Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №10

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Поиск расстояний во взвешенном графе»

**Выполнили студенты группы 21вв1:**

Нечаев А.Д.

Киреев Д.А.

**Приняли**

Юрова О.В.

Акифьев И.В.

Пенза 2022

**Методические указания.**

**Задание 1**

1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного взвешенного графа *G*. Выведите матрицу на экран.

 Для сгенерированного графа осуществите процедуру поиска расстояний, реализованную в соответствии с приведенным выше описанием. При  реализации алгоритма в качестве очереди используйте класс **queue** из стандартной библиотеки С++.

**Задание 2**

1. Для каждого из вариантов сгенерированных графов (ориентированного и не ориентированного) определите радиус и диаметр.

**Описание метода решения задачи:**

**Вход**: G – матрица смежности графа, v – исходная вершина.

**Выход**: DIST – вектор расстояний до всех вершин от исходной.

**Алгоритм ПОШ**

1.1. для всех i положим DIST [i] =  -1 пометим как "не посещенную";

1.2. **ВЫПОЛНЯТЬ** BFSD (v).

1.3  для всех i вывести DIST [i] на экран;

**Алгоритм** BFSD(v):

2.1. Создать пустую очередь Q = {};

2.2. Поместить v в очередь Q.push(v);

2.3. Обновить вектор расстояний  DIST [ x ] = 0;

2.4. **ПОКА**  Q != ∅ очередь не пуста **ВЫПОЛНЯТЬ**

2.5. v = Q.front() установить текущую вершину;

2.6. Удалить первый элемент из очереди Q.pop();

2.7. вывести на экран v;

2.8. **ДЛЯ** i = 1 **ДО** size\_G **ВЫПОЛНЯТЬ**

2.9.   **ЕСЛИ**  G(v,i) > 0 **И** DIST = = -1

2.10.         **ТО**

2.11. Поместить i в очередь Q.push(i);

2.12. Обновить вектор расстояний DIST [ i ] = DIST [ v ] + G(v,i);

**Листинг:**

#include <queue>

#include <iostream>

#include <locale>

using namespace std;

void BFS(int\* (\*mass), int\* be, int pos, int n)

{

queue <int> q;

q.push(pos);

be[pos] = 0;

while (!q.empty())

{

pos = q.front();

printf("%d ", pos + 1);

q.pop();

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

{

if (mass[pos][i] != 0 && be[i] == -1)

{

q.push(i);

be[i] = be[pos] + mass[pos][i];

}

}

}

}

int main()

{

setlocale(LC\_ALL, "Rus");

int\*\* a, n;

int i = 0, j = 0, w;

printf("Введите кол-во строк и столбцов\n");

scanf\_s("%d", &n);

a = (int\*\*)malloc(n \* sizeof(int\*));

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

{

a[i] = (int\*)malloc(n \* sizeof(int));

}

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

{

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

{

/\*w = rand() % 2;

if (w == 1)

{

a[i][j] = rand() % 10;

a[j][i] = a[i][j];

}

else {

a[i][j] = 0;

a[j][i] = a[i][j];

}

if (i == j)

{

a[i][j] = 0;

}\*/

if (i == j) { a[i][j] = 0; }

else

{

a[i][j] = rand() % 6;

a[j][i] = a[i][j];

}

}

}

for (int i = 0; i < n + 1; i++)

{

for (int j = 0; j < n + 1; j++)

{

if (i == 0 && j == 0)

{

printf("# ");

}

else

{

if (i == 0 || j == 0)

{

if (i == 0) { printf("%d ", j); }

else { printf("%d ", i); }

}

else

{

printf("%d ", a[i - 1][j - 1]);

}

}

}

printf("\n");

}

int pos = 0;

int\* be = (int\*)malloc(n \* sizeof(int));

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

{

be[i] = -1;

}

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

{

if (be[i] == -1)

{

printf("Введите вершину\n");

scanf\_s("%d", &pos);

BFS(a, be, pos - 1, n);

}

}

printf("\n");

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

{

printf("расстояние от %d до %d = %d\n", pos, i + 1, be[i]);

}

int max = 0, min = 0;

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

{

if (be[i] > be[max])

{

max = be[i];

}

if(be[i]<be[min])

{

min = be[i];

}

}

printf("диаметр= %d\n", max);

printf("радиус= %d\n", min);

free(a);

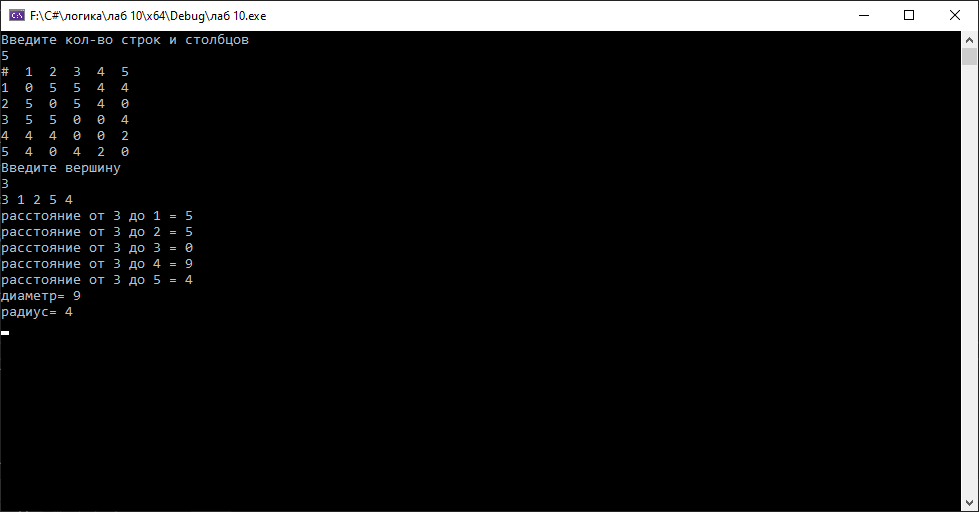
getchar();

getchar();

}

**Результаты работы программы:**

**Задание :**

****

**Вывод:** в ходе выполнения лабораторной работы мы написали программу поиска расстояния во взвешенном графе. Результаты работы программ совпали с результатами трассировки, следовательно программы работают без ошибок.