МКУ «Научно-методический центр г. Курска»
МБОУ «Гимназия № 4»

«Методические основы решения заданий 24, 25, 26, 27 компьютерного ЕГЭ по информатике и ИКТ»

Филиппов Роман Сергеевич,
заместитель директора по учебно-воспитательной работе
МБОУ «Гимназия № 4» города Курска

Перечень алгоритмов, проверяемый в заданиях 24-27 на КЕГЭ по информатике и ИКТ

- ✓ **Алгоритмы анализа и преобразования записей чисел в позиционной системе счисления.**

- ✓ **Алгоритмы, связанные с делимостью целых чисел. Алгоритм Евклида для определения НОД двух натуральных чисел.**

- ✓ **Алгоритмы линейной (однопроходной) обработки последовательности чисел без использования дополнительной памяти, зависящей от длины последовательности (вычисление максимума, суммы, линейный поиск и т.п.). Обработка элементов последовательности, удовлетворяющих определённому условию (вычисление суммы заданных элементов, их максимума и т.п.).**

- ✓ **Алгоритмы обработки массивов. Примеры: перестановка элементов данного одномерного массива в обратном порядке; циклический сдвиг элементов массива; заполнение двумерного числового массива по заданным правилам; поиск элемента в двумерном массиве; вычисление максимума и суммы элементов двумерного массива. Вставка и удаление элементов в массиве.**

- ✓ **Алгоритмы анализа символьных строк, в том числе: подсчёт количества появлений символа в строке; разбиение строки на слова по пробельным символам; поиск подстроки внутри данной строки; замена найденной подстроки на другую строку.**

Работа с текстовыми файлами в Python. Ввод-вывод данных

Текст может храниться в двух форматах: (.txt) — простой текст и (.rtf) — «формат обогащенного текста»

Функция Open () - открывает файл для чтения или записи

f = open (file_name, access_mode),

где ***file_name = имя открываемого файла***

access_mode = режим открытия файла.

Метод Readline () - читает **одну целую строку** из файла (конечный символ новой строки ’\n’ сохраняется в строке.

S = file.readline([size])

где ***file - имя файловой переменной***

size - количество байт.

Метод Readlines () – читает файл построчно, пока не достигнет конца файла EOF и возвращает список, содержащий строки (конечный символ новой строки ’\n’ сохраняется в строке.

A = file.readlines([sizehint])

где ***file - имя файловой переменной***

size - количество байт.

Работа с текстовыми файлами в Python. Ввод-вывод данных

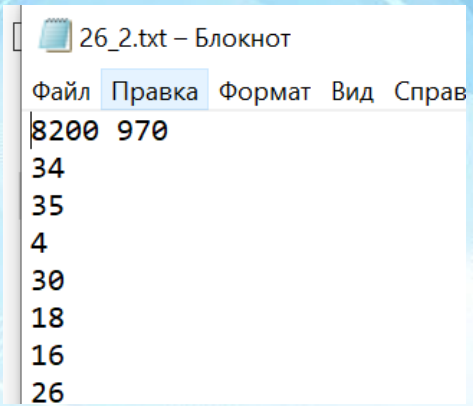
Конструкции для чтения данных из файла и «прохода» по файлу

```
f = open('24.txt')
for x in f:
    n = int(x)
```

```
f = open('26.txt');
k = f.readline(); n = f.readline()
for x in f:
    a.append(list(map(int,x.split())))
```

```
for x in open('17.txt'):
    n = int(x)
```

```
count=0
for x in open('9.txt'):
    a = list(map(int,x.split()))
```




```
f=open('26_2.txt')
s,n = list(map(int,f.readline().split()))
a=[]
#Записываем данные в список a
for i in range(n):
    a.append(int(f.readline()))
```

Задание № 24

Уровень сложности: высокий

Требуется использование специализированного программного обеспечения: да

Максимальный балл: 1 первичный балл

Примерное время выполнения задания: 18 минут

Проверяемые элементы содержания: умение создавать собственные программы (10–20 строк) для обработки символьной информации

Элементы содержания, проверяемые заданием: цепочки (конечные последовательности), деревья, списки, графы, матрицы (массивы)

