Программное решение 26ых задач

Курс подготовки к ЕГЭ по информатике 2026

Вступление

Перед началом разбора отсылаю вас к блоку по основам программирования, без которого, как мне кажется, невозможно будет понять, о чём сейчас пойдёт речь.

Подробное описание того, как работают сортировки, представлено здесь. Этот файл избыточен: главное, что надо из него понять, — как работают сортировки по ключу; остальной материал, представленный там, скорее носит ознакомительный характер и служит для общего понимания сортировок, но не является необходимым.

ЕГЭ 2022

В супермаркете проводится акция «на каждый пятый товар в чеке скидка 75%». Покупатель расположил товары на ленте так, чтобы заплатить за покупку одним чеком как можно меньше с учётом проходящей акции. Однако выяснилось, что программа для кассового аппарата не учитывает расположение товаров на ленте и сортирует цены товаров в чеке таким образом, чтобы стоимость покупки в рублях была максимально возможной.

Входные данные представлены в текстовом файле следующим образом. В первой строке входного файла записано число N — количество товаров, которые хочет оплатить покупатель (натуральное число, не превышающее 10000). В каждой из следующих N строк записана цена товара (натуральное число, не превышающее 10000).

Запишите в ответе два целых числа: сначала сумму, которую предполагал заплатить покупатель, а затем сумму, которую он заплатил за товары.

Пример входного файла:

8
80
30
50
40
115
120
15
10

При таких исходных данных если «на каждый пятый товар в чеке скидка 75%», предполагаемая и действительная суммы соответственно равны

$80 + 30 + 50 + 40 + 115 + (120 - 120 * 0,75) + 15 + 10 = 370$ и

$80 + 30 + 50 + 40 + 115 + 120 + 15 + (10 - 10 * 0,75) = 452,5$.

Ответ: 370 452, 5.

Решение

Обратим внимание, что воспроизводить порядок расположения товаров на ленте для расчёта суммы — нет необходимости, так как в одном случае скидка будет на пятую часть (округлённую вниз) самых дорогих товаров, а во втором случае — самых дешёвых. Поэтому просто сортируем по убыванию/возрастанию и считаем пятую часть товаров, оказавшихся в начале, со скидкой, а остальные по полному прайсу.

```python
f=open("26_EGE2022.txt")
n=int(f.readline())

mas=[int(x) for x in f]

mas.sort()
maxSum=0

for i in range(n//5):
    maxSum+=mas[i]*0.25

for i in range(n//5,n):
    maxSum+=mas[i]

mas.reverse()
minSum=0

for i in range(n//5):
    minSum+=mas[i]*0.25

for i in range(n//5,n):
    minSum+=mas[i]

print(minSum,maxSum)
```

ЕГЭ 2023 1 и ЕГЭ 2025 1

Входной файл содержит сведения о заявках на проведение занятий в конференц-зале. В каждой заявке указаны время начала и время окончания мероприятия (в минутах от начала суток). Если время начала одного мероприятия меньше времени окончания другого, то провести можно только одно из них. Если время окончания одного мероприятия совпадает с временем начала другого, то провести можно оба. Определите максимальное количество мероприятий, которое можно провести в конференц-зале, и самое позднее время окончания последнего мероприятия.

Входные данные

Первая строка входного файла содержит натуральное число N ($1 \leqslant N \leqslant 1000$) — количество заявок на проведение мероприятий. Следующие N строк содержат пары чисел, обозначающих время начала и время окончания мероприятий. Каждое из чисел натуральное, не превосходящее 1440.

Запишите в ответе два числа: максимальное количество мероприятий, которое можно провести в конференц-зале, и самое позднее время окончания последнего мероприятия (в минутах от начала суток).

Пример входного файла:

5
10 150
100 110
131 170
131 180
120 130

При таких исходных данных можно провести максимум три мероприятия, например, по заявкам 2, 3 и 5. Конференц-зал освободится самое позднее на 180-й минуте, если состоятся мероприятия по заявкам 2, 4, 5. Ответ: 3 180.

