Дальний Восток. Версия от 18.06, 7:15

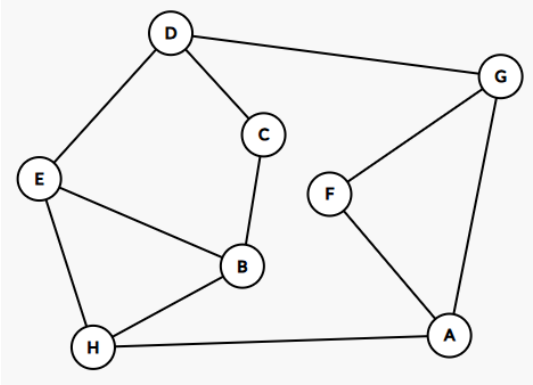
Содержание

Задача №1

Задача 1.1

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

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1		13	2	5				
2	13			74				39
3	2							1
4	5	74					8	
5						21	30	
6					21		53	3
7				8	30	53		
8		39	1			3		



Так как таблицу и схему рисовали независимо друг от друга, то нумерация населённых пунктов в таблице никак не связана с буквенными обозначениями на графе.

Определите, какова сумма протяжённостей дорог из пункта F в пункт G и из пункта B в пункт C. В ответе запишите целое число.

Задача №3

Задача 3.1

В файле приведён фрагмент базы данных «Хозтовары» о поставках бытовой химии и средств гигиены в магазины районов города. База данных состоит из трёх таблиц.

Таблица «Движение товаров» содержит записи о поставках товаров в магазины в течение сентября 2023 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. Заголовок таблицы имеет следующий вид.

ID операции	Дата	ID магазина	Артикул	Количество упаковок, шт.	Тип операции

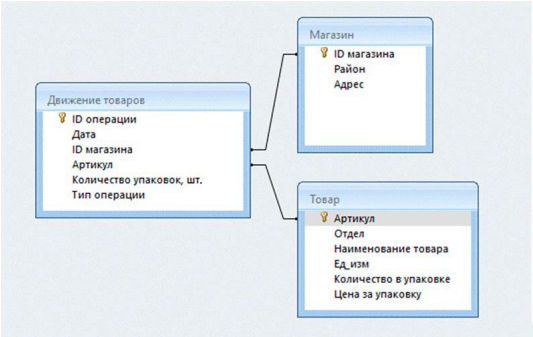
Таблица «Товар» содержит информацию об основных характеристиках каждого товара. Заголовок таблицы имеет следующий вид.

Артикул	Отдел	Наименование товара	Ед. изм.	Количество в упаковке	Цена за упаковку

Таблица «Магазин» содержит информацию о местонахождении магазинов. Заголовок таблицы имеет следующий вид.

ID магазина	Район	Адрес

На рисунке приведена схема указанной базы данных.



Используя информацию из приведённой базы данных, определите общую стоимость (в руб.) упаковок ультрапастеризованного молока (всех видов), полученных магазинами Нагорного района за период со 2 по 9 октября включительно. В ответе запишите только число.

Задача №4

Задача 4.1

Для кодирования последовательности, состоящей из букв К, О, Л, Б, решили использовать неравномерный двоичный код, удовлетворяющий условию Фано. Букве К соответствует двоичный код 0, букве О – код 10.

Какова наименьшая суммарная длина кодовых слов для всех букв в слове КОЛОБОК?

Решение

Код для буквы К по условию равен 0. Следовательно, все остальные коды обязаны начинаться с единицы. Код для буквы О равен 10, поэтому будем рассматривать ветку, начинающуюся с 11.

Нам необходимо закодировать 2 дополнительные буквы — Л и Б, поэтому «раздвоим» единственную оставшуюся ветку и получим коды: 110 и 111. Поскольку других вариантов нет, они являются минимальными по длине, благодаря чему суммарная длина кодовых слов для всех букв в слове КОЛОБОК будет наименьшей.

Посчитаем, сколько раз каждая буква встречается в слове КОЛОБОК: К — 2 раза, О — 3 раза, Л — 1 раз, Б — 1 раз. Остаётся найти общую длину: 2 * 1 + 3 * 2 + 1 * 3 + 1 * 3 = 14

Ответ: 14

Задача №5

Задача 5.1

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

1. Строится двоичная запись числа N.
2. К этой записи дописываются справа ещё два разряда по следующему правилу:

– складываются все цифры двоичной записи числа N, и остаток от деления суммы на 2 дописывается в конец числа (справа);

– над полученной записью производятся те же действия – справа дописывается остаток от деления суммы её цифр на 2.

Полученная таким образом запись является двоичной записью искомого числа R. Укажите наименьшее число N, для которого результат работы алгоритма больше числа 253.

В ответе запишите это число в десятичной системе счисления.

Решение

```
for n in range(1, 1000):
    s = bin(n)[2:]
    s += str(s.count('1') % 2)
    s += str(s.count('1') % 2)
    r = int(s, 2)
    if r > 253:
        print(n)
        break
```

Ответ: 64

Задача №6

Задача 6.1

Исполнитель Черепаха передвигается по плоскости и оставляет след в виде линии. Черепаха может выполнять три команды: **Вперёд n** (*n* – число), **Направо m** (*m* – число) и **Налево m** (*m* – число). По команде **Вперёд n** Черепаха перемещается вперёд на *n* единиц. По команде **Направо m** Черепаха поворачивается на месте на *m* градусов по часовой стрелке, при этом соответственно меняется направление дальнейшего движения. По команде **Налево m** Черепаха поворачивается на месте на *m* градусов против часовой стрелки, при этом соответственно меняется направление дальнейшего движения.

В начальный момент Черепаха находится в начале координат и направлена вверх (вдоль положительного направления оси ординат).

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

Черепаха выполнила следующую программу:

Направо 45 Повтори 7 [Вперёд 5 Направо 45 Вперёд 10 Направо 135].

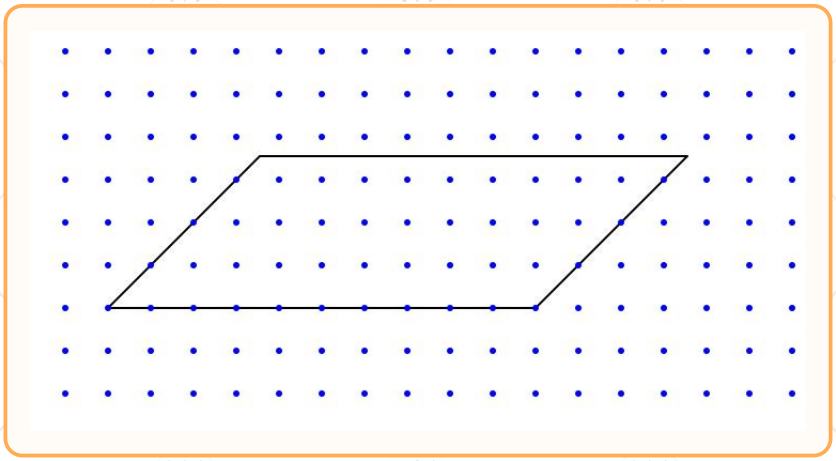
Определите, сколько точек с целочисленными координатами будут находиться внутри области, которая ограничена линией, заданной алгоритмом. Точки на линии учитывать не следует.

