МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Севастопольский государственный университет»

Институт информационных технологий и управления в технических системах

кафедра Информационные системы

09.03.02 Информационные системы и технологии (уровень бакалавриата)

РАСЧЕТНО–ГРАФИЧЕСКОЕ ЗАДАНИЕ

по дисциплине: «Теория принятия решений»

Вариант – 2

Выполнил

студент 3 курса группы ИС/б-33-о

Генералов Николай Николаевич

Отметка о зачете\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

(дата)

Проверил

доц. \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кротов К. В.

(должность) (подпись) (фамилия, инициалы)

\_

г. Севастополь

2018 г.

# ЦЕЛЬ РАБОТЫ

Исследовать применение аппарата бинарных отношений при принятии решений по выбору альтернатив.

# ПОСТАНОВКА ЗАДАЧИ

* Выполнить разработку программы, реализующей определение упорядоченного множества решений для множества Х, руководствуясь формами графов отношений №1 и 2, заданных на рисунках 1, 2.
* При разработке программы использовать приведенные в теоретическом введении правила формирования множества с учетом рассмотрения вершин-источников на каждом шаге алгоритма.
* При формировании упорядоченного множества решений указывать номер яруса, на котором находятся решения.
* Определить эффективные решения.

x4

x3

x5

x1

x2

x6

x7

Рисунок 1 – Граф отношений №1

x4

x3

x5

x1

x2

Рисунок 2 – Граф отношений №2

# ХОД РАБОТЫ

* 1. Построение матриц отношений

Для заданных по заданию графов рис. 1 и 2, построим матрицы инцидентности:

Рисунок 3 – Матрица отношений, соответствующая модели графа №1.

Рисунок 4 – Матрица отношений, соответствующая модели графа №2.

* 1. Аналитические вычисления

Для матрицы рис. 3 было определено множество эффективных решений MaxR и выполнено его упорядочивание по ярусам:

Таблица 1. Распределение решений графа №1.

|  |  |
| --- | --- |
| Номер яруса | Эффективное решение |
| 1 | x1, x5 |
| 2 | x4 |
| 3 | x2 |
| 4 | x3 |
| 5 | x6 |
| 6 | x7 |

Аналогичные действия были выполнены для матрицы на рис. 4:

Таблица 2. Распределение эффективных решений графа №2.

|  |  |
| --- | --- |
| Номер яруса | Эффективное решение |
| 1 | x1 |
| 2 | x2 |
| 3 | x3 |
| 4 | x4, x5 |

# Текст программы

#include <iostream>

#include <fstream>

#include <vector>

using namespace std;

void loadMatrixFromFile(string fileName, vector<vector<bool>> &matrix);

void printBoolMatrix(vector<vector<bool>> boolMatrix);

void printResultMatrix(const vector<vector<int>> &matr);

void searchRootsOrder(const vector<vector<bool>> &mainMatrix, vector<vector<int>> &resultMatrix);

int main(int argc, char \*argv[]) {

vector<vector<bool>> matr;

loadMatrixFromFile("1.txt", matr);

cout << "Source matrix: " << endl;

printBoolMatrix(matr);

cout << endl;

vector<vector<int>> recSolutions;

searchRootsOrder(matr, recSolutions);

printResultMatrix(recSolutions);

return 0;

}

void searchRootsOrder(const vector<vector<bool>> &mainMatrix, vector<vector<int>> &resultMatrix) {

int rootAmount = mainMatrix.back().size();

vector<bool> MaxR(rootAmount, 0);

int solutionsAmount = rootAmount;

while (solutionsAmount > 0) {

resultMatrix.push\_back(vector<int>());//create new stage

for (int j = 0; j < rootAmount; ++j) {

if (!MaxR[j]) {

bool colIsIndependence = true;

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

if ((mainMatrix[i][j] != 0) && (!MaxR[i])) {

colIsIndependence = false;

break;

}

}

if (colIsIndependence) {

resultMatrix.back().push\_back(j);//added recommended root

}

}

}

int solutionsAmount = resultMatrix.back().size();

if (!solutionsAmount) {

resultMatrix.pop\_back();

return;

}

for (int a = 0; a < solutionsAmount; ++a) {

MaxR[resultMatrix.back()[a]] = true;

}

solutionsAmount -= solutionsAmount;

};

}

void printResultMatrix(const vector<vector<int>> &matr) {

if (matr.back().size() == 0) {

cout << "Source roots are not exists";

return;

}

for (int i = 0; i < matr.size(); ++i) {

cout << "Stage #" << i << endl;

for (int j = 0; j < matr[i].size(); ++j) {

cout << "x" << matr[i][j] + 1 << " ";

}

cout << endl;

}

}

void loadMatrixFromFile(string fileName, vector<vector<bool>> &matrix) {

ifstream fileStream = ifstream(fileName, ios::in);

if (fileStream) {

int rootAmount = 0;

fileStream >> rootAmount;

bool tempValue = false;

for (int j = 0; j < rootAmount; ++j) {

matrix.push\_back(vector<bool>());

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

fileStream >> tempValue;

matrix.back().push\_back(tempValue);

}

}

fileStream.close();

}

else {

exit(1);

}

}

void printBoolMatrix(vector<vector<bool>> boolMatrix) {

for (int i = 0; i < boolMatrix.size(); ++i) {

for (int j = 0; j < boolMatrix[i].size(); ++j) {

cout << boolMatrix[i][j] + 1 << " ";

}

cout << endl;

}

}

* 1. Результаты выполнения программы

В качестве исходных данных для программы, выступает матрица инцидентности графа.

При выполнении программы для матрицы, модели графа №1, был получен результат отображенный на рис. 5.

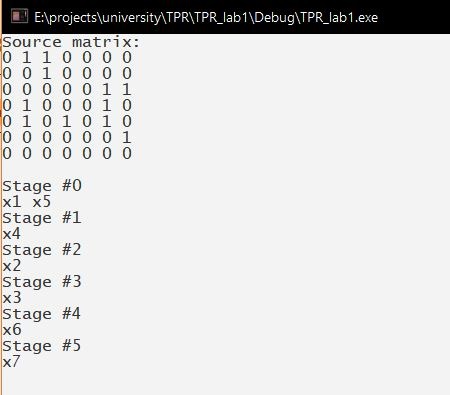


Рисунок 5 – Результат выполнения программы для графа №1

При выполнении программы для матрицы, модели графа №1, был получен результат отображенный на рис. 6.

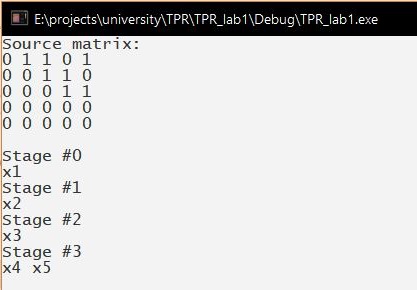


Рисунок 6 – Результат выполнения программы для графа №2

# ВЫВОД

В ходе выполнения лабораторной работы было выполнено решение задачи, связанной с построением упорядоченного множества решений, формируемого на основе множества X и отношений между его элементами, представленными в виде графа. На основании двух моделей графов, были построены их матрицы инцидентности. Аналитическим и программным путем был определен порядок расположения решений по ярусам, для этого соответственно была разработана программа на языке С++.

По результатам выполнения программы видно, что они совпали с аналитическими расчетами и можно сделать вывод, что поставленная задача выполнена.