МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

## Факультет информационных технологий и робототехники

Кафедра программного обеспечения вычислительной техники

и автоматизированных систем

**Отчет по лабораторной работе №4**

по дисциплине: “Методы и алгоритмы принятия решений ”

на тему: ***“***Решение матричных игр***”***

Вариант 5

Выполнил**:** студент группы: 10701118 Березневич И. В.

Приняла**:** ст.пр. Борисова И.М

Минск 2020

**Задание**

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

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

**Задание 1:** Решение матричной игры в смешанных стратегиях с помощью ЗЛП в Excel:

**Математическая модель:**

f(x) = x1 + x2 + x3 ->min

Ограничения:

4\*x1 + 2\*x2 + 3\*x3 >= 1

2\*x1 + 5\*x2 + 0\*x3 >= 1

0\*x1 + 2\*x2 + 5\*x3 >= 1

f(y) = y1 + y2 + y3 -> max

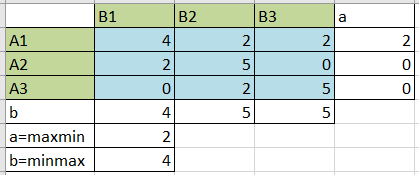
Ограничения:

4\*y1 + 2\*y2 + 0\*y3 <= 1

2\*y1 + 5\*y2 + 0\*y3 <= 1

0\*y1 + 2\*y2 + 5\*y3 <= 1

В начале проверяем является ли игра с чистой стратегией. Для этого находим **нижнюю и верхнюю чистые цены игры**. **Нижняя цена** –максимально возможный выигрыш первого игрока, а **верхняя цена** – минимальный проигрыш второго:

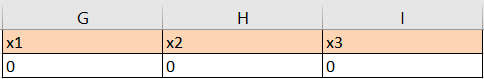


Данная игра **не имеет чистой стратегии**, так как и нет **седловой точки**.

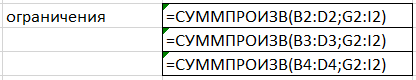
Приступаем к нахождению решения игры в **смешанных стратегиях.**

Сначала, необходимо ввести переменные **x1,** **х2, x3,** где целевая функция для первого игрока будет равна  **сумме (х1,**  **х2, x3):**

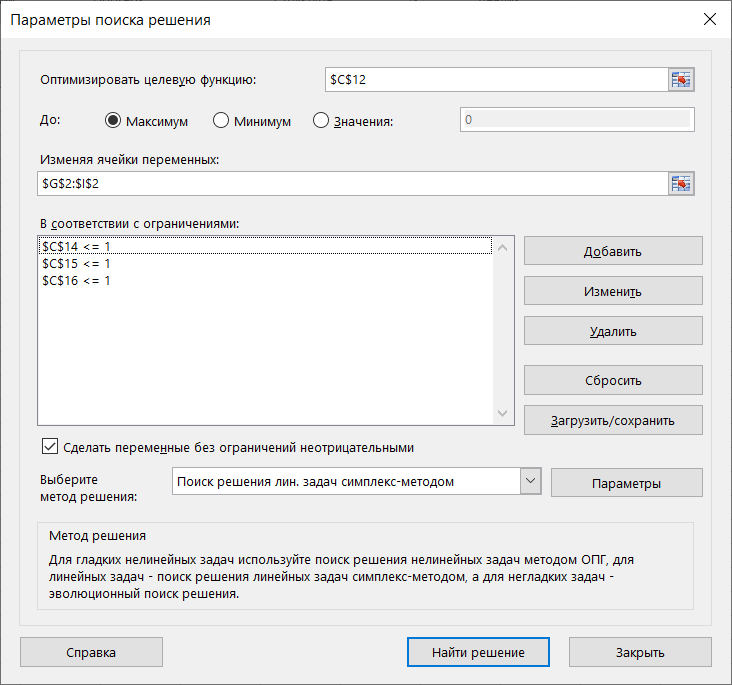


****

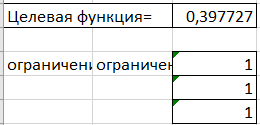
Задаём ограничения для переменных **х1**, **х2, x3,** чтобы найти их значения и в дальнейшем найти целевую функцию для первого игрока:



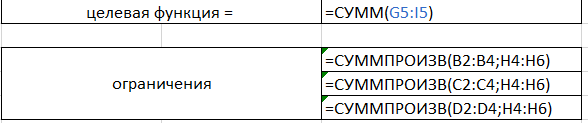
Применяем **Поиск решений** и задаем необходимые данные для нахождения целевой функции и переменных **x1, x2, х3**:



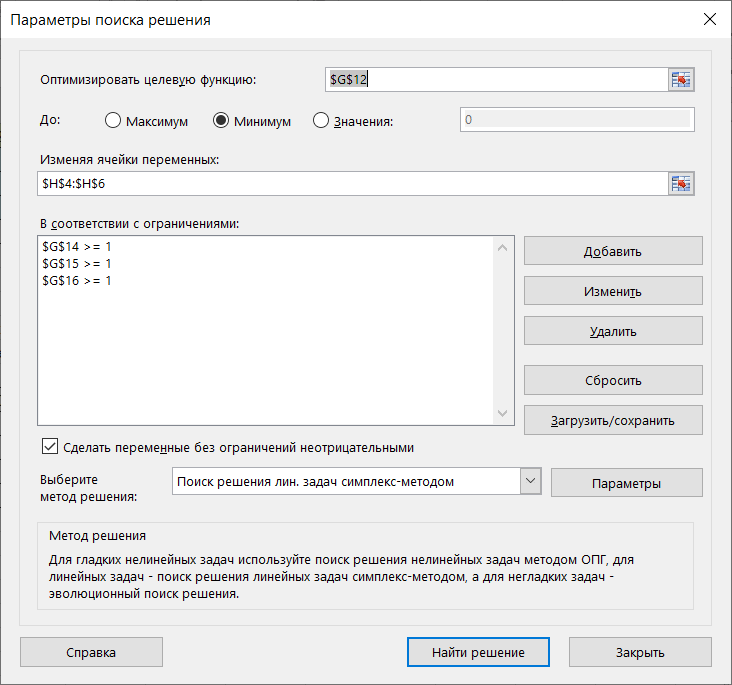
Получаем:

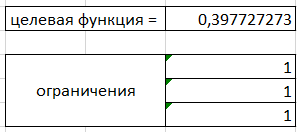
Те же самые действия проделываем и для **y1,** **y2, y3**, чтобы найти целевую функцию для второго игрока:

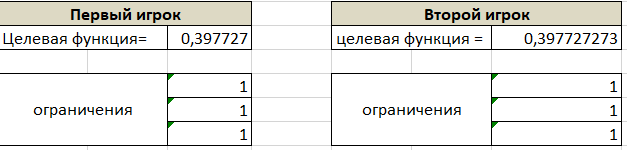


Используем **Поиск решений**:



Получаем целевую функцию и переменные:



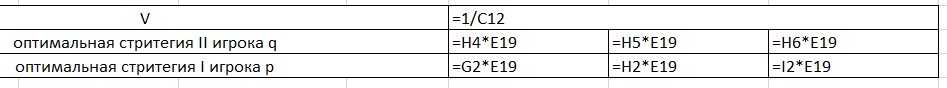


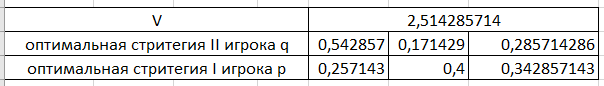
Так как целевые функции двух игроков совпадают, значит **цена игры** для двух игроков **– одинакова.**

Для нахождения **цены игры**, необходимо **единицу поделить на значение целевой функции.**



Определяем вероятность использования игроками той или иной стратегии.  
За **p** обозначим выбор стратегии первого игрока, а за **q** – второго игрока.