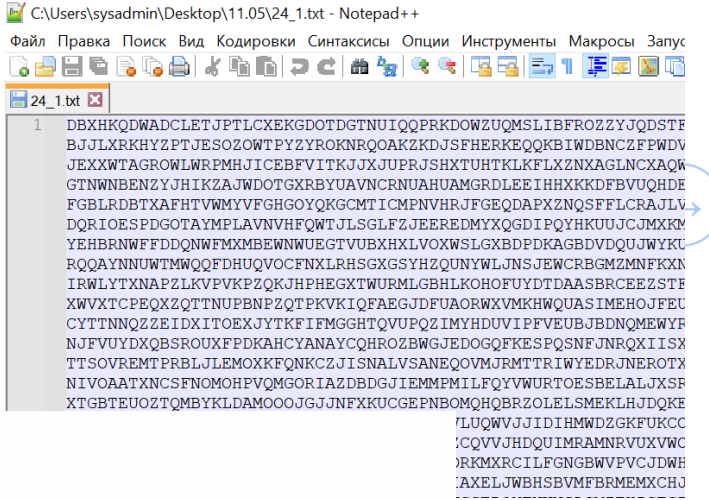
Задание 24 (№7624).

Текстовый файл состоит из прописных символов латинских букв.

Определите максимальное количество подряд идущих символов в прилагаемом файле, среди которых не содержится два символа из набора букв *X*, *Y* и *Z* (с учётом повторений), стоящих рядом.

Для выполнения этого задания следует написать программу.

1. XKYXKJXZKL
2. XKXXKJXXKL
3. XKX XKJX XKL


C:\Users\sysadmin\Desktop\11.05\24_1.txt - Notepad++
Файл Правка Поиск Вид Кодировки Синтаксисы Опции Инструменты Макросы Запус
24_1.txt
1
DBXHKQDWADCLETJPTLCXEKGDOTDGTNUIQQPRKDOWZUQMSLIBFROZZYJQDSTF
BJJLXRKHYZPTJESOZOWTPYZYROKNRQOAKZKDJSFHERKEQQKBIWDBNCZFPWDV
JEXXWTAGROWLWRPMHJICEBFVITKJJXJUPRJSHXTUHTKLKFLXZNXAGLNCXAQW
GTNWNBENZYJHIKZAJWDOTGXRBYUAVNCRNUAHUAMGRDLEEIHHXKKDFBVUQHDE
FGBLRDBTXAFHTVWMYVFGHGOYQKGCMTICMPNVHRJFGEQDAPXZNQSFFLCRAJLV
DQRIOESPDGOTAYMPLAVNVHFQWTJLSGLFZJEERERDMYXQGDIPQYHKUUJCJMXKM
YEHBRNWFFDDQNWFMXMBEWNWUEGTVUBXHXLVOXWSLGXBDPDKAGBDVDQUJWYKU
RQQAYNNUWTMWQQFDHUQVOCFNXLRHSGXGSYHZQUNYWLJNSJEWCRBGMZMNFKXN
IRWLYTXNAPZLKVPVKPZQKJHPHEGXTWURMLGBHLKOHOFUYDTDAASBRCEEZSTF
XWVXTCPEQXZQTTNUPBNPZQTPKVKIQFAEGJDFUAORWXVMKHWQUASIMEHOJFEU
CYTTNNQZZEIDXITOEXJYTKFIFMGGHTQVUPQZIMYHDUVIPFVEUBJBDNQMEWYF
NJFVUYDXQBSROUXFPDKAHCYANAYCQHROZBWGJEDOGQFKESPQSNFJNRQXIISX
TTSOVREMTPRBLJLEMOXKFQNKCZJISNALVSANEQOVMJRMTTRIWYEDRJNEROTX
NIVOAATXNCSFNOMOHPVQMGORIAZDBDGJIEMMPMILFQYVWURTOESBELALJXSR
XTGBTEUOZTQMBYKLDAMOOOJGJJNFXKUCGEPNBOMQHQBRZOLELSMEKLHJDQKE
```
1  f = open('24_1.txt') # открываем файл
2  s = f.readline() # считываем строку
3  s = s.replace('Y','X').replace('Z','X')   # делаем замену
4  s = s.replace('XX', 'X X') # разбиваем пробелом
5  a = s.split()  # создаем список (массив)
6  print(max(map(len,a))) # получем длину каждого элемента списка и выводим максимум
7
```

