Основная волна. Версия от 19.06

Содержание

Задача №3 **4**
Задача 3.1 (Сибирь) . . . 4

Задача №4 **6**
Задача 4.1 (Дальний восток) . . . 6
Задача 4.2 (Сибирь) . . . 6

Задача №5 **8**
Задача 5.1 (Дальний восток) . . . 8

Задача №6 **9**
Задача 6.1 (Дальний восток) . . . 9
Задача 6.2 (Сибирь) . . . 10

Задача №7 **13**
Задача 7.1 (Дальний восток) . . . 13
Задача 7.2 (Дальний восток) . . . 13
Задача 7.3 (Сибирь) . . . 14

Задача №8 **15**
Задача 8.1 (Дальний восток) . . . 15
Задача 8.2 (Дальний восток) . . . 16

Задача №9 **19**
Задача 9.1 (Дальний восток) . . . 19
Задача 9.2 (Сибирь) . . . 20

Задача №10 **23**
Задача 10.1 (Дальний восток) . . . 23
Задача 10.2 (Сибирь) . . . 23

Задача №11 **24**
Задача 11.1 (Дальний восток) . . . 24
Задача 11.2 (Дальний восток) . . . 25
Задача 11.3 (Дальний восток) . . . 26

Задача №12 **27**
Задача 12.1 (Сибирь) . . . 27

Задача №13 **30**
Задача 13.1 (Дальний восток) . . . 30
Задача 13.2 (Дальний восток) . . . 31
Задача 13.3 (Дальний восток) . . . 32
Задача 13.4 (Сибирь) . . . 33

Задача №14 **35**
Задача 14.1 (Дальний восток) . . . 35
Задача 14.2 (Дальний восток) . . . 36
Задача 14.3 (Дальний восток) . . . 36

Задача №15 **38**
Задача 15.1 (Дальний восток) . . . 38
Задача 15.2 (Дальний восток) . . . 38
Задача 15.3 (Сибирь) . . . 39

Задача №16 **40**
Задача 16.1 (Дальний восток) . . . 40

Задача №17 **41**
Задача 17.1 (Дальний восток) . . . 41

Задача №18 **42**
Задача 18.1 (Дальний восток) . . . 42

Задача №19-21 **44**
Задача 19.1 (Дальний восток) . . . 44
Задача 20.1 (Дальний восток) . . . 45
Задача 21.1 (Дальний восток) . . . 46
Задача 19.2 (Центр) . . . 47
Задача 20.2 (Центр) . . . 48
Задача 21.2 (Центр) . . . 49

Задача №23 **51**
Задача 23.1 (Дальний восток) . . . 51

Задача №24 **52**
Задача 24.1 (Дальний восток) . . . 52
Задача 24.2 (Дальний восток) . . . 53

Задача №25 **57**
Задача 25.1 (Дальний восток) . . . 57
Задача 25.2 (Дальний восток) . . . 59
Задача 25.3 (Дальний восток) . . . 60
Задача 25.4 (Сибирь) . . . 61

Задача №26 **64**
Задача 26.1 (Дальний восток) 64
Задача 26.2 (Сибирь) 66

Задача №27 **69**
Задача 27.1 (Дальний восток) 69
Задача 27.2 (Сибирь) 73

Задача №3

Задача 3.1 (Сибирь)

В файле приведён фрагмент базы данных «Хозтовары» о поставках товаров для ухода, уборки и дома. База данных состоит из трёх таблиц.

Таблица **«Движение товаров»** содержит записи о поставках товаров в магазины в течение июля 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

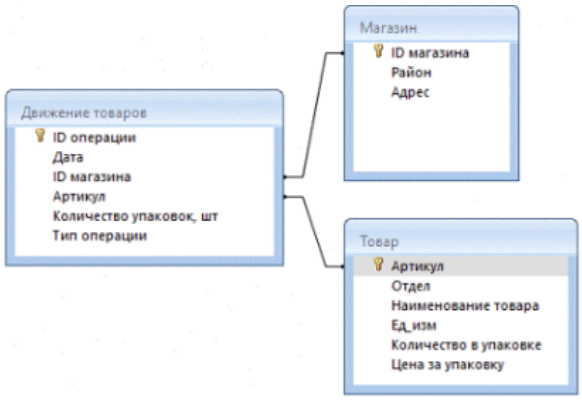
Таблица **«Товар»** содержит информацию об основных характеристиках каждого товара. Заголовок таблицы имеет следующий вид.

Артикул	Отдел	Наименование товара	Единица измерения	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 общее количество литров всех видов шампуня, проданных магазинами, расположенными на улице Металлургов, за период с 7 по 22 июля включительно.

В ответе запишите только целую часть полученного числа.

Решение

В таблице «Движение товаров» создаем три новых столбца: «Вид товара», «Улица», «Литров в упаковке». Для получения вида товара в ячейке G2 запишем и протянем вниз формулу:

$$=\text{ВПР}(\text{D2};\$\text{Товар.A:F};3;0)$$

Для нахождения улицы магазина в ячейке H2 запишем и протянем вниз формулу:

$$=\text{ВПР}(\text{C2};\$\text{Магазин.A:C};3;0)$$

Наконец, для нахождения литров в одной упаковке запишем в I2 и протянем вниз формулу:

$$=\text{ВПР}(\text{D2};\$\text{Товар.A:F};5;0)/1000$$

Необходимо делить на 1000, так как в таблице «Товар» для всех видов шампуня единица измерения – миллилитры, а у нас просятся литры.

Остаётся вычислить объём партии, умножив количество упаковок на объём одной. Для этого в ячейке J2 запишем и протянем вниз формулу:

$$=\text{I2}*\text{E2}$$

Теперь воспользуемся фильтром: по дате (с 7 по 22 июля), улице (Металлургов), виду товара (шампунь всех видов) и по типу операции (продажа). Выделяем последний столбец и получаем искомый объём.

Ответ: 1274

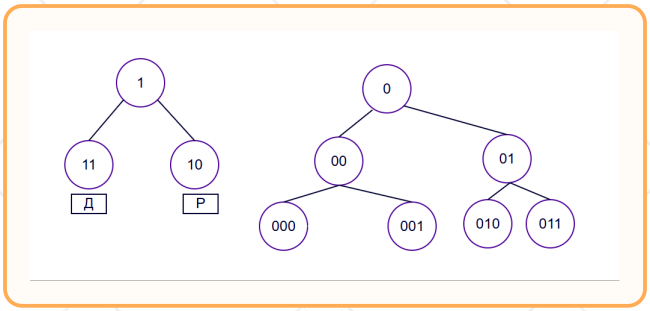
Задача №4

Задача 4.1 (Дальний восток)

Для кодирования некоторой последовательности, состоящей из букв К, О, Р, Л, Ё, Н, Д, используется неравномерный двоичный код, удовлетворяющий условию Фано. Для букв Р и Д используются кодовые слова: Р – 10, Д – 11. Какова наименьшая возможная длина закодированной последовательности для слова КОРОЛЁНОК?

Решение

Построим дерево Фано и подпишем известные коды:



Нам необходимо закодировать ещё 5 букв: К, О, Л, Ё, Н. Все они используются в слове КОРОЛЁНОК. При это буква О используется трижды, буква К – дважды, буквы Л, Ё и Н по одному разу.

Рассмотрим два варианта присваивания кодов:

1) Букве О присвоим код 00, букве К – 010, код 011 продлим до 0110, 01110 и 01111 и присвоим их буквам Л, Ё, Н.

Тогда общая длина слова КОРОЛЁНОК равна:

$$3 \cdot 2 + 2 \cdot 3 + 2 + 4 + 5 + 5 = 28$$

2) Продлим коды 00 и 01 до пяти кодов: 000, 001, 010, 0110 и 0111. Буквам О и К присвоим коды 000 и 001; буквам Л, Ё, Н – оставшиеся.

Тогда общая длина слова КОРОЛЁНОК равна:

$$3 \cdot 2 + 3 \cdot 3 + 2 + 3 + 4 + 4 = 28$$

Оба способа дали длину слова КОРОЛЁНОК, равную 28.

Ответ: 28

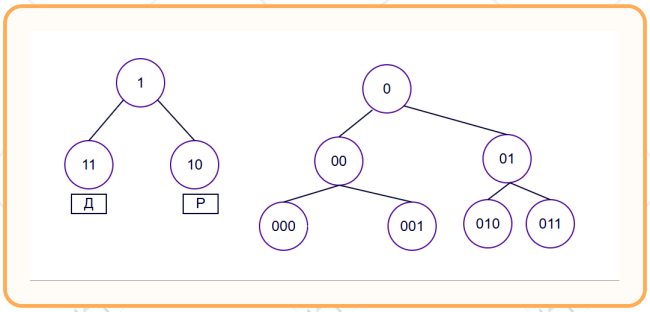
Задача 4.2 (Сибирь)

Для кодирования некоторой последовательности, состоящей из букв А, К, Л, Н, И, Ч, используется неравномерный двоичный код, удовлетворяющий условию Фано. Для букв Л и Н используются кодовые слова: Л – 1, Н – 10. Какова наименьшая возможная длина закоди-

рованной последовательности для слова КАЛАЧИК?

Решение

Построим дерево Фано и подпишем известные коды:



Для начала распишем коды на древе тех букв, что нам известны. Буква Л имеет код 1, а буква Н – 01. Значит, что все последующие коды будут начинаться с 0. Из оставшихся букв чаще всего встречаются буквы К и А, для них нужно дать коды покороче, по этой причине букве К дадим код 000. Букве А мы не может дать код 001, поскольку в таком случае для оставшихся букв не останется свободных кодов. Продлим ветку 001 и букве А дадим код 0010, а оставшимся буквам коды 00110 и 00111. Посчитаем длину итогового слова:

$$2 \cdot 3 + 2 \cdot 4 + 1 + 5 + 5 = 25$$

Ответ: 25

Задача №5

Задача 5.1 (Дальний восток)

Автомат получает на вход число N и преобразует его в число R.
1. Строится двоичная запись числа N.
2. Если сумма цифр двоичного числа нечетное число, то дописывается 01.
3. Если сумма цифр двоичного числа четное число, то дописывается 10.
4. Результат переводится в десятичную систему и он же является числом R.

Какое наименьшее число R, большее 123, может, быть результатом работы этой программы?

Решение

Для каждого числа построим его двоичную запись с помощью функции bin(), отбросив служебный префикс '0b' через срез [2:]. Сумма цифр в двоичном числе равна количеству единиц в его записи, поэтому для её подсчета воспользуемся методом .count('1'). Если это количество нечетно, допишем в конец строки '01', а если четно – '10'. Полученный результат переведем обратно в десятичную систему с помощью int(s, 2).

Все полученные значения R, которые превосходят 123, сохраним в список, чтобы в конце работы программы вывести из них минимальное.

```
results = []
for n in range(1000):
    s = bin(n)[2:]  # Строим двоичную запись числа N
    sum_digits = s.count('1')  # Считаем сумму цифр (количество единиц)

    # Дописываем окончание по правилам алгоритма
    if sum_digits % 2 != 0:
        s += '01'
    else:
        s += '10'

    r = int(s, 2)  # Получаем результат R в десятичной системе

    # Если результат больше 123, сохраняем его
    if r > 123:
        results.append(r)

# Находим и выводим наименьший результат R
print(min(results))
```

Ответ: 125

Задача №6

Задача 6.1 (Дальний восток)

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.

Черепахе был дан для исполнения следующий алгоритм:

Повтори 4 [Вперёд 28 Направо 90 Вперёд 26 Направо 90]
Поднять хвост
Вперёд 12 Направо 90 Вперёд 13 Налево 90
Опустить хвост
Повтори 4 [Вперёд 67 Направо 90 Вперёд 76 Направо 90]

Определите площадь пересечения фигур, ограниченного заданными алгоритмом линиями.

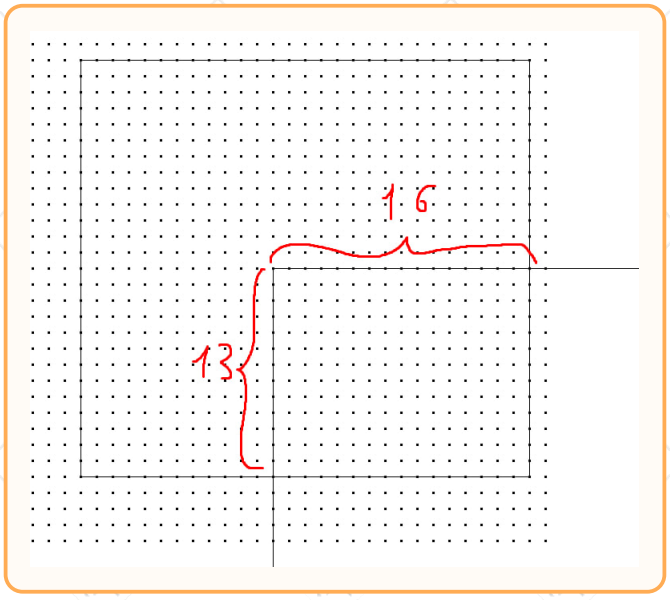
Решение

Напишем программу, используя библиотеку turtle для визуализации движения Черепахи по заданному алгоритму. Основная цель — построить фигуры по алгоритму, а затем проверить, какие точки с целочисленными координатами лежат внутри их пересечения. Для этого сначала выполняется алгоритм Черепахи, который рисует границы двух фигур. После этого необходимо отрисовать точки с целочисленными координатами, чтобы после завершения построения подсчитать их количество. Для удобства визуализации применяется масштабирование через переменную m, иначе фигуры будут слишком маленькими. Все перемещения вперёд умножаются на m.

```
from turtle import *

tracer(0)  # Ускоряем анимацию
k = 25  # Масштаб
# Добавляем ползунки справа и снизу
screensize(3000, 3000)

for i in range(4):
    forward(28 * k)  # Вперёд с учётом масштаба
    right(90)
    forward(26 * k)
```

```
    right(90)
pu()  # Поднимаем хвост
forward(12 * k)
right(90)
forward(13 * k)
left(90)
pd()  # Опускаем хвост
for i in range(4):
    forward(67 * k)
    right(90)
    forward(76 * k)
    right(90)

pu()  # Поднимаем хвост
# Отрисовываем координатную сетку
for x in range(-20, 30):
    for y in range(-30, 20):
        goto(x * k, y * k)
        dot(4)  # Ставим точку
done()  # Не даём программе завершиться
```

После запуска программы получаем картинку:



Осталось посчитать площадь пересечения фигур, включая точки на границах этого пересечения. Для этого нужно перемножить количество точек, вычтя 1, на сторонах нашей фигуры.

