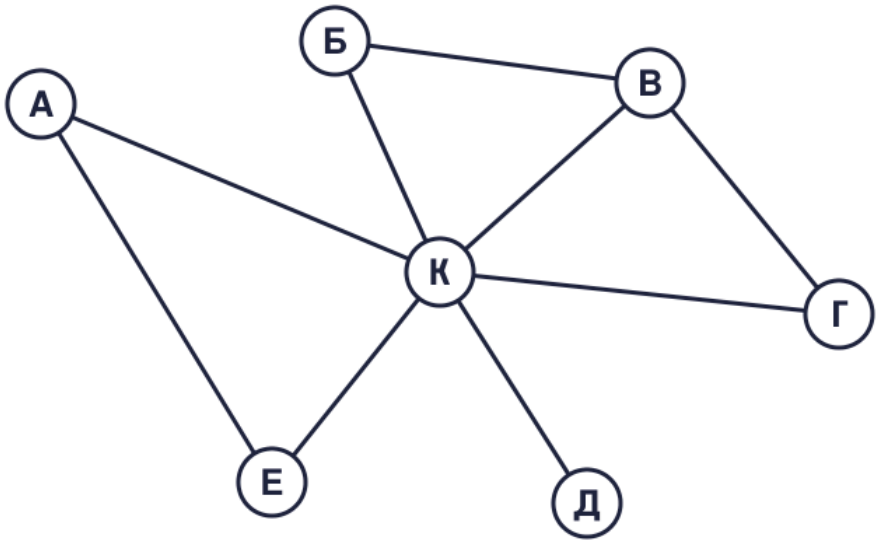
ЕГЭНИЙ · Информатика

Вариант 19 · 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		55	24	17	31	46	45
2	55		43				
3	24	43					
4	17					39	
5	31						
6	46			39			25
7	45					25	



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 ВК отличается от длины дороги ДК.

В ответе запишите целое число.

Ответ 15

Посчитаем степени вершин (количество дорог из пункта). На графе: А — 2, Б — 2, В — 3, Г — 2, Д — 1, Е — 2, К — 6. В таблице: П1 — 6, П2 — 2, П3 — 2, П4 — 2, П5 — 1, П6 — 3, П7 — 2.

Сопоставляем вершины по степеням и по соседям: степень 3 на графе только у вершины В, в таблице — только у пункта 6: В = П6; степень 1 на графе только у вершины Д, в таблице — только у пункта 5: Д = П5;

степень 6 на графе только у вершины К, в таблице — только у пункта 1: К = П1.

Дорога ВК — это П6–П1, её длина 46 км. Дорога ДК — это П5–П1, её длина 31 км.

Разность: |46 – 31| = **15**.

Задание 2

Миша заполнял таблицу истинности логической функции F

((x ≡ ¬z) → ¬(w → x)) ∨ ¬y,

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	0			0
	1	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 wxyz

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

answers = set()
for a1, a2, a3, a4, a5, a6, a7 in product([0, 1], repeat=7):
    t = [(a1, 1, a2, a3), (0, 0, a4, a5), (a6, 1, 1, a7)]
    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: **wxyz**.

Задание 3

Задание выполняется с использованием прилагаемого файла.

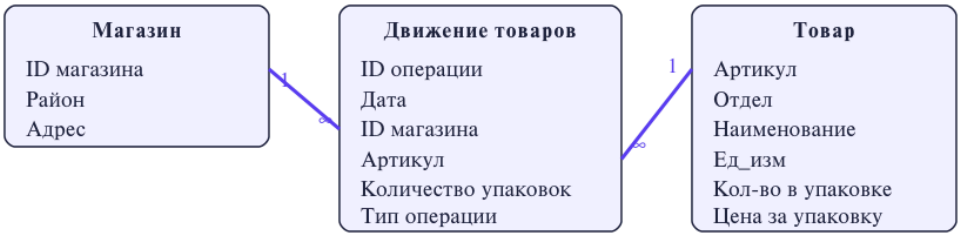
В файле приведён фрагмент базы данных «Канцтовары» о поставках товаров в магазины районов города. База данных состоит из трёх таблиц.

