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:

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

**Задания**

1. Изучить алгоритмы нахождения кратчайших путей между каждой парой вершин во взвешенном орграфе.
2. Разработать и реализовать алгоритм решения задачи (см. варианты заданий).
3. Подобрать тестовые данные. Результат представить в виде диаграммы графа.

**Задача**

Используя алгоритм нахождения кратчайших расстояний между каждой парой вершин в орграфе определить:

а) является ли заданный орграф сильно связным;

б) является ли заданный орграф односторонне связным.

**Выполнение**

#include **<stdio.h>**#include **<stdlib.h>**#define **MAX** 1000  
  
**typedef struct**{  
 **int** t; *//вершина, предшествующая вершине j на кратчайшем пути от i до j* **int** d; *//длина кратчайшего пути от i до g (W[i,j].d)*} BaseType;  
  
  
**int**\*\* InitMem(**int** N){ *//инициализация памяти* **int** i; **int**\*\* G;  
 G = (**int**\*\*)malloc(N\***sizeof**(**int**\*));  
 **for**(i = 0; i < N; i++)  
 G[i] = (**int**\*)calloc(N, **sizeof**(**int**));  
 **return** G;  
}  
  
**void** FreeMem(**int**\*\* G, **int** N)  
{ **int** i; **for**(i = 0; i < N; i++)  
 free(G[i]); free(G);  
}  
  
**void** WriteGraph(**int**\*\* G, **int** N)  
{ *//вывод графа* **int** i, j;  
 printf(**"Орграф:\n"**);  
 **for**(i = 0; i < N; i++)  
 { **for**(j = 0; j < N; j++)  
 **if**(i!=j && G[i][j] == 0)  
 printf(**"I\t"**);*//бесконечность = infinity* **else** printf(**"%i\t"**, G[i][j]); *//вывод числа* printf(**"\n"**);  
 }  
 printf(**"\n"**);  
}  
  
**int**\*\* ReadGraph(**int** \*N)  
{*//считывание графа* **int**\*\* G, tmp, i, j; FILE \*f;  
 f = fopen(**"input.txt"**,**"r"**); *//чтение из файла с именем input* fscanf(f,**"%i"**,&tmp);  
 G = InitMem(tmp);  
 **for**(i = 0; i < tmp; i++){  
 **for**(j = 0; j < tmp; j++)  
 fscanf(f, **"%i"**, &G[i][j]);  
 }  
 fclose(f);  
 (\*N)=tmp;  
 **return** G;  
}  
  
BaseType\*\* ModMatr(**int**\*\* G, **int** N)*//создание матрицы весов*{  
 */\*int t; //вершина, предшествующая вершине j на кратчайшем пути от i до j  
 int d; //длина кратчайшего пути от i до g (W[i,j].d)\*/* BaseType\*\* M;  
 **int** i, j;  
 M = (BaseType\*\*)malloc(N\***sizeof**(BaseType\*));  
 **for**(i = 0; i < N; i++)  
 M[i] = (BaseType\*)malloc(N\***sizeof**(BaseType)); *//выделение памяти на матрицувесов М* **for**(i = 0; i < N; i++)  
 **for**(j = 0; j < N; j++)  
 {  
 **if**(G[i][j] == 0) *//если нет путей* M[i][j].d = **MAX**; *//то к .d присвоим бесконечность* **else** M[i][j].d = G[i][j]; *//иначе сущесвуюущую длину* **if**(i==j) *//если главная диагональ, то нули* {  
 M[i][j].t = 0;  
 M[i][j].d = 0;  
 }  
 **else** *//иначе, если элемент не равен нулю* {  
 **if**(G[i][j] != 0)*//если элемент не равен нулю* M[i][j].t = G[i][j]; *//присвоим его* **else** M[i][j].t = -1; *//иначе -1 (такого нет)* }  
 }  
 **return** M;  
}  
**void** WriteMatrWes(BaseType\*\* G, **int** N)  
{*//печать матрицы весов* **int** i, j;  
 **for**(i = 0; i < N; i++)  
 {  
 **for**(j = 0; j < N; j++)  
 {*//вывод матрицы весов, если беск, то печать I, иначе печать содержимого* (G[i][j].d == **MAX**) ? printf(**"(I; "**) : printf(**"(%d; "**, G[i][j].d);  
 (G[i][j].t == **MAX**) ? printf(**"I)"**) : printf(**"%d)"**, G[i][j].t);  
 printf(**"\t"**);  
 }  
 printf(**"\n"**);  
 }  
 printf(**"\n"**);  
}  
  
**void** CopyMatr (BaseType\*\* M, BaseType\*\* W, **int** N) *//копирование М в W*{  
 **int** i, j;  
 **for** (i = 0; i < N; i++)  
 **for** (j = 0; j < N; j++)  
 W[i][j] = M[i][j];  
}  
  
BaseType \*\*AlgFloyd (BaseType\*\* M, **int** N) *// алгоритм флойда*{  
 BaseType\*\* W;  
 **int** z, x, y;  
 W = (BaseType\*\*)malloc(N\***sizeof**(BaseType\*)); *//выделение памяти* **for**(z = 0; z < N; z++)  
 W[z] = (BaseType\*)malloc(N\***sizeof**(BaseType));  
 CopyMatr(M, W, N); *//копирвоание матрицы весов М в матрицу W* **for** (z = 0; z < N; z++)  
 **for** (x = 0; x < N; x++)  
 **for** (y = 0; y < N; 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** CheckGraf (BaseType\*\* W, **int** N) *//проверка матрицы*{  
 **int** flag1 = (W[0][1].d != **MAX**) || (W[1][0].d != **MAX**); *//слабосвязный* **int** flag2 = (W[0][1].d != **MAX**) && (W[1][0].d != **MAX**); *//сильносвязный* **int** i, j;  
 **if** (flag2) *//сначала проверим условие для сильно связанного графа* { *//т.к. сильносвязный явл. слабо связанным и тогда проверка будет не корректной* **for** (i = 0; i < N; i++)  
 **for** (j = i+1; j < N; j++)  
 flag2 = (W[i][j].d != **MAX**) && (W[j][i].d != **MAX**);  
 **if** (flag2) **return** 2;  
 }  
 **if** (flag1)  
 {  
 **for** (i = 0; i < N; i++)  
 **for** (j = i+1; j < N; j++)  
 flag1 = (W[i][j].d != **MAX**) || (W[j][i].d != **MAX**);  
 **if** (flag1)  
 **return** 1;  
 }  
 **return** 0;  
}  
  
**int** main()  
{  
  
 **int**\*\* G, N;  
 BaseType \*\*M, \*\*W;  
 G = ReadGraph(&N); *//считывание графа* WriteGraph(G, N); *//вывод графа* M = ModMatr(G, N); *//Создание матрицы весов* W = AlgFloyd(M, N); *//Применение алгоритма Флойда* WriteMatrWes(W, N); *//Вывод матрицы после выполения алгоритма флойда* **switch** (CheckGraf(W, N))  
 {  
 **case** 0: { printf(**"Связь слабая или отсутствует.\n"**); **break**; }  
 **case** 1: { printf(**"Односторонне связанный граф.\n"**); **break**; }  
 **case** 2: { printf(**"Сильно связанный граф.\n"**); **break**; }  
 }  
 **return** 0;  
}





