|  |  |
| --- | --- |
|  | **Министерство образования и науки Российской Федерации**  Калужский филиал федерального государственного бюджетного образовательного учреждения высшего образования  ***«Московский государственный технический университет  имени Н.Э. Баумана***  ***(национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

|  |  |
| --- | --- |
| **ФАКУЛЬТЕТ** | "Электроника, информатика и управление" |
| **КАФЕДРА** | "Информационная безопасность автоматизированных систем" |

##### 

**О Т Ч Е Т**

ЛАБОРАТОРНАЯ РАБОТА №2

|  |  |  |
| --- | --- | --- |
| **ДИСЦИПЛИНА:** | | " Теория принятия решений в условиях информационных конфликтов " |
| **ТЕМА:** | " Матричные игры n x n" | |

|  |  |
| --- | --- |
| Выполнил: студент гр. БАС.И-71 | Демичев А.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Проверил: | Макарова О.C.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

|  |  |  |
| --- | --- | --- |
| Дата сдачи (защиты) лабораторной работы: | |  |
| Результаты сдачи (защиты):  Количество рейтинговых баллов |  | |
| Оценка |  | |

Калуга, 2017 г.

**Цель работы:** получить практические навыки решения игровых задач размерности .

**Задание:**

Напишите программу, которая по заданной платёжной матрице выводит на экран нижнюю цену игры, верхнюю цену и значение седловой точки при ее наличии. В случае отсутствия седловой точки в матрице, программа находит оптимальные смешанные стратегии игроков, используя метод Крамера.

Элементы матрицы задаются пользователем с клавиатуры или случайным образом.

**Теоретическая часть:**

Пусть платёжная матрица выглядит следующим образом:



Требуется определить оптимальные стратегии игроков ,  и цену игры .

Замечание: Поскольку данный метод опирается на теорему об активных стратегиях, перед началом решения задачи необходимо убедиться в том, что все стратегии являются активными, т.е. отсутствуют седловая точка и заведомо невыгодные стратегии.

Определим оптимальную стратегию  игрока А в предположении, что А применяет свою оптимальную смешанную стратегию, а В – свои чистые стратегии:



 - условие нормировки.

Далее для нахождения  применяем метод Крамера:



Применив соотношение нормировки, получим:



Подставив выражение для цены игры в формулы для расчета , получим: 

Аналогично определяем оптимальную стратегию игрока В, считая, что В применяет оптимальную смешанную стратегию, а А – свои чистые стратегии: 

 - условие нормировки.

В результате использования метода Крамера для решения данной системы уравнений получаем вероятности применения стратегий игрока В: 

**Практическая часть:**

Код программы:

using System;

using System.Linq;

using System.Windows.Forms;

namespace LW1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

int[,] matrix;

private void button1\_Click(object sender, EventArgs e)

{

int row, column;

if (!int.TryParse(textBox1.Text, out row))

{

MessageBox.Show("Error! Invalid input!");

}

else

{

column = row;

dataGridView1.RowCount = row;

dataGridView1.ColumnCount = column;

matrix = new int[column, row];

for (int i = 0; i < dataGridView1.ColumnCount; i++)

{

dataGridView1.Columns[i].Width = 20;

}

Random random = new Random();

for (int i = 0; i < row; i++)

{

for (int j = 0; j < column; j++)

{

matrix[j, i] = random.Next(20);

dataGridView1[j, i].Value = matrix[j, i];

}

}

}

}

private void button2\_Click(object sender, EventArgs e)

{

int[] alphaMas, betaMas;

int alpha, beta;

bool IsSaddlePoint;

double[] a, x, b, y;

double da, db, aSum, bSum, va, vb;

double[,] tempMatrix;

int n = dataGridView1.RowCount;

string Enter = "";

for (int i = 0; i < n; i++) //перезапись матрицы

{

for (int j = 0; j < n; j++)

{

matrix[j, i] =Convert.ToInt32(dataGridView1[j, i].Value);

}

}

alphaMas = new int[n];

betaMas = new int[n];

for (int i = 0; i < n; i++) //нахождение альфы

{

int[] temp = new int[n];

for (int j = 0; j < n; j++)

{

temp[j] = matrix[j, i];

}

alphaMas[i] = temp.Min();

}

alpha = alphaMas.Max();

for (int i = 0; i < n; i++)//нахождение бэты

{

int[] temp = new int[n];

for (int j = 0; j < n; j++)

{

temp[j] = matrix[i, j];

}

betaMas[i] = temp.Max();

