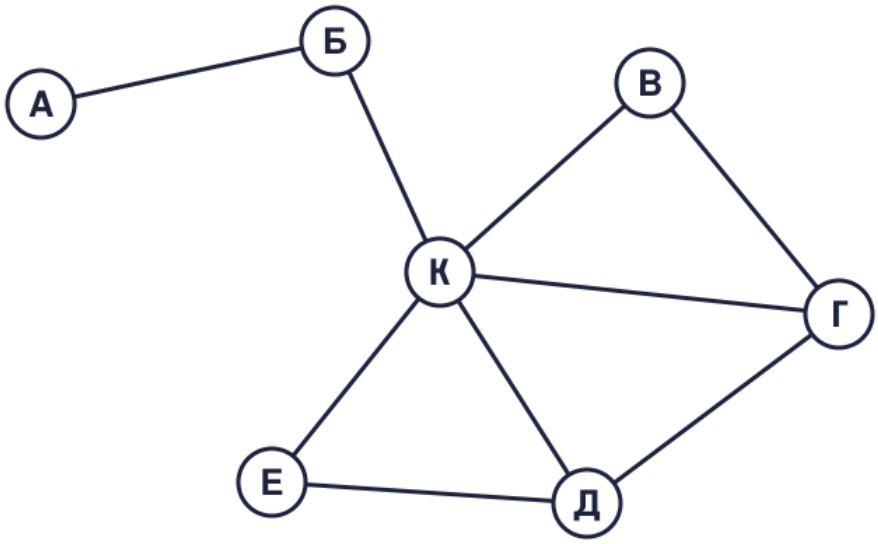
ЕГЭНИЙ · Информатика

Вариант 13 · Вариант с ответами и решениями

Задание 1

На рисунке схема дорог N-ского района изображена в виде графа, в таблице содержатся сведения о протяжённости каждой из этих дорог (в километрах).

Номер пункта	1	2	3	4	5	6	7
1		22	49	40	33		36
2	22				35		
3	49			21	14		
4	40		21				
5	33	35	14				
6							8
7	36					8	



Так как таблицу и схему рисовали независимо друг от друга, нумерация населённых пунктов в таблице никак не связана с буквенными обозначениями на графе. Определите, на сколько километров длина дороги АБ отличается от длины дороги БК.

В ответе запишите целое число.

Ответ 28

Посчитаем степени вершин (количество дорог из пункта). На графе: А — 1, Б — 2, В — 2, Г — 3, Д — 3, Е — 2, К — 5. В таблице: П1 — 5, П2 — 2, П3 — 3, П4 — 2, П5 — 3, П6 — 1, П7 — 2.

Сопоставляем вершины по степеням и по соседям: степень 1 на графе только у вершины А, в таблице — только у пункта 6: А = П6; Б — вершина степени 2, соединённая с А; этому соответствует только П7;

степень 5 на графе только у вершины К, в таблице — только у пункта 1: К = П1.

Дорога АБ — это П6–П7, её длина 8 км. Дорога БК — это П7–П1, её длина 36 км.

Разность: |8 – 36| = **28**.

Задание 2

Миша заполнял таблицу истинности логической функции F

(x ∧ ¬z) ∨ (z ≡ y) ∨ w,

но успел заполнить лишь фрагмент из трёх различных её строк, даже не указав, какому столбцу таблицы соответствует каждая из переменных w, x, y, z.

???	???	???	???	F
0	1	1		0
	0	1		0
1				0

Определите, какому столбцу таблицы соответствует каждая из переменных w, x, y, z.

В ответе напишите буквы w, x, y, z в том порядке, в котором идут соответствующие им столбцы (сначала буква, соответствующая первому столбцу; затем буква, соответствующая второму столбцу, и т. д.). Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно.

Например, если функция задана выражением ¬x ∨ y, зависящим от двух переменных, а фрагмент таблицы имеет вид

???	???	F
0	1	0

то первому столбцу соответствует переменная y, а второму — переменная x. В ответе следует написать: yx.

Ответ yxzw

Функция равна 0 только в 3 строках полной таблицы истинности из 16, поэтому фрагмент почти однозначно задаёт расстановку переменных. Переберём программой все способы заполнить пустые клетки и все перестановки переменных по столбцам: подходят только заполнения, при которых три строки различны, а функция во всех трёх строках равна 0.

Импликацию a → b в Python удобно записывать как `a <= b`, тождество — как `a == b`.

Программа на Python

```
from itertools import product, permutations

def F(x, y, z, w):
    return (x and not z) or (z == y) or w

answers = set()
for a1, a2, a3, a4, a5, a6 in product([0, 1], repeat=6):
    t = [(0, 1, 1, a1), (a2, 0, 1, a3), (1, a4, a5, a6)]
    if len(set(t)) == 3:              # строки фрагмента различны
        for p in permutations('xyzw'):
            if all(F(**dict(zip(p, r))) == 0 for r in t):
                answers.add(''.join(p))
print(*answers)
```

Программа выводит единственную перестановку: **yxzw**.

Задание 3

Задание выполняется с использованием прилагаемого файла.

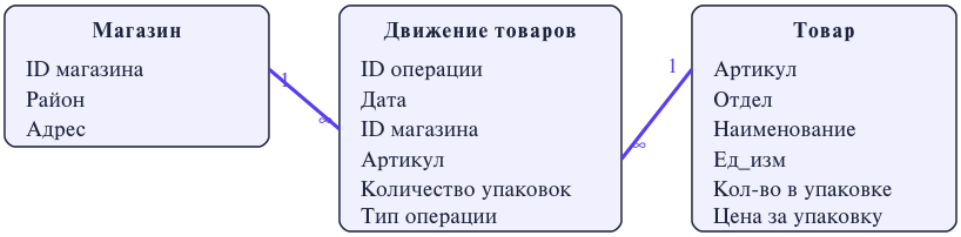
В файле приведён фрагмент базы данных «Канцтовары» о поставках товаров в магазины районов города. База данных состоит из трёх таблиц.

Таблица «Движение товаров» содержит записи о поступлении товаров со склада в магазины в течение июня 2026 г., а также информацию о проданных товарах. Поле *Тип операции* содержит значение *Поступление* или *Продажа*, а в соответствующее поле *Количество упаковок, шт.* внесена информация о

том, сколько упаковок товара поступило в магазин или было продано в течение дня.