Решение

Логика решения следующая: чем позже заканчивается мероприятие, тем больше времени остаётся для проведения остальных. Соответственно, отсортируем мероприятия по времени окончания (второму столбцу) и будем брать новое мероприятие каждый раз, когда время его начала не меньше времени окончания последнего выбранного мероприятия.

```python
def keySort(x):
    return x[1],x[0]

f=open("26_EGE2023_1.txt")
n=int(f.readline())
mas=[list(map(int,x.split())) for x in f]

mas.sort(key=keySort)
c=0
events=[]

for x in mas:
    if events==[] or x[0]>=events[-1][1]:
        events.append(x)
        c+=1

print(c)
```

Теперь, когда все мероприятия на руках, просто найдём мероприятие с самым поздним окончанием, начало которого не меньше времени окончания предпоследнего уже взятого мероприятия.

```python
def keySort(x):
    return x[1],x[0]

f=open("26_EGE2023_1.txt")
n=int(f.readline())
mas=[list(map(int,x.split())) for x in f]

mas.sort(key=keySort)
c=0
events=[]

for x in mas:
    if events==[] or x[0]>=events[-1][1]:
        events.append(x)
        c+=1

res2=0
for x in mas:
    if x[0]>=events[-2][1]:
        res2=max(res2,x[1])

print(c,res2)
```

ЕГЭ 2023 2 и ЕГЭ 2025 2

На вход программе подаётся N пар чисел (все числа различны): время шлифовки детали и время её покраски. Детали размещаются на конвейерной ленте по следующему принципу:

— все заданные числа сортируются в порядке возрастания;

— если очередное значение — это время шлифовки некоторой детали, то эта деталь размещается на первое свободное место с начала конвейера;

— если очередное значение — это время покраски некоторой детали, то эта деталь размещается на первое свободное место с конца конвейера;

— если очередное значение — это время шлифовки или покраски некоторой детали, которая уже размещена на конвейере, то это значение просто пропускается.

Необходимо найти номер последней детали (при условии, что они пронумерованы с 1), которая будет размещена на конвейере, и количество деталей, которые будут отшлифованы и окрашены до неё.

Входные данные

В первой строке входного файла находится натуральное число N ($N \leqslant 1000$) — количество деталей. Следующие N строк содержат пары чисел, обозначающих время шлифовки и время покраски конкретной детали. Каждое из чисел натуральное, не превосходящее 10000. Запишите в ответе два числа: номер последней детали, которая будет размещена на конвейере, и количество деталей, которые будут отшлифованы и окрашены до неё.

Типовой пример организации данных во входном файле

5
106 146
48 108
49 32
38 67
149 79

При таких исходных данных номер последней детали, которая будет размещена на конвейере — 1, и до неё будет отшлифовано 2 детали.

Типовой пример имеет иллюстративный характер. Для выполнения задания используйте данные из прилагаемых файлов.

Решение

Заметим, что для определения позиции детали на конвейерной ленте достаточно знать минимальное число для каждой детали. То есть если минимальное число в паре — это время шлифовки, то деталь отправляется в начало конвейера. Если время покраски, то в конец. Максимальное число при этом никак не используется.

Далее обратим внимание на сам вопрос. Нужно определить номер последней детали, которая будет размещена на конвейере. То есть, по сути, если отсортировать все пары по минимальному значению в них, то элемент, который окажется в конце, и будет размещён последним.

```python
def keySort(x):
    return min(x[1:])

f=open("26_EGE2023_2.txt")
n=int(f.readline())
mas=[]
for i in range(n):
    mas.append([i+1]+list(map(int,f.readline().split())))

mas.sort(key=keySort)
print(mas[-1][0])
```

