Задание 26 (№1) Сибигатулин Рамиль
https://vk.com/infmat100
Условие:

На премьеру нового фильма пришли практически все жители небольшого города. Известны номера ряда и места выкупленных билетов. Зал имеет форму квадрата, причём кресла, расположенные на диагоналях квадрата ярко-красного цвета, а остальные – темно-красного. Таким образом, зал диагоналями разбивается на четверти. Оказалось, что в этот день празднуется день «Самой занятой четверти кинозала». Всем, кто на этой четверти полагается по окончании киносеанса вернуть деньги за просмотр (Если таких четвертей будет несколько, то деньги вернут тем людям, которые сидели в четверти с минимальным номером, дабы напомнить о несправедливости этого мира). Определите, день какой четверти будет праздноваться в этом году (верхняя – 1, правая – 2, нижняя – 3, левая - 4) и скольким посетителям вернут деньги, с учетом того, что люди, кресла которых ярко-красные, НЕ могут претендовать на возврат денег?

В первой строке входного файла содержится число N – количество рядов в кинозале, во второй строке M – общее количество занятых мест, а в остальных строках по два числа: номер ряда и места выкупленного билета.

В ответ запишите два числа: сначала номер самой занятой четверти зала (цифру 1-4) и количество людей на ней.

Решение: (программа)

1. Для начала считаем N и M, количество рядов в кинозале и общее количество занятых мест соответственно.
2. Для решения нам нужно создать матрицу (mat), чтобы наглядно отобразить ситуацию в зале. Создавая матрицу, заполним ее точками.
3. Проходимся по файлу циклом, считываем номер ряда и места и ставим на эти места в матрицу *, таким образом в конце получим: «*» - занятые места, «.» – пустые места.

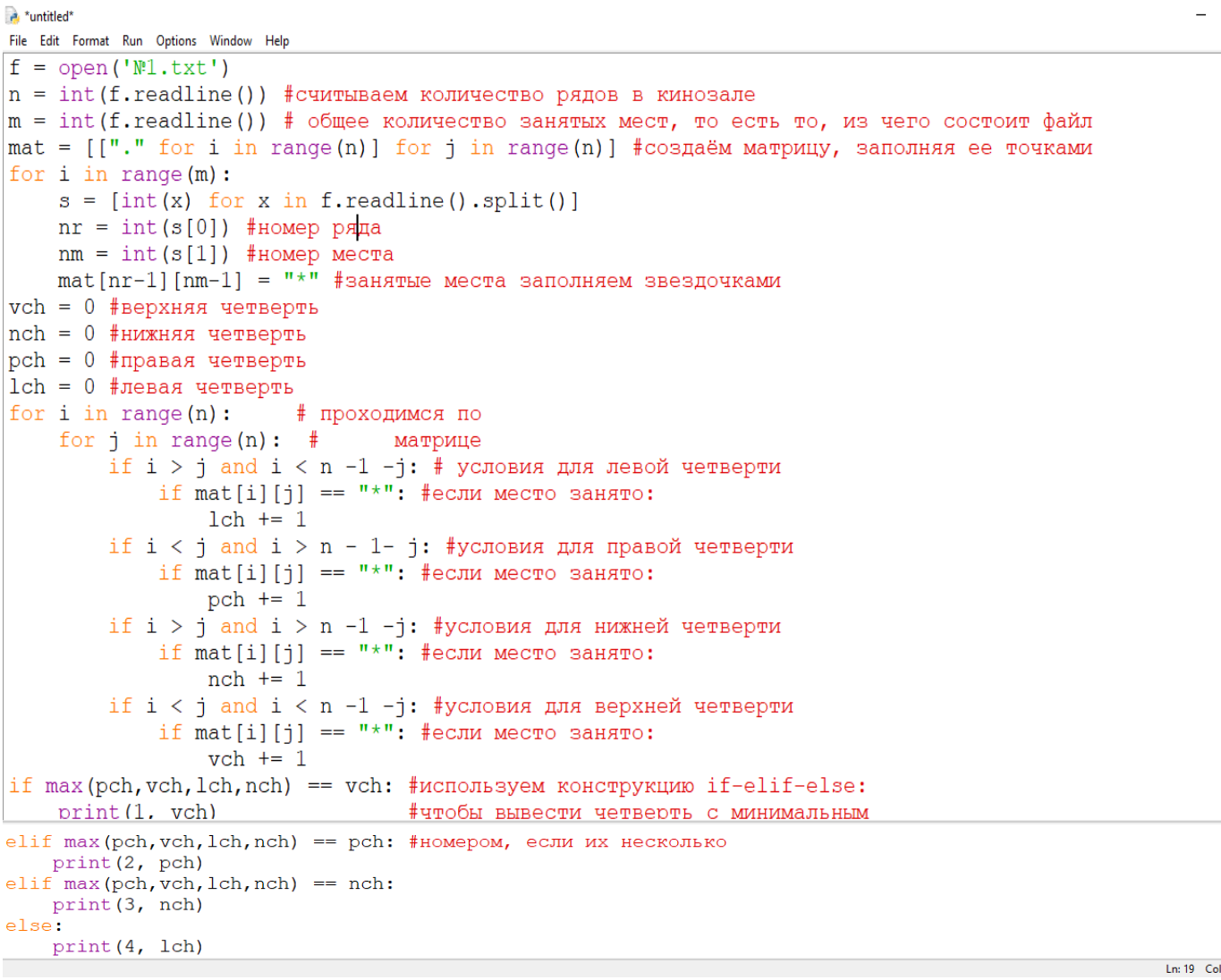
   Примечание: для того чтобы найти место и ряд, нам нужно вычитать единицу от номеров и ряда и места:
   `mat[nr-1][nm-1] = "*"` Это нужно для того чтобы мы не выходили за пределы, потому что индексы с 0, а номера с единицы.