Ответ: 208

Задача 6.2 (Сибирь)

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.

Черепахе был дан для исполнения следующий алгоритм:

Повтори 7 [Вперед 8 Направо 90 Вперед 7 Направо 90]
Поднять хвост
Вперед 3 Направо 90 Вперед 4 Налево 90
Опустить хвост
Повтори 7 [Вперед 72 Налево 90 Вперед 6 Налево 90]

Определите площадь пересечения фигур, ограниченного заданными алгоритмом линиями.

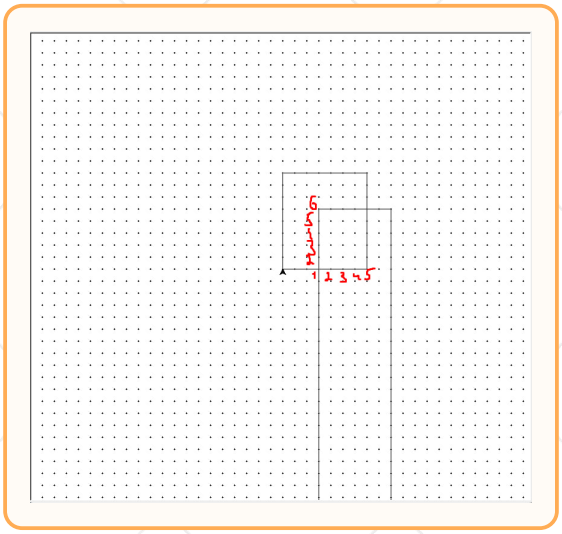
Решение

Используем модуль turtle и переписываем код алгоритма в python

```
from turtle import *
left(90)
tracer(0)
m = 20
pd()
for i in range(7):
    forward(8*m)
    right(90)
    forward(7*m)
    right(90)
pu()
forward(3*m)
right(90)
forward(4*m)
left(90)
pd()
for i in range(7):
    forward(72*m)
    left(90)
    forward(6*m)
```

```
    left(90)
pu()
for x in range(-20,21):
    for y in range(-20,21):
        goto(x*m,y*m)
        dot(3)
done()
```

После запуска программы получим следующую фигуру.



Посчитаем длины сторон фигуры пересечения фигур и перемножим для того чтобы получить площадь пересечения

Ответ: 30

Задача №7

Задача 7.1 (Дальний восток)

Рома записывает голосовое сообщение для своего брата. Перед отправкой сообщение оцифровывается в формате моно с частотой дискретизации 48 000 Гц и глубиной кодирования 16 бит. Определите наименьшее целое количество Кбайт, необходимое для сохранения сообщения в памяти (без учёта заголовка), если его длительность – 1 минута 15 секунд. В ответе укажите только число.

Решение

Объём звука можно найти по формуле: количество каналов · частота (в Гц) · длительность записи (в секундах) · глубина кодирования. Также важно помнить типы форматов, моно содержит 1 канал. Получаем $1 \cdot 48000 \cdot 75 \cdot 16$ бит $= 57\ 600\ 000$ бит.

Ответ просится записать в Кбайтах, переводим, поделив на 8 и 1024: $\frac{57\ 600\ 000}{8 \cdot 1024} = 7031,25$ Кбайт. Округляем вверх, чтобы весь объём записи влез.

Ответ: 7032

Задача 7.2 (Дальний восток)

Настя записывала голосовое сообщение Дане. Перед отправкой сообщение оцифровывается в формате стерео с частотой дискретизации 28 000 Гц и глубиной кодирования 8 бит.

Определите наименьшее целое количество Кбайт, необходимое для сохранения сообщения в памяти (без учета 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:

Формат стерео означает, что запись двухканальная (количество каналов = 2).

Частота дискретизации = 28000 Гц.

Глубина кодирования = 8 бит (это ровно 1 байт).

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:

$$\frac{7840000}{1024} = 7656.25 \text{ Кбайт}$$

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 7.3 (Сибирь)

Рома записывает голосовое сообщение для своего брата. Перед отправкой сообщение оцифровывается в формате моно с частотой дискретизации 128 000 Гц и глубиной кодирования 16 бит. Определите наименьшее целое количество Кбайт, необходимое для сохранения сообщения в памяти (без учёта заголовка), если его длительность – 2 минуты 30 секунд. В ответе укажите только число.

Решение

Объём звука можно найти по формуле: количество каналов * частота (в Гц) * длительность записи (в секундах) * глубина кодирования.

Также важно помнить типы форматов, моно содержит 1 канал.

Получаем:

$$1 \cdot 128\,000 \cdot 150 \cdot 16 \text{ бит} = 307\,200\,000 \text{ бит}.$$

Ответ просится записать в Мбайтах, переводим, поделив на 8, 1024 и 1024:

$$\frac{307\,200\,000}{(8 \cdot 1024 \cdot 1024)} = 36,6 \text{ Мбайт}.$$

Округляем вверх до целого, чтобы весь объём записи полностью влез.

Ответ: 37

Задача №8

Задача 8.1 (Дальний восток)

Все пятибуквенные слова, составленные из букв Л, Е, Б, Е, Д, Ь, записаны в алфавитном порядке и пронумерованы. Вот начало списка:

ББББЬ
ББББД
ББББЕ
ББББЛ
ББББЬ

Определите, под каким номером в этом списке стоит последнее слово, которое начинается с гласной буквы, содержит ровно две буквы Л и не содержит ни одной буквы Ь.

Решение

Решение руками

Для начала дадим каждой букве из алфавита числовое обозначение

Б - 0
Д - 1
Е - 2
Л - 3
Ь - 4

Мы получили числа 5сс. Теперь составим максимальное по значению число, удовлетворяющее всем условиям:

$$ЕЛЛЕЕ - 23322_5$$

Переведем данное число в 10сс и получим число 1712_{10}

Для того, чтобы узнать порядковый номер данного слова, увеличим число на 1 и получим ответ 1713.

Решение программой с помощью циклов

Будем последовательно перебирать все возможные пятибуквенные слова из заданного алфавита в том порядке, в котором они расположены в условии. Для каждого полученного слова будем увеличивать счётчик, который хранит его номер в общем списке. Затем проверим выполнение всех условий задачи: первая буква должна быть Е, слово должно содержать ровно две буквы Л и не должно содержать букву Ь. Если все условия выполняются, выводим номер слова.

```
# Строка с буквами в алфавитном порядке
a = "БДЕЛЬ"
# Счётчик номеров слов
count = 0
# Перебираем первую букву слова
for x1 in a:
```

```
    # Перебираем вторую букву слова
    for x2 in a:
        # Перебираем третью букву слова
        for x3 in a:
            # Перебираем четвёртую букву слова
            for x4 in a:
                # Перебираем пятую букву слова
                for x5 in a:
                    # Собираем слово из выбранных букв
                    s = x1+x2+x3+x4+x5
                    # Переходим к следующему номеру слова
                    count += 1
                    # Проверяем выполнение всех условий задачи
                    if s[0] == "Е" and s.count("Л") == 2 and "Ь" not in s:
                        print(count)
```

Вместо пяти вложенных циклов воспользуемся функцией `product()`, которая автоматически генерирует все возможные пятибуквенные слова из заданного набора букв. Поскольку буквы указаны в алфавитном порядке, функция будет выдавать слова в том же порядке, в котором они должны находиться в общем списке. Для каждого слова будем вести подсчёт номера и проверять выполнение условий задачи.

Если все условия выполняются одновременно, выводим номер слова.

```
# Импортируем функцию product для генерации всех комбинаций
from itertools import product
# Счётчик номеров слов
count = 0
# Генерируем все возможные пятибуквенные слова
for i in product("БДЕЛЬ",repeat = 5):
    # Преобразуем кортеж символов в строку
    s = "".join(i)
    # Увеличиваем номер текущего слова
    count += 1
    # Проверяем выполнение всех условий задачи
    if s[0] == "Е" and s.count("Л") == 2 and "Ь" not in s:
        print(count)
```

Ответ: 1713

Задача 8.2 (Дальний восток)

Все шестибуквенные слова, составленные из букв К, О, М, Б, И записаны в алфавитном порядке и пронумерованы.

Ниже приведено начало списка:

1. БББББК

2. БББББИ
3. БББББК
4. БББББМ
5. БББББО
6. БББББИБ
...

Под каким номером в списке стоит последнее слово с нечётным номером, которое не начинается с буквы М, содержит не менее двух букв К и не содержит букв И?

Решение

Решение руками

Для начала дадим каждой букве из алфавита числовое обозначение

Б - 0
И - 1
К - 2
М - 3
О - 4

Мы получили числа 5сс. Теперь составим максимальное по значению число, удовлетворяющее всем условиям:

$$OOOOKK - 444422_5$$

Переведем данное число в 10сс и получим число 15612_{10}

Для того чтобы узнать порядковый номер данного слова увеличим число на 1 и получим ответ 15613

Решение программой с помощью циклов

Будем последовательно перебирать все возможные шестибуквенные слова из заданного алфавита в том порядке, в котором они должны располагаться в списке. Для каждого сформированного слова будем вести подсчёт его номера. После получения очередного слова проверим выполнение всех условий задачи: слово не должно начинаться с буквы М, должно содержать не менее двух букв К, не должно содержать букву И, а его номер должен быть нечётным.

Если все условия одновременно выполняются, выводим номер слова.

```
# Записываем все допустимые буквы в алфавитном порядке
a = "БИКМО"
# Счётчик номеров слов
count = 0
# Перебираем первую букву слова
for x1 in a:
    # Перебираем вторую букву слова
    for x2 in a:
        # Перебираем третью букву слова
        for x3 in a:
```

```
            # Перебираем четвёртую букву слова
            for x4 in a:
                # Перебираем пятую букву слова
                for x5 in a:
                    # Перебираем шестую букву слова
                    for x6 in a:
                        # Собираем слово из выбранных букв
                        s = x1+x2+x3+x4+x5+x6
                        # Увеличиваем номер текущего слова
                        count += 1
                        # Проверяем выполнение всех условий задачи
                        if (s[0] != "М") and (s.count("К") >= 2) and ("И" not in s)
                        and (count % 2 != 0):
                            print(count)
```

Решение программой с помощью модуля itertools

Вместо шести вложенных циклов воспользуемся функцией `product()` из модуля `itertools`. Эта функция автоматически генерирует все возможные слова заданной длины из указанного набора букв. Поскольку буквы в строке записаны в нужном порядке, генерация также происходит в правильном порядке нумерации слов. Для каждого слова будем вести подсчёт его номера и проверять выполнение условий задачи.

Если все условия одновременно выполняются, выводим номер слова.

```
# Импортируем функцию для генерации всех возможных комбинаций
from itertools import product
# Счётчик номеров слов
count = 0
# Генерируем все возможные шестибуквенные слова
for i in product("БИКМО",repeat = 6):
    # Преобразуем кортеж символов в строку
    s = "".join(i)
    # Увеличиваем номер текущего слова
    count += 1
    # Проверяем выполнение всех условий задачи
    if s[0] != "М" and s.count("К") >= 2 and "И" not in s and count % 2 != 0:
        print(count)
```

Ответ: 15613

Задача №9

Задача 9.1 (Дальний восток)

Откройте файл электронной таблицы, содержащей в каждой строке четыре натуральных числа. Определите количество строк таблицы, для чисел которых выполнено условие:
– сумма трех наименьших чисел умноженная на 2 больше чем сумма трех наибольших чисел.
В ответе запишите только число.

Решение

Решение в электронной таблице

Для решения этой задачи будем использовать функции автоматической сортировки значений в строке: $=НАИМЕНЬШИЙ()$ и $=НАИБОЛЬШИЙ()$.

Для каждой строки с числами в столбцах A, B, C и D запишем в ячейку $E1$ одну общую формулу, которая проверит условие напрямую:

$$=ЕСЛИ(((НАИМЕНЬШИЙ(A1:D1;1)+НАИМЕНЬШИЙ(A1:D1;2)+$$

$$+НАИМЕНЬШИЙ(A1:D1;3))*2)>(НАИБОЛЬШИЙ(A1:D1;1)+НАИБОЛЬШИЙ(A1:D1;2)+$$

$$+НАИБОЛЬШИЙ(A1:D1;3));1;0)$$

Как работает эта формула: В левой части складываются три наименьших числа строки (1-е, 2-е и 3-е по возрастанию) и умножаются на 2. В правой части складываются три наибольших числа строки (1-е, 2-е и 3-е по убыванию).

Если сумма трех наименьших, умноженная на 2, строго больше суммы трех наибольших, формула выведет 1, иначе – 0.

