыми числами и удовлетворяют уравнению $xy - 2y + 2x = 7$.

Решение

Преобразуем уравнение к виду $(x - 2)(y + 2) = 3$. Поскольку x и y — целые числа, то возможны следующие четыре варианта: 1) $\begin{cases} x - 2 = 1 \\ y + 2 = 3 \end{cases}$; 2) $\begin{cases} x - 2 = 3 \\ y + 2 = 1 \end{cases}$; 3) $\begin{cases} x - 2 = -1 \\ y + 2 = -3 \end{cases}$; 4) $\begin{cases} x - 2 = -3 \\ y + 2 = -1 \end{cases}$. Решениями этих систем являются пары чисел, которые соответствуют точкам на плоскости $A(3; 1)$; $B(5; -1)$; $C(1; -5)$; $D(-1; -3)$. Дополнениями четырехугольника $ABCD$ до прямоугольника $KLMN$ являются равнобедренные прямоугольные треугольники. Поэтому $ABCD$ — прямоугольник. Его стороны равны $2\sqrt{2}$ и $4\sqrt{2}$, площадь равна 16.

Ответ: 16.

Задача I.2.2.3. (20 баллов)*Темы: числа, делимость.*

Сумма трех натуральных чисел равна 2021. Если в двух из них зачеркнуть цифры в разряде единиц и сложить полученные числа, то получится третье число. Найдите третье число.

Решение

Обозначим искомые числа A , B и C . Пусть $A = 10a + x$, $B = 10b + y$, где x и y — цифры единиц чисел A и B соответственно. Тогда:

$$\begin{cases} A + B + C = 2021 \\ a + b = C \end{cases} \Leftrightarrow \begin{cases} 10a + x + 10b + y + C = 2021 \\ a + b = C \end{cases} \Leftrightarrow \begin{cases} 11C = 2021 - (x + y) \\ a + b = C \end{cases}$$

Поскольку $(x + y) \in [0; 18]$, то для делимости $2021 - (x + y)$ на 11 имеется ровно одна возможность $x + y = 8$. При этом $C = 183$.

Ответ: 183.

Задача I.2.2.4. (20 баллов)*Темы: вероятность.*

У Сергея есть сундук с карточками, на которых написаны все семизначные числа (по одному на каждой карточке). Сергей называет число *зеркальным*, если оно одинаково читается слева направо и справа налево. Сергей случайным образом достает из сундука одну карточку. Какова вероятность, что Сергей достанет карточку, на которой записано зеркальное число, делящееся на 3? (Ответ запишите в виде десятичной дроби с точностью до 6 знаков после запятой).

Решение

Всего семизначных чисел 9 000 000.

Разобьем зеркальное число на центральную цифру Π и две группы по 3 цифры. Левая группа образует трехзначное число \mathcal{L} , правая получается записью цифр левого в обратном порядке. Число N делится на 3 если его сумма цифр $s(N)$ делится на 3. Так как у правой и левой группы суммы цифр одинаковы, то общая сумма цифр равна $2s(\mathcal{L}) + \Pi$. Она делится на 3 тогда и только тогда, когда $s(\mathcal{L})$ и Π дают одинаковые остатки при делении на 3.

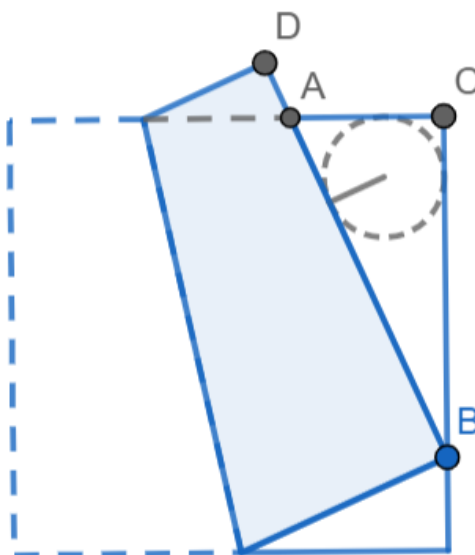
Всего есть 900 трехзначных чисел. Разобьем их на тройки так, чтобы в каждой встречались по разу остатки 0, 1, 2 от деления суммы цифр на 3. Самый простой пример разбиения — на тройки последовательных: (100, 101, 102), ..., (997, 998, 999). В каждой такой тройке числа дают разные остатки, тогда, по свойству равноостаточности, и суммы цифр дают разные остатки. Каждый остаток (0, 1, 2) имеет по $900 : 3 = 300$ чисел. Итак, для каждой цифры Π есть 300 вариантов \mathcal{L} , дающих кратное 3 зеркальное число. Значит, всего искомым чисел 3000, а вероятность равна $3000/9000000 \approx 0,000333$.

Ответ: 0,000333.

Задача I.2.2.5. (30 баллов)

Темы: планиметрия.

Вася перегнул квадратный лист бумаги со стороной 20 см так, как показано на рисунке. Найдите радиус окружности, вписанной в треугольник ABC , если известно, что $AD : AB = 1 : 14$. Ответ (в см) округлите до двух знаков после запятой. Дробную часть от целой отделяйте запятой. В ответ вводите только число, без указания единиц измерения.



Решение

1. Перегибание листа бумаги с геометрической точки зрения означает симметрию относительно прямой GF . При этом точка E переходит в D , а точка K — в B . Следовательно, $EF = DF$, а $KG = BG$. По свойству равнобедренного треугольника $\angle KVG = \angle GKB$.
2. Опустим перпендикуляр KH на прямую BD . Тогда $\angle HKB = \angle GKB$ как накрест лежащие при параллельных прямых KH и GB и секущей BK .
3. Из пп. 1 и 2 следует равенство углов HKB и LKB . Тогда прямоугольные треугольники KLB и KHB равны по общей гипотенузе и равным острым углам. Отсюда $KH = KL$ и $HВ = LB$.
4. Прямоугольные треугольники KEA и KHA равны по общей гипотенузе KA и равным катетам KE и KH ($KE = KL$ как стороны квадрата, а $KH = KL$ из п. 3). Следовательно, $EA = HA$.
5. Чтобы найти радиус вписанной окружности прямоугольного треугольника ABC , воспользуемся формулой $r = \frac{a + b - c}{2}$, где a и b — катеты, c — гипотенуза:

$$\begin{aligned} r &= \frac{1}{2}(AC + BC - AB) = \frac{1}{2}(EC - EA + CL - LB - AH - BH) = \\ &= EC - (AH + HB) = DA. \end{aligned}$$

6. Поскольку $AD : AB = 1 : 14$, имеем: $r = AD = \frac{1}{15}AB = \frac{1}{15} \cdot 20 \approx 1,33$ (см).

Ответ: 1,33.

Вторая попытка. Задачи 8–9 класса

Задача I.2.3.1. (15 баллов)

Темы: уравнения, текстовая задача.

В 18 веке член Петербургской академии наук медик Иосий Вейтбрехт предложил свою шкалу измерения температуры. Нулевое значение соответствовало температуре кипения воды (100 градусов по используемой сейчас шкале Цельсия), температуре замерзания воды (0 градусов по шкале Цельсия) соответствовало 150 градусов по шкале Вейтбрехта. При какой температуре термометр со шкалой Вейтбрехта и термометр со шкалой Цельсия покажут одинаковые значения? Зависимость одной шкалы температуры от другой считайте линейной.

Решение

Температура по Вейтбрехту зависит от температуры по Цельсию по закону $t_B = -1,5t_C + 150$. Если $t_B = t_C$, то это будет при $t_C = -1,5t_C + 150$, то есть при $t_B = t_C = 60$.

Ответ: 60.

Задача I.2.3.2. (15 баллов)

Темы: геометрия, площадь.

План сада имеет форму прямоугольника, разделенного дорожками на девять прямоугольных частей. Садовник Александр знает площади некоторых из частей (они указаны на рисунке). Помогите Александру, не выполняя измерений, узнать площадь левой верхней части сада. Считайте ширину дорожек нулевой.

?	240	
	112	84
108		144

Решение

Заметим, что если прямоугольник разрезан горизонтальным и вертикальным отрезками на четыре прямоугольника, то произведения площадей противоположных частей равны. Тогда последовательно находим площади нижней средней части ($112 \cdot 144 : 84 = 192$), средней левой части ($108 \cdot 112 : 192 = 63$) и, наконец, верхней левой части ($63 \cdot 240 : 112 = 135$).

135	240	
63	112	84
108	192	144

Ответ: 135.

Задача I.2.3.3. (20 баллов)

Темы: логика.

На юбилее Солнечного Города должны выступить Знайка, Винтик, Шпунтик, Пилюлькин и Незнайка. Сколькими способами их можно расположить в списке выступающих при условии, что до Незнайки должны выступать Винтик и Шпунтик (в каком-то порядке)?

Решение

Обозначим выступающих буквами З, В, Ш, П и Н. Н мог выступать третьим, четвертым или пятым. Разберем эти варианты.

1. Если Н выступает третьим, то перед ним выступают В и Ш — 2 варианта расстановки, после Н выступают З и П — 2 варианта. Всего 4 варианта в списке.
2. Если Н выступает четвертым, то перед ним выступают В и Ш и кто-то из З и П. Для выбора З или П — 2 варианта, расстановка этих трех докладчиков — 6 вариантов. Всего 12 вариантов списка.
3. Если Н завершает список, то перед ним 4 докладчика, для которых есть $4! = 24$ вариантов составить список.

Всего $4 + 12 + 24 = 40$ различных списков.

Ответ: 40.

Задача I.2.3.4. (20 баллов)

Темы: числа, делимость.

Саша записал на доске натуральное число $P \geq 2$, Сергей приписал к нему такое же число P . Оказалось, что полученное число \overline{PP} кратно P^2 . Найдите частное от деления \overline{PP} на P^2 .

(Поясним, что означает запись \overline{PP} , на примере: если $P = 9876$, то $\overline{PP} = 98769876$.)

Решение

Если P — однозначное число, то $\overline{PP} = 11 \cdot P$ и, следовательно, $11:P$. Поскольку $P > 1$, этот случай невозможен.

Пусть теперь P — n -значное число, где $n \geq 2$. Тогда $\overline{PP} = P \cdot \underbrace{100\dots001}_{n+1\text{разряд}}$ и для делимости на P^2 необходимо, чтобы $\underbrace{100\dots001}_{n+1\text{разряд}}$ делилось на P . Частное может быть только однозначным числом, большим 1 и являющимся делителем числа $\underbrace{100\dots001}_{n+1\text{разряд}}$. Это число нечетно, поэтому оно не кратно 2, 4, 6, 8. Сумма его цифр равна 2, поэтому это число некратно 3 и 9. Последняя цифра числа 1, поэтому оно некратно 5. Единственный возможный делитель 7.

Пример: $P = 143$, тогда $\overline{PP} = 143\,143$, $\overline{PP} : P^2 = 143\,143 : 143^2 = 7$.

Ответ: 7.

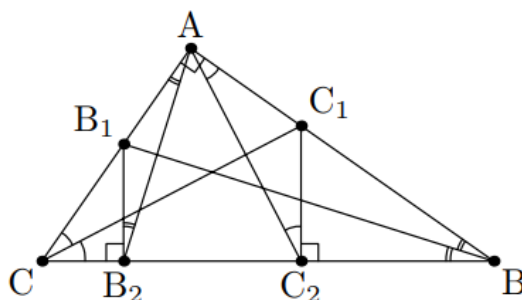
Задача I.2.3.5. (30 баллов)

Темы: геометрия, треугольники.

В прямоугольном треугольнике ABC с прямым углом A проведены биссектрисы BB_1 и CC_1 . Из точек B_1 и C_1 на гипотенузу BC опущены перпендикуляры B_1B_2 и C_1C_2 . Найдите величину угла B_2AC_2 . Ответ запишите в градусах.

Решение

Заметим, что $C_1C_2 = C_1A$ (так как биссектриса CC_1 равноудалена от сторон AC и BC). Значит, треугольник AC_1C_2 равнобедренный, и, следовательно, $\angle C_1AC_2 = \frac{1}{2}\angle BC_1C_2 = \frac{1}{2}\angle ACB$, так как углы BC_1C_2 и ACB оба дополняют угол ABC до прямого. Аналогично получаем, что $\angle B_1AB_2 = \angle ACB$, откуда имеем: $\angle C_1AC_2 + \angle B_1AB_2 = \frac{1}{2}(\angle ACB + \angle ABC) = 45^\circ$.



Ответ: 45.

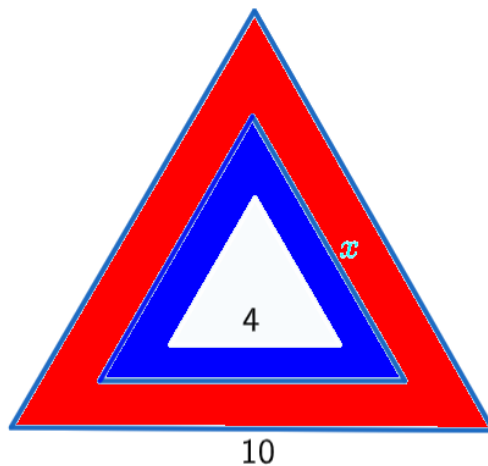
Вторая попытка. Задачи 10–11 класса

Задача I.2.4.1. (15 баллов)

Темы: планиметрия, площадь.

Сергей придумал эмблему нового прибора в виде трех правильных треугольников, имеющих общий центр и попарно параллельные стороны. При этом Сергей хочет

добиться, чтобы средний треугольник делил площадь фигуры, заключенной между внутренним и внешним, в отношении 3 : 4, считая от внутреннего. Какой длины должна быть сторона среднего треугольника, если стороны внутреннего и внешнего треугольников равны 4 см и 10 см соответственно? Ответ в сантиметрах округлите до двух знаков после запятой.



Решение

Как известно, площадь правильного треугольника со стороной a равна $\frac{a^2\sqrt{3}}{4}$. Если обозначить неизвестную сторону среднего треугольника за x см, получим уравнение:

$$\frac{(x^2 - 4^2)\sqrt{3}}{4} : \frac{(10^2 - x^2)\sqrt{3}}{4} = 3 : 4.$$

Отсюда $4(x^2 - 16) = 3(100 - x^2)$, тогда $x = \sqrt{52} \approx 7,21$ (см).

Ответ: 7,21.

Задача I.2.4.2. (15 баллов)

Темы: турниры, алгебра, неравенства.

Команда игроков набрала вместе в игре некоторое количество очков. Лучший игрок команды набрал $1/7$ общего количества очков, а игрок, набравший наименьшее количество очков, набрал $1/9$ от общего количества. Сколько игроков было в команде?

Решение

Пусть команда набрала x очков и было n игроков. Тогда каждый игрок набрал не менее, чем $\frac{x}{9}$ очков и не более, чем $\frac{x}{7}$ очков. Значит, все игроки набрали не менее, чем $\frac{x}{9} + \frac{x}{7} + \frac{x}{9}(n - 2)$ очков, но не более, чем $\frac{x}{9} + \frac{x}{7} + \frac{x}{7}(n - 2)$ очков. Таким образом,

выполняется неравенство $\frac{x}{9} + \frac{x}{7} + \frac{x}{9}(n-2) \leq x \leq \frac{x}{9} + \frac{x}{7} + \frac{x}{7}(n-2)$. Упростив его, получим $\frac{n}{9} + \frac{2}{63} \leq 1 \leq \frac{n}{7} - \frac{2}{63}$, то есть $7\frac{2}{9} \leq n \leq 8\frac{5}{7}$. Поскольку n — целое, то $n = 8$.

Ответ: 8.

Задача I.2.4.3. (20 баллов)

Темы: функции, экстремум.

Найдите наименьшее значение выражения:

$$x^2 + 5y^2 + 6z^2 - 4xy + 4yz - 2x + 6y + 4z + 5.$$

Решение

Рассмотрим данное выражение как квадратичную функцию относительно переменной x :

$$f(x) = x^2 - 2(2y + 1)x + 5y^2 + 6z^2 + 4yz + 6y + 4z + 5.$$

Она принимает наименьшее значение при $x_{\text{в}} = 2y + 1$. Это наименьшее значение $f_{\text{мин}} = y^2 + 6z^2 + 4yz + 2y + 4z + 4$ рассмотрим как квадратичную функцию относительно переменной y : $g(y) = y^2 + 2(2z + 1)y + 6z^2 + 4z + 4$. Эта функция принимает наименьшее значение при $y_{\text{в}} = -(2z + 1)$. Это наименьшее значение $g_{\text{мин}} = 2z^2 + 3$ будет наименьшим при $z = 0$ и будет равно 3.

Ответ: 3.

Задача I.2.4.4. (20 баллов)

Темы: вероятность.

У Саши есть 9 карточек с написанными цифрами от 1 до 9 (на каждой карточке по одной цифре). Саша, не глядя, берет некоторые (может быть, одну или все) из них. Набор он считает хорошим, если сумма цифр в наборе четная. Найдите вероятность, что случайно выбранный набор окажется хорошим. Ответ запишите в виде десятичной дроби с 3 знаками после запятой.

Решение

Введем пустой набор. Пусть он будет хорошим. Тогда количество всех наборов равно 2^9 . Заметим, что сумма всех цифр на карточках — нечетное число. Поэтому любому хорошему набору соответствует набор из оставшихся карточек с нечетной суммой. И наоборот. Поэтому хороших наборов ровно половина от всех, то есть $2^9/2 = 2^8$. Поскольку мы добавили один хороший набор, то настоящих хороших наборов $2^8 - 1 = 255 \cdot \frac{2^8 - 1}{2^9 - 1}$.

Ответ: 0,499.

Задача I.2.4.5. (30 баллов)*Темы: уравнения.*

Решите в действительных числах уравнение:

$$x + \sqrt{x^2 - 16} = \frac{2(x + 4)}{(x - 4)^2}.$$

Если корней несколько, запишите наибольший из них. Ответ округлите до двух знаков после запятой.

Решение

Пусть $x + \sqrt{x^2 - 16} = t$. Тогда, $\sqrt{x^2 - 16} = t - x \Leftrightarrow \begin{cases} t \geq x \\ x^2 - 16 = t^2 - 2tx + x^2 \end{cases}$.

Из второго уравнения следует, что $2tx = t^2 + 16$ и, в частности, $t \neq 0$. Отсюда $x = \frac{t^2 + 16}{2t}$, $x \pm 4 = \frac{t^2 \pm 8t + 16}{2t} = \frac{(t \pm 4)^2}{2t}$. Значит, исходное уравнение будет иметь

вид $t = \frac{2(t + 4)^2}{2t} \cdot \frac{4t^2}{(t - 4)^4}$, или $(t - 4)^4 = 4(t + 4)^2$. Отсюда $\begin{cases} (t - 4)^2 = 2(t + 4) \\ (t - 4)^2 = -2(t + 4) \end{cases} \Leftrightarrow$

$\begin{cases} t^2 - 10t + 8 = 0 \\ t^2 - 6t + 24 = 0 \end{cases}$. Второе уравнение действительных корней не имеет. Корни первого

уравнения $t = 5 \pm \sqrt{17}$ при этом $x = \frac{t^2 + 16}{2t} = \frac{15 \mp \sqrt{17}}{2}$. Учитывая условие $t \geq x$,

получаем, что корнем исходного уравнения является только $x = \frac{15 - \sqrt{17}}{2}$. С учетом округления по математическим правилам до двух знаков после запятой получаем $x \approx 5,44$.

Ответ: 5,44.**Третья попытка. Задачи 8–9 класса****Задача I.2.5.1. (15 баллов)***Темы: логика, принцип Дирихле.*

В школьном кружке «Физики и лирики», где каждый участник или физик или лирик, 27 человек. Сергей провел исследование и выяснил, что у любых двух физиков из кружка количество друзей-лириков из этого кружка не совпадает. Какое наибольшее количество физиков может быть в этом кружке?

Решение

Если в кружке 14 физиков, то количество их друзей-лириков может быть любым целым числом от 0 до 13 (14 различных вариантов), что соответствует условию. Если же физиков больше 14, то лириков в кружке будет не больше 12, а значит, различных вариантов будет не больше 13 (от 0 до 12). Поэтому, по принципу Дирихле, хотя бы у

двух физиков окажется одно и то же количество друзей-лириков, что противоречит условию.

Ответ: 14.

Задача I.2.5.2. (15 баллов)

Темы: алгебра, текстовые задачи.

Удав решил измерить свой рост и попросил попугая ему помочь. Но лежать на месте удаву скучно, поэтому во время измерения он полз с постоянной скоростью. Попугай сначала прошел от хвоста удава к его голове и насчитал при этом 80 своих шагов, а затем немедленно развернулся и на обратном пути к хвосту удава насчитал 20 своих шагов (и от хвоста к голове удава, и обратно попугай шел с одной и той же скоростью). Чему равен рост удава в попугайских шагах?

(Под ростом удава мы, как и в известном мультфильме, понимаем его длину.)

Решение

За время движения туда и обратно попугай сделал $80 + 20 = 100$ шагов, а удав прополз расстояние в $80 - 20 = 60$ шагов. Следовательно, скорость удава в $\frac{100}{60} = \frac{5}{3}$ раз меньше скорости попугая, и за то время, что попугай дошел от хвоста до головы удава, удав переместился на $80 : \frac{5}{3} = 48$ шагов. Следовательно, рост удава (расстояние от головы до хвоста, посчитанное в момент, когда попугай повернул обратно) составляет $80 - 48 = 32$ попугайских шага.

Замечание. В общем случае, если по пути от хвоста к голове удава попугай сделал a шагов, а обратно — b шагов ($a > b$), рост удава составляет $\frac{2ab}{a+b}$ шагов, т. е. *среднее гармоническое* чисел a и b .

Ответ: 32.

Задача I.2.5.3. (20 баллов)

Темы: логика, графы.

Сергею досталась карта острова сокровищ. На ней изображены несколько сундуков с золотом, которые расположены в пещерах (в каждой пещере по одному сундуку). Пещеры соединены тоннелями, прорытыми гномами (каждый тоннель обозначен своим цветом, при этом известно, что тоннелей больше одного). Изучив карту, Сергей обнаружил, что от каждой пещеры до любой другой можно добраться, не меняя тоннеля. Кроме того, каждые два тоннеля пересекаются ровно в одной пещере. А вдоль каждого тоннеля расположены ровно три пещеры.

Сколько всего сундуков с золотом изображено на карте острова сокровищ?

Решение

Пусть a — один из тоннелей, а B — пещера, через которую тоннель a не проходит. Каждый тоннель, проходящий через пещеру B , пересекается с a , причем разные тоннели — в разных пещерах.

Следовательно, через пещеру B проходят ровно три тоннеля. Каждый из них проходит через две пещеры, отличные от B . Таким образом, всего на карте изображено $3 \cdot 2 + 1 = 7$ сундуков.

Замечание. Поскольку через каждую пару пещер (а их $C_7^2 = 21$) проходит один тоннель, а вдоль каждого тоннеля расположено $C_3^2 = 3$ пар пещер, то всего тоннелей $21 : 3 = 7$, т. е. столько же, сколько сундуков.

Пример. Если занумеровать сундуки/пещеры цифрами от 1 до 7, то семь тоннелей могут проходить через сундуки (каждый через три) с номерами: 123, 146, 157, 247, 256, 367 и 345.

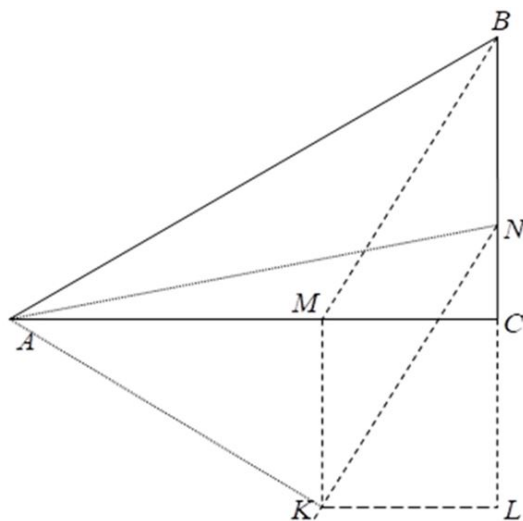
Ответ: 7.

Задача I.2.5.4. (20 баллов)

Темы: геометрия, прямоугольный треугольник.

В прямоугольном треугольнике ABC с прямым углом при вершине C на катете AC выбрана точка M так, что $AM = BC$, а на катете BC — точка N так, что $BN = MC$. Найдите угол между прямыми AN и BM .

Ответ запишите в градусах.

Решение

На перпендикуляре к AC в точке M вне $\triangle ABC$ отложим отрезок $MK = MC$. Затем построим точку L , для которой $MCLK$ — квадрат. Тогда:

$$\triangle AMK = \triangle MCB = \triangle KNL.$$

Поэтому $AK = KN$ и:

$$\angle AKN = \angle AKM + \angle MKN = \angle AKM + \angle KNL = \angle AKM + \angle KAM = 90^\circ.$$

Тогда $\angle ANK = 45^\circ$, а это и есть искомый угол между прямыми, поскольку $BM \parallel KN$.

Ответ: 45.

Задача I.2.5.5. (30 баллов)

Темы: логика.

Саша на числовой прямой отметил все целые точки. Сергей соединяет числа a и b дугой, если расстояние между ними — простое число. Какое наименьшее количество цветов необходимо Саше для окраски всех точек, чтобы любые два числа, которые соединил дугой Сергей, были покрашены в разные цвета?

Решение

Покажем, что меньше чем 4 цветами не обойтись, так как числа 0; 2; 5; 7 попарно соединены дугами, а поэтому все должны быть окрашены в разные цвета. Таким образом необходимо не менее 4 цветов. А теперь просто покрасим числа вида $4n + k$, $k \in \{0, 1, 2, 3\}$ в цвет k . Тогда разность между какими-либо двумя одноцветными числами равна $4m$ и не является простой, поэтому дугой они не соединены.

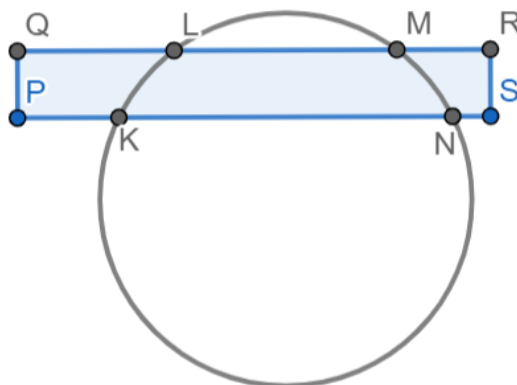
Ответ: 4.

Третья попытка. Задачи 10–11 класса

Задача I.2.6.1. (15 баллов)

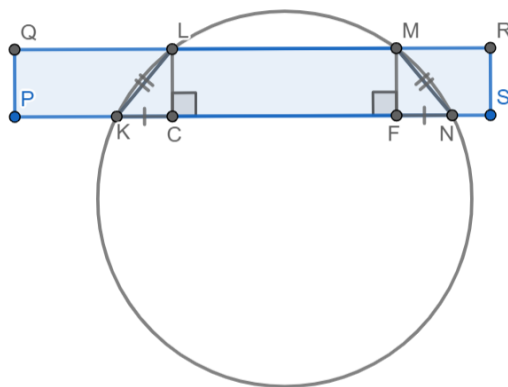
Темы: планиметрия.

Окружность пересекает прямоугольник $PQRS$ так, как показано на рисунке. Известно, что $PK = 5$, $QL = 7$ и $LM = 4$. Найдите KN .



Решение

$KLMN$ — вписанная трапеция, поэтому она равнобедренная. Опустим перпендикуляры LC и MF на прямую PS , тогда прямоугольные треугольники KCL и NFM равны (по гипотенузе и катету). Т. к. $PQLC$ — прямоугольник, то $QL = PC$, отсюда $KC = PC - PK = 7 - 5 = 2$. Окончательно $KN = KC + CF + FN = 8$.



Ответ: 8.

Задача I.2.6.2. (15 баллов)

Темы: числа.

