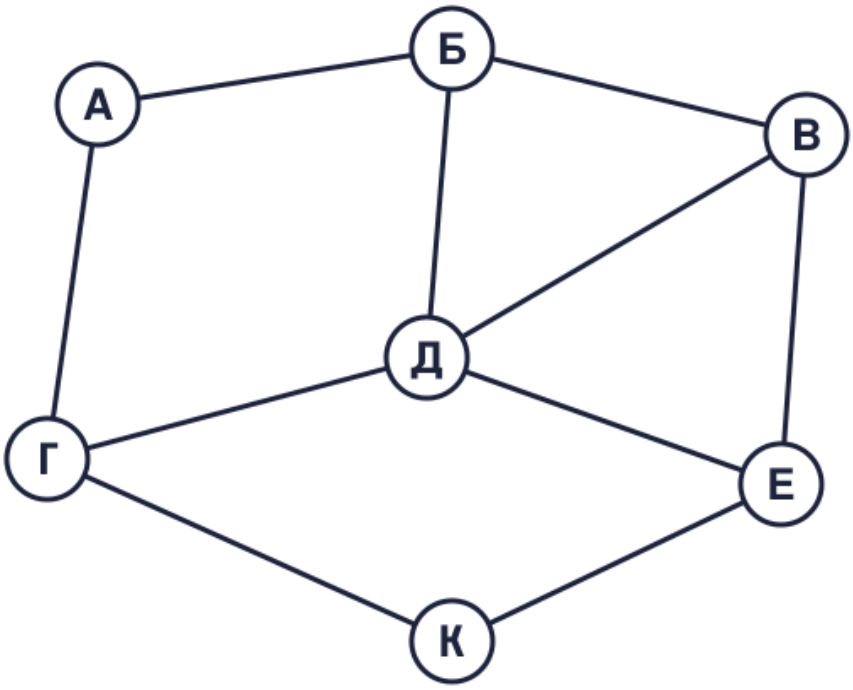
ЕГЭНИЙ · Информатика

Вариант 1 · 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		35		31	18		
2	35			12	27	46	
3					47	19	
4	31	12					14
5	18	27	47				
6		46	19				38
7				14		38	


Б
А
В
Д
Г
Е
К

Так как таблицу и схему рисовали независимо друг от друга, нумерация населённых пунктов в таблице никак не связана с буквенными обозначениями на графе. Определите, какова сумма протяжённостей дорог из пункта Г в пункт Д и из пункта В в пункт Д.

В ответе запишите целое число.

Ответ 81

Посчитаем степени вершин (количество дорог из пункта). На графе: А — 2, Б — 3, В — 3, Г — 3, Д — 4, Е — 3, К — 2. В таблице: П1 — 3, П2 — 4, П3 — 2, П4 — 3, П5 — 3, П6 — 3, П7 — 2.

Сопоставляем вершины по степеням и по соседям: В — вершина степени 3, степени её соседей: 3, 3, 4; так устроен только П1; Г — вершина степени 3, не соединённая с В; этому соответствует только П6; степень 4 на графе только у вершины Д, в таблице — только у пункта 2: Д = П2.

Дорога ГД — это П6–П2, её длина 46 км. Дорога ВД — это П1–П2, её длина 35 км.

Сумма: 46 + 35 = **81**.

Задание 2

Миша заполнял таблицу истинности логической функции F

(x ∧ ¬y) ∨ (y ≡ z) ∨ 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
	1			0
	0	1		0
1		1	0	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 xzyw

Переберём программой все способы заполнить пустые клетки и все перестановки переменных по столбцам. Подходят только заполнения, при которых три строки различны, и перестановки, при которых 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 y) or (y == z) or w

answers = set()
for a1, a2, a3, a4, a5, a6 in product([0, 1], repeat=6):
    t = [(a1, 1, a2, a3), (a4, 0, 1, a5), (1, a6, 1, 0)]
    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: **xzyw**.