Растягиваем формулу в столбце E на все строки таблицы. Ответ будет равен сумме всех значений в столбце E.

Решение программой

На каждом шаге считываем строку, разбиваем её на числа и сортируем их по возрастанию методом $sorted()$. Сортировка гарантирует, что: Элементы $a[0]$, $a[1]$, $a[2]$ будут тремя наименьшими числами в строке. Элементы $a[1]$, $a[2]$, $a[3]$ будут тремя наибольшими числами в строке.

Проверяем условие задачи. Если оно выполняется, увеличиваем значение счетчика на 1.

```
# Открываем файл с данными
f = open("9.txt")

ans = 0 # Счетчик подходящих строк

for line in f:
    # Считываем строку, разбиваем на числа и сразу сортируем по возрастанию
    a = sorted(map(int, line.split()))

    # Проверяем условие:
```

```
    # Сумма трех наименьших (a[0], a[1], a[2]), умноженная на 2,
    # должна быть больше суммы трех наибольших (a[1], a[2], a[3])
    if (a[0] + a[1] + a[2]) * 2 > (a[1] + a[2] + a[3]):
        ans += 1

print(ans) # Выводим итоговое количество
```

Задача 9.2 (Сибирь)

Откройте файл электронной таблицы, содержащей в каждой строке пять натуральных чисел. Определите наименьший номер строки, для которой выполнены оба условия:
– все числа в строке различны;
– удвоенная сумма максимального и минимального больше суммы остальных трех чисел.

Решение

Решение Excel:

Сначала проверим 1 условие. Запишем в ячейку $F1$ следующую формулу и растянем её вниз до конца таблицы:

=ЕСЛИ(ИЛИ(A1=B1;A1=C1;A1=D1;A1=E1;B1=C1;B1=D1;B1=E1;C1=D1;C1=E1;D1=E1);0;1)

Затем проверим 2 условие. Запишем в ячейку $G1$ следующую формулу и растянем её вниз до конца таблицы:

=ЕСЛИ((МАКС(A1:E1)+МИН(A1:E1))*2>СУММ(A1:E1)-МАКС(A1:E1)-МИН(A1:E1);1;0)

Для того чтобы выполнялись оба условия, в ячейку $H1$ запишем формулу и растянем её вниз до конца таблицы:

=F1*G1

Находим первую строку, в которой в столбце H стоит единичка и получаем ответ – 4.

Решение программой:

Сначала мы открываем файл с данными. В каждой строке файла содержится пять натуральных чисел. Для доступа к данным используем функцию `open()`, которая позволяет читать файл построчно.

Далее создаём переменную-счётчик, которая будет хранить количество строк, удовлетворяющих условиям задачи. Изначально счётчик равен нулю.

После этого организуем цикл `for`, который последовательно перебирает все строки файла. На каждой итерации переменная `line` содержит одну строку текста.

Сразу же вводим счётчик строк с помощью переменной `count`.

Затем преобразуем строку в список чисел. Для этого выполняем следующие действия:

1. Разбиваем строку на отдельные элементы с помощью метода `split()`, который разделяет строку по пробелам.
2. Каждый элемент преобразуем в целое число с помощью функции `int()`.
3. Формируем из этих чисел список `a`.

После получения списка выполняем проверку условий задачи.

– Первое условие: все числа в строке различны.

Для проверки этого условия используем множество. Множество автоматически удаляет повторяющиеся элементы.

Если длина множества `set(a)` равна 5 (число элементов в строке), значит все числа различны.

– Второе условие: удвоенная сумма максимального и минимального числа строки больше суммы оставшихся трёх чисел. Для этого:

- находим максимальное число в списке с помощью функции `max(a)`;
- находим минимальное число в списке с помощью функции `min(a)`;
- вычисляем удвоенную сумму максимального и минимального числа (`2 * (max(a) + min(a))`);
- вычисляем сумму оставшихся трёх чисел, вычитая максимальное и минимальное число из общей суммы списка (`sum(a) - max(a) - min(a)`);
- проверяем неравенство `2 * (max(a) + min(a)) > sum(a) - max(a) - min(a)`.

Если оба условия выполняются, выводим номер строки и прекращаем цикл.

```
# Открываем файл для чтения строк
f = open("9-2.txt")
# Создаём переменную-счётчик для подсчёта подходящих строк
count = 0
# Перебираем все строки файла по одной
for line in f:
    count += 1
    # Разбиваем строку на элементы, преобразуем их в целые числа
    # и формируем список из пяти чисел
    a = [int(x) for x in line.split()]
    # Проверяем выполнение двух условий:
    # 1) все числа различны
    # 2) удвоенная сумма максимального и минимального числа
    # больше суммы оставшихся трёх чисел
    if len(set(a)) == 5 and 2*(max(a)+min(a)) > sum(a)-max(a)-min(a):
        # Если оба условия выполняются, выводим номер
        # первой подошедшей строки
        print(count)
        break
```

Ответ: 4

Задача №10

Задача 10.1 (Дальний восток)

Определите, сколько раз в файле, содержащем книгу братьев Стругацких «Трудно быть богом», встречается сочетание букв «где» в составе других слов, включая сложные слова, соединённые дефисом, но не как отдельное слово. Строчные и заглавные буквы в этом задании не различаются.

Решение

Открываем файл в *LibreOffice* и нажимаем Ctrl+f и в появившемся строке в левом нижнем углу нажимаем на значок лупы.

В открывшемся окне в поисковую строку вводим слово «где» и проставляем галочку у настройки «Слово целиком». Выполняем поиск и получаем 82 вхождения. Запоминаем это число – это все слова, где слово «где» не является частью другого слова.

Далее убираем галочку у настройки «Слово целиком» и выполняем поиск повторно. Получаем 84 вхождения – это все слова, где упоминается «где».

При поиске с галочкой «Слово целиком», поиск учитывает слова, где сочетание «где» записано частью другого через дефис как отдельное слово.

Поэтому в строку вводим слово «где-» без галочки «Слово целиком» и выполняем поиск – таких слов 21.

Значит, всего слов, где слово «где» является частью других слов: $84 - 82 + 21 = 23$

Ответ: 23

Задача 10.2 (Сибирь)

С помощью текстового редактора определите, сколько раз встречается сочетание букв «да» или «Да» только в составе других слов, в том числе в сложных словах, соединённых дефисом, но не как отдельное слово, в тексте главы III повести А. И. Куприна «Поединок».

В ответе запишите только число.

Решение

Открываем имеющийся текстовый документ, а также создаём новый – в него мы будем добавлять проверяемые части текста.

Заходим во вкладку «Вид» → «Навигатор», выбираем нужные главы, после чего копируем их и вставляем в созданный файл.

Теперь открываем меню поиска при помощи специальной кнопки или сочетания клавиш «Ctrl + H» → «Найти».

Находим «да» в основном документе, поставив галочку возле «Только слово целиком». Полученное число записываем в качестве ответа.

Задача №11

Задача 11.1 (Дальний восток)

На предприятии каждой изготовленной детали присваивают серийный номер, состоящий из 200 символов. В базе данных каждый серийный номер занимает одинаковое и минимально возможное число байт. При этом используется посимвольное кодирование серийных номеров, все символы кодируются одинаковым и минимально возможным числом бит. Известно, что для хранения 85 536 серийных номеров потребовалось не менее 9 Мбайт памяти. Определите минимально возможную мощность алфавита, используемого для записи серийных номеров. В ответе запишите только целое число.

Решение

Сначала переведем объем памяти в 9 Мбайт в байты:

$$9 \text{ Мбайт} = 9 \cdot 1024 \cdot 1024 = 9\,437\,184 \text{ байт}$$

.

По условию задачи, для хранения 85 536 серийных номеров потребовалось не менее 9 Мбайт памяти. Найдем минимальный объем памяти в байтах, который должен занимать один серийный номер (обозначим его за V):

$$85\,536 \cdot V \geq 9\,437\,184$$

$$V \geq \frac{9\,437\,184}{85536} \approx 110.33 \text{ байт.}$$

Поскольку для хранения каждого серийного номера отводится целое число байт, минимально возможное значение V равно 111 байт.

Каждый серийный номер состоит из 200 символов, при этом каждый символ кодируется одинаковым и минимально возможным целым числом бит i.

Объем памяти для хранения одного серийного номера в битах равен $200 \cdot i$ бит.

Выразим этот объем в байтах:

$$V = \frac{200 \cdot i}{8} = 25 \cdot i \text{ байт.}$$

Так как число байт V кратно 25 и должно удовлетворять условию $V \geq 111$, то минимально возможным значением объема одного серийного номера является ближайшее большее или равное число, кратное 25:

$$V = 125 \text{ байт}(\text{ при } i = 5 \text{ бит}).$$

При $i = 4$ бит объем одного номера составил бы $25 \cdot 4 = 100$ байт, тогда для всех номеров потребовалось бы 8 553 600 байт, что меньше 9 Мбайт.

Следовательно, минимальное число бит на символ i равно 5.

Определим минимально возможную мощность алфавита M, для посимвольного кодирования которого требуется 5 бит:

$$2^4 < M \leq 2^5, \text{ то есть } 16 < M \leq 32.$$

Минимально возможная мощность алфавита M равна 17 (так как при $M = 16$ было бы достаточно 4 бит).

Ответ: 17

Задача 11.2 (Дальний восток)

На заводе каждой изготовленной детали присваивают уникальный код, состоящий из 450 символов. В базе данных каждый серийный номер занимает одинаковое и минимально возможное число байт. При этом используется посимвольное кодирование кодов, все символы кодируются одинаковым и минимально возможным числом бит. Известно, что для хранения 780 100 таких кодов потребовалось не более 83 Мбайт памяти. Определите максимально возможную мощность алфавита, используемого для записи кодов. В ответе запишите только целое число.

Решение

Сначала переведем объём памяти в 83 Мбайта в байты: 83 Мбайт $= 83 \cdot 1024 \cdot 1024 = 87031808$ байт.

По условию задачи, для хранения 780100 серийных номеров потребовалось не более 83 Мбайт памяти. Найдем максимальный объем памяти в байтах, который должен занимать один серийный номер (обозначим его за V):

$$780100 \cdot V \leq 87031808$$

$$V \leq \frac{87031808}{780100} \approx 111.56 \text{ байт}.$$

Поскольку для хранения каждого серийного номера отводится целое число байт, максимально возможное значение V равно 111 байт.

Каждый серийный номер состоит из 450 символов, при этом каждый символ кодируется одинаковым и минимально возможным целым числом бит i. Объем памяти для хранения одного серийного номера в битах равен $450 \cdot i$ бит.

Найдём, сколько бит выходит на 1 символ:

$$i = \frac{111 \cdot 8}{450} \approx 1.97 \text{бит}.$$

Взять $i = 2$ не получится, так как в этом случае мы получим как минимум $\frac{450 \cdot 2}{8} = 112.5$ байт $>$ 111 байт.

Значит, нам подходит $i = 1$. Тогда максимальная мощность алфавита вычисляется по формуле: $N = 2^i = 2^1 = 2$.

Ответ: 2

Задача 11.3 (Дальний восток)

На предприятии каждой изготовленной детали присваивают серийный номер, состоящий из 20 символов. В базе данных каждый серийный номер занимает одинаковое и минимально возможное целое число байт. При этом используется посимвольное кодирование серийных номеров, все символы кодируются одинаковым и минимально возможным целым числом бит. Известно, что для хранения 13944700 серийных номеров требуется не менее 2 Гбайт памяти. Определите минимально возможную мощность алфавита, используемого для записи серийных номеров. В ответе запишите только целое число.

Решение

Сначала переведем объём памяти в 2 Гбайта в байты:

$$2 \text{ Гбайт} = 2 \cdot 1024 \cdot 1024 \cdot 1024 = 2147483648 \text{ байт.}$$

По условию задачи, для хранения 13 944 700 серийных номеров потребовалось не менее 2 Гбайт памяти. Найдем минимальный объем памяти в байтах, который должен занимать один серийный номер (обозначим его за V):

$$13\,944\,700 \cdot V \to 2147483648$$
$$V \to 2147483648 \,/\, 13\,944\,700 \sim 153.99 \text{ байт.}$$

Поскольку для хранения каждого серийного номера отводится целое число байт, максимально возможное значение V = 154 байт.

Каждый серийный номер состоит из 20 символов, при этом каждый символ кодируется одинаковым и минимально возможным целым числом бит i.

Объем памяти для хранения одного серийного номера в битах равен $20 \cdot i$ бит.

Найдём, сколько бит выходит на 1 символ:

$$i = \frac{154 \cdot 8}{20} = 61,6 \text{ бит}$$

Взять $i = 61$ не получится, так как в этом случае для хранения 13 944 700 потребуется менее 2 Гбайт памяти.

Значит, нам подходит $i = 62$. Тогда минимальная мощность алфавита вычисляется по формуле:

$$N = 2^(i - 1) + 1 = 2^{61} + 1 = 2305843009213693953.$$

Ответ: 2305843009213693953

Задача №12

Задача 12.1 (Сибирь)

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	...
								$\blacktriangle q_0$	

Конечное состояние исполнителя после завершения выполнения программы:

...	λ	λ	X	X	X	X	λ	λ	...
		$\blacktriangle q_1$							

Выполните задание.