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 общую выручку (в рублях) от продажи всех видов тетрадей в магазинах Первомайского района за период с 8 по 23 июня включительно. В ответе запишите только число.

Файл к заданию: 3.xlsx (107 КБ)

Ответ 170420

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)` и цену за упаковку (столбец 6 листа «Товар»).

Включите автофильтр и оставьте: тип операции — Продажа, даты с 08.06.2026 по 23.06.2026, район — Первомайский, наименование начинается со слова «Тетрадь». Выручка строки = количество упаковок · цена за упаковку; цена указана за упаковку, поэтому количество товара в упаковке не нужно. 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 typ == 'Продажа' and 8 <= day <= 23 and shops[shop] == 'Первомайский' and g[2].startswith('Тетрадь')
        total += int(cnt) * int(g[5])      # упаковки * цена за упаковку
print(total)
```

Ответ: **170420**.

Задание 4

По каналу связи передаются сообщения, содержащие только буквы из набора: Г, Д, К, О, Р. Для передачи используется двоичный код, удовлетворяющий условию Фано. Кодовые слова для некоторых букв известны: Г – 11, Д – 010, Р – 011. Для двух оставшихся букв К и О кодовые слова неизвестны.

Какое наименьшее количество двоичных знаков потребуется для кодирования слова ГОРОДОК, 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 16

Нарисуем двоичное дерево известных кодов. Свободные вершины, с которых может начинаться новый код: 00, 10. Чем чаще буква встречается в слове, тем короче должен быть её код.

Количество букв в слове: Г — 1, Д — 1, К — 1, О — 3, Р — 1. Оптимально, например: К – 00, О – 10.

Длина кода слова: 2·1 + 3·1 + 2·1 + 2·3 + 3·1 = **16**.

Проверим перебором всех кодов длиной до 5 для неизвестных букв:

Программа на Python

```
from itertools import product

known = {'Г': '11', 'Д': '010', 'Р': '011'}
word = 'ГОРОДОК'
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=2):
    codes = dict(known)
    codes.update(zip(['К', 'О'], 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: **16**.

Задание 5

На вход алгоритма подаётся натуральное число N (N ≥ 4). 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 делится на 3, то к этой записи справа дописываются три её последние цифры;
б) если число N на 3 не делится, то остаток от деления на 3 умножается на 3, переводится в двоичную запись и дописывается в конец записи числа N.

Полученная таким образом запись является двоичной записью искомого числа R.

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

Например, для исходного числа $14_{10} = 1110_2$ результатом является число $1110110_2 = 118_{10}$, а для исходного числа $20_{10} = 10100_2$ результатом является число $10100110_2 = 166_{10}$.

Укажите минимальное число N, после обработки которого с помощью этого алгоритма получается число R, не меньшее 335. В ответе запишите это число в десятичной системе счисления.

Ответ 42

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 % 3 == 0:
        b = b + b[-3:]
    else:
        b = b + bin(n % 3 * 3)[2:]
    return int(b, 2)

print(min(n for n in range(4, 1000) if R(n) >= 335))
```

Ответ: **42**.

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 2 [Вперёд 26 Направо 90 Вперёд 14 Направо 90]
Поднять хвост
Вперёд 10 Направо 90 Вперёд 7 Налево 90
Опустить хвост
Повтори 4 [Вперёд 15 Направо 90 Вперёд 21 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 479

Первый прямоугольник: $0 \le x \le 14$, $0 \le y \le 26$. Перемещение без рисования приводит Черепаху в точку (7; 10) головой вверх. Второй прямоугольник: $7 \le x \le 28$, $10 \le y \le 25$.

Точки на нарисованных линиях не считаются — в том числе на отрезках сторон, оказавшихся внутри другого прямоугольника. Проще всего посчитать программой: точка подходит, если она лежит строго внутри хотя бы одного прямоугольника и не лежит ни на одной из линий.

Программа на Python

```
A = (0, 14, 0, 26)      # x1, x2, y1, y2
B = (7, 28, 10, 25)

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(-100, 101):
    for y in range(-100, 101):
        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: **479**.

