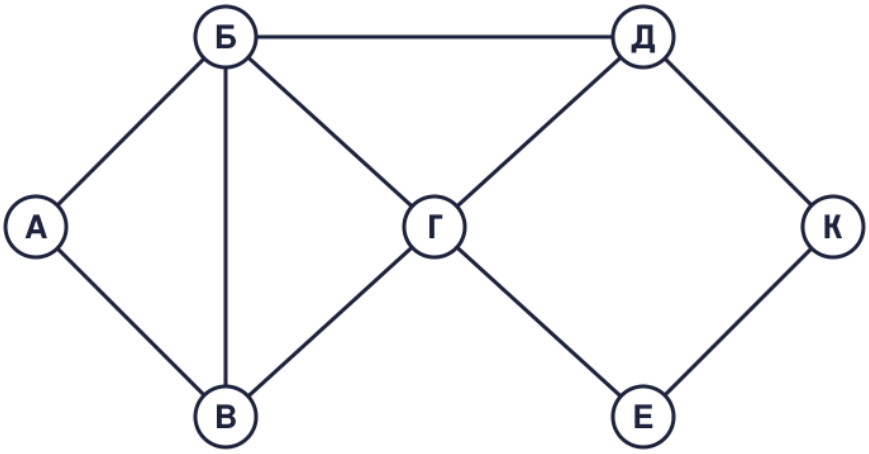
ЕГЭНИЙ · Информатика

Вариант 2 · 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			16	53	49		29
2					38	46	
3	16				13	43	45
4	53						52
5	49	38	13				
6		46	43				
7	29		45	52			



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 Б в пункт Д.

В ответе запишите целое число.

Ответ 62

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

Сопоставляем вершины по степеням и по соседям: А — вершина степени 2, степени её соседей: 3, 4; так устроен только П4; Б — вершина степени 4, соединённая с А; этому соответствует только П1; В — вершина степени 3, соединённая с А, Б; этому соответствует только П7; Г — вершина степени 4, соединённая с Б, В; этому соответствует только П3; Д — вершина степени 3, соединённая с Б, Г; этому соответствует только П5.

Дорога ГД — это П3–П5, её длина 13 км. Дорога БД — это П1–П5, её длина 49 км.

Сумма: 13 + 49 = **62**.

Задание 2

Миша заполнял таблицу истинности логической функции F

¬(w → z) ∨ (x ≡ y) ∨ (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		0
0	1		0	0
1		0	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 zxyw

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 (not (w <= z)) or (x == y) or (y and not w)

answers = set()
for a1, a2, a3, a4, a5 in product([0, 1], repeat=5):
    t = [(a1, a2, 0, a3), (0, 1, a4, 0), (1, a5, 0, 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: **zxyw**.

Задание 3

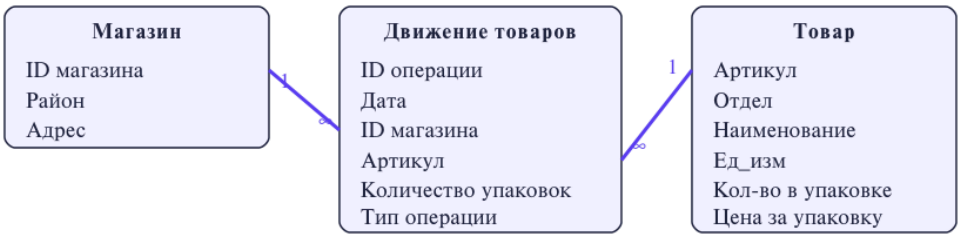
Задание выполняется с использованием прилагаемого файла.

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.


Магазин
ID магазина
Район
Адрес
1
∞
Движение товаров
ID операции
Дата
ID магазина
Артикул
Количество упаковок
Тип операции
1
∞
Товар
Артикул
Отдел
Наименование
Ед_изм
Кол-во в упаковке
Цена за упаковку

```
        return list(csv.reader(f, delimiter=';'))[1:]

goods = {r[0]: r for r in read('товар.csv')}
shops = {r[0]: r[1] for r in read('магазин.csv')}
total = 0
for oid, date, shop, art, cnt, typ in read('движение.csv'):
    day = int(date[:2])
    g = goods[art]
    if typ == 'Продажа' and 11 <= day <= 20 and shops[shop] == 'Центральный' and g[2].startswith('Тетрадь')
        total += int(cnt) * int(g[5])      # упаковки * цена за упаковку
print(total)
```

Ответ: **99050**.

