

## ЕГЭ. Информатика. Задание 17

Яцулевич Владимир Владимирович

Данная задача представляет из себя разработку переборного алгоритма. Задание часто формулируется следующим образом.

Дан файл, содержащий целые числа. Необходимо в этом файле найти количество пар, которые бы удовлетворяли определённым условиям. Где под парой может подразумеваться либо два рядом стоящих элемента, либо различные два элемента.

### Считывание информации из файла

Для начала, необходимо считать данные из файла и подготовить список для перебора чисел. Для этого разместим файл в корень проекта и напишем следующую команду:

```
file = open('file_name', 'r')
```

Здесь функция `open()` принимает два аргумента: название файла и способ взаимодействия с файлом. Параметр `'r'` означает, что файл открывается только для чтения, параметр `'w'` означает, что файл открывается только для записи. Теперь в переменной `file` записана вся информация о файле. Чтобы извлечь непосредственно содержимое нужно к объекту `file` применить метод `read()`.

```
text = file.read()
```

На данном этапе все числа записаны в одну строку `text`. Поэтому их нужно разделить. Числа отделяет друг от друга символ переноса каретки на новую строку `'\n'`. Поэтому сформируем список чисел пока ещё в строковом формате. Для этого применим метод `split()` к строке `text`.

```
lst = text.split('\n')
```

Также стоит отметить, что часто в файле имеется последняя пустая строка, которую нам учитывать не нужно. Поэтому добавим в нашу команду удаление последнего элемента. Важно подчеркнуть, что в разных файлах может быть разное форматирование, поэтому где-то не нужно удалять последний символ.

```
lst = text.split('\n')[:-1]
```

Последний шаг — это преобразование всех строковых элементов списка к числу. Для этого воспользуемся функцией `map()`.

```
lst = list(map(int, lst))
```

По итогу, считывание информации из файла можно описать следующим набором команд.

```
file = open('file_name', 'r')
text = file.read()
lst = text.split('\n')[:-1]
lst = list(map(int, lst))
```

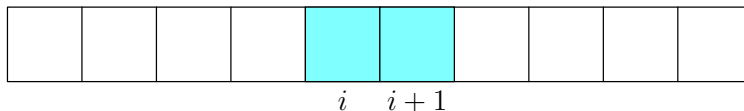
### Алгоритм

При решении задания очень важно обратить внимание на определение пары. Обычно можно увидеть следующие два определения.

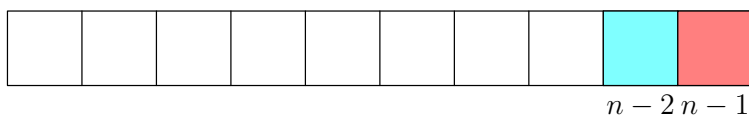
1. Пара — два соседних элемента.
2. Пара — два различных элемента.

#### Два соседних элемента

В зависимости от определения пары, будет меняться и перебор. Рассмотрим алгоритм перебора в случае первого определения. В данном случае будет достаточно одного цикла `for`, поскольку при переборе первого элемента пары, положение второго будет чётко определено. Например, если первый элемент будет находиться на  $i$ -ой позиции, то его сосед будет располагаться на  $i + 1$ -ой позиции.



Но в этом случае есть небольшая проблема. Если мы будем находиться на последнем элементе, то при обращении к следующему элементу будет выдаваться ошибка. Ошибка будет появляться из-за того, что мы будем пытаться получить доступ к элементу, которого нет.



В результате будем перебирать все элементы, за исключением последнего. Пусть числа записаны в списке `lst`.

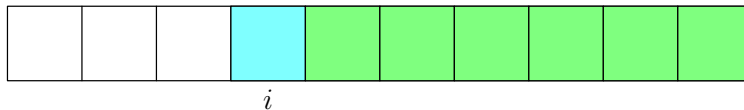
```

n = len(lst)
for i in range(n - 1):
    first = lst[i]
    second = lst[i + 1]
    ...

```

### Два различных элемента

В случае двух различных элементов необходимо будет осуществлять перебор «каждый с каждым». Но здесь нужно быть аккуратным. Поскольку если сделать буквально перебор всех пар, то пара  $(i, j)$  будет учтена дважды, поскольку пара  $(j, i)$  в переборе будет считаться другой. Чтобы избежать этой проблемы будем перебирать только такие пары  $(i, j)$ , для которых  $i < j$ .