Задание 7

Голосовое сообщение оцифровывается в формате моно с частотой дискретизации 32 000 Гц и глубиной кодирования 32 бит. Определите наименьшее количество Кбайт, необходимое для сохранения сообщения в памяти (без учёта заголовка), если его длительность — 1 мин 2 с. В ответе укажите только число.

Ответ 7750

Длительность: 1 · 60 + 2 = 62 с. Объём: 1 · 32 000 · 4 байта · 62 = 7 936 000 байт.

В Кбайтах: 7 936 000 / 1024 ≈ 7750,00: **7750**.

Задание 8

Определите количество четырёхзначных чисел, записанных в восьмеричной системе счисления, в записи которых цифра 5 встречается ровно один раз, а никакие две нечётные цифры не стоят рядом.

Ответ 472

Переберём все строки из 4 цифр восьмеричной системы (0–7), отбросим начинающиеся с нуля и проверим условие. Всего строк $8^4 = 4096$ — перебор мгновенный.

Программа на Python

```
from itertools import product

c = 0
for t in product('01234567', repeat=4):
    if t[0] != '0' and t.count('5') == 1 and not any(int(a) % 2 == 1 and int(b) % 2 == 1 for a, b in zip(t,
        c += 1
print(c)
```

Ответ: **472**.

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;
– удвоенная сумма повторяющихся чисел больше суммы неповторяющихся.

В ответе запишите только число.

Файл к заданию: 9.xlsx (185 КБ)

Ответ 1302

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 2 * sum(rep) > sum(uniq):
```

```
            c += 1
print(c)
```

Ответ: **1302**.

Задание 10

Файл 10-1.odt содержит текст повести А. С. Пушкина «Дубровский». Определите, сколько раз в тексте глав IX, X и XI (от заголовка «ГЛАВА IX» до заголовка «ГЛАВА XII») встречается слово «он», записанное строчными буквами, как отдельное слово. Слова «Он» с прописной буквы 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 44

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 IX\n'):t.index('ГЛАВА XII\n')]
print(word(t, 'он'))
```

Задание 11

При регистрации в компьютерной системе каждому объекту сопоставляется идентификатор из 30 символов, для записи которого используется алфавит мощностью 264. В базе данных для хранения каждого идентификатора отведено одинаковое и минимально возможное целое число байт, все символы кодируются одинаковым и минимально возможным количеством бит.

Кроме идентификатора, для каждого объекта в базе хранятся дополнительные сведения, для которых отведено одинаковое для всех объектов целое число байт. Для хранения сведений о 40 объектах потребовалось 2760 байт. Сколько байт выделено для хранения дополнительных сведений об одном объекте? В ответе запишите только целое число.

Ответ 35

Символов 264: $2^8 < 264 \le 2^9$ — по 9 бит на символ. Идентификатор: $30 \cdot 9 = 270$ бит → 34 байт (округляем вверх).

На один объект: 2760 / 40 = 69 байт. Дополнительные сведения: 69 – 34 = **35**.

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 1084 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, S, q1	1, S, q1	0, L, q1

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

Ответ 1085

1084 = 10000111100_2.

q0 сдвигает головку на последнюю цифру. q1 прибавляет единицу: единицы заменяет нулями и идёт влево (перенос), на первом нуле или пустой ячейке пишет 1 и останавливается. Программа прибавляет к числу 1.

Проверим, проследив такты программой.

Программа на Python

```
prog = {
    'q0': {'λ': ('λ', 'L', 'q1')},
    'q1': {'λ': ('1', 'S', 'q1'), '0': ('1', 'S', 'q1'), '1': ('0', 'L', 'q1')},
}
tape = dict(enumerate('10000111100'))
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 10000111101_2 = **1085**.

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 100.185.27.210 и маска 255.255.248.0. 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 309

Октеты маски, равные 255, переносят соответствующие октеты адреса без изменений, октеты, равные 0, дают 0. Считать нужно только октет, где маска равна 248.

27 = 00011011_2, 248 = 11111000_2. Поразрядная конъюнкция: 00011000_2 = 24.

Адрес сети: 100.185.24.0. Сумма октетов: 100 + 185 + 24 + 0 = **309**.

Проверка программой

