Информатика. Задача 17. Методичка

В задании №17 требуется написать программу, которое обработывает последовательность/набор чисел.

Рассмотрим пример данной задачи: (ДЕМО ЕГЭ-2022)

В файле содержится последовательность целых чисел. Элементы последовательности могут принимать целые значения от 10000 до 10000 включительно. Определите и запишите в ответе сначала количество пар элементов последовательности, в которых хотя бы одно число делится на 3, затем максимальную из сумм элементов таких пар. В данной задаче под парой подразумевается два идущих подряд элемента последовательности. Например, для последовательности из пяти элементов: 6; 2; 9; –3; 6 - ответ: 4 11.

Обратим внимание на особенности данного задания:
- в различных прототипах количество чисел небольшое, в отличие от задачи 27Б, поэтому вполне возможно придумать неэффективное решение и быстро получить ответ
- в основном задача касается обработки пар (рядомстоящих и любых) чисел в последовательности
- в начале файла могут не писать число N - точное количество чисел в файле

Как считывать?

Первым делом возникает вопрос: как правильно считать значения из файла, если N не задано в начале файла? Варианта ответа тут два:
1) Посмотреть сколько строк в файле и вписать это значение в программу руками
2) Считывать до конца файла

Второй вариант менее ошибочный, поэтому посмотрим на него:

```
file = open('17.txt') # открываем файл на чтение
for num in file: # Цикл по строкам в файле: num - это СТРОКА!
    print(int(num)) # Печатаем числовое значение строки,
    # т.е. переводим строку в число
```

Дальше зависит от задачи, эти значения можно добавить в цикл для удобства:

```
file = open('17.txt')
a = [] # заводим массив
for num in file:
    a.append(int(num)) # добавляем в массив значение
```

Как обрабатывать рядомстоящие пары?

Рядомстоящей парой является набор двух элементов, разница в индексах которых равняется 1. Отсюда логично предположить, что i—ый рядомстоящий с $i-1$-ым и $i+1$-ым элементами. Пример в цикле:

```
for i in range(len(a) - 1):
    print(a[i], a[i + 1])
```

ВАЖНО: если в цикле есть обработка i + a элемента, где a - числовое значение, то в range обязательно нужно вычесть a, чтобы программа не получила ошибку выхода за границу цикла

Как обрабатывать любые пары?

Вспоминаем идею вложенных циклов и задействуем различные пары:
Важно не учесть элементы сами с собой и не дублировать пары друг с другом. Именно поэтому во втором цикле используется $i+1$

```
for i in range(len(a)):
    for j in range(i + 1, len(a)):
        print(a[i], a[j])
```

Основные характеристики

В задании №17 обязательно нужно проверять сумму/произведение или каждый элемент пары на какую-то конкретную характеристику

```
(a + b) % 10 == 5 # сумма оканчивается на 5
(a * b) % 3 == 0 # произведение кратно 3
(a % 3 == 0 or b % 3 == 0) # ХОТЯ БЫ один элемент кратно 3
(a % 5 == 0 and b % 5 == 0) # ОБА числа кратны 5
```

```
(a + b) / 2 # Среднее арифметическое, может получится вещественное число
(a * b) ** (1/2) # Среднее геометрическое,
# может получится вещественное число
```

Считаем количество пар, обладающим условиями, и ищем максимальную/минимальную характеристику среди этой пары/этой характеристики

Найдем количество чётных сумм и максимальную такую сумму:

```
count = 0
maxim = a[0] # можно проинициализировать начальным числом или
# ОЧЕНЬ МАЛЕНЬКИМ. Смотрите на ограничения в задаче, они могут помочь
# в инициализации!
for i in range(len(a)):
    s = (a[i] + a[i + 1])
    if s % 2 == 0: # сумма чётная?
        count += 1 # считаем количество
        # и среди таких сумм ищем максимум
        if s > maxim:
            maxim = s
    # если просят искать не среди ТАКИХ пар,
    # то проверку осуществляем не в условии
```

Найдем количество нечётных сумм и минимальное число среди таких пар, которые дают нечётную сумму:

```
count = 0
minim = a[0] # можно проинициализировать начальным числом или ОЧЕНЬ БОЛЬШИМ.
# Смотрите на ограничения в задаче, они могут помочь
# в инициализации!
for i in range(len(a)):
    s = (a[i] + a[i + 1])
    if s % 2 != 0: # сумма нечётная?
        count += 1 # считаем количество
        # и среди таких сумм ищем максимум
        if a[i] < minim:
```

```
            minim = a[i]
        if a[i + 1] < minim:
            minim = a[i + 1]
    # если просят искать не среди ТАКИХ пар,
    # то проверку осуществляем не в условии
```