Задание 3

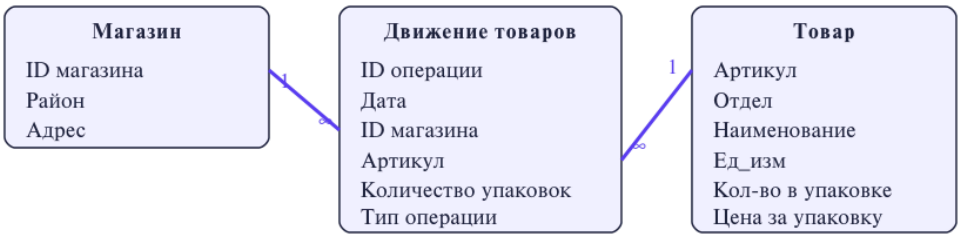
Задание выполняется с использованием прилагаемого файла.

В файле приведён фрагмент базы данных «Продукты» 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 общую массу (в килограммах) всех видов сыра, проданных магазинами Заречного района за период с 4 по 13 июня включительно. В ответе запишите только число; если оно дробное, запишите его десятичной дробью без округления.

Файл к заданию: 3.xlsx (113 КБ)

Ответ 356,135

В электронной таблице. На листе «Движение товаров» добавьте столбцы: район магазина `=ВПР(C2;$Магазин.$A$2:$B$21;2;0)`, наименование `=ВПР(D2;$Товар.$A$2:$F$100;3;0)`, единицу измерения (столбец 4 листа «Товар») и количество в упаковке (столбец 5).

Включите автофильтр и оставьте: тип операции — Продажа, даты с 04.06.2026 по 13.06.2026, район — Заречный, наименование начинается со слова «Сыр».

Обратите внимание на единицы измерения: часть сыров учтена в граммах, часть — в килограммах. Масса строки в килограммах = количество упаковок · количество в упаковке, для граммов — ещё делённое на 1000. Сумму отфильтрованных 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 typ == 'Продажа' and 4 <= day <= 13 and shops[shop] == 'Заречный' and g[2].startswith('Сыр '):
        q = float(g[4].replace(',', '.'))
        if g[3] == 'г':
            q = q / 1000            # граммы -> килограммы
        total += int(cnt) * q
print(round(total, 3))
```

Ответ: **356,135**.

Задание 4

По каналу связи передаются сообщения, содержащие только буквы из набора: А, Д, Н, О, Р, Т. Для передачи используется двоичный код, удовлетворяющий условию Фано. Кодовые слова для некоторых букв известны: А – 01, Д – 100, Т – 111. Для трёх оставшихся букв Н, О и Р кодовые слова неизвестны.

Какое наименьшее количество двоичных знаков потребуется для кодирования слова РОТОНДА, 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 18

Свободные ветви двоичного дерева кодов: занято 01 (А), 100 (Д), 111 (Т). Свободны вершины 00, 101 и 110 — из них и можно брать коды, не нарушая условие Фано. Букве О, которая встречается в слове дважды, выгодно дать самый короткий код 00. Оставшимся Р и Н — 101 и 110.

В слове РОТОНДА: Р (3) + О (2) + Т (3) + О (2) + Н (3) + Д (3) + А (2) = **18**.

Проверим перебором: программа перебирает коды длиной до 4 для трёх неизвестных букв и оставляет только наборы, удовлетворяющие условию Фано.

Программа на Python

```
from itertools import product

known = {'А': '01', 'Д': '100', 'Т': '111'}
word = 'РОТОНДА'
words = [''.join(t) for n in range(1, 5) 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 r, o, n in product(words, repeat=3):
    codes = dict(known, Р=r, О=o, Н=n)
    if len(set(codes.values())) == 6 and fano(codes.values()):
        best = min(best, sum(len(codes[c]) for c in word))
print(best)
```

Ответ: **18**.

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 дописываются две её последние цифры;
б) если число N нечётное, то к этой записи слева дописывается 1, а справа — цифры 01.

Полученная таким образом запись является двоичной записью искомого числа R.

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

Например, для исходного числа $6_{10} = 110_2$ результатом является число $11010_2 = 26_{10}$, а для исходного числа $5_{10} = 101_2$ это число $110101_2 = 53_{10}$.

Укажите наибольшее число N, для которого результат работы данного алгоритма меньше 300. В ответе запишите это число в десятичной системе счисления.

Ответ 74

Результат растёт вместе с N не строго монотонно (у чётных и нечётных N разные правила), поэтому надёжнее перебрать N программой и взять наибольшее подходящее.

Перебор до 1000 заведомо достаточен: при N ≥ 128 в записи R не меньше 9 разрядов, то есть R ≥ 256, а при N ≥ 512 — R ≥ 2048.

Программа на Python