Таблица «Товар» содержит информацию об основных характеристиках каждого товара: артикул, отдел, наименование, единицу измерения, количество в упаковке и цену за упаковку. Таблица «Магазин» содержит информацию о местонахождении магазинов: ID магазина, район, адрес.

На рисунке приведена схема указанной базы данных.



Используя информацию из приведённой базы данных, определите, на сколько увеличилось количество упаковок всех видов тетрадей в магазинах Октябрьского района за период с 2 по 13 июня включительно. В ответе запишите только число.

Файл к заданию: 3.xlsx (107 КБ)

Ответ 351

В электронной таблице. На листе «Движение товаров» добавьте столбцы: район магазина `=ВПР(C2;$Магазин.$A$2:$B$30;2;0)`, наименование `=ВПР(D2;$Товар.$A$2:$F$100;3;0)`.

Включите автофильтр и оставьте: даты с 02.06.2026 по 13.06.2026, район — Октябрьский, наименование начинается со слова «Тетрадь». Увеличение = все поступившие упаковки минус все проданные за этот период. В таблице удобно завести столбец со знаком: количество для поступления и минус количество для продажи. Сумму видимых строк даст `=ПРОМЕЖУТОЧНЫЕ.ИТОГИ(9;…)`.

Программой. Сохраните каждый лист как CSV с разделителем «;» (Файл → Сохранить как → Текст CSV) и выполните программу.

Программа на Python

```
import csv

def read(name):
    with open(name, encoding='utf-8') as f:
        return list(csv.reader(f, delimiter=';'))[1:]

goods = {r[0]: r for r in read('товар.csv')}      # артикул -> строка
shops = {r[0]: r[1] for r in read('магазин.csv')}  # магазин -> район
total = 0
for oid, date, shop, art, cnt, typ in read('движение.csv'):
    day = int(date[:2])
    g = goods[art]
    if 2 <= day <= 13 and shops[shop] == 'Октябрьский' and g[2].startswith('Тетрадь'):
        if typ == 'Поступление':
            total += int(cnt)
        else:
            total -= int(cnt)
print(total)
```

Ответ: **351**.

Задание 4

По каналу связи передаются сообщения, содержащие только буквы из набора: К, Л, О, П, Т. Для передачи используется двоичный код, удовлетворяющий условию Фано. Кодовые слова для некоторых букв известны: О – 1010, Т – 100. Для трёх оставшихся букв К, Л и П кодовые слова неизвестны.

Какое наименьшее количество двоичных знаков потребуется для кодирования слова ПОТОЛОК, если известно, что оно закодировано минимально возможным количеством двоичных знаков?

Примечание. Условие Фано означает, что никакое кодовое слово не является началом другого кодового слова. Это обеспечивает возможность однозначной расшифровки закодированных сообщений.

Ответ 21

Нарисуем двоичное дерево известных кодов. Свободные вершины, с которых может начинаться новый код: 0, 1011, 11. Чем чаще буква встречается в слове, тем короче должен быть её код.

Количество букв в слове: К — 1, Л — 1, О — 3, П — 1, Т — 1. Оптимально, например: К – 11, Л – 01, П – 00.

Длина кода слова: 2·1 + 2·1 + 4·3 + 2·1 + 3·1 = **21**.

Проверим перебором всех кодов длиной до 5 для неизвестных букв:

Программа на Python

```
from itertools import product

known = {'О': '1010', 'Т': '100'}
word = 'ПОТОЛОК'
words = [''.join(t) for n in range(1, 6) for t in product('01', repeat=n)]

def fano(codes):
    return all(not b.startswith(a) for a in codes for b in codes if a != b)

best = 10 ** 9
for combo in product(words, repeat=3):
    codes = dict(known)
    codes.update(zip(['К', 'Л', 'П'], combo))
    if len(set(codes.values())) == len(codes) and fano(codes.values()):
        best = min(best, sum(len(codes[c]) for c in word))
print(best)
```

Ответ: **21**.

Задание 5

На вход алгоритма подаётся натуральное число N. Алгоритм строит по нему новое число R следующим образом.

1. Строится двоичная запись числа N.

2. Далее эта запись обрабатывается по следующему правилу:

а) если число N чётное, то к этой записи справа и слева дописываются по две единицы;
б) если число N нечётное, то в конец двоичной записи (справа) дописываются два нуля, а в начало (слева) дописывается единица.

Полученная таким образом запись является двоичной записью искомого числа R.

3. Результат переводится в десятичную систему и выводится на экран.

Например, для исходного числа $10_{10} = 1010_2$ результатом является число $11101011_2 = 235_{10}$, а для исходного числа $28_{10} = 11100_2$ результатом является число $111110011_2 = 499_{10}$.

Укажите минимальное число R, большее 168, которое может быть получено с помощью описанного алгоритма. В ответе запишите это число в десятичной системе счисления.

Ответ 196

Результат работы алгоритма растёт вместе с N не строго монотонно (для разных N работают разные правила), поэтому надёжнее перебрать N программой. В записи R на несколько разрядов больше, чем в записи N, так что R больше N и перебора до 1000 с запасом достаточно.

Программа на Python

```
def R(n):
    b = bin(n)[2:]
    if n % 2 == 0:
        b = '11' + b + '11'
    else:
        b = '1' + b + '00'
    return int(b, 2)

print(min(R(n) for n in range(1, 1000) if R(n) > 168))
```

Ответ: **196**.

Задание 6

Исполнитель Черепаха действует на плоскости с декартовой системой координат. В начальный момент Черепаха находится в начале координат, её голова направлена вдоль положительного направления оси ординат, хвост опущен. При опущенном хвосте Черепаха оставляет на поле след в виде линии. В каждый конкретный момент известно положение исполнителя и направление его движения. У исполнителя существует 6 команд: Поднять хвост, означающая переход к перемещению без рисования; Опустить хвост, означающая переход в режим рисования; Вперёд n (где n – целое число), вызывающая передвижение Черепахи на n единиц в том направлении, куда указывает её голова; Назад n (где n – целое число), вызывающая передвижение в противоположном голове направлении; Направо m (где m – целое число), вызывающая изменение направления движения на m градусов по часовой стрелке; Налево m (где m – целое число), вызывающая изменение направления движения на m градусов против часовой стрелки.

Запись Повтори k [Команда1 Команда2 … КомандаS] означает, что последовательность из S команд повторится k раз.

Черепахе был дан для исполнения следующий алгоритм.

