МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий

Кафедра «Информационные системы»

**ОТЧЕТ**

о выполнении лабораторной работы №2

по дисциплине “Теория принятия решений”

Выполнил: ст. гр. ИС/б-20-2-о

Белик Г. М.

Проверил: доц. каф.

«Информационные системы»

Кротов К. В.

Севастополь

2022

**ЛАБОРАТОРНАЯ РАБОТА №2**

**«ИССЛЕДОВАНИЕ ПРИМЕНЕНИЯ АППАРАТА ТЕОРИИ ОДНОМЕРНОЙ ПОЛЕЗНОСТИ ДЛЯ РЕШЕНИЯ ЗАДАЧ ВЫБОРА АЛЬТЕРНАТИВ»**

**Цель работы**

Исследовать применение аппарата теории полезности при принятии решений по выбору альтернатив.

**Задачи**

Вариант 1. Используя метод, реализующий формирование множеств и , а также их последующий анализ (с точки зрения = ∅ ( ≠ ∅), = ∅ ( ≠ ∅), ∩ = ∅, ∩ ≠ ∅) , выполнить для заданного вида матрицы отношения предпочтения определение значений функции полезности U() решений и определение по формируемым значениям функции полезности эффективных решений ∈ X. Матрица отношения предпочтения имеет следующий вид:

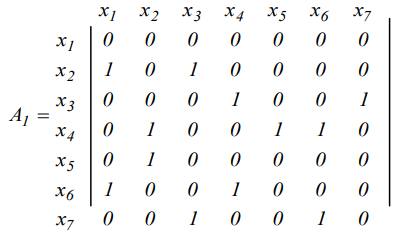
****

Рисунок 1 – Матрица отношения

Для варианта задания, связанного с использованием множеств и , предусматривается следующий порядок действий по выполнению лабораторной работы:

1. реализовать объявление и инициализацию матрицы отношений между решениями в соответствии с вариантом задания;
2. реализовать процедуру определения для каждого рассматриваемого решения соответствующих ему множеств и , которые определяют для решения не худшие по отношению к нему решения (множество ) и не лучшие по отношению к нему решения (множество ); при определении множества необходимо выполнять просмотр (n+1)-го столбца матрицы отношений, при определении множества необходимо выполнять просмотр (n+1)-ой строки матрицы отношений, для рассматриваемого элемента выполнить вывод множеств , ;
3. реализовать процедуру выполнения условий = ∅ ( ≠ ∅ ), = ∅ ( ≠ ∅), ∩ = ∅, ∩ ≠ ∅; тем самым определяется способ вычисления значений функции полезности для решения ; реализовать вывод информации о выполняющемся условии;
4. реализовать процедуру вычисления значения функции полезности для текущего рассматриваемого решения ;
5. реализовать процедуру управления процессом вычисления значений функции полезности для каждого элемента множества Х (решения множества Х); реализовать в рассматриваемой процедуре определение максимального значения функции полезности и соответствующего ему решения; выполнить вывод всех решений ∈ X и соответствующих им значений функции полезности.

**Ход работы**

Для начала по заданному варианту графа отношений предпочтения между решениями была сформирована матрица А отношения R.