Счетовод Сергей для каждого n от 1 до 2021 считает сумму первых n четных чисел и записывает на доске. Экономист Саша для экономии места после того, как Сергей записывает очередное число, стирает у него все цифры, кроме цифры в разряде единиц. Какова будет сумма чисел на доске?

Решение

Заметим, что Сергей вместе с Сашей для числа n напишет на доску последнюю цифру числа $n(n + 1)$. Эти последние цифры, как нетрудно убедиться перебором, периодичны с периодом 5. Период имеет вид 2, 6, 2, 0, 0. 2021 число содержит 404 периода с суммой в каждом 10 и еще число 2.

Ответ: 4042.

Задача I.2.6.3. (20 баллов)

Темы: функциональные уравнения.

Функция $f(x)$ при любых x принимает положительные значения, причем для любых x и y выполнено соотношение $f(x - y) = \frac{f(x)}{f(y)}$. Найдите значение $f(2021)$, если $f(-2021) = 25$.

Решение

Подставив в формулу $f(x - y) = \frac{f(x)}{f(y)}$ значения $x = y = 1$, получим:

$$f(0) = \frac{f(1)}{f(1)} = 1.$$

Теперь подставим $x = 0$ и $y = -2021$: $f(2021) = \frac{f(0)}{f(-2021)} = \frac{1}{25} = 0,04$.

Ответ: 0,04.

Задача I.2.6.4. (20 баллов)

Темы: вероятность.

Вероятность того, что компьютер успешно загрузится в течение 10 секунд после включения, равна 0,7 (вероятность успешной загрузки при каждом включении одинакова). Саша и Сергей тестируют работу компьютера путем многократных включений/выключений и каждый раз записывают, успел ли компьютер загрузиться за 10 секунд. Тестирование идет до тех пор, пока число либо успешных, либо неуспешных загрузок достигнет шести (не обязательно подряд). В первом случае компьютер будет считаться годным, а во втором — неисправным.

После первых семи включений компьютер успешно загрузился три раза (и, соответственно, четырежды не загрузился) за отведенные 10 секунд.

Какова вероятность, что компьютер будет признан годным? Ответ запишите в виде десятичной дроби.

Решение

Обозначим за $p = 0,7$ вероятность успешной загрузки компьютера при включении, тогда $q = 1 - p = 0,3$ — вероятность того, что компьютер не загрузится при очередном включении.

Чтобы компьютер был признан годным, ему нужно еще 3 раза успешно загрузиться, а для признания неисправным нужно, чтобы он не загрузился хотя бы еще 2 раза. В любом случае до окончания тестирования остается не более 4 включений.

Рассмотрим возможные цепочки успешных ($У$) и неуспешных ($Н$) включений в течение следующих 4 включений (даже если результат тестирования будет известен раньше). Например, цепочка $ННУН$ приведет к браковке компьютера (с вероятностью q^3p), а цепочка $УНУУ$ — к признанию годным (с вероятностью p^3q).

Для признания компьютера годным нужно, чтобы буква «У» в цепочке встретилась 3 или 4 раза. Вероятность этого равна:

$$C_4^4 p^4 + C_4^3 p^3 q = p^3(p + 4q) = 0,7^3 \cdot (0,7 + 4 \cdot 0,3) = 0,6517.$$

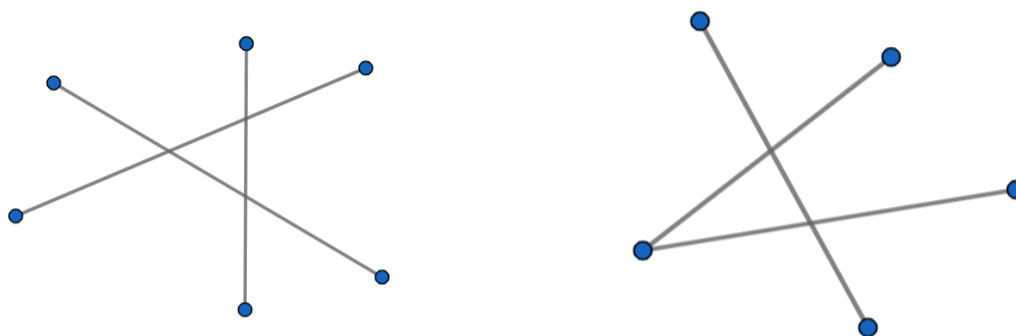
Ответ: 0,6517.

Задача I.2.6.5. (30 баллов)

Темы: планиметрия, комбинаторика.

Сергей отметил на плоскости вершины правильного 17-угольника. Сколько существует троек отрезков с концами в этих вершинах, таких, что каждый пересекает каждый (возможно, в концах)?

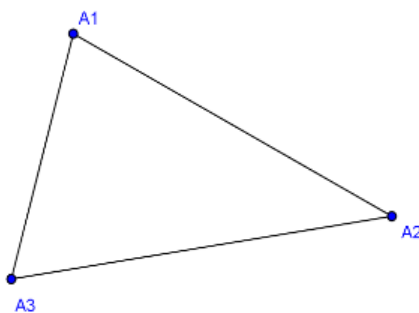
(На рисунках изображены некоторые возможные случаи взаимного расположения отрезков.)



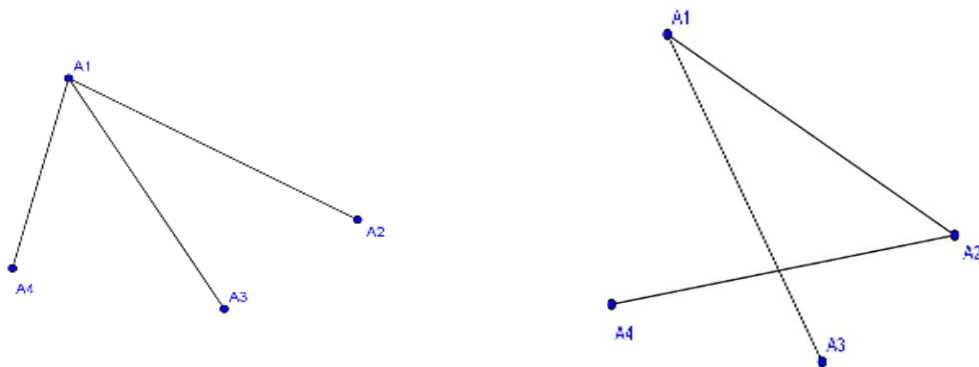
Решение

Три отрезка могут опираться на 3, 4, 5 или 6 точек. Разберем эти случаи.

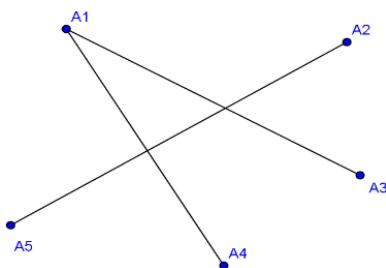
1. Отрезки опираются ровно на 3 точки, которые можно выбрать C_{17}^3 способами. Соединить их отрезками можно единственным способом.



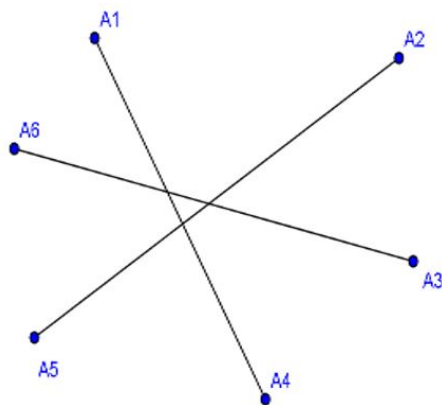
2. Отрезки опираются ровно на 4 точки. Из одной точки (пусть из точки A_1) обязательно должны выходить два отрезка, но есть две возможности для оставшихся отрезков, показанные на рисунке. Для каждого выбора четырех точек, которых C_{17}^4 , существует по 8 способов их соединить отрезками.



3. Отрезки опираются ровно на 5 точек. В этом случае ровно два отрезка имеют общую вершину, третий отрезок соединяет две оставшихся. Для каждого выбора пяти точек, которых C_{17}^5 , существуют пять вариантов (по количеству точек, в которых сходятся два отрезка) проведения отрезков.



4. Отрезки опираются ровно на 6 точек. Для любого выбора шести точек, которых C_{17}^6 , существует ровно один способ проведения отрезков, поскольку каждый отрезок должен оставлять по две точки по разные стороны от него.



Всего способов проведения отрезков будет

$$C_{17}^3 + C_{17}^4 \cdot 8 + C_{17}^5 \cdot 5 + C_{17}^6 = \frac{17 \cdot 16 \cdot 15}{2 \cdot 3} + \frac{17 \cdot 16 \cdot 15 \cdot 14 \cdot 8}{2 \cdot 3 \cdot 4} + \frac{17 \cdot 16 \cdot 15 \cdot 14 \cdot 13 \cdot 5}{2 \cdot 3 \cdot 4 \cdot 5} + \frac{17 \cdot 16 \cdot 15 \cdot 14 \cdot 13 \cdot 12}{2 \cdot 3 \cdot 4 \cdot 5 \cdot 6} = \frac{17 \cdot 16 \cdot 15}{2 \cdot 3} \left(1 + 28 + \frac{14 \cdot 13}{4} + \frac{14 \cdot 13 \cdot 12}{4 \cdot 5 \cdot 6} \right) = 63036.$$

Ответ: 63036.

Четвертая попытка. Задачи 8–9 класса

Задача I.2.7.1. (15 баллов)

Темы: алгебра, преобразования.

Известно, что $\frac{3y+x}{3y-x} + \frac{3y-x}{3y+x} = \frac{41}{20}$. Чему равно значение выражения $\frac{2x^2+y^2}{x^2-y^2}$?

Ответ запишите в виде десятичной дроби.

Решение

Приведем выражение в левой части к общему знаменателю:

$$\frac{(3y+x)^2 + (3y-x)^2}{9y^2 - x^2} = \frac{41}{20}.$$

Отсюда $40(9y^2 + x^2) = 41(9y^2 - x^2)$, или $y^2 = 9x^2$. Тогда:

$$\frac{2x^2 + y^2}{x^2 - y^2} = \frac{2x^2 + 9x^2}{x^2 - 9x^2} = -\frac{11}{8} = -1,375.$$

Ответ: $-1,375$.

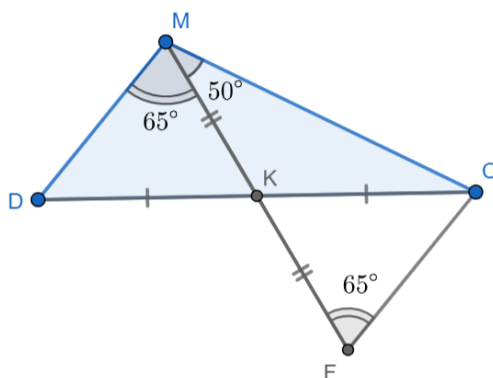
Задача I.2.7.2. (15 баллов)

Темы: геометрия, медиана.

В треугольнике MCD медиана MK вдвое меньше стороны CM и образует с ней угол 50° . Найдите разность углов DMK и CMK . Ответ запишите в градусах (знак градуса вводить не нужно, только число).

Решение

1. На продолжении луча MK за точку K отложим отрезок $KF = MK$. Тогда треугольник MCF равнобедренный ($MF = MC$), поэтому $\angle MCF = \angle MFC = \frac{180^\circ - \angle FMC}{2} = 65^\circ$.
2. Треугольники DKM и CKF равны по двум сторонам и углу между ними ($DK = CK$ по условию, $MK = FK$ по построению, $\angle DKM = \angle CKF$ как вертикальные), поэтому $\angle DMK = \angle CFK = 65^\circ$.
3. Окончательно имеем: $\angle DMK = \angle CMK = 65^\circ - 50^\circ = 15^\circ$.



Ответ: 15.

Задача I.2.7.3. (20 баллов)

Темы: неравенства, алгебра.

Несколько бобров строили плотину на ручье из веток. Самый старательный бобр собрал $\frac{1}{10}$ всех веток, а самый ленивый — $\frac{1}{12}$ всех веток. Сколько бобров строили плотину?

Решение

Пусть бобров было x , и в сумме они собрали N веток. Тогда самый старательный собрал $\frac{N}{10}$ веток, а самый ленивый — $\frac{N}{12}$. В среднем каждый бобр принес $\frac{N}{x}$ веток. Очевидно, что $\frac{N}{12} < \frac{N}{x} < \frac{N}{10}$, тогда $10 < x < 12$. Поскольку x — целое число, $x = 11$.

Пример. Пусть всего было 540 веток, и самый результативный принес 54, самый ленивый — 45, а остальные девять — по 49.

Ответ: 11.

Задача I.2.7.4. (20 баллов)

Темы: комбинаторика.

В пятницу у Сергея 6 уроков: алгебра, геометрия, русский язык, история, химия и физкультура. Сколькими способами можно составить расписание на пятницу при условии, что до физкультуры должны быть алгебра и геометрия (в каком-то порядке)?

Решение

Обозначим предметы буквами А, Г, Р, И, Х и Ф соответственно. Ф может быть третьим, четвертым, пятым или шестым. Разберем эти варианты.

1. Если Ф будет третьей, то перед ней должны быть А и Г — 2 варианта расстановки, после Ф будут Р, И и Х — 6 вариантов. Всего 12 вариантов расписания.

2. Если Φ будет четвертой, то перед ней будут А и Г и что-то из Р, И и Х. Для выбора Р, И или Х — 3 варианта, расстановка этих трех предметов — 6 вариантов. Расстановка двух предметов после Φ — 2 варианта. Всего 36 вариантов расписания.
 3. Если Φ будет пятой, то перед ней будут А и Г и еще пара предметов из Р, И и Х. Для выбора этой пары из Р, И и Х — 3 варианта, расстановка этих четырех предметов — 24 варианта. Всего 72 варианта расписания
 4. Если Φ завершает день, то перед ней 5 предметов, для которых есть $5! = 120$ вариантов составить расписание.
- Всего $12 + 36 + 72 + 120 = 240$ различных списков.

Ответ: 240.

Задача I.2.7.5. (30 баллов)

Темы: делимость, целые числа.

На какую наибольшую степень числа 2021 делится число $2021! = 1 \cdot 2 \cdot \dots \cdot 2021$?

Решение

$2021 = 43 \cdot 47$ (43 и 47 — простые числа). Посмотрим, сколько раз в разложении числа $2021!$ на простые множители встречаются числа 43 и 47.

1. число 43 встречается 48 раз (47 раз за счет делимости чисел 43, 86, 129 и т. д. вплоть до 2021, и один дополнительный раз за счет числа 43^2).
2. число 47 встречается 43 раза.

Итак, $2021! = 43^{48} \cdot 47^{43} \cdot k$, где число k взаимно просто с 2021. Поэтому $2021! = (43 \cdot 47)^{43} \cdot t = 2021^{43} \cdot t$, где число t не кратно 2021.

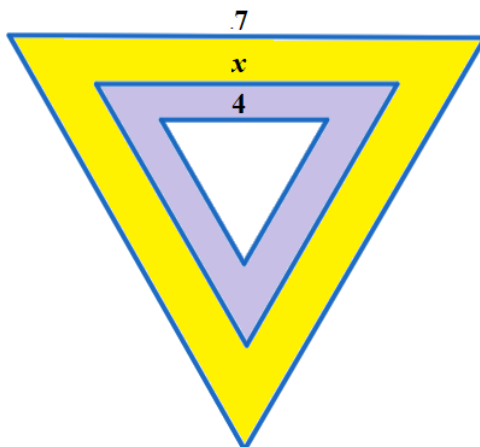
Ответ: 43.

Четвертая попытка. Задачи 10–11 класса

Задача I.2.8.1. (15 баллов)

Темы: планиметрия, площадь.

Сергей придумал эмблему нового прибора в виде трех правильных треугольников, имеющих общий центр и попарно параллельные стороны. При этом Сергей хочет добиться, чтобы отношение площади сиреновой (внутренней) части эмблемы к площади желтой (внешней) части составляло $2 : 5$. Какой длины должна быть сторона среднего треугольника, если стороны внутреннего и внешнего треугольников равны 4 см и 7 см соответственно? Ответ в сантиметрах округлите до двух знаков после запятой.



Решение

Как известно, площадь правильного треугольника со стороной a равна $\frac{a^2\sqrt{3}}{4}$. Если обозначить неизвестную сторону среднего треугольника за x см, получим уравнение:

$$\frac{(x^2 - 4^2)\sqrt{3}}{4} : \frac{(7^2 - x^2)\sqrt{3}}{4} = 2 : 5.$$

Отсюда $5(x^2 - 16) = 2(49 - x^2)$, тогда $x = \sqrt{\frac{178}{7}} \approx 5,04$ (см).

Ответ: 5,04.

Задача I.2.8.2. (15 баллов)

Темы: вероятность, комбинаторика.

Саша забыл 4-значный код от банковской карты. Он помнит, что все цифры в коде различные и нечетные. Еще он помнит, что сумма каких-то двух цифр равна сумме оставшихся цифр. Какова вероятность, что Саша наберет верный код с первой попытки? Ответ запишите в виде десятичной дроби с округлением до двух знаков после запятой.

Решение

Существуют только три группы из четырех различных нечетных цифр, для которых сумма двух из них равна сумме оставшихся:

$$1, 3, 5, 7 (1 + 7 = 3 + 5)$$

$$1, 3, 7, 9 (1 + 9 = 3 + 7)$$

$$3, 5, 7, 9 (3 + 9 = 5 + 7)$$

Количество комбинаций в каждой группе равно $4! = 24$, то есть всего возможных комбинаций 72, из которых только одна правильная, то есть искомая вероятность равна $\frac{1}{72} \approx 0,013 \approx 0,01$.

Ответ: 0,01.

Задача I.2.8.3. (20 баллов)

Темы: функции.

Найдите наименьшее значение выражения:

$$8x^2 + 2y^2 + z^2 + 4xy - 2zy - 12x - 2y - 2z + 9.$$

Решение

Запишем исходное выражение в виде квадратного трехчлена относительно переменной z :

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

Минимальным это выражение будет в случае равенства 0 всех выражений под квадратами. Это достигается при $x = 0,5$, $y = 1$, $z = 2$. Минимальное значение равно 3.

Ответ: 3.

Задача I.2.8.4. (20 баллов)

Темы: делимость, целые числа.

Саша купил 20 пирожков с мясом и 21 пирожок с капустой, потратив все деньги, которые были у него в кошельке. Проанализировав покупку, Саша понял, что цены на пирожки могли быть только такими, чтобы он мог потратить все свои деньги и купить то же количество пирожков каждого вида (то есть если бы цены были какими-нибудь другими, то Саша бы не смог потратить то количество денег, что он потратил в итоге, и одновременно с этим купить то же самое количество пирожков каждого вида). Известно, что пирожок каждого вида стоит целое положительное число рублей.

Какое наибольшее количество рублей могло быть у Саши в кошельке?

Решение

Пусть пирожок с мясом стоит a рублей, пирожок с капустой — b рублей, а в кошельке у Саши s рублей. Тогда $s = 20a + 21b$, причем известно, что такие натуральные числа a и b единственны. Докажем, что наибольшее возможное значение s равно 840.

Оценка. Если $s > 2 \cdot 420 = 840$, то стоимость всех пирожков хотя бы одного из видов (с мясом или с капустой) больше 420 рублей. Пусть пирожки с мясом стоят больше 420 рублей, тогда $a > 21$. Если взять $c = a - 21$ и $d = b + 20$, мы получим альтернативный способ потратить все Сашины деньги. В самом деле, если пирожок с мясом будет стоить c рублей, а пирожок с капустой — d рублей, то общая сумма составит $20c + 21d = 20(a - 21) + 21(b + 20) = 20a + 21b = s$. Если же пирожки с капустой стоят больше 420 рублей, то $b > 20$ и можно взять $c = a + 21$, $d = b - 20$, причем $20c + 21d = 20a + 21b = s$.

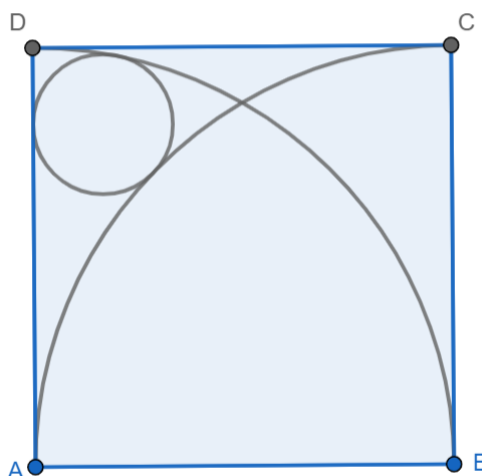
Пример. Если $s = 840$, то уравнение $20a + 21b = 840$ имеет ровно одно решение в натуральных числах: $a = 21$, $b = 20$. В самом деле, перепишем уравнение в виде $20k + b = 840$, где $k = a + b$, тогда $b = 20(42 - k)$, т. е. $b : 20$, а поскольку $b \in \mathbb{N}$, то $b \geq 20$. Аналогично, переписав уравнение в виде $21k - a = 840$, получим $a = 21(k - 40)$, поэтому $a : 21$, а с учетом того, что $a \in \mathbb{N}$, $a \geq 21$. Тогда $20a + 21b \geq 20 \cdot 21 + 21 \cdot 20 = 840$, причем равенство возможно только при наименьших значениях: $a = 21$, $b = 20$.

Ответ: 840.

Задача I.2.8.5. (30 баллов)

Темы: планиметрия, окружность.

В квадрате $ABCD$ проведены дуги с центрами A и B и радиусом, равным стороне квадрата. Окружность радиуса 2 касается стороны AD и двух данных дуг (см. рисунок). Найдите сторону квадрата.

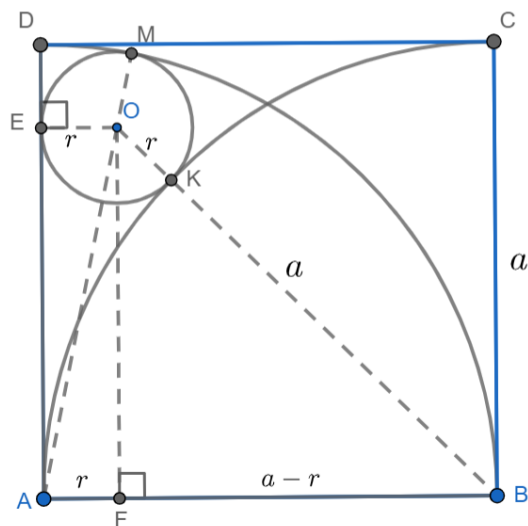


Решение

Обозначим сторону квадрата за a , радиус вписанной окружности за r . Пусть точки касания E , M , K . Также опустим перпендикуляр OF на сторону AB .

1. Поскольку точка касания двух окружностей находится на прямой, проходящей через центры окружностей, точки A , O и M лежат на одной прямой, поэтому $AO = AM - MO = a - r$. По теореме Пифагора для прямоугольного треугольника AFO : $OF^2 = AO^2 - AF^2$.

2. Аналогично, точки B , K и O лежат на одной прямой, поэтому $BO = BK + KO = a + r$. По теореме Пифагора для прямоугольного треугольника BFO : $OF^2 = BO^2 - BF^2$.
3. Из пп. 1 и 2 следует, что $AO^2 - AF^2 = BO^2 - BF^2$, или $(a - r)^2 - r^2 = (a + r)^2 - (a - r)^2$. Отсюда $a = 6r$. Подставляя значение $r = 2$, получим $a = 12$.



Ответ: 12.

Второй отборочный этап

Индивидуальная часть

Задача П.1.1. Лазерный забор (44 баллов)

Темы: математика, алгоритмы.

Эта задача проверяет ваш навык работы с математическими моделями, что пригодится при работе со стендом на финале.

Условие

Ученый с мировым именем Иннокентий соорудил на своем дачном участке прототип Вечного Двигателя™ и решил возвести вокруг него **забор**, чтобы уберечь агрегат от недоброжелателей. За этим Иннокентий обратился к знакомым строителям.

Правда, в меру безумный ученый в качестве чертежа дал рабочим **систему линейных неравенств**, которая задает участок в виде **выпуклого многоугольника**. Пока строители пребывают в замешательстве, помогите им и напишите программу для вычисления **периметра** участка, чтобы узнать, сколько материалов нужно закупить.

Формат входных данных

Список неравенств вида $y \leq / \geq kx + b$, представленных тройками $(k, b, sign)$, где k и b — коэффициенты уравнения, $sign$ определяет знак неравенства (**True** соответствует \geq , **False** — \leq). Пример формата:

```
[(1, 0, False), (-1, -10, True), (100.5, 10.5, False)]
```

Формат выходных данных

Единственное вещественное число, периметр фигуры. Ответ принимается, если значение различается не больше, чем на 10^{-6} .

Ограничение по времени выполнения программы: 1 секунда.

Ограничение по памяти: 256 МВ.

Решение

Для решения этой задачи нужно определить упорядоченные точки искомого многоугольника, после чего периметр находится тривиально. Авторское решение предлагает следующий алгоритм:

- Попарно определить пересечения всех прямых. Это задача из аналитической геометрии, решение которой в данном случае тривиально.
- Отфильтровать точки, не удовлетворяющие неравенствам с помощью подстановки в уравнения прямых.
- Упорядочить точки по или против часовой стрелки. В авторском решении используется алгоритм выпуклой оболочки (`convex hull`).

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1  import math
2
3  def dist(p, q):
4      return math.sqrt(sum((px - qx) ** 2.0
5                          for px, qx in zip(p, q)))
6
7  def intersect_equs(a, b):
8      if a[0] == b[0]:
9          return None
10     x = (b[1] - a[1]) / (a[0] - b[0])
11     y = a[0] * x + a[1]
12     return (x, y)
13
14  def cross(o, a, b): # 2D cross product
15     return (a[0] - o[0]) * (b[1] - o[1]) - (a[1] - o[1]) * (b[0] - o[0])
16
17  def convex_hull(points):
18     # returns points in counter-clockwise order
19     points = sorted(set(points))
20     if len(points) <= 1:
21         return points
22     lower = []
23     for p in points:
24         while len(lower) >= 2 and cross(lower[-2], lower[-1], p) <= 0:
25             lower.pop()
26         lower.append(p)
27     upper = []
28     for p in reversed(points):
29         while len(upper) >= 2 and cross(upper[-2], upper[-1], p) <= 0:
30             upper.pop()
31         upper.append(p)
32     return lower[:-1] + upper[:-1]
33
34  def perimeter(p):
35     return sum(dist(p[i], p[i-1]) for i in range(len(p)))
36
37  equs = eval(input())
38  EPS = 1e-6
39  pts, pts2 = [], []
40  for i in range(len(equs)):
41     for j in range(i+1, len(equs)):
42         s = intersect_equs(equs[i], equs[j])
43         if s:
44             pts.append(s)
45  for (x, y) in pts:
46     for (k, b, sign) in equs:
47         yr = k * x + b

```

```

48     if abs(yr - y) <= EPS: continue
49     if sign != (y > yr): break
50     else:
51         pts2.append((x, y))
52     print(perimeter(convex_hull(pts2)))

```

Задача II.1.2. Сам себе предсказатель (16 баллов)

Темы: алгоритмы, анализ данных.

Эта задача проверяет ваш навык анализа данных, что является частью финальной задачи.

Условие

Программист-радиоэлектронщик Саша Текстолизов собрался доработать свой электросамокат и добавить ему **прогноз** расстояния, которое можно проехать на оставшемся заряде аккумулятора. А так как на холоде аккумулятор разряжается быстрее, нужно было учесть зависимость скорости разряда от температуры. Для этого Саша целый год собирал данные и, проверив их, выяснил, что эта зависимость представляет собой **непрерывную гладкую** функцию. Чтобы не подбирать значения этой функции, он решил реализовать **интерполяцию** по собранным данным. Саша хочет поручить эту задачу **вам**, но для начала предлагает **потренироваться**.

Известно некоторое количество **точек** неизвестной функции, равномерно распределенных в пределах области ее определения. **Вычислите** с максимально возможной точностью значения Y функции по известным аргументам X . Обработайте индивидуальный набор данных.

Формат входных данных

Первая строка — два числа через пробел, N известных точек и M аргументов для вычисления. После идет N строк, в каждой через пробел приводятся аргумент X_i и соответствующее значение величины Y_i . После идет M строк, в каждой единственное число X'_j , аргумент, для которого необходимо вычислить значение величины Y'_j . Например,

```

4 3
0 0
1 1
2 2
3 3
1.5
1.75
-1

```

Формат выходных данных

M строк, в каждой — соответствующее значение Y'_j величины для перечисленных аргументов. Ответ принимается, если максимальная погрешность всех значений не превышает 1%.

Решение

Как можно прочесть из условия, суть задачи в **интерполяции** значения функции по доступным ее точкам. Ограничения задачи позволяют использовать линейную интерполяцию, основанную на пропорции между двумя ближайшими точками к измеряемой:

$$Y' = Y_1 + (Y_2 - Y_1) \frac{X' - X_1}{X_2 - X_1}.$$

Несмотря на «ручной» формат задачи, рациональнее всего написать программу, которая выполняет бинарный поиск ближайшего значения X и считает пропорцию относительно него.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 from bisect import bisect_left
2 N, M = map(int, input().split())
3 pts = [tuple(float(x) for x in input().split()) for _ in range(N)]
4 pts.sort()
5 xt, yt = zip(*pts)
6 xs = [float(input()) for _ in range(M)]
7 for x in xs:
8     i = bisect_left(xt, x)
9     print(yt[i] + (x - xt[i]) * (yt[i+1] - yt[i]) / (xt[i+1] - xt[i]))

```

Задача II.1.3. Сетевая инвентаризация (20 баллов)

Темы: алгоритмы, графы.