4. Далее создаём переменные, которые будут отражать количество занятых мест в четвертях:

```
vch = 0 #верхняя четверть
nch = 0 #нижняя четверть
pch = 0 #правая четверть
lch = 0 #левая четверть
```

5. Теперь один из самых сложных шагов. Нужно пройтись по матрице с помощью вложенных циклов, понять какая это четверть, и прибавить 1 к соответствующей переменной. Рассмотрим диагонали квадрата (зала). Элементы на диагональ с правого верхнего угла в левый нижний имеют индексы i = j, то есть их можно найти как mat[i] [i]. А вот элементы на диагонали с нижнего левого угла в правый верхний имеют такую зависимость в индексах: i = n – 1 – j, следовательно это mat[i][n-j-1]. То есть элементы в области, которая выше первой диагонали имеют i < j, а в той, которая ниже – i > j. Аналогично, элементы выше второй диагонали имеют индексы: i < n – 1- j, а которая выше – i > n – 1 – j. Объединив эти ограничения, получим: верхняя четверть находится выше и той, и другой диагонали, следовательно это все элементы, у которых индекс i < j and i < n-1-j, правая четверть находится выше первой и ниже второй диагоналей, следовательно i < j and I > n – 1- j. И так далее с остальными четвертями. Стоит заметить, что знаки “>”, ”<” строгие, потому что люди, сидящие на местах, расположенных на диагоналях, нас не интересуют (есть такое условие в задаче).
6. После того, как определили к какой четверти относится текущий элемент, стоит проверить, занято это место или нет условием: if mat[i] [j] == “*”, и только после этого стоит прибавлять единицу к соответствующей переменной, что будут означать количество занятых мест в каждой четверти по отдельности.
7. В конце программы нужно воспользоваться конструкцией if-elif-else, для того чтобы вывести четверть с минимальным номером, если интересующих нас четвертей одинаковое количество.