Сохранить ответ

Поиск в файлах | Поиск | Стек Данных | I/O Отладки | Исключения | Сообщения | Команды ОС

Отладки ввода/вывода (стандартный ввод, стандартный вывод, стандартный вывод) появляется ниже

786

Задание 24 .

(А. Рогов) Текстовый файл состоит из символов, обозначающих прописные буквы латинского алфавита.

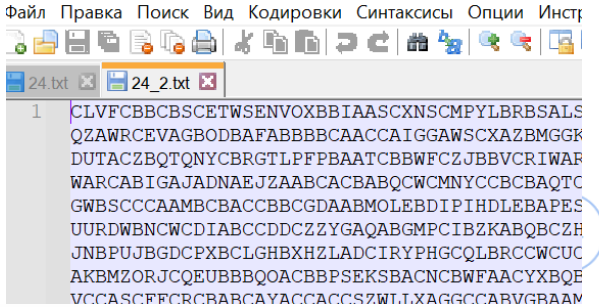
Определите максимальное количество идущих подряд пар символов, каждая из которых содержит только символы из набора букв A, B и C (с учётом повторений).

Для выполнения этого задания следует написать программу.

Пример входных данных

QWAABAD

Для приведенного примера максимальное количество идущих подряд пар это 2: AA и BA.




```
1  s = open('24_2.txt').readline()  # открываем файл и считываем строку
2  l = 1; maxd = 1   #   текущая длина цепочки   и  максимальная длина цепочки
3  for i in range(len(s)-1):   # проходим последовательно по элементам строки
4      if (s[i] in 'ABC') and (s[i+1] in 'ABC'):   # проверяем два соседних символа
5          l+=1                                    # если равны, увеличиваем длину цепочки на 1
6      else:
7          maxd = max(maxd,l)                      # если нет, проверяем длину текущей цепочки и максимальной
8          l = 1                                   # начинаем считать следующую цепочку
9  print(maxd//2)                                  # выводим количество пар
```

Поиск в файлах | Поиск | Стек Данных | I/O Отладки | Исключения | Сообщения | Команды ОС

Отладки ввода/вывода (стандартный ввод, стандартный вывод, стандартный вывод) появляется ниже

```
9
```

Текстовый файл `24_3.txt` состоит не более чем из 10^6 заглавных латинских букв (A..Z). Текст разбит на строки различной длины. Необходимо найти строку, содержащую наименьшее количество букв A (если таких строк несколько, надо взять ту, которая в файле встретилась раньше). Определите, какая буква встречается в этой строке чаще всего. Если таких букв несколько, надо взять ту, которая стоит последней в алфавите. Запишите в ответе эту букву, а затем – сколько раз она встречается во всем файле.

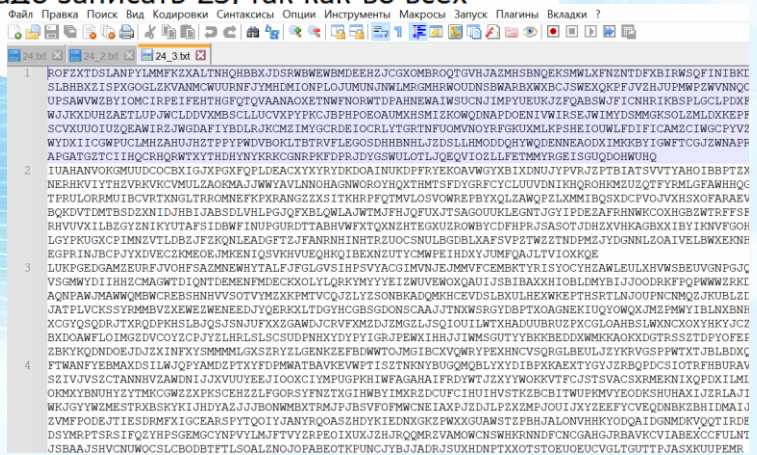
Пример. Исходный файл:

ZZQAQ

ZAVLAB

KRAKTU

В этом примере в первой и третьей строках по одной букве A, во второй – две. Берём первую строку, т.к. она находится в файле раньше. В этой строке чаще других встречаются буквы Z и Q (по два раза), выбираем букву Z, т. к. она позже стоит в алфавите. В ответе для этого примера надо записать Z3. так как во всех строках файла буква Z встречается 3 раза.



