Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №3**

По дисциплине «Математические основы интеллектуальных систем»

Тема: «Нахождение кратчайшего пути в графе»

**Выполнил:**

Студент 2 курса

Группы ИИ-21

Кабак Д. Н.

**Проверил:**

Козинский А. А.

Брест 2022

**Цель:** научиться с помощью алгоритма Дейкстры и Флойда-Уоршолла находить кратчайшие пути в графе.

**Ход работы:**

**Вариант 3**

#include <iostream>

#define inf 1000000000

using namespace std;

int indexOfMin(int \*list, int n){

int min = inf;

int index = -1;

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

if(list[i] < min){

min = list[i];

index = i;

}

}

return index;

}

string toString(int num){

if(num == 0) return "0";

string str = "";

while(num > 0){

str = (char)(num % 10 + '0') + str;

num /= 10;

}

return str;

}

void Dejkstra(int (\*graph)[7], int vertex){

int list[7], distances[7]; string visited[7];

bool STOP[7];

for (int i = 0; i < 7; i++){

STOP[i] = false; // Флаг остановки

list[i] = inf; // текущие расстояния от вершины до всех остальных

distances[i] = 0; // растояния с начальной вершнины до всех остальных

visited[i] = "";

}

for (int x = 0; x < 6; x++){

STOP[vertex] = true;

for(int i = 0; i < 7; i++){

if (graph[vertex][i] + distances[vertex] < list[i] && !STOP[i]){

if (list[i]!= inf)

visited[i] += "\b";

list[i] = graph[vertex][i] + distances[vertex];

visited[i] += toString(vertex + 1);

}

}

list[vertex] = inf;

vertex = indexOfMin(list, 7);

distances[vertex] = list[vertex];

}

for (int i = 0; i < 7; i++){

cout << i+1 << ": " << distances[i] << "\t";

}

cout << endl;

}

void Floyd(int (\*graph)[7], int vertex){

int distances[7][7];

for (int i = 0; i < 7; i++){

for (int j = 0; j < 7; j++){

distances[i][j] = graph[i][j];

}

}

for (int k = 0; k < 7; k++){

for (int i = 0; i < 7; i++){

for (int j = 0; j < 7; j++){

if (distances[i][k] + distances[k][j] < distances[i][j]){

distances[i][j] = distances[i][k] + distances[k][j];

}

}

}

}

for (int i = 0; i < 7; i++){

cout << i+1 << ": " << distances[i][vertex] << "\t";

}

cout << endl;

}

int main(){

const int tops = 7, edges = 10;

int adjacencyMatrix[tops][tops] = {{0, 5, inf, 6, 8, inf, inf},

{5, 0, 6, 3, inf, inf, inf},

{inf, 6, 0, 6, inf, inf, inf},

{6, 3, 6, 0, 4, 2, inf},

{8, inf, inf, 4, 0, inf, 5},

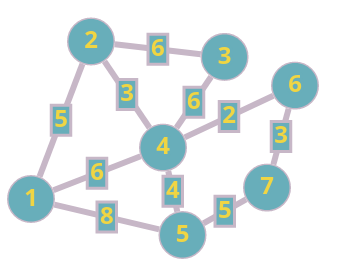
{inf, inf, inf, 2, inf, 0, 3},

{inf, inf, inf, inf, 5, 3, inf}};

Dejkstra(adjacencyMatrix, 0);

Floyd(adjacencyMatrix, 0);

}



1: 0 2: 5 3: 11 4: 6 5: 8 6: 8 7: 11

1: 0 2: 5 3: 11 4: 6 5: 8 6: 8 7: 11

**Вывод:** в ходе лабораторной работы я научился находить кратчайшие пути в графе.