Эта задача тренирует навыки работы с алгоритмами и графами, что потребуется в решении финальной задачи.

Условие

Сетевая компания «Энергия хаоса» решила произвести **ревизию** ранее построенных сетей, а также выявить их **изолированные** участки в рамках подготовки к работам по повышению отказоустойчивости. В распоряжении компании есть информация о построенных между крупными узлами **линиях электропередачи**. Используя эту информацию, напишите программу для **определения** отдельных сетей и их состава.

Отдельной сетью называется **связный** подграф из узлов, каждый из которых условно обозначен порядковым номером (**нумерация с 0**). Известно, что таких сетей больше одной.

Формат входных данных

Список ребер (до 160), представленных **парами** номеров узлов. Узлы нумеруются с 0. Пример формата:

```
[(1, 2), (4, 3), (5, 6), (6, 4), (10, 7)]
```

Формат выходных данных

Произвольное количество строк по числу отдельных сетей, в каждой строке номера узлов сети через **пробел**. Порядок вывода сетей и узлов внутри сети может быть любой. Например,

1 2 3

4 5

30 10 20 15 25

Ограничение по времени выполнения программы: 2 секунды.

Ограничение по памяти: 256 МВ.

Решение

Если избавиться от легенды, задача сводится к поиску компонент связности в графе. Алгоритмы для этого находятся в открытом доступе (в т. ч. на ресурсах по спортивному программированию). Авторское решение использует интуитивный алгоритм, основанный на двух словарях: номер сети — список узлов, номер узла — номер его сети (индекс). Каждое прочитанное из входных данных ребро обрабатывается следующим образом:

- если оба узла ребра не принадлежат какой-либо сети — они образуют новую сеть;
- если оба узла принадлежат разным сетям — перемещаем узлы из одной сети в другую;
- если только один узел принадлежит сети — присоединяем в нее другой.

В конце чтения остается вывести списки узлов из словаря, содержащие составы компонент связностей.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 data = eval(input())
2 pos = dict()
3 nets = dict()
4 maxnet = 0
5 for (a, b) in data:
6     if a not in pos and b not in pos:
7         pos[a] = maxnet
8         pos[b] = maxnet
9         nets[maxnet] = {a, b}
10        maxnet += 1
11        continue
12    if a in pos and b in pos:
13        na = pos[a]
14        nb = pos[b]
15        if na == nb: continue
16        for x in nets[nb]:
17            pos[x] = na

```

```

18     nets[na].update(nets[nb])
19     del nets[nb]
20     if a not in pos:
21         pos[a] = pos[b]
22         nets[pos[b]].add(a)
23         continue
24     if b not in pos:
25         pos[b] = pos[a]
26         nets[pos[a]].add(b)
27         continue
28     print("\n".join(" ".join(map(str, n))
29                   for n in nets.values()))

```

Задача П.1.4. Миф меритократии (50 баллов)

Темы: теория вероятностей.

Эта задача знакомит вас с одним теоретико-игровым явлением, а также проверяет ваши знания теории вероятностей, что потребует в решении финальной задачи.

Условие

Подзадача 1. Третьеразрядник Пешкин любит **играть** в шахматы, особенно с шахматными ботами, которые никогда не устают. Когда же Пешкин устает сам, он сталкивает ботов между собой, подкручивая им **уровень игры**, представляющий собой **вещественное** число в диапазоне от 1 до 100 (чем больше число, тем лучше играет бот).

Он заметил, что практически всегда результат игры предreshался **соотношением** уровней игры ботов (т. е. у кого выше уровень, тот и победит), но настоящие игроки могут ошибаться или, наоборот, делать неожиданно сильные ходы. Однако такое случалось и с ботами, особенно если параллельно с игрой операционная система «незаметно» что-то обновляет. Тогда он предположил, что колебания нагрузки на компьютер с ботами дают погрешность в уровне игры каждого бота. Пусть эта **погрешность** будет равна $\pm 1\%$ и распределена **равномерно**.

Тут Пешкин задумался: какова в таком случае вероятность того, что в круговом турнире из **1000 ботов**, уровень которых **равномерно** распределен по всей шкале, самый сильный бот наберет **максимум** (т. е. наиболее возможное число) очков? Вычислите эту **вероятность** и дайте ответ в **процентах** с точностью до 10^{-2} .

Для решения этой задачи у вас есть 15 попыток.

Введите численный ответ.

Подзадача 2. Напишите программу, которая даст ответ для произвольного числа ботов N и погрешности $p\%$.

Формат входных данных

Два целых числа через пробел, число игроков и погрешность в процентах. Например, 1000 1.

Формат выходных данных

Единственное вещественное число, вероятность сильного бота получить максимум очков. Ответ принимается, если он отличается от авторского **не более**, чем на 1%.

Для решения этой задачи у вас есть 15 попыток.

Ограничение по времени выполнения программы: 1 секунда.

Ограничение по памяти: 256 МВ.

Решение

Рассмотрим решение общего случая (т. е. второй подзадачи). Для начала определим величины L (назначенный уровень игры) и P (искомая вероятность):

$$L_i = 1 + 99 \cdot \frac{i}{N} \quad i = 1 \dots N.$$

$$P = \prod_{i=1}^{N-1} (1 - Q_i).$$

Так, задача сводится к поиску значений величины Q_i — вероятности i -го игрока (более слабого) проиграть игроку N (самому сильному). Обозначим этих игроков A и B соответственно и рассмотрим диапазоны фактических уровней игры $L_a \pm P\%$ и $L_b \pm P\%$.

Введем величины (X — соответствующий игрок):

- худший фактический уровень игры $WorstX = L_x \times (1 - P)$;
- лучший уровень $BestX = L_x \times (1 + P)$;
- размах уровня $DeltaX = BestX - WorstX$.

Если $BestA < WorstB$, то игрок A заведомо проиграл ($Q = 0$). Иначе мы имеем три случая:

- A играет слишком плохо ($L'_a < WorstB$) — победа B ;
- B играет слишком хорошо ($L'_b > BestA$) — победа B ;
- уровни попали на пересечение ($WorstB < \{L'_a, L'_b\} < BestA$) — игрок A победит с вероятностью 50%, что можно доказать геометрически с учетом того, что уровни игры — равномерные равномерные величины.

Пусть $\Delta = BestA - WorstB$, тогда $Q = \frac{\Delta}{DeltaA} \times \frac{\Delta}{DeltaB} \times \frac{1}{2}$.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1  minSkill, maxSkill = 1, 100
2
3  # вероятность сильного бота проиграть слабому
4  def compare(weak, strong, fault):
5      bestMul = 1 + fault
6      worstMul = 1 - fault
7      rangeMul = bestMul - worstMul

```

```

8     bestWeak = bestMul * weak
9     worstStrong = worstMul * strong
10    # очевидная победа?
11    if (bestWeak <= worstStrong):
12        return 0
13    delta = bestWeak - worstStrong
14    fullWeak = rangeMul * weak
15    fullStrong = rangeMul * strong
16    return (delta/fullWeak) * (delta/fullStrong) / 2
17
18    count, fault = map(int, input().split())
19    fault = fault / 100
20    participants = [
21        minSkill + i/(count-1) * (maxSkill-minSkill)
22        for i in range(count)
23    ]
24
25    result = 1
26    best = participants.pop()
27
28    for p in participants:
29        x = compare(p,best,fault)
30        result *= (1 - x)
31
32    print(result*100)

```

Задача II.1.5. Проклятие победителя (20 баллов)

Темы: теория игр.

Эта задача проверяет ваш навык обработки данных (в том числе программной) и работы с теоретико-игровыми моделями, что потребуется при работе с финальной задачей. Помимо этого, задача использует ряд понятий из теории аукционов. Подробнее о них вы можете узнать в модуле 3 учебного курса <https://onti.polyus-nt.ru/course/view.php?id=2>.

Условие

Подзадача 1. На аукционе с N участниками выставляется M лотов (положим $N = 20$, $M = 100$). Все участники являются **покупателями**. У каждого лота есть истинная ценность, а у каждого участника — представления о ней, находящиеся в коридоре $\pm 10\%$ от истинной. Каждый лот выставляется на аукцион **первой цены**. Какой участник получит **наибольшую выгоду** от участия в аукционе?

Формат входных данных

$N+1$ строка, в каждой из которых приведено M вещественных чисел через пробел. В первой строке приведены истинные ценности соответствующих лотов, в следующих строках — оценки участников (вторая строка — участник под номером 0, третья — 1 и т. д.). Пример формата для 5 лотов и 4 участников:

```

1 2 3 4 5
1.1 1.9 3.15 3.8 4.5
1.2 1.8 3.18 3.3 4.6

```

1.3 1.7 3.13 3.2 4.3
 1.4 1.6 3.11 3.5 4.2

Формат выходных данных

Единственное целое число, номер участника с **наибольшей** выгодой (нумерация идет с 0). Решение принимается, если выгода выбранного вами участника больше выбранного авторским решением, либо отличается меньше, чем на 10^{-6} .

Обработайте индивидуальный набор данных.

Подзадача 2. Напишите **программу**, решающую аналогичную задачу для **произвольных** N (до 30) и M (до 500). Входной и выходной форматы аналогичны прошлой подзадаче, но теперь на входе первой строкой через пробел идут N и M , дальше данные аналогичны. Например,

```
4 5
1 2 3 4 5
1.1 1.9 3.15 3.8 4.5
1.2 1.8 3.18 3.3 4.6
1.3 1.7 3.13 3.2 4.3
1.4 1.6 3.11 3.5 4.2
```

Для решения этой задачи у вас есть 15 попыток.

Ограничение по времени выполнения программы: 1 секунда.

Ограничение по памяти: 256 МВ.

Решение

Прежде всего необходимо определить, что есть выгода. В аукционе первой цены — это разница между истинной ценностью и оценкой участника: если участник сделал ставку больше истинной ценности, он забирает лот, но получает отрицательную выгоду. К слову, в этом и проявляется «проклятие победителя».

Так, зная определение выгоды, мы можем перебрать лоты, определив в них победителей и их ставки (сортировкой или поиском максимума) и просуммировав в каждом случае для победителя разницу истинной ценности и его ставки. После этого определить индекс максимума в списке суммарных выгод, который и будет являться ответом.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
1 players, lots = map(int, input().split())
2 utils = [float(x) for x in input().split()]
3 A = [[float(y) for y in input().split()] for x in range(players)]
4
5 profits = [0 for _ in range(players)]
6 for lot in range(lots):
7     winners = sorted(((A[p][lot], p) for p in range(players)))
8     winner = winners[-1][1]
```

```

9     profits[winner] += utils[lot] - winners[-1][0]
10
11     (_, best) = max((profits[pl], pl) for pl in range(players))
12     print(best)

```

Командная часть

Задача II.2.1. Неслабое звено (119 баллов)

Темы: графы, алгоритмы.

Эта задача тренирует навыки работы с алгоритмами и графами, что потребуется в решении финальной задачи.

Условие

После *сетевой инвентаризации* компания «Энергия хаоса» приступила к работам по повышению отказоустойчивости, а именно — **соединению** отдельных своих сетей в одну целую. Для этого им необходимо **построить** линии электропередачи между изолированными сетями. В распоряжении компании есть все та же информация о построенных между крупными узлами **линиях электропередачи**, а также **координаты** этих узлов на условной декартовой плоскости. Используя эту информацию, **определите** узлы, между которыми нужно построить ЛЭП, чтобы объединить изолированные сети в одну. При этом суммарная длина прокладываемых ЛЭП должна быть **минимальной**.

Формат входных данных

Два списка. Первый — список координат узлов (до 100) на декартовой плоскости. Координаты представлены парой (X, Y) . Второй — список ребер (до 160), представленных парами номеров узлов. Узлы нумеруются с 0, их номер соответствует индексу в первом списке. Все эти данные представлены в виде строки, которую можно проинтерпретировать командой `eval`. Пример формата:

```
((0, 0), (1, 1), (2,3)], [(1, 2), (4, 3), (5, 6), (6, 4), (10, 7)])
```

Формат выходных данных

Число строк, соответствующее количеству новых соединений. В каждой строке два целых числа через пробел, соответствующие номерам соединяемых узлов (нумерация с нуля). Порядок вывода ребер и перечисления узлов внутри ребра может быть **любой**. Например,

```
1 2
```

```
4 5
```

```
30 10
```

Решение **засчитывается**, если новые соединения делают сеть связной, и суммарная длина ребер не превышает выданную авторским решением больше, чем на 5%.

Для решения этой задачи у команды есть 20 попыток.

Ограничение по времени выполнения программы: 1 секунда.

Ограничение по памяти: 256 МВ.

Решение

Как намекает условие, эта задача является продолжением задачи «Сетевая инвентаризация» индивидуальной части. В частности, первый шаг решения этой задачи полностью повторяет решение «приквела» — поиск компонент связности в графе. Этот алгоритм довольно распространен, поэтому его описание здесь не приводится (но есть в разделе с прошлой задачей).

После выявления компонента связности необходимо определить минимальные ребра, соединяющие каждую из пар выявленных компонент. Несмотря на ограничение в секунду, лобовой попарный перебор узлов каждой из компонент и поиск ребра с минимальной длиной проходит по ограничениям. Тем не менее, допуск в 5% позволяет использовать быстрое приближительное решение: вычисляем центр масс каждой из компонент (среднее арифметическое по каждой из координат), после чего находим узел первой компоненты, ближайший к центру второй, после узел второй, ближайший к выбранному узлу первой. Такой алгоритм требует меньше итераций и укладывается в допуск.

После выявления соединяющих ребер остается построить их дерево, т. е. выбрать $N - 1$ ребро, которое соединяет все компоненты связности. Для этого можно использовать модифицированный алгоритм поиска компонент связности. Теперь каждое ребро само по себе является компонентой связности, и мы добавляем в граф ребра (предварительно отсортировав их в порядке возрастания длины). Если ребро объединяет компоненты связности — оно добавляется в ответ, если нет — игнорируется. И так до тех пор, пока в графе из компонент связности не останется единственная компонента.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1  def get_nets(egs):
2      pos = dict()
3      nets = dict()
4      maxnet = 0
5      for (a, b) in egs:
6          if a not in pos and b not in pos:
7              pos[a] = maxnet
8              pos[b] = maxnet
9              nets[maxnet] = {a, b}
10             maxnet += 1
11             continue
12         if a in pos and b in pos:
13             na = pos[a]
14             nb = pos[b]
```

```

15         if na == nb: continue
16         for x in nets[nb]:
17             pos[x] = na
18             nets[na].update(nets[nb])
19             del nets[nb]
20     if a not in pos:
21         pos[a] = pos[b]
22         nets[pos[b]].add(a)
23         continue
24     if b not in pos:
25         pos[b] = pos[a]
26         nets[pos[a]].add(b)
27         continue
28     return list(nets.values())
29
30 def dist(a, b):
31     return ((a[0]-b[0])**2 + (a[1]-b[1])**2)**0.5
32
33 def center(xs):
34     l = len(xs)
35     x = sum(a[0] for a in xs) / l
36     y = sum(a[1] for a in xs) / l
37     return (x, y)
38
39 def connect(coords, nets, brute=False):
40     net_coords = [(i, net) for i, net in enumerate(nets)]
41     centers = [center(net) for net in net_coords]
42     n = len(nets)
43
44     egs = []
45     for i in range(0, n-1):
46         for j in range(i+1, n):
47             net_a, net_b = net_coords[i], net_coords[j]
48             if brute:
49                 eg = min((dist(a, b), (a[2], b[2]))
50                        for a in net_a for b in net_b)[1]
51                 egs.append(eg)
52             else:
53                 # берем ближайшие к противоположному центру,
54                 # затем ближайшие к ближайшим, они должны образовать
55                 # разумное решение
56                 center_a, center_b = centers[i], centers[j]
57                 closest_a = min((dist(n, center_b), n) for n in net_a)[1]
58                 d1, near_b = min((dist(n, closest_a), n) for n in net_b)
59                 closest_b = min((dist(n, center_a), n) for n in net_b)[1]
60                 d2, near_a = min((dist(n, closest_b), n) for n in net_a)
61                 if d1 < d2:
62                     egs.append((closest_a[2], near_b[2]))
63                 else:
64                     egs.append((closest_b[2], near_a[2]))
65
66     # берем ребра в порядке возрастания длины
67     # по мере соединения сетей в один граф
68     net_vertex = [None for _ in coords]
69     for i, net in enumerate(nets):
70         for x in net:
71             net_vertex[x] = i
72
73     egs.sort(key=lambda a: dist(coords[a[0]], coords[a[1]]))
74     cs = [i for i in nets]

```

```

75     cn = n
76     ans = []
77     for (ca, cb) in egs:
78         if cn == 1: break
79         a = net_vertex[ca]
80         b = net_vertex[cb]
81         if cs[a] == cs[b]:
82             continue
83         old = cs[b]
84         for i in range(n):
85             if cs[i] == old:
86                 cs[i] = cs[a]
87         cn -= 1
88         ans.append((ca, cb))
89
90     return ans
91
92     coords, egs = eval(input())
93     nets = get_nets(egs)
94     sol = connect(coords, nets, brute=True)
95     print("\n".join(" ".join(map(str, n))
96                   for n in sol))

```

Задача II.2.2. Энергия дождя (77 баллов)

Темы: алгоритмы.

Эта задача проверяет навык писать эффективные алгоритмы анализа данных, что потребуется в решении финальной задачи.

Условие

Подзадача 1. Энергию можно добывать не только из солнца, ветра или течения рек, но даже из падающих капель дождя. Гипотез на этот счет много, и для проверки одной из них, а именно, испытания экспериментальной дождевой электростанции, юным ученым нужно выбрать подходящий **участок** местности, где осадков больше всего. Они нашли подходящий массив в тропиках и имеют на руках информацию о **среднегодовой норме осадков**.

Эта информация представлена **картой**, поделенной на **секторы**, для каждого из которых указана норма осадков. Карта имеет **размер** 200 на 200 секторов. Известно, что ученым нужно выбрать на этой карте участок **размером** 5 на 5 секторов с **максимальной** суммарной нормой осадков. Помогите им выполнить необходимые вычисления.

Формат входных данных

Двухмерная матрица 200 на 200 из вещественных чисел, перечисленных по строкам через пробел, выглядит вот так:

```

1 2 3 4
5 6 7 8
9 10 11 12

```

13 14 15 16

Формат выходных данных

Два целых числа через **пробел**, координаты X (по горизонтали) и Y (по вертикали вниз) левого верхнего **углового** сектора будущего участка. Координаты нумеруются **с нуля!**

Если несколько участков **одинаково** подходят по увлажнению, выведите лексикографически https://neerc.ifmo.ru/wiki/index.php?title=Лексикографический_порядок минимальную пару координат. То есть из двух пар координат берется та, где первая координата меньше, при равных первых — вторая. Например, (2, 5) лучше, чем (2, 7), но (1, 8) лучше обеих этих пар.

Для решения этой задачи у команды есть 20 попыток.

Ограничение по времени выполнения программы: 1 секунда.

Ограничение по памяти: 256 МВ.

Подзадача 2. Задача остается та же (в том числе ограничение по времени), но теперь матрица имеет размер 400 на 400 секторов, а требуемый участок — 100 на 100 секторов.

Для решения этой задачи у команды есть 20 попыток.

Ограничение по времени выполнения программы: 1 секунда.

Ограничение по памяти: 512 МВ.

Решение

Разберем сразу вторую подзадачу, т. к. она является более строгой модификацией первой, требующей оптимальное решение. При том, что первая допускает наивное решение, а именно цикл по матрице с суммированием, оно имеет вычислительную сложность $O(N^2M^2)$. Для оптимизации решения необходимо заметить, что при вычислении суммы подматрицы элементы в середине суммируются многократно, хотя фактически при сдвиге матрицы на ячейку влево или вправо мы можем просто вычесть старые элементы (оказавшиеся за пределами) и прибавить новые. На этом основан авторский алгоритм динамического программирования.

Сначала мы выполняем проход по горизонталям матрицы, вычисляя суммы по строкам. При этом для большей оптимизации мы буквально прибавляем новый элемент и вычитаем старый. На выходе будет матрица N на $(N - M + 1)$, в каждой ячейке которой хранится сумма M элементов по горизонтали.

Следующий проход выполняется аналогично в вертикальном направлении. Так, суммируя горизонтальные суммы по вертикали, на выходе мы получим матрицу $(N - M + 1)$ на $(N - M + 1)$, содержащую в каждой ячейке искомые суммы подматриц.

Остается пройти по этой матрице последний раз для поиска максимума и вывести его координату в качестве ответа.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1  n, m = 400, 100
2  data = [[float(x) for x in input().split()] for l in range(n)]
3  hors = [[0 for _ in range(n-m+1)] for _ in range(n)]
4  for x in range(n):
5      s = hors[x][0] = sum(data[x][:m])
6      for y in range(m, n):
7          s = hors[x][y-m+1] = s - data[x][y-m] + data[x][y]
8  vers = [[0 for _ in range(n-m+1)] for _ in range(n-m+1)]
9  for x in range(n-m+1):
10     s = vers[0][x] = sum(hors[i][x] for i in range(m))
11     for y in range(m, n):
12         s = vers[y-m+1][x] = s - hors[y-m][x] + hors[y][x]
13     (_, (x, y)) = min((-vers[x][y], (x,y)) for x in range(n-m+1) for y in
14         ↪ range(n-m+1))
14     print(f"{x} {y}")

```

Задача II.2.3. Слава победителя (93 баллов)

Темы: теория игр.

Эта задача проверяет ваш навык программной обработки данных и работы с теоретико-игровыми моделями, что потребуется при работе с финальной задачей. Помимо этого, задача использует ряд понятий из теории аукционов. Подробнее о них вы можете узнать в модуле 3 учебного курса <https://onti.polyus-nt.ru/course/view.php?id=2>.

Условие

На аукционе с N участниками выставляется M лотов. Все участники являются **покупателями**. Каждый участник по-разному может использовать выигранный лот, чтобы извлечь из него свою экономическую выгоду. Каждый лот выставляется на аукцион **второй цены**. Зная ценность каждого лота для каждого участника, найдите того, кто получит **наибольшую выгоду** от участия в аукционе.

Формат входных данных

В первой строке через пробел указаны целые N и M . Затем идет N строк, в каждой из которых приведено M вещественных чисел через пробел — ценности лотов для участника (первая строка — для участника под номером 0, вторая — 1 и т. д.). Пример формата для 5 лотов и 4 участников:

```

4 5
1.1 1.9 3.15 3.8 4.5
1.2 1.8 3.18 3.3 4.6
1.3 1.7 3.13 3.2 4.3
1.4 1.6 3.11 3.5 4.2

```

Формат выходных данных

Единственное целое число, номер участника с **наибольшей** выгодой (**начиная с 0**). Решение принимается, если выгода выбранного вами участника больше выбранного авторским решением, либо отличается меньше, чем на 10^{-6} .

Для решения этой задачи у команды есть 20 попыток.

Ограничение по времени выполнения программы: 1 секунда.

Ограничение по памяти: 256 МВ.

Решение

Задача является идейным продолжением «Проклятия победителя» из индивидуального модуля. В отличие от той задачи, здесь отсутствуют истинные ценности лотов, но используется аукцион второй цены. Если обратиться к разделу тематического курса о теории аукционов, можно определить, что выгода победителя здесь — разница между его ставкой и фактически выплаченной ценой (то есть ставкой следующего за победителем).

Зная это, мы можем смоделировать аукцион, отсортировав ставки по каждому из лотов и просуммировав для соответствующих победителей их выгоды согласно вышеупомянутой метрики. После чего остается найти в максимум в массиве суммарных выгод и вывести индекс.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1  players, lots = map(int, input().split())
2  A = [[float(y) for y in input().split()] for x in range(players)]
3
4  profits = [0 for _ in range(players)]
5  for lot in range(lots):
6      winners = sorted((A[p][lot], p) for p in range(players))
7      winner = winners[-1][1]
8      profits[winner] += winners[-1][0] - winners[-2][0]
9
10 (_, best) = max((profits[p1], p1) for p1 in range(players))
11 print(best)

```

Задача II.2.4. Электрическая инвентаризация (135 баллов)

Темы: графы, математические модели.

Эта задача тренирует навыки работы с графами и данным, что потребуется в решении финальной задачи.

Условие

Сетевая компания «Энергия хаоса» собирается модернизировать оснащение одной из своих подсетей. Эта сеть имеет вид **дерева**, состоящего из подстанций и просьюмеров. Каждый **просьюмер** обладает индивидуальным суточным **паттерном**

потребления и генерации (для простоты даны средние значения на каждый час). Просьюмеры подключаются к **подстанциям**. Подстанции не обладают собственной мощностью, но занимаются **диспетчеризацией** мощностей с дочерних узлов (в т. ч. других подстанций), а именно: подстанция самостоятельно **балансирует** мощности дочерних узлов, после чего запрашивает недостающую мощность из внешней для себя сети или передает туда избыточную через **входную линию**, направленную к корню дерева, а именно, главной подстанции. Входная линия **главной подстанции** ведет к внешнему поставщику.

Для модернизации подстанций специалистам необходимо вычислить **максимальную абсолютную мощность**, которая проходит через входную линию подстанции в течение суток. Напишите **программу**, выполняющую этот расчет для всех подстанций сети.

Формат входных данных

Два списка. Первый — список паттернов мощностей узлов. Индекс в этом списке соответствует номеру узла (**нумерация с 0**). Для подстанций вместо графика записан `None`, для остальных узлов — список из 24 вещественных чисел, мощности узла в определенный час суток (положительное число — генерация, отрицательное — потребление). Второй — список ребер дерева (до 64), представленных **неупорядоченными парами** номеров узлов (которые соответствуют индексу в первом списке). **Главная подстанция** имеет индекс 0. Все эти данные представлены в виде строки, которую можно проинтерпретировать командой `eval`. Пример входных данных:

```
([None, [-12, -11, -10, -9.5, -8, -7, -6, -5, -4, -3, -2, -1, 1, 2, 3, 4, 5, 6, 7, 8, 9.5, 10, 11, 12], None, [-12, -11, -10, -9.5, -8, -7, -6, -5, -4, -3, -2, -1, 1, 2, 3, 4, 5, 6, 7, 8, 9.5, 10, 11, 12], [-12, -11, -10, -9.5, -8, -7, -6, -5, -4, -3, -2, -1, 1, 2, 3, 4, 5, 6, 7, 8, 9.5, 10, 11, 12]], [(0, 1), (2, 0), (2, 3), (4, 2)])}
```

Формат выходных данных

Число строк, соответствующее количеству подстанций. В каждой строке два числа через пробел: номер узла-подстанции и максимальная **абсолютная** мощность. Порядок вывода подстанций может быть **любой**. Пример формата:

```
1 2.1
4 5.24
30 10.3
```

Решение **засчитывается**, если вычисленная мощность подстанций различается с авторским решением **не более**, чем на 10^{-7} .