```
def R(n):
    b = bin(n)[2:]
    if n % 2 == 0:
        b = b + b[-2:]
    else:
        b = '1' + b + '01'
    return int(b, 2)

print(max(n for n in range(1, 1000) if R(n) < 300))
```

Ответ: **74**.

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 4 [Вперёд 18 Направо 90 Вперёд 26 Направо 90]
Поднять хвост
Назад 5 Налево 90 Вперёд 9 Направо 90
Опустить хвост
Повтори 4 [Вперёд 31 Направо 90 Вперёд 22 Направо 90]

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

Ответ 266

Первый цикл рисует прямоугольник: Черепаха идёт вверх на 18, поворачивает направо и идёт на 26. Получается прямоугольник с вершинами (0; 0) и (26; 18): 0 ≤ x ≤ 26, 0 ≤ y ≤ 18. Повторов 4, поэтому он

обводится дважды.

Перемещение без рисования: Назад 5 — точка (0; –5), Налево 90 — голова смотрит влево, Вперёд 9 — точка (–9; –5), Направо 90 — голова снова вверх.

Второй прямоугольник: -9 ≤ x ≤ 13, -5 ≤ y ≤ 26.

Пересечение: 0 ≤ x ≤ 13, 0 ≤ y ≤ 18. Целых точек: (13 – 0 + 1) · (18 – 0 + 1) = 14 · 19 = **266**.

Проверка программой

```
# прямоугольники из условия: x1 <= x <= x2, y1 <= y <= y2
A = (0, 26, 0, 18)
B = (-9, 13, -5, 26)
inside = lambda r, x, y: r[0] <= x <= r[1] and r[2] <= y <= r[3]
print(sum(1 for x in range(-50, 51) for y in range(-50, 51) if inside(A, x, y) and inside(B, x, y)))
```

Задание 7

Музыкальный фрагмент был записан в формате квадро (четырёхканальная запись), оцифрован с частотой дискретизации 24 000 Гц и разрешением 24 бита и сохранён без сжатия. Размер полученного файла без учёта заголовка не превышает 30 Мбайт.

Определите максимально возможную длительность фрагмента в секундах. В ответе запишите целое число.

Ответ 109

Объём одной секунды записи: 4 канала · 24 000 · 24 бита = 4 · 24 000 · 3 байта = 288 000 байт.

30 Мбайт = 30 · 2^{20} = 31 457 280 байт.

31 457 280 / 288 000 ≈ 109,2. Длительность должна помещаться целиком, поэтому округляем вниз: **109**.

Задание 8

Все пятибуквенные слова, составленные из букв Д, И, С, К, О, Р, записаны в алфавитном порядке и пронумерованы. Вот начало списка:

1. ДДДДД
2. ДДДДИ
3. ДДДДК
4. ДДДДО
5. ДДДДР
6. ДДДДС

……

Определите, под каким номером в этом списке стоит первое слово с нечётным номером, которое не начинается с букв Д или И, содержит не менее двух букв О и не содержит двух букв Р подряд.

Примечание. Слово — последовательность идущих подряд букв, не обязательно осмысленная.

Ответ 2719

Важно взять буквы в алфавитном порядке: Д, И, К, О, Р, С — тогда product перебирает слова в том же порядке, что и в списке, а нумерация начинается с 1.

Программа на Python