Код Ниже (скриншот и текст)



f = open('№1.txt')

n = int(f.readline()) #считываем количество рядов в кинозале

m = int(f.readline()) # общее количество занятых мест, то есть то, из чего состоит файл

mat = [["." for i in range(n)] for j in range(n)] #создаём матрицу, заполняя ее точками

for i in range(m):

    s = [int(x) for x in f.readline().split()]

    nr = int(s[0]) #номер ряда

    nm = int(s[1]) #номер места

    mat[nr-1][nm-1] = "*" #занятые места заполняем звездочками

vch = 0 #верхняя четверть

nch = 0 #нижняя четверть

```
pch = 0 #правая четверть
lch = 0 #левая четверть
for i in range(n):     # проходимся по
    for j in range(n):  #      матрице
        if i > j and i < n -1 -j: # условия для левой четверти
            if mat[i][j] == "*": #если место занято:
                lch += 1
        if i < j and i > n - 1- j: #условия для правой четверти
            if mat[i][j] == "*": #если место занято:
                pch += 1
        if i > j and i > n -1 -j: #условия для нижней четверти
            if mat[i][j] == "*": #если место занято:
                nch += 1
        if i < j and i < n -1 -j: #условия для верхней четверти
            if mat[i][j] == "*": #если место занято:
                vch += 1
if max(pch,vch,lch,nch) == vch: #используем конструкцию if-elif-else:
    print(1, vch)               #чтобы вывести четверть с минимальным
elif max(pch,vch,lch,nch) == pch: #номером, если их несколько
    print(2, pch)
elif max(pch,vch,lch,nch) == nch:
    print(3, nch)
else:
    print(4, lch)
```

Задание 26 (№2)

Условие:

Местный Загс решил перевыполнить план и выследить своих потенциальных клиентов, ужинающих в самом престижном ресторане Клод

Моне. С этой целью сотрудник Загса получил некую информацию от персонала ресторана, представленную в входном файле. Требуется определить, сколько влюбленных парочек посетили ресторан, если по наблюдениям официантов, такие всегда выдерживали максимум 20 минут, а после уходили в более уединенное место для продолжения свидания. Именно это условие исключает возможность того, ужинали родственники или просто друзья, ведь те оставались подольше. Также нужно определить максимальное количество парочек, которые были в ресторане одновременно.

В первой строке входного файла содержится N – общее число людей, посетивших ресторан, а в следующих N строк содержится по 3 числа: номер столика, время прихода от начала суток, время ухода от начала суток (оба вторых числа - время в минутах, не превышающее 1440 и не меньшее 480).

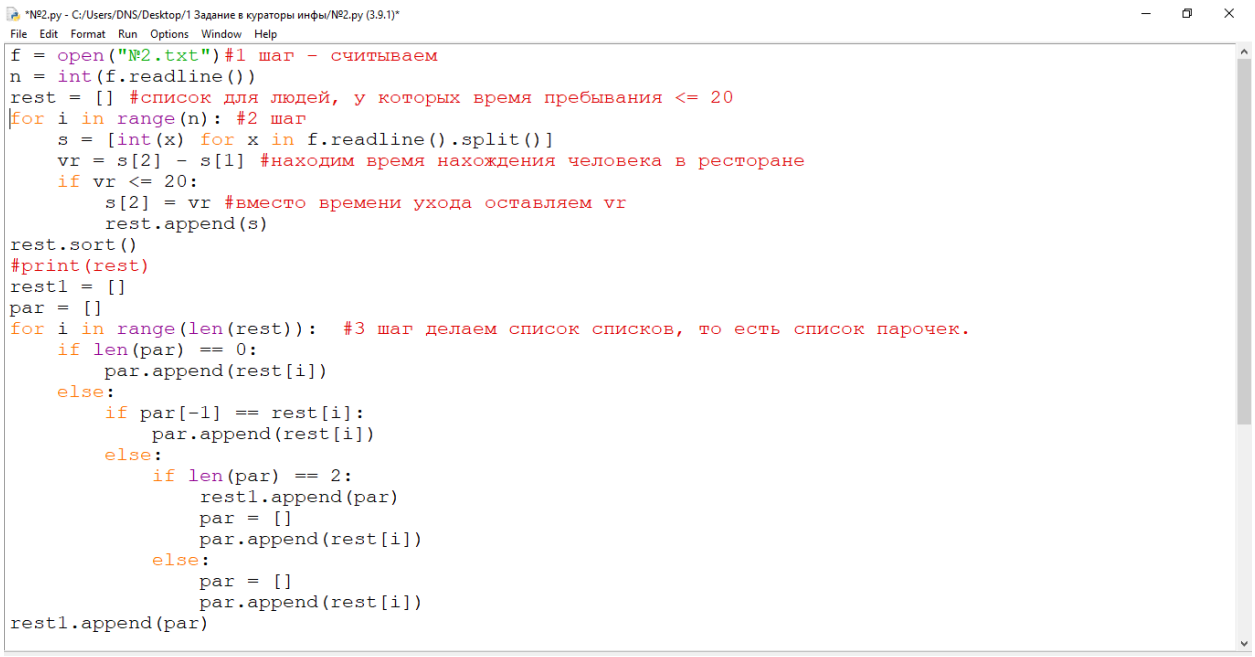
Решение: (Программа)