И теперь, чтобы ответить на вопрос, сколько деталей будут отшлифованы и окрашены до неё, необходимо найти, сколько среди остальных деталей были помещены в начало конвейера, так как именно эти детали в итоге будут отшлифованы и окрашены до последней.

```python
def keySort(x):
    return min(x[1:])

f=open("26_EGE2023_2.txt")
n=int(f.readline())
mas=[]

for i in range(n):
    mas.append([i+1]+list(map(int,f.readline().split())))

mas.sort(key=keySort)

res=0
for i in range(n-1):
    if min(mas[i][1:])==mas[i][1]:
        res+=1

print(mas[-1][0],res)
```

ЕГЭ 2023 3

Компания, занимающаяся перевозкой грузов, упаковывает каждый груз в отдельный контейнер. В связи со сбоем в работе программы, определяющей, какой груз упаковывается в какой контейнер, под каждый груз контейнер был выделен случайным образом (некоторые грузы не помещаются в выделенные под них контейнеры). Необходимо написать программу, которая для заданных грузов перераспределит имеющиеся контейнеры таким образом, чтобы упаковать как можно больше грузов.

Входные данные В первой строке входного файла находится натуральное число N ($N \leqslant 1000$) — количество пар грузов и контейнеров. В каждой из следующих N строк находятся два натуральных числа, не превышающих 100000 — масса груза и максимальная масса, которую вмещает контейнер.

В ответе запишите два числа: максимальное количество грузов, которые возможно упаковать, и максимальную массу груза, который возможно упаковать.

Типовой пример организации данных во входном файле

10
12 31
16 34
51 73
22 28
83 18
10 75
37 29
19 71
36 14
65 11

При таких исходных данных ответом будут являться числа 8 и 65. Например, мы можем выделить следующие пары грузов и контейнеров: 10 и 11; 65 и 73; 12 и 14; 51 и 75; 16 и 18; 22 и 34; 37 и 71; 19 и 29.

Типовой пример имеет иллюстративный характер. Для выполнения задания используйте данные из прилагаемых файлов.

Решение

Если разделить грузы и контейнеры на два списка, оба отсортировать по возрастанию, то задача сведётся к тому, чтобы перебирать контейнеры по возрастанию и для каждого брать минимальный по массе подходящий груз среди тех, которые ещё не помещены ни в один контейнер. Если такого груза нет, то этот контейнер просто пропускается.

```python
f=open("26_EGE2023_3.txt")
n=int(f.readline())
items=[]
boxes=[]

for i in range(n):
    it,bx=map(int,f.readline().split())
    items.append(it)
    boxes.append(bx)

items.sort()
boxes.sort()

nextItem=0
for i in range(n):
    if items[nextItem]<=boxes[i]:
        nextItem+=1

print(nextItem)
```

Так как в Python нумерация начинается с нуля, индекс следующего груза совпадает с количеством уже упакованных в контейнеры грузов, а значит, именно его мы и выводим как первый ответ.

Второй вопрос, по сути, не связан с первым. Мы можем просто найти максимальный груз, который помещается в самый большой контейнер, что и будет вторым ответом. Для этого развернём отсортированный по возрастанию items и найдём первый элемент, который помещается в boxes[-1].

```python
f=open("26_EGE2023_3.txt")
n=int(f.readline())
items=[]
boxes=[]

for i in range(n):
    it,bx=map(int,f.readline().split())
    items.append(it)
    boxes.append(bx)

items.sort()
boxes.sort()

nextItem=0
for i in range(n):
    if items[nextItem]<=boxes[i]:
        nextItem+=1

print(nextItem)

items.reverse()
for i in range(n):
    if items[i]<=boxes[-1]:
        print(items[i])
        break
```

ЕГЭ 2024 1 и Досрок 2026

Входной файл содержит информацию о товарах в некотором магазине. У каждого товара есть артикул, цена и статус (продан товар или находится в наличии). Гарантируется, что у товаров с одинаковым артикулом — одинаковая цена. Товары делятся на дешёвые и дорогие. Дорогим товар считается, если его цена не меньше среднего арифметического всех товаров (вне зависимости от того, продан товар или находится в наличии). Остальные считаются дешёвыми.