На ленте в соседних ячейках записано двоичное представление числа 1024 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
q_0	λ, L, q_1		
q_1	λ, S, q_1	$1, L, q_1$	$0, R, q_2$
q_2	$0, S, q_2$	$0, R, q_2$	$1, R, q_2$

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

Решение

Запишем число 1024 в двоичном представлении:

$$1024_{10} = 10000000000_2$$

Проанализируем состояния МТ:

В состоянии q0 мы переходим с ближайшей справа лямбды на число и переходим в состоянии q1. В состоянии q1 мы меняем все 0 на 1 и первую встреченную 1 меняем на 0, переходим на направо и в состояние q2. В состоянии q2 мы просто проходимся до правого края и меняем ближайшую справа лямбду на 0.

В итоге мы получаем то, что алгоритм меняет все нули на единицы, первая встреченная единица заменяется на ноль и в конце числа дописывается ноль. Запишем результат обработки алгоритма входного числа:

$10000000000 \to 01111111110 = 2046_{10}$

Ответ: 2046

Задача №13

Задача 13.1 (Дальний восток)

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. Широковещательным адресом называется специализированный адрес, в котором на месте нулей в маске стоят единицы.

Адрес сети и широковещательный адрес не могут быть использованы для адресации сетевых устройств.

Сеть задана одним из входящих в неё узлов с IP-адресом 152.14.220.61 и маской сети 255.255.254.0. Определите широковещательный адрес и в ответе запишите сумму значений его октетов IP-адреса.

Например, если бы найденный адрес был равен 100.20.3.4, то в ответе следовало бы записать: 127.

Решение

Решение руками

Переведём IP-адрес узла и маску сети в двоичную систему счисления:

IP узла	10011000.00001110.11011100.00111101
Маска	11111111.11111111.11111110.00000000

Узлы в сети имеют вид: 10011000.00001110.1101110x.$xxxxxxxx$

Нам нужно найти широковещательный IP-адрес – то есть поместить на все места символа x единички:

$$10011000.00001110.11011101.11111111_2 = 152.14.221.255_{10}$$

Сумма октетов равна:

$$152 + 14 + 221 + 255 = 642$$

Решение программой

Для нахождения широковещательного IP-адреса необходимо определить адрес сети и взять последний возможный адрес.

В Python для работы с IP-адресами удобно использовать модуль ipaddress, который позволяет легко выполнять операции с сетевыми диапазонами. Создаем объект сети с помощью метода ip_network(), используя IP-адрес узла и маску подсети. Затем, используя индексацию [-1], получаем последний адрес в сети, который и будет искомым широковещательным.

```
# Импортируем модуль для работы с IP-адресами
from ipaddress import *

# Создаем объект сети с указанным IP-адресом и маской
net = ip_network("152.14.220.61/255.255.254.0", 0)
```

```
# Получаем последний адрес в сети:
# net[-1] - последний адрес (широковещательный)
print(net[-1])

# Вывод: 152.14.221.255
# Сумма октетов: 152 + 14 + 221 + 255 = 642
```

Ответ: 642

Задача 13.2 (Дальний восток)

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. Широковещательным адресом называется специализированный адрес, в котором на месте нулей в маске стоят единицы.

Адрес сети и широковещательный адрес не могут быть использованы для адресации сетевых устройств.

Сеть задана IP-адресом одного из входящих в неё узлов 167.66.136.176 и сетевой маской 255.254.0.0.

Определите наименьший IP-адрес данной сети, который может быть присвоен компьютеру. В ответе укажите сумму октетов найденного IP-адреса

Решение

Решение руками:

Сначала найдем адрес самой сети, применив поразрядную конъюнкцию (логическое «И») к IP-адресу узла и маске.

Первый байт: 167 & 255 = 167.

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

66 в двоичной системе: 01000010

254 в двоичной системе: 11111110

Перемножим разряды:

01000010 & 11111110 = 01000010

Полученное число в десятичной системе равно 66.

Третий байт: 136 & 0 = 0.

Четвертый байт: 176 & 0 = 0.

Таким образом, адрес сети равен 167.66.0.0.

По условию требуется найти наименьший IP-адрес, который может быть присвоен компьютеру. Сам адрес сети (167.66.0.0) под адресацию устройств использоваться не может. Следовательно, наименьшим допустимым адресом является следующий за адресом сети: 167.66.0.1.

Сложим значения октетов найденного IP-адреса:

$$167 + 66 + 0 + 1 = 234$$

Решение Python:

Параметр strict= 0 позволяет использовать адрес узла для инициализации сети. Объект сети хранит все адреса последовательно. Под индексом 0 в ней находится адрес самой сети (167.66.0.0), а под индексом 1 — первый применимый для компьютера адрес (167.66.0.1). Преобразуем этот адрес 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('167.66.136.176/255.254.0.0', strict=0)

# Получаем первый допустимый хост в сети (индекс 1) в виде строки
min_ip = str(net[1])

# Разбиваем строку по точкам, переводим части в числа и суммируем их
ans = sum(int(x) for x in min_ip.split('.'))
print(ans)
```

Ответ: 234

Задача 13.3 (Дальний восток)

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. Широковещательным адресом называется специализированный адрес, в котором на месте нулей в маске стоят единицы.

Адрес сети и широковещательный адрес не могут быть использованы для адресации сетевых устройств.

Сеть задана IP-адресом одного из входящих в неё узлов 134.128.220.14 и сетевой маской 255.192.0.0.

Определите наибольший IP-адрес данной сети, который может быть присвоен компьютеру. В ответе укажите найденный IP-адрес без разделителей.

Например, если бы найденный адрес был равен 111.22.3.44, то в ответе следовало бы записать 11122344.

Решение

Решение руками

Переведём IP-адрес узла и маску сети в двоичную систему счисления:

IP узла	10000110.10000000.11011100.00001110
Маска	11111111.11000000.00000000.00000000

Узлы в сети имеют вид: 10000110.10*xxxxxx*.*xxxxxxxx*.*xxxxxxxx*

Нам нужно найти наибольший IP-адрес данной сети, который может быть присвоен компьютеру. Но мы не можем взять последний IP-адрес, который является широковещательным. Поэтому ответом будет являться:

$$10000110.10111111.11111111.11111110_2 = 134.191.255.254_{10}$$

Решение программой

Для нахождения максимального IP-адреса, который может быть назначен компьютеру, необходимо определить адрес сети и взять предпоследний возможный адрес.

В Python для работы с IP-адресами удобно использовать модуль ipaddress, который позволяет легко выполнять операции с сетевыми диапазонами. Создаем объект сети с помощью метода ip_network(), используя IP-адрес узла и маску подсети. Затем, используя индексацию [-2], получаем предпоследний адрес в сети, который и будет максимальным искомым.

```
# Импортируем модуль для работы с IP-адресами
from ipaddress import *

# Создаем объект сети с указанным IP-адресом и маской
net = ip_network("134.128.220.14/255.192.0.0", 0)

# Получаем последний адрес в сети:
# net[-1] - последний адрес (широковещательный)
# net[-2] - предпоследний адрес (максимальный), который
# может быть назначен компьютеру
print(net[-2])

# Вывод: 134.191.255.254
```

Ответ: 134191255254

Задача 13.4 (Сибирь)

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. Широковещательным адресом называется специализированный адрес, в котором на месте нулей в маске стоят единицы.

Адрес сети и широковещательный адрес не могут быть использованы для адресации сетевых устройств.

Сеть задана одним из входящих в неё узлов с IP-адресом 146.180.173.153 и маской сети 255.192.0.0. Найдите наибольший в данной сети IP-адрес. В ответе укажите найденный IP-адрес без разделителей.

Например, если бы найденный адрес был равен 111.22.3.44, то в ответе следовало бы записать: 11122344.

Решение

Решение руками:

Наибольшим возможным IP-адресом в сети является её широковещательный адрес. По определению, в широковещательном адресе все биты, которые в маске равны нулю, заменяются на единицы.

Распишем IP-адрес узла 146.180.173.153 и маску 255.192.0.0 по байтам:

Первый байт маски равен 255 (все единицы), поэтому первый байт искомого адреса совпадает с первым байтом узла: 146.

Для второго байта переведем числа в двоичную систему:

180 в двоичной системе: 10110100

192 в двоичной системе: 11000000

Первые два разряда маски равны 1 (это сетевая часть), остальные шесть разрядов равны 0 (это часть номера).

Сетевая часть второго байта узла: 10.

Заменим оставшиеся шесть хостовых нулей на единицы:

Получаем число 10111111, которое в десятичной системе равно 191.

Третий и четвертый байты маски равны 0. Заменим все их биты на единицы:

В третьем байте получим 255.

В четвертом байте получим 255.

Таким образом, наибольший IP-адрес в данной сети равен 146.191.255.255.

Запишем его без разделителей (точек): 146191255255.

Решение Python:

Наибольший IP-адрес сети (широковещательный адрес) хранится в свойстве broadcast_address объекта сети. Полученный адрес преобразуем в строку и удалим из неё точки с помощью метода .replace('.', '').

```
from ipaddress import ip_network

# Создаем объект сети
net = ip_network('146.180.173.153/255.192.0.0', strict=0)

# Получаем широковещательный адрес (наибольший IP-адрес) в виде строки
broad_addr = str(net.broadcast_address)

# Удаляем точки-разделители и выводим результат
ans = broad_addr.replace('.', '')
print(ans)
```

Ответ: 146191255255

Задача №14

Задача 14.1 (Дальний восток)

Значение арифметического выражения

$$5^{127} + 5^{27} - x,$$

где x – натуральное число, не превышающее 3000, записали в четверичной системе счисления.

Определите наименьшее значение x, при котором в четверичной записи числа, являющегося значением данного арифметического выражения, содержится наибольшее количество нулей.

В ответе запишите число в десятичной системе счисления.

Решение

В данной программе требуется найти такое значение переменной x из диапазона от 1 до 3000, при котором в четверичной записи числа

$$5^{127} + 5^{27} - x$$

содержится максимальное количество цифр 0. Для этого будем последовательно перебирать все возможные значения x, вычислять соответствующее значение выражения, переводить полученное число в систему счисления по основанию 4 и подсчитывать количество нулей в его записи.

```
# Функция перевода числа из десятичной системы счисления в четверичную
def cc4(x):
    # Строка для накопления результата перевода
    s = ""
    # Выполняем перевод, пока число больше нуля
    while x > 0:
        # Добавляем очередную цифру четверичной записи в начало строки
        s = str(x % 4) + s
        # Переходим к следующему шагу алгоритма деления
        x = x // 4
    # Возвращаем полученную четверичную запись
    return s
# Максимальное количество найденных нулей
mx_count = 0
# Значение x, соответствующее текущему максимуму
mx = 0
# Перебираем все возможные значения x от 1 до 3000
for x in range(1,3001):
    # Вычисляем выражение и переводим результат в четверичную систему счисления
    res = cc4(5**127 + 5**27 - x)
    # Если количество нулей больше ранее найденного максимума
```

```
    if res.count("0") > mx_count:
        # Запоминаем новое максимальное количество нулей
        mx_count = res.count("0")
        # Сохраняем соответствующее значение x
        mx = x
# Выводим найденное значение x
print(mx)
```

Ответ: 2666

Задача 14.2 (Дальний восток)

Значение арифметического выражения

$$4^{265} + 4^{150} + 4^{50} - x,$$

где x – целое положительное число, не превышающее 3003, записали в 4-ричной системе счисления. Определите наименьшее значение x, при котором количество нулей в 4-ричной записи числа, являющегося значением данного арифметического выражения, равно 218. В ответе запишите число в десятичной системе счисления.

Решение

Для каждого x мы вычисляем значение выражения, а затем переводим его в 4-ричную систему счисления, подсчитывая количество нулей. Первый найденный x, для которого количество нулей равно 218, и будет наименьшим.

```
# Перебираем x от 1 до 3003
for x in range(1, 3004):
    val = 4 ** 265 + 4 ** 150 + 4 ** 50 - x

    # Считаем количество нулей в 4-ричной записи числа val
    zeros = 0
    s = val
    while s > 0:
        if s % 4 == 0:
            zeros += 1
        s //= 4

    # Если количество нулей равно 218, мы нашли наименьший x
    if zeros == 218:
        print(x)
        break
```

Ответ: 256

Задача 14.3 (Дальний восток)

Значение арифметического выражения

$$7^{170} + 7^{100} - x,$$

где x – целое положительное число, не превышающее 2030, записали в 7-ричной системе счисления. Определите наименьшее значение x, при котором количество нулей в 7-ричной записи числа, являющегося значением данного арифметического выражения, равно 72. В ответе запишите число в десятичной системе счисления.

Решение

Требуется найти такое значение переменной x, при котором в семеричной записи числа

$$7^{170} + 7^{100} - x$$

содержится ровно 72 цифры 0. Для этого будем последовательно перебирать все возможные значения x из заданного диапазона, вычислять значение выражения, переводить полученное число в систему счисления с основанием 7 и подсчитывать количество нулей в полученной записи.

Поскольку перебор выполняется в порядке возрастания, найденное значение будет первым подходящим значением из заданного диапазона.