Задание 4

Для кодирования некоторой последовательности, состоящей из букв А, Б, В, Г, Д, Е, решили использовать неравномерный двоичный код, удовлетворяющий условию Фано. Для букв А, Б, В, Г использовали такие кодовые слова: А – 00, Б – 011, В – 101, Г – 1100.

Укажите кратчайшее возможное кодовое слово для буквы Д, при котором код будет удовлетворять условию Фано и для буквы Е останется возможность подобрать кодовое слово. Если таких кодов несколько, укажите код с наименьшим числовым значением.

Примечание. Условие Фано означает, что никакое кодовое слово не является началом другого кодового слова.

Ответ 010

Нарисуем двоичное дерево кодов. Заняты листья 00 (А), 011 (Б), 101 (В), 1100 (Г). Свободные вершины, с которых может начинаться новый код: 010, 100, 111, 1101.

Кодов длины 1 и 2 нет: 0 и 1 — начала занятых кодов, 01, 10, 11 тоже. Длины 3 подходят 010, 100 и 111. После выбора любого из них для Е ещё остаются свободные ветви. Наименьшее числовое значение у **010**.

Программа на Python

```
from itertools import product

known = ['00', '011', '101', '1100']
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)

good = []
for d in words:
    if d not in known and fano(known + [d]):
        # для буквы Е должен найтись ещё один код
        if any(e not in known + [d] and fano(known + [d, e]) for e in words):
            good.append(d)
good.sort(key=lambda w: (len(w), int(w, 2)))
print(good[0])
```

Ответ: **010**.

Задание 5

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

1. Строится троичная запись числа N.

2. Далее эта запись обрабатывается по следующему правилу:
а) если число N делится на 3, то слева к записи дописывается цифра 1, а справа — цифры 02;
б) если число N на 3 не делится, то справа дописывается остаток от деления на 3 суммы цифр троичной записи числа N.

Полученная таким образом запись является троичной записью искомого числа R.

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

Например, для исходного числа $11_{10} = 102_3$ результатом является число $1020_3 = 33_{10}$, а для исходного числа $12_{10} = 110_3$ это число $111002_3 = 353_{10}$.

Укажите минимальное число R, большее 180, которое может быть получено с помощью описанного алгоритма. В ответе запишите это число в десятичной системе счисления.

Ответ 185

Переберём N программой. Для N, не кратных 3, R ≥ 3N, для кратных — ещё больше, поэтому перебора до 1000 с запасом достаточно.

Программа на Python

```
def tern(n):
    s = ''
    while n > 0:
        s = str(n % 3) + s
        n //= 3
    return s

def R(n):
    t = tern(n)
    if n % 3 == 0:
        t = '1' + t + '02'
    else:
        t = t + str(sum(map(int, t)) % 3)
    return int(t, 3)

print(min(R(n) for n in range(1, 1000) if R(n) > 180))
```

Ответ: **185**.

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. У исполнителя существует 6 команд: Поднять хвост; Опустить хвост; Вперёд n — передвижение на n единиц в том направлении, куда указывает голова; Назад n — передвижение в противоположном голове направлении; Направо m — поворот на m градусов по часовой стрелке; Налево m — поворот 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 2 [Вперёд 14 Направо 90 Вперёд 30 Направо 90]
Поднять хвост
Направо 90 Вперёд 8 Направо 90 Вперёд 6 Налево 180
Опустить хвост
Повтори 2 [Вперёд 25 Направо 90 Вперёд 12 Направо 90]

Определите, сколько точек с целочисленными координатами находятся внутри объединения фигур, ограниченных заданными алгоритмом линиями. Точки, лежащие на любой из нарисованных линий (в том числе на участках, оказавшихся внутри другой фигуры), учитывать не следует.

Ответ 450

Первый прямоугольник: 0 ≤ x ≤ 30, 0 ≤ y ≤ 14.