Текстовый файл 24_3.txt состоит не более чем из 10^6 заглавных латинских букв (A..Z). Текст разбит на строки различной длины. Необходимо найти строку, содержащую наименьшее количество букв A (если таких строк несколько, надо взять ту, которая в файле встретилась раньше). Определите, какая буква встречается в этой строке чаще всего. Если таких букв несколько, надо взять ту, которая стоит последней в алфавите. Запишите в ответе эту букву, а затем – сколько раз она встречается во всем файле.

Пример. Исходный файл:

ZZQAQ
ZAVLAB
KRAKTU

В этом примере в первой и третьей строках по одной букве A, во второй – две. Берём первую строку, т.к. она находится в файле раньше. В этой строке чаще других встречаются буквы Z и Q (по два раза), выбираем букву Z, т. к. она позже стоит в алфавите. В ответе для этого примера надо записать Z3, так как во всех строках файла буква Z

```
1   s = open('24_3.txt').readlines()  # считываем строки из файла
2   n = [c.count('A') for c in s]   # генерируем список состоящий из числа букв A в каждой строке
3   s1 = s[n.index(min(n))]   # определяем строку с наименьшим количеством букв A, стоящую раньше других в общем списке
4   a=[0]*100      # создаем список для подстета количества букв (индекс элемента массива - код символа в таблице ASCI)
5   for i in s1:   # проходим по всем символам найденной строки
6       a[ord(i)] += 1    # увеличиваем счетчик (значение элемента масссива с индексом - код символа в ASCI)
7   k = max(a)           # определяем максимальное число - количество букв
8   for i in range(len(a)):   # ищем последнюю букву, которая встречается k раз
9       if a[i] == k:
10          ind = i           # фиксируем индекс буквы в массиве - код символа в таблице ASCI
11  # считаем сколько раз буква встречается в исходном файле
12  summa = 0
13  for c in s:
14      summa += c.count(chr(ind))
15  print(chr(ind),summa)
```

Поиск в файлах | Поиск | Стек Данных | I/O Отладки | Исключения | Сообщения | Команды ОС

Отладки ввода/вывода (стандартный ввод, стандартный вывод, стандартный вывод) появляется ниже

```
V 38429
```

Задание № 25

Уровень сложности: высокий

Требуется использование специализированного программного обеспечения: да

Максимальный балл: 1 первичный балл

Примерное время выполнения задания: 20 минут

Проверяемые элементы содержания: умение создавать собственные программы (10–20 строк) для обработки целочисленной информации

Элементы содержания, проверяемые заданием: построение алгоритмов и практические вычисления (метод динамического программирования, анализ алгоритмов)

1) Назовём маской числа последовательность цифр, в которой также могут встречаться следующие символы:

- символ «?» означает ровно одну произвольную цифру;
- символ «*» означает любую последовательность цифр произвольной длины; в том числе «*» может задавать и пустую последовательность.

Среди натуральных чисел, не превышающих 10^8, найдите все числа, соответствующие маске 12*4?65, делящиеся на 161 без остатка. В ответе запишите в первом столбце таблицы все найденные числа в порядке возрастания, а во втором столбце – соответствующие им результаты деления этих чисел на 161.

```
# 100000000
# 12**4?65

for i in range(124065,12994966):    # перебираем все числа в рамках действия маски
    s = str(i)                      # преобразуем число в строку
    if s[:2] == '12' and s[-4] == '4' and s[-2:] == '65':    # проверка соответствия маске с
        if i % 161 == 0:    # проверка картности 161
            print(i, i//161)   # вывод результата
```

Отладки ввода/вывода (с

```
1234065 7665
12004965 74565
12214265 75865
12294765 76365
12504065 77665
12584565 78165
12874365 79965
12954865 80465
```

12) Назовём маской числа последовательность цифр, в которой также могут встречаться следующие символы:

— символ «?» означает ровно одну произвольную цифру;

— символ «*» означает любую последовательность цифр произвольной длины; в том числе «*» может задавать и пустую последовательность.