```
a = [100, 185, 27, 210]
m = [255, 255, 248, 0]
net = [x & y for x, y in zip(a, m)]
print(net, sum(net))
```

Задание 14

Значение арифметического выражения

49^(127)+7^(67)−7^(5)−x,

где x — целое положительное число, не превышающее 3000, записали в семеричной системе счисления. Определите наибольшее значение x, при котором в данной записи значения арифметического выражения содержится ровно 188 нулей.

В ответе запишите число в десятичной системе счисления.

Ответ 2996

49127=7254: в семеричной записи это единица и 254 нулей. Слагаемое 767 добавляет ещё одну единицу,

а вычитание 75+x «занимает» единицу из старшего разряда: младшие разряды превращаются в цифры 6, 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(3000, 0, -1):
    if zeros(49 ** 127 + 7 ** 67 - 7 ** 5 - x, 7) == 188:
        print(x)
        break
```

Ответ: **2996**.

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 = [61; 163].

Для какого наибольшего натурального числа A логическое выражение

ДЕЛ(x, A) ∨ ((x ∈ B) → ¬ДЕЛ(x, 34))

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

Ответ 34

Выражение ложно, только если x ∈ B, x делится на 34 и не делится на A. Значит, A должно делить все кратные 34 на отрезке: 68, 102, 136.

Наибольшее такое A — их НОД: **34**.

Проверка программой

```
def F(x, A):
    return x % A == 0 or (not (61 <= x <= 163)) or x % 34 != 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 ≥ 2030;
F(n) = n + F(n + 3), если n < 2030.

Чему равно значение выражения F(2015) – F(2024)?

Ответ 6054

F(2015) = 2015 + F(2018) = 2015 + 2018 + F(2021) = … Раскрывая определение 3 раз, получаем F(2015) = 2015 + 2018 + 2021 + F(2024).

Поэтому F(2015) – F(2024) = 2015 + 2018 + 2021 = **6054**. Глубина рекурсии здесь небольшая, программа считает напрямую.

Проверка программой

```
def F(n):
    if n >= 2030:
        return n
    return n + F(n + 3)

print(F(2015) - F(2024))
```

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 остаток от деления на 38 хотя бы одного из элементов равен минимальному элементу последовательности.

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 (48 КБ)

Ответ 443 194809

Минимальный элемент последовательности равен 15. Для каждой пары соседних элементов проверяем остатки от деления на 38.

Программа на Python

```
a = [int(x) for x in open('17.txt')]
m = min(a)
c, best = 0, 0
for i in range(len(a) - 1):
    x, y = a[i], a[i + 1]
    if x % 38 == m or y % 38 == m:
        c += 1
        best = max(best, x + y)
print(c, best)
```

Ответ: 443 194809.

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 = 12), каждая ячейка которой соответствует клетке квадрата. Внутренние и внешние стены обозначены утолщёнными линиями.

Файл к заданию: 18.xlsx (6 КБ)

Ответ 1489 870

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 (7 шт.): в них Робот может попасть, но выйти не может, и маршрут через них до правой нижней клетки не дойдёт. Формулы учитывают это сами: 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: 1489 870.

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 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 99. Победителем считается игрок, сделавший последний ход, то есть первым получивший такую позицию, при которой в кучах суммарно 99 камней или больше. В начальный момент в первой куче было 9 камней, во второй куче — S камней; $1 \le S \le 89$.

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 23

Ване нужно, чтобы после хода Пети он смог одним ходом довести сумму до 99. 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 = **23**.

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) >= 99:
        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, 90)
          if any(sum(m) < 99 and game(m) == 1 for m in moves((9, 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 22 39

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

Все такие S: 22, 39, 40, 41, 42. Два наименьших: 22 и 39.

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) >= 99:
        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, 90) if game((9, 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 35

Нужны S, для которых game возвращает –2: Петя проигрывает, а Ваня при лучшей игре Пети выигрывает своим вторым ходом. Наименьшее такое S — **35**.

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) >= 99:
        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, 90) if game((9, 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 50

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: **50**.

Задание 23

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

A. Прибавить 1
B. Прибавить 3
C. Умножить на 3

Первая команда увеличивает число на экране на 1, вторая увеличивает его на 3, третья умножает на 3. Программа для исполнителя — это последовательность команд.

Сколько существует программ, для которых при исходном числе 1 результатом является число 36, и при этом траектория вычислений содержит число 17 и не содержит числа 28?

Траектория вычислений программы — это последовательность результатов выполнения всех команд программы. Например, для программы CAB при исходном числе 7 траектория будет состоять из чисел 21, 22, 25.

Ответ 165771

Траектория проходит через 17, поэтому количество программ = (число программ 1 → 17) · (число программ 17 → 36). Число 28 запрещено — функция возвращает 0, если траектория в него попала.

Программа на Python