Необходимо определить артикул дорогого товара, единиц которого было продано больше всего. Если таких товаров несколько, берётся товар, цена которого является наименьшей. Если и таких товаров несколько, берётся товар с наибольшим артикулом. Для найденного товара необходимо записать выручку, которую получил магазин от реализации данного товара, и количество единиц данного товара в наличии.

Входные данные

В первой строке входного файла находится натуральное число N ($N \leqslant 1000$) — количество единиц товаров. В каждой из следующих N строк находятся три числа: артикул товара (натуральное число, не превышающее 100000), цена за единицу данного товара (натуральное число, не превышающее 1000) и статус товара (0 — соответствует тому, что товар продан; 1 — соответствует тому, что товар находится в наличии).

Запишите в ответе два числа — выручку за дорогой товар, количество проданных единиц которого максимально, и количество единиц данного товара, которое находится в наличии.

Типовой пример организации данных во входном файле

8
105 50 1
57 60 0
57 60 1
105 50 0
105 50 0
57 60 1
35 10 1
36 10 1

При таких исходных данных ответом будут являться числа 100 и 1. Нам подходит товар с артикулом 105 (дорогой товар, количество проданных единиц которого равно 2). За него магазин получил выручку в размере 100, и сейчас в магазине в наличии есть 1 такой товар.

Типовой пример имеет иллюстративный характер. Для выполнения задания используйте данные из прилагаемых файлов.

Решение

Для начала найдём среднюю цену всех товаров.

```
f=open("26_EGE2024_1.txt")
n=int(f.readline())
mas=[]
for i in range(n):
    mas.append(list(map(int,f.readline().split())))

sr=0
for i in range(n):
    sr+=mas[i][1]
sr/=n
```

Когда эта информация получена, мы можем отсортировать товары по цене, в случае совпадения — по наличию, а если и наличие совпадает — по артикулу. Здесь нам важна не столько сама сортировка, сколько группировка товаров: чтобы все проданные товары с некоторым артикулом находились рядом друг с другом.

```
def keySort(x):
    return x[1],x[2],x[0]

f=open("26_EGE2024_1.txt")
n=int(f.readline())
mas=[]
for i in range(n):
    mas.append(list(map(int,f.readline().split())))

sr=0
for i in range(n):
    sr+=mas[i][1]
sr/=n

mas.sort(key=keySort)
```

После чего пройдёмся по всем товарам и найдём артикул дорогого товара, у которого количество проданных единиц максимально.

```python
def keySort(x):
    return x[1],x[2],x[0]

f=open("26_EGE2024_1.txt")
n=int(f.readline())
mas=[]
for i in range(n):
    mas.append(list(map(int,f.readline().split())))

sr=0
for i in range(n):
    sr+=mas[i][1]
sr/=n

mas.sort(key=keySort)

maxCount=0
count=0
art=0
for i in range(n):
    if mas[i][1]>=sr:
        if mas[i][0]!=mas[i-1][0]:
            count=0
        if mas[i][2]==0:
            count+=1
    else:
        count=0
    if count>=maxCount:
        maxCount=count
        art=mas[i][0]
```

Теперь дело за малым: посчитать выручку от продажи этого товара и количество его единиц, находящихся в наличии.

```python
def keySort(x):
    return x[1],x[2],x[0]

f=open("26_EGE2024_1.txt")
n=int(f.readline())
mas=[]
for i in range(n):
    mas.append(list(map(int,f.readline().split())))

sr=0
for i in range(n):
    sr+=mas[i][1]
sr/=n

mas.sort(key=keySort)

maxCount=0
count=0
art=0
for i in range(n):
    if mas[i][1]>=sr:
        if mas[i][0]!=mas[i-1][0]:
            count=0
        if mas[i][2]==0:
            count+=1
    else:
        count=0
    if count>=maxCount:
        maxCount=count
        art=mas[i][0]

summ=0
countHave=0
for i in range(n):
    if mas[i][0]==art:
        if mas[i][2]==0:
            summ+=mas[i][1]
        else:
            countHave+=1

print(summ,countHave)
```