Например, маске 123*4?5 соответствуют числа 123405 и 12300425.

Найдите все натуральные числа, не превышающие 10^{10}, которые соответствуют маске 1?1?1?1*1 и при этом без остатка делятся на 2023, а сумма цифр каждого числа равна 22. В ответе запишите все найденные числа в порядке возрастания, справа от каждого запишите частное от его деления на 2023.

```
data = [''] + [str(c)  for c in range(100)] + ['0'+str(c) for c in range(10)] # формируем комбинации для позиции *
data2 = '0123456789' # формируем комбинации для позиции ?
a = []  # список ответов
for x in data:   # перебираем позицию *
    for y1 in data2:       # перебираем позиции ?
        for y2 in data2:
            for y3 in data2:
                s = '1'+y1+'1'+y2+'1'+y3+'1'+x+'1'  # формируем строку-число
                if (int(s)%2023 == 0) and (sum(map(int,s)) == 22):   # проверка кратности и суммы циф
                    a.append(int(s)) # добавляем число в список
a.sort()  # сортируем значения по возрастанию
for x in a:    # вывод результата
    print(x,x//2023)
```

Отладки ввода/вывода (

```
19131511 9457
1012141291 500317
1319111311 652057
1516111051 749437
```

7) Среди натуральных чисел, принадлежащих промежутку [100 000 000; 1 000 000 000], найдите все числа, имеющие ровно 15 делителей, делящихся на 7 и отличных от самого числа. В ответ запишите 5 наименьших и 5 наибольших найденных чисел в порядке возрастания, справа от них укажите их максимальный делитель, отличный от самого числа.

(нижний)

```
def deliteli( n ):  # функция подсчета делителей согласно условию
    divs = set()    # множество делителей
    for i in range(2,round(n**0.5)+1):   # цикл поиска делителей числа n
        if n % i == 0:
            divs.add( i )
            divs.add( n // i )
    if sum([1 for c in divs if c%7==0])==15:    # проверка условия (15 делителей кратных 7)
        return max(divs)  # возвращаем максимальный делитель
    else: return 0    # возвращаем 0
k, ch = 0, 100000000
while k < 5:                # цикл поиска первых пяти чисел
    if  deliteli(ch) != 0:
        print(ch, deliteli(ch))
        k+=1
    ch+=1
k, ch = 0, 1000000000
while k < 5:                # цикл поиска последних пяти чисел
    if  deliteli(ch) != 0:
        print(ch, deliteli(ch))
        k+=1
    ch-=1
```

Отладки ввода/вывода (стан

```
100000054 50000027
100000131 33333377
100000222 50000111
100000250 50000125
100000264 50000132
999999896 499999948
999999875 199999975
999999854 499999927
999999798 499999899
999999791 142857113
```

Задание № 26

Уровень сложности: высокий

Требуется использование специализированного программного обеспечения: да

Максимальный балл: 2 первичных балла

Примерное время выполнения задания: 35 минут

Проверяемые элементы содержания: умение обрабатывать целочисленную информацию с использованием сортировки

Элементы содержания, проверяемые заданием: различные виды сортировки данных, формализация понятия алгоритма

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.