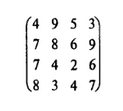
**Ответ:** данная платёжная матрица не имеет седловой точки т.к., при а=2, b=4; Отсутствие седловой точки позволяет использовать смешанную стратегию для поиска решения игры. С помощью Поиска решений, была найдена цена игры (V = 2,5) и с помощью указанных выше формул, находим оптимальные стратегии игроков, которые для исходной матрицы.с помощью указанных выше формул, находим оптимальные стратегии игроков, которые для исходной матрицы выглядят так:

**Для первого игрока:**

**Для второго игрока:**

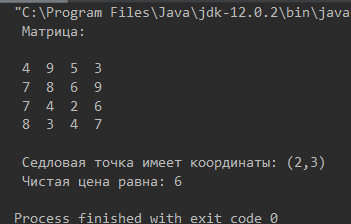
**Задание 2**

Разработать приложение, реализующее принцип минимакса. Для платежных матриц, представленных ниже, определить наличие седловых точек. При наличии cедловых точек определить чистую цену игры и оптимальное решение.



Данная программа находит minimax и maxmin и их положение в матрице, после чего сравнивает между собой. Если они равны, то выводятся координаты седловой точки и сама седловая точка.

Результат выполнения программы:



**Ответ:** Найденная седловая точка означает максимальный выигрыш первого игрока и минимальный проигрыш второго, а положение этого числа в платежной матрице – задействованные в ней стратегии первого и второго игроков.

ЛИСТИНГ:

**import** java.util.Scanner;  
  
**public class** Lab {

**public static void** main(String[] args) {  
 System.***out***.println(**" Матрица:\n"**);  
 **int** table[][] = {{4, 9, 5 ,3}, {7, 8, 6, 9}, {7, 4, 2 ,6}, {8, 3, 4, 7}};  
 **int** n = 4;  
 **int** minA[] = {0, 0}, minB[] = {0, 0};  
 minA = *searchA*(table, n);  
 minB = *searchB*(table, n);  
 **for**(**int** i = 0; i < n; i++) {  
 **for**(**int** j = 0; j < n; j++) {  
 System.***out***.print(**" "** + table[i][j] + **" "** );  
 }  
 System.***out***.println(**""**);  
 }  
  
 System.***out***.println(**"\n Седловая точка имеет координаты: ("** + (minA[1]+1) + **","** +  
 (minB[1]+1) + **") \n Чистая цена равна: "** + minA[0] );  
 }  
  
 **public static int**[] searchA(**int**[][] array, **int** n) {  
 **int** min = array[0][0];  
 **int**[] arr= {0, 0, 0, 0};  
 **int**[] minCord = {0 ,0};  
  
 **for**(**int** i = 0; i < n; i++) {  
 **for**(**int** j = 0; j < n; j++) {  
 **if**(array[i][j] < min && j + 1 != 4) {  
 min = array[i][j];  
 arr[i] = min;  
 }  
 **if**(j + 1 == 4 && i + 1 != 4) {  
 arr[i] = min;  
 min = array[i+1][0];  
 }  
  
 }  
 }  
  
 **for**(**int** i = 0; i < n; i++) {  
 **if**(arr[i] > min) {  
 min = arr[i];  
 *//значение* minCord[0] = min;  
 *//номер строки* minCord[1] = i;  
 }  
 }  
  
 **return** minCord;  
 }  
  
 **public static int**[] searchB(**int**[][] array, **int** n) {  
 **int** max = array[0][0];  
 **int**[] arr= {0, 0, 0, 0};  
 **int**[] minCord = {0 ,0};  
  
 **for**(**int** j = 0; j < n; j++) {  
 **for**(**int** i = 0; i < n; i++){  
 **if**(array[i][j] > max && j+1 != 5) {  
 max = array[i][j];  
 arr[j] = max;  
 }  
 **if**(i + 1 == 4 && j + 1 != 4) {  
 arr[j] = max;  
 max = array[0][j+1];  
 }  
 **else if** (i + 1 == 4 && j + 1 == 4)  
 arr[j] = max;  
  
 }  
 }  
  
 **for**(**int** i = 0; i < n; i++) {  
 **if**(arr[i] < max) {  
 max = arr[i];  
 minCord[0] = max;  
 minCord[1] = i;  
 }  
  
 }  
  
 **return** minCord;  
 }  
}