Перемещение без рисования: Направо 90 — голова вправо, Вперёд 8 — точка (8; 0), Направо 90 — голова вниз, Вперёд 6 — точка (8; –6), Налево 180 — голова вверх.

Второй прямоугольник: 8 ≤ x ≤ 20, -6 ≤ y ≤ 19. Он выходит за первый снизу и сверху.

Точки, лежащие на нарисованных линиях, не считаются — в том числе отрезки сторон, оказавшиеся внутри другого прямоугольника. Поэтому удобно посчитать программой: точка подходит, если она не лежит ни на одной линии и находится внутри хотя бы одного прямоугольника.

Программа на Python

```
A = (0, 30, 0, 14)      # x1, x2, y1, y2
B = (8, 20, -6, 19)

def strictly(r, x, y):
    return r[0] < x < r[1] and r[2] < y < r[3]

def on_border(r, x, y):
    return (r[0] <= x <= r[1] and r[2] <= y <= r[3]) and not strictly(r, x, y)

c = 0
for x in range(-50, 51):
    for y in range(-50, 51):
```

```
        if (strictly(A, x, y) or strictly(B, x, y)) and not on_border(A, x, y) and not on_border(B, x, y):
            c += 1
print(c)
```

Ответ: **450**.

Задание 7

Камера автоматически делает фотографии размером 1536×1024 пикселей. Для кодирования цвета используется палитра из 4096 цветов, коды пикселей записываются в файл один за другим без промежутков, сжатие не применяется. К каждой фотографии добавляется заголовок объёмом 2 Кбайт.

Какое наибольшее количество фотографий поместится на карту памяти объёмом 512 Мбайт?

Ответ 227

Палитра из 4096 = 2^{12} цветов — 12 бит на пиксель.

Один снимок: 1536 · 1024 · 12 бит = 1536 · 1024 · 1,5 байта = 2 359 296 байт = 2304 Кбайт. С заголовком — 2306 Кбайт.

512 Мбайт = 524 288 Кбайт. 524 288 / 2306 ≈ 227,4, целых фотографий — **227**.

Задание 8

Определите количество шестизначных чисел, записанных в восьмеричной системе счисления, в записи которых все цифры различны и никакие две чётные и никакие две нечётные цифры не стоят рядом.

Ответ 1008

Чётные и нечётные цифры должны чередоваться. Восьмеричных цифр 8: чётные 0, 2, 4, 6 и нечётные 1, 3, 5, 7.

Если число начинается с нечётной: позиции 1, 3, 5 — три разные нечётные (4 · 3 · 2 = 24 способа), позиции 2, 4, 6 — три разные чётные, любые, включая 0 (24 способа): 24 · 24 = 576.

Если начинается с чётной: первая цифра — 2, 4 или 6 (3 способа), ещё две чётные на позициях 3, 5 — из оставшихся трёх (3 · 2 = 6), нечётные на позициях 2, 4, 6 — 24 способа: 3 · 6 · 24 = 432.

Всего 576 + 432 = **1008**.

Проверка программой

```
from itertools import product

c = 0
for t in product('01234567', repeat=6):
    if t[0] == '0' or len(set(t)) < 6:
        continue
    if all(int(a) % 2 != int(b) % 2 for a, b in zip(t, t[1:])):
        c += 1
print(c)
```

Задание 9

Задание выполняется с использованием прилагаемого файла.

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

– в строке все числа различны;
– среднее арифметическое трёх наибольших чисел строки меньше удвоенного среднего арифметического четырёх остальных её чисел.

В ответе запишите только число.

Файл к заданию: 9.xlsx (194 КБ)

Ответ 1262

В электронной таблице. Все числа различны, если `=СЧЁТ(A1:G1)=СУММПРОИЗВ(1/СЧЁТЕСЛИ(A1:G1;A1:G1))`, или проще — если для каждой ячейки `СЧЁТЕСЛИ($A1:$G1;A1)=1`. Сумма трёх наибольших: `=НАИБОЛЬШИЙ(A1:G1;1)+НАИБОЛЬШИЙ(A1:G1;2)+НАИБОЛЬШИЙ(A1:G1;3)`, четырёх остальных — сумма строки минус она.