```
from itertools import product

letters = sorted('ДИСКОР')        # Д, И, К, О, Р, С
for i, w in enumerate(product(letters, repeat=5), 1):
    w = ''.join(w)
    if i % 2 == 1 and w[0] not in 'ДИ' and w.count('О') >= 2 and 'РР' not in w:
        print(i, w)
        break
```

Ответ: **2719**.

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;

– среднее арифметическое неповторяющихся чисел строки не больше повторяющегося числа.

В ответе запишите только число.

Файл к заданию: 9.xlsx (165 КБ)

Ответ 1152

В электронной таблице. Справа от данных для каждой ячейки строки посчитайте, сколько раз число встречается в строке: в G1 `=СЧЁТЕСЛИ($A1:$F1;A1)`, протяните на G1:L1 и вниз. Строка подходит под первое условие, если двоек ровно две, а единиц — четыре:
`=И(СЧЁТЕСЛИ(G1:L1;2)=2;СЧЁТЕСЛИ(G1:L1;1)=4)`.

Повторяющееся число: `=СУММЕСЛИ(G1:L1;2;A1:F1)/2`, среднее неповторяющихся: `=СУММЕСЛИ(G1:L1;1;A1:F1)/4`. Итог по строке — `=ЕСЛИ(И(…первое условие…; среднее<=повторяющееся);1;0)`, ответ — сумма 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 sum(uniq) / 4 <= rep[0]:
            c += 1
print(c)
```

Ответ: **1152**.

Задание 10

Файл 10-1.odt содержит текст повести А. С. Пушкина «Дубровский». С помощью текстового редактора определите, сколько раз во всём тексте файла встречается слово «Дубровский» (именно в этой форме и с прописной буквы) как отдельное слово. Другие формы («Дубровского», «Дубровскому») 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 100

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()
print(word(t, 'Дубровский'))
```

Задание 11

При регистрации изделий каждому из них присваивается серийный номер длиной 280 символов. Для записи номеров используется алфавит из 3000 различных символов (буквы нескольких алфавитов, цифры и специальные знаки).

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 64 000 серийных номеров. В ответе запишите только целое число — количество Кбайт.

Ответ 26250

$2^{11} = 2048 < 3000 \le 4096 = 2^{12}$, значит, на символ нужно 12 бит.

Номер: 280 · 12 = 3360 бит = 420 байт (целое число, округлять не нужно).

64 000 номеров: 420 · 64 000 = 26 880 000 байт = 26 880 000 / 1024 = **26250** 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 1999 без ведущих нулей. Ячейки справа и слева от последовательности заполнены пустыми символами «λ». В начальный момент времени головка расположена в ближайшей справа к последовательности ячейке.

Программа работы исполнителя:

	λ	0	1
q0	λ, L, q1		
q1		1, L, q2	0, L, q3
q2	1, S, q2	1, S, q2	0, L, q2
q3	0, L, q2	0, L, q2	1, L, q2

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

Ответ 2002

1999 = 11111001111_2.

q0 сдвигает головку на последнюю цифру. q1 прибавляет 1 к младшему разряду: 0 → 1 и переход в q2, 1 → 0 и переход в q3 (перенос).

q2 прибавляет 1 к текущему разряду: на 0 или λ пишет 1 и останавливается, на 1 пишет 0 и идёт влево с переносом. q3 оставляет текущий разряд без изменений и переходит в q2 на следующем разряде — то есть добавляет единицу в разряд двоек.

Итого программа прибавляет к числу 1 + 2 = 3: 1999 + 3 = 2002. Проверим, проследив такты программой.

Программа на Python

```
prog = {
    'q0': {'λ': ('λ', 'L', 'q1')},
    'q1': {'0': ('1', 'L', 'q2'), '1': ('0', 'L', 'q3')},
    'q2': {'λ': ('1', 'S', 'q2'), '0': ('1', 'S', 'q2'), '1': ('0', 'L', 'q2')},
    'q3': {'λ': ('0', 'L', 'q2'), '0': ('0', 'L', 'q2'), '1': ('1', 'L', 'q2')},
}
tape = dict(enumerate('11111001111'))
pos, state = 11, '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 11111010010_2 = **2002**.

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 172.16.203.141 и маска 255.255.240.0. 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 380

Октеты маски, равные 255, переносят соответствующие октеты адреса без изменений, октеты, равные 0, дают 0. Считать нужно только октет, где маска равна 240.

$203 = 11001011_2$, $240 = 11110000_2$. Поразрядная конъюнкция: $11000000_2 = 192$.

Адрес сети: 172.16.192.0. Сумма октетов: 172 + 16 + 192 + 0 = **380**.