ЕГЭ 2024 2

Входной файл содержит информацию о занятых местах в концертном зале. Покупатель хочет купить билет на такое место в ряду, чтобы непосредственно перед ним как можно больше идущих подряд кресел с таким же номером было свободно. Если места, удовлетворяющие этому условию, есть в нескольких рядах, то нужно выбирать ряд, расположенный как можно ближе к сцене. В ответ необходимо записать два числа: номер ряда, в котором находится подходящее место, и непосредственно номер искомого места.

Входные данные

В первой строке входного файла находятся три натуральных числа N — количество занятых мест ($N \leqslant 10000$), M — количество рядов в зале ($M \leqslant 1000$), K — количество мест в каждом ряду ($K \leqslant 1000$). В каждой из следующих N строк находятся пары чисел: номер ряда и номер места занятого кресла соответственно (первое число не больше M, второе не больше K). Запишите в ответе два числа: номер ряда и номер места в этом ряду, перед которым находится максимальное количество идущих подряд свободных кресел.

Типовой пример организации данных во входном файле

4 4 4
1 2
3 4
2 1
4 3

При таких исходных данных ответом будут являться 3 и 3. В 3 ряду можно посадить человека на 3 место, и перед ним будет ровно 2 свободных места.

Типовой пример имеет иллюстративный характер. Для выполнения задания используйте данные из прилагаемых файлов.

Решение

Идея будет следующая. Мы будем перебирать все занятые места и в качестве потенциального ответа рассматривать места, находящиеся непосредственно перед занятыми. Для удобства добавим несуществующий ряд после последнего, чтобы рассматривать в том числе свободные места на последнем ряду.

```
f=open("26_EGE2024_2.txt")
n,m,k=map(int,f.readline().split())
mas=[list(map(int,f.readline().split())) for i in range(n)]

for i in range(k):
    mas.append([m+1,i+1])
```

Теперь отсортируем по номерам мест, а в случае совпадения — по ряду.

```python
def keySort(x):
    return x[1],x[0]

f=open("26_EGE2024_2.txt")
n,m,k=map(int,f.readline().split())
mas=[list(map(int,f.readline().split())) for i in range(n)]

for i in range(k):
    mas.append([m+1,i+1])

mas.sort(key=keySort)
```

Теперь первый элемент в списке — это занятое место с номером 1 в самом близком к сцене ряду. Его возьмём как потенциальный ответ.

```python
def keySort(x):
    return x[1],x[0]

f=open("26_EGE2024_2.txt")
n,m,k=map(int,f.readline().split())
mas=[list(map(int,f.readline().split())) for i in range(n)]

for i in range(k):
    mas.append([m+1,i+1])

mas.sort(key=keySort)

maxCount=mas[0][0]-2
res=mas[0][0]-1
res2=mas[0][1]
```

Теперь начнём перебирать остальные элементы. Если это первое место с таким номером, то, посадив человека прямо перед ним, мы увидим, что все места до сцены свободны. Если же это не первое место с таким номером, то свободные места кончаются на месте с тем же номером в предыдущем занятом ряду. Далее проверяем, максимальное ли это количество.

```python
def keySort(x):
    return x[1],x[0]

f=open("26_EGE2024_2.txt")
n,m,k=map(int,f.readline().split())
mas=[list(map(int,f.readline().split())) for i in range(n)]

for i in range(k):
    mas.append([m+1,i+1])

mas.sort(key=keySort)

maxCount=mas[0][0]-2
res=mas[0][0]-1
res2=mas[0][1]

for i in range(1,n):
    if mas[i][1]!=mas[i-1][1]:
        count=mas[i][0]-2
    else:
        count=mas[i][0]-2-mas[i-1][0]
    if count>maxCount:
        maxCount=count
        res=mas[i][0]-1
        res2=mas[i][1]
    elif count==maxCount and mas[i][0]-1<res:
        maxCount=count
        res=mas[i][0]-1
        res2=mas[i][1]

print(res,res2)
```