Решение

```
from turtle import * # Модуль для работы с Черепахой

tracer(50)  # Ускорение анимации движения черепахи
scale = 20  # Масштаб для увеличения видимости рисунка
# Начальная ориентация Черепахи
# (по умолчанию - вдоль оси X, поворачиваем на 90° влево)
left(90)

# Основной алгоритм Черепахи
right(45)  # Направо 45

for _ in range(7):  # Повторить 7 раз
    forward(5 * scale)   # Вперёд 5 (умножаем на scale для масштабирования)
    right(45)            # Направо 45
    forward(10 * scale)  # Вперёд 10
    right(135)           # Направо 135

# Расставляем точки с целыми координатами
up()  # Поднимаем хвост чтобы не рисовать лишние линии
for x in range(-30, 30):  # Перебор абсцисс точек
    for y in range(-30, 30):  # Перебор ординат точек
```

```
        goto(x * scale, y * scale)  # Перемещение к точке (x, y)
        dot(3, "blue")  # Ставим точку синего цвета размера 3

update()  # Обновление экрана с конечным рисунком от черепахи
done() # Завершение работы (окно остаётся открытым)
```

Остается посчитать количество точек с целочисленными координатами, находящихся внутри фигуры, не считая точки на линии.



Ответ: 27

Задача 6.2

Исполнитель Черепаха передвигается по плоскости и оставляет след в виде линии. 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.

В начальный момент Черепаха находится в начале координат и направлена вверх (вдоль положительного направления оси ординат).

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

Черепаха выполнила следующую программу:

Повтори 2 [Вперёд 14 Налево 278 Назад 12 Направо 98]
Поднять хвост
Вперёд 9 Направо 90 Назад 7 Налево 90
Опустить хвост
Повтори 2 [Вперёд 13 Направо 90 Вперёд 6 Направо 98]

Определите, сколько точек с целочисленными координатами находятся внутри пересечения фигур, ограниченного заданными алгоритмом линиями, включая точки на линиях.

Задача №7

Задача 7.1

Лена записывает голосовое сообщение для своей подруги. Перед отправкой сообщение оцифровывается в формате стерео с частотой дискретизации 28000 Гц и глубиной кодирования 8 бит.

Определите **наименьшее целое количество** Кбайт, необходимое для сохранения сообщения в памяти (без учёта заголовка), если его длительность – 2 минуты 20 секунд.

В ответе укажите только число.

Решение

Для нахождения объема аудиофайла используется формула: Объем = Количество каналов * Частота дискретизации * Глубина кодирования * Время

Определим параметры записи по условию: 1. Формат стерео означает, что запись двухканальная (количество каналов = 2). 2. Частота дискретизации = 28000 Гц. 3. Глубина кодирования = 8 бит (это ровно 1 байт). 4. Длительность записи = 2 минуты 20 секунд. Переведем время в секунды: 2 * 60 + 20 = 140 секунд.

Найдем объем аудиозаписи в байтах: Объем = 2 * 28000 * 1 байт * 140 = 56000 * 140 = 7 840 000 байт.

Переведем полученный объем в килобайты, разделив значение на 1024: 7 840 000 / 1024 = 7656.25 Кбайт.

По условию требуется найти наименьшее целое количество Кбайт, необходимое для сохранения файла в памяти. Если округлить результат в меньшую сторону до 7656 Кбайт, то оставшаяся часть файла (0.25 Кбайт) не поместится и сообщение сохранится не полностью. Для сохранения всего файла округляем значение в большую сторону до ближайшего целого: 7656.25 Кбайт округляем до 7657 Кбайт.

Ответ: 7657

Задача №8

Задача 8.1

Определите количество пятизначных чисел, записанных в девятеричной системе счисления, в записи которых ровно две цифры 3, и при этом никакая нечётная цифра не стоит рядом с цифрой 2.

Решение

Решение программой через itertools:

```
from itertools import *
c = 0
for i in product('012345678', repeat=5):
    s = ''.join(i)
    t = all(j+'2' not in s and '2'+j not in s for j in '1357')
    if s[0] != '0' and s.count('3') == 2 and t:
        c += 1
print(c)
```

Решение программой через циклы:

```
alf = '012345678'
pos1 = '12345678'
c = 0
for i in pos1:
    for j in alf:
        for k in alf:
            for l in alf:
                for m in alf:
                    s = i+j+k+l+m
                    t = all(j + '2' not in s and '2' + j not in s for j in '1357')
                    if s.count('3') == 2 and t:
                        c += 1
print(c)
```

Ответ: 3352

Задача №9

Задача 9.1

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

– одно число повторяется 3 раза, другое 2 раза, остальные различны;

– максимальное из повторяющихся меньше наибольшего из неповторяющихся.

Решение

Решение в электронных таблицах:

Для начала откроем файл электронной таблицы. Далее посчитаем, сколько раз каждое число встречается в строке. Для этого в ячейку H1 запишем следующую формулу и растянем её до ячейки N1 вправо и вниз до конца таблицы:

=СЧЁТЕСЛИ($A1:$G1;A1)

Теперь выделим повторяющиеся числа. В ячейку O1 запишем формулу и растянем вправо до U1 и вниз до конца таблицы. Если число повторяется (частота > 1), записываем само число, иначе 0:

=ЕСЛИ(H1>1;A1;0)

Аналогично выделим неповторяющиеся числа. В ячейку V1 запишем формулу и растянем вправо до AB1 и вниз до конца таблицы. Если число встречается ровно 1 раз, записываем его, иначе 0:

=ЕСЛИ(H1=1;A1;0)

В столбце AC проверим первое условие: одно число повторяется три раза, другое - два раза, остальные два числа различны. В ячейку AC1 запишем формулу и растянем её вниз до конца таблицы:

=ЕСЛИ(И(СЧЁТЕСЛИ(H1:N1;3)=3;СЧЁТЕСЛИ(H1:N1;2)=2;СЧЁТЕСЛИ(H1:N1;1)=2);1;0)

В столбце AD проверим второе условие: максимальное из повторяющихся чисел меньше наибольшего из неповторяющихся. В ячейку AD1 запишем формулу и растянем её вниз до конца таблицы:

=ЕСЛИ(МАКС(O1:U1)<МАКС(V1:AB1);1;0)

В столбце AE проверим выполнение обоих условий. В ячейку AE1 запишем формулу и растянем вниз до конца таблицы:

=AC1*AD1

Выделяем (суммируем) столбец AE и получаем ответ

Решение программой:

Будем построчно считывать числа из .csv файла, числа разделены точкой с запятой. Для каждой строки собираем числа по частоте: встречающиеся 3 раза, 2 раза и 1 раз. Проверяем первое условие: длины списков равны 3, 2 и 2 соответственно. Для второго условия проверяем, что максимум среди повторяющихся чисел меньше максимума среди неповторяющихся.

```
# Открываем файл таблицы
file = open("9.csv")

# Счётчик подходящих строк
count = 0

# Перебираем строки таблицы
for line in file:
    # Считываем числа из строки
    numbers = list(map(int, line.split(";")))
    # Собираем числа, встречающиеся 3 раза
    rep3 = [x for x in numbers if numbers.count(x) == 3]
    # Собираем числа, встречающиеся 2 раза
    rep2 = [x for x in numbers if numbers.count(x) == 2]
    # Собираем числа, встречающиеся 1 раз
    rep1 = [x for x in numbers if numbers.count(x) == 1]
    # Условие 1: одно число x3, одно x2, два x1
    if len(rep3) == 3 and len(rep2) == 2 and len(rep1) == 2:
        # Условие 2: max повторяющихся < max неповторяющихся
        if max(rep3 + rep2) < max(rep1):
            count += 1

# Выводим количество подходящих строк
print(count)
```

Задача №11

Задача 11.1

На предприятии каждой изготовленной детали присваивают серийный номер из 225 символов. Для хранения каждого номера отведено одинаковое и минимально возможное число байт; используется посимвольное кодирование – все символы кодируются одинаковым и минимально возможным числом бит. Известно, что для хранения 1270 серийных номеров отведено не более 104 Кбайт памяти.

Определите максимально возможную мощность алфавита, используемого для записи серийных номеров.

Решение

Переводим общий объём в байты:

$$104 \text{ Кбайт } = 104 \cdot 1024 = 106496 \text{ байт}$$

На один номер отведено одинаковое число байт. Максимум байт на номер:

$$\frac{106496}{1270} = 83,85 \approx 83 \text{ байта } = 664 \text{ бита}$$

Округляем в меньшую сторону, иначе превысим допустимый объем.

Номер из 225 символов по i бит занимает $225 \cdot i$ бит, при этом он не должен превышать 664 бит.

$$i = \frac{664}{225} \approx 2,95 = 2 \text{ бита}$$

Округляем в меньшую сторону, иначе превысим допустимый объем.

Максимальная мощность алфавита при i битах на символ 2^i:

$$2^2 = 4$$

Ответ: 4

Задача №12

Задача 12.1

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 (множество символов $\boldsymbol{A = \{a_0, a_1, \ldots, a_{n-1}\}}$), включая специальный пустой символ $\boldsymbol{a_0}$. Время работы исполнителя делится на дискретные такты (шаги). На каждом такте головка МТ находится в одном из множества допустимых состояний $\boldsymbol{Q = \{q_0, q_1, \ldots, q_{n-1}\}}$. В начальный момент времени головка находится в начальном состоянии $\boldsymbol{q_0}$.

На каждом такте головка обозревает одну ячейку ленты, называемую текущей ячейкой. За один такт головка исполнителя может переместиться в ячейку справа или слева от текущей, не меняя находящийся в ней символ, или заменить символ в текущей ячейке без сдвига в соседнюю ячейку. После каждого такта головка переходит в новое состояние или остаётся в прежнем состоянии.

Программа работы исполнителя МТ задаётся в табличном виде.

	a_0	a_1	...	a_{n-1}
q_0	команда	команда	...	команда
q_1	команда	команда	...	команда
...	команда	команда	...	команда
q_{n-1}	команда	команда	...	команда

В первой строке перечислены все возможные символы в текущей ячейке ленты, в первом столбце – возможные состояния головки. На пересечении i-й строки и j-го столбца находится команда, которую выполняет МТ, когда головка обозревает j-й символ, находясь в i-м состоянии. Если пара «символ – состояние» невозможна, то клетка для команды остаётся пустой. Каждая команда состоит из трёх элементов, разделённых запятыми: первый элемент – записываемый в текущую ячейку символ алфавита (может совпадать с тем, который там уже записан). Второй элемент – один из четырёх символов «L», «R», «N», «S». Символы «L» и «R» означают сдвиг в левую или правую ячейки соответственно, «N» – отсутствие сдвига, «S» – завершение работы исполнителя МТ после выполнения текущей команды. Сдвиг происходит после записи символа в текущую ячейку. Третий элемент – новое состояние головки после выполнения команды. Например, команда 0, L, q_3 выполняется следующим образом: в текущую ячейку записывается символ «0», затем головка сдвигается в соседнюю слева ячейку и переходит в состояние q_3.

Приведём пример выполнения программы, заданной таблично.

На ленте записано неизвестное ненулевое количество расположенных подряд в соседних ячейках символов «Z», все остальные ячейки ленты заполнены пустым символом «λ». В начальный момент времени головка находится на неизвестном ненулевом расстоянии справа от самого правого символа «Z».

Программа

	λ	Z
q_0	λ, L, q_0	X, L, q_1
q_1	λ, S, q_1	X, L, q_1

заменяет на ленте все символы «Z» на «X» и останавливает исполнителя в первой ячейке слева от последовательности символов «X».

Возможное начальное состояние исполнителя:

...	λ	λ	Z	Z	Z	Z	λ	λ	...
								▲q_0	

Конечное состояние исполнителя после завершения выполнения программы:

...	λ	λ	X	X	X	X	λ	λ	...
		▲q_1							

Выполните задание.

На ленте в соседних ячейках записано двоичное представление числа 2026 без ведущих нулей. Ячейки справа и слева от последовательности заполнены пустыми символами «λ». В начальный момент времени головка расположена в ближайшей слева от последовательности ячейке.

Программа работы исполнителя:

	λ	0	1
q_0	λ, R, q_1		
q_1	$1, R, q_2$	O, R, q_1	$1, R, q_1$
q_2	λ, R, q_3		
q_3	λ, S, q_3		

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

Решение

Решение руками:

Исполнитель идет слева направо, не производя замен символов исходной последовательности. При этом символ λ справа от последовательности заменяется на 1.

Исходная последовательность: 11111101010

Добавляем 1 справа и находим конечное число:

$$111111010101_2 = 4053_{10}$$

Решение Python:

```
a = {
    'q0.': ['.', 'R', 'q1'],
    'q10': ['0', 'R', 'q1'],
    'q11': ['1', 'R', 'q1'],
    'q1.': ['1', 'R', 'q2'],
    'q2.': ['.', 'R', 'q3'],
    'q3.': ['.', 'S', 'q3'],
}

# двоичная запись числа 2026
s1 = bin(2026)[2:]
# лента с пустыми символами по краям
s = list('...' + s1 + '...')

# головка стоит на ближайшей слева пустой ячейке
i, c, p = 2, 0, 'q0'

while c != 'S':
    s[i], c, p = a[p + s[i]]

    if c == 'R':
        i += 1

print(int(''.join(s).replace('.', ''), 2))
```

Ответ: 4053

Задача №13

Задача 13.1

Сеть задана IP-адресом одного из входящих в неё узлов 102.162.200.51 и сетевой маской 255.255.255.0.

Найдите в данной сети наибольший IP-адрес, который может быть назначен компьютеру. 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.