Повтори 6 [Вперёд 18 Направо 90 Вперёд 27 Направо 90]
Поднять хвост
Вперёд 6 Налево 90 Вперёд 12 Направо 90
Опустить хвост
Повтори 2 [Вперёд 28 Направо 90 Вперёд 18 Направо 90]

Определите, сколько точек с целочисленными координатами находятся внутри области пересечения фигур, ограниченных заданными алгоритмом линиями, включая точки на границах этого пересечения.

Ответ 91

Первый цикл рисует прямоугольник: Черепаха идёт вверх на 18, поворачивает направо и идёт на 27. Получается прямоугольник $0 \le x \le 27$, $0 \le y \le 18$.

Перемещение без рисования приводит Черепаху в точку (-12; 6), голова снова смотрит вверх.

Второй цикл: вверх на 28, затем вправо на 18. Второй прямоугольник: $-12 \le x \le 6$, $6 \le y \le 34$.

Пересечение: $0 \le x \le 6$, $6 \le y \le 18$. Целых точек: $(6 - (0) + 1) \cdot (18 - (6) + 1) = 7 \cdot 13 =$ **91**.

Проверка программой

```
A = (0, 27, 0, 18)      # x1, x2, y1, y2
B = (-12, 6, 6, 34)
inside = lambda r, x, y: r[0] <= x <= r[1] and r[2] <= y <= r[3]
print(sum(1 for x in range(-100, 101) for y in range(-100, 101) if inside(A, x, y) and inside(B, x, y)))
```

Задание 7

Голосовое сообщение оцифровывается в формате моно с частотой дискретизации 48 000 Гц и глубиной кодирования 24 бит. Определите наименьшее количество Кбайт, необходимое для сохранения сообщения в памяти (без учёта заголовка), если его длительность — 2 мин 7 с. В ответе укажите только число.

Ответ 17860

Длительность: 2 · 60 + 7 = 127 с. Объём: 1 · 48 000 · 3 байта · 127 = 18 288 000 байт.

В Кбайтах: 18 288 000 / 1024 ≈ 17859,38. Нужно целое число Кбайт, чтобы поместилось всё сообщение, — округляем вверх: **17860**.

Задание 8

Определите количество четырёхзначных чисел, записанных в семеричной системе счисления, в записи которых цифра 1 встречается ровно один раз, а никакие две нечётные цифры не стоят рядом.

Ответ 384

Переберём все строки из 4 цифр семеричной системы (0–6), отбросим начинающиеся с нуля и проверим условие. Всего строк $7^4 = 2401$ — перебор мгновенный.

Программа на Python

```
from itertools import product

c = 0
for t in product('0123456', repeat=4):
    if t[0] != '0' and t.count('1') == 1 and not any(int(a) % 2 == 1 and int(b) % 2 == 1 for a, b in zip(t,
        c += 1
print(c)
```

Ответ: **384**.

Задание 9

Задание выполняется с использованием прилагаемого файла.

Откройте файл электронной таблицы, содержащей в каждой строке шесть натуральных чисел. Определите количество строк таблицы, для которых выполнены оба условия:

– в строке только одно число повторяется ровно два раза, остальные четыре числа различны;
– повторяющееся число больше максимального из неповторяющихся.

В ответе запишите только число.

Файл к заданию: 9.xlsx (148 КБ)

Ответ 408

В электронной таблице. Справа от данных посчитайте для каждой ячейки, сколько раз число встречается в строке: в G1 `=СЧЁТЕСЛИ($A1:$F1;A1)`, протяните на G1:L1 и вниз. Первое условие: `=И(СЧЁТЕСЛИ(G1:L1;2)=2;СЧЁТЕСЛИ(G1:L1;1)=4)`. Повторяющееся число — `=СУММЕСЛИ(G1:L1;2;A1:F1)/2`, сумма неповторяющихся — `=СУММЕСЛИ(G1:L1;1;A1:F1)`, наибольшее из неповторяющихся — `=МАКСЕСЛИ(A1:F1;G1:L1;1)`. Итог по строке — 1, если выполнены оба условия, ответ — сумма этого столбца.

Программой. Сохраните лист как CSV с разделителем «;».

Программа на Python

```
c = 0
for line in open('9.csv'):
    a = [int(x) for x in line.split(';')]
    rep = [x for x in a if a.count(x) == 2]          # оба вхождения повторяющегося числа
    uniq = [x for x in a if a.count(x) == 1]
    if len(rep) == 2 and len(uniq) == 4:
        if rep[0] > max(uniq):
            c += 1
print(c)
```

Ответ: **408**.

Задание 10

Файл 10-1.odt содержит текст повести А. С. Пушкина «Дубровский». Определите, сколько раз в тексте глав I–VIII (от заголовка «ГЛАВА I» до заголовка «ТОМ ВТОРОЙ») встречается сочетание букв «ник» (только строчными буквами) в составе других слов. Вхождения, где «ник» образует отдельное слово, не учитываются. Слова, записанные через дефис (например, «что-то»), считаются состоящими из отдельных слов. В ответе запишите только число.

Текст для проверки программой: TXT.

Ответ 64

Откройте ODT в текстовом редакторе. Используйте поиск целых слов, учитывая регистр и указанный в условии фрагмент.

```
import re
L = 'A-Za-zA-Яа-яЁё0-9'           # символы, из которых состоят слова
def word(t, w, flags=0):          # вхождения w как отдельного слова
    return len(re.findall(rf'(?<![{L}]){w}(?![{L}])', t, flags))

t = open('10-1.txt', encoding='utf-8').read()
t = t[t.index('ГЛАВА I\n'):t.index('ТОМ ВТОРОЙ')]
print(t.count('ник') - word(t, 'ник'))
```

Задание 11

При регистрации изделий каждому из них присваивается серийный номер длиной 144 символа. Для записи номеров используется алфавит мощностью 510.

В базе данных для хранения каждого серийного номера отведено одинаковое и минимально возможное целое число байт. При этом используется посимвольное кодирование серийных номеров, все символы кодируются одинаковым и минимально возможным числом бит.

Определите объём памяти (в Кбайт), необходимый для хранения 12 800 серийных номеров. В ответе запишите только целое число — количество Кбайт.

Ответ 2025

$2^8 = 256 < 510 \le 512 = 2^9$, значит, на символ нужно 9 бит.

Номер: $144 \cdot 9 = 1296$ бит = 162 байт.

