МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

**Факультет информационных технологий и робототехники (ФИТР)**

**Отчёт по лабораторной работе №4**

По дисциплине: «Методы и алгоритмы принятия решений»

На тему: «РЕШЕНИЕ МАТРИЧНЫХ ИГР»

Вариант 10

**Выполнил:**  студент группы 10701118 Дубоделов А.В.

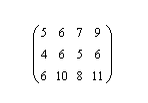
**Приняла ст. преподаватель:** Борисова И.М.

Минск 2020

**Цель работы:** изучить способы решения матричной игры в чистых стратегиях, используя принцип минимакса, а также решения игр в смешанных стратегиях с помощью ЗЛП.

**Постановка задачи:**

1.Разработать приложение для решения матричной игры в чистых стратегиях, используя принципы минимакса.



2.Решеть матричную игру в смешанных стратегиях с помощью ЗЛП в EXCEL.

****

**Приложения для решения матричной игры в чистых стратегиях.**

1. Описание приложения

Программа по решению матричной игры в чистых стратегиях реализована на языке программирования Python, и состоит из 1 модуля game.py.

На вход принимаются строки, которые конвертируются в целочисленные матрицы. Для решения задачи реализованы функции

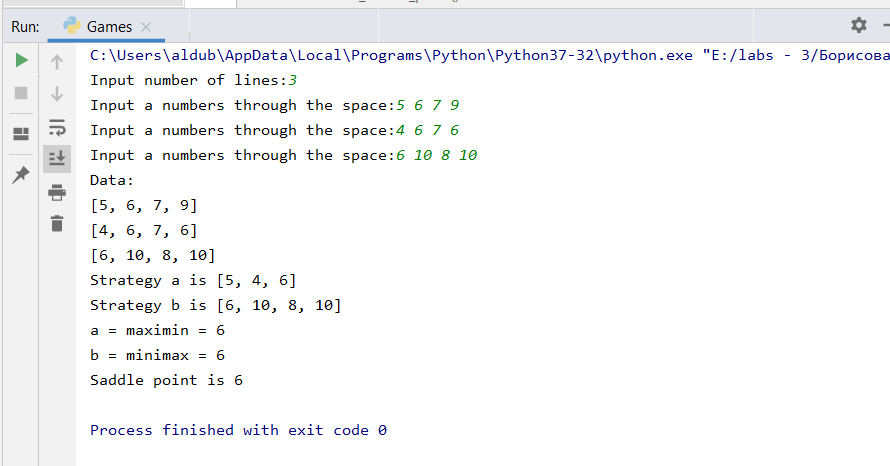
*getMinValuesInRows(matrix) –* функция нахождения минимального значения в строке

*getMaxValuesInColumns(matrix) –* функция нахождения максимального значения в колонке.

*searchSaddlePoint(matrix) –* функция поиски седловых точек.

*convert(array\_str) –* используются для перевода строковых данных в массив int

1. Результаты выполнения представлен в виде (стратегию а, стратегию b, нижнюю цену игры(а), верхнюю цену игры(b), чистую цену игры)



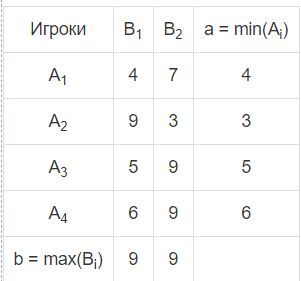
1. Листинг

Листинг программы представлен в конце отчета.

**Решить матричную игру в смешанных стратегиях с помощью ЗЛП.**

1. Проверяем, имеет ли платежная матрица седловую точку. Если да, то выписываем решение игры в чистых стратегиях.

Считаем, что игрок 1 выбирает свою стратегию так, чтобы получить максимальный свой выигрыш, а игрок 2 выбирает свою стратегию так, чтобы минимизировать выигрыш игрока 1.



1. Определяем гарантированный выигрыш, определяемый нижней ценой игры a = max(a) = 6, которая указывает на максимальную чистую стратегию А4. Верхняя цена игры b = min(b) = 9.

Что свидетельствует об отсутствии седловой точки, так как a ≠ b, тогда цена игры находится в пределах 6 ≤ y ≤ 9. Находим решение игры в смешанных стратегиях.

1. Проверяем платежную матрицу на доминирующие строки и доминирующие столбцы.

Стратегия A3 доминирует над стратегией A1 (все элементы строки 3 больше или равны значениям 1-ой строки), следовательно, исключаем 1-ую строку матрицы. Вероятность p1 = 0.

Стратегия A4 доминирует над стратегией A3 (все элементы строки 4 больше или равны значениям 3-ой строки), следовательно, исключаем 3-ую строку матрицы. Вероятность p3 = 0.



Cвели игру 4 x 2 к игре 2 x 2.

Так как игроки выбирают свои чистые стратегии случайным образом, то выигрыш игрока I будет случайной величиной. В этом случае игрок I должен выбрать свои смешанные стратегии так, чтобы получить максимальный средний выигрыш.

1. Находим решение игры в смешанных стратегиях

Запишем систему уравнений.