Решение

Решение руками:

По условию задачи IP-адрес узла равен 102.162.200.51, а маска сети - 255.255.255.0. Сетевая маска 255.255.255.0 означает, что первые три байта IP-адреса относятся к адресу сети, а последний байт отвечает за номер узла в этой сети.

Таким образом, адрес сети равен 102.162.200.0.

Диапазон всех адресов в этой сети - от 102.162.200.0 до 102.162.200.255.

Адрес сети (102.162.200.0) и широковещательный адрес (102.162.200.255) не могут быть назначены компьютерам.

Следовательно, наибольший IP-адрес, который может быть назначен компьютеру, равен 102.162.200.254.

Найдем сумму числовых значений октетов этого адреса: $102 + 162 + 200 + 254 = 718$.

Решение программой:

Полученная сеть содержит список всех адресов от адреса сети (индекс 0) до широковещательного адреса (последний индекс, или -1). Максимальный доступный для компьютера адрес (последний хост) будет находиться по индексу -2.

Далее преобразуем этот адрес в строку, разобьем по символу точки на отдельные октеты, приведем их к целочисленному типу и найдем сумму.

```
from ipaddress import ip_network

# Создаем сеть по адресу узла и маске
net = ip_network('102.162.200.51/255.255.255.0', strict=0)

# Получаем наибольший IP-адрес, доступный для компьютера
max_ip = net[-2]

# Преобразуем IP-адрес в список целых чисел (октетов) и находим их сумму
octets_sum = sum(map(int, str(max_ip).split('.')))

print(octets_sum)
```

Ответ: 718

Задача 13.2

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. Обычно маска записывается по тем же правилам, что и IP-адрес, – в виде 4 байтов, причем каждый байт записывается в виде десятичного числа. Адрес сети получается в результате применения поразрядной конъюнкции к заданному IP-адресу узла и его маске.

Например, если IP-адрес узла равен 231.32.255.131, а маска равна 255.255.240.0, то адрес сети равен 231.32.240.0.

Узел имеет IP-адрес 210.185.140.126. Маска сети равна 255.255.255.252. Определите наименьший возможный IP-адрес в этой сети (адрес сети) и в ответе запишите сумму значений его октетов.

Решение

Решение руками:

Наименьшим возможным IP-адресом в сети является адрес самой сети. Чтобы его найти, применим поразрядную конъюнкцию (логическое «И») к IP-адресу узла и маске.

Так как первые три байта маски равны 255, первые три байта адреса сети полностью совпадают с IP-адресом: 210.185.140.

Для четвертого байта выполним операцию «И» в двоичном виде:

126 в двоичной системе: 01111110

252 в двоичной системе: 11111100

Перемножим разряды:

$$01111110\&11111100 = 01111100$$

Полученное двоичное число 01111100 в десятичной системе равно 124. Таким образом, адрес сети равен 210.185.140.124.

Сложим значения его октетов:

$$210 + 185 + 140 + 124 = 659$$

Решение Python:

Параметр strict= 0 позволяет использовать адрес узла. Адрес сети (наименьший IP) хранится в свойстве network_address. Преобразуем его в строку, делим по точкам на части, переводим их в целые числа и складываем.

```
from ipaddress import ip_network

# Создаем объект сети
net = ip_network('210.185.140.126/255.255.255.252', strict=0)

# Получаем адрес сети в виде строки
net_addr = str(net.network_address)

# Разбиваем строку по точкам, переводим части в числа и суммируем их
```

```
ans = sum(int(x) for x in net_addr.split('.'))
print(ans)
```

Ответ: 659

Задача №14

Задача 14.1

Операнды арифметического выражения записаны в системе счисления с основанием 22.

$$63x89875_{22} + 17x51_{22} + 75x3_{22}$$

В записи чисел переменной x обозначена неизвестная цифра из алфавита 22-ричной системы счисления. Определите наибольшее значение x, при котором значение данного арифметического выражения кратно 21. Для найденного значения x вычислите частное от деления значения арифметического выражения на 21 и укажите его в ответе в десятичной системе счисления. Основание системы счисления указывать не нужно.

Решение

```
alf = '0123456789abcdefghijkl'

for x in alf:
    n = int(f'63{x}89875', 22) + int(f'17{x}51', 22) + int(f'75{x}3', 22)
    if n % 21 == 0:
        print(x, n//21)
```

Ответ: 733155579

Задача 14.2

Значение арифметического выражения

$$4 \cdot 16^{25} + 2 \cdot 8^{30} - 64^{10}$$

записали в двоичной системе счисления. Сколько цифр 0 содержится в этой записи?

Решение

Для решения задачи мы вычисляем значение арифметического выражения, преобразуем его в двоичную строку с помощью встроенной функции bin() (отсекая технический префикс 0b с помощью среза [2:]) и подсчитываем количество нулей методом .count('0').

```
num = 4 * 16**25 + 2 * 8**30 - 64**10
# Переводим число в двоичный вид и убираем служебный префикс '0b'
binary_str = bin(num)[2:]
# Считаем количество нулей в полученной строке
zeros = binary_str.count('0')
print(zeros)
```

Ответ: 71

Задача №15

Задача 15.1

Обозначим через ДЕЛ(n, m) утверждение «натуральное число n делится без остатка на натуральное число m». Задан отрезок $B = [15; 30]$. Для какого наибольшего натурального числа A формула

$$\text{ДЕЛ}(x, A) \vee (\text{ДЕЛ}(x, 23) \to \neg(x \in B))$$

тождественно истинна (т.е. принимает значение 1) при любом натуральном значении переменной x?

Решение

Решение руками:

Для того чтобы формула была тождественно истинна, необходимо, чтобы при ложности правой части выражения:

$$(\text{ДЕЛ}(x, 23) \to \neg(x \in B)) = 0$$

левая часть выражения обязательно была истинной:

$$\text{ДЕЛ}(x, A) = 1 \text{ (то есть x делится на A)}$$

Логическое следование (импликация) $P \to Q$ ложно только в одном случае: когда P истинно, а Q ложно. В нашем случае импликация $\text{ДЕЛ}(x, 23) \to \neg(x \in B)$ принимает значение 0, только когда одновременно:

ДЕЛ(x, 23) = 1 (число x делится на 23);

$\neg(x \in B) = 0$, что означает $x \in B$ (число x принадлежит отрезку [15; 30]).

Найдем все такие натуральные числа x, которые одновременно делятся на 23 и лежат на отрезке [15; 30]. На данном отрезке существует только одно число, кратное 23 – это само число 23.

Таким образом, единственным значением переменной, при котором правая часть формулы становится ложной, является $x = 23$. При всех остальных x правая часть истинна.

Чтобы вся формула оставалась истинной и при $x = 23$, левая часть ДЕЛ(23, A) обязана быть истинной. То есть число 23 должно делиться на A без остатка.

Нам требуется найти наибольшее натуральное число A. Так как 23 – простое число, его наибольшим натуральным делителем является само число 23.