Второе условие: `=сумма3/3 < 2*сумма4/4`. Подсчитайте строки, где оба условия истинны.

Программой. Сохраните лист как CSV с разделителем «;». После сортировки строки три наибольших

числа — последние три.

Программа на Python

```
c = 0
for line in open('9.csv'):
    a = sorted(int(x) for x in line.split(';'))
    if len(set(a)) == 7:
        if sum(a[4:]) / 3 < 2 * sum(a[:4]) / 4:
            c += 1
print(c)
```

Ответ: **1262**.

Задание 10

Файл 10-2.odt содержит первые три главы поэмы Н. В. Гоголя «Мёртвые души». С помощью текстового редактора определите, сколько раз во всём тексте файла встречается слово «Чичиков» (именно в этой форме, с учётом регистра) как отдельное слово. Слова, записанные через дефис (например, «что-то»), считаются состоящими из отдельных слов. В ответе запишите только число.

Текст для проверки программой: TXT.

Ответ 103

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('10-2.txt', encoding='utf-8').read()
print(word(t, 'Чичиков'))
```

Задание 11

В информационной системе для каждого из 30 000 пользователей хранится пароль длиной 64 символа. Для хранения каждого пароля отведено одинаковое и минимально возможное целое число байт, при этом используется посимвольное кодирование, все символы кодируются одинаковым и минимально возможным числом бит.

Известно, что для хранения всех паролей потребовалось не более 1 500 000 байт. Определите максимально возможную мощность алфавита, используемого для записи паролей. В ответе запишите только целое число.

Ответ 64

На один пароль приходится не более 1 500 000 / 30 000 = 50 байт = 400 бит.

64 символа по i бит займут 64·i бит, то есть 8·i байт (целое число), Условие $8 \cdot i \le 50$ даёт $i \le 6{,}25$, значит, i = 6,

При 6 битах на символ можно закодировать до 2^6 = **64** символов.

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 1052 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 995

$1052 = 10000011100_2$.

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 11-разрядного числа даёт $(2^{11} - 1) - 1052 = 995$; ведущие нули на значение не влияют.

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('10000011100'))
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 1111100011_2 = **995**.

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 10.214.87.173 и маска 255.255.255.224. 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 471

Октеты маски, равные 255, переносят соответствующие октеты адреса без изменений, октеты, равные 0, дают 0. Считать нужно только октет, где маска равна 224.

$173 = 10101101_2$, $224 = 11100000_2$. Поразрядная конъюнкция: $10100000_2 = 160$.

Адрес сети: 10.214.87.160. Сумма октетов: 10 + 214 + 87 + 160 = **471**.

Проверка программой

```
a = [10, 214, 87, 173]
m = [255, 255, 255, 224]
net = [x & y for x, y in zip(a, m)]
print(net, sum(net))
```

Задание 14

Операнды арифметического выражения записаны в системе счисления с основанием 19:

$$\mathbf{6x31_{19} + x75_{19} + 2x9x_{19}}$$

В записи чисел переменной x обозначена неизвестная цифра из алфавита 19-ричной системы счисления. Определите наименьшее значение x, при котором значение данного арифметического выражения кратно 17. Для найденного x вычислите частное от деления значения арифметического выражения на 17 и укажите его в ответе в десятичной системе счисления. Основание системы счисления указывать не нужно.

Ответ 3887

Перебираем цифру x от 0 до 18. Функция int(строка, 19) переводит запись из 19-ричной системы; цифры больше 9 обозначаются буквами A–I.

Программа на Python