12 800 номеров: $162 \cdot 12\,800 = 2\,073\,600$ байт = 2 073 600 / 1024 = **2025** Кбайт.

Задание 12

Исполнитель МТ представляет собой читающую и записывающую головку, которая может передвигаться вдоль бесконечной горизонтальной ленты, разделённой на равные ячейки. В каждой ячейке находится ровно один символ из алфавита исполнителя, включая специальный пустой символ λ.

Время работы исполнителя делится на такты. На каждом такте головка находится в одном из допустимых состояний q0, q1, …; в начальный момент — в состоянии q0. За один такт головка может изменить символ в текущей ячейке, сдвинуться в соседнюю ячейку слева или справа и перейти в новое состояние (или остаться в прежнем).

Программа задаётся таблицей: в первой строке перечислены символы, в первом столбце — состояния. На пересечении стоит команда из трёх элементов через запятую: записываемый символ; L, R или S — сдвиг влево, сдвиг вправо или остановка после выполнения команды; новое состояние. Например, команда 0, L, q3: записать в текущую ячейку «0», сдвинуться влево и перейти в состояние q3. Пустая клетка означает, что такая пара «символ – состояние» невозможна.

Выполните задание. На ленте в соседних ячейках записано двоичное представление числа 1395 без ведущих нулей. Ячейки справа и слева от последовательности заполнены пустыми символами «λ». В начальный момент времени головка расположена над самой левой цифрой записи.

Программа работы исполнителя:

	λ	**0**	**1**
q0	λ, L, q1	1, R, q0	0, R, q0
q1	λ, R, q2	0, L, q1	1, L, q1
q2		λ, R, q2	1, S, q2

Определите результат выполнения программы. В ответе запишите получившееся на ленте двоичное число в десятичной системе счисления.

Ответ 652

$1395 = 10101110011_2$.

q0 идёт вправо и заменяет каждую цифру противоположной (0 ↔ 1). Дойдя до пустой ячейки, q1 возвращает головку к левому краю записи, а q2 стирает ведущие нули и останавливается на первой единице. Инверсия k-разрядного числа N даёт $(2^k - 1) - N$.

Проверим, проследив такты программой.

Программа на Python

```
prog = {
    'q0': {'λ': ('λ', 'L', 'q1'), '0': ('1', 'R', 'q0'), '1': ('0', 'R', 'q0')},
    'q1': {'λ': ('λ', 'R', 'q2'), '0': ('0', 'L', 'q1'), '1': ('1', 'L', 'q1')},
    'q2': {'0': ('λ', 'R', 'q2'), '1': ('1', 'S', 'q2')},
}
tape = dict(enumerate('10101110011'))
pos, state = 0, 'q0'
while True:
    w, move, state2 = prog[state][tape.get(pos, 'λ')]
    tape[pos] = w
    state = state2
    if move == 'S':
        break
    pos += 1 if move == 'R' else -1
res = ''.join(tape[i] for i in sorted(tape)).strip('λ')
print(res, int(res, 2))
```

На ленте остаётся 1010001100_2 = **652**.

Задание 13

В терминологии сетей TCP/IP маской сети называют двоичное число, которое показывает, какая часть IP-адреса узла сети относится к адресу сети, а какая — к адресу узла в этой сети. Адрес сети получается в результате применения поразрядной конъюнкции к заданному адресу узла и его маске.

Даны адрес узла 172.46.111.132 и маска 255.255.255.192. Найдите адрес сети.

В ответе укажите сумму числовых значений октетов найденного IP-адреса. Например, если бы найденный адрес был равен 100.20.3.4, то в ответе следовало бы записать: 127.

Ответ 457

Октеты маски, равные 255, переносят соответствующие октеты адреса без изменений, октеты, равные 0, дают 0. Считать нужно только октет, где маска равна 192.

$132 = 10000100_2$, $192 = 11000000_2$. Поразрядная конъюнкция: $10000000_2 = 128$.

Адрес сети: 172.46.111.128. Сумма октетов: 172 + 46 + 111 + 128 = **457**.

Проверка программой

```
a = [172, 46, 111, 132]
m = [255, 255, 255, 192]
net = [x & y for x, y in zip(a, m)]
print(net, sum(net))
```

Задание 14

Значение арифметического выражения

9^(133)+3^(228)−3^(2)−x,

где x — целое положительное число, не превышающее 2030, записали в троичной системе счисления. Определите наибольшее значение x, при котором в данной записи значения арифметического выражения содержится ровно 41 нуль.

В ответе запишите число в десятичной системе счисления.

Ответ 2028

9133=3266: в троичной записи это единица и 266 нулей. Слагаемое 3228 добавляет ещё одну единицу, а вычитание 32+x «занимает» единицу из старшего разряда: младшие разряды превращаются в цифры 2, а несколько последних зависят от x.

Точное количество нулей для каждого x удобнее посчитать программой — Python работает с длинными числами напрямую.

Программа на Python

```
def zeros(n, b):
    c = 0
    while n > 0:
        if n % b == 0:
            c += 1
        n //= b
    return c

for x in range(2030, 0, -1):
    if zeros(9 ** 133 + 3 ** 228 - 3 ** 2 - x, 3) == 41:
        print(x)
        break
```

Ответ: **2028**.

Задание 15

Обозначим через ДЕЛ(n, m) утверждение «натуральное число n делится без остатка на натуральное число m»; пусть на числовой прямой дан отрезок B = [281; 384].

Для какого наибольшего натурального числа A логическое выражение

(x ∈ B) → (ДЕЛ(x, A) ∨ ¬ДЕЛ(x, 6) ∨ ¬ДЕЛ(x, 27))

истинно (то есть принимает значение 1) при любом целом положительном значении переменной x?

Ответ 54

Выражение может стать ложным только при x ∈ [281; 384], делящемся и на 6, и на 27, то есть на НОК(6, 27) = 54. Для всех таких x должно выполняться ДЕЛ(x, A).

Кратные 54 на отрезке: 324, 378. A должно делить оба числа, наибольшее такое A — НОД = **54**.

Проверка программой

```
def F(x, A):
    return (not (281 <= x <= 384)) or x % A == 0 or x % 6 != 0 or x % 27 != 0

for A in range(3000, 0, -1):
    if all(F(x, A) for x in range(1, 4000)):
        print(A)
        break
```

Задание 16

Алгоритм вычисления значения функции F(n), где n — натуральное число, задан следующими соотношениями:

F(n) = n при n ≥ 2013;
F(n) = n + F(n + 3), если n < 2013.