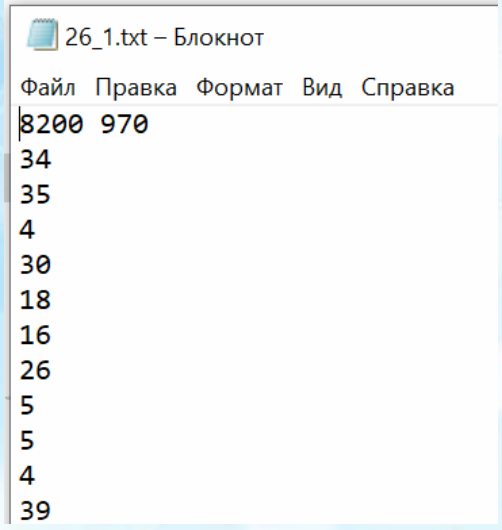
Входные данные.
В первой строке входного файла находятся два числа: S – размер свободного места на диске (натуральное число, не превышающее 10 000) и N – количество пользователей (натуральное число, не превышающее 1000). В следующих N строках находятся значения объёмов файлов каждого пользователя (все 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



```python
f=open('26_1.txt')  # открываем файл
s,n = list(map(int,f.readline().split())) # считываем объем и кол-во пользователей
a=[]   # пустой список для записи исходных данных
# записываем данные в список
for i in range(n):
    a.append(int(f.readline()))
a.sort()  # сортируем список по возрастанию
b=[] # список для файлов, включенных в архив
for i in range(n):  # перебираем список a и добавляем файлы в список b пока выполняется
    if sum(b) + a[i] <= s:
        b.append(a[i])
    else:
        break   # остановка при превышении объема
b=b[:-1]   # формируем список без последнего файла (текущего наибольшего)
# перебираем  список a с конца и подставляем наибольший файл ддо выполнения условия
for i in range(len(a)-1, -1, -1):
    if sum(b) + a[i] <= s:
        b.append(a[i])
        break
print(len(b), b[-1]) # вывод результата
```

26_1.txt – Блокнот

Файл Правка Формат Вид Справка

```
8200 970
34
35
4
30
18
16
26
5
5
4
39
```

Поиск в файлах | Поиск

Отладки ввода/вывода (стан

```
568 50
```

26

В магазине для упаковки подарков есть N кубических коробок. Самой интересной считается упаковка подарка по принципу матрёшки – подарок упаковывается в одну из коробок, та в свою очередь в другую коробку и т.д. Одну коробку можно поместить в другую, если длина её стороны хотя бы на 3 единицы меньше длины стороны другой коробки. Определите наибольшее количество коробок, которое можно использовать для упаковки одного подарка, и максимально возможную длину стороны самой маленькой коробки, где будет находиться подарок. Размер подарка позволяет поместить его в самую маленькую коробку.

Входные данные

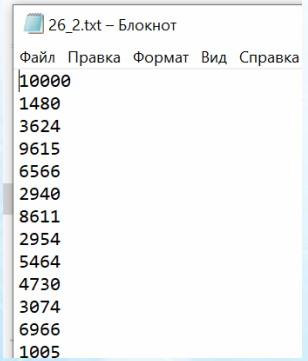
В первой строке входного файла находится число N – количество коробок в магазине (натуральное число, не превышающее 10 000). В следующих N строках находятся значения длин сторон коробок (все числа натуральные, не превышающие 10 000), каждое – в отдельной строке.

Запишите в ответе два целых числа: сначала нав
коробок, которое можно использовать для упаковки с
максимально возможную длину стороны самой мален
наборе.

Типовой пример организации данных во входном файле

```
5
43
40
32
40
30
```

*Пример входного файла приведён для пяти коро
минимальная допустимая разница между длинам
подходящих для упаковки «матрёшкой», составляет 3
При таких исходных данных условию задачи удо
коробок с длинами сторон 30, 40 и 43 или 32, 40 и
т.е. количество коробок равно 3, а длина стороны сам
равна 32.*

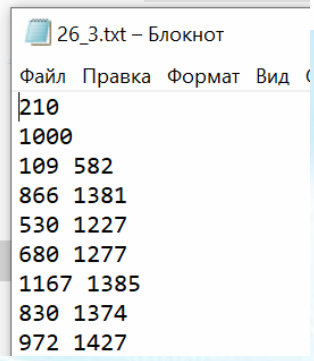



```python
f=open('26_2.txt')  # открываем файл
n=int(f.readline())  # количество коробок
a=[] # список исходных даннных
for i in range(n):  # считываем данные
    a.append(int(f.readline()))
a.sort(reverse=True)  # сортировка списка по убыванию
k=1    # количесвто коробок в текущий момент
p=a[0]  # длина последней коробки, которую используем
for i in range(1, len(a)):  # цикл перебора
    if p-a[i]>=3:
        k=k+1
        p=a[i]
print(k, p)
```

Задание 26 (№7626).