Для игрока I:

|  |
| --- |
| 9y1+3y2 ≤ 1  6y1+9y2 ≤ 1  Z(y) = y1+y2 → max |

Для игрока II :

|  |
| --- |
| 9x1+6x2 ≥ 1  3x1+9x2 ≥ 1  F(x) = x1+x2 → min |

1. Решаем прямую задачу линейного программирования симплекс методом.

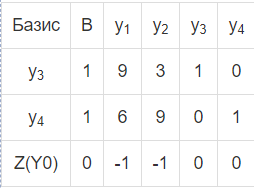
Определяем максимальное значение целевой функции:

|  |
| --- |
| Z(Y) = y1+y2 |

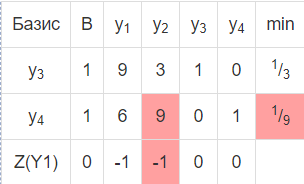
При ограничениях:

|  |
| --- |
| 9y1+3y2≤1  6y1+9y2≤1 |

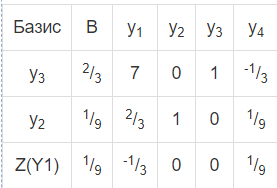
1. Для построения опорного плана систему неравенств приведем к системе уравнений путем введения дополнительных переменных и решим систему.
2. Первый опорный план



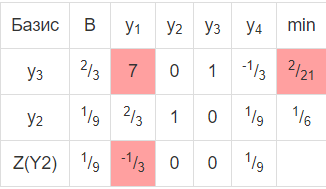
1. Так как текущий опорный план не оптимален, в качестве ведущего столбца выберем столбец y2, так как это наибольший коэффициент по модулю.



1. Получаем новую симплекс-таблицу.



1. Так как полученный план также не оптимален, повторно применяем пункт 8.



1. Формируем следующую часть симплексной таблицы. Вместо переменной y3 в план 2 войдет переменная y1.



1. Найден оптимальный план.

|  |
| --- |
| y1 = 2/21  y2 = 1/21  Z(y) = 1/7 |

1. Цена игры будет равна g= 1/Z(y), а вероятности применения стратегий:

|  |
| --- |
| qi = g\*yi; pi = g\*xi.  *Цена игры:* g = 1 : 1/7 = 7  p1 = 7\*1/21 = 1/3  p2 = 7\*2/21 = 2/3 |

1. Оптимальная смешанная стратегия игрока I:

|  |
| --- |
| P = (1/3; 2/3)  q1 = 7\*2/21 = 2/3  q2 = 7\*1/21 = 1/3 |

Оптимальная смешанная стратегия игрока II:

|  |
| --- |
| Q = (2/3; 1/3) |

Цена игры: v=7.

**Выводы:** в результате выполнения лабораторной работы разработана программа для решения матричной игры в чистых стратегиях, используя принципы минимакса, в результате выполнения которой выводится информация о стратегиях, верхней и нижней цене игры, а также чистая цена игры. А также решил матричную игру в смешанных стратегиях с помощью ЗЛП в EXCEL. Изначально убрав все заведомо ложные стратегии, сократив размер матрицы, далее воспользовавшись и найдя промежуток цены игры, вычислив верхнюю и нижнюю ее пределы. Далее, решая задачу в EXCEL, был выведен оптимальный план, найдены оптимальные стратегии для каждого игрока и цена игры.

**Листинг**

Game.py

**def** get\_min\_values\_in\_rows(matrix):  
 min\_values = []  
 **for** i **in** matrix:  
 min\_values.append(min(i))  
 **return** min\_values  
  
  
**def** get\_max\_values\_in\_columns(matrix):  
 max\_values = []  
 **for** i **in** range(0, len(matrix[0])):  
 max\_values.append(max([row[i] **for** row **in** matrix]))  
 **return** max\_values  
  
  
**def** search\_saddle\_point(matrix):  
 a\_strategy = get\_min\_values\_in\_rows(matrix)  
 b\_strategy = get\_max\_values\_in\_columns(matrix)  
 a = max(a\_strategy)  
 b = min(b\_strategy)  
 print(**"Strategy a is "** + str(a\_strategy))  
 print(**"Strategy b is "** + str(b\_strategy))  
 print(**"a = maximin = "** + str(a))  
 print(**"b = minimax = "** + str(b))  
 **if** a == b:  
 print(**"Saddle point is "** + str(a))  
 **else**:  
 print(**"There is not saddle point."**)  
  
  
**def** convert(array\_str):  
 array\_int = []  
 **for** i **in** array\_str:  
 array\_int.append(int(i))  
 **return** array\_int  
  
  
row\_count = int(input(**"Input number of lines:"**))  
*# data = [[2, 3, 6, 5, 7], [1, 2, 4, 3, 4], [5, 4, 8, 6, 9], [0, 3, 2, 5, 1]]*data = []  
**for** i **in** range(0, row\_count):  
 line = input(**"Input a numbers through the space:"**).split(**" "**)  
 data.append(convert(line))  
print(**"Data: "**)  
**for** i **in** data:  
 print(i)  
search\_saddle\_point(data)