```
alph = '0123456789ABCDEFGHI'          # цифры 19-ричной системы
for x in alph:
    s = int('6' + x + '31', 19) + int(x + '75', 19) + int('2' + x + '9' + x, 19)
    if s % 17 == 0:
        print(x, s // 17)
        break
```

Наименьшее подходящее x = 10, частное — **3887**.

Задание 15

Обозначим через m & n поразрядную конъюнкцию неотрицательных целых чисел m и n. Так, например, 14 & 5 = 1110_2 & 0101_2 = 0100_2 = 4.

Для какого наименьшего неотрицательного целого числа A формула

((x & 26 ≠ 0) ∧ (x & 13 = 0)) → (x & A ≠ 0)

тождественно истинна (то есть принимает значение 1 при любом неотрицательном целом значении переменной x)?

Ответ 18

26 = 11010_2, 13 = 01101_2. Посылка истинна, когда в x нет битов 8, 4 и 1 (x & 13 = 0), но есть хотя бы один из битов 16, 8, 2 (x & 26 ≠ 0). С учётом первого — хотя бы один из битов 16 и 2.

Такими могут быть, например, x = 2 и x = 16. Чтобы x & A ≠ 0 для всех таких x, A должно содержать оба бита 16 и 2: иначе x, состоящий только из второго из них, сделает формулу ложной.

Наименьшее A = 16 + 2 = **18**.

Проверка программой

```
def F(x, A):
    return (not ((x & 26) != 0 and (x & 13) == 0)) or (x & A) != 0

for A in range(0, 1024):
    if all(F(x, A) for x in range(0, 1024)):
        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) = F(n / 2) + 1, если n > 1 и n чётное;
F(n) = F(n – 1) + n, если n > 1 и n нечётное.

Определите количество натуральных значений n из отрезка [1; 1000], для которых значение F(n) кратно 7.

Ответ 137

Глубина рекурсии здесь небольшая (каждые два шага n уменьшается примерно вдвое), поэтому функцию можно вычислить напрямую. Декоратор lru_cache запоминает уже вычисленные значения и ускоряет перебор.

Программа на Python

```
from functools import lru_cache

@lru_cache(None)
def F(n):
    if n == 1:
        return 1
    if n % 2 == 0:
```

```
        return F(n // 2) + 1
    return F(n - 1) + n

print(sum(1 for n in range(1, 1001) if F(n) % 7 == 0))
```

Ответ: **137**.

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 37.

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 (53 КБ)

Ответ 667 8607

Минимальный положительный элемент, кратный 37, равен 37. Ноль не является ни положительным, ни отрицательным: условие x < 0 его не учитывает, x > 0 — тоже. 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 % 37 == 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: 667 8607.

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.

Определите максимальную и минимальную денежные суммы, которые может собрать Робот, пройдя из левой верхней клетки в правую нижнюю. Маршруты, которые заканчиваются в тупике (клетке со стенами справа и снизу, отличной от правой нижней), не учитываются. В ответе укажите два числа — сначала максимальную сумму, затем минимальную.

Исходные данные представляют собой электронную таблицу размером N × N (в файле N = 16), каждая ячейка которой соответствует клетке квадрата. Внутренние и внешние стены обозначены утолщёнными линиями.

Файл к заданию: 18.xlsx (6 КБ)

Ответ 2208 1154

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.

В этом поле есть тупики (5 шт.): в них Робот может попасть, но не может выйти, и маршрут через них до правой нижней клетки не дойдёт. Формулы это учитывают автоматически: клетки после тупика получают значение только от других соседей. Если у клетки оба соседа отрезаны стенами, в неё прийти нельзя — оставьте её пустой и не ссылайтесь на неё.

Ответ — значения в правой нижней клетке таблиц максимумов и минимумов: 2208 1154.

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]
print(mx[n - 1][n - 1], mn[n - 1][n - 1])
```

Задание 19

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

– добавить в кучу один камень;
– добавить в кучу три камня;
– увеличить количество камней в куче в два раза.

Например, имея кучу из 15 камней, за один ход можно получить кучу из 16, 18 или 30 камней. У каждого игрока, чтобы делать ходы, есть неограниченное количество камней.

Игра завершается в тот момент, когда количество камней в куче становится не менее 59. Победителем считается игрок, сделавший последний ход, то есть первым получивший кучу, в которой будет 59 или больше камней. В начальный момент в куче было S камней; $1 \le S \le 58$.

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

Укажите минимальное значение S, при котором Петя не может выиграть за один ход, но при любом ходе Пети Ваня может выиграть своим первым ходом.

Ответ 29

Выиграть одним ходом можно из позиций $S \ge 30$: удвоение даёт не меньше $60 \ge 59$. Нужна позиция $S < 30$, из которой любой ход Пети ведёт в такую позицию: $S + 1 \ge 30$, $S + 3 \ge 30$ и $2S \ge 30$.

Перебор программой (см. ниже) даёт наименьшее такое S = **29**.

Программа для заданий 19–21

