ЕГЭ-информатика
Задание №26
«Обработка целочисленной информации»

Что нужно знать при написании программы:

Чтение данных из файла

- в языке Python для чтения данных удобно использовать менеджер контекста (**with ... as**), который открывает файл и закрывает его; например, код

```
with open("26.txt") as Fin:  # программа и файл в одной папке
   ... # какие-то операции с файлом
   # при завершении работы менеджера контекста
   # файл автоматически закрывается
```

- если в текущей строке файла находится целое число, то прочитать его в переменную **x** можно так:

```
x = int( Fin.readline() )
```

- если в строке записаны два числа, после чтения (**Fin.readline()**) строку нужно разбить на отдельные части по пробелам между числами (каждая часть – символьная запись числа) и затем каждую часть преобразовать в целое число; например, чтение двух чисел:

```
s = Fin.readline()
symData = s.split()
x, y = map( int, symData )
```

Что нужно знать:

Хранение массива данных

- в языке Python для хранения массива данных используется список; следующая программа показывает чтение массива данных размера N в список **data** из файла «26.txt» (данные записаны в столбик, по одному числу в строке):

```
data = [0]*N
with open("26.txt") as Fin:
  for i in range(N):
    data[i] = int( Fin.readline() )
```

или с помощью генератора списка

```
with open("26.txt") as Fin:
  data = [ int( Fin.readline() )
             for i in range(N) ]
```

Что нужно знать:

Сортировка массива

Для сортировки имеет смысл использовать встроенные функции языков программирования.

Категорически НЕ рекомендуется писать собственные реализации алгоритмов сортировки.

- В языке Python для сортировки массива (списка) «на месте» вызывается метод **sort**:

```
data.sort()
```

- при этом числа сортируются по возрастанию. Для сортировки по убыванию в вызов метода добавляем именованный аргумент **reverse** со значением **True**:

```
data.sort( reverse = True )
```

- Для сортировки по другому критерию (например, по последней цифре числа) добавляют именованный аргумент **key**, который указывает на функцию, вычисляющую нужно значение, например:

```
def lastDigit( n ):
  return n % 10
... # заполнение массива data
data.sort( key = lastDigit )
```

Системный администратор раз в неделю создаёт архив пользовательских файлов. Однако объём диска, куда он помещает архив, может быть меньше, чем суммарный объём архивируемых файлов. Известно, какой объём занимает файл каждого пользователя.

По заданной информации об объёме файлов пользователей и свободном объёме на архивном диске определите максимальное число пользователей, чьи файлы можно сохранить в архиве, а также максимальный размер имеющегося файла, который может быть сохранён в архиве, при условии, что сохранены файлы максимально возможного числа пользователей.

Входные данные.

Задание 26

В первой строке входного файла находятся два числа: *S* — размер свободного места на диске (натуральное число, не превышающее 10 000) и *N* — количество пользователей (натуральное число, не превышающее 1000). В следующих *N* строках находятся значения объёмов файлов каждого пользователя (все числа натуральные, не превышающие 100), каждое в отдельной строке.

Запишите в ответе два числа: сначала наибольшее число пользователей, чьи файлы могут быть помещены в архив, затем максимальный размер имеющегося файла, который может быть сохранён в архиве, при условии, что сохранены файлы максимально возможного числа пользователей.

Пример входного файла:

100 4

80

30

50

40

При таких исходных данных можно сохранить файлы максимум двух пользователей. Возможные объёмы этих двух файлов 30 и 40, 30 и 50 или 40 и 50. Наибольший объём файла из перечисленных пар — 50, поэтому ответ для приведённого примера:

2 50

```
f = open('26_demo.txt')
data = f.readlines()
s = data[0].split()
s = int(s[0])
del (data[0])  # первая строка больше не нужна, удаляем
ее
for i in range(0, len(data)):
    data[i] = int(data[i])
data = sorted(data)
summa = 0
for count in range(0, len(data)):
    if summa + data[count] > s: break
    summa += data[count]
print(count)
zapas = s - summa
for i in range(0, len(data)):
    if data[i] - data[count - 1] <= zapas:
        itog = data[i]
print(itog)
```

Ответ: 568 50