Для решения этой задачи у команды есть 20 попыток.

Ограничение по времени выполнения программы: 2 секунды.

Ограничение по памяти: 256 МВ.

Решение

Если выделить модель задачи, мы получаем следующее: просуммировать векторы (а мощности представляют собой вектор из 24 элементов) от листьев в корню и вывести максимальные по модулю элементы векторов для узлов, не являющихся листьями.

Для этого считываем ребра и строим списки смежности для искомой сети. Затем запускаем алгоритм перебора в глубину (DFS), который возвращает вектор для листа, а для остальных узлов суммирует выводы с дочерних узлов, записывает максимальный по модулю элемент вектора в словарь-ответ и передает этот вектор родительскому узлу.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1  pows, egs = eval(input())
2  adjs = dict()
3
4  def vsum(vs):
5      return [sum(x) for x in zip(*vs)]
6
7  def dfs(idx, parent):
8      e = adjs[idx]
9      if idx in sol:
10         pw = vsum(dfs(i, idx) for i in e if i != parent)
11         sol[idx] = max(abs(x) for x in pw)
12         return pw
13     else:
14         return pows[idx]
15
16 for (a, b) in egs:
17     adjs.setdefault(a, set()).add(b)
18     adjs.setdefault(b, set()).add(a)
19 sol = {k: None for k, v in adjs.items() if len(v) > 1}
20
21 dfs(0, None)
22
23 print("\n".join(f"{k} {v}" for (k, v) in sol.items()))

```

Задача II.2.5. Nim Online (100 баллов)

Темы: теория игр, программирование.

Эта задача проверяет ваш навык промышленного программирования и знания теории игр, что потребуется в решении финальной задачи.

Условие

Мы предлагаем сыграть вам в классическую игру **Ним**. Правила ее просты: есть несколько кучек, в каждой лежит некоторое число камней (оно может различаться). Два игрока поочередно берут **ненулевое** число камней из **одной** кучки на свой выбор. **Выигрывает** тот, кто забирает последний камень.

ем Ним-сумму из элемента, где это возможно (совпадение по битам), либо определяем желаемый элемент (применяя Ним-сумму через XOR), и если мы можем получить его вычитанием, забираем нужное число камней. Эти варианты возможны всегда из выигрышного положения, поэтому гарантированно приводят к победе.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1  import functools
2  import operator
3  import random
4  import socket
5
6  KEY = "0" * 67
7
8  def make_move(pos):
9      nims = functools.reduce(operator.xor, pos)
10     straights = [(x, i) for (i, x) in enumerate(pos) if x & nims == nims]
11     if straights:
12         return (max(straights)[1], nims)
13     for (i, x) in enumerate(pos):
14         if nims ^ x < x:
15             return (i, x - (nims ^ x))
16     raise Exception("I can't lose!")
17
18 with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
19     sock.connect(("0xdeadbeef.ru", 1901))
20
21     print(KEY)
22     sock.sendall(bytes(KEY + "\n", "utf-8"))
23
24     active = True
25     while active:
26         lines = sock.recv(512).decode().splitlines()
27         for l in lines:
28             print("<", l)
29             if l.startswith("KEY "):
30                 print(l[4:])
31                 active = False
32                 break
33             if l.startswith("START ") or l.startswith("NEXT "):
34                 can_drop = l.startswith("START ")
35                 pos = [int(x) for x in l.split()[2:]]
36                 nims = functools.reduce(operator.xor, pos)
37                 if nims == 0:
38                     if can_drop:
39                         print("> X")
40                         sock.sendall(b"X\n")
41                     else:
42                         raise Exception("derp")
43             else:
44                 k, n = make_move(pos)
45                 print(">", k, n)
46                 sock.sendall(f"{k} {n}\n".encode())
47             continue
48         print(l)
49         active = False
50     break

```

Задача П.2.6. Предсказатель Плюс (40 баллов)

Темы: алгоритмы, анализ данных.

Эта задача проверяет ваш навык программного анализа данных, что является частью финальной задачи.

Это продолжение задачи «Сам себе предсказатель» из индивидуально-го модуля. Предполагается, что вы знакомы с ее условием.

Условие

Заметив, что вам удалось совладать с **одним** набором данных, программист-радиоэлектронщик Саша Текстолитов, наконец, поручил вам реализацию модуля, на основе исторических данных подбирающего значение скорости разряда в зависимости от температуры. «Тем более, наверняка вы без дела не сидели», — добавил Саша, хитро улыбаясь.

Условие задачи не изменилось: есть некоторая **зависимость** Y от X , график которой непрерывен. Для этой зависимости известно некоторое количество **точек**, равномерно распределенных в пределах области определения. **Вычислите** с максимально возможной точностью значение Y' по известным аргументам X' . Гарантируется, что значения аргументов X' находятся в области определения исходной зависимости.

Формат также остался прежним. Но **будьте внимательнее**, точность вычисляемых значений должна быть в пределах 0,5%.

Формат входных данных

Первая строка — два числа через пробел, N известных точек и M аргументов для вычисления. После идет N строк, в каждой через пробел приводятся X_i и Y_i , аргумент и соответствующее значение величины. После идет M строк, в каждой единственное число X'_j , аргумент, который необходимо вычислить. Пример входных данных:

```
4 3
0 0
1 1
2 2
3 3
1.5
1.75

0
```

Формат выходных данных

M строк, в каждой — соответствующее значение Y'_j величины для перечисленных аргументов. Ответ принимается, если максимальная погрешность всех значений не превышает 0,5%.

Для решения этой задачи у команды есть 20 попыток.

Ограничение по времени выполнения программы: 2 секунды.

Ограничение по памяти: 256 МВ.

Решение

Задача является прямым продолжением «Сам себе предсказатель» из индивидуального модуля, и несмотря на меньший допуск в 0,5%, решается идентично: написанием программы, которая выполняет бинарный поиск ближайшего значения X и считает линейную интерполяцию на двух ближайших к X' точках:

$$Y' = Y_1 + (Y_2 - Y_1) \frac{X' - X_1}{X_2 - X_1}.$$

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 from bisect import bisect_left
2 N, M = map(int, input().split())
3 pts = [tuple(float(x) for x in input().split()) for _ in range(N)]
4 pts.sort()
5 xt, yt = zip(*pts)
6 xs = [float(input()) for _ in range(M)]
7 ys = []
8 for x in xs:
9     i = bisect_left(xt, x)
10    ys.append(yt[i] + (x - xt[i]) * (yt[i+1] - yt[i]) / (xt[i+1] - xt[i]))
11 print("\n".join(map(str, ys)))

```

Задача II.2.7. По краю пропасти (150 баллов)

Темы: теория вероятностей, математика.

Эта задача проверяет ваш навык работы с математическими вероятностными моделями, что потребуется в решении финальной задачи.

Условие

Подзадача 1. Тренер готовит спортсмена к важным **соревнованиям** (не по шахматам) и составляет ему **план тренировок на неделю**. У тренера есть **десять** разных типов проведения тренировки (в том числе отдых, который на самом деле является важным видом тренировки), из которых необходимо назначить **по одной** на каждый день недели (свободных дней не предусмотрено).

Каждая из тренировок несет свой вклад в вероятную победу, представленный условной «**полезностью**». Но эта «полезность» растет **нелинейно**: если повторять тренировку одного и того же типа, она будет приносить меньше пользы. А именно, **общая** «полезность» N тренировок конкретного типа в неделю будет равна $c_i \cdot \sqrt{N}$, где c_i — коэффициент полезности для этого типа. Очевидно, общая «полезность» тренировок по плану равна сумме полезностей по выбранным типам.

Но кроме «полезности», для каждого вида тренировки есть **вероятность** получить травму, из-за которой спортсмен пропустит соревнования (а все тренировки пройдут впустую).

Определите такой план тренировок (сколько раз в неделю проводится та или иная тренировка), чтобы максимизировать **математическое ожидание** суммарной «полезности» и, соответственно, шансы на победу.

Формат входных данных

10 строк, в каждой два вещественных числа через пробел. Первое число — вероятность (доля единицы) получить травму во время одной тренировки. Второе — коэффициент полезности c_i .

Формат выходных данных

10 строк, в каждой — сколько раз назначается соответствующий тип тренировки (целое число).

Решение принимается, если ваш план тренировок обеспечит **такое же или большее** математическое ожидание полезности, чем представленное авторским решением.

Для решения этой задачи у команды есть 20 попыток.

Ограничение по времени выполнения программы: 5 секунды.

Ограничение по памяти: 256 МВ.

Подзадача 2. Теперь тренер составляет более подробный план для **одной отдельной тренировки**. Она длится 4 часа и состоит из **упражнений**, на каждое из которых отводится по 15 минут. Всего упражнений 10 видов (в т. ч. отдых), и они имеют те же характеристики, что и тренировки в первой подзадаче. Необходимо определить план полностью, то есть суммарное число упражнений соответствует числу 15-ти минутных слотов.

Определите **оптимальный** план упражнений. Форматы данных и принцип проверки остаются прежними.

Для решения этой задачи у команды есть 20 попыток.

Ограничение по времени выполнения программы: 1 секунда.

Ограничение по памяти: 256 МВ.

Решение

Из условия видно, что любая травма обнуляет итоговую полезность, поэтому единственное ненулевое событие — полное отсутствие травм (т. е. одновременное наступление события «не получил травму» для всех тренировок). Зная это, а также формулу полезности, определим формулу математического ожидания суммарной полезности:

$$M = \prod (1 - p_i)^{N_i} \cdot \sum (c_i \sqrt{N_i}).$$

Как можно заметить, перебор всех возможных комбинаций неоптимален, т. к. их число растет по экспоненте (соответственно 10^7 и 10^{16}). Тем не менее, мы можем выполнить оптимальный перебор с помощью вышеуказанной формулы: поочередно

пробуем добавить в текущую комбинацию тренировку/упражнение каждого из 10 видов и вычислить новое матожидание, после чего выбираем вариант с наибольшим матожиданием. Повторив это 7 (или 16) раз, мы гарантированно получаем оптимальный план.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
1 import functools, operator
2 from math import sqrt
3
4 SLOTS = 7 # 16 во второй подзадаче
5
6 def reel_with(params, vs, idx):
7     n = len(params)
8     vs[idx] += 1
9     util = sum(params[i][1] * sqrt(vs[i]) for i in range(n))
10    prob = functools.reduce(operator.mul, ((1-params[i][0])**vs[i]
11                                           for i in range(n)))
12    vs[idx] -= 1
13    return util * prob
14
15 ns = [tuple(map(float, l))
16       for l in map(str.split, [input() for _ in range(10)])]
17 n = len(ns)
18 vs = [0 for _ in range(n)]
19 for _ in range(SLOTS):
20     ni = max((reel_with(ns, vs, i), i) for i in range(n))[1]
21     vs[ni] += 1
22
23 print("\n".join(str(x) for x in vs))
```

Заключительный этап

Индивидуальный предметный тур

Информатика. 8–11 класс

Задача III.1.1.1. Фигура с битыми пикселями (10 баллов)

Дана матрица состоящая из 0 и 1. Значениями 1 в матрице нарисована фигура (см. рисунок III.1.1, единицы — белые клетки). Фигура может быть «испорчена».

Испорченность определяется по следующему правилу: если в последовательности из трех (по вертикали или горизонтали) соседних значений не хватает значения в среднем положении, т. е. значение равно нулю, то в этом месте пропущена единица.

Пропущенных единиц может быть несколько. Необходимо определить площадь фигуры с учетом испорченных значений. Площадь фигуры равна количеству единиц, из которых состоит фигура.

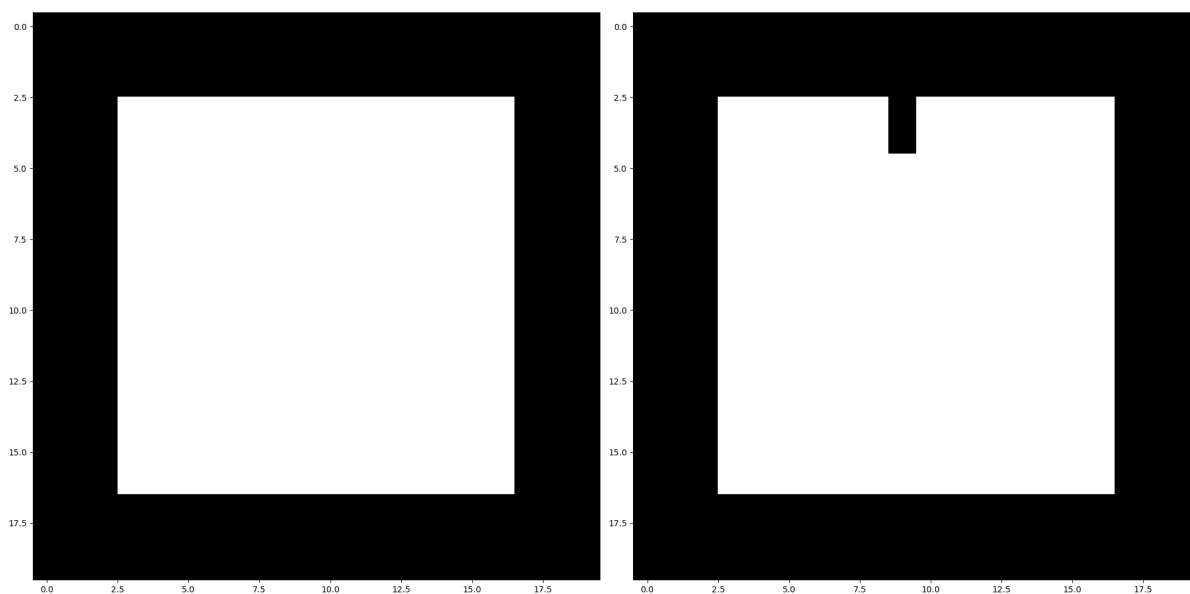


Рис. III.1.1. Пример входных изображений. Слева фигура не испорчена и имеет площадь 196. Справа фигура испорчена, отсутствуют две единицы, площадь после исправления равна 196

Формат входных данных

Матрица размером 20 на 20. В строках заданы значения матрицы по строкам и столбцам. Фигура не соприкасается с границами матрицы.

Формат выходных данных

Площадь фигуры в виде одного целого числа.

Примеры

Пример №1

Стандартный ввод
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0
0 0
0 0
Стандартный вывод
196

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 def area(matrix):
2     result = 0
3     for i in range(1, len(matrix) - 1):
4         for j in range(1, len(matrix[0]) - 1):
5             if matrix[i][j] == 1:
6                 result += 1
7             elif matrix[i-1][j] == 1 and matrix[i+1][j] == 1:
8                 result += 1
9             elif matrix[i][j-1] == 1 and matrix[i][j+1] == 1:
10                result += 1
11     return result
12
13
14 matrix = []
15
16 for i in range(20):
17     line = map(int, input().split())

```

```
18     matrix.append(list(line))
19
20     print(area(matrix))
```

Задача III.1.1.2. Подсчет квадратов (20 баллов)

Дана матрица, состоящая из 0 и 1. Значениями 1 в матрице нарисован квадрат (см. пример на рисунке III.1.2). Внутри квадрата есть квадратные отверстия. Определить их количество.

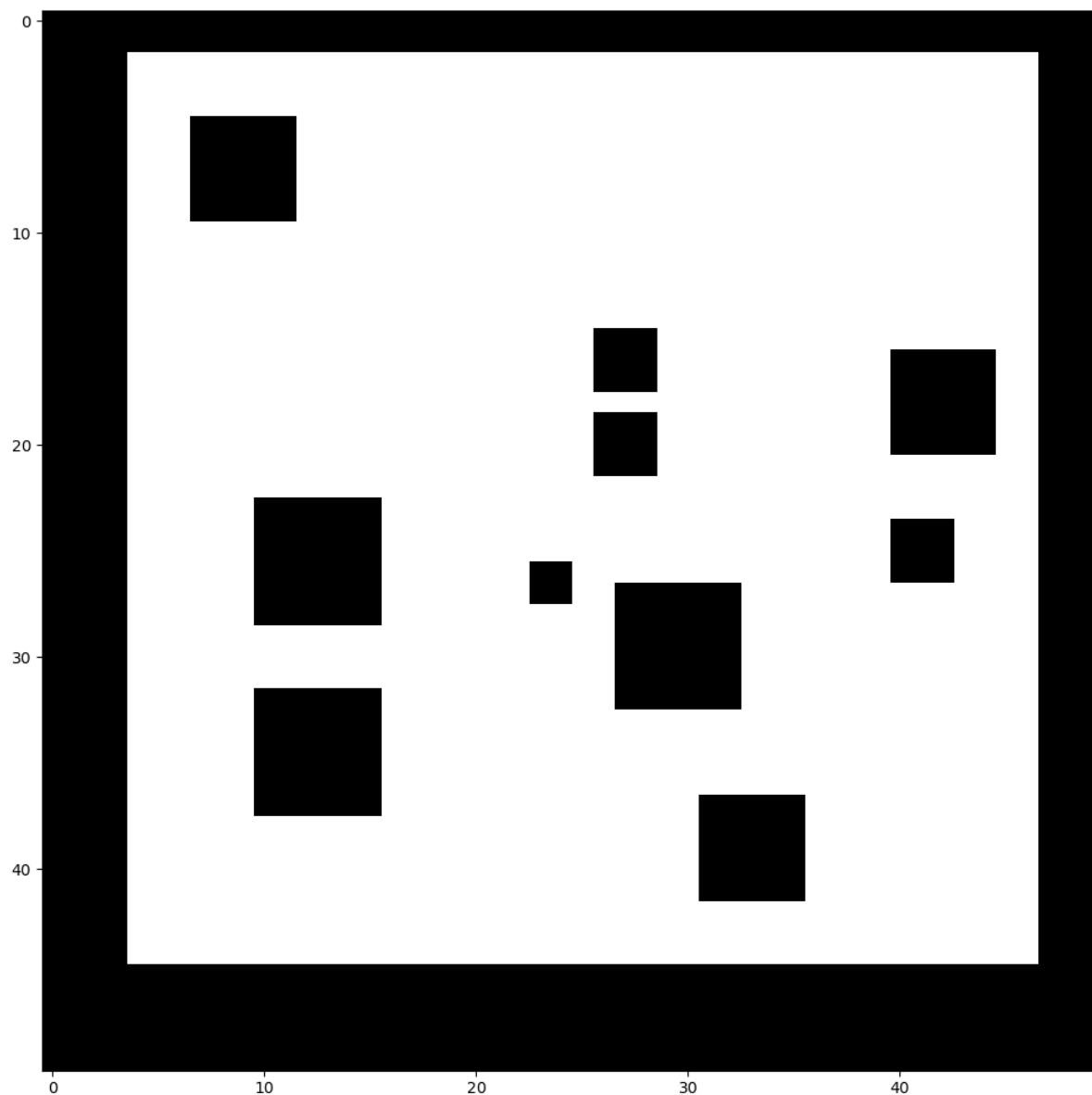


Рис. III.1.2. Пример входного квадрата. Внутри квадрата 10 отверстий

Формат входных данных

Матрица размером 50 на 50. В строках заданы значения матрицы по строкам и столбцам. Квадрат не соприкасается с границами матрицы.

Формат выходных данных

Количество квадратных отверстий внутри квадрата.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
1  corners = [  
2      [[1, 1],[1, 0]],  
3      [[1, 1],[0, 1]],  
4      [[0, 1],[1, 1]],  
5      [[1, 0],[1, 1]],  
6  ]  
7  
8  def count_rects(matrix):  
9      count = 0  
10     for i in range(1, len(matrix)-1):  
11         for j in range(1, len(matrix[0])-1):  
12             region = [row[j-1:j+1] for row in matrix[i-1:i+1]]  
13             if region in corners:  
14                 count += 1  
15     return count // 4  
16  
17  matrix = []  
18  
19  for i in range(50):  
20      line = map(int, input().split())  
21      matrix.append(list(line))  
22  
23  print(count_rects(matrix))
```

Задача III.1.1.3. Найди монетки (25 баллов)

Дана матрица, состоящая из 0 и 1. Значениями 1 в матрице нарисованы монетки (см. пример на рисунке III.1.3). Форма монетки может быть любой. Необходимо определить количество и сумму всех монеток для заданного номинала.

Номинал линейно зависит от размера монетки, чем больше монета, тем больше номинал.

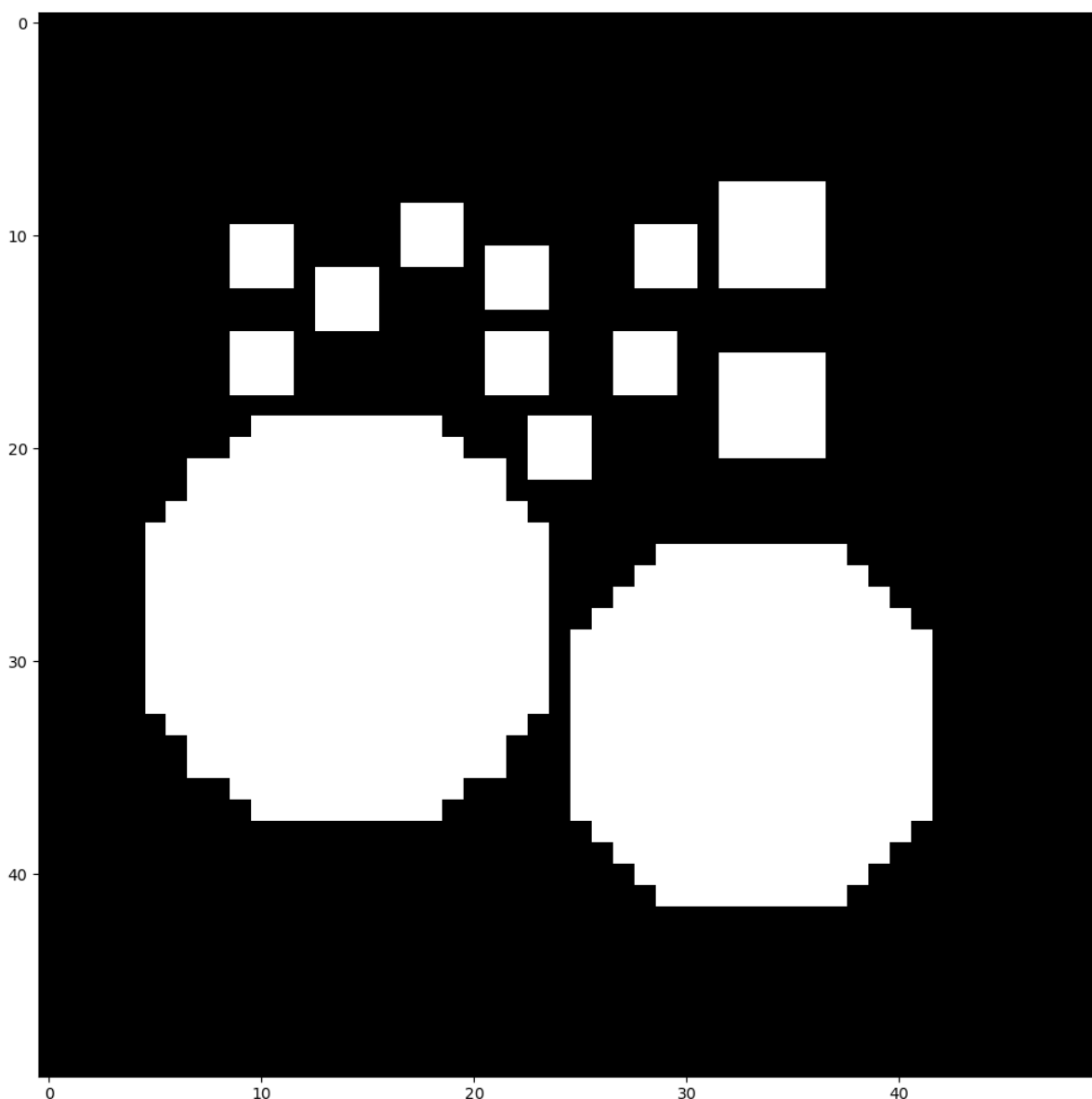


Рис. III.1.3. Пример входного квадрата

Формат входных данных

В первой строке через пробел заданы высота h и ширина w матрицы ($50 \leq (h, w) \leq 200$).

Во второй строке заданы номиналы монет (количество номиналов от 3 до 7, значение каждого номинала в пределах от 3 до 20), количество номиналов соответствует количеству размеров монет. В следующих строках заданы значения матрицы по строкам и столбцам.

Монетки отстоят друг от друга минимум на одно значение.

Формат выходных данных

Количество монет и их сумма разделенные пробелом (целые числа).