Решение Python:

Если хотя бы для одного x выражение ложно, то это A нам не подходит. В конце работы программы последним выведенным числом будет наибольшее подходящее A.

```
for a in range(1, 1000):
    # флаг: 0 - условие выполняется для всех x, 1 - хотя бы один случай нарушает
    c = 0
    for x in range(1, 1000):
        if ((x % a == 0) or ((x % 23 == 0) <= (not (15 <= x <= 30)))) == False:
            c = 1 # если выражение ложно, ставим флаг
```

```
            break
    # если флаг остался 0, выражение истинно для всех x
    if c == 0:
        print(a) # выводим A
```

Ответ: 23

Задача №16

Задача 16.1

Алгоритм вычисления значения функции $F(n)$, где n — целое неотрицательное число, задан следующими соотношениями:
$F(n) = n$, если $n < 10$;
$F(n) = n + F(n-3)$, если $n \geq 10$;
Определите значение $F(2566)/F(2557)$. В ответе запишите целую часть.

Решение

```
from functools import *

@lru_cache(None)
def f(n):
    if n < 10:
        return n
    if n >= 10:
        return n + f(n-3)

# Заполняем кэш от наименьшего к наибольшему,
# поскольку при n < 10 функция вычисляется нерекурсивно
for i in range(2558):
    f(i)
print(f(2566)/f(2557))
```

Ответ: 1

Задача 16.2

Алгоритм вычисления значения функции $F(n)$, где n — натуральное число, задан следующими соотношениями:
$F(n) = 1$, если $n = 1$;
$F(n) = n \cdot F(n-1)$, если $n > 1$.
Определите значение $(F(3038) + 3 \cdot F(3037))/F(3036)$.

Задача 16.3

Алгоритм вычисления значения функции $F(n)$, где n — целое неотрицательное число, задан следующими соотношениями:
$F(n) = n$, , если $n \leq 2$;
$F(n) = F(n-2) + 3 \cdot n$, если $n > 2$ и нечетно;

$F(n) = (n+5) \cdot F(n-1) \cdot n$, если $n > 2$ и четно;
Определите значение $F(2026)/F(2025)$.

Решение

Для решения необходимо создать рекурсивную функцию, которая будет повторять логику, описанную в условии. Однако, чтобы не вызвать проблем с глубиной рекурсии, стоит добавить декоратор `lru_cache`, сохраняющий полученные значения функции, а затем "прогреть"кэш, просчитав результаты выполнения функции для числа n, начиная с базового случая (то есть от $n \leq 2$) и заканчивая 2026 (ближайшим числом, требуемым программой для получения ответа). В конце значение $f(2026)/f(2025)$ выводится на экран.

```
from functools import lru_cache  # импортируем декоратор lru_cache из functools


@lru_cache(None)  # Добавляем декоратор lru_cache, чтобы записывать полученные
# значения функции
def f(n):  # создаём рекурсивную функцию, повторяющую формулы из условия
    if n <= 2:
        return n
    if n % 2 != 0 and n > 2:
        return f(n - 2) + 3 * n
    if n % 2 == 0 and n > 2:
        return (n + 5) * f(n - 1)


# "Прогреваем" кэш, просчитывая значения функции для n от 1 до 2026
for n in range(1, 2027):
    f(n)
print(f(2026) / f(2025))  # выводим ответ на экран
```

Ответ: 2031

Задача №17

Задача 17.1

В файле содержится последовательность целых чисел. Определите количество пар идущих подряд элементов, в которых элементы не равны друг другу, а абсолютная разность между ними кратна минимальному положительному элементу последовательности, который кратен 9. В ответе запишите два числа: сначала количество найденных пар, затем максимальную сумму элементов таких пар.

Решение

```
f = open('1')
a = [int(x) for x in f]
# 1)Число положительно, если оно больше нуля
# 2)Число кратно 9, если остаток при делении на 9 равен 0
minim9 = min([x for x in a if x>0 and x%9==0])
c = 0
mx = -10**10
for i in range(len(a)-1):
    # 1)Элементы пары различны
    # 2)Абсолютная разность элементов кратна minim9
    if a[i] != a[i+1] and abs(a[i] - a[i+1]) % minim9 == 0:
        c += 1
        mx = max(mx,a[i]+a[i+1])
print(c,mx)
```

Ответ:

Задача №18

Задача 18.1

Исполнитель Робот стоит в левом верхнем углу поля, разлинованного на клетки. Он 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. 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.

Исходные данные записаны в файле в виде электронной таблицы, каждая ячейка которой соответствует клетке поля. В ответе запишите два числа – сначала максимальную сумму, которую может собрать Робот, затем – минимальную.

Задача 18.2

Исполнитель Робот может перемещаться по клеткам квадратного поля размером $N \times N$, заполненного числами. За один ход Робот может переместиться на одну клетку вправо или вниз.

Маршрут Робота начинается в левой верхней клетке и должен обязательно завершиться в одной из нескольких финишных клеток, расположенных в самом нижнем ряду таблицы, номера столбцов которых делятся на 3.

Определите максимальную и минимальную денежную сумму, которую может собрать Робот, пройдя по такому маршруту. В ответе укажите сначала максимальное значение, затем минимальное через пробел.

Задача 18.3

Исполнитель Робот может перемещаться по клеткам квадратного поля размером $N \times N$, заполненного числами. За один ход Робот может переместиться на одну клетку вправо или вниз.

В «угловых» клетках поля – тех, которые справа и снизу ограничены стенами, Робот не может продолжать движение, поэтому накопленная сумма считается итоговой. Итоговой, может считаться только сумма, кратное 3, если в уголовой клетке число не кратное 3, оно

не рассматривается. 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.

Определите максимальную и минимальную денежную сумму, которую может собрать Робот, пройдя по такому маршруту. В ответе укажите сначала максимальное значение, затем минимальное через пробел.

Решение

Открываем файл в редакторе электронных таблиц. Копируем исходную таблицу и вставляем ниже только с учётом форматов, чтобы сохранить стены.

Робот стартует из левой верхней клетки. Поэтому в ячейку A22 вставляем значение из клетки A1.

Перейдём к заполнению клеток первой строки. Для этого в ячейку B22 вставляем формулу и растягиваем её вправо конца строки:

$$=A22+B1$$

Перейдём к заполнению клеток первого столбца. Для этого в ячейку A23 вставляем формулу и растягиваем её вниз до конца столбца:

$$=A22+A2$$

Остается заполнить остальную часть поля. Для этого в ячейку B23 вставляем формулу и растягиваем её на остаток поля:

$$=МАКС(A23;B22)+B2$$

После этого у нас пропали все стены, поэтому снова копируем исходную таблицу и вставляем поверх нашей только с учётом форматов. Очищаем ячейки, в которые Робот вообще не может попасть и обозначаем «угловые» клетки. Также сменим формулы около стен. В те клетки, которые находятся правее от стены, запишем формулу из первого столбца таблицы. В клетках, которые находятся под стеной, нужно записать формулу из первой строки.

Максимальное значение будет равно максимуму, кратному 3, из «угловых» клеток.