ЕГЭ 2024 3

Входной файл содержит информацию о занятых местах в концертном зале. Покупатели хотят получить два билета на такие соседние места в одном ряду, чтобы все кресла перед ними с такими же номерами были свободны, а ряд находился как можно дальше от сцены. Нумерация рядов и мест ведётся с 1. Гарантируется, что хотя бы одна такая пара мест в зале присутствует. Определите искомый номер ряда и номер наибольшего из подходящих мест.

Входные данные

В первой строке входного файла находятся три натуральных числа N — количество занятых мест ($N \leqslant 10000$), M — количество рядов в зале ($M \leqslant 100000$), K — количество мест в каждом ряду ($K \leqslant 100000$). В каждой из следующих N строк находятся пары чисел: номер ряда и номер места занятого кресла соответственно (первое число не больше M, второе не больше K).

Запишите в ответе два числа: максимальный номер ряда, в котором есть два соседних свободных места таких, что перед ними все места с такими же номерами свободны, и максимальный номер места, которое может оказаться в такой паре в найденном ряду.

Типовой пример организации данных во входном файле

4 5 5
1 2
3 4
2 1
4 3

При таких исходных данных ответом будут являться числа 2 и 5. В 5 и 4 рядах подходит только пятое место. В 3 ряду подходят 3 и 5 места, но они не являются соседними, а во 2 ряду подходят 3, 4 и 5 места.

Типовой пример имеет иллюстративный характер. Для выполнения задания используйте данные из прилагаемых файлов.

Решение

Считаем данные, добавим несуществующий задний ряд и отсортируем по аналогии с предыдущей задачей.

```
def keySort(x):
    return x[1],x[0]

f=open("26_EGE2024_3.txt")
n,m,k=map(int,f.readline().split())
mas=[list(map(int,f.readline().split())) for i in range(n)]

for i in range(k):
    mas.append([m+1,i+1])

mas.sort(key=keySort)
```

Далее обратим внимание, что среди мест с одинаковым номером нас интересуют только те, которые находятся в ряду, наиболее близком к сцене, то есть в ряду с минималь-

ным номером. Это необходимо, чтобы все места непосредственно перед выбранным были свободны. Отфильтруем данные: создадим список, в который для каждого номера места будем записывать только минимальный номер ряда.

```
def keySort(x):
    return x[1],x[0]

f=open("26_EGE2024_3.txt")
n,m,k=map(int,f.readline().split())
mas=[list(map(int,f.readline().split())) for i in range(n)]

for i in range(k):
    mas.append([m+1,i+1])

mas.sort(key=keySort)

cleanMas=[mas[0]]
for i in range(1,n):
    if cleanMas[-1][1]!=mas[i][1]:
        cleanMas.append(mas[i])
```

Далее для каждой пары мест будем брать ту, которая находится в ряду, наиболее близком к сцене. А в качестве потенциального ответа брать пару мест с этими же номерами, расположенную непосредственно перед выбранным местом. Среди потенциальных ответов выбираем тот, у которого ряд наибольший.

```python
def keySort(x):
    return x[1],x[0]

f=open("26_EGE2024_3.txt")
n,m,k=map(int,f.readline().split())
mas=[list(map(int,f.readline().split())) for i in range(n)]

for i in range(k):
    mas.append([m+1,i+1])

mas.sort(key=keySort)

cleanMas=[mas[0]]
for i in range(1,n):
    if cleanMas[-1][1]!=mas[i][1]:
        cleanMas.append(mas[i])

res=0
res2=0
for i in range(1,len(cleanMas)):
    row=min(cleanMas[i-1][0],cleanMas[i][0])-1
    if row>=res:
        res=row
        res2=i+1

print(res,res2)
```

