Министерство образования и науки Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Санкт-Петербургский государственный архитектурно-строительный университет»**

Строительный факультет

|  |  |
| --- | --- |
| Факультет: | Инженерной экологии и городского хозяйства |
| Кафедра: | Информационных систем и технологий |

|  |
| --- |
| ОТЧЕТ |
| Решение в чистых стратегиях |
|  |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент: | Мельниченко Дмитрий Сергеевич | | |
| Направление подготовки | | 01.03.02 – Прикладная математика и информатика | |
|  | |  | |
| Группа: | ПМИб-4 | | |
|  | |  | |
| Руководитель СПбГАСУ:  Яркова Ольга Николаевна | | |  |
|  | | |  |

Санкт-Петербург

2023 г

Модели и методы анализа конфликтных ситуаций

Задание:

1. Выполнить доминирование стратегий матриц и найти решение игры (седловую точку либо интервал). (Рисунок 1)

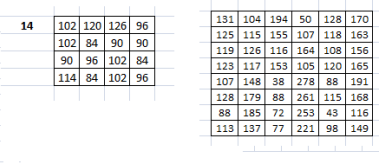


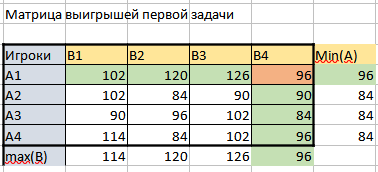
Рисунок 1 Исходные данные

1. Разработать программу для нахождения верхней и нижней цены стратегической игры, применения принципов доминирования стратегий. Входная информация: количество стратегий игрока А, количество стратегий игрока B, матрица выигрышей игрока А. Выходная информация: матрица игры после применения принципов доминирования; верхняя цена игры, нижняя цена игры, вывод о наличии/отсутствии решения в чистых стратегиях, при наличии седловой точки - решение игры и оптимальные стратегии.

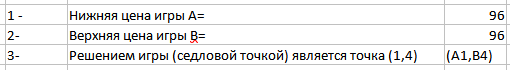
Этапы метода:

1. Провести доминирование по строкам
2. Если второй игрок – не природа, провести доминирование по столбцам
3. В оставшейся матрице найти максимальные значения по столбцам и минимальные значения по строкам (метод максимина)
4. Найти минимум по максимумам и максимум по минимумам.
5. Оценить нижнюю и верхнюю цены игры.
6. Проанализировать наличии/отсутствии решения в чистых стратегиях, наличие седловой точки и оптимальных стратегий.

**Решение первого задания:**

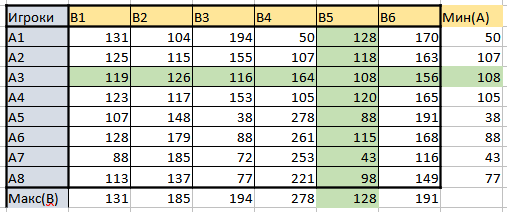


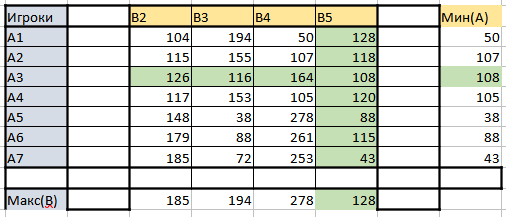


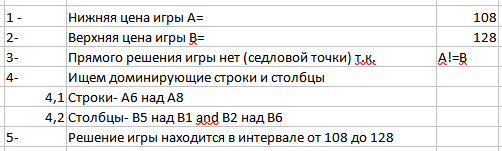


Ввиду того, что нижняя и верхняя цена игры совпадают, мы можем сделать вывод о том, что данная игра решается в чистых стратегиях и имеет седловую точку.

Решение второго задания:







Ввиду того, что игра не имеет решения в чистых стратегиях, то можно найти решение на интервале.