Для поиска минимального значения с помощью сочетания клавиш Ctrl+H заменяем МАКС на МИН и определяем его как минимальное значение, кратное 3, из «угловых» клеток.

Задача №19-21

Задача 19.1

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) 4 камня;

– увеличить количество камней в одной из куч (по своему выбору) в 2 раза.

Например, пусть в одной куче 20 камней, а в другой 30 камней; такую позицию в игре обозначим (20, 30). Тогда за один ход можно получить любую из четырёх позиций: (24, 30), (20, 34), (40, 30), (20, 60).

Для того чтобы делать ходы, у каждого игрока есть неограниченное количество камней. Игра завершается в тот момент, когда суммарное количество камней в двух кучах становится не менее 154. Победителем считается игрок, сделавший последний ход, то есть первым получивший такую игровую позицию, при которой в двух кучах суммарно 154 камня или больше. В начальный момент в первой куче 11 камней, во второй куче – S камней; $1 \leq S \leq 142$.

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

Известно, что Ваня выиграл своим первым ходом после неудачного хода Пети. Укажите минимальное значение S, при котором это возможно.

Решение

Решение руками:

Чтобы Ваня победил своим первым ходом, Петя должен совершить такой неудачный ход, после которого Ваня следующим же действием сможет набрать в сумме не менее 154 камней. Быстрее всего увеличить кучу можно с помощью удваивания. Сначала Петя удваивает вторую кучу S, а затем Ваня удваивает полученную кучу, в то время как первая куча остается без изменений (11 камней). При каких S будет выполняться неравенство 2 * 2 * S + 11 >= 154? 4 * S + 11 >= 154 4 * S >= 143 S >= 35.75 Минимальное целое значение S, удовлетворяющее этому условию, равно 36. Следовательно, ответ 36.

Решение программой:

Для решения на Python используем рекурсию с проверкой всех возможных ходов (+ 4, * 2), чтобы определить, чья это выигрышная позиция.

Функция возвращает 0, если игра уже завершена (камней в сумме $\geq$ 154), положительное число — если при оптимальной игре побеждает текущий игрок, и отрицательное — если побеждает соперник.

Так как первый ход делает Петя, цикл запускается от его лица: если значение функции положительное, выигрывает Петя, а если отрицательное — Ваня.

Если функция вернула 1, то Петя победил первым ходом. Но он походил неудачно и передал победу Ване.

При переборе возможных значений программа выводит минимум 36.

```
from functools import lru_cache

@lru_cache(None)
def game(first_heap, second_heap):
    if first_heap + second_heap >= 154:  # Если камней в куче стало больше 154
        return 0  # Прекращаем игру
    moves = [
        game(first_heap + 4, second_heap),
        game(first_heap, second_heap + 4),
        game(first_heap * 2, second_heap),
        game(first_heap, second_heap * 2),
    ]  # Генерация всех возможных ходов
    petya_win = [i for i in moves if i <= 0]
    if petya_win:
        return -max(petya_win) + 1
    else:
        return -max(moves)

for s in range(1, 143):
    # Если в данной позиции после неудачного хода Пети возможен выигрыш Вани
    if game(11 + 4, s) == 1 or game(11, s + 4) == 1 \
        or game(11 * 2, s) == 1 or game(11, s * 2) == 1:
        print(s)  # Вывод минимального значения S
        break
```

Ответ: 36

Задача 20.1

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 ходов Вани.

Решение

Решение руками

Петя должен выиграть своим вторым ходом независимо от ходов Вани. Это означает, что Петя своим первым ходом должен получить такую позицию, из которой Ваня любым ходом создаст выигрышное положение для Пети на следующем ходу.

Такими позициями для Вани (с которых он вынужден подставиться под выигрыш соперника) являются позиции, сумма которых при удваивании большей кучи лежит в диапазоне от 150 до 153. Тогда любой ход Вани приведет к тому, что Петя сможет набрать в сумме 154 или более камней.

Петя может получить такие позиции своими первыми ходами:

Если Петя удваивает вторую кучу и получает (11, 70), сумма при удваивании Ваней составит 151 (диапазон от 150 до 153). Это возможно при начальном S = 35.

Если Петя удваивает первую кучу и получает (22, 64), сумма при удваивании Ваней составит 150 (диапазон от 150 до 153). Это возможно при начальном S = 64.

Следовательно, два наименьших значения S – это 35 и 64.

Решение БУ (программой):

Для решения на Python используем рекурсию с проверкой всех возможных ходов (+ 4, * 2), чтобы определить, чья это выигрышная позиция.

Функция возвращает 0, если игра уже завершена (камней в сумме $\geq$ 154), положительное число – если при оптимальной игре побеждает текущий игрок, и отрицательное – если побеждает соперник.

Так как первый ход делает Петя, цикл запускается от его лица: если значение функции положительное, выигрывает Петя, а если отрицательное – Ваня.

Если функция вернула 2, то Петя победил вторым ходом.

Программа перебирает и выводит значения S, в ответ берем два наименьших (два первые) числа.

```
from functools import lru_cache

@lru_cache(None)
def game(first_heap, second_heap):
    if first_heap + second_heap >= 154:  # Если камней в куче стало больше 154
        return 0  # Прекращаем игру
    moves = [
        game(first_heap + 4, second_heap),
        game(first_heap, second_heap + 4),
        game(first_heap * 2, second_heap),
        game(first_heap, second_heap * 2),
    ]  # Генерация всех возможных ходов
    petya_win = [i for i in moves if i <= 0]
    if petya_win:
        return -max(petya_win) + 1
    else:
        return -max(moves)

for s in range(1, 143):
    if game(11, s) == 2:
        print(s)  # Вывод минимального значения S
```

Ответ: 35, 64

Задача 21.1

Найдите минимальное значение S, при котором:
– у Вани есть выигрышная стратегия, позволяющая ему выиграть первым или вторым ходом

при любой игре Пети;
– у Вани нет стратегии, гарантирующей выигрыш первым ходом.

Задача №22

Задача 22.1

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 **максимальное** количество процессов, которые начнут выполняться не ранее 15-й секунды (время начала 15 секунд также учитывается).

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.

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

Решение

Для начала откроем файл электронной таблицы и разделим процессы в столбце C при помощи функции "Данные по столбцам"(разделитель — точка с запятой). На месте пустых ячеек с номерами процессов и в самом низу столбца A необходимо вписать 0, дабы функции ВПР() и МАКС() корректно находили искомые данные. Теперь в ячейке F2 запишем и растянем вниз следующую формулу:

$$=\text{ВПР}(C2;\ \$A:\$I;\ 9;\ 0)$$

Необходимо дополнительно протянуть данную формулу от столбца F до H, чтобы найти перенести время выполнения каждого процесса. Далее в ячейке I2 запишем и растянем данную формулу:

$$=\text{МАКС}(F2:H2)\ +\ B2$$

Таким образом мы получим время завершения каждого процесса в таблице. Теперь необходимо посчитать время начала каждого процесса — для этого в ячейке J2 запишем и растянем следующую формулу:

$$= I2 - B2 + 1$$

Остаётся посчитать количество процессов, которые начнутся не ранее, чем с 15 секунды — в ячейке K2 запишем и растянем данную формулу:

$$= ЕСЛИ(J2 >= 15; 1; 0)$$

Выделим столбец J и найдём сумму его значений, получив тем самым ответ.

