Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**Отчёт**

по лабораторной работе №7

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Поиск расстояний во взвешенном графе»

Выполнил:

 студент группы 20ВВ2

Лазутин Д.Д.

 Принял:

 к.т.н., доцент Юрова О.В.

Пенза 2020

Задание 1

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

2. 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++.

3.\* Сгенерируйте (используя генератор случайных чисел) матрицу смежности для ориентированного взвешенного графа *G*. Выведите матрицу на экран и осуществите процедуру поиска расстояний, реализованную в соответствии с приведенным выше описанием.

Задание 2\*

1. Модернизируйте программу так, чтобы получить возможность запуска программы с параметрами командной строки. В качестве параметра должны указываться тип графа (взвешенный или нет) и наличие ориентации его ребер (есть ориентация или нет).

**Описание метода решения:**

Объявлены переменные: матрица G, размер матрицы S, массив посещённых вершин графа Visited, счётчики i и j, глобальная переменная – очередь q, номер стартовой вершины N, индикатор взвешенности графа Ves, индикатор ориентированности графа O.

Присваиваем каждому элементу массива Distance максимально возможное значение, определённое макросом MAXINT. Создаём и заполняем матрицу смежности в соответствии с введёнными пользователем параметрами (размер, взвешенность, ориентированность), выводим её на экран. Далее выполняем обход графа в ширину посредством вызова функции BFS.

Элемент массива Distance[i], равный MAXINT, считается номером изолированной вершины, а элемент, равный 0 – номером стартовой вершины (т.е. откуда начался обход). Остальные элементы будут больше 0 и меньше MAXINT, т.е. достижимы из стартовой вершины через 1 или несколько рёбер.

**Листинг** LZ\_7.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <queue>

#include <iostream>

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include <locale.h>

using namespace std;

#define MAXINT 32767

queue <int> q;

void BFS(int v, int \*\*M, int \*Dist, int Size)

{

int i;

q.push(v); Dist[v]=0;

while (!q.empty())

{

v=q.front();

q.pop();

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

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

{

if((Dist[i]>Dist[v]+M[v][i])&&(M[v][i]>0))

{

q.push(i);

Dist[i]=Dist[v]+M[v][i];

}

}

}

}

void main()

{

int \*\*G=NULL, i=0, j=0, S=0, \*Distance=NULL, N, Ves, O;

setlocale(LC\_ALL, "Rus");

srand(time(NULL));

printf("Введите размерность матрицы смежности: "); scanf("%d", &S);

printf("Введите:\n1-граф взвешенный, 2-граф невзвешенный: "); scanf("%d", &Ves);

printf("Введите:\n1-граф неориентированный, 2-граф ориентированный: "); scanf("%d", &O);

while(S<=0 || (Ves<1 || Ves>2) || (O<1 || O>2))

{

printf("Неправильно введённые параметры! Введите положительные числа (или те, что Вам предложены):\n");

scanf("%d%d%d", &S, &Ves, &O);

}

G=(int\*\*)malloc(S\*sizeof(int));

Distance=(int\*)malloc(S\*sizeof(int));

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

Distance[i]=MAXINT;

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

{

G[i]=(int\*)malloc(S\*sizeof(int));

for(j=0;j<S;j++)

{

if(Ves==1)

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

else

G[i][j]=rand()%2;

G[i][i]=0;

}

}

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

{

printf("\n%4d|", i+1);

for(j=0;j<S;j++)

{

if(O==1)

G[j][i]=G[i][j];

printf("%2d", G[i][j]);

}

}

printf("\n\nС какой вершины начать обход в ширину? "); scanf("%d", &N);

while (N<1 || N>S)

{

printf("Некорректно введённое число! Введите целое положительное число в пределах размера матрицы смежности: ");

scanf("%d", &N);

}

printf("Порядок обхода графа в ширину (возможны повторяющиеся вершины в связи с обновлением расстояний):\n");

BFS(N-1, G, Distance, S);

printf("\nВектор расстояний от выбранной вершины до всех остальных: ");

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

printf("%d ", Distance[i]);

printf("\n");

free(Distance);

free(G);

}

**Результаты работы программ**

Результаты работы программы для 1-4-го заданий представлены на рис.1–6.