1. Открываем файл, считываем число N.
2. Создаём список rest и проходимся циклом по файлу, считывая каждую строчку в список s, обновляющийся на каждом шагу. У каждого посетителя заменяем время ухода на время пребывания в ресторане, для того, чтобы понять, кто находился в ресторане <= 20 и только таких посетителей добавляем в список rest. Теперь у нас есть только те посетители, которые нам могут подойти. Отсортируем наш список.
3. Но нужно проверить, сколько людей сидели за одним столиком, ведь если их было двое, то это влюбленная парочка и она нам подходит, но это может быть один, а может быть три и больше человек, которые пришли в одно время, за один столик и, просидев меньше 20 минут, просто ушли. Про это в условии ничего не написано, а значит такое может быть. Создаем список rest1, в который будем добавлять только парочек. Проходимся циклом по списку rest и добавляем посетителей с одинаковыми данными в вспомогательный список par. После проверяем длину списка par, если она == 2, то добавляем par в rest1, а иначе, просто забываем эти данные. В конце нужно не забыть, после всего цикла, добавить последний par в rest1, потому что он не пройдет по циклу до конца.
4. Теперь нужно избавиться от кучи вложенных списков и удалить из списка rest1 дубликаты. Я это делаю программой, но можно и в экселе. Я создаю список rest снова, который будет содержать данные о парах, следовательно, его длина и будет первым ответом – количество всех пар.
5. Теперь осталось найти максимальное число пар, ужинающих одновременно, для этого создаем переменную K, которая будет считать текущее количество пар в зале и на каждом шагу сравниваем с maxk, переменной, которая была создана заранее и содержит самый

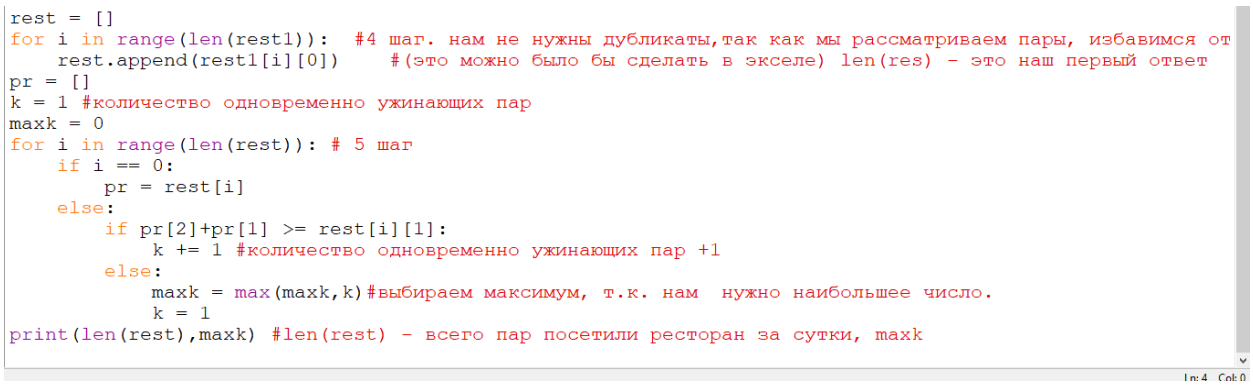
максимум кол-во пар. Проходимся по списку rest, запоминая прошлый элемент в переменную pr. Нужно, чтобы время прихода нового было меньше либо равно времени ухода прошлого, тогда к += 1, иначе сравнимаем с maxk и к = 1