Задача №23

Задача 23.1

Исполнитель Робот преобразует число, записанное на экране. У исполнителя есть две команды, которым присвоены номера:

1. Прибавь 1;
2. Поменять цифры в разряде единиц и десятков местами, если разряд десятков меньше разряда единиц.

Сколько есть программ, которые преобразуют число 100 в число 150?

Решение

Создаём функцию, которая будет составлять траекторию. x - текущая позиция, y - целевая. Если мы уже превысили конец (x > y), возвращаем 0, путь невозможен. Если дошли до конца, то всё хорошо, возвращаем 1. Если ни в один из этих случаев не зашли, то путь продолжается, делаем ходы. Для трёхзначного числа индекс 0 отвечает за разряд сотен, 1 - за десятки, 2 - за единицы. ВАЖНО: это актуально только для трёхзначного числа! Далее переводим наше текущее число икс в строку, берём срезом разряд десятков и единиц, сравниваем. Если разряд единиц больше, меняем их местами (не забывая про ход + 1), иначе только прибавляем 1

```
def f(x, y):
    if x > y:
        return 0
    if x == y:
        return 1
    # str(x)[0] - первый разряд, сотни
    # str(x)[1] - десятки
    # str(x)[2] - единицы
    if str(x)[1] < str(x)[2]:
        return f(x + 1, y) + f(int(str(x)[0] + str(x)[2] + str(x)[1]), y)
    else:
        return f(x + 1, y)

print(f(100, 150))
```

Ответ: 35

Задача 23.2

Исполнитель преобразует число на экране. У исполнителя есть две команды:

1. Вычти 1
2. Найди целую часть от деления на 2

Сколько существует программ, которые преобразуют исходное число 40 в число 6, при этом траектория вычислений обязательно содержит число 15?

Решение

Создаём функцию, которая будет считать количество траекторий из x в y, где x — текущая позиция, y — целевая. Если программа получила число, меньшее y, то возвращаем 0, ведь дальше получить искомый путь невозможно. Если программа дошла до конца, получив y, то необходимо засчитать данную траекторию, вернув 1. Если ни в один из этих случаев не зашли, то рекурсия продолжается — выполняем доступные команды. В конце, чтобы учесть наличие числа 15 в каждой траектории, необходимо посчитать количество путей из 40 в 15, а затем перемножить его на количество путей из 15 в 6.

```
def f(x, y):  # Задаём рекурсивную функцию для поиска количества траектория
    if x < y:  # Если текущее значение "x" меньше числа, до которого нужно найти,
        return 0  # возвращаем нулевое количество траекторий
    if x == y:  # Если текущее значение "x" достигло искомого числа,
        return 1  # возвращаем единицу, засчитывав тем самым данную траекторию
    # Возвращаем суммарное количество возможных траекторий
    return f(x - 1, y) + f(x // 2, y)


# Находим количество траекторий из 40 в 15 и перемножаем их на количество
# траекторий из 15 в 6, учитывая тем самым обязательное наличие числа 15
# в траектории от 40 до 6
print(f(40, 15) * f(15, 6))
```

Ответ: 60

Задача 23.3

Исполнитель преобразует число на экране. У исполнителя есть две команды, которые обозначены номерами:

1. Прибавь 1
2. Поменяй местами

Первая из этих команд увеличивает число на экране на 1. Вторая команда может применяться только к числу, у которого цифра разряда десятков по значению меньше цифры, стоящей в разряде единиц, и действует, заменяя число на экране числом, в котором цифры двух младших разрядов поменяны местами. Программа для исполнителя – это последовательность команд.

Сколько существует программ, для которых при исходном числе 110 результатом является число 154?

Задача №24

Задача 24.1

Текстовый файл состоит не более чем из 10^6 символов и содержит только десятичные цифры, а также знаки «+» и «*» (сложения и умножения). Определите максимальное количество символов в непрерывной последовательности, являющейся корректным арифметическим выражением с целыми неотрицательными числами (без знака). В этом выражении никакие два знака арифметических операций не стоят рядом. В записи чисел отсутствуют незначащие (ведущие) нули. В ответе укажите количество символов в найденном выражении.

Решение

Для решения данной задачи воспользуемся регулярными выражениями. В начале необходимо записать шаблон числа: `"[1-9][0-9]*|0"`, где `"[1-9]"` — первая цифра, `"[0-9]*"` — остальные цифры в числе, `0"` — "или 0". После этого записываем общий шаблон арифметического выражения: `"([1-9][0-9]*|0)([+*]([1-9][0-9]*|0))+"`, где `"([1-9][0-9]*|0)"` — первое число, `"([+*]([1-9][0-9]*|0))+"` — число со знаком, повторяющееся 1 и более раз. Остаётся перебрать все подстроки при помощи функции `finditer()` и найти длину самого длинного арифметического выражения.

```
from re import *  # импортируем модуль re для работы с регулярными выражениями

s = open("test1.txt").readline()  # открываем файл и считываем всю строку
patters = "([1-9][0-9]*|0)([+*]([1-9][0-9]*|0))+"  # создаём регулярное
# выражение под условие задачи
max_len = 0  # создаём переменную, в которую будет записан наш ответ
for i in finditer(patters, s):  # перебираем все найденные выражения
    max_len = max(max_len, i.group())  # записываем максимальную длину
print(max_len)  # выводим ответ на экран
```

Задача №25

Задача 25.1

Напишите программу, которая перебирает целые числа, большие 7 513 048, в порядке возрастания и ищет среди них числа, представленные в виде произведения ровно двух простых множителей, не обязательно различных, каждый из которых содержит в своей записи хотя бы одну цифру 1 и одну цифру 6.

В ответе для первых 5 найденных чисел запишите само число и наибольший из его простых множителей в соответствующие столбцы таблицы.

Решение

```
def divisors(n):
    # функция для поиска нетривиальных делителей
    divs = set()  # заводим множество делителей
    for i in range(2, int(n ** 0.5) + 1):
        # если i - делитель
        if n % i == 0:
            # добавляем i и его парный делитель
            divs.add(i)
            divs.add(n // i)
    return divs


def prime(x):
    # функция для проверки на простоту
    for j in range(2, int(x ** 0.5) + 1):
        # если нашли хотя бы один делитель
        # кроме 1 и себя - число не простое
        if x % j == 0:
            return False
    # не забываем, что 1 - не простое число
    return x > 1


cnt = 0  # счётчик выведенных чисел
for x in range(7_513_048 + 1, 10_000_000):
    d = divisors(x)
    # если получили 1 повторяющийся делитель или два
    if (len(d) == 1 or len(d) == 2) and all(prime(x) for x in d):
        # проверяем, что все делители содержат 1 и 6
        if all('1' in str(x) and '6' in str(x) for x in d):
            print(x, max(d))
```

```
        cnt += 1
        if cnt == 5:
            # вывели 5 чисел - остановили цикл
            break
```

Сначала вводится функция `divisors(n)`, которая находит все нетривиальные делители числа, то есть все, кроме 1 и самого числа.