Проверка программой

```
a = [172, 16, 203, 141]
m = [255, 255, 240, 0]
net = [x & y for x, y in zip(a, m)]
print(net, sum(net))
```

Задание 14

Значение арифметического выражения

25^(160)+5^(210)−125−x,

где x — целое положительное число, не превышающее 2500, записали в пятеричной системе счисления. Определите наибольшее значение x, при котором в данной записи значения арифметического выражения содержится ровно 112 нулей.

В ответе запишите число в десятичной системе счисления.

Ответ 2494

25160=5320, поэтому 25160+5210 в пятеричной записи — единица, затем нули, в разряде 210 ещё одна единица и снова нули (321 цифра).

Вычитание 125 + x «занимает» единицу из разряда 210: он становится нулём, разряды ниже превращаются в четвёрки, а несколько младших разрядов зависят от x. Поэтому нулей около 110, а точное количество для каждого x проще 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(2500, 0, -1):
    if zeros(25 ** 160 + 5 ** 210 - 125 - x, 5) == 112:
        print(x)
        break
```

Ответ: **2494**.

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 = [200; 300].

Для какого наибольшего натурального числа A логическое выражение

(x ∈ B) → (ДЕЛ(x, A) ∨ ¬ДЕЛ(x, 28) ∨ ¬ДЕЛ(x, 42))

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

Ответ 252

Выражение может стать ложным только при x ∈ [200; 300], делящемся и на 28, и на 42, то есть на НОК(28, 42) = 84. Для таких x должно выполняться ДЕЛ(x, A).

Кратные 84 на отрезке [200; 300]: только 252 (168 < 200, 336 > 300). Значит, A должно быть делителем 252, наибольшее A = **252**.

Частая ошибка — ответ 84: он верен для всех кратных 84, а здесь отрезок оставляет единственное такое число.

Проверка программой

```
def F(x, A):
    return (not (200 <= x <= 300)) or x % A == 0 or x % 28 != 0 or x % 42 != 0

for A in range(1000, 0, -1):
    if all(F(x, A) for x in range(1, 1000)):
        print(A)
        break
```

Задание 16

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

F(n) = 1 при n = 1;
F(n) = (n + 1) × F(n – 1), если n > 1.

Чему равно значение выражения (F(2025) – 3 × F(2024)) / F(2023)?

Ответ 4096575

Выразим F(2025) и F(2024) через F(2023): F(2024) = 2025 · F(2023), F(2025) = 2026 · F(2024) = 2026 · 2025 · F(2023).

(F(2025) – 3·F(2024)) / F(2023) = 2026 · 2025 – 3 · 2025 = 2025 · 2023 = **4096575**.

Программой тоже можно: Python хранит длинные числа точно, но глубина рекурсии больше стандартного предела 1000, его нужно увеличить.

Проверка программой

```
import sys
sys.setrecursionlimit(10000)

def F(n):
    if n == 1:
        return 1
    return (n + 1) * F(n - 1)

print((F(2025) - 3 * F(2024)) // F(2023))
```

Задание 17

Задание выполняется с использованием прилагаемого файла.

В файле содержится последовательность натуральных чисел. Её элементы могут принимать целые значения от 1 до 100 000 включительно. Определите количество пар последовательности, в которых ровно один из двух элементов является пятизначным числом, а сумма элементов пары не меньше максимального элемента последовательности, запись которого оканчивается на 21.

В ответе запишите количество найденных пар, затем максимальную из сумм элементов таких пар. В данной задаче под парой подразумевается два идущих подряд элемента последовательности.

Файл к заданию: 17.txt (53 КБ)

Ответ 357 109327

Сначала находим максимальный элемент, оканчивающийся на 21: он равен 98521. Число 100 000 шестизначное, поэтому пятизначность проверяем двойным неравенством. Условие «ровно один из двух» удобно записать как сумму логических значений, равную 1.

Программа на Python

```
a = [int(x) for x in open('17.txt')]
m = max(x for x in a if x % 100 == 21)

def five(x):
    return 10000 <= x <= 99999

c, best = 0, 0
for i in range(len(a) - 1):
    x, y = a[i], a[i + 1]
    if five(x) + five(y) == 1 and x + y >= m:
        c += 1
        best = max(best, x + y)
```