6. Print(len(rest),maxk) – ответ.

Код ниже:



```
*№2.py - C:/Users/DNS/Desktop/1 Задание в кураторы инфы/№2.py (3.9.1)*
File  Edit  Format  Run  Options  Window  Help
f = open("№2.txt")#1 шаг - считываем
n = int(f.readline())
rest = [] #список для людей, у которых время пребывания <= 20
for i in range(n): #2 шаг
    s = [int(x) for x in f.readline().split()]
    vr = s[2] - s[1] #находим время нахождения человека в ресторане
    if vr <= 20:
        s[2] = vr #вместо времени ухода оставляем vr
        rest.append(s)
rest.sort()
#print(rest)
rest1 = []
par = []
for i in range(len(rest)):  #3 шаг делаем список списков, то есть список парочек.
    if len(par) == 0:
        par.append(rest[i])
    else:
        if par[-1] == rest[i]:
            par.append(rest[i])
        else:
            if len(par) == 2:
                rest1.append(par)
                par = []
                par.append(rest[i])
            else:
                par = []
                par.append(rest[i])
rest1.append(par)
```



```
rest = []
for i in range(len(rest1)):  #4 шаг. нам не нужны дубликаты,так как мы рассматриваем пары, избавимся от
    rest.append(rest1[i][0])    #(это можно было бы сделать в экселе) len(res) - это наш первый ответ
pr = []
k = 1 #количество одновременно ужинающих пар
maxk = 0
for i in range(len(rest)): # 5 шаг
    if i == 0:
        pr = rest[i]
    else:
        if pr[2]+pr[1] >= rest[i][1]:
            k += 1 #количество одновременно ужинающих пар +1
        else:
            maxk = max(maxk,k)#выбираем максимум, т.к. нам  нужно наибольшее число.
            k = 1
print(len(rest),maxk) #len(rest) - всего пар посетили ресторан за сутки, maxk
```

f = open("№2.txt")#1 шаг - считываем

n = int(f.readline())

rest = [] #список для людей, у которых время пребывания <= 20

for i in range(n): #2 шаг

```
    s = [int(x) for x in f.readline().split()]
    vr = s[2] - s[1] #находим время нахождения человека в ресторане
    if vr <= 20:
        s[2] = vr #вместо времени ухода оставляем vr
        rest.append(s)
rest.sort()
#print(rest)
rest1 = []
par = []
for i in range(len(rest)):  #3 шаг делаем список списков, то есть список парочек.
    if len(par) == 0:
        par.append(rest[i])
    else:
        if par[-1] == rest[i]:
            par.append(rest[i])
        else:
            if len(par) == 2:
                rest1.append(par)
                par = []
                par.append(rest[i])
            else:
                par = []
                par.append(rest[i])
rest1.append(par)


rest = []
for i in range(len(rest1)):  #4 шаг. нам не нужны дубликаты,так как мы рассматриваем пары, избавимся от них так.
```

```
    rest.append(rest1[i][0])    #(это можно было бы сделать в экселе) len(res) - это наш первый ответ
pr = []
k = 1 #количество одновременно ужинающих пар
maxk = 0
for i in range(len(rest)): # 5 шаг
    if i == 0:
        pr = rest[i]
    else:
        if pr[2]+pr[1] >= rest[i][1]:
            k += 1 #количество одновременно ужинающих пар +1
        else:
            maxk = max(maxk,k)#выбираем максимум, т.к. нам  нужно наибольшее число.
            k = 1
print(len(rest),maxk) #len(rest) - всего пар посетили ресторан за сутки, maxk
```