```
# Функция перевода числа из десятичной системы счисления в семеричную
def cc7(x):
    # Строка для накопления результата перевода
    s = ""
    # Выполняем перевод, пока число больше нуля
    while x > 0:
        # Добавляем очередную цифру семеричной записи в начало строки
        s = str(x % 7) + s
        # Переходим к следующему шагу алгоритма деления
        x = x // 7
    # Возвращаем полученную семеричную запись
    return s
# Перебираем все возможные значения x от 1 до 2030
for x in range(1,2031):
    # Вычисляем выражение и переводим результат в семеричную систему счисления
    res = cc7(7**170 + 7**100 - x)
    # Проверяем количество нулей в полученной записи
    if res.count("0") == 72:
        # Выводим найденное значение x
        print(x)
        # Завершаем перебор после нахождения первого подходящего значения
        break
```

Ответ: 49

Задача №15

Задача 15.1 (Дальний восток)

Заданы два отрезка $P = [3; 18]$ и $Q = [11; 24]$, лежащие на числовой прямой.

Укажите наименьшую возможную длину такого отрезка A, для которого логическое выражение

$$((x \in P) \wedge (x \in Q)) \to (x \in A)$$

истинно (т. е. принимает значение 1) при любом значении переменной x.

Решение

Преобразуем исходное выражение, раскрыв импликацию и применив закон де Моргана:

$$(x \notin P) \vee (x \notin Q) \vee (x \in A)$$

Приравняем известную часть к 0 и получим:

$$(x \notin P) \vee (x \notin Q) = 0$$

$$(x \in P) \wedge (x \in Q) = 1$$

Значит, выражение даёт ложь в известной части при x, принадлежащих одновременно обоим отрезкам. Нам нужно, чтобы все значения x из данного пересечения принадлежали отрезку A, поэтому необходимо взять $A = [11; 18]$.

Тогда минимальная возможная длина данного отрезка равна:

$$18 - 11 = 7$$

.

Ответ: 7

Задача 15.2 (Дальний восток)

На числовой прямой даны три отрезка: $P = [55; 57]$, $O = [22; 98]$ и $D = [20, 150]$. Укажите наименьшую возможную длину такого отрезка A, что логическое выражение

$$(x \notin A) \to ((x \in P) \vee (x \notin O) \vee (x \notin D))$$

истинно (т.е. принимает значение 1) при любом значении переменной x.

Решение

Преобразуем исходное выражение, раскрыв импликацию:

$$(x \in A) \vee ((x \in P) \vee (x \notin O) \vee (x \notin D))$$

Приравняем известную часть к 0 и получим:

$$(x \in P) \vee (x \notin O) \vee (x \notin D) = 0$$

$$(x \notin P) \wedge (x \in O) \wedge (x \in D) = 1$$

Значит, выражение даёт ложь в известной части при x, принадлежащих одновременно отрезкам O и D, но не принадлежащих отрезку P. Таким значениям на числовой прямой соответствуют промежутки [22; 55) и (57; 98]. Нам нужно, чтобы все эти значения принадлежали отрезку A, поэтому необходимо взять $A = [22; 98]$. Тогда минимальная возможная длина данного отрезка равна $98 - 22 = 76$.

Ответ: 76

Задача 15.3 (Сибирь)

Заданы два промежутка $B = (390; 750)$ и $C = [210; 575]$, лежащие на числовой прямой.

Укажите наименьшую возможную длину такого отрезка A, для которого логическое выражение

$$((x \notin B) \wedge (x \in C)) \to (((x \notin A) \wedge (x \in C)) \to (x \in B))$$

истинно (т. е. принимает значение 1) при любом значении переменной x.

Решение

Операции будет записывать в следующем виде:

$$(x \in B) - B, (x \in A) - A, (x \in C) - C$$

Для начала упростим выражение

$$(B \vee \neg C) \vee ((\neg A \wedge C) \to B)$$

$$(B \vee \neg C) \vee ((A \vee \neg C) \vee B)$$

$$A \vee B \vee \neg C$$

Теперь рассмотрим ситуацию когда известная часть $(B \vee \neg)$ равна 0, тогда ее отрицание $(\neg \wedge)$ должно равняться 1

$\neg B \wedge C$

подходящий должен быть в и не быть в . Это отрезок [210; 390]. Его длина: $390 - 210 = 180$

Ответ: 180

Задача №16

Задача 16.1 (Дальний восток)

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

$F(n) = 1$, при $n = 1$;

$F(n) = (n-1) \cdot F(n-1)$, если $n > 1$.

Определите значение выражения:

$$(F(10938)/2 - F(10937))/F(10936)$$

Решение

Решение руками:

$$\frac{\frac{F(10938)}{2} - F(10937)}{F(10936)} = \frac{\frac{10937 \cdot F(10937)}{2} - F(10937)}{F(10936)} = \frac{10935 \cdot F(10937)}{2 \cdot F(10936)} =$$

$$= \frac{10935 \cdot 10936 \cdot F(10936)}{2 \cdot F(10936)} = 59792580$$

Ответ: 59792580

Задача №17

Задача 17.1 (Дальний восток)

В файле содержится последовательность целых чисел. Её элементы могут принимать целые значения от −100000 до 100000 включительно. Определите количество троек последовательности, в которых сумма элементов делится на максимальный элемент последовательности, оканчивающийся на 3. В ответе запишите количество найденных троек, затем максимальную из сумм элементов таких троек. В данной задаче под тройкой подразумевается три идущих подряд элемента последовательности.

Решение

```
# Открываем файл '17.txt' для чтения
f = open('17.txt')
# Читаем все строки из файла, преобразуем каждую в целое число
a = [int(x) for x in f]
# Находим максимальное число среди всех элементов списка a,
# у которых последняя цифра (модуль числа) равна 3
m3 = max(x for x in a if abs(x)%10==3)
maxi = -10**10
c = 0
# Перебираем все тройки подряд идущих чисел в списке a
for i in range(len(a)-2):
    # Берём тройку чисел: a[i], a[i+1], a[i+2]
    t = a[i:i+3]
    # Проверяем, делится ли сумма элементов тройки нацело на m3
    if sum(t) % m3 == 0:
        # Если да, то увеличиваем счётчик на 1
        c += 1
        # Обновляем максимальную сумму тройки,
        # если текущая сумма больше сохранённой
        maxi = max(maxi,sum(t))

print(c,maxi) # Выводим полученные результаты на экран
```

Задача №18

Задача 18.1 (Дальний восток)

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.

Решение

Нам дано поле 20 на 20, создадим рядом еще одно поле такого же размера (ячейки $A23 : T42$) . В левую верхнюю клетку нового поля, записываем значение из левой верхней клетки исходного поля – 27.

Сначала просимулируем ход вправо. Для этого в ячейке B23 запишем формулу и протянем до конца строки:

$$=\text{A23}+\text{B1}$$

Теперь просимулируем ход вниз. Для этого в ячейке A24 запишем формулу и протянем до нижнего конца таблицы:

$$=\text{A23}+\text{A2}$$

Найдем максимальное значение суммы. Рассмотрим ячейку $B2$, в нее мы можем попасть из $B1$ и $A2$, тогда, чтобы в этой клетке суммы была максимальной, необходимо выбрать максимальную сумму из тех двух клеточек, из которых можем попасть в эту. В ячейку $B24$ запишем формулу:

$$=\text{МАКС(A24;B23)}+\text{B2}$$

После растягивания формулы, стенки в новом поле пропали, для того чтобы их вернуть скопируем изначальное поле и на новое вставим следующим образом: ПКМ – Специальная вставка – форматирование. Стенки вернулись. Теперь в ячеках, которые находятся правее и/или ниже стенок необходимо заменить формулы. В ячейках которые ниже стенок напишем формулы по аналогии с

теми, которые мы записывали в верхней строке, а в ячейках, которые правее стенок – по аналогии с самым левым столбцом.

Заметим, что в периметр с M32 по N37 робот не сможет попасть, так как робот движется только вниз и вправо, поэтому выделим этот периметр и сотрем этот прямоугольник.

Теперь найдем ячейки, где робот может остановиться: O42, T42, S40, Q34, и выделим их красным цветом. Теперь напишем в свободной ячейке формулу:

$$=\text{МАКС}(\text{O42};\text{Q34};\text{S40};\text{T42})$$

Посмотрим на нее, увидим первый ответ на задание: 2362.

Для того чтобы найти минимальное заменим все МАКС на МИН. Сделать это можно с помощью функции Найти и Заменить. Увидим значение минимума в той же ячейке, где смотрели максимум: 1205.

Ответ: 2362 1205

Задача №19-21

Задача 19.1 (Дальний восток)

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) 3 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: (23, 30), (20, 33), (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 173. Победителем считается игрок, сделавший последний ход, то есть первым получивший такую игровую позицию, при которой в двух кучах суммарно 173 камня или больше. В начальный момент в первой куче 27 камней, во второй куче – S камней; $1 \leq S \leq 150$.

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 173 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 (27 камней). При каких S будет выполняться неравенство $2 \cdot 2 \cdot S + 27 \geq 173$?

$4 \cdot S + 27 \geq 173$

$4 \cdot S \geq 146$

$S \geq 36.5$

Минимальное целое значение S, удовлетворяющее этому условию, равно 37.

Решение программой

Для решения на Python используем рекурсию с проверкой всех возможных ходов (+3, ·2), чтобы определить, чья это выигрышная позиция. Функция возвращает 0, если игра уже завершена (камней в сумме ≥ 173), 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. Рассмотрим все ходы Пети и проверим, что после хотя бы одного из них Ваня мог выиграть. Также важно учесть, что Петя не должен гарантированно проигрывать, иначе нельзя назвать его ход неудачным.

При переборе возможных значений программа выводит минимум 37.

```
from functools import lru_cache


@lru_cache(None)
def game(first_heap, second_heap):
    if first_heap + second_heap >= 173:
        return 0  # Прекращаем игру
    moves = [
        game(first_heap + 3, second_heap),
        game(first_heap, second_heap + 3),
        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, 151):
    # Если в данной позиции после неудачного хода Пети возможен выигрыш Вани
    # и Петя не был в гарантированно проигрышной позиции
    if ((game(27 + 3, s) == 1 or game(27, s + 3) == 1
         or game(27 * 2, s) == 1 or game(27, s * 2) == 1)
            and game(27, s) != -1):
        print(s)  # Вывод минимального значения S
        break
```

Ответ: 37

Задача 20.1 (Дальний восток)

Найдите наименьшее и наибольшее 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

Для решения на Python используем рекурсию с проверкой всех возможных ходов (+3, ·2), чтобы определить, чья это выигрышная позиция. Функция возвращает 0, если игра уже завершена (камней в сумме ≥ 173), 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 наименьшее и наибольшее.

```
from functools import lru_cache

@lru_cache(None)
def game(first_heap, second_heap):
    if first_heap + second_heap >= 173:
        return 0  # Прекращаем игру
    moves = [
        game(first_heap + 3, second_heap),
        game(first_heap, second_heap + 3),
        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, 151):
    if game(27, s) == 2:
        print(s)
```

Ответ: 36 71

Задача 21.1 (Дальний восток)

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.

Решение

Для решения на Python используем рекурсию с проверкой всех возможных ходов (+3, * 2), чтобы определить, чья это выигрышная позиция.

Функция возвращает 0, если игра уже завершена (камней в сумме $\geq$ 173), 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 Ваня гарантированно побеждает своим первым или вторым ходом. Программа перебирает значения S и выводит минимальное подходящее число.

```
from functools import lru_cache

@lru_cache(None)
def game(first_heap, second_heap):
    if first_heap + second_heap >= 173:
        return 0  # Прекращаем игру
    moves = [
        game(first_heap + 3, second_heap),
        game(first_heap, second_heap + 3),
        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)

# Поиск минимального значения S для выигрыша Вани первым или вторым ходом
for s in range(1, 151):
    if game(27, s) == -2:
        print(s)  # Вывод минимального значения S
        break
```

Ответ: 67

Задача 19.2 (Центр)

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 184. Победителем считается игрок, сделавший последний ход, то есть первым получивший такую игровую позицию, при которой в двух кучах суммарно 184 камня или больше. В начальный момент в первой куче 28 камней, во второй куче – S камней; $1 \leq S \leq 153$.

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

Для решения задачи используем рекурсивную функцию, которая будет определять состояние каждой игровой позиции. Позиция задаётся двумя числами: количеством камней в первой куче и количеством камней во второй куче. Для каждой позиции функция будет возвращать специальное число, показывающее, является ли данная позиция выигрышной или проигрышной для игрока, который должен делать ход.

Будем рассматривать все возможные позиции как вершины игрового дерева. Из каждой позиции можно сделать четыре различных хода:

1. Увеличить первую кучу на 4 камня.
2. Увеличить вторую кучу на 4 камня.
3. Увеличить первую кучу в 2 раза.
4. Увеличить вторую кучу в 2 раза.

Для каждой новой позиции снова запускается та же функция, поэтому решение строится рекурсивно.