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1  def neighbors(i, j):
2      return ((i - 1, j),
3              (i + 1, j),
4              (i, j - 1),
5              (i, j + 1))
6
7  def _label(matrix, i, j, label):
8      matrix[i][j] = label
9      for ni, nj in neighbors(i, j):
10         if matrix[ni][nj] == -1:
11             _label(matrix, ni, nj, label)
12
13 def label_matrix(matrix):
14     for i in range(len(matrix)):
15         for j in range(len(matrix[0])):
16             matrix[i][j] *= -1
17     label = 0
18     for i in range(len(matrix)):
19         for j in range(len(matrix[0])):
20             if matrix[i][j] == -1:
21                 label += 1
22                 _label(matrix, i, j, label)
23     return label, matrix
24
25 def count_area(labeled, label):
26     area = 0
27     for i in range(len(labeled)):
28         for j in range(len(labeled[0])):
29             if labeled[i][j] == label:
30                 area += 1
31     return area
32
33
34 def sum_coins(matrix, nominals):
35     max_label, labeled = label_matrix(matrix)
36     areas = {}
37     for label in range(1, max_label+1):
38         area = count_area(labeled, label)
39         if area not in areas:
40             areas[area] = 0
41         areas[area] += 1
42     sum_ = 0
43     for i, key in enumerate(sorted(areas)):
44         sum_ += nominals[i] * areas[key]
45     return max_label, sum_
46
47 h, w = map(int, input().split())
48 nominals = sorted(map(int, input().split()))
49
50 matrix = []
51
52 for i in range(h):
53     line = map(int, input().split())
54     matrix.append(list(line))
55
56 print(*sum_coins(matrix, nominals))

```

Задача III.1.1.4. Поиск дорог (30 баллов)

На двумерном поле размером 30 на 30 заданы расположения ключевых точек и дороги, соединяющие их (см. рисунок III.1.4).

Ключевая точка представляет собой знак «плюс», т. е. состоит из 5 значений: одного центрального и четырех значений вокруг: сверху, снизу, слева и справа. Необходимо по заданным координатам определить ключевые точки и суммарное расстояние по всем дорогам, соединяющим эти точки. Центры ключевых точек входят в суммарное расстояние.

Тупики при определении расстояния не учитываются. Тупиком является любая дорога, которая не оканчивается с двух сторон ключевой точкой. Дорога в ключевую точку может заходить только сверху, снизу, слева или справа от центра.

Дороги могут изгибаться только под прямыми углами. Ширина дороги всегда равна 1. Центры ключевых точек отстоят друг от друга минимум на 1 значение. Если ключевая точка соединяется сама с собой дорогой, то такая дорога учитывается для подсчета расстояния. Две ключевые точки могут быть связаны больше, чем одной дорогой.

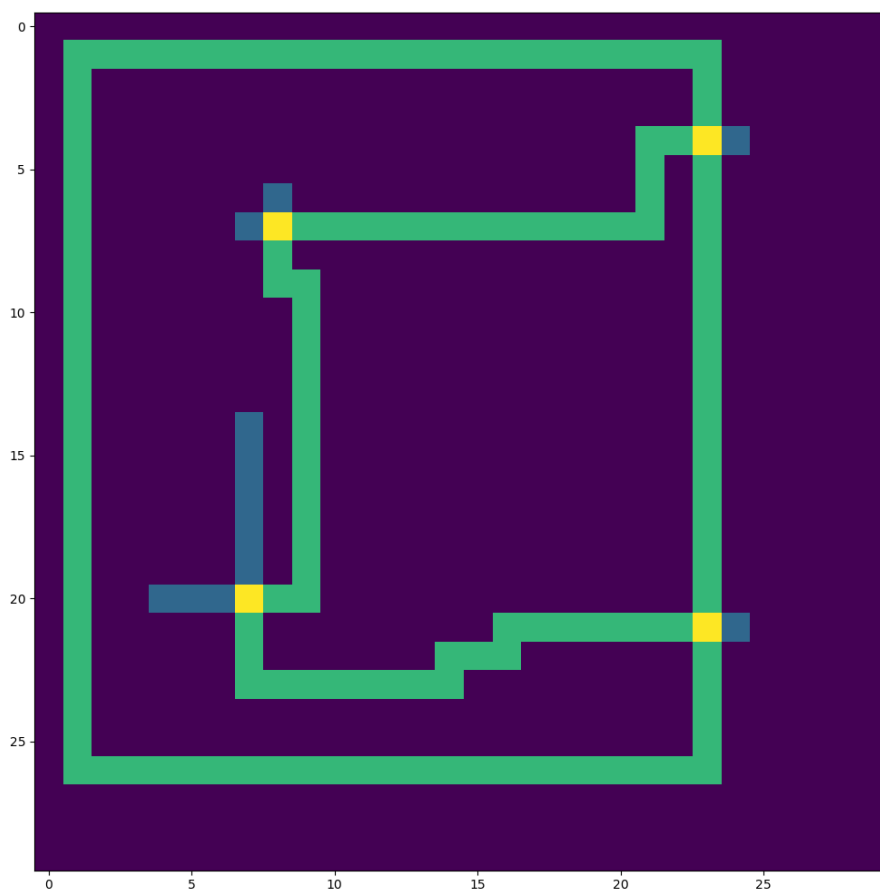


Рис. III.1.4. Иллюстрация входных данных. На изображении четыре ключевые точки. Желтым цветом показаны центры ключевых точек. Светло-зеленым цветом показаны дороги, соединяющие ключевые точки. Темно-зеленым цветом показаны тупики

Формат входных данных

В первой строке задано количество координат, которыми заданы ключевые точки и дороги. В следующих строках заданы координаты в виде номера строки и столбца разделенных пробелом.

Формат выходных данных

Количество ключевых точек и суммарное расстояние разделенные пробелом. Если ключевых точек нет, то расстояние равно 0.

Примеры*Пример №1*

Стандартный ввод
23
13 7
13 23
14 6
14 7
14 8
14 9
14 10
14 11
14 12
14 13
14 14
14 15
14 16
14 17
14 18
14 19
14 20
14 21
14 22
14 23
14 24
15 7
15 23
Стандартный вывод
2 17

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 def neighbours(i, j):
2     return ((i - 1, j),
3            (i + 1, j),
4            (i, j - 1),
```

```
5         (i, j + 1))
6
7 def is_key_point(nbs, matrix):
8     for nb in nbs:
9         if not matrix[nb[0]][nb[1]] == -1:
10            return False
11    return True
12
13 def get_list_of_key_points(matrix):
14    key_points = []
15    for i in range(1, len(matrix)-1):
16        for j in range(1, len(matrix[0])-1):
17            nbs = neighbours(i, j)
18            if is_key_point(nbs, matrix):
19                key_points.append((i, j))
20    return key_points
21
22
23 def set_matrix_values(matrix, value, points):
24    for p in points:
25        matrix[p[0]][p[1]] = value
26
27 def search_roads(key_point, matrix):
28    roads = []
29    nbs = neighbours(*key_point)
30    for nb in nbs:
31        road = []
32        _search(key_point, nb, matrix, road)
33        if road:
34            if road[-1] != (-1, -1):
35                roads.append(road)
36    return roads
37
38 def _search(nb_prev, nb, matrix, road):
39    if matrix[nb[0]][nb[1]] == -1:
40        road.append(nb)
41        matrix[nb[0]][nb[1]] = 1
42        nbs = [_nb for _nb in neighbours(*nb) if _nb != nb_prev]
43        vals = [matrix[nb[0]][nb[1]] == 0
44                for nb in nbs if matrix[nb[0]][nb[1]] != 1]
45        if all(vals):
46            road.append((-1, -1))
47            return
48        for _nb in nbs:
49            _search(nb, _nb, matrix, road)
50
51 def search_all_roads(key_points, matrix):
52    roads = []
53    for p in key_points:
54        for road in search_roads(p, matrix):
55            roads.extend(road)
56    return roads
57
58 n = int(input())
59 points = []
60
61 for _ in range(n):
62     points.append(tuple(map(int, input().split())))
63
64 matrix = [[0 for i in range(30)] for j in range(30)]
```

```

65
66 set_matrix_values(matrix, -1, points)
67
68 key_points = get_list_of_key_points(matrix)
69
70 set_matrix_values(matrix, 3, key_points)
71
72 ln = len(key_points)
73
74 if ln == 0:
75     print(0, 0)
76 else:
77     all_roads = search_all_roads(key_points, matrix)
78     print(ln, ln + len(all_roads))

```

Задача III.1.1.5. Облака точек (15 баллов)

В трехмерном пространстве «разбросаны» облака точек (см. левый рисунок). Каждое облако состоит из 10 точек. Облаков в пространстве может быть несколько (от 1 до 10) и при этом каждое облако отдалено от другого на значительное расстояние.

Необходимо определить, какое суммарное количество точек попадет в сферу заданного радиуса, помещенную в центр каждого облака (см. правый рисунок). В качестве центра облака выступает точка со средними координатами по всем осям для всех точек, входящих в это облако. Синей точкой обозначен рассчитанный центр облака. Зеленым цветом обозначены точки внутри сферы, красным — снаружи.

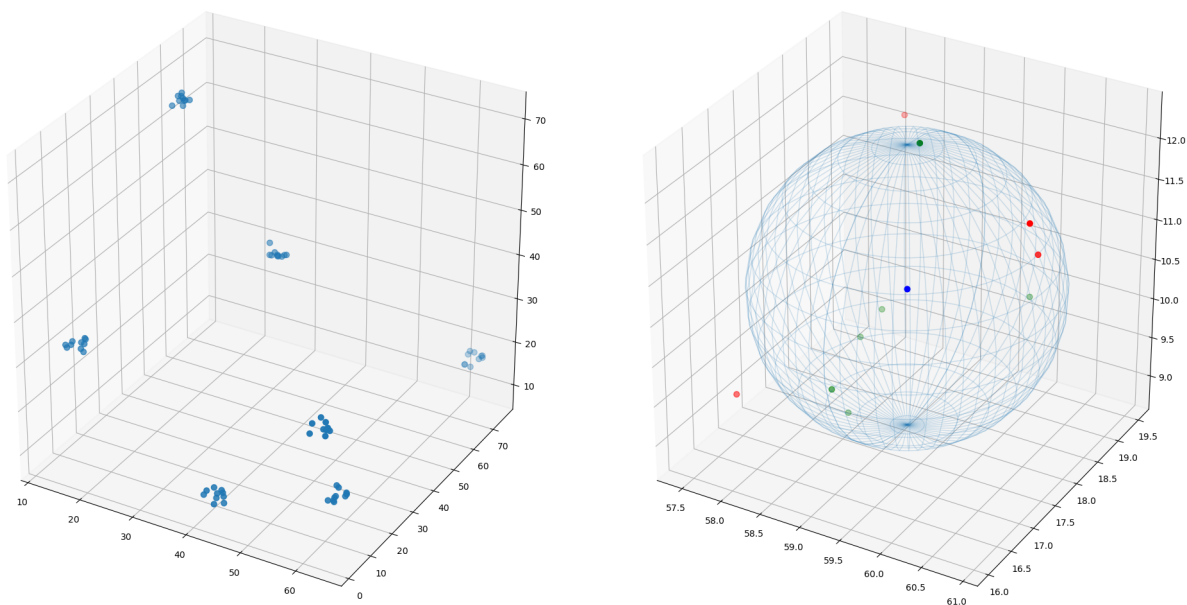


Рис. III.1.5. В левой части рисунка пример «разбросанных» облаков точек. В правой части рисунка пример одного облака и сферы, помещенной в центр этого облака. Синей точкой обозначен рассчитанный центр облака. Зеленым цветом обозначены точки внутри сферы, красным — снаружи.

Формат входных данных

В первой строке через пробел задано целым числом количество точек и вещественным числом радиус сферы. В следующих строках через пробел заданы координаты каждой точки в виде трех вещественных координат x , y , z .

Координаты точек облаков перемешаны.

Формат выходных данных

Суммарное количество всех точек по всем облакам, которые попали в сферу заданного радиуса.

Примеры

Пример №1

Стандартный ввод
50 1.25
48.61 35.57 80.40
52.76 16.70 41.63
78.65 46.25 28.49
21.57 16.84 57.36
54.67 14.42 37.65
52.38 37.08 81.00
47.75 34.61 77.96
53.72 12.98 38.99
28.67 16.14 80.86
29.00 14.87 78.83
18.00 16.20 55.69
48.97 36.80 79.98
49.09 35.92 80.23
27.55 15.46 80.33
80.02 46.58 29.21
28.05 12.54 79.68
19.55 12.74 54.25
20.18 15.43 54.71
80.76 45.63 31.44
82.03 45.31 27.76
30.80 12.67 78.96
29.89 14.30 79.75
50.21 32.87 81.24
53.54 16.00 40.92
20.56 17.35 55.78
54.32 13.88 39.87
50.28 34.79 79.67
20.34 13.30 52.95
50.49 35.89 80.18
54.03 15.57 38.71

Стандартный ввод
53.83 15.35 40.04
79.53 45.31 29.47
28.09 16.33 81.91
18.61 17.38 53.86
81.87 46.60 30.62
54.91 16.67 40.27
31.92 15.95 81.02
78.53 44.57 29.42
81.93 44.08 30.49
56.52 12.85 42.17
82.01 47.04 28.68
22.08 16.23 57.28
28.88 13.60 81.12
53.44 17.41 42.36
18.21 15.56 53.03
50.02 35.15 82.41
29.74 14.74 81.98
52.02 36.58 82.47
21.81 16.78 56.31
82.39 43.55 29.88
Стандартный вывод
7

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 def distance(p1, p2):
2     dist = (p1[0] - p2[0]) ** 2 + (p1[1] - p2[1])\
3         ** 2 + (p1[2] - p2[2]) ** 2
4     return dist ** 0.5
5
6
7 def find_cloud(points):
8     ps = points.pop()
9     dists = {}
10    for pe in points:
11        dists[pe] = distance(ps, pe)
12    dists = sorted(dists.items(), key=lambda x: x[1])
13    cloud = []
14    cloud.append(ps)
15    for ds in dists[:9]:
16        cloud.append(ds[0])
17        points.remove(ds[0])
18    return cloud
19
20 def find_all_clouds(points):
21    clouds = []
22    while points:
23        clouds.append(find_cloud(points))
24    return clouds
25
26 def find_inside_points(clouds, radius):
27    n = 0

```