Дополнительно вводится функция `prime(x)`, которая проверяет число на простоту. Она перебирает возможные делители от 2 до корня: если находится хотя бы один делитель, число нам не подходит, возвращаем ложь.

Далее перебираются все числа x, большие 7 513 048. Проверяем, что нашлось 1 или 2 делителя: `len(d) == 1 or len(d) == 2`. Затем проверяем, что оба делителя простые: `all(prime(x) for x in d)`.

После этого проверяется второе условие: у каждого найденного делителя в записи должно быть хотя бы по одной цифре 1 и 6.

Как только таких чисел становится 5, перебор завершается, ответ получен.

Ответ: 35

Задача №26

Задача 26.1

Входной файл содержит информацию о заявках граждан, обращающихся во многофункциональный центр (МФЦ) в течение календарных суток. В заявке указаны время начала и время окончания приёма специалистом (в минутах от начала суток). Рабочие места специалистов МФЦ (окна) пронумерованы натуральными числами начиная с 1. Приём одного гражданина ведёт свободный специалист в окне с минимальным номером. Новый посетитель может обратиться к освободившемуся специалисту, начиная со следующей минуты после завершения приёма предыдущего. Если в момент обращения в МФЦ свободных специалистов нет, то гражданин уходит. Определите, сколько граждан сможет попасть на приём в МФЦ в течение 24 часов, и каков номер окна специалиста, который начнёт принимать посетителя последним. Если таких окон несколько, укажите наименьший номер окна.

Входные данные представлены в файле следующим образом. Первая строка входного файла содержит натуральное число K, не превышающее 1000, – количество окон в МФЦ. Во второй строке записано натуральное число N ($N \leq 10\ 000$), обозначающее количество граждан. Каждая из следующих N строк содержит два натуральных числа, каждое из которых не превышает 1440: указанные в заявке время начала и время окончания приёма (в минутах от начала суток).

Запишите в ответе два числа: количество граждан, которые смогут воспользоваться услугами МФЦ, и номер окна, в котором специалист примет последнего гражданина.

Решение

В самом начале необходимо получить все данные из файла, после чего отсортировать время прихода и ухода каждого посетителя по возрастанию.

Далее создаётся список places, который будет содержать время ухода для K посетителей, обслуживаемых в данный момент.

Затем происходит последовательный подбор мест для каждого посетителя из списка: если время прихода текущего посетителя больше времени ухода предыдущего, значит человек может занять это место.

При записи нового посетителя необходимо увеличивать счётчик общего количества занятых мест на 1, а также записывать номер последнего занятого места (важно помнить, что нумерация идёт с 1).

```
f = open('file.txt')  # открываем файл
k = int(f.readline())  # считаем число K
n = int(f.readline())  # считываем число N
a = [list(map(int, i.split())) for i in f]  # считываем заявки из файла
a.sort()  # сортируем заявки по возрастанию времени начала и конце

places = [-1 for _ in range(k)]  # создаём k мест, которые можно занять
count = last_place = 0  # создаём переменные, в которые будет записан ответ
```

```
for start, end in a:  # проверяем время прихода и ухода каждого посетителя
    for j in range(k):  # подбираем подходящее место
        if start > places[j]:  # если время прихода текущего посетителя больше
                               # времени ухода предыдущего,
            places[j] = end  # занимаем данное место, присвоив ему время ухода
                             # текущего посетителя
            count += 1  # увеличиваем счётчик количества посетителей, которые
                        # смогли прийти
            last_place = j + 1  # записываем номер последнего место (помним,
                                # что нумерация начинается с 1)
            break  # останавливаем подбор подходящих мест

print(count, last_place)  # выводим ответ на экран
```

Задача 26.2

В центр обработки информации приходят запросы на сервер, имеющий ограниченный запас памяти. Для каждого запроса дана время регистрации, идентификатор клиентского устройства и объём данных. Если в какой-то момент приходит информация, а свободного объёма памяти сервера не хватает для сохранения этой информации, сервер делает резервную копию и отправляет её в облако, после чего память сервера обнуляется, новая информация добавляется на сервер.

Входные данные

Первая строка входного файла содержит два натуральных числа: N – количество строк, – вместимость специального раздела памяти сервера в Кб. Каждая из следующих N строк содержит информацию об одном выполненном запросе: время регистрации в формате ЧЧ:ММ:СС и два натуральных числа: – идентификатор клиентского устройства, S – объем данных запроса в Кб.

Выходные данные

Два целых положительных числа: сначала идентификатор клиентского устройства, который отправил наибольший объём запросов, а затем максимальный суммарный объём двух резервных копий, которые отправлялись в облако до 12 часов дня.

Задача №27

Задача 27.1

Учёный решил провести кластеризацию некоторого множества звёзд по их расположению на карте звёздного неба. Кластер звёзд – это набор звёзд (точек) на графике. Каждый кластер имеет форму прямоугольника, причём эти прямоугольники между собой не пересекаются. Центр кластера – это одна из звёзд на графике, сумма расстояний от которой до всех остальных звёзд кластера минимальна.

В файле А хранятся данные о звёздах 2-х кластеров, в файле Б хранятся данные о звёздах 3-х кластеров. Для каждой звезды дана характеристика: тип цвета, тип светимости и её размер в соответствии с таблицей.

Обозначение	Цвет	Обозначение	Размер
G	белый	I	сверхгигант
J	зелёный	II	яркий гигант
L	синий	III	гигант
N	оранжевый	IV	субгигант
Y	красный	V	карлик
S	голубой	VI	субкарлик
Z	жёлтый	VII	квазар

Полученные значения записаны в характеристике слитно: обозначение цвета, светимость (обозначается цифрой 1...9) и обозначение размера (римские цифры).

Расстояние между двумя точками $A(x_1, y_1)$ и $B(x_2, y_2)$ вычисляется по формуле:

$$\rho(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Даны два входных файла (файл А и файл Б). Для файла А определите координаты центра каждого кластера, затем найдите два числа: A_1 – минимальное расстояние от центра кластера с наименьшим количеством точек до красного гиганта, и A_2 – максимальное расстояние от центра кластера с наименьшим количеством точек до красного гиганта.

Для файла Б определите координаты центра каждого кластера, затем найдите два числа: B_1 – минимальное расстояние между двумя различными жёлтыми карликами, расположенными в одном и том же кластере, и B_2 – расстояние между центрами кластеров с минимальным и максимальным количеством жёлтых карликов.

В ответе запишите четыре числа: в первой строке – целую часть произведения $A_1 \times 10000$, затем целую часть произведения $A_2 \times 10000$; во второй строке – сначала целую часть произведения $B_1 \times 10000$, затем целую часть произведения $B_2 \times 10000$.