Чему равно значение выражения F(2000) – F(2009)?

Ответ 6009

F(2000) = 2000 + F(2003) = 2000 + 2003 + F(2006) = … Раскрывая определение 3 раз, получаем F(2000) = 2000 + 2003 + 2006 + F(2009).

Поэтому F(2000) – F(2009) = 2000 + 2003 + 2006 = **6009**. Глубина рекурсии здесь небольшая, программа считает напрямую.

Проверка программой

```
def F(n):
    if n >= 2013:
        return n
    return n + F(n + 3)

print(F(2000) - F(2009))
```

Задание 17

Задание выполняется с использованием прилагаемого файла.

В файле содержится последовательность целых чисел. Её элементы могут принимать целые значения от –10 000 до 10 000 включительно. Определите количество троек последовательности, в которых ровно два элемента отрицательны, а сумма элементов тройки больше минимального положительного элемента последовательности, кратного 22.

В ответе запишите количество найденных троек, затем максимальную из сумм элементов таких троек. В данной задаче под тройкой подразумевается три идущих подряд элемента последовательности.

Файл к заданию: 17.txt (52 КБ)

Ответ 590 9277

Минимальный положительный элемент, кратный 22, равен 22. Ноль не является ни положительным, ни отрицательным: условия x < 0 и x > 0 его не учитывают. Начальное значение максимума берём заведомо меньше любой суммы.

Программа на Python

```
a = [int(x) for x in open('17.txt')]
m = min(x for x in a if x > 0 and x % 22 == 0)
c, best = 0, -10 ** 9
for i in range(len(a) - 2):
    t = a[i:i + 3]
    neg = sum(1 for x in t if x < 0)
    if neg == 2 and sum(t) > m:
        c += 1
        best = max(best, sum(t))
print(c, best)
```

Ответ: 590 9277.

Задание 18

Задание выполняется с использованием прилагаемого файла.

Квадрат разлинован на N × N клеток (1 < N < 30). Исполнитель Робот может перемещаться по клеткам, выполняя за одно перемещение одну из двух команд: вправо или вниз. По команде вправо Робот перемещается в соседнюю правую клетку, по команде вниз — в соседнюю нижнюю. Квадрат ограничен внешними стенами. Между соседними клетками квадрата также могут быть внутренние стены. Сквозь стену Робот пройти не может.

Перед каждым запуском Робота в каждой клетке квадрата лежит монета достоинством от 1 до 100. Посетив клетку, Робот забирает монету с собой; это также относится к начальной и конечной клеткам маршрута Робота.

В «угловых» клетках поля — тех, которые справа и снизу ограничены стенами, Робот не может продолжать движение, поэтому накопленная сумма считается итоговой. Таких конечных клеток на поле может быть несколько, включая правую нижнюю клетку поля. При разных запусках итоговые накопленные суммы могут различаться.

Определите максимальную и минимальную денежные суммы среди всех возможных итоговых сумм, которые может собрать Робот, пройдя из левой верхней клетки в конечную клетку маршрута. В ответе укажите два числа — сначала максимальную сумму, затем минимальную.

Исходные данные представляют собой электронную таблицу размером N × N (в файле N = 19), каждая ячейка которой соответствует клетке квадрата. Внутренние и внешние стены обозначены утолщёнными линиями.

Файл к заданию: 18.xlsx (7 КБ)

Ответ 2705 901

В электронной таблице. Рядом с полем (например, начиная со столбца W) постройте таблицу того же размера, где в каждой клетке — максимальная сумма, с которой Робот может в неё прийти. В левой верхней клетке — значение монеты, в первой строке `=X1+B1`, в первом столбце `=W1+A2`, в остальных клетках `=МАКС(W2;X1)+B2`. Затем исправьте формулы у клеток рядом со стенами: если слева стена, берите только клетку сверху, если сверху — только слева. Формулы удобнее всего протянуть по всему полю и поправить вручную клетки у стен — их немного. Так же постройте таблицу минимумов с функцией МИН.

Конечные клетки — те, у которых стены и справа, и снизу. В этом поле их 3: R10, M16, S19. Ответ — максимум из таблицы максимумов и минимум из таблицы минимумов по всем этим клеткам. Частая ошибка — взять только правую нижнюю клетку.

Ответ: 2705 901.

Проверка дома программой (нужна библиотека openpyxl, на экзамене её нет)

```
import openpyxl

ws = openpyxl.load_workbook('18.xlsx').active
n = ws.max_row
```

```
a = [[ws.cell(i + 1, j + 1).value for j in range(n)] for i in range(n)]

def thick(side):
    return side is not None and side.style in ('thick', 'medium')

def wall_right(i, j):
    return j == n - 1 or thick(ws.cell(i + 1, j + 1).border.right) or thick(ws.cell(i + 1, j + 2).border.le

def wall_down(i, j):
    return i == n - 1 or thick(ws.cell(i + 1, j + 1).border.bottom) or thick(ws.cell(i + 2, j + 1).border.t

mx = [[None] * n for _ in range(n)]
mn = [[None] * n for _ in range(n)]
mx[0][0] = mn[0][0] = a[0][0]
for i in range(n):
    for j in range(n):
        if i == j == 0:
            continue
        came = []
        if j > 0 and not wall_right(i, j - 1) and mx[i][j - 1] is not None:
            came.append((mx[i][j - 1], mn[i][j - 1]))
        if i > 0 and not wall_down(i - 1, j) and mx[i - 1][j] is not None:
            came.append((mx[i - 1][j], mn[i - 1][j]))
        if came:
            mx[i][j] = max(c[0] for c in came) + a[i][j]
            mn[i][j] = min(c[1] for c in came) + a[i][j]
ends = [(i, j) for i in range(n) for j in range(n)
        if wall_right(i, j) and wall_down(i, j) and mx[i][j] is not None]
print(max(mx[i][j] for i, j in ends), min(mn[i][j] for i, j in ends))
```

Задание 19

Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежат две кучи камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может:

– добавить в одну из куч (по своему выбору) четыре камня;
– увеличить количество камней в одной из куч (по своему выбору) в 2 раза.

Для того чтобы делать ходы, у каждого игрока есть неограниченное количество камней. Например, пусть в одной куче 10 камней, а в другой 5 камней; такую позицию в игре обозначим (10, 5). Тогда за один ход можно получить любую из четырёх позиций: (14, 5), (20, 5), (10, 9), (10, 10).