**Задание 2 – написание программного кода:**

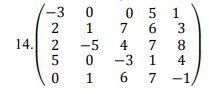


Рисунок 1 Исходные данные

Результат программы: Рисунок 2

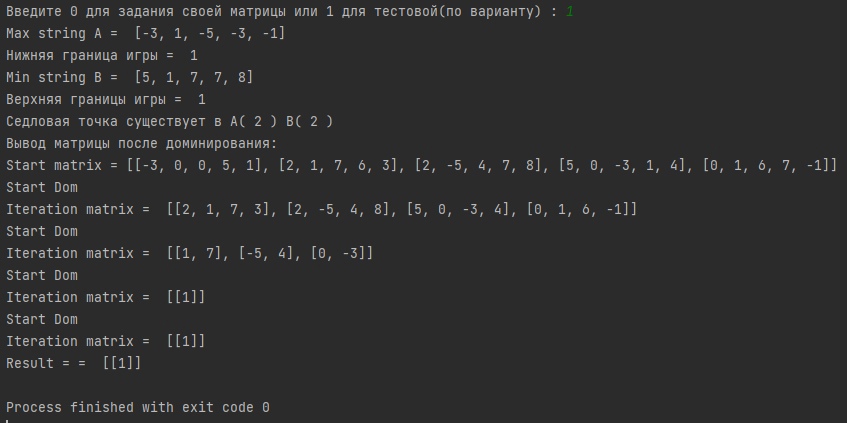


Рисунок 2 Результат работы с исходной матрицей (Рисунок 1).

**Результат работы с пользовательской матрицей:**



Рисунок 3 Исходные данные

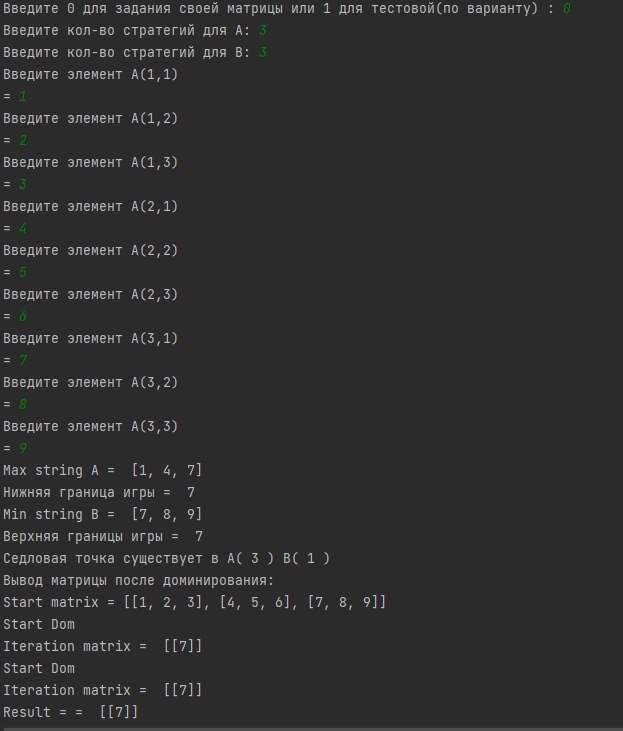


Рисунок 4 Результат работы с исходной матрицей (Рисунок 3).

**ПРИЛОЖЕНИЕ (код)**

Inp\_type:int = int(input("Введите 0 для задания своей матрицы или 1 для тестовой(по варианту) : "))  
size\_A =0  
size\_B =0  
  
if Inp\_type == 0:  
 size\_A:int = int(input("Введите кол-во стратегий для А: "))  
 size\_B:int = int(input("Введите кол-во стратегий для B: "))  
 matrix\_a =[]  
 for i in range(0,size\_A):  
 matrix\_a.append([])  
 for j in range(0,size\_B):  
 print("Введите элемент A(" + str(i+1) + "," + str(j+1) + ")")  
 matrix\_a[i].append(int(input("= ")))  
if Inp\_type == 1:  
 size\_A:int = 5  
 size\_B:int = 5  
 matrix\_a =[[-3,0,0,5,1],[2,1,7,6,3],[2,-5,4,7,8],[5,0,-3,1,4],[0,1,6,7,-1]]  
  
min\_A = []\*size\_A  
max\_B = []\*size\_B  
  
#Find down\_line  
for i in range(0,size\_A):  
 min\_val = matrix\_a[i][0]  
 for j in range(0,size\_B):  
 if matrix\_a[i][j] < min\_val:  
 min\_val = matrix\_a[i][j]  
 min\_A.append(min\_val)  
  
print("Max string A = ",min\_A)  
print("Нижняя граница игры = ",max(min\_A))  
  