```
from functools import lru_cache

def moves(h):
    return [h + 1, h + 3, h * 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 h >= 59:
        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, 59) if game(s) == -1))
```

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 26 28

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

Все такие S: 26, 28. Два наименьших: 26 и 28.

Программа на Python

```
from functools import lru_cache

def moves(h):
    return [h + 1, h + 3, h * 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 h >= 59:
        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, 59) if game(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 25

Нужны S, для которых game возвращает –2 (Петя проигрывает, Ваня выигрывает своим вторым ходом при лучшей игре Пети). Подходящие значения: 25, 27; наименьшее — **25**.

Программа на Python

```
from functools import lru_cache

def moves(h):
    return [h + 1, h + 3, h * 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 h >= 59:
        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, 59) if game(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 количество процессов, которые выполняются параллельно на 23-й мс. Считать, что каждый процесс начинается в самое раннее допустимое время. Нумерация миллисекунд начинается с 1.

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

Для приведённой таблицы процесс 3 начинается на 8-й мс, заканчивается на 9-й мс, а на 7-й мс выполняются два процесса — 2 и 5.

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

Файл к заданию: 22.xlsx (6 КБ)

Ответ 8

Сначала для каждого процесса находим момент окончания (как в классическом задании 22), затем момент начала = окончание − время. Процесс, начавшийся в момент s и закончившийся в момент f, занимает миллисекунды с номерами s + 1, …, f. В примере процесс 3: s = 7, f = 9 — он занимает 8-ю и 9-ю мс.

Остаётся посчитать процессы, для которых $s < 23 \le f$.

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):
    if p not in finish:
        finish[p] = max([fin(q) for q in deps[p]], default=0) + time[p]
    return finish[p]
```

```
ms = 23
c = 0
for p in time:
    begin = fin(p) - time[p]          # процесс занимает миллисекунды begin+1 … fin(p)
    if begin < ms <= fin(p):
        c += 1
print(c)
```

Ответ: **8**.

Задание 23

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

A. Прибавить 1
B. Поменять местами
C. Прибавить 10

Первая команда увеличивает число на экране на 1, третья — на 10. 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 10 результатом является число 57?

Траектория вычислений программы — это последовательность результатов выполнения всех команд программы. Например, для программы ABC при исходном числе 13 траектория состоит из чисел 14, 41, 51.

Ответ 3166

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

Вторая команда меняет число так: x – 10·t – u + 10·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 + 10, 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(10, 57))
```

Ответ: **3166**.

Задание 24

Задание выполняется с использованием прилагаемого файла.

Текстовый файл состоит из заглавных букв латинского алфавита. Определите максимальное количество идущих подряд символов, среди которых никакие две соседние буквы не совпадают, а буква Q встречается не более двух раз.

В ответе укажите количество символов.

Файл к заданию: 24.txt (977 КБ)

Ответ 255

Метод двух указателей: правая граница идёт по строке, левая сдвигается, когда окно перестаёт удовлетворять условию. Если две соседние буквы совпали, ни одно окно не может их содержать — начинаем окно с текущей буквы. Если букв Q стало три, сдвигаем левую границу, пока одна из Q не выйдет из окна.

Программа на Python

```
s = open('24.txt').read().strip()
best = 0
left = 0          # начало текущего окна
```

```
q = 0            # сколько букв Q в окне
for r in range(len(s)):
    if r > 0 and s[r] == s[r - 1]:
        left, q = r, 0               # две одинаковые соседние: окно начинаем заново
    if s[r] == 'Q':
        q += 1
    while q > 2:                     # лишняя Q — сдвигаем левую границу
        if s[left] == 'Q':
            q -= 1
        left += 1
    best = max(best, r - left + 1)
print(best)
```

Ответ: **255**.

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 800 000, в порядке возрастания и ищет среди них такие, для которых M оканчивается на 4. 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 800004 400004
800009 114294
800013 266674
800024 400014
800033 61554

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 = 800001
while found < 5:
    m = M(n)
    if m % 10 == 4 and m != 0:
        print(n, m)
        found += 1
    n += 1
```

Ответ:
800004 400004
800009 114294
800013 266674
800024 400014
800033 61554

Задание 26

Задание выполняется с использованием прилагаемого файла.