Игра завершается в тот момент, когда суммарное количество камней в двух кучах становится не менее 93. Победителем считается игрок, сделавший последний ход, то есть первым получивший такую позицию, при которой в кучах суммарно 93 камней или больше. В начальный момент в первой куче было 11 камней, во второй куче — S камней; $1 \le S \le 81$.

Будем говорить, что игрок имеет выигрышную стратегию, если он может выиграть при любых ходах противника.

Известно, что Ваня выиграл своим первым ходом после неудачного первого хода Пети. Укажите минимальное значение S, при котором такая ситуация возможна.

Ответ 21

Ване нужно, чтобы после хода Пети он смог одним ходом довести сумму до 93. Самый сильный ход — удвоить бóльшую кучу. Ищем наименьшее S, при котором у Пети есть ход, после которого (не выиграв сам) он оставляет такую позицию.

Перебор программой (функция game возвращает, кто и каким по счёту ходом выигрывает) даёт S = **21**.

Программа для заданий 19–21

```
from functools import lru_cache

def moves(h):
    a, b = h
    return [(a + 4, b), (a * 2, b), (a, b + 4), (a, b * 2)]

@lru_cache(None)
def game(h):
    # >0: игрок, который ходит, выигрывает этим по счёту своим ходом
    # <0: игрок, который ходит, проигрывает; соперник выигрывает этим по счёту ходом
    #  0: игра уже окончена (выиграл предыдущий игрок)
    if sum(h) >= 93:
        return 0
    res = [game(m) for m in moves(h)]
    lose = [r for r in res if r <= 0]
```

```
    if lose:
        return 1 + min(-r for r in lose)
    return -max(res)

# 19: Петя сделал ход (не выигрышный), после которого Ваня выигрывает первым ходом
print(min(s for s in range(1, 82)
          if any(sum(m) < 93 and game(m) == 1 for m in moves((11, s)))))
```

Задание 20

Для игры, описанной в задании 19, найдите два наименьших значения S, при которых у Пети есть выигрышная стратегия, причём одновременно выполняются два условия:

– Петя не может выиграть за один ход;
– Петя может выиграть своим вторым ходом независимо от того, как будет ходить Ваня.

Найденные значения запишите в ответе в порядке возрастания.

Ответ 20 34

Нужны S, для которых game возвращает 2: Петя выигрывает своим вторым ходом и не может выиграть первым.

Все такие S: 20, 34, 35, 36, 37, 38. Два наименьших: 20 и 34.

Программа на Python

```
from functools import lru_cache

def moves(h):
    a, b = h
    return [(a + 4, b), (a * 2, b), (a, b + 4), (a, b * 2)]

@lru_cache(None)
def game(h):
    # >0: игрок, который ходит, выигрывает этим по счёту своим ходом
    # <0: игрок, который ходит, проигрывает; соперник выигрывает этим по счёту ходом
    #  0: игра уже окончена (выиграл предыдущий игрок)
    if sum(h) >= 93:
        return 0
    res = [game(m) for m in moves(h)]
    lose = [r for r in res if r <= 0]
    if lose:
        return 1 + min(-r for r in lose)
    return -max(res)

print(*[s for s in range(1, 82) if game((11, s)) == 2][:2])
```

Задание 21

Для игры, описанной в задании 19, найдите наименьшее значение S, при котором одновременно выполняются два условия:

– у Вани есть выигрышная стратегия, позволяющая ему выиграть первым или вторым ходом при любой игре Пети;
– у Вани нет стратегии, которая позволит ему гарантированно выиграть первым ходом.

Ответ 30

Нужны S, для которых game возвращает –2: Петя проигрывает, а Ваня при лучшей игре Пети выигрывает своим вторым ходом. Наименьшее такое S — **30**.

Программа на Python

```
from functools import lru_cache

def moves(h):
    a, b = h
    return [(a + 4, b), (a * 2, b), (a, b + 4), (a, b * 2)]

@lru_cache(None)
def game(h):
    # >0: игрок, который ходит, выигрывает этим по счёту своим ходом
    # <0: игрок, который ходит, проигрывает; соперник выигрывает этим по счёту ходом
    #  0: игра уже окончена (выиграл предыдущий игрок)
    if sum(h) >= 93:
        return 0
    res = [game(m) for m in moves(h)]
    lose = [r for r in res if r <= 0]
```

```
    if lose:
        return 1 + min(-r for r in lose)
    return -max(res)

print(min(s for s in range(1, 82) if game((11, s)) == -2))
```

Задание 22

Задание выполняется с использованием прилагаемого файла.

В файле содержится информация о совокупности N вычислительных процессов, которые могут выполняться параллельно или последовательно. Приостановка выполнения процесса не допускается. Будем говорить, что процесс B зависит от процесса A, если для выполнения процесса B необходимы результаты выполнения процесса A. В этом случае процессы A и B могут выполняться только последовательно.

Информация о процессах представлена в файле в виде таблицы. В первом столбце таблицы указан идентификатор процесса (ID), во втором столбце таблицы — время его выполнения в миллисекундах, в третьем столбце перечислены с разделителем «;» ID процессов, от которых зависит данный процесс. Если процесс независимый, то в таблице указано значение 0.

Определите минимальное время (в мс), через которое завершится выполнение всей совокупности процессов, при условии, что все независимые друг от друга процессы могут выполняться параллельно.

Типовой пример организации данных в файле:

ID процесса B	Время выполнения процесса B (мс)	ID процесса(-ов) A
1	3	0
2	4	1
3	2	2;4
4	5	0
5	8	1;4
6	3	1

В этом примере процессы завершатся через 13 мс: процесс 4 закончится к 5-й мс, процесс 5 начнётся после него и займёт ещё 8 мс.

Типовой пример имеет иллюстративный характер. Для выполнения задания используйте данные из прилагаемого файла.

Файл к заданию: 22.xlsx (6 КБ)

Ответ 39

Каждый процесс начинается, как только закончатся все процессы, от которых он зависит: окончание = max(окончаний зависимостей) + время процесса. Ответ — наибольшее время окончания.

В электронной таблице в столбце D считают время окончания: для независимых `=B2`, для зависимых — `=МАКС(D…;D…)+B…` со ссылками на строки процессов из столбца C.

Программой. Сохраните лист как CSV с разделителем «;».