```
28     for cloud in clouds:
29         xs = []
30         ys = []
31         zs = []
32         for p in cloud:
33             xs.append(p[0])
34             ys.append(p[1])
35             zs.append(p[2])
36
37         ln = len(xs)
38         xc = sum(xs) / ln
39         yc = sum(ys) / ln
40         zc = sum(zs) / ln
41
42         for p in cloud:
43             if distance(p, [xc, yc, zc]) < radius:
44                 n += 1
45
46     return n
47
48
49 s = input().split()
50 n = int(s[0])
51 radius = float(s[1])
52
53 coords = []
54 for _ in range(n):
55     coords.append(tuple(map(float, input().split())))
56
57 clouds = find_all_clouds(coords)
58 print(find_inside_points(clouds, radius))
```

Математика. 8–9 классы

Задача III.1.2.1. (30 баллов)

Логотип компании состоит из двух пересекающихся окружностей с одинаковыми радиусами. Первая окружность описана около прямоугольного равнобедренного треугольника, а вторая касается его катетов. Найдите площадь непересекающихся областей окружностей, если площадь треугольника равна $S_{\Delta} = 50 \text{ см}^2$. Ответ округлить до целого числа квадратных сантиметров.

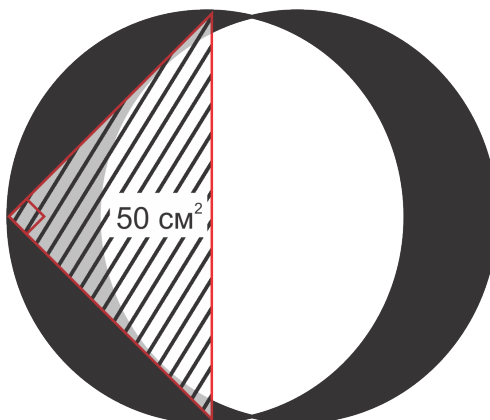


Рис. III.1.6. Обозначения и дополнительные построения к задаче III.1.2.1

Критерии оценивания

- Получено выражение для $\sin\alpha$ и/или $\sin\left(\frac{\alpha}{2}\right)$ (10 баллов).
- Получено соотношение между площадью треугольника и площадью искомой области (15 баллов).
- Правильно рассчитана (с округлением до целого) площадь искомой области (5 баллов).

Решение

Введем обозначения и сделаем дополнительные построения, как показано на рисунке. Из $\triangle ABO_2$ находим: $AB = BO_2 = R$, $AO_2 = \sqrt{2}R$, где R - радиус окружности. Тогда $O_1O_2 = AO_2 - AO_1 = \sqrt{2}R - R = R(\sqrt{2} - 1)$.

$$\text{Из } \triangle OO_1D \text{ найдем } \sin\left(\frac{\alpha}{2}\right) = \frac{O_1O_2}{2R} = \frac{\sqrt{2} - 1}{2}.$$

$$\text{Можно также выразить } \sin(\alpha) = 2\sin\left(\frac{\alpha}{2}\right)\cos\left(\frac{\alpha}{2}\right) = \frac{(\sqrt{2} - 1)\sqrt{2\sqrt{2} + 1}}{2}.$$

Так как $\angle CO_2D = \pi - \alpha$, то площади круговых секторов, ограниченных точками C и D , малого (BCO_2D) и большого (ACO_1D), равны соответственно:

$$S_{BCO_2D} = \angle CO_2D \frac{R^2}{2} = \left(\frac{\pi}{2} - \frac{\alpha}{2}\right) R^2,$$

$$S_{ACO_1D} = \pi R^2 - \angle CO_1D \frac{R^2}{2} = \left(\frac{\pi}{2} + \frac{\alpha}{2} \right) R^2.$$

Так как площадь ромба O_1CO_2D равна $S_{O_1CO_2D} = R^2 \sin(\alpha)$, то искомая площадь одной половинки «непересекающейся» области составит

$$S_{ACBD} = S_{ACO_1D} + S_{O_1CO_2D} - S_{BCO_2D} = R^2(\sin\alpha + \alpha).$$

Далее можно заметить, что площадь вписанного по условию задачи треугольника $S_{\Delta} = R^2$, а значит, искомая площадь будет определяться формулой

$$S = 2S_{ACBD} = 2 \cdot 50(\sin\alpha + \alpha).$$

После подстановки численных значений и округления до целых получим $S \approx 82 \text{ см}^2$.

Ответ: $S \approx 82 \text{ см}^2$.

Задача III.1.2.2. (10 баллов)

Решить уравнение $f(f(f(f(f(x)))))) = 0$, если $f(x) = x^2 + 6x + 6$.

Критерии оценивания

- Ход решения верный, решение неверное из-за арифметической ошибки (7 баллов).
- Решение верное, ответ верен (10 баллов).

Решение

Преобразуем функцию: $f(x) = x^2 + 6x + 6 = x^2 + 6x + 9 - 3 = (x + 3)^2 - 3$.

Тогда $f(f(x)) = ((x + 3)^2 - 3 + 3)^2 - 3 = (x + 3)^4 - 3$.

Аналогично $f(f(f(x))) = ((x + 3)^4 - 3 + 3)^2 - 3 = (x + 3)^8 - 3$. Соответственно, $f(f(f(f(x)))) = ((x + 3)^8 - 3 + 3)^2 - 3 = (x + 3)^{16} - 3$.

И наконец, $f(f(f(f(f(x)))))) = ((x + 3)^{16} - 3 + 3)^2 - 3 = (x + 3)^{32} - 3$.

Приравниваем к нулю, и получаем, что $(x + 3)^{32} = 3$, откуда $x = -3 \pm \sqrt[32]{3}$ или $x_1 \approx -1,9651$, $x_2 \approx -4,0349$.

Ответ: $x = -3 \pm \sqrt[32]{3}$.

Задача III.1.2.3. (20 баллов)

Команда участников олимпиады по виртуальной реальности состоит из 12 человек. Для выполнения заданий олимпиады ребят распределяют так, чтобы каждый из них смог поработать вместе с каждым человеком из своей команды. Какое минимальное число заданий организаторы могут дать, чтобы каждый член команды смог поработать с каждым из остальных членов хотя бы по одному разу, если для каждого задания нельзя использовать более 6 человек?

Критерии оценивания

- Приведен ответ без обоснования (10 баллов).
- Приведены рассуждения, используемые при решении задачи, ответ верен, но не указано, почему это решение минимальное (15 баллов).
- Приведены рассуждения, используемые при решении задачи, ответ верен и доказано, что это решение минимальное (20 баллов).

Решение

Занумеруем участников от 1 до 12. Выпишем возможные разные команды по 6 человек для выполнения заданий с тем, чтобы каждый поработал с каждым, но минимальное количество раз. Если в первую группу объединить ребят с 1 по 6, а во вторую — с 7 по 12, то, например, первому останется поработать только с шестью ребятами из второй группы. Как это сделать минимальным способом? Возьмем теперь нечетные номера вместе и четные номера вместе. И наконец, сгруппируем нечетные номера из первой группы с четными из второй и наоборот:

$$\begin{aligned}
 &1 - 2 - 3 - 4 - 5 - 6, \\
 &7 - 8 - 9 - 10 - 11 - 12, \\
 &1 - 3 - 5 - 7 - 9 - 11, \\
 &2 - 4 - 6 - 8 - 10 - 12, \\
 &1 - 3 - 5 - 8 - 10 - 12, \\
 &2 - 4 - 6 - 7 - 9 - 11.
 \end{aligned}$$

Получилось, что минимальное количество таких групп — 6, причем каждый человек работает с каждым не более 3х раз.

Ответ: 6 заданий.

Задача III.1.2.4. (25 баллов)

Бригада из пяти электромонтеров собирает электрическую цепь для энергопитания и обслуживания умного дома. Первый и третий монтеры всегда работают вместе. Объединившись со вторым, они выполняют работу за 7,5 ч. Без второго, но с пятым — закончат за 5 ч, вместе с четвертым эта парочка справится с заданием за 6 ч, в то время, как второй, четвертый и пятый управятся вместе за 4 ч. За сколько времени справятся с работой все члены бригады вместе?

Критерии оценивания

- Получена правильная система уравнений (10 баллов).
- Решение системы в целом верное, но допущены арифметические ошибки (20 баллов).
- Решение задачи и ответ верны (25 баллов).

Решение

Обозначим всю работу за 1, а производительность i -го работника — за x_i , $i = 1, 2, 3, 4, 5, 6$. Тогда из условий задачи получаем следующую систему уравнений:

$$\frac{1}{x_1 + x_2 + x_3} = 7,5,$$

$$\frac{1}{x_1 + x_3 + x_5} = 5,$$

$$\frac{1}{x_1 + x_3 + x_4} = 6,$$

$$\frac{1}{x_2 + x_4 + x_5} = 4.$$

Требуется найти $\frac{1}{x_1 + x_2 + x_3 + x_4 + x_5}$. Перемножим уравнения крест-накрест, и получим систему:

$$7,5x_1 + 7,5x_2 + 7,5x_3 = 1,$$

$$5x_1 + 5x_3 + 5x_5 = 1,$$

$$6x_1 + 6x_3 + 6x_4 = 1,$$

$$4x_2 + 4x_4 + 4x_5 = 1.$$

Умножим первое уравнение на $\frac{1}{3}$, второе уравнение — на $\frac{1}{2}$, и вычтем, затем умножим второе уравнение на $\frac{1}{5}$, третье — на $\frac{1}{6}$, и вычтем, и, наконец, последнее уравнение поделим на 4. Получим систему на 3 переменные:

$$2,5x_2 - 2,5x_5 = -1/6,$$

$$x_5 - x_4 = 1/30,$$

$$x_2 + x_4 + x_5 = 1/4.$$

Выразим из первого уравнения x_2 , из второго x_4 , и подставим в последнее:

$$x_2 = x_5 - 1/15,$$

$$x_4 = x_5 - 1/30,$$

$$x_5 - 1/15 + x_5 - 1/30 + x_5 = 1/4,$$

откуда $x_5 = 7/60$, $x_2 = 3/60$, $x_4 = 5/60$.

Подставляя полученные значения в любое из первых трех уравнений исходной системы, получим, что $x_1 + x_3 = 5/60$. Таким образом, $\frac{1}{x_1 + x_2 + x_3 + x_4 + x_5} = 3$ ч.

Ответ: за 3 часа.

Задача III.1.2.5. (15 баллов)

В компьютерной игре игроку предстоит засеять треугольное поле двумя культурами: кукурузой и соевыми бобами. Недолго думая, игрок решил вспахать поле трактором параллельно одной стороне треугольника. Получилось 9 параллельных полос одинаковой ширины. Игрок решил засеивать полосы попеременно кукурузой и соевыми бобами, засеив самую маленькую треугольную полосу кукурузой, следующую, трапециевидную — соевыми бобами, и т. д. В результате оказалось, что кукурузой было засеяно больше квадратных единиц площади. На сколько процентов

меньше площади было отведено под соевые бобы в сравнении с площадью, отданной под кукурузу (ее принимаем за 100 процентов)? Замечание: треугольник был произвольный (не равнобедренный, не равносторонний, не прямоугольный).

Критерии оценивания

- Ответ верен, но не до конца обоснован (10 баллов).
- Решение верное, ответ не соответствует вопросу задачи (12 баллов).
- Решение верное, ответ верен (15 баллов).

Решение

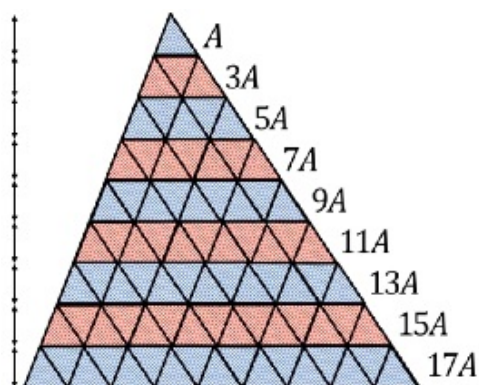


Рис. III.1.7. Рисунок к задаче 5.

Обозначим площадь самой маленькой верхней треугольной полосы за A . Тогда в силу того, что полосы параллельны и одной ширины, в следующей полосе содержатся 3 таких треугольника (см. рисунок). В последующей уже 5, и т.д.

Таким образом, все синие (нечетные, начиная сверху) полосы суммарно содержат $A + 5A + 9A + 13A + 17A = 45$ кв. ед. А все красные (четные, начиная сверху) полосы суммарно содержат $3A + 7A + 11A + 15A = 36A$ кв. ед, что составляет 80 процентов от площади, засеянной кукурузой. То есть бобами было засеяно на 20 процентов меньше площади, чем кукурузой.

Ответ: бобами было засеяно на 20 процентов меньше площади, чем кукурузой.

Математика. 10–11 классы

Задача III.1.3.1. (10 баллов)

В виртуальной реальности все — не то, что кажется. Вот герой оказался на морском дне, и кто может его спасти? Перед глазами — лишь уравнение:

$$1 - \ln x = -\ln(c - px).$$

С помощью «честных» математических действий преобразуйте уравнение в слово — и оно спасет героя!

Подсказка: слово читается на русском языке.

Критерии оценивания

- Ход решения верный, получилось слово, но не то, например, «череп» (5 баллов).
- Решение верное, ответ верен (10 баллов).

Решение

Дано: $1 - \ln x = -\ln(\psi - pn)$. По свойствам логарифмов минус перед логарифмом занесем в степень выражения под знаком логарифма:

$$1 - \ln x = \ln(\psi - pn)^{-1}.$$

Представим единицу как натуральный логарифм числа e :

$$\ln e - \ln x = \ln(\psi - pn)^{-1}.$$

По свойствам логарифмов:

$$\ln \frac{e}{x} = \ln(\psi - pn)^{-1}.$$

Избавляемся от логарифмов:

$$e/x = 1/(\psi - pn).$$

Перемножаем крест-накрест:

$$e\psi - epn = x.$$

Сомножители можно поменять местами:

$$\psi e - pen = x.$$

Домножаем правую и левую часть равенства на a : $(\psi e - pen)a = xa$. Получилось слово: «черепаха».

Ответ: «черепаха».

Задача III.1.3.2. (10 баллов)

В электрическую цепь включены элементы $a_1, a_2, a_3, a_4, a_5, a_6$ по схеме, изображенной на рисунке. Все элементы сети работают независимо друг от друга. Вероятности $p_i, i = 1, 2, \dots, 6$ выхода из строя каждого элемента в каком-то заданном промежутке времени приведены в таблице. Найти вероятность разрыва цепи в данный промежуток времени.

a_i	a_1	a_2	a_3	a_4	a_5	a_6
p_i	0,3	0,1	0,4	0,5	0,2	0,6

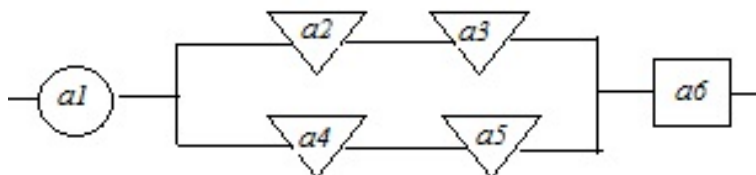


Рис. III.1.8. Рисунок к задаче III.1.3.2

Критерии оценивания

- Ход решения верный, но неверно посчитали вероятность невыхода из строя центрального блока (5 баллов).
- Решение верное, но нашли вероятность того, что цепь работает (7 баллов).
- Решение верное, ответ верен (10 баллов).

Решение

Вероятность того, что центральный блок НЕ выйдет из строя:

$$0,9 \cdot 0,6 + 0,5 \cdot 0,8 - 0,9 \cdot 0,6 \cdot 0,5 \cdot 0,8 = 0,724.$$

Вероятность того, что вся цепь работает:

$$0,7 \cdot 0,724 \cdot 0,4 = 0,20272.$$

Вероятность разрыва цепи:

$$1 - 0,20272 = 0,79728.$$

Ответ: 0,79728.

Задача III.1.3.3. (20 баллов)

В сказке Пушкина «белка песенки поет, да орешки все грызет. А орешки не простые, все скорлупки золотые, ядра — чистый изумруд». Если белка, грызя орешки, случайно расколлет ядро массой p на две части так, что суммарная стоимость двух полученных камней окажется меньше стоимости исходного изумрудного ядра в k раз, то какова будет масса этих частей, если цена изумруда находится в пропорциональной зависимости от квадрата его массы? В каком случае общая стоимость двух частей будет наименьшей?

Критерии оценивания

- Найден ответ на первый вопрос, но не указано, что k должно быть меньше 2 (7 баллов).

- Найден ответ на второй вопрос, но не доказано, что $p_1 = p/2$ действительно точка минимума (7 баллов).
- Найден ответ на один из вопросов задачи (10 баллов).
- Найдены верные ответы на оба вопроса задачи (20 баллов).

Решение

Пусть изумруд массой p карат разбит на две части массой p_1 и p_2 : $p = p_1 + p_2$. Пусть стоимость первоначального изумруда ap^2 . Тогда стоимость полученных кусков будет равна ap_1^2 и ap_2^2 . Так как суммарная стоимость двух полученных камней оказалась меньше стоимости исходного изумрудного ядра в k раз, $ap_1^2 + ap_2^2 = ap^2/k$. Получили систему уравнений:

$$\begin{aligned} a(p_1^2 + p_2^2) &= ap^2/k, \\ p_1 + p_2 &= p. \end{aligned}$$

Возведем второе уравнение в квадрат, и вычтем из него первое: $2p_1p_2 = p^2 - p^2/k$. Получили систему:

$$\begin{aligned} p_1 \cdot p_2 &= p^2 \frac{k-1}{2k}, \\ p_1 + p_2 &= p. \end{aligned}$$

Эта система напоминает теорему Виета. Если p_1 и p_2 — корни квадратного уравнения $x^2 + bx + c = 0$, то $b = -p$, $c = p^2 \frac{k-1}{2k}$, и уравнение имеет вид:

$$x^2 - px + p^2 \frac{k-1}{2k} = 0.$$

Умножим уравнение для удобства на $2k$:

$$2kx^2 - 2kpx + p^2(k-1) = 0.$$

Половинный дискриминант полученного уравнения $D = k^2p^2 - 2k(k-1)p^2 = p^2(2k - k^2)$ больше или равен нулю при $k \leq 2$. В этом случае корни уравнения

$$p_{1,2} = \frac{kp \pm p\sqrt{2k - k^2}}{2k}.$$

Мы нашли массу двух кусков ядра.

Теперь ответим на вопрос, при каких условиях общая стоимость двух частей будет наименьшей. Общая стоимость двух частей равна:

$$f = a(p_1^2 + p_2^2) = a(p_1^2 + (p - p_1)^2) = a(2p_1^2 - 2pp_1 + p^2),$$

где p_1 — переменная величина, а p — параметр. Минимум полученной функции достигается, когда производная равна нулю: $f' = 4p_1 - 2p = 0$, то есть при $p_1 = p/2$. Проверим, что в этой точке действительно минимум по достаточному условию экстремума: $f' < 0$ при $p_1 < p/2$ и $f' > 0$ при $p_1 > p/2$, следовательно, функция убывает при $p_1 < p/2$ и возрастает при $p_1 > p/2$, и при $p_1 = p/2$ действительно достигает своего минимума. Таким образом, стоимость двух частей будет наименьшей, если белка расколется камень ровно пополам.

Ответ:

1. $p_{1,2} = \frac{kp \pm p\sqrt{2k - k^2}}{2k}$ при $k \leq 2$.
2. Стоимость двух частей будет наименьшей, если белка расколется камень ровно пополам.

Задача III.1.3.4. (30 баллов)

Доказать, что

$$\left[\frac{n+2^0}{2^1} \right] + \left[\frac{n+2^1}{2^2} \right] + \dots + \left[\frac{n+2^{n-1}}{2^n} \right] = n.$$

Здесь квадратные скобки означают операцию взятия целой части от числа, например, $[1, 3] = 1$, $[2, 7] = 2$.

Подсказка: стоит перейти в двоичную систему исчисления.

Критерии оценивания

- Формула проверена при $n = 0, 1, 2$, и возможно при больших значениях, но не доказана (5 баллов).
- Формула проверена при $n = 0, 1, 2$, найдена рекуррентная зависимость $f(n)$ от предыдущих значений функции, но эта зависимость не доказана (15 баллов).
- Формула частично доказана (20 баллов).
- Формула полностью доказана (30 баллов).

Решение

Перейдем в двоичную систему исчисления. Заметим, что $\left[\frac{x}{2^n} \right]$ означает, что мы убираем последние n битов из числа x , например:

$$\frac{(1101101)_2}{2^3} = \frac{(1101000)_2}{2^3} + \frac{(101)_2}{2^3} = (1101)_2 + \frac{(101)_2}{2^3},$$

так как $(1101000)_2$ кратно $2^3 = (1000)_2$, а целая часть данного числа равна $(1101)_2$, так как $(101)_2 < 2^3$.

Пусть n — двоичное число из $m + 1$ разряда: $n = (b_m b_{m-1} \dots b_2 b_1 b_0)_2$. Рассмотрим

$$\left[\frac{n + 2^k}{2^{k+1}} \right] = \frac{(b_m \dots b_k \dots b_0)_2 + \overbrace{(10 \dots 0)_2}^k}{2^{k+1}}.$$

Если $b_k = 1$, то в числе числителя на k -м месте имеем $1+1=0$ и 1 «в уме» (идет в следующий разряд). Если $b_k = 0$, то в числе числителя на k -м месте имеем $0+1=1$ и 0 идет в следующий разряд.

То есть в любом случае то, что добавляется к $(k+1)$ -му разряду, совпадает с b_k . Далее число числителя делится на 2^{k+1} , следовательно, после добавки того, что «в уме», нужно убрать из числа $k+1$ битов. Получаем, что

$$\left[\frac{n + 2^k}{2^{k+1}} \right] = \frac{(b_m \dots b_k \dots b_0)_2 + (10 \dots 0)_2}{2^{k+1}} = (b_m \dots b_{k+1})_2 + (b_k)_2.$$

Теперь рассмотрим сумму таких слагаемых при $k = 0, 1, \dots, n-1$:

$$f(n) = f((b_m \dots b_0)_2) = (b_m \dots b_1)_2 + (b_0)_2 + (b_m \dots b_2)_2 + (b_1)_2 + \dots + (b_m)_2 + (b_{m-1})_2 + (0)_2 + (b_m)_2.$$

Посмотрим на примере, как расписывается такая сумма:

$$f(1101)_2 = (110)_2 + (1)_2 + (11)_2 + (0)_2 + (1)_2 + (1)_2 + (0)_2 + (1)_2.$$

Заметим, что последние три строчки представляют собой $f((110)_2)$, следовательно, в общем случае функцию можно выразить рекуррентно:

$$f((b_m \dots b_0)_2) = f((b_m \dots b_1)_2) + (b_m \dots b_1)_2 + (b_0)_2.$$

По этой формуле найдем:

$$f((0)_2) = (0)_2 + (0)_2 = (0)_2, f((1)_2) = (1)_2 + (0)_2 = (1)_2,$$

$$f((10)_2) = (1+0)_2 + (0+1)_2 = (1)_2 + (1)_2 = (10)_2,$$

$$f((11)_2) = (1+1)_2 + (0+1)_2 = (10)_2 + (1)_2 = (11)_2.$$

Можно сделать вывод, что $f(n) = n$ для всех чисел n в двоичной системе исчисления, то есть $f((b_m \dots b_0)_2) = (b_m \dots b_0)_2$.

Проверим по индукции. Если $f((b_m \dots b_1)_2) = (b_m \dots b_1)_2$, то, подставляя найденное значение $f((b_m \dots b_1)_2)$ в рекуррентную формулу, получим:

$$f((b_m \dots b_0)_2) = (b_m \dots b_1)_2 + (b_m \dots b_1)_2 + (b_0)_2 = 2 \times (b_m \dots b_1)_2 + (b_0)_2.$$

Умножение на 2 в двоичной системе исчисления эквивалентно дописанию нолика справа, следовательно, $f((b_m \dots b_0)_2) = (b_m \dots b_1 0)_2 + (b_0)_2 = (b_m \dots b_0)_2$, то есть действительно $f(n) = n$, для всех чисел n , что и требовалось доказать.

Ответ:

$$\left[\frac{n + 2^0}{2^1} \right] + \left[\frac{n + 2^1}{2^2} \right] + \dots + \left[\frac{n + 2^{n-1}}{2^n} \right] = n.$$

Задача III.1.3.5. (30 баллов)

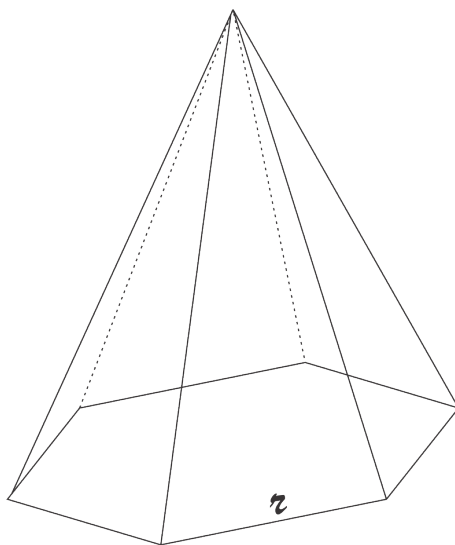


Рис. III.1.9. Чертеж к задаче III.1.3.5

В качестве одного из призов победителям в олимпиаде по дополненной реальности решили вручать кубок. Заготовка для кубка представляет собой правильную шестиугольную пирамиду. Площадь полной поверхности пирамиды фиксирована и составляет $S = 403 \text{ см}^2$. Найдите длину стороны в основании пирамиды, если известно, что пирамида должна иметь наибольший объем. Найдите этот объем. Ответ округлите до целого числа кубических сантиметров.

Критерии оценивания

- Получено функциональное выражение для объема пирамиды относительно стороны ее основания (15 баллов).
- Правильно найдена производная и составлено уравнение для нахождения наибольшего объема (5 баллов).
- Правильно решено уравнение и найдено значение стороны основания (5 баллов).
- Правильно рассчитан наибольший объем с округлением до целого числа кубических сантиметров (5 баллов).

Решение

Введем обозначения, как показано на рисунке. Из рисунка очевидно, что основание пирамиды составлено из шести равносторонних треугольников со стороной r и высотой d . Причем $d = \sqrt{r^2 - \left(\frac{r}{2}\right)^2} = \frac{\sqrt{3}r}{2}$. Площадь боковой поверхности пирамиды определяется формулой $S_b = 6 \frac{lr}{2} = 3lr$, где l — высота боковой стороны, а площадь основания равна $S_o = 6 \frac{rd}{2} = \frac{3\sqrt{3}}{2}r^2$.

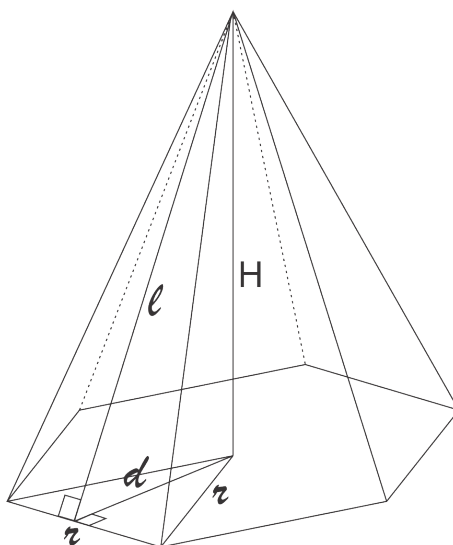


Рис. III.1.10. Обозначения и дополнительные построения к задаче III.1.3.5

Площадь полной поверхности, соответственно, равна:

$$S = S_o + S_b = 3lr + \frac{3\sqrt{3}}{2}r^2.$$

Рассматривая данную формулу как уравнение относительно высоты боковой поверхности, получим для нее выражение:

$$l = \frac{S}{3r} - \frac{\sqrt{3}}{2}r.$$

Высота пирамиды по теореме Пифагора равна $H = \sqrt{l^2 - d^2} = \sqrt{l^2 - \frac{3}{4}r^2}$, поэтому объем пирамиды определяется функцией от r :

$$V(r) = \frac{1}{3}HS_0 = \frac{\sqrt{3}}{2}r^2\sqrt{l^2 - \frac{3}{4}r^2} = \frac{\sqrt{3}}{6}rS\sqrt{1 - \frac{3\sqrt{3}r^2}{S}}.$$

Приравнивая производную от объема по r нулю, получим уравнение на величину стороны основания, соответствующую наибольшему объему:

$$\frac{dV(r)}{dr} = \frac{\sqrt{3}}{6}S\sqrt{1 - \frac{3\sqrt{3}}{S}r^2} - \frac{3r^2}{2\sqrt{1 - \frac{3\sqrt{3}}{S}r^2}} = 0.$$

Откуда получаем $r_{max} = \sqrt{\frac{S}{6\sqrt{3}}} \approx 6,227$ см и, соответственно, для наибольшего объема $V(r_{max}) = \frac{S\sqrt{S}}{12\sqrt[4]{3}} \approx 512$ см³.

Ответ: $r_{max} \approx 6,227$ см; $V(r_{max}) \approx 512$ см³.

Командный практический тур

Основная задача финала профиля – моделирование энергосистемы и разработка алгоритмов управления энергообеспечения. Цель задачи - разработать стратегию оптимального построения умной энергосети и управления ею в условиях локальной конкуренции.

Соревнования проводятся на стенде «Интеллектуальные энергетические системы», разработанном компанией «Полюс-НТ», — это модель поселения с потребителями энергии (жилые дома, больницы, заводы), электростанциями на альтернативных источниках энергии. Стенд воссоздает погодные условия (ветер и освещенность) и рынок электроэнергии (многоуровневая биржа микроконтрактов).

Участники должны разработать экономические стратегии, основываясь на работе с прогнозами погоды и потребления, и писать скрипты по управлению энергообъектами.

Требования к команде

Команде предстоит освоить работу с компьютерным зрением и нейронными сетями в различных сферах робототехники: беспилотный автомобиль на базе компьютерного зрения, квадрокоптер в режиме автономного полета, автоматизированный мостовой кран сортировочного хаба.

Количество участников в команде: 3-5. Состав команды приведен ниже.

- **Data-аналитик** способен работать с данными, анализировать, выбирать те что наиболее информативны в нужный момент времени. На этом человеке может быть совмещена роль тестера в команде.
- **Системный аналитик** анализирует стратегию игры, занимается топологией сети, анализирует поведение других команд на площадке и на других распределенных площадках. Здесь понадобятся все навыки и знания из теории игр, умение комбинировать и искать оптимальные стратегии. Совместно аналитики должны быть способны определить, какие данные важны для создания алгоритмов управления и что будет являться управляющими параметрами.
- **Программист.** В команде желательно два программиста, которые способны реализовывать алгоритмы управления и вывода необходимых данных, строить обратные связи, работать с реализацией стратегии. Вдвоем они должны быть способны делать ревью кода друг друга. Здесь должны пригодиться навыки совместного программирования, на отработку которых направлен хакатон профиля.
