**МИНОБРНАУКИ РОССИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

дисциплина: Дискретная математика

Лабораторная работа №4.5

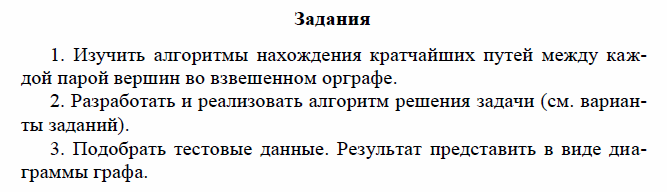
тема: «Кратчайшие пути между каждой парой вершин во взвешенном орграфе»

|  |  |
| --- | --- |
|  | Выполнил: ст. группы ПВ-21  Проверил: Рязанов Ю.Д. |

Белгород

2019

**Цель занятия**: изучить алгоритмы нахождения кратчайших путей между каждой парой вершин во взвешенном орграфе, научиться использовать их при решении различных задач.



**Вариант 3**

Во взвешенном орграфе найти все пары вершин vi и vj, такие, что кратчайшее расстояние от vi до vj равно кратчайшему расстоянию от vj до vi. Вывести кратчайшие пути между найденными парами вершин.



**Код программы на языке С:**

main.c

/\*Во взвешенном орграфе найти все пары вершин vi и vj,

такие, что кратчайшее расстояние от vi до vj равно кратчайшему расстоянию от vj до vi.

Вывести кратчайшие пути между найденными парами вершин.\*/

#include <stdio.h>

#include <stdlib.h>

#define INF 100

#define SIZE 6

typedef struct el {

//длина дуги ij

int d;

//дуга:'i'-есть, '-1'-(i=j), '-2'-нет

int t;

}element;

void print(element \*\*M) {

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

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

printf("{d = %d, t = %d} ", M[i][j].d, M[i][j].t);

}

printf("\n");

}

}

//копирование

element\*\* eqTo(element(\*M)[SIZE]) {

element \*\*W = (element \*\*)malloc(SIZE \* sizeof(element \*));

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

W[row] = (element \*)malloc(SIZE \* sizeof(element));

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

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

W[i][j].d = M[i][j].d;

W[i][j].t = M[i][j].t;

}

return W;

}

//M - модифицированная матрица весов

//n - число вершин в графе

element\*\* floydAlgorithm(element(\*M)[SIZE], int n) {

element \*\*W = (element \*\*)malloc(SIZE \* sizeof(element \*)); //Формируемая матрица W - матрица кратчайших путей между вершинами

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

W[row] = (element \*)malloc(SIZE \* sizeof(element));

W = eqTo(M);//Присвоим матрице W присвоим M

for (int z = 0; z < SIZE; z++)

for (int x = 0; x < SIZE; x++)

for (int y = 0; y < SIZE; y++)

if ((W[x][z].d + W[z][y].d < W[x][y].d)) {

W[x][y].d = W[x][z].d + W[z][y].d;

W[x][y].t = W[z][y].t;

}

return W;

}

int main() {

element M[SIZE][SIZE] = {

{{0, -1}, {1, 0}, {9, 0}, {INF, -2}, {INF, -2}, {INF, -2}},

{{INF, -2}, {0, -1}, {7, 1}, {2, 1}, {INF, -2}, {INF, -2}},

{{INF, -2}, {INF, -2}, {0, -1}, {INF, -2}, {1, 2}, {INF, -2}},

{{INF, -2}, {INF, -2}, {4, 3}, {0, -1}, {8, 3}, {2, 3}},

{{INF, -2}, {INF, -2}, {INF, -2}, {INF, -2}, {0, -1}, {5, 4}},

{{INF, -2}, {INF, -2}, {INF, -2}, {INF, -2}, {5, 5}, {0, -1}} };

int n = SIZE;

int f=0;

//Формируемая матрица W - матрица кратчайших путей между вершинами

element \*\*W = (element \*\*)malloc(SIZE \* sizeof(element \*));

for (int row = 0; row < SIZE; row++) {

W[row] = (element \*)malloc(SIZE \* sizeof(element));

}

W = floydAlgorithm(M, n);

print(W);

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

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

if (i != j && W[i][j].d == W[j][i].d && W[j][i].d != 100)

{

printf("(%d,%d)\n", i, j);

f=1;

}

if(!f)

printf("Takih Vershin Net");

system(“pause”);

return 0;

}

**Результаты работы программы:**