Программа на Python

```
import csv

rows = list(csv.reader(open('22.csv', encoding='utf-8'), delimiter=';'))[1:]
time, deps = {}, {}
for pid, t, d in rows:
    time[int(pid)] = int(t)
    deps[int(pid)] = [] if d == '0' else [int(x) for x in d.split(';')]

finish = {}
def fin(p):                     # момент окончания процесса p
    if p not in finish:
        finish[p] = max([fin(q) for q in deps[p]], default=0) + time[p]
    return finish[p]

print(max(fin(p) for p in time))
```

Ответ: **39**.

Задание 23

Исполнитель преобразует число на экране. У исполнителя есть три команды, которые обозначены латинскими буквами:

A. Прибавить 1
B. Поменять местами
C. Прибавить 11

Первая команда увеличивает число на экране на 1, третья — на 11. Вторая команда применяется только к числу, у которого цифра в разряде десятков по значению меньше цифры, стоящей в разряде единиц, и заменяет число на экране числом, в котором цифры двух младших разрядов поменялись местами (например, 125 → 152).

Программа для исполнителя — это последовательность команд. Сколько существует программ, для которых при исходном числе 12 результатом является число 83?

Траектория вычислений программы — это последовательность результатов выполнения всех команд программы. Например, для программы ABC при исходном числе 13 траектория состоит из чисел 14, 41, 52.

Ответ 132357

Перестановка двух младших цифр при $t < u$ увеличивает число на $9\cdot(u - t)$, поэтому все команды увеличивают число и рекурсия конечна: как только число превысило цель, путь обрывается.

Вторая команда меняет число так: $x - 10\cdot t - u + 10\cdot u + t$, где t и u — цифры десятков и единиц.

Программа на Python

```
def f(x, y):
    if x > y:
        return 0
    if x == y:
        return 1
    res = f(x + 1, y) + f(x + 11, y)
    t, u = x // 10 % 10, x % 10        # цифры десятков и единиц
    if t < u:
        res += f(x - 10 * t - u + 10 * u + t, y)
    return res

print(f(12, 83))
```

Ответ: **132357**.

Задание 24

Задание выполняется с использованием прилагаемого файла.

Текстовый файл состоит из цифр 0, 2, 4, 5, 7 и знаков арифметических операций «+» и «-» (сложение и вычитание). Определите максимальное количество символов в непрерывной последовательности, которая является корректным арифметическим выражением с целыми неотрицательными числами. В этом выражении никакие два знака арифметических операций не стоят рядом, в записи чисел отсутствуют незначащие (ведущие) нули и число 0 не имеет знака.

В ответе укажите количество символов.

Файл к заданию: 24.txt (244 КБ)

Ответ 125

Идём по строке и поддерживаем длину cur самого длинного корректного выражения, которое заканчивается на текущем символе. Знак продолжает выражение, если перед ним стоит цифра; два знака подряд обрывают его. Цифра после знака начинает новое число. Цифра после цифры продолжает число, если оно не начинается с нуля; иначе выражение приходится начинать заново с текущей цифры.

Ответ обновляем только на цифрах: корректное выражение не может заканчиваться знаком.

Программа на Python

```
s = open('24.txt').read().strip()
best = cur = 0       # cur — длина самого длинного корректного выражения, оканчивающегося здесь
zero = False         # последнее число в нём начинается с нуля
for i, c in enumerate(s):
    if c in '+-':
        if i > 0 and s[i - 1] not in '+-' and cur > 0:
            cur += 1                    # знак после цифры продолжает выражение
        else:
            cur = 0                     # два знака подряд
```

```
    else:
        if i > 0 and s[i - 1] in '+-' and cur > 0:
            cur += 1                    # новое число после знака
            zero = c == '0'
        elif i > 0 and s[i - 1] not in '+-' and not zero:
            cur += 1                    # продолжаем текущее число
        else:
            cur = 1                     # начинаем выражение заново с этой цифры
            zero = c == '0'
        best = max(best, cur)
print(best)
```

Ответ: **125**.

Задание 25

Пусть M — сумма минимального и максимального натуральных делителей целого числа, не считая единицы и самого числа. Если таких делителей у числа нет, то считается, что значение M равно нулю.

Напишите программу, которая перебирает целые числа, бóльшие 1 295 668, в порядке возрастания и ищет среди них такие, для которых M оканчивается на 2. В ответе запишите в первом столбце таблицы первые пять найденных чисел в порядке возрастания, а во втором столбце — соответствующие им значения M.

Например, для числа 20 M = 2 + 10 = 12.

Количество строк в таблице для ответа избыточно.

Ответ 1295677 17822
1295680 647842
1295687 8112
1295697 431902
1295700 647852

Наименьший делитель числа (кроме 1) — первый делитель d при переборе от 2, а наибольший (кроме самого числа) — n // d. Перебор достаточно вести до √n: если делителя до корня нет, число простое и M = 0.

Программа на Python

```
def M(n):
    for d in range(2, int(n ** 0.5) + 1):
        if n % d == 0:
            return d + n // d     # наименьший делитель d, наибольший — n // d
    return 0                      # простое число

found = 0
n = 1295669
while found < 5:
    m = M(n)
    if m % 10 == 2 and m != 0:
        print(n, m)
        found += 1
    n += 1
```

Ответ:
1295677 17822
1295680 647842
1295687 8112
1295697 431902
1295700 647852

Задание 26

Задание выполняется с использованием прилагаемого файла.

Сервер выполняет запросы на передачу данных, при этом сведения о каждом выполненном запросе (время регистрации, идентификатор клиента, объём переданных данных) сохраняются в журнале работы, а переданные данные — в специальном разделе памяти сервера, имеющем ограниченный объём. Каждый раз, когда остаётся недостаточно свободной памяти, сервер создаёт резервную копию всех накопленных там данных, после чего освобождает раздел и продолжает выполнение запросов.

Напишите программу обработки журнала работы сервера и определите, сколько всего резервных копий было создано за время, отражённое в журнале, а также объём (в Кбайт) наибольшей резервной копии, созданной не ранее 12:00:00.

Входные данные. Первая строка входного файла (журнала работы сервера) содержит два натуральных числа: N (N < 1 000 000) — количество строк в журнале и K (K < 1 000 000) — вместимость специального раздела памяти сервера в Кбайт. Каждая из следующих N строк содержит информацию об одном выполненном запросе: время регистрации запроса в формате ЧЧ:ММ:СС и два натуральных числа: C (C < 1 000 000) — идентификатор клиентского устройства и S (S < K) — объём данных запроса в Кбайт. Записи в журнале идут в порядке возрастания времени.