В камере хранения аэропорта есть *K* ячеек для хранения багажа туристов. Все ячейки пронумерованы, начиная с единицы. Известно время, в которое каждый турист придёт оставить свой багаж, и в какое время он заберёт его. С приходом каждого туриста его багаж кладётся в свободную ячейку с наименьшим номером. Для того, чтобы разгрузить или загрузить ячейку багажом, необходима 1 минута. Со следующей минуты можно положить в освободившуюся ячейку багаж другого туриста. Если турист пришёл, но свободных ячеек нет – он багаж оставить не может, поэтому уходит.

Определите, сколько всего туристов придут и оставят свой багаж в ячейках за 24 часа и номер ячейки, в которую положат последний багаж. Если вариантов выбрать ячейку несколько – выберите свободную ячейку с наименьшим номером.



Входные данные

В первой строке входного файла находится число *K* – количество ячеек в аэропорту (натуральное число, не превышающее 1000). Во второй строке находится число *N* – количество туристов, которые собираются воспользоваться ячейками для багажа. В следующих *N* строках находятся два значения: минута размещения багажа и минута, до которого планируется хранить багаж в ячейке, отсчёт ведётся от начала суток (все числа неотрицательные, не превышающие 1440), для каждого туриста – в отдельной строке.

Запишите в ответе два целых числа: сначала количество туристов, которое сможет воспользоваться ячейками для багажа за 24 часа, затем наименьший номер ячейки, в которую положат последний багаж.

Типовой пример организации данных во входном файле

2
5
30 60
40 1110
59 60
61 120
1230 1440

При таких исходных данных первый, второй, четвёртый и пятый туристы смогут воспользоваться ячейками. Последний турист оставит свой багаж в первой ячейке (так как первая и вторая ячейка будут свободны).

```
f = open('26_3.txt')  # открываем файл
k = int(f.readline()); n = int(f.readline()) # количество ячеек и тури
a = [] # пустой список для исходных данных
for i in range(n):  # записываем данные в список (пары чисел)
    a.append(list(map(int,f.readline().split())))
a.sort()  # сортируем список по возрастанию
b = [-1]*k # список занятости ячеек
count = 0; p = -1  # количество туристов, разместивших багаж и номер ячейки
for x in a:  # просматриваем пары чисел
    start,end = x     # время начала и окончания размещение текущего багажа
    for i in range(k):  #перебор ячеек (куда положить)
        if start > b[i]:
            b[i] = end
            count+=1
            p = i+1
            break       # если разместили, то переходим к следующему
print(count,p)
```

26_3.txt – Блокнот

Файл Правка Формат Вид

```
210
1000
109 582
866 1381
530 1227
680 1277
1167 1385
830 1374
972 1427
```

Поиск в файлах

Отладки ввода/в

581 59

Задание № 27

Уровень сложности: высокий

Требуется использование специализированного программного обеспечения: да

Максимальный балл: 2 первичных балла

Примерное время выполнения задания: 40 минут

Проверяемые элементы содержания: умение создавать собственные программы (20–40 строк) для анализа числовых последовательностей

Элементы содержания, проверяемые заданием: построение алгоритмов и практические вычисления

Задание 27 (№7275).

У медицинской компании есть N пунктов приёма биоматериалов на анализ. Все пункты расположены вдоль автомагистрали и имеют номера, соответствующие расстоянию от нулевой отметки до конкретного пункта. Известно количество пробирок, которое ежедневно принимают в каждом из пунктов.Компания планирует открыть лабораторию в одном из имеющихся пунктов. Перевозить биоматериалы разрешается на расстояние не более M. Пробирки перевозят в специальных транспортировочных контейнерах вместимостью не более 30 штук. Каждый транспортировочный контейнер используется для доставки пробирок только из одного пункта приёма, при этом из каждого пункта приёма может быть доставлено не более одного контейнера с неполной загрузкой. Пункт для лаборатории выбрали таким образом, чтобы количество доставляемых туда контейнеров с пробирками было максимальным. Определите необходимое количество контейнеров для доставки пробирок в эту лабораторию.