Системный администратор раз в неделю создаёт архив пользовательских файлов. Однако объём диска, куда он помещает архив, может быть меньше, чем суммарный объём архивируемых файлов. Известно, какой объём занимает файл каждого пользователя.

По заданной информации об объёме файлов пользователей и свободном объёме на архивном диске определите максимальное число пользователей, чьи файлы можно сохранить в архиве, а также максимальный размер имеющегося файла, который может быть сохранён в архиве, при условии, что сохранены файлы максимально возможного числа пользователей.

Входные данные. В первой строке входного файла находятся два числа: S — размер свободного места на диске (натуральное число, не превышающее 10 000 000) и N — количество пользователей (натуральное число, не превышающее 10 000). В следующих N строках находятся значения объёмов файлов каждого пользователя (все числа натуральные, не превышающие 10 000), каждое в отдельной строке.

Выходные данные. Запишите в ответе два числа: сначала наибольшее число пользователей, чьи файлы могут быть помещены в архив, затем максимальный размер имеющегося файла, который может быть сохранён в архиве, при условии, что сохранены файлы максимально возможного числа пользователей.

Типовой пример организации данных во входном файле:

```
100 4
80
30
50
40
```

При таких исходных данных можно сохранить файлы максимум двух пользователей. Возможные объёмы этих двух файлов 30 и 40, 30 и 50 или 40 и 50. Наибольший объём файла из перечисленных пар — 50, поэтому ответ для приведённого примера: 2 50.

Файл к заданию: 26.txt (43 КБ)

Ответ 2400 2924

Чтобы сохранить как можно больше файлов, берём самые маленькие: сортируем и набираем, пока помещаются. Так получаем наибольшее количество k.

Чтобы в архиве оказался как можно больший файл, оставляем k – 1 самых маленьких, а вместо k-го берём наибольший файл, который помещается в оставшееся место.

Программа на Python

```
f = open('26.txt')
s, n = map(int, f.readline().split())
a = sorted(int(f.readline()) for _ in range(n))

total, k = 0, 0
while k < n and total + a[k] <= s:     # берём самые маленькие файлы
    total += a[k]
    k += 1

# заменяем самый большой из взятых файлов на наибольший, который ещё помещается
free = s - (total - a[k - 1])
best = max(x for x in a[k - 1:] if x <= free)
print(k, best)
```

Ответ: 2400 2924.

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 их 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 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 4 кластера (K = 4) с R = 2,0 для каждого.

Для каждого кластера определите его центр, затем найдите два числа: Q1 — наименьшее евклидово расстояние между центрами кластеров, и Q2 — среднее арифметическое модулей 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 (91 КБ)

Ответ 109053 145086

Считаем модуль импульса каждой частицы, сортируем и режем список там, где разность соседних значений больше R = 2,0 — получаем четыре кластера.

Центр каждого кластера — частица с минимальной суммой модулей разностей импульса (её можно найти перебором или как медиану). Q1 — минимальное из шести расстояний между центрами. Q2 — среднее значение импульса в самом маленьком по числу частиц кластере.

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))      # импульс, координаты

parts.sort()
clusters, cur = [], [parts[0]]
for p in parts[1:]:
    if p[0] - cur[-1][0] > 2.0:
        clusters.append(cur)
        cur = []
    cur.append(p)
clusters.append(cur)
print(len(clusters))                        # должно быть 4

centers = [min(c, key=lambda p: sum(abs(p[0] - r[0]) for r in c)) for c in clusters]
q1 = min(dist(a[1:], b[1:]) for i, a in enumerate(centers) for b in centers[i + 1:])
small = min(clusters, key=len)
q2 = sum(p[0] for p in small) / len(small)
print(int(q1 * 10000), int(q2 * 10000))
```

Ответ: 109053 145086.