Выходные данные. В ответе запишите два числа: сначала количество созданных резервных копий, затем объём наибольшей резервной копии, созданной не ранее 12:00:00.

Типовой пример организации данных во входном файле:

```
8 140000
01:01:01 101 20000
03:03:03 202 110000
05:05:05 101 90000
07:07:07 303 62000
10:10:10 101 48000
15:15:15 202 12000
21:21:21 303 120000
23:23:23 404 134000
```

При таких исходных данных резервное копирование выполняется четыре раза: в 05:05:05 (130 000 Кбайт), в 07:07:07 (90 000 Кбайт), в 21:21:21 (122 000 Кбайт) и в 23:23:23 (120 000 Кбайт). Всего передано 596 000 Кбайт: 158 000, 122 000, 182 000 и 134 000 Кбайт от клиентов 101, 202, 303 и 404 соответственно. Ответ для приведённого примера: 4 122000.

Типовой пример имеет иллюстративный характер. Для выполнения задания используйте данные из прилагаемого файла.

Файл к заданию: 26.txt (93 КБ)

Ответ 705 199886

Моделируем работу сервера: храним, сколько занято в разделе. Если очередной запрос не помещается (занято + S > K), создаётся резервная копия объёмом «занято», раздел освобождается, и только потом записываются данные запроса. Время копии — время запроса, который не поместился.

Время в формате ЧЧ:ММ:СС с ведущими нулями можно сравнивать как строки.

Программа на Python

```
f = open('26.txt')
n, k = map(int, f.readline().split())
used = 0
count = 0
best = 0
for _ in range(n):
    t, c, s = f.readline().split()
    s = int(s)
    if used + s > k:                # места не хватает: резервная копия
        count += 1
        if t >= '12:00:00':         # строки ЧЧ:ММ:СС можно сравнивать как текст
            best = max(best, used)
        used = 0
    used += s
print(count, best)
```

Ответ: 705 199886.

Задание 27

Задание выполняется с использованием прилагаемого файла.

Рассматриваются частицы на плоскости, обладающие следующими характеристиками: декартовы координаты, вектор скорости, масса, а также признак, характеризующий внутреннее строение частицы, обозначаемый числами от I до VII (в римской системе счисления).

Учёный решил провести кластеризацию частиц по модулю импульса, то есть разбить их множество на K непересекающихся непустых подмножеств (кластеров), таких, что модуль разности модулей импульса любых двух частиц каждого подмножества не превосходит значения R. Гарантируется, что такое разбиение существует и единственно для заданного R.

Будем называть центром кластера такую его частицу, для которой сумма модулей разности модулей импульса со всеми остальными частицами этого кластера минимальна. Для каждого кластера гарантируется единственность его центра.

В каждой строке текстового файла хранится информация об одной частице: координаты x и y, проекции вектора скорости V_x и V_y, масса m и признак. Значения в строке разделяются одним или несколькими пробелами и/или символами табуляции, дробная часть отделяется точкой. Количество строк в файле не превышает 10 000. Абсолютная величина каждого числового значения не превышает 100,0.

Известно, что все описанные в файле частицы подразделяются ровно на три кластера (K = 3) с R = 2,0 для каждого.

Найдите два числа: Q1 — наибольшее евклидово расстояние между частицами одного кластера, имеющими признак V, и Q2 — среднее арифметическое значений модуля импульса частиц кластера, содержащего наименьшее количество частиц.

В ответе запишите два числа: сначала целую часть произведения Q1 × 10 000, затем целую часть произведения Q2 × 10 000.

Для справки. Модуль импульса p частицы массы m, обладающей скоростью $v = (V_x, V_y)$, вычисляется по формуле: $p = m\sqrt{(V_x^2 + V_y^2)}$.

Евклидово расстояние между двумя точками на плоскости $A(x_1, y_1)$ и $B(x_2, y_2)$ вычисляется по формуле: $d(A,B) = \sqrt{((x_2 - x_1)^2 + (y_2 - y_1)^2)}$.

Типовой пример организации данных во входном файле (три строки файла для трёх частиц):

```
0.67  -2.14  3.0  -4.0  0.2  V
3.14  7.22  3.2  4.3  0.7  II
1.33  5.56  0.00  5.22  0.456  IV
```

Для частицы из первой строки примера модуль импульса равен 1,0.

Типовой пример имеет иллюстративный характер. Для выполнения задания используйте данные из прилагаемого файла.

Файл к заданию: 27.txt (53 КБ)

Ответ 1538111 31146

Посчитаем для каждой частицы модуль импульса и отсортируем частицы по этой величине. Разбиение единственно: внутри кластера соседние значения отличаются не больше чем на R, а между кластерами разрыв больше R. Поэтому кластеры получаются разрезанием отсортированного списка там, где разность соседних значений больше 2,0.

Центр кластера ищем по определению — перебором сумм модулей разностей (это медиана кластера). Затем считаем Q1 и Q2 по условию.

Программа на Python

```
from math import dist, sqrt

parts = []
for line in open('27.txt'):
    x, y, vx, vy, m, f = line.split()
    x, y, vx, vy, m = map(float, (x, y, vx, vy, m))
    parts.append((m * sqrt(vx ** 2 + vy ** 2), x, y, f))     # величина, координаты, признак

parts.sort()                                # по величине
clusters, cur = [], [parts[0]]
for p in parts[1:]:
    if p[0] - cur[-1][0] > 2.0:             # разрыв больше R — новый кластер
        clusters.append(cur)
        cur = []
    cur.append(p)
clusters.append(cur)
print(len(clusters))                        # должно быть 3

# центр: минимальная сумма модулей разностей
centers = [min(c, key=lambda p: sum(abs(p[0] - r[0]) for r in c)) for c in clusters]
q1 = 0
for c in clusters:
    f = [p for p in c if p[3] == 'V']
    for i in range(len(f)):
        for j in range(i + 1, len(f)):
            q1 = max(q1, dist(f[i][1:3], f[j][1:3]))
small = min(clusters, key=len)
q2 = sum(p[0] for p in small) / len(small)
print(int(q1 * 10000), int(q2 * 10000))
```

Ответ: 1538111 31146.