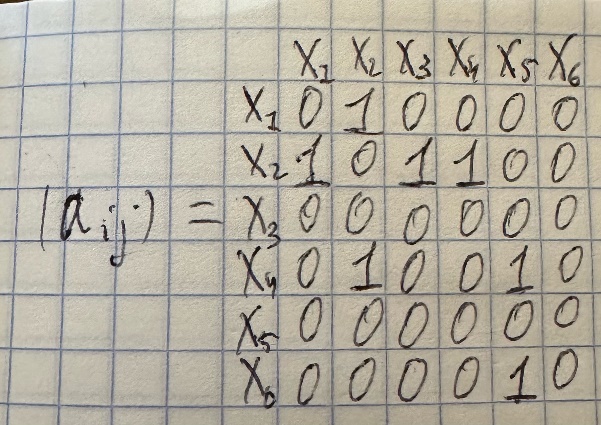
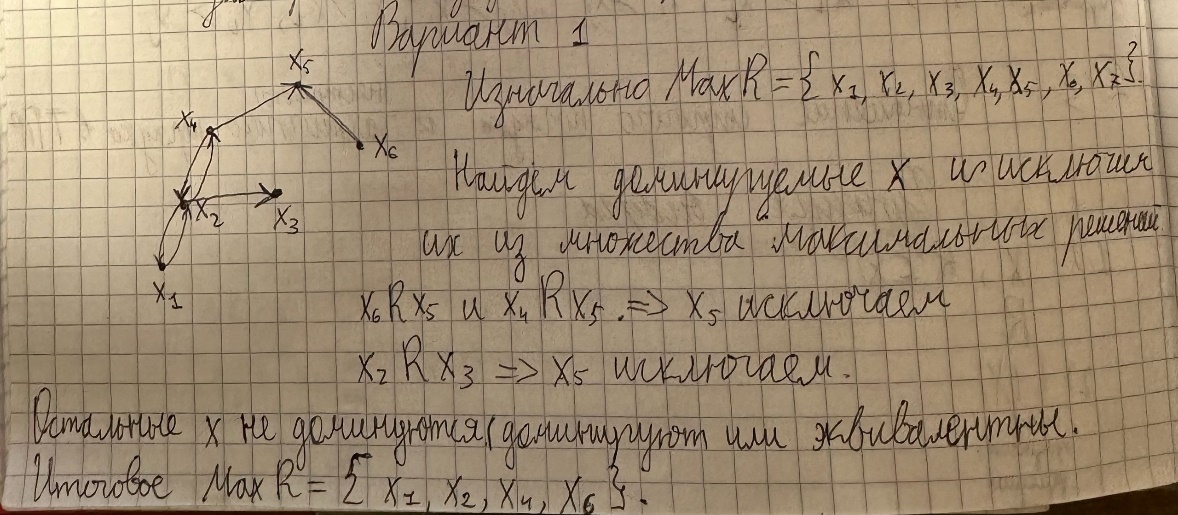


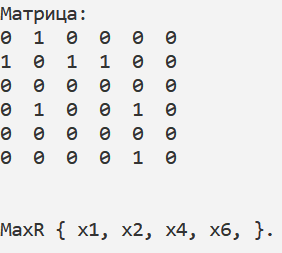
Рисунок 2 – Матрица отношений

Затем вручную было выполнено формирование множества MaxR для заданного вида графа и соответствующего ему вида матрицы А.

 Рисунок 3 – Расчеты множества MaxR

Следующим шагом был сформирован программный код на языке C++ соответствующей процедуры определения множества MaxR, представленный в приложении А.

После этого выведем результат работы процедуры.

 Рисунок 4 – Матрица отношений и найденное ее максимальное множество

Результаты, полученные в процедуре аналогичны сформированным аналитически.

Изменим исходные данные программы и увидим, что для них программа также находит верные значения максимального множества.

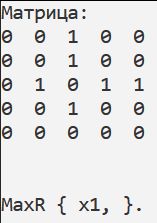


Рисунок 5 – Матрица отношений и найденное ее максимальное множество

**Выводы**

В ходе выполнения лабораторной работы был исследован аппарат бинарных отношений при принятии решений по выбору альтернатив. По заданному графу отношений предпочтения была сформирована матрица отношений R, аналитически нашли множество максимальных элементов MaxR, был написан программный код для нахождения MaxR, сопоставлены полученные значения, и произвели повторную проверку правильности работы программы на другом графе бинарных отношений.

**ПРИЛОЖЕНИЕ А**

#include <iostream>

#include <locale>

using namespace std;

int main()

{

setlocale(LC\_ALL, "");

//Заполнение матрицы

int n;

cout << " Введите размерность матрицы:\n";

cin >> n;

int\*\* a = new int\* [n];

cout << " Заполните масив из "<<n\*n<<" элементов:\n";

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

{

a[i] = new int[n];

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

{

cout << "massiv[" << i + 1 << "][" << j + 1 << "] = ";

cin >> a[i][j];

}

}

// Печать матрицы

cout << endl << "Матрица:" << endl;

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

{

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

cout << a[i][j] << " ";

cout << endl;

}

cout << endl;

// Расчеты

int\* MaxR = new int[n];

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

MaxR[i] = 1;

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

{

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

{

if (a[i][j] == 1)

{

if (a[j][ i] == 0)

MaxR[j] = 0;

if ((a[j][i] == 1) && MaxR[i] == 0)

MaxR[j] = 0;

}

}

}

cout << endl;

cout << "MaxR { ";

// Вывод ответа

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

{

if (MaxR[i] == 1)

cout << "x" <<i+1<<", ";

}

cout << "}."<<endl;

}