```
from functools import lru_cache # Подключаем механизм запоминания результатов
@lru_cache(None) # Сохраняем результаты вычислений для одинаковых позиций

def game(first_heap, second_heap):
    if first_heap+second_heap >= 184: # Если выполнено условие завершения игры
        return 0 # Возвращаем признак конечной позиции
    # Формируем список всех возможных ходов из текущей позиции
    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)]
    # Оставляем только те ходы, которые ведут соперника в невыгодную позицию
    petya_win = [i for i in moves if i <= 0]
    if petya_win: # Если существует выигрышный ход
        return -max(petya_win) + 1 # Возвращаем номер выигрышной позиции
    return -max(moves) # Если все ходы выгодны сопернику

for i in range(1,154): # Перебираем все возможные значения второй кучи
    # Проверяем все варианты первого хода Пети
    if (game(28+4,i) == 1 or game(28*2,i) == 1
        or game(28,i+4) == 1 or game(28,i*2) == 1):
        print(i) # Выводим найденное значение
        break # Завершаем перебор после первого найденного ответа
```

Ответ: 39

Задача 20.2 (Центр)

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.

Решение

```
# Импортируем кэширование
from functools import lru_cache
@lru_cache(None)

# Функция игры
def game(first_heap, second_heap):
    # Условие выигрыша
    if first_heap+second_heap >= 184:
        # Завершаем партию
        return 0
    # Всевозможные ходы в партии
    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)]
    # Может ли Петя победить
    petya_win = [i for i in moves if i <= 0]
    if petya_win:
        # Победа Пети
        return -max(petya_win) + 1
    # Победа Вани
    return -max(moves)

for i in range(1,154):
    if game(28,i) == 2:
        print(i)
```

Ответ: 38

Задача 21.2 (Центр)

Найдите наименьшее и наибольшее 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

```
# Импортируем кэширование
from functools import lru_cache
@lru_cache(None)

# Функция игры
def game(first_heap, second_heap):
    # Условие выигрыша
    if first_heap+second_heap >= 184:
        # Завершаем партию
        return 0
    # Всевозможные ходы в партии
    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)]
    # Может ли Петя победить
    petya_win = [i for i in moves if i <= 0]
    if petya_win:
        # Победа Пети
        return -max(petya_win) + 1
    # Победа Вани
    return -max(moves)

for i in range(1,154):
    if game(28,i) == -2:
        print(i)
```

Ответ: 71

Задача №23

Задача 23.1 (Дальний восток)

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

1. Прибавь 2
2. Замени 1 на 3

Первая из этих команд увеличивает число на экране на 2. Вторая команда может применяться только к числу, в десятичной записи которого содержится хотя бы одна цифра «1», и действует, заменяя все цифры «1» в записи числа на цифры «3» (например, число 11 превратится в 33, а 21 - в 23).

Программа для исполнителя – это последовательность команд.

Сколько существует программ, для которых при исходном числе 10 результатом является число 44?

Решение

```
def f(a, b):
    if a > b:
        return 0
    if a == b:
        return 1
    t = f(a + 2, b)
    if '1' in str(a):
        t += f(int(str(a).replace('1', '3')), b)
    return t

print(f(10, 44))
```

Ответ: 6

Задача №24

Задача 24.1 (Дальний восток)

Текстовый файл состоит из римских цифр I, V, X, L, C, D, M и знаков арифметических операций «+» и «-» (сложение и вычитание). Определите максимальное количество символов в непрерывной последовательности, которая является корректным арифметическим выражением с корректными римскими числами.

В ответе укажите количество символов.

Примечание: Римские числа записываются с помощью комбинации семи основных символов латинского алфавита: I, V, X, L, C, D, M. У каждого из них фиксированное значение:

I	1
V	5
X	10
L	50
C	100
D	500
M	1000

Символы I, X, C, M могут повторяться не более 3 раз подряд, а комбинации для 4, 9, 40, 90, 400, 900 записываются через «отнимание» (например, IV, IX, XL, XC, CD, CM). Числа записываются слева направо от большего значения к меньшему.

Если символ с меньшим значением стоит после символа с большим или равным значением, их значения складываются.

Если символ с меньшим значением стоит перед символом с большим значением, его значение вычитается из значения большего. Однако это правило применяется только для определённых комбинаций:

I перед V или X (IV = 4, IX = 9)

X перед L или C (XL = 40, XC = 90)

C перед D или M (CD = 400, CM = 900)

Решение

```
from re import finditer

# Считываем строку из файла
s = open('24_roman_1M.txt').readline()

# Регулярное выражение для корректного римского числа
# Проверка (?=[IVXLCDM]) не позволяет числу быть пустым
roman = r'(?=[IVXLCDM])(M{0,3}(CM|CD|D?C{0,3})(XC|XL|L?X{0,3})(IX|IV|V?I{0,3}))'

# Регулярное выражение для арифметического выражения,
```

```
# состоящего из римских чисел и знаков + или -
expr = f'{roman}([+-]{roman})+'

# Максимальная длина найденного выражения
mx = 0

# Перебираем все найденные выражения
for match in finditer(expr, s):

    # Получаем найденную подстроку
    cur = match.group()

    # Обновляем ответ
    mx = max(mx, len(cur))

# Выводим максимальную длину
print(mx)
```

Ответ: 53

Задача 24.2 (Дальний восток)

Текстовый файл состоит из заглавных букв латинского алфавита. Определите в прилагаемом файле максимальное количество идущих подряд символов, образующих корректную римскую запись десятичного числа. Если найдено несколько таких последовательностей, укажите ту, которая обозначает наименьшее десятичное число. В ответе укажите найденную последовательность римских цифр.

Примечание: Римские числа записываются с помощью комбинации семи основных символов латинского алфавита: I, V, X, L, C, D, M. У каждого из них фиксированное значение:

I	1
V	5
X	10
L	50
C	100
D	500
M	1000

– Символы V, L, D никогда не повторяются;
– Символы I, X, C, M могут повторяться не более 3 раз подряд.
– если меньшая цифра стоит слева от большей, её значение вычитается (это касается только пар IV, IX, XL, XC, CD, CM; вычитаемое не может быть меньше одной десятой вычитателя);
– если цифра стоит справа от большей или равной, их значения складываются;

– цифры в записи числа располагаются слева направо в порядке убывания их значений (за исключением случаев вычитания).
Например, римская запись MMXXVI корректна и обозначает число 2026.

Решение

Решение с помощью регулярных выражений

```
from re import finditer

# Функция перевода римского числа в десятичное
def romanToInt(s):
    # Словарь соответствия римских символов и их значений
    roman = {
        "M": 1000,
        "CM": 900,
        "D": 500,
        "CD": 400,
        "C": 100,
        "XC": 90,
        "L": 50,
        "XL": 40,
        "X": 10,
        "IX": 9,
        "V": 5,
        "IV": 4,
        "I": 1
    }
    ans = 0 # Итоговое десятичное число
    i = 0 # Текущая позиция в строке

    # Проходим по всем символам римского числа
    while i < len(s):
        # Если можно образовать специальную пару
        # (IV, IX, XL, XC, CD, CM),
        # прибавляем её значение
        if (i != len(s) - 1) and (s[i] + s[i + 1]) in roman:
            ans += roman[s[i] + s[i + 1]]
            i += 2
        else: # Иначе прибавляем значение одного символа
            ans += roman[s[i]]
            i += 1
    return ans
```

```
f = open("24_roman_1M.txt") # Считываем строку из файла
s = f.readline()

# Регулярное выражение для непустого корректного римского числа
roman = r'(?=[IVXLCDM])(M{0,3}(CM|CD|D?C{0,3})(XC|XL|L?X{0,3})(IX|IV|V?I{0,3}))'

mx = 0 # Максимальная длина найденной римской записи

# Искомая римская запись
ans = ''

# Ищем все корректные римские числа в строке
for match in finditer(roman, s):

    # Получаем найденное число
    cur = match.group()

    # Если нашли более длинную запись,
    # обновляем ответ
    if len(cur) > mx:
        mx = len(cur)
        ans = cur

    # Если длины совпадают,
    # выбираем число с меньшим десятичным значением
    elif len(cur) == mx:
        if romanToInt(ans) > romanToInt(cur):
            ans = cur

# Выводим найденную римскую запись
print(ans)
```

Комбинированное решение Python + Excel

Найдём максимальное число R, которое получится представить в виде римской записи для данных цифр:

- Цифру M, обозначающую 1000, нельзя использовать больше 3-х раз, а цифры с большим значением отсутствуют;

- Если мы просто сложим все оставшиеся цифры из таблицы, взяв их максимальное допустимое количество раз (3 раза I, 1 раз V, 3 раза X, 1 раз L, 3 раза C, 1 раз D), то получим 999.

Так как цифру M можно использовать не более 3-х раз, R = 3 * 1000 + 999 = 3999.

Теперь открываем пустую электронную таблицу. Необходимо записать в ячейке A1 число 1. Далее в A2 прописываем формулу A1 + 1 и протягиваем данную ячейку до тех пор, пока не получим столбец с числами от 1 до 3999.

Затем в ячейке B1 нужно записать и протянуть формулу РИМСКОЕ(A1) до строки с числом 3999. Таким образом мы получили столбец, в котором находятся все возможные вариации римских

чисел, встречаемых в файле к задаче. Необходимо выделить второй столбец, скопировать его, после чего создать пустой текстовый файл и вставить в него найденные числа.

Приступаем к написанию программы для решения задачи: открываем файл со строкой и считываем её в отдельную переменную. Теперь открываем файл с римскими числами и заносим каждое из них в список при помощи генератора, предварительно применив метод strip(), чтобы убрать символы '\n'. Далее создадим 2 переменные – max_len = 0, в которой будет храниться максимальная длина найденного числа, и res_str = '', в которой сохранится сама римская запись, подходящая нам в качестве ответа. Остаётся перебрать все римские числа из списка, проверяя их длину и наличие в строке. Если условие выполняется, то необходимо изменять переменные, указанные выше. Важный момент: знак '>' позволяет учесть то, что в случае нахождения чисел одинаковой длины, нужно взять меньшее, ведь все элементы списка расположены в порядке возрастания их числовых значений. В конце ответ выводится на экран.

```
f = open("24_roman_1M.txt")
s = f.readline()
# Открываем файл с римскими числами и помещаем их в список
numbers = [i.strip() for i in open('numbers.txt')]
# Задаём переменную, которая будет хранить максимальную длину найденной записи
max_len = 0
# Задаём переменную, которая будет хранить искомую запись в виде строки
res_str = ''
# Перебираем все возможные числа из списка
for n in numbers:
    # Если длина текущего числа больше максимальной и само число есть в строке,
    if (len(n) > max_len) and (n in s):
        max_len = len(n)  # обновляем значение максимальной длины
        res_str = n  # обновляем значение искомой римской записи
print(res_str)  # Выводим ответ на экран
```

Ответ: MMCCCXXII

Задача №25

Задача 25.1 (Дальний восток)

Пусть M – сумма минимального и максимального простых натуральных делителей целого числа, не считая самого числа. Если таких делителей у числа нет, то значение M считается равным нулю.

Напишите программу, которая перебирает целые числа, большие 5 800 000, в порядке возрастания и ищет среди них такие, для которых M больше 80 000 и является палиндромом, т.е. одинаково читается слева направо и справа налево.

В ответе запишите в первом столбце таблицы первые пять найденных чисел в порядке возрастания, а во втором столбце – соответствующие им значения M. Например, для числа 298 $M = 2 + 149 = 151$.

Решение

Будем последовательно перебирать числа из заданного диапазона и для каждого числа находить его простые делители. Из найденных простых делителей определим минимальный и максимальный, после чего вычислим их сумму. Затем проверим дополнительные условия задачи: полученная сумма должна быть больше 80000 и являться палиндромом. Все подходящие числа будем выводить сразу после нахождения. После получения первых пяти результатов перебор будет остановлен.

```
# Функция для поиска всех делителей числа, кроме 1 и самого числа
def divs(x):
    # Создаём множество для хранения делителей
    d = set()
    # Перебираем возможные делители от 2 до корня числа
    for i in range(2,int(x**0.5)+1):
        # Проверяем, делится ли число без остатка
        if x % i == 0:
            # Добавляем найденный делитель
            d.add(i)
            # Добавляем парный делитель
            d.add(x//i)
    # Возвращаем отсортированный список делителей
    return sorted(d)
# Функция проверки числа на простоту
def is_prime(x):
    # Перебираем возможные делители от 2 до корня числа
    for i in range(2,int(x**0.5)+1):
        # Если найден делитель, число не является простым
        if x % i == 0:
            return False
    # Возвращаем результат проверки
    return x > 1
```

```
# Функция поиска минимального и максимального простых делителей
def find_m(x):
    # Начальное значение для поиска минимального простого делителя
    mn_d = 10**10
    # Начальное значение для поиска максимального простого делителя
    mx_d = -1
    # Перебираем все делители числа
    for i in divs(x):
            # Если делитель простой, обновляем минимум
            if is_prime(i):
                mn_d = min(mn_d, i)
            # Если парный делитель простой, обновляем максимум
            if is_prime(x//i):
                mx_d = max(mx_d, x//i)
            # Если оба значения найдены, возвращаем результат
            if mn_d < 10**10 and mx_d > 0:
                return mn_d, mx_d
    # Если подходящие значения не найдены, возвращаем специальные значения
    return -1,-1

count = 0 # Счётчик найденных чисел
# Перебираем числа из заданного диапазона
for i in range(5_800_001,10_000_000):
    # Находим минимальный и максимальный простые делители
    mn_d, mx_d = find_m(i)
    # Проверяем, что значения были найдены
    if mn_d > 0 and mn_d > 0:
        # Вычисляем сумму найденных делителей
        m = mn_d + mx_d
        # Проверяем условия задачи
        if m > 80_000 and str(m) == str(m)[::-1]:
            # Выводим найденное число и вычисленное значение
            print(i,m)
            # Увеличиваем количество найденных результатов
            count += 1
            # После нахождения пяти чисел завершаем перебор
            if count == 5:
                break
```

Ответ:

5800020 96669
5801724 161161
5804045 89298

5805494 93639
5806156 1451541

Задача 25.2 (Дальний восток)

Пусть M – сумма минимального и максимального простых натуральных делителей целого числа, не считая самого числа. Если таких делителей у числа нет, то значение M считается равным нулю.

Напишите программу, которая перебирает целые числа, большие 7 800 000, в порядке возрастания и ищет среди них такие, для которых M больше 100 000 и является палиндромом, т.е. одинаково читается слева направо и справа налево. В ответе запишите в первом столбце таблицы первые пять найденных чисел в порядке возрастания, а во втором столбце – соответствующие им значения M.

Например, для числа 298 $M = 2 + 149 = 151$.

Решение

Сначала вводится функция $is\_prime(n)$, которая проверяет число на простоту. Она перебирает возможные делители от 2 до корня из числа: если находится хотя бы один делитель, то число нам не подходит, возвращаем ложь. Далее вводится функция $find\_m(n)$, которая находит значение M для текущего числа. Для этого она проверяет все возможные делители: первые найденные простые числа i и $n//i$ будут подходить, так как при продолжении цикла значения будут только уменьшаться. Если минимальный или максимальный простые делители не будут найдены, то функция вернёт -1. В конце остаётся перебрать числа, превышающие 7800000, и выбрать из них первые 5, для которых число $M > 100000$, а также является палиндромом. Чтобы реализовать вторую проверку, используем разворот строки: если число M в виде строки совпадает со своей перевёрнутой записью, значит оно является палиндромом.