}

beta = betaMas.Min();

if (alpha == beta) { IsSaddlePoint = true; Enter += "IsSaddlePoint = true\r\n"; } //проверка седловой точки

else { IsSaddlePoint = false; Enter += "IsSaddlePoint = false\r\n"; }

tempMatrix = new double[n, n];

if (!IsSaddlePoint) //если седловой точки нет

{

//для игрока А

a = new double[n]; //массив определителей а(n)

aSum = 0;

for (int i = 0; i < n; i++) //фиксируем столбец

{

for (int j = 0; j < n; j++) //столбцы

{

for (int k = 0; k < n; k++) //строки

{

if (j == i) tempMatrix[j, k] = 1;

else tempMatrix[j, k] = (double)matrix[k, j]; //записываем наоборот

}

}

a[i] = Determinant(tempMatrix);

aSum += a[i];

}

for (int j = 0; j < n; j++)

{

for (int k = 0; k < n; k++)

{

tempMatrix[k, j] = (double)matrix[k, j];

}

}

da = Determinant(tempMatrix);

va = da / aSum;

x = new double[n]; //нахождение х

for (int i = 0; i < n; i++)

{

x[i] = a[i] / aSum;

}

b = new double[n]; //массив определителей b(n)

bSum = 0;

for (int i = 0; i < n; i++) //фиксируем столбец

{

for (int j = 0; j < n; j++)

{

for (int k = 0; k < n; k++)

{

if (k == i) tempMatrix[k, j] = 1;

else tempMatrix[k, j] = (double)matrix[k, j];

}

}

b[i] = Determinant(tempMatrix);

bSum += b[i];

}

for (int j = 0; j < n; j++)

{

for (int k = 0; k < n; k++)

{

tempMatrix[k,j] = (double)matrix[k,j];

}

}

db = Determinant(tempMatrix);

vb = db / bSum;

y = new double[n]; //нахождение y

for (int i = 0; i < n; i++){

y[i] = b[i] / bSum;

}

//

Enter += ("A: \t\tB:\r\n" + "vA = " + va.ToString("0.00000") + "\tvB = " + vb.ToString("0.00000") + "\r\n");

for (int i = 0; i < n; i++)

{

Enter += (x[i].ToString("0.00000") + "; \t\t" + y[i].ToString("0.00000") + "; \r\n");

}

}

MessageBox.Show("alpha = " + alpha.ToString() + "\t\tbeta = " + beta.ToString() + "\r\n" + Enter);

}

public static double[,] GetMinor(double[,] matrix, int row, int column)

{

if (matrix.GetLength(0) != matrix.GetLength(1)) throw new Exception(" Число строк в матрице не совпадает с числом столбцов");

double[,] buf = new double[matrix.GetLength(0) - 1, matrix.GetLength(0) - 1];

for (int i = 0; i < matrix.GetLength(0); i++)

for (int j = 0; j < matrix.GetLength(1); j++)

{

if ((i != row) || (j != column))

{

if (i > row && j < column) buf[i - 1, j] = matrix[i, j];

if (i < row && j > column) buf[i, j - 1] = matrix[i, j];

if (i > row && j > column) buf[i - 1, j - 1] = matrix[i, j];

if (i < row && j < column) buf[i, j] = matrix[i, j];

}

}

return buf;

}

public static double Determinant(double[,] matrix)

{

if (matrix.GetLength(0) != matrix.GetLength(1)) throw new Exception(" Число строк в матрице не совпадает с числом столбцов");

double det = 0;

int Rank = matrix.GetLength(0);

if (Rank == 1) det = matrix[0, 0];

if (Rank == 2) det = matrix[0, 0] \* matrix[1, 1] - matrix[0, 1] \* matrix[1, 0];

if (Rank > 2)

{

for (int j = 0; j < matrix.GetLength(1); j++)

{

det += Math.Pow(-1, 0 + j) \* matrix[0, j] \* Determinant(GetMinor(matrix, 0, j));

}

}

return det;

}

}

}

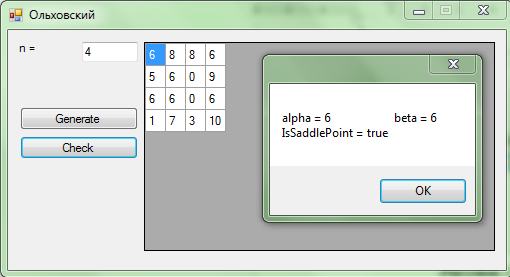


Рисунок 1 - Работа программы – седловая точка существует

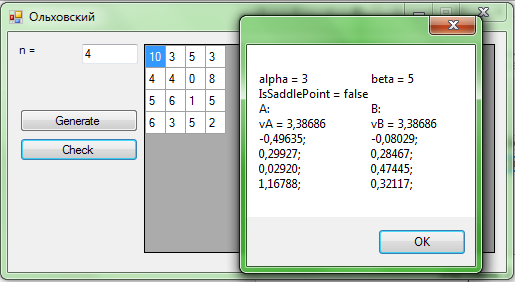


Рисунок 2 - Работа программы – седловой точки нет

**Вывод:** сформированы практические навыки решения игровых задач размерности n x n.