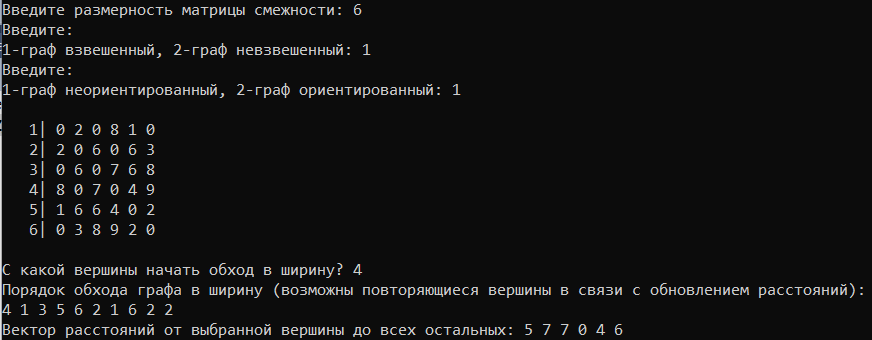


Рис.1

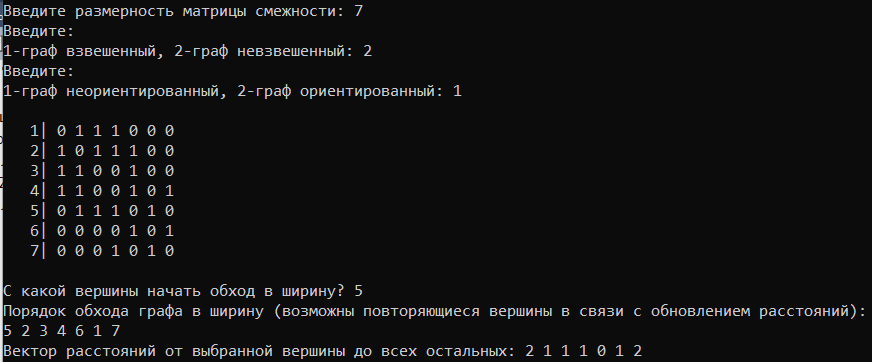


Рис.2

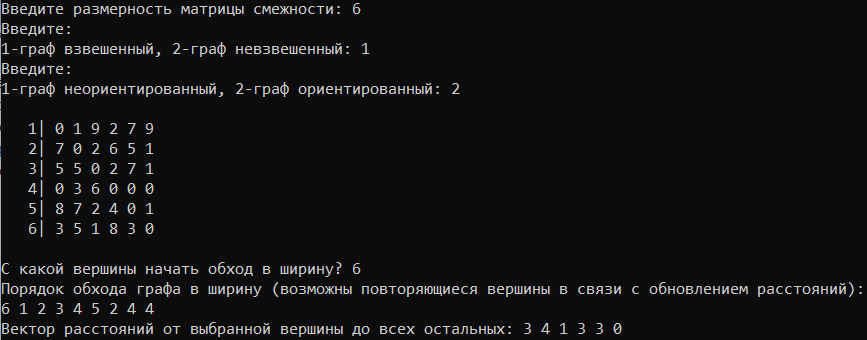


Рис.3

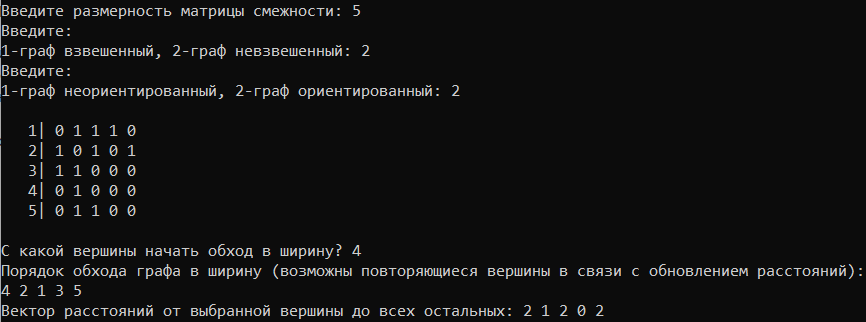


Рис.4