ФИПИ 290F15 и ЕГЭ 2023 4

Входной файл содержит заявки пассажиров, желающих сдать свой багаж в камеру хранения. В заявке указаны время сдачи багажа и время освобождения ячейки (в минутах от начала суток). Багаж одного пассажира размещается в одной свободной ячейке с минимальным номером. Ячейки пронумерованы начиная с единицы. Размещение багажа в ячейке или её освобождение происходит в течение 1 мин. Багаж можно поместить в только что освобождённую ячейку начиная со следующей минуты. Если есть выбор, кого из пассажиров обслужить, пассажир выбирается таким образом, чтобы количество обслуженных пассажиров в итоге было максимальным. Если в момент сдачи багажа свободных ячеек нет, то пассажир уходит. Определите, сколько пассажиров сможет сдать свой багаж в течение 24 ч и какой номер будет иметь ячейка, которую займут последней. Если таких ячеек несколько, укажите минимальный номер ячейки.

Входные данные

В первой строке входного файла находится натуральное число K, не превышающее 1000, — количество ячеек в камере хранения.

Во второй строке — натуральное число N ($N \leqslant 1000$), обозначающее количество пассажиров. Каждая из следующих N строк содержит два натуральных числа, каждое из которых не превышает 1440: указанное в заявке время размещения багажа в ячейке и время освобождения ячейки (в минутах от начала суток).

Запишите в ответе два числа: количество пассажиров, которые смогут воспользоваться камерой хранения, и номер последней занятой ячейки.

Типовой пример организации данных во входном файле

2
5
30 60
40 1000
59 60
61 1000
1010 1440

При таких исходных данных положить вещи в камеру хранения смогут первый, второй, четвёртый и пятый пассажиры. Последний пассажир положит вещи в ячейку 1, так как ячейки 1 и 2 будут свободны.

Типовой пример имеет иллюстративный характер. Для выполнения задания используйте данные из прилагаемых файлов.

Решение

Идея в том, чтобы буквально симулировать данный процесс. Если мы отсортируем список всех людей по времени, когда они приходят, то далее задача сведётся к тому, чтобы каждую минуту проверять, не пришли ли новые люди, и брать их багаж в соответствующие ячейки. Давайте это реализуем.

```
f=open("26_290F15.txt")
k=int(f.readline())
n=int(f.readline())
people=[]
for i in range(n):
    people.append(list(map(int,f.readline().split())))

people.sort()

boxes=[0]*k
nextHuman=0
for time in range(1,1441):
    while(nextHuman<n and people[nextHuman][0]<=time):
        for j in range(len(boxes)):
            if(boxes[j]==0):
                boxes[j]=people[nextHuman][1]
                break
        nextHuman+=1
    for j in range(len(boxes)):
        if(boxes[j]==time):
            boxes[j]=0
```

В данном случае boxes — список ячеек для хранения, и в нём содержится либо время, когда багаж из ячейки должны забрать, либо 0, в случае, если ячейка свободна.

Теперь осталось ввести переменные, с помощью которых мы будем подсчитывать количество обслуженных пассажиров и запоминать номер ячейки, в которую был помещён багаж последнего из них.

```
f=open("26_290F15.txt")
k=int(f.readline())
n=int(f.readline())
people=[]
for i in range(n):
    people.append(list(map(int,f.readline().split())))

people.sort()

boxes=[0]*k
nextHuman=0
c=0
last=0
for time in range(1,1441):
    haveHuman=False
    while(nextHuman<n and people[nextHuman][0]<=time):
        for j in range(len(boxes)):
            if(boxes[j]==0):
                boxes[j]=people[nextHuman][1]
                c+=1
                if haveHuman==False:
                    last=j+1
                    haveHuman=True
                break
        nextHuman+=1
    for j in range(len(boxes)):
        if(boxes[j]==time):
            boxes[j]=0

print(c,last)
```

С помощью haveHuman мы гарантируем, что запишем минимальный номер ячейки среди тех, которые были заняты в последнюю минуту, когда вообще происходило обслуживание.