```
print(c, best)
```

Ответ: 357 109327.

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 = 18), каждая ячейка которой соответствует клетке квадрата. Внутренние и внешние стены обозначены утолщёнными линиями.

Файл к заданию: 18.xlsx (7 КБ)

Ответ 2364 756

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 6: N10, K14, M15, B18, D18, R18. Ответ — максимум из таблицы максимумов и минимум из таблицы минимумов по всем этим клеткам. Частая ошибка — взять только правую нижнюю клетку.

Ответ: 2364 756.

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) один камень;
– 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: (11, 5), (20, 5), (10, 6), (10, 10).

Игра завершается в тот момент, когда суммарное количество камней в двух кучах становится не менее 81. Победителем считается игрок, сделавший последний ход, то есть первым получивший такую позицию, при которой в кучах суммарно 81 камня или больше. В начальный момент в первой куче было 7 камней, во второй куче — S камней; $1 \le S \le 73$.

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 19

Ване нужно, чтобы после хода Пети он смог одним ходом довести сумму до 81. Самый сильный ход Вани — удвоить бóльшую кучу. Значит, ищем наименьшее S, при котором у Пети есть ход, после которого (не выиграв сам) он оставляет позицию, где удвоение даёт сумму не меньше 81.

Перебор программой (функция game возвращает, кто и каким по счёту ходом выигрывает, см. ниже) даёт S = **19**.

Программа для заданий 19–21

```
from functools import lru_cache

def moves(h):
    a, b = h
    return [(a + 1, b), (a * 2, b), (a, b + 1), (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) >= 81:
        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, 74)
          if any(sum(m) < 81 and game(m) == 1 for m in moves((7, 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 бýдет ходить Ваня.

Найденные значения запишите в ответе в порядке возрастания.

Ответ 33 36

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

Все такие S: 33, 36. Два наименьших: 33 и 36.

Программа на Python

```
from functools import lru_cache

def moves(h):
    a, b = h
    return [(a + 1, b), (a * 2, b), (a, b + 1), (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) >= 81:
        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, 74) if game((7, 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 32

Нужны S, для которых game возвращает –2 (Петя проигрывает, Ваня выигрывает своим вторым ходом при лучшей игре Пети). Подходящие значения: 32, 35; наименьшее — **32**.

Программа на Python

```
from functools import lru_cache

def moves(h):
    a, b = h
    return [(a + 1, b), (a * 2, b), (a, b + 1), (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) >= 81:
        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, 74) if game((7, 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 57

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. Порядок строк в файле не совпадает с порядком выполнения, поэтому формулы удобно писать, пользуясь тем, что таблица пересчитывается сама.

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: **57**.

Задание 23

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

A. Прибавить 1
B. Прибавить 2
C. Умножить на 2

Первая команда увеличивает число на экране на 1, вторая увеличивает его на 2, третья умножает на 2. Программа для исполнителя — это последовательность команд.

Сколько существует программ, для которых при исходном числе 3 результатом является число 30, и при этом траектория вычислений содержит число 12 и не содержит числа 20?

Траектория вычислений программы — это последовательность результатов выполнения всех команд программы. Например, для программы CAB при исходном числе 7 траектория будет состоять из чисел 14, 15, 17.

Ответ 95580

Траектория проходит через 12, поэтому количество программ = (число программ 3 → 12) · (число программ 12 → 30). Число 20 запрещено — функция возвращает 0, если попала в него.

Программа на Python

```
def f(x, y):
    if x > y or x == 20:
        return 0
    if x == y:
        return 1
    return f(x + 1, y) + f(x + 2, y) + f(x * 2, y)

print(f(3, 12) * f(12, 30))
```

Ответ: **95580**.

Задание 24

Задание выполняется с использованием прилагаемого файла.

Текстовый файл состоит из цифр 0, 1, 2, 3, 4, 5 и знаков арифметических операций «+» и «*» (сложение и умножение). 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 (977 КБ)

Ответ 270

Идём по строке и поддерживаем длину cur самого длинного корректного выражения, которое заканчивается на текущем символе (выражение, оканчивающееся знаком, считаем «незаконченным»). 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 («0» и следом цифра) 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: **270**.