# Find up\_line  
for i in range(0, size\_B):  
 max\_val = matrix\_a[0][i]  
 for j in range(0, size\_A):  
 if matrix\_a[j][i] > max\_val:  
 max\_val = matrix\_a[j][i]  
 max\_B.append(max\_val)  
  
print("Min string B = ",max\_B)  
print("Верхняя границы игры = ", min(max\_B))  
  
if min(max\_B) == max(min\_A) :  
 index\_a = 0  
 for i in range(0,len(min\_A)):  
 if max(min\_A) == min\_A[i]:  
 index\_a = i  
 break  
 index\_b = 0  
 for i in range(0, len(max\_B)):  
 if min(max\_B) == max\_B[i]:  
 index\_b = i  
 break  
 print("Седловая точка существует в A(",index\_a+1,") B(",index\_b+1,")")  
else:  
 print("Седловая точка не существует")  
  
print("Вывод матрицы после доминирования: ")  
  
a\_ok = 1  
b\_ok = 1  
new\_size\_a = size\_A  
new\_size\_b = size\_B  
  
new\_matrix = matrix\_a  
print("Start matrix =",new\_matrix)  
while a\_ok or b\_ok:  
 print("Start Dom")  
 #print("Size A =",new\_size\_a)  
 #print("Size B =",new\_size\_b)  
  
 a\_ok = 0  
 b\_ok = 0  
 index\_a\_for\_delete = set()  
  
  
 for start\_row in range(0, new\_size\_a-1):  
 for index\_a in range(start\_row+1,new\_size\_a):  
 index\_buf = index\_a  
 for index\_b in range(0,new\_size\_b):  
 if new\_matrix[index\_a][index\_b] > new\_matrix[start\_row][index\_b]:  
 index\_buf = -1  
 break  
 if index\_buf != -1:  
 a\_ok = 1  
 index\_a\_for\_delete.add(index\_buf)  
  
 for start\_row in range(new\_size\_a-1, 0,-1):  
 for index\_a in range(start\_row-1,-1,-1):  
 index\_buf = index\_a  
 for index\_b in range(0,new\_size\_b):  
 if new\_matrix[index\_a][index\_b] > new\_matrix[start\_row][index\_b]:  
 index\_buf = -1  
 break  
 if index\_buf != -1:  
 a\_ok =1  
 index\_a\_for\_delete.add(index\_buf)  
  
 new\_matrix\_buf = []\*len(index\_a\_for\_delete)  
 new\_size\_a\_buf = 0  
 for i in range(0,new\_size\_a):  
 if not (i in index\_a\_for\_delete):  
 new\_size\_a\_buf += 1  
 new\_matrix\_buf.append(new\_matrix[i])  
 new\_size\_a = new\_size\_a\_buf  
 new\_matrix = new\_matrix\_buf  
 index\_b\_for\_delete = set()  
  
 for start\_column in range(0, new\_size\_b - 1):  
 for index\_b in range(start\_column + 1, new\_size\_b):  
 index\_buf = index\_b  
 for index\_a in range(0, new\_size\_a):  
 if new\_matrix[index\_a][index\_b] < new\_matrix[index\_a][start\_column]:  
 index\_buf = -1  
 break  
 if index\_buf != -1:  
 b\_ok = 1  
 index\_b\_for\_delete.add(index\_buf)  
 for start\_column in range(new\_size\_b - 1, 0, -1):  
 for index\_b in range(start\_column - 1, -1, -1):  
 index\_buf = index\_b  
 for index\_a in range(0, new\_size\_a):  
 if new\_matrix[index\_a][index\_b] < new\_matrix[index\_a][start\_column]:  
 index\_buf = -1  
 break  
 if index\_buf != -1:  
 b\_ok = 1  
 index\_b\_for\_delete.add(index\_buf)  
  
  
 new\_matrix\_buf = []  
 new\_size\_b\_buf = 0  
 for index\_a in range(0,new\_size\_a):  
 new\_matrix\_buf.append([])  
 for index\_b in range(0, new\_size\_b):  
 if not (index\_b in index\_b\_for\_delete):  
 new\_size\_b\_buf += 1  
 new\_matrix\_buf[index\_a].append(new\_matrix[index\_a][index\_b])  
  
 new\_matrix = new\_matrix\_buf  
 new\_size\_b = int(new\_size\_b\_buf/new\_size\_a)  
 print("Iteration matrix = ",new\_matrix)  
  
  
  
print("Result = = ",new\_matrix)