```
def f(x, y):
    if x > y or x == 28:
        return 0
    if x == y:
        return 1
    return f(x + 1, y) + f(x + 3, y) + f(x * 3, y)

print(f(1, 17) * f(17, 36))
```

Ответ: **165771**.

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 Z встречается не более одного раза.

В ответе укажите количество символов.

Файл к заданию: 24.txt (244 КБ)

Ответ 209

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 Z стало больше 1, сдвигаем левую границу, пока одна из них не выйдет из окна.

Программа на Python

```
s = open('24.txt').read().strip()
best = 0
left = 0          # начало текущего окна
k = 0             # сколько букв Z в окне
for r in range(len(s)):
    if r > 0 and s[r] == s[r - 1]:
        left, k = r, 0               # две одинаковые соседние: окно начинаем заново
    if s[r] == 'Z':
        k += 1
    while k > 1:                     # лишняя Z — сдвигаем левую границу
        if s[left] == 'Z':
            k -= 1
        left += 1
    best = max(best, r - left + 1)
print(best)
```

Ответ: **209**.

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 3?12?14*5, делящиеся на 21915 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 21915.

Количество строк в таблице для ответа избыточно.

Ответ 381211425 17395
3112214895 142013
3512514285 160279

Перебирать все числа до 10^{10} слишком долго, а кратных 21915 — около 456 тысяч, это быстро. Модуль fnmatch понимает маски с «?» и «*» в том же смысле, что и в условии.

Другой способ — перебрать сами числа, подходящие под маску (вместо «*» — несколько цифр, вместо «?» — любая цифра), и проверить делимость.

Программа на Python

```
from fnmatch import fnmatch

for x in range(21915, 10 ** 10 + 1, 21915):       # перебираем только кратные 21915
    if fnmatch(str(x), '3?12?14*5'):
        print(x, x // 21915)
```

Ответ:
381211425 17395
3112214895 142013
3512514285 160279

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 идентификатор клиентского устройства, с которого на сервер был передан наибольший общий объём данных, а также сумму объёмов (в Кбайт) двух наибольших резервных копий специального раздела, созданных не позднее 11:59:59.

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 идентификатор устройства, с которого был передан наибольший суммарный объём данных, а затем сумму объёмов (в Кбайт) двух наибольших резервных копий специального раздела, выполненных не позднее 11:59:59.

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: 303 220000.

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

Файл к заданию: 26.txt (109 КБ)

Ответ 186 399869

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
vol = {}
backups = []
for _ in range(n):
    t, c, s = f.readline().split()
    s = int(s)
    vol[c] = vol.get(c, 0) + s
    if used + s > k:                # места не хватает: резервная копия
        if t <= '11:59:59':
            backups.append(used)
        used = 0
    used += s
top = max(vol, key=vol.get)
backups.sort(reverse=True)
print(top, backups[0] + backups[1])
```

Ответ: 186 399869.

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 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.

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

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 1511332 28058

Посчитаем для каждой частицы кинетическую энергию 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 * (vx ** 2 + vy ** 2) / 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] == 'VII']
    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]))
q2 = min(p[0] for p in centers)
print(int(q1 * 10000), int(q2 * 10000))
```

Ответ: 1511332 28058.