```
def is_prime(n):  # Функция проверки числа на простоту
    for i in range(2, int(n ** 0.5) + 1):  # Перебираем возможные делители числа n
        if n % i == 0:  # Если число n поделилось на i,
            return False  # возвращаем ложь - оно не простое
    return n > 1  # Возвращаем истину, если число не является 1,
                  #при этом цикл выше закончился


def find_m(n):  # Функция нахождения числа M
    min_d = max_d = 0
    for i in range(2, int(n ** 0.5) + 1):  # Перебираем возможные делители числа n
        if n % i == 0:  # Если число n поделилось на i, то проверяем основной делитель
                        # и парный ему на простоту:
            if is_prime(i) and min_d == 0:  # Если i - простое число,
                                            #при этом min_d ещё не найден,
                min_d = i  # записываем его в качестве мин. простого делителя
            if is_prime(n // i) and max_d == 0:  # Если n // i - простое число,
```

```
                                                # при этом max_d ещё не найден,
                max_d = n // i  # записываем его в качестве макс. простого делителя
        if min_d > 0 and max_d > 0:  # Если искомые делители нашлись,
            return min_d + max_d  # возвращаем значение m
    return -1  # иначе возвращаем значение -1


c = 0  # Задаём счётчик количества выведенных строк
# Перебираем числа, превышающие 7_800_000
for x in range(7_800_000 + 1, 10_000_000_000):
    m = find_m(x)  # Находим число M
    # Если m больше 80000 и является палиндромом,
    if m > 100000 and str(m) == str(m)[::-1]:
        print(x, m)  # выводим искомые значения на экран
        c += 1  # увеличиваем счётчик на 1
    if c == 5:  # Если количество выведенных строк равно 5
        break  # останавливаем перебор
```

Ответ:

7806926 105501
7809177 2603062
7812174 1302031
7814182 3907093
7814209 601106

Задача 25.3 (Дальний восток)

Пусть S – сумма всех натуральных делителей целого числа. Если таких делителей у числа нет, то значение S считается равным нулю. Напишите программу, которая перебирает целые числа, большие 8 494 154, в порядке возрастания и ищет среди них такие, у которых есть ровно 4 различных натуральных делителя, а значение S является палиндромом (то есть читается слева-направо и справа- налево одинаково).

В ответе запишите в первом столбце таблицы первые 5 найденных чисел в порядке возрастания, а во втором столбце – соответствующие им значения S. Например, для числа 20 $S = 1 + 2 + 4 + 5 + 10 + 20 = 42$.

Решение

```
# Счётчик найденных чисел
count = 0

# Перебираем числа из заданного диапазона
for x in range(8_494_154 + 1, 10_000_000):
    d = set()  # множество для хранения делителей
```

```
    # Флаг для преждевременного выхода из цикла,
    # если делителей стало больше 4
    flag = False

    # Перебираем всевозможные делители
    for i in range(1, int(x ** 0.5) + 1):
        # Проверяем, делится ли число без остатка
        if x % i == 0:
            # Добавляем найденный делитель
            d.add(i)
            # Добавляем парный делитель
            d.add(x // i)

            if len(d) > 4:  # если делителей больше чем 4
                flag = True
                break  # останавливаем перебор
    if flag:
        continue  # переходим к следующему числу

    if len(d) == 4:  # если делителей ровно 4
        S = sum(d)

        if str(S) == str(S)[::-1]:  # если S - палиндром
            print(x, S)
            # Увеличиваем количество найденных результатов
            count += 1
            # После нахождения пяти чисел завершаем перебор
            if count == 5:
                break
```

Ответ:

8495303 8864688
8519453 8889888
8580323 8616168
8602333 8812188
8602883 8610168

Задача 25.4 (Сибирь)

Пусть R – сумма 4 наибольших делителей числа. Напишите программу, которая перебирает целые числа, большие 1151 996, в порядке возрастания и ищет среди них такие, для которых R является простым числом и палиндромом, т.е. одинаково читается слева направо и справа налево. В ответе запишите в первом столбце таблицы первые пять найденных чисел в порядке

возрастания, а во втором столбце – соответствующие им значения R. Количество строк в таблице для ответа избыточно.

Решение

Сначала вводится функция is_prime(n), которая проверяет число на простоту. Она перебирает возможные делители от 2 до корня из числа: если находится хотя бы один делитель, то число нам не подходит, возвращаем ложь.

Далее вводится функция find_divs(n), находящая отсортированный список делителей числа x. Для этого она проверяет все возможные делители, выполняя цикл от 1 до корня проверяемого числа и добавляя все найденные значения во множество divs. Поскольку для поиска числа R нам нужны только 4 наибольших делителя, добавим дополнительное условие if len(divs) == 8: break — оно прекратит поиск остальных делителей, когда 4 наибольших уже будут найдены.

В конце остаётся перебрать числа, превышающие 1151996, и выбрать среди них первые 5, для которых R является простым числом, а также палиндромом. Чтобы реализовать вторую проверку, используем разворот строки: если число R в виде строки совпадает со своей перевёрнутой записью, значит оно является палиндромом.

```
def is_prime(n):  # Функция проверки числа на простоту
    for i in range(2, int(n ** 0.5) + 1):  # Перебираем возможные делители числа n
        if n % i == 0:  # Если число n поделилось на i,
            return False  # возвращаем ложь - оно не простое
    # Возвращаем истину, если число не является 1, при этом цикл выше завершился
    return n > 1


def find_divs(n):  # Функция нахождения делителей числа n
    divs = set()  # Множество делителей числа n
    for i in range(1, int(n ** 0.5) + 1):  # Перебираем возможные делители числа n
        if n % i == 0:  # Если число n поделилось на i,
            divs.add(i)  # добавляем во множество основной делитель
            divs.add(n // i)  # добавляем во множество парный делитель
            # Если количества делителей хватает для составления числа R,
            if len(divs) == 8:
                break  # завершаем цикл
    return sorted(divs)  # Возвращаем отсортированный список делителей числа n


c = 0  # Задаём счётчик количества выведенных строк
# Перебираем числа, превышающие 1_151_996
for x in range(1_151_996 + 1, 10_000_000_000):
    d = find_divs(x)  # Находим делители числа x
    if len(d) >= 4:  # Если количества делителей хватает для составления числа R,
        r = sum(d[-4:])  # задаём число R
        # Если R - простое число, являющееся палиндромом,
```

```
        if str(r) == str(r)[::-1] and is_prime(r):
            print(x, r)  # выводим число x и соответствующее ему значение r
            c += 1  # увеличиваем счётчик на 1
    if c == 5:  # Если количество выведенных строк равно 5
        break  # останавливаем перебор
```

Ответ:

1803148 3155513
1991908 3485843
2310962 3470743
4263796 7461647
4349236 7611167

Задача №26

Задача 26.1 (Дальний восток)

Производите детского питания производит оптовую закупку винограда у фермерских хозяйств региона. Используется виноград двух типов A и B – светлый и темный. На закупку выделена определённая сумма денег.

У фермерских хозяйств каждая партия винограда имеет свою стоимость в рублях. На выделенные деньги необходимо приобрести как можно больше винограда типа A независимо от партии и не менее одной партии винограда B. Если у фермерских хозяйств закончится виноград A, то на оставшиеся деньги необходимо приобрести как можно больше винограда B.

Если существует несколько способов закупить максимальное количество винограда следует выбрать такой при котором будет приобретено как можно больше винограда A и при этом потрачено наименьшее количество денег.

Известны выделенная для закупки сумма, а также количество и цена различных партий винограда у фермерских хозяйств. Определите для максимального количества купленного винограда наибольшее возможное число партий винограда A, а так же оставшуюся сумму денег.

Входные данные.

Первая строка входного файла содержит два целых числа: N — общее количество партий винограда у фермерских хозяйств и M — сумма денег, выделенных на закупку (в рублях). Каждая из следующих N строк описывает одну партию и содержит целое число (стоимость партии в рублях) и один символ (латинская буква A или B), определяющий тип винограда. Все данные в строках входного файла отделены одним пробелом.

Выходные данные.

В ответе запишите два целых числа: сначала наибольшее возможное число партий винограда A, затем оставшуюся сумму денег.

Пример входного файла:

6 110
40 B
50 A
50 B
30 B
20 A
10 B

В данном случае можно купить не более четырех партий винограда, из них не более двух партий типа A. Минимальная цена такой покупки 110 рублей (покупаем партии 10 B 20 A 30 B 50 A).

Остнется 0 рублей. Ответ: 2 0.

Решение

```
f = open("26.txt")
n, m = map(int, f.readline().split())
# создаём список для хранения всех партий в виде [цена, "тип"]
p = []
for i in range(n):
    # считываем стоимость и тип, разделенные пробелом
    cost, p_type = f.readline().split()
    p.append([int(cost), p_type])
# сортируем все партии по возрастанию цены,
# чтобы взять как можно больше штук
p.sort()

# находим самую дешёвую партию типа B,
# чтобы сразу закрыть обязательное условие
for ind in range(len(p)):
    cost, p_type = p[ind]
    if p_type == "B":
        # останавливаемся, индекс ind первой партии B сохранён
        break

s = p[ind][0]  # стартовая сумма равна стоимости этой B
taken = [p[ind]]  # список для купленных партий, сразу кладём найденную
cnt_b = 1  # счетчик купленных партий типа B, # фиксируем наличие одной B
p.pop(ind)  # удаляем эту партию из списка, чтобы основной цикл не купил её повторно

# жадный алгоритм - набираем
# макс. количество самых дешевых партий
while len(p) != 0:
    cost, p_type = p[0]
    # проверяем, хватает ли на текущую партию
    if cost + s > m:
        # если деньги кончились, прерываем цикл
        break
    if p_type == "B":
        cnt_b += 1  # считаем купленные партии B
    # удаляем первую (самую дешёвую) партию и добавляем её в список взятых
    taken.append(p.pop(0))  # добавляем партию
    s += cost  # увеличиваем потраченную сумму

# сортируем купленные партии по убыванию цены
# (самые дорогие окажутся в начале)
taken.sort(reverse=True)
for j in range(len(taken)):
    # ищем купленную партию B, которую можно заменить на A
```

```
    # при этом если осталась лишь 1 штука, не можем её заменить
    if taken[j][1] == "B" and cnt_b != 1:
        # ищем некупленную партию A в оставшемся хвосте изначального списка
        # предыдущий цикл закончился на индексе i,
        # значит, товар p[i] мы не смогли взять
        for k in range(len(p)):
            if p[k][1] == "A" and s - taken[j][0] + p[k][0] <= m:
                # обновляем потраченную сумму
                s = s - taken[j][0] + p[k][0]
                # меняем партии местами
                taken[j], p[k] = p[k], taken[j]
                # уменьшаем счетчик B, так как заменили
                cnt_b -= 1
                break

# финал - подсчёт количества купленных партий типа A
cnt_a = 0
for prod in taken:
    if prod[1] == "A":
        cnt_a += 1
print(cnt_a, m - s)
```

Ответ:

Задача 26.2 (Сибирь)

Фермерское хозяйство занимается закупкой винограда у местных поставщиков для производства соков. Используется виноград двух типов: A и B. В процессе закупки работают K сборщиков, которые принимают партии. Сборщики с нечётными номерами принимают только виноград типа A, сборщики с чётными номерами – только виноград типа B. Нумерация сборщиков начинается с 1.

Каждый поставщик может приехать на склад в любое время, но новый поставщик не может начать разгрузку раньше чем через 5 секунд после конца разгрузки предыдущего поставщика. Если в момент прибытия поставщика все подходящие ему сборщики заняты или ближайшее свободное время не позволяет вписаться в график, то партия винограда отправляется на рынок без участия сборщиков.

Известны общее количество поставщиков N и количество сборщиков K, а также для каждого поставщика: время начала доступности партии, время окончания возможности сдачи, и тип винограда (A или B).

Необходимо определить, сколько всего поставок было успешно принято сборщиками, и номер последнего сборщика, который принял партию.

Решение

Первым делом считываем поставки в список и сортируем по времени старта. Для сборщиков создаём массив collectors, заполненный -100, чтобы хранить время их освобождения и гарантированно принять самые первые партии.

Затем запускаем перебор поставок. Так как нумерация в списках начинается с нуля, нечётные сборщики (1, 3, 5...) лежат под чётными индексами (0, 2, 4...), а чётные сборщики - под нечётными. Поэтому для типа A начинаем поиск с 0-го индекса, для B - с 1-го, и перебираем сборщиков с шагом 2. Ищем первого, у которого с прошлой разгрузки прошло 5 и более секунд.

Если нашёлся, то записываем ему время конца текущей разгрузки, плюсуем счётчик успешных поставок, сохраняем реальный номер сборщика (индекс списка + 1) и завершаем подбор.

В конце выводим счётчик принятых партий и номер последнего задействованного сборщика.