- **Капитан,** человек который в состоянии принимать решения, находясь на слой выше в этом потоке информации. Он должен обладать знаниями по всем основным темам, уметь разбираться в коде, анализировать задачи, строить стратегии. Отвечает за постановку задачи, видит картину целиком, отвечает за распределение задач в команде, быстрое перераспределение задач в случае необходимости, отделяет главное от второстепенного, перенаправляет усилия

команды. От работы капитана во многом зависит исход игры.

Совмещение ролей возможно, не рекомендуется совмещать роль капитана, с ролью программиста, так как грамотные действия капитана иногда являются критическими и решающими, а находясь одновременно в позиции программиста, очень сложно отвлечься от текущей реализации и принять необходимое решение.

Оборудование и программное обеспечение

Требования к рабочему месту

- Процессор Intel Pentium i3–5005 поколения или мощнее.
- 4 ГБ ОЗУ и выше.
- Скорость интернета не ниже 20 Мбит/с.
- Монитор с разрешением Full HD и выше.
- Микрофон и вебкамера желательны.
- ОС, подходящая для запуска актуальной версии Chrome: Windows 7 и новее, Mac OS X El Capitan (10.11) или более поздней версии, дистрибутив на основе Linux 5.3 и новее.
- Установлено ПО Wireguard версии 0.5.3 и новее: <https://www.wireguard.com/install/> (Взаимодействие со стендом осуществляется через веб-интерфейс, идентичный используемому в очном формате. Для обеспечения безопасности доступ к веб-интерфейсу производится через туннель Wireguard).

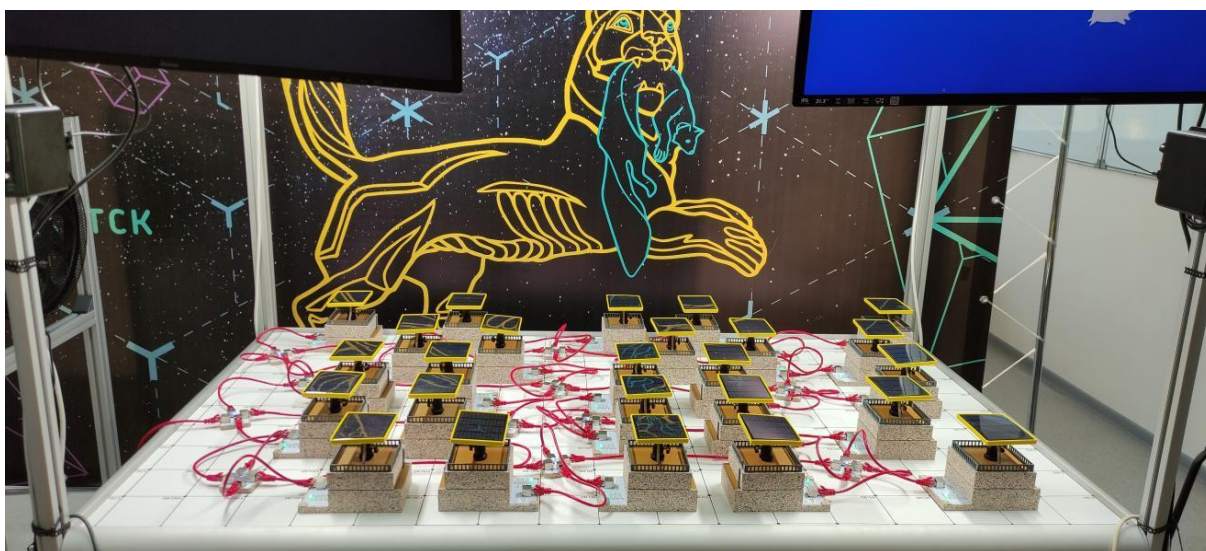
Описание стенда-тренажера «ИЭС» 2021–2022 года

Стенд-тренажер «ИЭС» 2021-2022 года состоял из следующих основных частей:

1. Модельная поверхность, на которой располагается поле ветряных электростанций (ВЭС).



2. Модельная поверхность, на которой располагается поле солнечных электростанций (СЭС).



3. Терминалы управления энергосистемой. Это облачные компьютеры, объединенные со стендом в единую информационную сеть. Каждая команда работает со своим терминалом через предварительно настроенный туннель.
4. Светильники, моделирующие солнечное освещение. Они способны создавать освещенность в центре стола до 5 клк.
5. Мощный вентилятор, моделирующий ветровые условия. Он способен создавать ветер со скоростью до 5 м/с на расстоянии не менее 1 метра от плоскости вращения, и направлен на поле ветряков.
6. Управляющая электроника стенда. Эта часть скрыта от участников и с точки зрения участников олимпиады не имеет функциональной нагрузки.

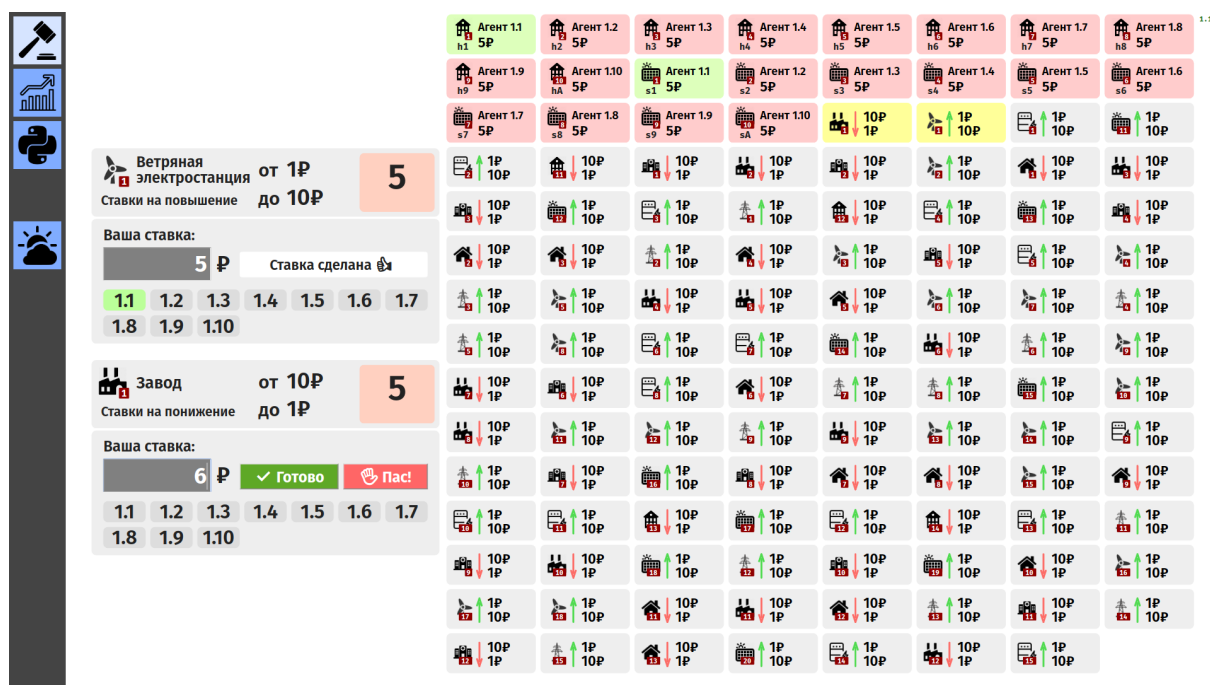


Рис. III.2.1. Интерфейс аукциона во время игры

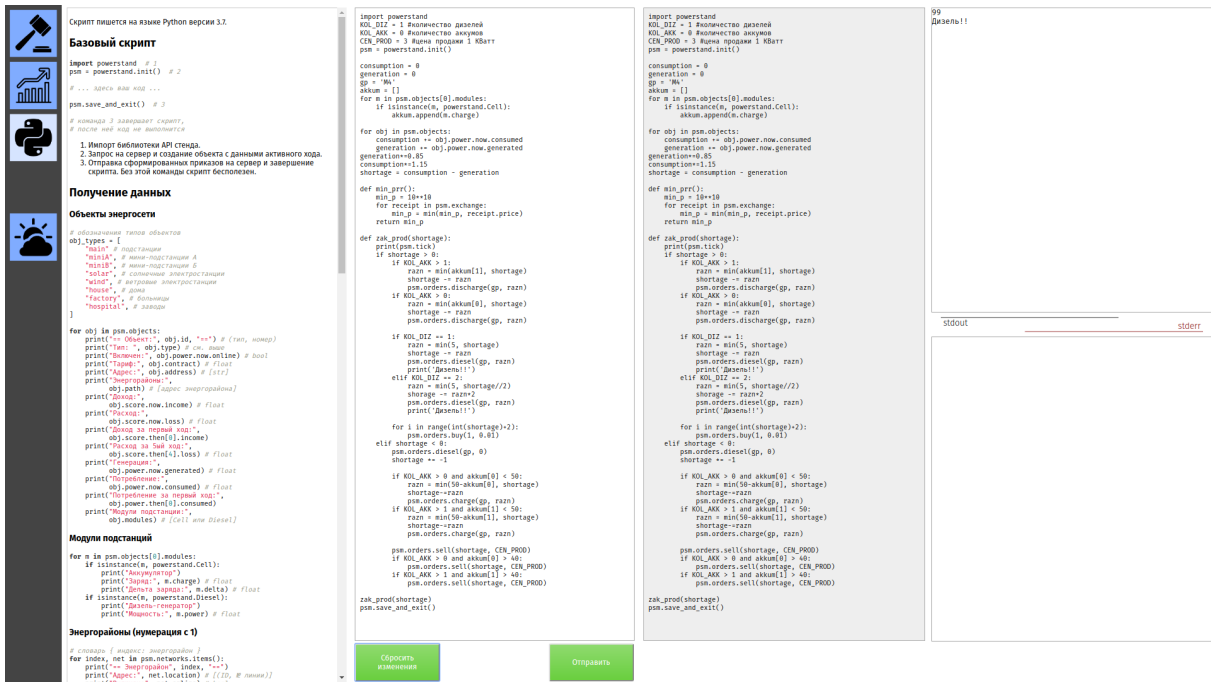
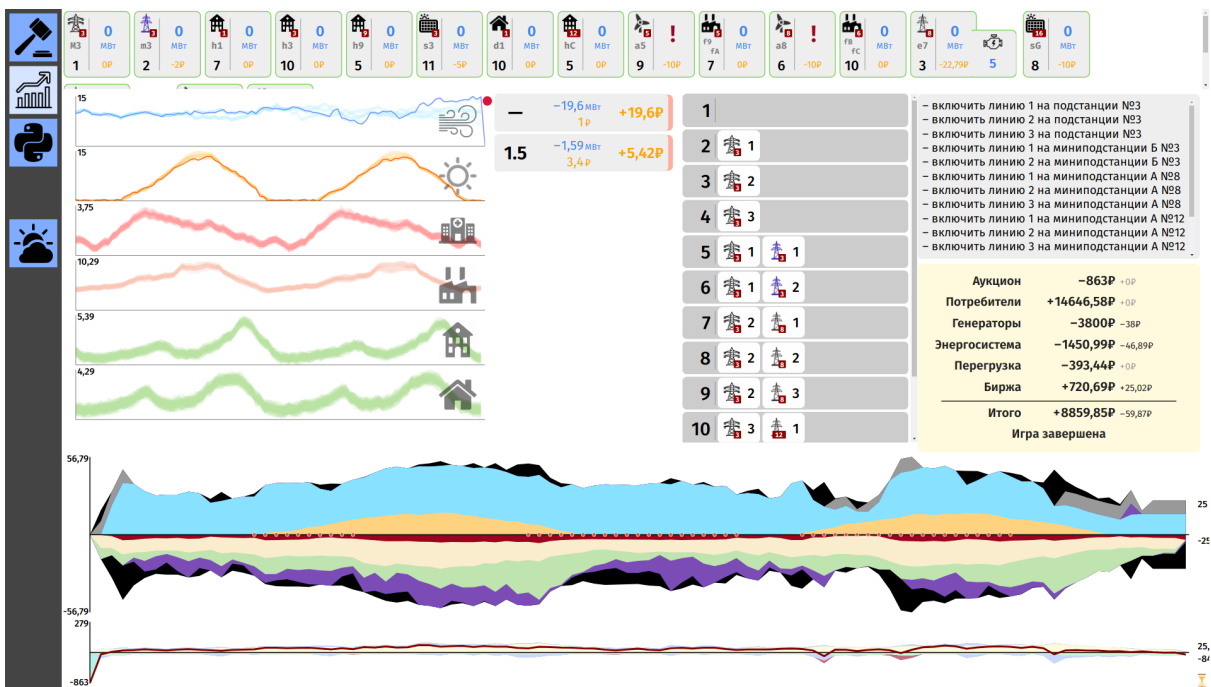


Рис. III.2.2. Скриншот интерфейса скрипта. Слева находится справка, в центре-слева редактор скрипта, в центре-справа загруженный скрипт, справа — стандартный вывод и вывод ошибок скрипта



Интерфейс моделирования. Внизу находятся графики энергетического и полного баланса, вверху — состав энергосистемы, слева — графики прогнозов погоды и потребления, справа — принятые приказы и итоговый счет. Остальные поля являются вспомогательными. В силу большого числа отображаемых данных участники осваиваются с этим интерфейсом на протяжении нескольких игр и в интерактивном режиме.

Описание задачи

В ходе заключительного этапа происходит следующее моделирование: энергосистемы объединены в единую сеть и подсоединены к внешней энергосистеме по схеме «микрогрид»: каждая из энергосистем сначала балансируется собственными ресурсами, а затем балансируется через внешнюю энергосистему, в том числе и с помощью оппонентов. Будущая энергосистема «разделена» между конкурирующими компаниями. Каждая команда становится одной из конкурирующих энергокомпаний и строит свою собственную энергосистему и управляет ею. Разделение происходит через цепочку аукционов, в которых участники изначально находятся в принципиально одинаковых условиях.

После этого команды проектируют собственные энергосистемы (из одних и тех же объектов можно составить очень разные по эффективности энергосистемы) и готовят скрипты для управления ими. Далее происходит моделирование нескольких «дней работы» энергосистемы, в ходе которого участники управляют своими энергосистемами посредством скриптов.

Команды в течение нескольких сессий анализируют прогнозы погоды и поведения потребителей, распределяют объекты потребления и генерации на аукционе, собирают из них энергосеть, настраивают автоматизированную систему управления (путем написания скриптов на языке Python) и испытывают ее в ходе моделирования энергосистемы в пробных играх. Конечной метрикой эффективности разработанной стратегии является прибыль, полученная в результате штатного функционирования энергосистемы. В последние дни заключительного этапа происходит экспериментальное измерение эффективности построенных энергосистем в соревновательных играх. Очки, набранные командами во время моделирования, пересчитываются в баллы, которые участники получают за командную часть олимпиады.

В этом году заключительный этап проходил в распределенном формате и участники подключались к терминалам своих энергосистем удаленно, и удаленно общались друг с другом и разработчиками задачи. В этом году физическая задача определения параметров генераторов была модифицирована и представляла собой в распределенном формате задачу больших данных: в отличие от очного формата после игры все участники получали данные со всех генераторов, а позиция генераторов на стенде не изменялась, что увеличивает равенство условий, но усложняет их обработку и построение стратегий.

Задача командного практического тура заключительного этапа представляет собой комплексную интегральную задачу, полное оптимальное решение которой чрезвычайно сложно. Однако, полное приближенное решение способна найти любая команда. Качество и сложность используемых приближений отражает глубину понимания, знаний и уровень компетенций участников. Система оценки полностью автоматизирована и спроектирована таким образом, чтобы однозначно и численно оценить качество найденных и реализованных участниками приближенных решений.

Проведение командного тура заключительного этапа происходило в виде турнира. Цель команд участников — набрать наибольшее число баллов в турнире.

Регламент турнира

1. Подготовка

Проводится уже сформированными командами. Это продолжительный по времени этап (4 дня), в течение которого участники знакомились с правилами, настраивали подключение к системе работы стенда, изучали предоставленную систему и готовили заготовки (управляющие скрипты, стратегии и вспомогательные программы). Во время подготовки проводился ряд пробных игр.

2. Финал турнира

Проводится три соревновательных игры. В каждой игре вся совокупность принятых участниками решений интегрально оценивается стендом в автоматическом режиме. Каждая игра проводится с разными прогнозами. За каждую игру очки, заработанные участниками, переводятся в рейтинговые баллы.

Игры во время командного тура заключительного этапа разделяются на два вида: пробные и соревновательные игры. Все пробные игры предназначены для отработки решений и проведения экспериментов. Никаких баллов за пробные игры не начисляется. Все командные баллы участники получают только за результаты соревновательных игр.

Этапы одной игры

1. **Анализ прогнозов погоды.** Минимум за 15 минут до игры участникам выдавались прогнозы погоды и потребления для каждого потребителя в предстоящей игре. За это время участники должны были спроектировать энергосистему, наиболее отвечающую предстоящим условиям.
2. **Основной аукцион.** На этом этапе определялось, какой объект к чьей энергосистеме будет подключен. Аукцион закрытого типа, с продолжением в случае одинаковых ставок. Одновременно разыгрывались по два лота. Этот этап практически невозможно успешно пройти без глубокого предварительного анализа и подготовленных программ для поддержки принятия решений.
3. **Дополнительный аукцион.** Проводился через 60 секунд после основного. Каждая команда имела право повторно «выставить на торги» любой из приобретенных ранее объектов. Этот этап давал командам шанс на исправление одной ошибки, если она не слишком велика.
4. **Проектирование сети.** Участникам предоставляется время на проектирование топологии сети. За это время участники находят оптимальную конфигурацию своей энергосистемы, записывают ее в файл установленного типа и передают разработчикам в закрытом режиме. В это же время команда адаптирует к получившейся энергосистеме составленные заранее управляющие скрипты.
5. **Загрузка скрипта в систему стенда.** Скрипт должен был быть написан и проверен заранее.
6. **Моделирование.** Данный этап начинается после подтверждения готовности всех команд в чате с разработчиками. Участники наблюдают за работой своих скриптов и могут их заменить, например, в случае обнаружения ошибки. Число загружаемых скриптов не ограничено. Все управление на этом этапе осуществляется только скриптами.
7. **Публикация результатов и данных генерации.**
8. **Анализ и обсуждение результатов** внутри команды, поиск гипотез и составление планов их проверки.

Регламент аукциона

Аукцион проводился по закрытой схеме.

Выигрывал предложивший наибольшую цену в аукционе на электростанции, и наименьшую — в аукционе на потребителей. Стартовая цена для электростанций составляла 1, для потребителей — 10.

На ставку отводилось 12 секунд. В случае близости наилучших ставок на 0,5 или менее, проводился дополнительный тур аукциона, в котором участвовали только те, чьи ставки попали в диапазон 0,5 от ставки лидера.

В случае, если один из участников достигает предельной цены, аукцион заканчивается. Если два и более участника достигли предельной цены, для них начинался аукцион по системе «all-pay» («платят все»). В этом аукционе свои ставки выплачивают все: и победители, и проигравшие. Итоговые ставки участников этого аукциона вычитались из их результата. Сумма ставок на аукционе «all-pay» не может превышать 1000, что контролируется системой управления ходом аукциона.

Порядок выставления лотов фиксирован и известен участникам заранее. В течение 60 секунд после окончания аукциона каждая команда имела право выставить на торги два из своих объектов. Команда ничего с этого не приобретает, но получает возможность избавиться от лишнего объекта, нарушающего баланс энергосистемы. Команда не имела права делать ставки на объект, который выставила на торги. У каждой команды в наличии по умолчанию имелось один дом и одна солнечная электростанция. На аукционе разыгрывалось параллельно по два лота.

Задача III.2.3.1.

Структура игры

Вы проектируете энергосистему в конкуренции с другими командами. Кто каких потребителей себе подключит и какие электростанции установит, решается через аукцион. За электростанции торг идет на повышение, а предмет торга — ежегодный платеж, который вы платите независимо от того, сколько вырабатывает электростанция. За потребителей предмет торга — тариф за 1 МВт поставленной мощности, и торг идет на понижение — потребитель выберет ту энергосистему, которая предложит цену ниже.

После аукциона вы проектируете сеть и решаете задачу оптимизации потерь в энергосистеме и увеличения ее надежности тем, как вы подключаете объекты. Ваши инструменты здесь — подстанции и их модули.

Затем проводится моделирование вашей энергосистемы, в результате которого определяется то, сколько денег она принесет — это и есть ваш счет. В этом этапе также участвует скрипт, который реагирует на происходящее в энергосистеме и предпринимает корректировки. Обратите внимание, что один только скрипт игру точно не выигрывает, но без хорошего скрипта за победу бороться трудно.

Ваша энергосистема и энергосистемы конкурентов объединены через внешнюю энергосистему — гарантирующего поставщика. Ему и через него можно продавать (или покупать) энергию, в случае дисбаланса в вашей энергосистеме.

Рекомендации по решению

Рекомендации не являются инструкцией. Задачу возможно решать и по-другому, они приведены для того, чтобы вам было, с чего начать. Скорее всего, все команды будут решать задачу иначе, но отличия от наших рекомендаций у всех будут разные.

Важное свойство стоящей перед вами задачи — что она не решается без большой подготовки. Даже если вы хорошо знаете, что делать, без заготовок вы сделать этого просто не успеете.

Более того, вы не сможете за имеющееся у вас время решить задачу полностью (вам достаточно решить ее лучше ваших оппонентов), и крайне важным будет распределение ваших усилий. Допустим, что задача состоит из трех подзадач; команда, решившая их на 50%, 50% и 50% выиграет у команды, решившей на 100%, 100% и 0%, со значительным отрывом. Далее дадим рекомендации по этапам игры, но еще раз повторим, что это рекомендации, а не предписания.

1. Анализ прогнозов и определение стратегий. Вам нужно определить, на каком фронте состязания вы будете превращать преимущество над оппонентами в лидирование в результатах. Как именно вы будете это делать — ваше дело, но без программирования и электронных таблиц вы вряд ли обойдетесь.
2. Аукцион. Он требует, исходя из выбранной стратегии, знания вашей ценности для каждого из торгуемых объектов. Также он требует дисциплины в команде. Если вы будете отвлекаться на споры и решение технических вопросов, то в ставках вы будете, скорее всего, ошибаться.
3. Проектирование сети. Проект скорее всего будет составлен еще при составлении стратегии, но результаты аукционов могут заставить внести в него правки. В результате у вас появляется файл с описанием энергосистемы, который вы отдаете организаторам.
4. Загрузка скрипта. Ваш скрипт должен быть написан и проверен заранее (но понятно, что тестовые игры нужны именно для тестирования), внести в него исправления на ходу у вас вряд ли получится безошибочно. Мы настоятельно рекомендуем перед зачетными играми не вносить в скрипт никаких изменений, насколько бы тривиальными и простыми они ни казались.
5. Анализ и обсуждение результатов внутри команды, поиск гипотез и составление планов их проверки. Это самый важный этап задачи, я победа будет коваться именно здесь.

Прогнозы

Прогнозы выдаются для солнца, ветра, и каждой категории потребителей. Прогноз выдается на каждый такт (48 тактов — одни сутки). Для каждого такта он состоит из единственного числа — центра коридора, в который попадет реальное значение прогнозируемой величины. Размеры коридоров:

Прогноз	Коридор
Солнце	0,5 (константа <code>corridorSun</code>)
Ветер	0,5 (константа <code>corridorWind</code>)
Больницы	0,25 (константа <code>corridorClass1</code>)
Заводы	0,5 (константа <code>corridorClass2</code>)
Дома А	0,5 (константа <code>corridorClass3A</code>)

Прогноз	Коридор
Дома Б	0,5 (константа <code>corridorClass3B</code>)

Значение в 5,2 с коридором в 0,5 означает, что реальное значение будет лежать в интервале от 4,7 до 5,7.

Для каждого объекта имеется 8 (константа `forecastsCount`) разных прогнозов. Верен только один. Прогнозы отличаются, но не являются абсолютно разными. Понять, какой из них верен можно до окончания моделирования, но не сразу.

У каждого типа объектов прогнозы свои, и все объекты одного типа полностью идентичны, что в прогнозах, что в реальных значениях. Прогнозы для ветра ведут себя немного иначе, чем остальные. У прогнозов ветра есть бифуркации каждые 25 тактов. Каждые 25 тактов прогнозы разделяются на два. Это означает, что первые 25 тактов игры все восемь прогнозов совпадают. Затем они делятся на две равные группы, затем на 4 и, наконец, после 75-го такта все 8 прогнозов становятся различными.

Аукцион

Аукцион проходит по закрытой системе, как тендеры.

На аукционе есть механика догоняющих ставок: если есть хоть одна ставка, которая отличается от лидирующей хотя бы на 0,5, то объявляется повторный тур аукциона. Во втором туре участвуют только те, чья ставка была догоняющей. В повторных турах догоняющие ставки должны быть увеличены хотя бы на 0,1.

Если одна из команд установила предельную цену (минимальную для потребителей или максимальную для электростанций), то механика догоняющих ставок не срабатывает, и эта команда сразу выигрывает лот. Конечно, если две или более команд установили предельную цену, то случается повторный тур, но в этом случае уже по другим правилам, которые называются All-Pay (платят все). В них ставки делаются не в виде тарифов, а в виде фиксированных сумм, которые участники готовы заплатить, чтобы лот по предельной цене достался именно им, но выплачивают свои ставки даже если они не выиграли лот.

Если на лот никто не выставил ставку, он исключается из игры. В конце аукциона возможно сбросить до двух лотов. Они выставляются на второй круг аукциона, где участвовать в борьбе за них будут те же команды, что и в первом круге, минус команды, их сбросившие.

На аукционе разыгрывается параллельно по два лота. Вам нужно очень аккуратно налаживать протоколы коммуникации и выставления ставок на аукционе, иначе темп работы вашей команды может оказаться ниже темпа аукциона.

Потребители

Потребители делятся на четыре типа в трех категориях: 1-я (больницы), 2-я (заводы) и 3-я (дома А и дома Б). Они отличаются, во-первых, потреблением и паттернами потребления, а во-вторых — размерами штрафов за отключения. За отключенный дом придется заплатить его удвоенный (константа `class3FineRate`) тариф за каждый недопоставленный МВт мощности, заводу — учетверенный (константа `class2FineRate`), а больнице — в 8 (константа `class1FineRate`) раз больше тарифа.

У больниц есть особенность — их обязательно подключать обоими входами так, чтобы путь от них к главной подстанции оказывался на разных ее ветках. У заводов входов тоже два, но столь жесткого правила нет, и один из их входов можно вообще не подключать.

Если больница или завод подключены обоими входами, то их потребление распределяется равномерно по обоим входам. Если произошло отключение по одной из линий подключения, то объект запитывается от второй и его нагрузка полностью ложится на нее.

Электростанции

Электростанции есть трех типов — солнечные, ветровые и тепловые. Солнечные и ветровые — реальные физические измерительные системы, и их расположение на стенде очень важно.

СЭС

Солнечные панели кроме перемещения по стенду могут еще и наклоняться. Инерция у солнечных батарей она небольшая, и их показания на текущем ходу от предыдущего почти не зависят.

Максимальная мощность солнечных электростанций — 15 МВт (константа `maxSolarPower`). Максимальная мощность ветровых — тоже 15 МВт (константа `maxWindPower`). Генерируемая мощность практически пропорциональна яркости солнца (на самом деле не совсем так, но это несложно определить).

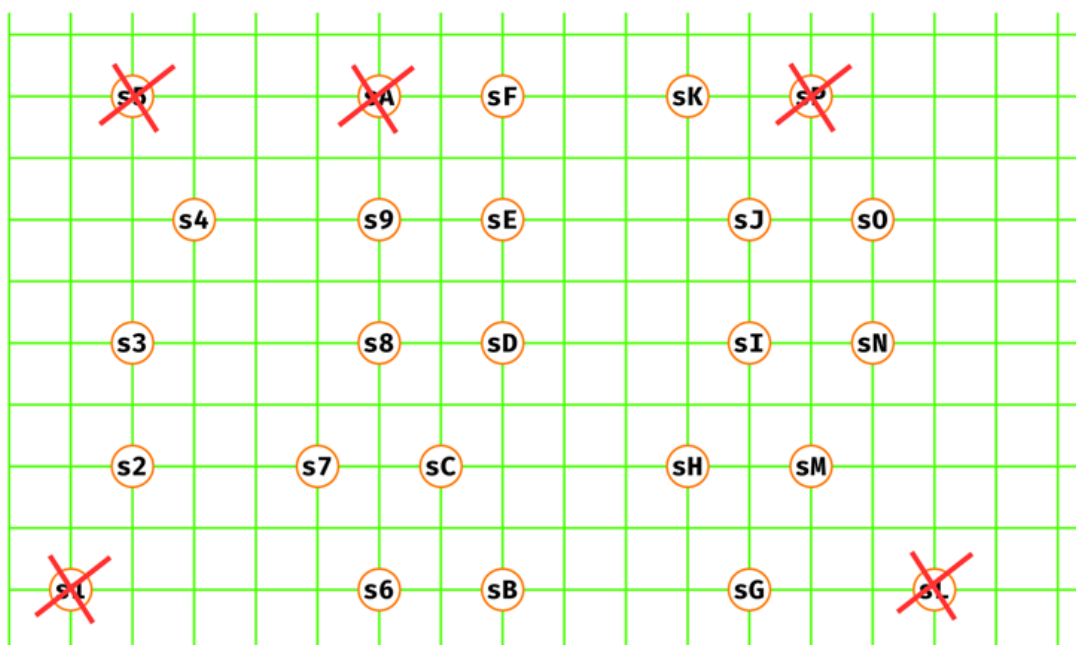


Рис. III.2.3. Расположение СЭС на сетке стола (1020×1720 мм)
Размер ячейки 100×100 мм

ВЭС

Генерируемая ветряками мощность пропорциональна кубу скорости ветра (точнее частоте их вращения, что не одно и то же). Есть предельная скорость их вращения (около 8 оборотов в секунду, константа `windBreakValue` — это доля от максимального значения, которое могут выдавать измерительные цепи ветряков), при достижении которой ветряк отключается и уходит в «штормовую защиту», из которой он выходит только при снижении его скорости до 87,5% от этой скорости (константы `windRecoveryValue/windBreakValue`). Кроме того, максимум генерации достигается при 86% (константы `windSummit/windBreakValue`) от предельной скорости, и далее не растет до самого отключения. Также обратите внимание, что скорость вращения ветряка при одном и том же ветре очень сильно зависит от его расположения.

Физическая инерция ветряков весьма велика, и может составлять до 30 секунд (это не программируемое значение, а физическое). Более точные данные вам нужно будет получить самим.

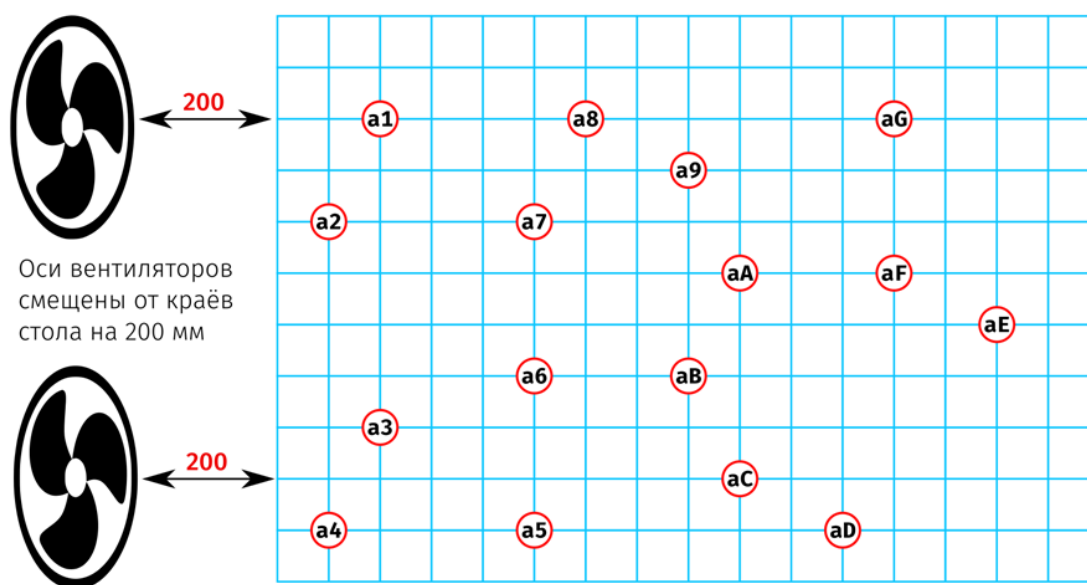


Рис. III.2.4. Расположение ВЭС на сетке стола (1100×1600 мм)
Размер ячейки 100×100 мм

ТЭС

Тепловая электростанция проще, чем СЭС и ВЭС в том, что она не имеет физической модели, однако это не обязательно означает, что с нею проще управляться.

На каждом такте ей нужно указывать, какой объем топлива ей следует сжигать. За топливо вы платите отдельно, сверх аукционного тарифа. 1 единица топлива превратится в 1 МВт мощности с учетом КПД и с добавлением инерции: к выработке сжиганием топлива добавится часть мощности, выработанной на предыдущем такте. Точнее, 60% (константа `tpsInertia`) от предыдущей мощности, уменьшенной на 0,5 (константа `tpsInertiaFriction`) МВт.

Единица топлива стоит 3,5 р (константа `tpsFuelCost`). При КПД в 100% ТЭС перерабатывала бы 1 единицу топлива в 1 МВт·такт энергии, однако ее КПД не только не идеален, но и не линеен. Он представляет собой параболу, максимум которой в 90% (константа `tpsEcePeakValue`) достигается при сжигании 8 единиц топлива (константа `tpsEcePeakPower`), а при сжигании 0 единиц топлива КПД составляет всего 40% (константа `tpsEceZeroValue`). Предельный объем сжигаемого ТЭС топлива — 10 единиц (константа `tpsMaxPower`).

Например, на прошлом такте ТЭС вырабатывала 10 МВт, на текущем ей указано сжечь 8 единиц топлива. Мощность на текущем такте составит $8 \cdot 0,9 + 0,6 \cdot (10 - 0,5) = 7,2 + 5,7 = 12,9$ МВт.

ТЭС имеют две точки подключения. Как и у заводов, если обе они активны, то мощность между ними распределяется поровну.

Накопители

Накопители разыгрываются на аукционе наравне с потребителями и электростанциями. Торгуются и тарифицируются они так же, как электростанции: ставки идут на повышение, и ставкой является фиксированная плата за каждый такт.

емкость накопителя составляет 100 МВт·такт (константа `cellObjectCapacity`), предельная скорость заряда или разряда — 15 МВт·такт (константы `cellObjectChargeRate` и `cellObjectDischargeRate`). Потерь энергии в них нет (это не означает, что их нет в реальных накопителях, это означает, что их модель на стенде заметно проще оригиналов). Зарядом и разрядом накопителей управляет ваш скрипт.

Монтаж сети

Проектирование сети

Сеть энергосистемы строится при помощи подстанций — главных, миниподстанций А (с тремя выходами) и миниподстанций Б (с двумя выходами). На каждую команду имеется одна миниподстанция Б. Ее стоимость 2 р./ход, и при желании ее можно не устанавливать. Тогда плата за нее взиматься не будет. Миниподстанции А разыгрываются на аукционе, и если приобрели одну из них, то установить ее вы обязаны.

Кольца и острова в энергосистеме создавать недопустимо, поэтому топология вашей энергосистемы — дерево. Энергосистема ветками (выходами и входами) подстанций разбивается на энергорайоны, внутри которых находятся потребители или электростанции. Энергорайон — это область «от подстанции до подстанции».

Каждый вход больницы должен приходить в разные ветки главной подстанции.

Электростанции и потребителей подключать в один энергорайон нельзя, они должны быть разделены подстанциями.

Проект сети

Проект сети вы передаете организаторам в виде формального описание в формате JSON, например:

```
[ { "address" : "h1"
  , "station" : "M1"
  , "line"    : 3
```

```

}
, { "address" : "m1"
  , "station" : "M1"
  , "line"    : 2
  , "comment" : "Наша миниподстанция"
}
, { "address" : "a8"
  , "station" : "m1"
  , "line"    : 1
}
, { "address" : "f5"
  , "station" : "m1"
  , "line"    : 2
}
, { "address" : "f6"
  , "station" : "M1"
  , "line"    : 1
}
]

```

Это список одинаковых объектов с тремя обязательными полями:

1. **«address»** — адрес объекта, который вы выиграли на аукционе. Он присваивается объекту по окончании аукциона.
2. **«station»** — адрес подстанции, к которому подключен объект. У каждой команды также главная подстанция, которая не входит в описание и сама никуда не подключается, в примере выше — «M1». Адрес главной подстанции каждой команды мы сообщим позднее.
3. **«line»** — номер линии, к которой подключается объект, нумерация с единицы. У главной подстанции три линии, у миниподстанции А — тоже три, у миниподстанции Б — две.

Другие поля нашей системой игнорируются. В примере выше — поле **«comment»** объекта «m1». Но обратите внимание, что при пересылке файлов иногда могут случаться странности с их кодировками, поэтому предпочтительно, чтобы файл проекта сети содержал только латиницу.

Биржа

В случае, если энергосистема не сбалансирована полностью (а так будет практически всегда), недостаток или избыток мощности ликвидируется через внешнюю по отношению к игроку энергосистему. Есть два способа это сделать:

1. Заранее отдать заявку на продажу или покупку нужной мощности. Через один ход (не на следующий!) заявка попадет в «биржевой стакан» (про него чуть ниже) и превратится в мощность и деньги.
2. Ничего не делать. Тогда дисбаланс будет ликвидирован на рынке мгновенной мощности, цены на котором фиксированы: 1 р/МВт (константа `exchange ExternalLastMomentSell`) для продажи и 10 р/МВт (константа `exchange ExternalLastMomentBuy`) для покупки.

Биржевой стакан устроен просто. Сначала все заявки делятся на два списка, на покупку и на продажу, и сортируются по невыгодности для своих отправителей. Затем заявки из этих двух списков сочетаются друг с другом, превращаясь в транзакции, начиная с самых невыгодных; цена сделки — средняя от цен заявок. Если мощ-

ности в заявках не совпадают, то неудовлетворенная часть большей заявки переходит дальше. Если заявки одного из типов закончились, то оставшиеся будут удовлетворены по фиксированным ценам — 2 р/МВт (константа `exchangeExternalAdvanceSell`) при продаже и 5 р/МВт (константа `exchangeExternalAdvanceBuy`) при покупке.

Также на бирже в случае заключения сделки между игроками действует комиссия: цена для покупателя поднимается на 10 копеек (константа `exchangeCommission`) от вычисленной, а цена для продавца — опускается.

Пример:

Заявки на покупку	Заявки на продажу
Заявка 1: 5 по 3,5	Заявка 3: 3 по 2,5
Заявка 2: 2 по 2,5	Заявка 4: 3 по 3

«Встречаются» заявки 1 и 3, заявка 3 удовлетворена полностью, а заявка 1 — частично:

Заявки на покупку	Заявки на продажу
Сделка 1.1–3: 3 по 3 (для покупателя 3 по 3,1, для продавца 3 по 2,9)	
Заявка 1.1: 3 по 3,5	Заявка 3: 3 по 2,5
Заявка 1.2: 2 по 3,5	Заявка 4: 3 по 3
Заявка 2: 2 по 2,5	

Встречаются оставшаяся часть заявки 1 и заявка 4, которая удовлетворяется частично:

Заявки на покупку	Заявки на продажу
Сделка 1.1–3: 3 по 3 (для покупателя 3 по 3,1, для продавца 3 по 2,9)	
Заявка 1.1: 3 по 3,5	Заявка 3: 3 по 2,5
Сделка 1.2–4.1: 2 по 3,25 (для покупателя 3 по 3,35, для продавца 3 по 3,15)	
Заявка 1.2: 2 по 3,5	Заявка 4.1: 2 по 3
Заявка 2: 2 по 2,5	Заявка 4.2: 1 по 3

Встречаются заявки 2 и остаток заявки 4. Заявка 2 удовлетворена частично:

Заявки на покупку	Заявки на продажу
Сделка 1.1–3: 3 по 3 (для покупателя 3 по 3,1, для продавца 3 по 2,9)	
Заявка 1.1: 3 по 3,5	Заявка 3: 3 по 2,5
Сделка 1.2–4.1: 2 по 3,25 (для покупателя 3 по 3,35, для продавца 3 по 3,15)	
Заявка 1.2: 2 по 3,5	Заявка 4.1: 2 по 3
Сделка 2.1–4.2: 1 по 2,75 (для покупателя 1 по 2,85, для продавца 1 по 2,65)	
Заявка 2.1: 1 по 2,5	Заявка 4.2: 1 по 3
Заявка 2.2: 1 по 2,5	

Для остатка заявки 2 не осталось встречных заявок. Она удовлетворяется из сети:

Заявки на покупку	Заявки на продажу
Сделка 1.1–3: 3 по 3 (для покупателя 3 по 3,1, для продавца 3 по 2,9)	
Заявка 1.1: 3 по 3,5	Заявка 3: 3 по 2,5
Сделка 1.2–4.1: 2 по 3,25 (для покупателя 3 по 3,35, для продавца 3 по 3,15)	
Заявка 1.2: 2 по 3,5	Заявка 4.1: 2 по 3
Сделка 2.1–4.2: 1 по 2,75 (для покупателя 1 по 2,85, для продавца 1 по 2,65)	
Заявка 2.1: 1 по 2,5	Заявка 4.2: 1 по 3
Сделка 2.2-Сеть: 1 по 5	
Заявка 2.2: 1 по 2,5	

Потери энергии

В энергосистеме есть потери. Они для простоты вычисляются на выходах подстанций, и пропорциональны квадрату проходящей через них мощности. Потери есть всегда и растут с нуля при нагрузке на энергорайон в 0 МВт до уровня в 25% (константа `lossesLimit`) при 30 МВт (константа `lossesThreshold`) (после чего не растут). Это значит, что если у вас на линии есть генерация в 33 МВт, то потери составят $33 \cdot 0,25 = 8,25$ МВт, и реальная нагрузка на линию (и количество энергии, которое из нее выйдет) составит 24,75 МВт. Если же у вас на линии есть потребление в 21 МВт, то потери составят $21 \cdot 0,175 = 3,675$ МВт, и реальная нагрузка (сколько в нее нужно закачать, чтобы удовлетворить потребителей) составит 24,675 МВт.

Соединение с внешней энергосистемой тоже имеет ограничение мощности, 50 МВт (константа `externallimit`), и если будет превышено оно, то отключена будет вся энергосистема целиком, что может быть баснословно убыточно. Впрочем, баснословно убыточно подойти даже наполовину к этому ограничению.

Износ подстанций

Оборудование подстанций, управляющее линиями, изнашивается. Износ накапливается, и скорость его накопления зависит от нагрузки на линию. При износе выше определенного уровня появляется вероятность аварии — события, при котором линия отключится на 5 тактов. Вероятность аварии зависит от износа по формуле:

$$p(W) = \frac{1}{1 + \exp(4 - 8z)},$$

где

$$z = \frac{W - 0,8}{1 - 0,8};$$

W — накопленный износ линии, от t до 1.

Если эту формулу упростить, можно получить что-то такое:

$$p(W) = \frac{1}{1 + \exp(36 - 40W)},$$

Если степень износа превысила 100%, то авария произойдет гарантированно.

Например, при значении износа 85% вероятность аварии составит 11,9%, при 90% — 50%, при 93% — 76,8%.

Накопление износа за такт (обозначено строчной w) зависит от нагрузки на линию по формуле:

$$w(x) = \frac{x^{1,9}}{6},$$

где x — отношение действительной нагрузки на линию к номинальной. Номинальная нагрузка составляет 30 МВт (константа `lossesThreshold`). Например, если нагрузка на линии с учетом потерь составляет 10 МВт, то ее износ увеличится на 2,06% (процентных пунктов), если 23 МВт — 10,6%, 35 МВт — 22,34%.

После аварии износ линии сбрасывается на 0. Линия при этом останется отключенной, ее нужно будет включить скриптом.

Если неаварийную линию принудительно отключить, то во время отключения на ней будет проводиться текущий ремонт. Износ линии будет уменьшаться на 40 процентных пунктов за такт (константа `wearRecoveryRate`).

Скрипты

Скрипты пишутся на языке Python 3. Они загружаются в систему через вкладку «Скрипты» пользовательского интерфейса. Загруженный скрипт может находиться в системе продолжительное время, и его можно в любой момент заменить. Внутри он выполняется один раз каждый такт игры. Он видит те же данные, что и вы в интерфейсе анализа, но в отличие от вас, может на них реагировать, отправляя управляющие воздействия — приказы. Их немного:

1. Включение/отключение энергорайона. Обратите внимание, что отключение энергорайона отключает и все зависимые от него, но включение — нет, и включать нужно будет все по-отдельности.
2. Отправить заявку на покупку или продажу энергии на бирже. Мощность ограничена 50-ю (константа `exchangeMaxAmount`) мегаваттами, а цена — вилкой цен гарантирующего поставщика: ее можно выставить от 2 до 5 (константы `exchangeExternalAdvanceSell`, `exchangeExternalAdvanceBuy`) р/МВт.
3. Установить количество топлива, которое следует сжечь ТЭС. Предельное количество — 10 единиц (константа `tpsMaxPower`). Если на такте указаний для ТЭС не пришло, считается, что пришел приказ «ноль».
4. Зарядить/разрядить накопитель.
5. Вывести данные в зону пользовательских графиков. О них подробнее в справке скрипта.

Подробнее о том, как с помощью скрипта получать данные об энергосистеме, смотрите в справке скрипта (она есть в интерфейсе и представлена в приложении 1).

Решение

В ходе выполнения задачи оценивалось комплексное решение задачи профиля. Выделение и формулирование подзадачи является частью интегральной (главной) задачи профиля. Проверкой решения подзадачи являлся ее вклад в результат игры.

Описанные ниже задачи выделены разработчиками и были известны участникам из описания. Распределение усилий между подзадачами принимали сами команды.

Определение взаимосвязи между яркостью освещения и генерацией солнечных батарей

Эта задача возникает на 5-м этапе игры, при моделировании энергосистемы. Вырабатываемая мощность солнечных батарей зависит от напряжения на солнечных панелях модели солнечной электростанции на стенде. Напряжение на солнечных панелях по отношению к яркости светильников, строго говоря, нелинейно. Однако характеристики солнечных панелей, измеряющих цепей и светильников подобраны так, что отклонение реальных значений от линейной их аппроксимации составляет не более 2%, и в дальнейшем откалиброваны до полной линейности внутренним ПО солнечных батарей.

Время релаксации измерительной системы солнечных батарей в 2 раза меньше минимального интервала между изменением яркости и измерением вырабатываемой мощности, поэтому генерация солнечных батарей зависит только от погоды на текущем такте игры.

В соревновании этого года зависимость генерации от фактической яркости освещения составила $P(L) = \max(0; 0,91L + 2, 81)$. Коридор отклонений модели от фактических данных (от $-0,2$ до $+0,3$) составляет меньше, чем заложенное в модель случайное отклонение фактических значений яркости освещения от прогнозных (от $-0,5$ до $+0,5$).

Определение взаимосвязи между скоростью ветра и генерацией ветряков

Эта задача возникает на 5-м этапе игры, при моделировании энергосистемы. Кроме того, команда должна иметь решение этой задачи для эффективной работы на этапах 1–3 — при проектировании энергосистемы и при участии в аукционе.

Задача вычисления генерации ветровых электростанций по данным погоды похожа на такую же задачу для солнечных батарей, но является более сложной.

Для получения величины генерации используется скорость вращения анемометра, который установлен на модели ветровой электростанции. Устройство анемометра — вертикально-осевой; его лопасти устроены так, что скорость его вращения линейно пропорциональна скорости ветра (если считать поток ветра гомогенным). В случае постоянной негомогенности ветрового потока (когда анемометр не перемещается) скорость вращения анемометра также линейна по отношению к максимальной скорости ветра в потоке.

Измерительная система — инерциальная система вращения с трением. Ее параметры нужно вычислять по данным прошедших игр. Время полной остановки анемометра с максимальной скорости вращения составляет около 5 тактов игры (оно зависит от максимальной скорости ветра в месте расположения анемометра). Время полного разгона аналогично составляет 2 такта игры.

Генерируемая мощность пропорциональна кубу скорости вращения анемометра. При достижении максимального уровня генерации (15 МВт) мощность далее не растет, но при увеличении скорости вращения еще на 30% ветряк уходит в «штормовой

режим» и генерирует 0 до дех пор, пока скорость вращения не уменьшится.

Зависимость генерации от скорости ветра можно оценить большим числом способов на основании данных прошедших игр. Возможно как построить собственную модель для каждого ветряка, так и построить общую для всех, со свободным параметром для калибровки модели под каждый конкретный ветряк.

Например, при построении единой модели как линейной комбинации от скорости ветра за 5 последних тактов зависимость будет такой:

$$P(w_0, w_{-1}, w_{-2}, w_{-3}, w_{-4}) = \\ = k \frac{(0,987w_0 + 0,332w_{-1} + 0,154w_{-2} + 0,099w_{-3} + 0,051w_{-4})^3}{1008},$$

где w_{-n} — скорость ветра n тактов назад, а коэффициент $k \in (0; 1]$ подбирается для каждого ветряка индивидуально по прошлым данным (знаменатель дроби подобран так, чтобы для наилучшего ветряка коэффициент k составлял единицу).

При вычислениях необходимо учитывать тот факт, что генерация ветровых электростанций ограничена правилами на уровне 15 МВт. Поэтому если прогнозируемая генерация превышает этот уровень, надо считать, что спрогнозировано именно 15.

Средняя величина ошибки такого набора коэффициентов между прогнозируемой и реальной генерацией на играх заключительного этапа составила 5,9%, что, с одной стороны, связано с большой инерционностью физического анемометра. С другой стороны, из-за кубической зависимости генерации от силы ветра, любые погрешности «в середине» диапазона скоростей ветра очень сильно влияют на прогнозируемую мощность. Если скорость ветра мала или велика, погрешности влияют очень слабо.

Проектирование энергосистемы

Эта задача возникает на 4-м этапе игры, сборке сети, или даже на этапах 1–3, если участники системно подходят к задачам проектирования энергосистемы. В ней имеющиеся объекты нужно распределить по энергосистеме с использованием подстанций, дополнительных модулей и с учетом ограничений правил. Цель проектирования сложна и зависит от участников. Ее возможные компоненты:

1. Минимизация суммарных потерь.
2. Минимизация стоимости обслуживания инфраструктуры.
3. Минимизация рисков перегрузки.
4. Минимизация экономических потерь в аварийном режиме внешней энергосистемы.
5. Минимизация потерь через активное маневрирование мощностью при помощи модулей.
6. Возможность решить эту задачу быстро для перепроектирования энергосистемы «на лету» во время аукциона.
7. Взвешенная минимизация рисков штрафов при возможных отключениях.
8. Увеличение возможности маневрировать мощностью при недостатке или чрезмерной стоимости энергии через отключения.

Это одна из интегральных задач стенда, в которой собираются все остальные (вторая такая — аукцион), для данной задачи существует множество решений близких к оптимальному. Задача точного поиска глобального экстремума является достаточно

сложной, однако нахождение локального экстремума, который незначительно отличается от глобального является не только разрешимой, но и используется нами для калибровки решений предлагаемых участниками. Каждая команда с этой задачей на том или ином уровне справилась, будь то через написание экспертной системы для автоматической оптимизации сети, системы прогнозирования и визуализации, или просто в тетрадке. Это задача, которую очень легко решить «на троечку», возможно решить «на пять с плюсом» и возможно решить полностью только в рамках выбранного командой сужения.

Вычисление полного энергетического баланса на основании данных прогнозов

Эта задача возникает на 1-м и 5-м этапах игры, при анализе прогнозов и моделировании энергосистемы.

Все время игры командам доступны все данные о прогнозах погоды и действующих контрактах. Из них можно вычислить прогноз дефицита или профицита мощности на каждый такт. Для этого нужно на основании составленных заранее игроками моделей вычислить из прогнозов погоды прогнозы генерации. Из результирующего множества случайных величин (вероятная генерация для каждой электростанции и вероятное потребление для каждого потребителя) нужно вычислить их сумму. Она будет представлять собой распределение вероятностей профицита/дефицита мощности в системе. Важно учитывать, что максимальный дефицит или профицит мощности ограничен главной подстанцией и установленными на ней объектами.

В задаче этого года, в отличие от прошлых лет, предельная возможная сложность энергосистем намного выше, поэтому приведенное ниже решение достаточно эффективно работает только для достаточно простых систем. В более сложных системах может понадобиться дополнительный учет потерь, который сильно зависит от конфигурации сети. Это приводит к тому, что эта задача становится значительно более интегрированной в «систему поддержки принятия решений», которую явно или неявно для себя создавали так или иначе все команды.

Вычисление баланса энергорайонов энергосистемы

Каждая энергосистема состоит из набора подстанций и подключенных к ним объектов. Эффективно энергосистема представляет собой дерево, в узлах которого находятся подстанции, а ребра представляют собой энергорайоны с множеством подключенных объектов и, что очень важно, двумя узлами подстанций, к которым подключен энергорайон (причем если энергорайон физически подключен только к одной подстанции, то второй узел виртуален). По правилам мощность, протекающая через один узел, не может быть больше 25 МВт (без учета потерь), иначе узел аварийно отключится. Это приводит к тому, что необходимо прогнозировать энергобаланс в каждом энергорайоне по-отдельности (полностью аналогично нахождению общего баланса), и учитывать складывание токов в энергосистеме при протекании их по дереву.

Вычисление экономического баланса энергосистемы на основании данных прогнозов

Эта задача возникает на 1-м и 5-м этапах игры, при анализе прогнозов и моделировании энергосистемы. Кроме того, команда должна иметь решение этой задачи для эффективной работы на этапах 2–3 — участии в аукционе.

Далее нужно решить задачу нахождения вероятностного распределения экономических потерь. Это делается почти тривиально при принятии консервативной оценки прибылей и убытков от взаимодействия с внешней энергосистемой: каждый мегаватт непредсказанной избыточной мощности эффективно несет убыток в 1 очко, а каждый мегаватт недостаточной — 2 очка. К этим значениям нужно добавить стоимость электроэнергии, закупленной/проданной во внешней энергосистеме по цене за 1 такт вперед. Далее математическое ожидание получившейся случайной величины нужно минимизировать, используя параметры направляемой/извлекаемой мощности из аккумуляторов и покупаемой/продаваемой мощности во внешней энергосистеме. Выгоды от торговли с другими командами и ранних закупок во внешней энергосистеме можно учитывать независимо.

Эта задача немного проще, чем кажется из-за того, что использование аккумуляторов экономически намного выгоднее взаимодействия с внешней энергосистемой, имеет смысл закупать/продавать электроэнергию в ней только в случае невозможности сделать это через аккумулятор. Эти два параметра оказываются связаны в один — «балансирующее воздействие на энергосистему», которое можно разбить на аккумуляторы и внешнюю энергосистему «задним числом» после решения этой задачи.

Эту задачу (нахождение балансирующего воздействия, минимизирующего математическое ожидание экономических потерь) недостаточно решить аналитически, алгоритм решения нужно реализовать также в управляющем скрипте (и других вспомогательных программах), что для большинства школьников является нетривиальной и неустойчивой к ошибкам задачей. Немного проще ее решить «перебором» — перебрав все значения балансирующего воздействия в размахе случайного распределения дефицита/профицита мощности в системе с фиксированным шагом, например, 0,1. Такой точности будет вполне достаточно, такое решение можно реализовать быстрее, чем точное, и впоследствии при наличии времени его можно точным заменить.

В дальнейшем на основании этих данных можно вычислить распределение вероятностей прибыли за всю игру.

Расчет предельной цены объекта

Эта задача возникает на 2-м и 3-м этапах игры, при участии в аукционе.

Это задача нахождения предельной цены контракта объекта на аукционе — такой цены, приобретение контракта по которой ожидаемо не принесет ни прибылей, ни убытков. Это расширение над задачей нахождения полного экономического баланса энергосистемы — достаточно вычислить суммарную прогнозируемую прибыль энергосистемы с этим объектом и без него. Для электростанций разницу между этими величинами нужно разделить на число тактов в игре. Для потребителей — разделить на суммарное потребление его за всю игру (это случайная величина; для примерной оценки ее можно заменить на математическое ожидание, но на этом этапе у команд

уже должно быть в избытке опыта вычислений со случайными величинами).

Получившееся число: для электростанций — максимальная осмысленная цена, для потребителей — минимальная.

С учетом того, что все объекты одного класса сделаны аналогичными, ее расчет несложно сделать даже средствами электронных таблиц Excel или их аналогов.

Составление и адаптация стратегии для аукционов

Эта задача возникает на 2-м и 3-м этапах игры, при участии в аукционе. Хотя команды могут найти оптимальную энергосистему для любого набора прогнозов погоды, эту энергосистему им еще нужно «отвоевать» у конкурентов. Эта игра в целом относится к рефлексивным, игры этого класса не имеют устойчивого решения, а использование смешанных стратегий едва ли оправдано на одном прогоне игры. Участникам нужно анализировать поведение оппонентов, предугадывать их цели, и искать способы помешать целям оппонентов и защитить свои. Для этого можно создать несколько вспомогательных инструментов:

1. Нахождение эффективных конфигураций энергосистем. Не только оптимальной, но всех, которые достаточно хороши. Все команды будут стремиться к какой-то из них, и для любой промежуточной ситуации на аукционе можно предположить, к какой из них будут стремиться оппоненты.
2. Нахождение оптимальной ставки на аукционе, исходя из ценности лота для себя и оппонентов. Ценности лота для оппонентов можно оценить из предположения об энергосистеме, к которой они стремятся, с использованием решения задачи расчета прогноза рентабельности. В этих условиях оптимальной ставкой будет максимальная ставка всех оппонентов, при условии, что она не превышает собственной максимальной ставки. Далее участникам будет интересно от этой ставки отклониться, чтобы рискнуть и увеличить выгоду, либо нарушить стратегию оппонентов (от одного приобретенного по некорректной цене объекта, согласно правилам аукциона, можно избавиться в конце аукциона).

Также в течение всего времени аукциона командам нужно оценивать риск того, что целевую энергосистему составить не удастся, и при необходимости переходить к альтернативным вариантам.

Работа с биржей электроэнергии

Эта задача, тривиальная сама по себе, значительно углубляет задачи вычисления экономического баланса энергосистемы и составления стратегии для аукциона.

Механика аукциона устроена таким образом, что те, кто устанавливают цены заявок хоть чуть-чуть хуже для себя, чем их оппоненты, получают преимущество и, потенциально, более выгодную цену. С другой стороны, в случае выставления на биржу дефицитного товара, сбивание цены приведет только к потере очков, и больше ни к чему. Более того, механика устроена таким образом, что не взаимодействовать с ней участники не могут, поэтому эту задачу можно во многом обойти, если спроектировать энергосистему так, что она всегда будет предлагать на биржу дефицитный товар (товар здесь — избыток или недостаток энергии, или, иными словами, генерация или потребление). В целом же задача сводится к прогнозированию того, будет на бирже в дефиците спрос или предложение, и в случае размещения не дефицитного товара, подстраивания цены под прошлое поведение оппонентов.

Задача осложнялась наличием механики аварийного режима на внешней ЛЭП. На тактах игры с 15 по 25, с 40 по 60 и с 65 по 85-й внешняя ЛЭП находилась в аварийном режиме, при котором переток мощности более 10 МВт приводил к штрафу в 10 р/МВт превышения. При этом взаимодействия между командами эта механика не затрагивала. Она приводила к необходимости либо тщательно балансировать собственную энергосистему, либо учитывать энергобаланс и ценовые тактики других команд.

Эта задача возникает на 5-м этапе игры, при моделировании энергосистемы.

Вычисление оптимальных моментов для вывода веток подстанций в ремонт

Согласно модели, каждая ветка подстанции изнашивается пропорционально квадрату проходящей через нее мощности. При достижении износа в 90% появляется вероятность отказа, которая увеличивается по логистической кривой и достигает 100% (гарантированного отказа) при 100%-ом износе. При аварийном отказе ветка отключается на 5 тактов игры. Если ветку отключить заранее, то она за каждый такт отключения будет терять 40 процентных пунктов износа.

Игнорирование этого условия приводит к большим штрафам за длительное отключение потребителей и большое количество выпадающей мощности, которую нужно компенсировать закупками дорогой энергии на бирже.

Учитывать это условие можно двумя способами. Во-первых, можно составить эвристическое управление, вроде «как только износ превышает 40%, отключаем ветку» или «если износ превышает 40% и если нагрузка на ветке составляет менее 6 МВт, то выводим ее на ремонт, а если износ превышает 80%, то выводим на ремонт безусловно» и тому подобное. Второй подход — по собственной модели энергосистемы на основании прогнозов решить оптимизационную задачу по составлению плана ремонтов.

На практике достаточно набора эвристических правил, либо гибридной стратегии: заранее составлять план отключений, но при значительном его расхождении с реальной ситуацией переключаться на эвристики. В целом при решении этой задачи очень быстро наступает момент, когда затраты на улучшение ее решения начинают превышать положительный эффект от улучшения, в сравнении с вложением времени в решение других задач.

Система поддержки принятия решений на аукционе

Это первая точка сборки решений предметных задач, возникающих в командном туре.

Основная задача таких систем — вычисление ценности лота при заданных параметрах энергосистемы.

Самый примитивный вариант такой системы может быть устроен следующим образом:

1. Имеются прогнозы погоды и потребления на следующую игру. Мы будем считать, что реальные значения будут точно соответствовать прогнозам.
2. Для каждого такта игры вычисляем генерацию. Например, солнечные батареи вычисляем из яркости солнца с найденным ранее коэффициентом конверсии

солнечной энергии в мощность, например, 1 МВт/284 лк.

3. Вычисляем энергобаланс в каждый такт игры.
4. Вычисляем изменение счета в каждый такт игры.
5. Добавляем в энергосистему интересующий нас объект.
6. Повторяем шаги 1–4.
7. Сравниваем результаты для энергосистемы при наличии и без наличия интересующего объекта.
8. Для электростанций оптимальная цена есть их цена плюс разница результатов энергосистем, деленная на число тактов игры.
9. Для потребителей оптимальная цена есть их цена плюс разница результатов игры, деленная на потребленную потребителем энергию.

Хорошую систему от примитивной могут отличать следующие характеристики:

- Учет погрешностей прогнозов. Расчет наихудшего варианта, наилучшего, наиболее вероятного и других статистических характеристик.
- Использование более точных оценок генерации.
- Расчет стоимости лота для энергосистем оппонентов — чтобы выиграть лот, нужно ставить не собственную цену, а цену, большую, чем у оппонентов. Для этого необходимо оценивать ценность лота для оппонентов.
- Прогноз эффективности задуманной энергосистемы.
- Расчет величины риска в зависимости от размера ставки — для этого нужно использовать данные об энергосистемах оппонентов и опыт взаимодействия с ними.

Важно, что такая система лишь помогает принимать решения, и не гарантирует того, что команда с наилучшей реализацией этой задачи наиболее эффективно проведет аукцион.

Разработанные командами программы варьировались по сложности от простых таблиц Excel, до веб-сервисов с элементами ИИ.

Управляющие скрипты

Это вторая точка сборки предметных задач, возникающих в командном туре.

Задача управляющего скрипта — используя возможность управления продажей электроэнергии во внешнюю энергосистему, управления аккумуляторами, прогнозирования генерации, максимизировать число очков, которое получит команда за командный тур.

Если эта задача не решена, то результаты команды будут плохими независимо от результатов аукциона и глубины решения предметных задач командного тура.

Самый простой вариант скрипта может действовать, например, по следующему алгоритму:

1. Включить все отключенные линии.
2. Оценить генерацию на следующем такте. Вычислить энергобаланс следующего такта.
3. ЕСЛИ энергобаланс положителен, ПЕРЕЙТИ к п. 6
4. Попытаться ликвидировать дефицит из аккумуляторов.

5. Ликвидировать дефицит из внешней энергосистемы.
6. ЗАВЕРШИТЬ РАБОТУ.
7. Попытаться ликвидировать профицит, перенаправив мощность в аккумуляторы.
8. Ликвидировать профицит, продав мощность во внешнюю энергосистему.
9. ЗАВЕРШИТЬ РАБОТУ.

Хороший скрипт может обладать следующими характеристиками:

- Оценки энергобаланса на всю игру вперед и ранняя закупка электроэнергии.
- Коррекция прогнозов генерации на основании реальных данных от электростанций.
- Использование распределения вероятных значений энергобаланса, чтобы вычислять средневзвешенную величину его коррекции: дефицитный энергобаланс обходится дороже профицитного, соответственно нужно минимизировать не модуль энергобаланса, а математическое ожидание экономических потерь в результате ошибок прогнозов.
- Такое управление аккумуляторами, которое полностью разряжает их к последнему такту игры.
- Прогнозирование возможностей аварийных отключений в энергосистеме и использование возможности ограниченного регулирования мощности потребителей и электростанций их избегания.
- Использование возможности ограниченного регулирования мощности потребителей и электростанций для увеличения экономической эффективности энергосистемы (в некоторых ситуациях плата за такое регулирование может быть меньше убытков на полноценное снабжение потребителя или плата за повышение генерации — меньше затрат на закупку мощности).
- Моделирование состояния энергосистем оппонентов для предсказания будущих состояний биржи.
- Уточнение моделей энергосистем оппонентов на основании реального состояния биржи.
- Управление риском: в ряде случаев осмысленно использование неоптимальных характеристик управления, которые несмотря на то, что они снижают математическое ожидание счета в командном этапе, увеличивают вероятность обойти другую команду.

Критерии оценивания

Проводится три соревновательных игры. В каждой игре вся совокупность принятых участниками решений интегрально оценивается стендом в автоматическом режиме. Каждая игра проводится с разными прогнозами. За каждую игру очки, заработанные участниками, переводятся в рейтинговые баллы по формуле:

$$p_i = 100 \cdot (\max(m, x_i) - m) / (M - m),$$

где:

p_i — рейтинговый балл i -й команды;

x_i — очки i -й команды за игру;

M — наилучший из всех команд результат за игру;

m — второй с конца результат за игру.

За все три игры результат складывается, после чего вычисляется итоговый балл (s_3) умножением на 100 и делением на максимальную сумму рейтинговых баллов.

Команда-победитель — это команда, набравшая наибольшее количество итоговых баллов.

Критерии определения победителей и призеров

Первый отборочный этап

В первом отборочном этапе участники решали задачи по двум предметам: математика и информатика, в каждом предмете максимально можно было набрать 100 баллов. Для того, чтобы пройти во второй этап участники должны были набрать в сумме по обоим предметам не менее 50 баллов, не зависимо 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 758 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 практический тур

Проводится три игры. В каждой игре вся совокупность принятых участниками решений интегрально оценивается стендом в автоматическом режиме. Каждая игра проводится с разными прогнозами. За каждую игру очки, заработанные участниками, переводятся в рейтинговые баллы по формуле:

$$p_i = 100 \cdot \frac{\max(m, x_i) - m}{M - m},$$

где:

- p_i — рейтинговый балл i -й команды;
- x_i — очки i -й команды за игру;
- M — наилучший из всех команд результат за игру;
- m — второй с конца результат за игру.

За все три игры результат складывается, после чего вычисляется итоговый балл (S_3) умножением на 100 и делением на максимальную сумму рейтинговых баллов.

Команда-победитель — это команда, набравшая наибольшее количество итоговых баллов.

В заключительном этапе олимпиады баллы участника складываются из двух частей, каждая из которых имеет собственный вес: баллы за индивидуальное решение задач по предметам (математика, информатика) с весом 0,2 каждый предмет и баллы за командное решение практической задачи в области интеллектуальных энергетических систем с весом 0,6. Итоговый балл определяется по формуле:

$$S = 0,2 \cdot (S1 + S2) + 0,6 \cdot S3,$$

где:

- $S1$ — балл первой части заключительного этапа по математике в стобалльной системе ($S1_{\text{макс}} = 100$);
- $S2$ — балл первой части заключительного этапа по информатике в стобалльной системе ($S2_{\text{макс}} = 100$);
- $S3$ — итоговый балл командного тура в стобалльной системе ($S3_{\text{макс}} = 100$).

Итого максимально возможный балл по условиям общего рейтинга:

$$0,2 \cdot (100 + 100) + 0,6 \cdot 100 = 100 \text{ баллов.}$$

Критерий определения победителей и призеров (независимо от класса)

рейтинг, где 9-ые классы участвовали на общих основания с 10–11 классами. С начала рейтинга были выбраны 3 победителя и 7 призеров (первые 25% участников рейтинга становятся победителями или призерами — первые 8% участников рейтинга становятся победителями, оставшиеся 17% — призерами).

Критерий определения победителей и призеров (независимо от уровня):

Категория	Количество баллов
Победители	74 и выше
Призеры	от 48,4 до 70