Задание 25

Назовём маской числа последовательность цифр, в которой также могут встречаться следующие символы:

– символ «?» означает ровно одну произвольную цифру;

– символ «*» означает любую последовательность цифр произвольной длины; в том числе «*» может задавать и пустую последовательность.

Например, маске 123*4?5 соответствуют числа 123405 и 12300405.

Среди натуральных чисел, не превышающих 10^{10}, найдите все числа, соответствующие маске 4?7*31?2, делящиеся на 31013 без остатка.

В ответе запишите в первом столбце таблицы все найденные числа в порядке возрастания, а во втором столбце — соответствующие им результаты деления этих чисел на 31013.

Количество строк в таблице для ответа избыточно.

Ответ 45713162 1474
427483192 13784
4077403162 131474
4578263112 147624
4673783152 150704

Перебирать все числа до 10^{10} слишком долго, а кратных 31013 — около 322 тысяч, это доли секунды. Модуль fnmatch понимает маски с «?» и «*» в том же смысле, что и в условии; для строк из цифр они работают как нужно.

Другой способ — перебрать сами числа, подходящие под маску (вместо «*» — от 0 до 3 цифр, вместо «?» — любая цифра), и проверить делимость: так получается всего около 111 тысяч кандидатов.

Программа на Python

```
from fnmatch import fnmatch

for x in range(31013, 10 ** 10 + 1, 31013):       # перебираем только кратные 31013
    if fnmatch(str(x), '4?7*31?2'):
        print(x, x // 31013)
```

Ответ:
45713162 1474
427483192 13784
4077403162 131474
4578263112 147624
4673783152 150704

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). Ответ для приведённого примера: 4 122000.

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

Файл к заданию: 26.txt (110 КБ)

Ответ 792 249936

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 (в примере копия в 05:05:05 имеет объём 20 000 + 110 000 = 130 000).

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: 792 249936.

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 значениям их кинетической энергии, то есть разбить их множество на K непересекающихся непустых подмножеств (кластеров), таких, что модуль разности кинетических энергий 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 кинетических энергий 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 даны в одинаковых для всех частиц единицах измерения, обозначения единиц измерения в файле не приводятся. 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 3 кластера (K = 3) с R = 1,5 для каждого.

Для каждого кластера определите его центр, затем найдите два числа: Q1 — наибольшее евклидово расстояние между частицами одного кластера, имеющими признак III, и Q2 — минимальное значение кинетической энергии для центра кластера.

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

Для справки. Кинетическая энергия E частицы массы m, обладающей скоростью $v = (V_x, V_y)$, вычисляется по формуле: $E = m(V_x^2 + V_y^2)/2$.

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 кинетическая энергия равна 2,5.

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

Файл к заданию: 27.txt (81 КБ)

Ответ 1618573 29822

Посчитаем энергию каждой частицы и отсортируем частицы по энергии. Разбиение на кластеры единственно: соседние по энергии частицы одного кластера отличаются не больше чем на R, а между кластерами — разрыв больше R. Поэтому кластеры получаются разрезанием отсортированного списка там, где разность соседних энергий больше 1,5.

Центр кластера ищем по определению — перебором: для каждой частицы считаем сумму модулей разностей энергии с остальными частицами кластера. Кластеры по 700–900 частиц, поэтому квадратичный перебор занимает несколько секунд. Ускорить можно, заметив, что центр — медиана энергий кластера.

Q1 — наибольшее расстояние между частицами с признаком III, взятыми из одного кластера (пары из разных кластеров не рассматриваются).

Программа на Python

```
from math import dist

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 * (vx ** 2 + vy ** 2) / 2, x, y, f))   # энергия, координаты, признак

parts.sort()                                # по энергии
clusters, cur = [], [parts[0]]
for p in parts[1:]:
    if p[0] - cur[-1][0] > 1.5:             # большой разрыв по энергии — новый кластер
        clusters.append(cur)
        cur = []
    cur.append(p)
clusters.append(cur)
print(len(clusters))                        # должно быть 3

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

Ответ: 1618573 29822.