Перебор первого индекса  $i$  будет осуществляться по всем значениям. А вот перебор второго индекса  $j$  будет осуществляться только по тем ячейкам, которые будут правее ячейки с индексом  $i$ .

```

n = len(lst)
for i in range(n):
    for j in range(i + 1, n):
        first = lst[i]
        second = lst[j]
        ...

```

### Пример решения задач

#### Задача 1

Файл содержит последовательность натуральных чисел, не превышающих 100 000. Назовём парой два идущих подряд элемента последовательности. Определите количество пар, для которых выполняются следующие условия:

- остаток от деления на 3 хотя бы одного числа из пары равен остатку от деления на 3 максимального элемента всей последовательности;
- остаток от деления на 7 хотя бы одного числа из пары равен остатку от деления на 7 минимального элемента всей последовательности.

В ответе запишите два числа: сначала количество найденных пар, затем макси-

мальную величину суммы элементов этих пар.

**Решение.** Для начала считаем данные из файла.

```
file = open('task_1', 'r')
text = file.read()
lst = text.split('\n')[-1]
lst = list(map(int, lst))
```

По условию задачи нам потребуются максимальный и минимальный элемент всей последовательности. Найдём их с помощью функций `max()` и `min()` соответственно. Также по условию задания нужно найти остаток от деления максимального элемента на 3 и остаток от деления минимального элемента на 7.

```
mx = max(lst)
mn = min(lst)
mx3 = mx % 3 # Остаток от деления на 3 максимального элемента всей
↳ последовательности
mn7 = mn % 7 # Остаток от деления на 7 минимального элемента всей
↳ последовательности
```

Теперь начнём перебор. В этой задаче пара — это два соседних элемента. Тогда перебор будет осуществлять одним циклом. Поскольку нам необходимо найти количество пар, то будем накапливать это число в переменной `acc`. Изначально накопитель должен быть заполнен нулём.

```
n = len(lst)
acc = 0
for i in range(n - 1):
    first = lst[i]
    second = lst[i + 1]
```

Для упрощения кода и для более аккуратного стиля условия внутри цикла сохраним в отдельные переменные.

```
condition1 = (first % 3 == mx3) or (second % 3 == mx3)
condition2 = (first % 7 == mn7) or (second % 7 == mn7)
condition = condition1 and condition2
```

Тогда подсчёт количества нужных пар будет выглядеть следующим образом.

```

for i in range(n - 1):
    first = lst[i]
    second = lst[i + 1]
    condition1 = (first % 3 == mx3) or (second % 3 == mx3)
    condition2 = (first % 7 == mn7) or (second % 7 == mn7)
    condition = condition1 and condition2
    if condition:
        acc += 1

```

Первая часть задания выполнена. Теперь нужно решать вторую часть задания. Во второй части необходимо определить максимальную величину суммы пар чисел. По сути это задача поиска максимального элемента. Инициализируем переменную `mx_sum`, которая изначально будет заполнена каким-нибудь не натуральным числом, например `-1`. Затем внутри цикла перебора будем обновлять её значение, если найдём большую сумму.

```

mx_sum = -1
for i in range(n - 1):
    first = lst[i]
    second = lst[i + 1]
    condition1 = (first % 3 == mx3) or (second % 3 == mx3)
    condition2 = (first % 7 == mn7) or (second % 7 == mn7)
    condition = condition1 and condition2
    if condition:
        acc += 1
        sm = first + second
        if sm > mx_sum:
            mx_sum = sm

```

Остаётся только вывести полученный ответ. Общее решение примет вид.

```

file = open('task_1', 'r')
text = file.read()
lst = text.split('\n')[:-1]
lst = list(map(int, lst))

mx = max(lst)
mn = min(lst)

```

```
mx3 = mx % 3 # Остаток от деления на 3 максимального элемента всей
↳ последовательности
mn7 = mn % 7 # Остаток от деления на 7 минимального элемента всей
↳ последовательности

n = len(lst)
acc = 0
mx_sum = -1

for i in range(n - 1):
    first = lst[i]
    second = lst[i + 1]
    condition1 = (first % 3 == mx3) or (second % 3 == mx3)
    condition2 = (first % 7 == mn7) or (second % 7 == mn7)
    condition = condition1 and condition2
    if condition:
        acc += 1
        sm = first + second
        if sm > mx_sum:
            mx_sum = sm

print(acc, mx_sum)
```