```
f = open('26_1.txt')
# считываем количество поставщиков и сборщиков
n, k = map(int, f.readline().split())
shipments = []  # список для хранения всех поставок
for i in range(n):
    line = f.readline().split()
    # считываем начало, конец поставки и тип винограда
    shipments.append([int(line[0]), int(line[1]), line[2]])

# задаём отрицательное время, чтобы первые поставщики 100% прошли проверку
collectors = [-100 for _ in range(k)]
shipments.sort()  # сортируем поставки по времени начала
delivered, last_num = 0, 0
# перебираем все поставки
for p in shipments:
    start, end, p_type = p

    # определяем стартовый индекс: 0 для нечётных (A), 1 для чётных (B)
    # так как в задаче нумерация начинается с 1, а в питоне - с нуля,
    # у нас нечётным номерам будут соответствовать чётные индексы
    if p_type == 'A':
        start_index = 0
    else:
        start_index = 1

    # идём по нужным сборщикам с шагом 2,
    # чтобы рассматривать только чётные или только нечётные
    for i in range(start_index, len(collectors), 2):
        # проверяем, прошло ли минимум 5 секунд с прошлой разгрузки
        if start - collectors[i] >= 5:
            collectors[i] = end  # обновляем время освобождения
            delivered += 1  # плюсуем успешную поставку
```

```
            last_num = i + 1  # запоминаем номер сборщика
            break  # товар собран, идём к следующему поставщику

print(delivered, last_num) # выводим ответ
```

Ответ: 3374 11

Задача №27

Задача 27.1 (Дальний восток)

Учёный решил провести кластеризацию некоторого множества звёзд по их расположению на карте звёздного неба. Кластер звёзд – это набор звёзд (точек) на графике, лежащий внутри прямоугольника высотой H и шириной W. Каждая звезда обязательно принадлежит только одному из кластеров.

Истинный центр кластера, или центроид, – это одна из звёзд на графике, сумма расстояний от которой до всех остальных звёзд кластера минимальна.

Под расстоянием понимается расстояние Евклида между двумя точками $A(x_1, y_1)$ и $B(x_2, y_2)$ на плоскости, которое вычисляется по формуле:

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

В файле A хранятся данные о звёздах **двух** кластеров, где $H = 6.5$, $W = 4,5$ для каждого кластера. В каждой строке записана информация о расположении на карте одной звезды: сначала координата x, затем координата y. Значения даны в условных единицах. Известно, что количество звёзд не превышает 1000.

В файле B хранятся данные о звёздах **трех** кластеров, где $H = 5$, $W = 4$ для каждого кластера. Известно, что количество звёзд не превышает 10000. Структура хранения информации о звездах в файле B аналогична файлу A.

Известно, что в файле имеются координаты ровно трёх «лишних» точек, представляющих аномалии, которые возникли в результате помех при передаче данных. Эти точки не относятся ни к одному из кластеров, их учитывать не нужно.

Для файла A определите координаты центра каждого кластера, затем найдите два числа: P_x – минимальную из абсцисс центров кластеров, и P_y – минимальную из ординат центров кластеров.

Для файла Б определите координаты центра каждого кластера, затем найдите два числа: Q_1 – минимальное расстояние между центрами кластеров, и Q_2 – максимальное расстояние между центрами кластеров.

В ответе запишите четыре числа: сначала абсолютную величину целой части произведения произведения $P_x \times 10000$, затем абсолютную величину целой части произведения $P_y \times 10000$, затем абсолютную величину целой части произведения $Q_1 \times 10000$, затем абсолютную величину целой части произведения $Q_2 \times 10000$.

Решение

Файл A

```
#Импортируем функцию расстояния
from math import dist


# реализация алгоритма DBSCAN
```

```
def dbscan(a, r):
    cl = [] # инициализируем список для хранения кластеров
    while a: # пока есть элементы в входном массиве 'a'
        cl.append([a.pop(0)])
        for i in cl[-1]: # проходим по элементам последнего кластера
            for j in a[:]:
                if dist(i, j) <= r: # вычисляем расстояние между звездами
                    cl[-1].append(j) # добавляем 'j' в текущий кластер
                    a.remove(j) # удаляем 'j' из списка 'a'
    return cl #возвращаем список кластеров

# Открываем файл
file = open("27_A.txt")

# Список звёзд
stars = []

# кол-во кластеров
n = 2

# Проходим по строкам файла
for line in file:
    star = list(map(float, line.replace(',', '.').split()))
    # Добавляем звезду
    stars.append(star)

# Кластеры
clusters = dbscan(stars, 0.5)

# избавляемся от аномалий
cl_total = [cluster for cluster in clusters if len(cluster) > 10]

# Центры кластеров
centers = []

# Проходим по кластерам
for cl in cl_total:
    # Ищем минимальное суммарное расстояние
    min_dist = 10 ** 100

    # Перебираем кандидатов на центр
    for cent in cl:
        # Суммарное расстояние от кандидата до остальных звёзд
        sum_dist = 0
```

```
        # Перебираем остальные звезды кластера
        for star in cl:
            # Суммируем расстояния
            sum_dist += dist(cent, star)

        # Обновляем минимальное суммарное расстояние и центр
        if sum_dist < min_dist:
            min_dist = sum_dist
            center = cent

    # Добавляем центр в список
    centers.append(center)

# минимальная из абсцисс центров кластеров
px = min(centers[0][0], centers[1][0])

# минимальная из ординат центров кластеров
py = min(centers[0][1], centers[1][1])

# Выводим ответ
print(abs(int(px * 10_000)), abs(int(py * 10_000)))
```

Файл Б

```
python
#Импортируем функцию расстояния
from math import dist


# реализация алгоритма DBSCAN
def dbscan(a, r):
    cl = [] # инициализируем список для хранения кластеров
    while a: # пока есть элементы в входном массиве 'a'
        cl.append([a.pop(0)])
        for i in cl[-1]: # проходим по элементам последнего кластера
            for j in a[:]:
                if dist(i, j) <= r: # вычисляем расстояние между звездами
                    cl[-1].append(j) # добавляем 'j' в текущий кластер
                    a.remove(j) # удаляем 'j' из списка 'a'
    return cl #возвращаем список кластеров

# Открываем файл
file = open("27_B.txt")
```

```
# Список звёзд
stars = []

# кол-во кластеров
n = 3

# Проходим по строкам файла
for line in file:
    star = list(map(float, line.replace(',', '.').split()))
    # Добавляем звезду
    stars.append(star)

# Кластеры
clusters = dbscan(stars, 0.5)

# избавляемся от аномалий
cl_total = [cluster for cluster in clusters if len(cluster) > 10]

# Центры кластеров
centers = []

# Проходим по кластерам
for cl in cl_total:
    # Ищем минимальное суммарное расстояние
    min_dist = 10 ** 100

    # Перебираем кандидатов на центр
    for cent in cl:
        # Суммарное расстояние от кандидата до остальных звёзд
        sum_dist = 0

        # Перебираем остальные звезды кластера
        for star in cl:
            # Суммируем расстояния
            sum_dist += dist(cent, star)

        # Обновляем минимальное суммарное расстояние и центр
        if sum_dist < min_dist:
            min_dist = sum_dist
            center = cent

    # Добавляем центр в список
    centers.append(center)
```

```
# минимальное расстояние между центрами кластеров
qx = min(dist(centers[0], centers[1]), dist(centers[0],
centers[2]), dist(centers[1], centers[2]))

# максимальное расстояние между центрами кластеров
qy = max(dist(centers[0], centers[1]), dist(centers[0],
centers[2]), dist(centers[1], centers[2]))

# Выводим ответ
print(abs(int(qx * 10_000)), abs(int(qy * 10_000)))
```

Задача 27.2 (Сибирь)

Фрагмент звёздного неба спроецирован на плоскость с декартовой системой координат. Учёный решил провести кластеризацию полученных точек, являющихся изображениями звёзд, то есть разбить их множество на N непересекающихся непустых подмножеств, таких, что точки каждого подмножества лежат внутри прямоугольника размера $H \times W$, причём эти прямоугольники между собой не пересекаются. Стороны прямоугольников не обязательно параллельны координатным осям.

Гарантируется, что такое разбиение существует и единственно для заданных значений размеров прямоугольника и количества кластеров.

Для каждой звезды дана характеристика: тип цвета, светимость и её размер в соответствии с таблицей.

Обозначение	Цвет
O	Голубой
B	Бело-голубой
A	Белый
F	Жёлто-белый
G	Жёлтый
K	Оранжевый
M	Красный

Обозначение	Размер
I	карлик
II	субкарлик
III	гигант
IV	сверхгигант
V	мегагигант
VI	супергигант

Полученные значения записаны в характеристике слитно: обозначение цвета, затем светимость (обозначается одной арабской цифрой), а затем размер звезды.

Будем называть антицентром кластера точку этого кластера, сумма расстояний от которой до всех остальных точек кластера максимальна. Для каждого кластера гарантируется единственность его антицентра. Расстояние между двумя точками на плоскости $A(x_1, y_1)$ и $B(x_2, y_2)$ вычисляется по формуле:

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

В файле A хранятся данные о звёздах двух кластеров, где для каждого кластера $H = 4$, $W = 3$. В каждой строке записана информация о расположении на карте одной звезды: сначала координата x, затем координата y, а затем характеристика звезды. Значения даны

в условных единицах. Известно, что количество точек не превышает 1000.

В файле Б хранятся данные о звёздах трех кластеров, где для каждого кластера $H = 4, W = 4$. Известно, что количество точек не превышает 20000. Структура хранения информации о звёздах в файле Б аналогична файлу А.

Для файла А определите координаты центра каждого кластера, а затем найдите два числа: A_1 – минимальное расстояние от голубого субкарлика до центра этого же кластера; A_2 – сумма расстояний центров до точки (-1,2).

Для файла Б определите координаты центра каждого кластера, а затем найдите два числа: B_1 – абциссу центра кластера с минимальным количеством субкарликов; B_2 – ординату центра кластера с минимальным количеством субкарликов;

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

Решение

Файл А

```
# Импортируем функцию расстояния
from math import dist

# Открываем файл
file = open("27_1_A.txt")
# Список кластеров
clusters = [[] for _ in range(2)]

# Проходим по строкам файла
for line in file:
    # Заменяем запятые на точки и разбиваем по пробелам
    x, y, c = line.replace(",", ".").split()
    # Преобразуем координаты в числа
    x, y = float(x), float(y)

    # Делим на кластеры по прямой y = 10
    if y < 10:
        clusters[0].append([x, y, c])
    else:
        clusters[1].append([x, y, c])

# Центры кластеров
centers = []
# Проходим по кластерам
for cl in clusters:
    # Ищем минимальное суммарное расстояние
    min_dist = 10 ** 100
```

```
    # Перебираем кандидатов на центр
    for cent in cl:
        # Суммарное расстояние от кандидата до остальных звёзд
        sum_dist = 0

        # Перебираем остальные звезды кластера
        for star in cl:
            # Суммируем расстояния
            sum_dist += dist(cent[:2], star[:2])

        # Обновляем минимальное суммарное расстояние и центр
        if sum_dist < min_dist:
            min_dist = sum_dist
            center = cent

    # Добавляем центр в список
    centers.append(center)

min_d = 10 ** 10
# Перебираем индексы кластеров
for i in range(len(clusters)):
    # Перебираем все звёзды
    for star in clusters[i]:
        # Смотрим, что звезда - голубой субкарлик
        if star[2][0] == "S" and star[2][2:] == "VI":
            # Считаем расстояние до центра
            d = dist(centers[i][:2], star[:2])
            # Если оно оказалось меньше текущего минимума,
            # то перезаписываем
            if d < min_d:
                min_d = d

# считаем сумму расстояний центров до точки (-1, 2)
sum_dist = dist(centers[0][:2], [-1, 2]) + dist(centers[1][:2], [-1, 2])
# Выводим ответ
print(int(min_d * 10_000), int(sum_dist * 10_000))
```

Файл Б

```
# Импортируем функцию расстояния
from math import dist

# Открываем файл
file = open("27_1_B.txt")
```

```
# Список кластеров
clusters = [[] for _ in range(3)]

# Проходим по строкам файла
for line in file:
    # Заменяем запятые на точки и разбиваем по пробелам
    x, y, c = line.replace(",", ".").split()
    # Преобразуем координаты в числа
    x, y = float(x), float(y)

    # Делим на кластеры по прямым y = 22 и y = 15
    if y > 22:
        clusters[0].append([x, y, c])
    elif y > 15:
        clusters[1].append([x, y, c])
    else:
        clusters[2].append([x, y, c])

# Центры кластеров
centers = []
# Проходим по кластерам
for cl in clusters:
    # Ищем минимальное суммарное расстояние
    min_dist = 10 ** 10

    # Перебираем кандидатов на центр
    for cent in cl:
        # Суммарное расстояние от кандидата до остальных звёзд
        sum_dist = 0

        # Перебираем остальные звезды кластера
        for star in cl:
            # Суммируем расстояния
            sum_dist += dist(cent[:2], star[:2])

        # Обновляем минимальное суммарное расстояние и центр
        if sum_dist < min_dist:
            min_dist = sum_dist
            center = cent

    # Добавляем центр в список
    centers.append(center)

# Список для хранения количества субкарликов в каждом кластере
```

```
counts = []
# Перебираем кластеры и подсчитываем субкарликов
for cl in clusters:
    count = 0  # Счётчик субкарликов
    for star in cl:
        if star[2][2:] == 'VI':
            count += 1
    counts.append(count)

# Кластер с минимальным количеством субкарликов
min_cluster_index = counts.index(min(counts))
# Центр этого кластера
center = centers[min_cluster_index]

# Выводим ответ
print(int(center[0] * 10_000), int(center[1] * 10_000))
```