27B.txt

```
1200000 12013
11 347
16 146
17 102
18 301
26 266
27 91
35 44
47 55
```

27A.txt – Блокнот

Файл Правка Формат

```
100 113
4 366
8 438
25 315
27 60
38 192
56 339
```

Входные данные

Даны два входных файла (файл A и файл B), каждый из которых в первой строке содержит два числа N и M (1 ≤ N ≤ 10 000 000, 1 ≤ M ≤ 10 000 000) – количество пунктов приёма биоматериалов и максимальное расстояние, на которое разрешено перевозить биоматериалы. В каждой из следующих N строк находятся два числа: номер пункта и количество пробирок, принимаемых на этом пункте за сутки (все числа натуральные, количество пробирок в каждом пункте не превышает 1000). Пункты перечислены в порядке их расположения вдоль автомагистрали, считая от нулевой отметки.

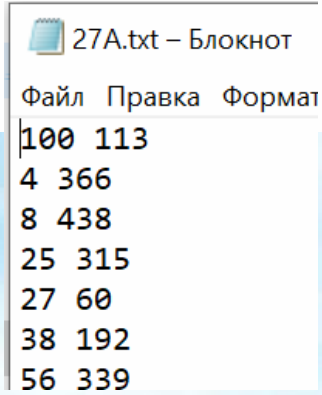
В ответе укажите два числа: сначала значение искомой величины для файла A, затем – для файла B.

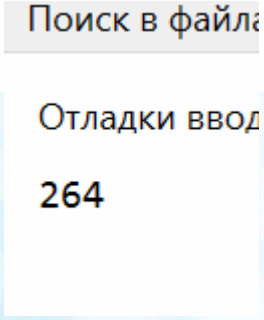
Типовой пример организации данных во входном файле

```
6 3
1 100
3 200
6 4
7 3
8 2
10 195
```

При таких исходных данных и вместимости транспортировочного контейнера, составляющей 96 пробирок, компании выгодно открыть лабораторию в пункте 3. В этом случае количество контейнеров в ней составит: 2 + 3 + 1.

```
f = open('27A.txt')   # открываем файл
n,m = map(int,f.readline().split())  # количество пунктов и максимальное расстояние
a = []  # список исходных данных
for x in f:  # формируем список перебором
    r,k = map(int,x.split())   # растояние до пункта и количество пробирок
    a.append([r,k//30+(k%30>0)])  # расстояние до пункта и количество контейнеров
maxK = 0   # максимальное количество контейнеров
for i in range(n):
    count = 0      # количесвто контейнеров для пункта i
    for j in range(n):   # перебор пунктов
        if abs(a[j][0]-a[i][0])<=m:      # проверка расстояния
            count+=a[j][1]
    maxK = max(maxK,count)  # поиск максимума
print(maxK)
```





```python
f = open('27B.txt')   # открываем файл
n,m = map(int,f.readline().split())  # количество пунктов и максимальное расстояние
a = [0]*10**7  # динамический массив (автомагистраль: расстояние (индекс) + количество контейнеров)
for x in f:  # формируем список перебором
    r,k = map(int,x.split())   # растояние до пункта и количество пробирок
    a[r] = k//30+(k%30>0)  # заполняем динамический массив (количество контейнеров в пункте r)
maxK = count = sum(a[:2*m+1])  # начальное maxK и текущее количество контейнеров в рамках диапазона 2*m+1
for i in range(1, len(a)-2*m):  # сдвигаем диапазон на одну позицию
    count = count - a[i-1] + a[i+2*m]   #  вычисляем текущее количесвто контейнеров в рамках диапазона
    if a[i+m] != 0:   # если есть в уентре диапазона пункт
        maxK = max(maxK,count)  # сравниваем максимальное и текущее количество контейнеров
print(maxK)
```