Министерство образования и науки Российской Федерации

Пензенский государственный университет

Факультет вычислительной техники

**Отчёт**

По лабораторной работе №6

По курсу "ЛиОАвИЗ"

Выполнили:

Студент группы 20ВВ1

Репин И.В.

Приняла:

Юрова О.В.

Пенза 2021

**Название:** Поиск расстояний во взвешенном графе

**Цель работы:** выполнить ряд заданий

**Лабораторное задание:**

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 (см. описание ниже). В

качестве параметра должны указываться тип графа (взвешенный или нет) и

наличие ориентации его ребер (есть ориентация или нет).

**Описание метода решения задачи:**

Задание 1.

1. Алгоритм ПОШ

1.1. для всех i положим DIST [i] = 1000 пометим как &quot;не посещенную&quot;;

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) &gt; 0 И DIST [ i ] &gt; DIST [ v ] + G(v,i)

2.10. ТО

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

2.12. Обновить вектор расстояний DIST [ i ] = DIST [ v ] +

G(v,i);

1. Аналогичную функцию реализуем для ориентированного взвешенного графа

**Листинг:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <queue>

#include <time.h>

#include <Windows.h>

using namespace std;

int\*\* m;

int\*\* mo;

int\*\* mnn;

int\*\* myn;

int num;

int\* vis;

int\* visno;

queue <int> q;

void BFSD(int s) {

vis[s] = 0;

q.push(s);

while (!q.empty()) {

int v = q.front();

q.pop();

printf("%4i", v);

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

if ((m[v][i] > 0) && (vis[i] > (vis[v] + m[v][i]))) {

q.push(i);

vis[i] = vis[v] + m[v][i];

}

}

}

printf("\nDists: ");

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

printf("%4i", vis[i]);

}

}

void BFSDO(int s) {

vis[s] = 0;

q.push(s);

while (!q.empty()) {

int v = q.front();

q.pop();

printf("%4i", v);

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

if ((mo[v][i] > 0) && (vis[i] > (vis[v] + mo[v][i]))) {

q.push(i);

vis[i] = vis[v] + mo[v][i];

}

}

}

printf("\nDists: ");

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

printf("%4i", vis[i]);

}

}

void BFS(int s) {

visno[s] = 0;

q.push(s);

while (!q.empty()) {

int v = q.front();

q.pop();

printf("%4i", v);

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

if ((mnn[v][i] == 1) and (visno[i] == -1)) {

q.push(i);

visno[i] = visno[v] + 1;

}

}

}

printf("\nDists: ");

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

printf("%4i", visno[i]);

}

}

void BFSO(int s) {

visno[s] = 0;

q.push(s);

while (!q.empty()) {

int v = q.front();

q.pop();

printf("%4i", v);

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

if ((myn[v][i] == 1) and (visno[i] == -1)) {

q.push(i);

visno[i] = visno[v] + 1;

}

}

}

printf("\nDists: ");

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

printf("%4i", visno[i]);

}

}

int main(int argc, char \* argv[])

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

//generate array

if(argc <= 2){

printf("Params not found");

return false;

}else{

printf("Params:\n");

printf("Oriented: %s\n", argv[1]);

printf("Weighted: %s\n", argv[2]);

}

int generate, choice;

printf("Input array size: ");

scanf("%i", &num);

m = new int\* [num];

mo = new int\* [num];

mnn = new int\* [num];

myn = new int\* [num];

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

m[i] = new int[num];

mo[i] = new int[num];

mnn[i] = new int[num];

myn[i] = new int[num];

}

srand(time(NULL));

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

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

if (j == i) {

m[i][j] = 0;

}

else if (j > i) {

generate = rand() % 100;

if (generate >= 30) {

m[i][j] = rand() % 100;

}

else {

m[i][j] = 0;

}

}

else {

m[i][j] = m[j][i];

}

}

}

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

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

if (j == i) {

mo[i][j] = 0;

}

else if (j > i) {

generate = rand() % 100;

if (generate >= 30) {

mo[i][j] = rand() % 100;

}

else {

mo[i][j] = 0;

}

}

else {

generate = rand() % 100;

if (generate >= 30) {

if (mo[j][i] == 0) {

int generate2 = rand() % 100;

if (generate2 >= 30) {

mo[i][j] = rand() % 100;

}

else {

mo[i][j] = 0;

}

}

else {

mo[i][j] = mo[j][i];

}

}

else {

mo[i][j] = 0;

}

}

}

}

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

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

if (j == i) {

mnn[i][j] = 0;

}

else if (j > i) {

generate = rand() % 100;

if (generate >= 30) {

mnn[i][j] = 1;

}

else {

mnn[i][j] = 0;

}

}

else {

mnn[i][j] = mnn[j][i];

}

}

}

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

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

if (j == i) {

myn[i][j] = 0;

}

else if (j > i) {

generate = rand() % 100;

if (generate >= 30) {

myn[i][j] = 1;

}

else {

myn[i][j] = 0;

}

}

else {

generate = rand() % 100;

if (generate >= 30) {

if (myn[j][i] == 0) {

int generate2 = rand() % 100;

if (generate2 >= 30) {

myn[i][j] = 1;

}

else {

myn[i][j] = 0;

}

}

else {

myn[i][j] = myn[j][i];

}

}

else {

myn[i][j] = 0;

}

}

}

}

//print array

if ((\*argv[1] == 'n') && (\*argv[2] == 'y')) {

printf("array:");

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

printf("\n");

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

printf("%4i", m[i][j]);

}

}

}

if ((\*argv[1] == 'y') && (\*argv[2] == 'y')) {

printf("\n array:");

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

printf("\n");

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

printf("%4i", mo[i][j]);

}

}

}

if ((\*argv[1] == 'n') && (\*argv[2] == 'n')) {

printf("\n array:");

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

printf("\n");

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

printf("%4i", mnn[i][j]);

}

}

}

if ((\*argv[1] == 'y') && (\*argv[2] == 'n')) {

printf("\n array:");

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

printf("\n");

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

printf("%4i", myn[i][j]);

}

}

}

printf("\nInput start point: ");

scanf("%i", &choice);

//create arrays

vis = new int[num];

visno = new int[num];

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

vis[i] = 1000;

visno[i] = -1;

}

//result

if((\*argv[1] == 'n') && (\*argv[2] == 'y')){

printf("\nBFSD: ");

BFSD(choice);

}

if((\*argv[1] == 'y') && (\*argv[2] == 'y')){

printf("\nBFSDO: ");

BFSDO(choice);

}

if ((\*argv[1] == 'n') && (\*argv[2] == 'n')) {

printf("\nBFS: ");

BFS(choice);

}

if ((\*argv[1] == 'y') && (\*argv[2] == 'n')) {

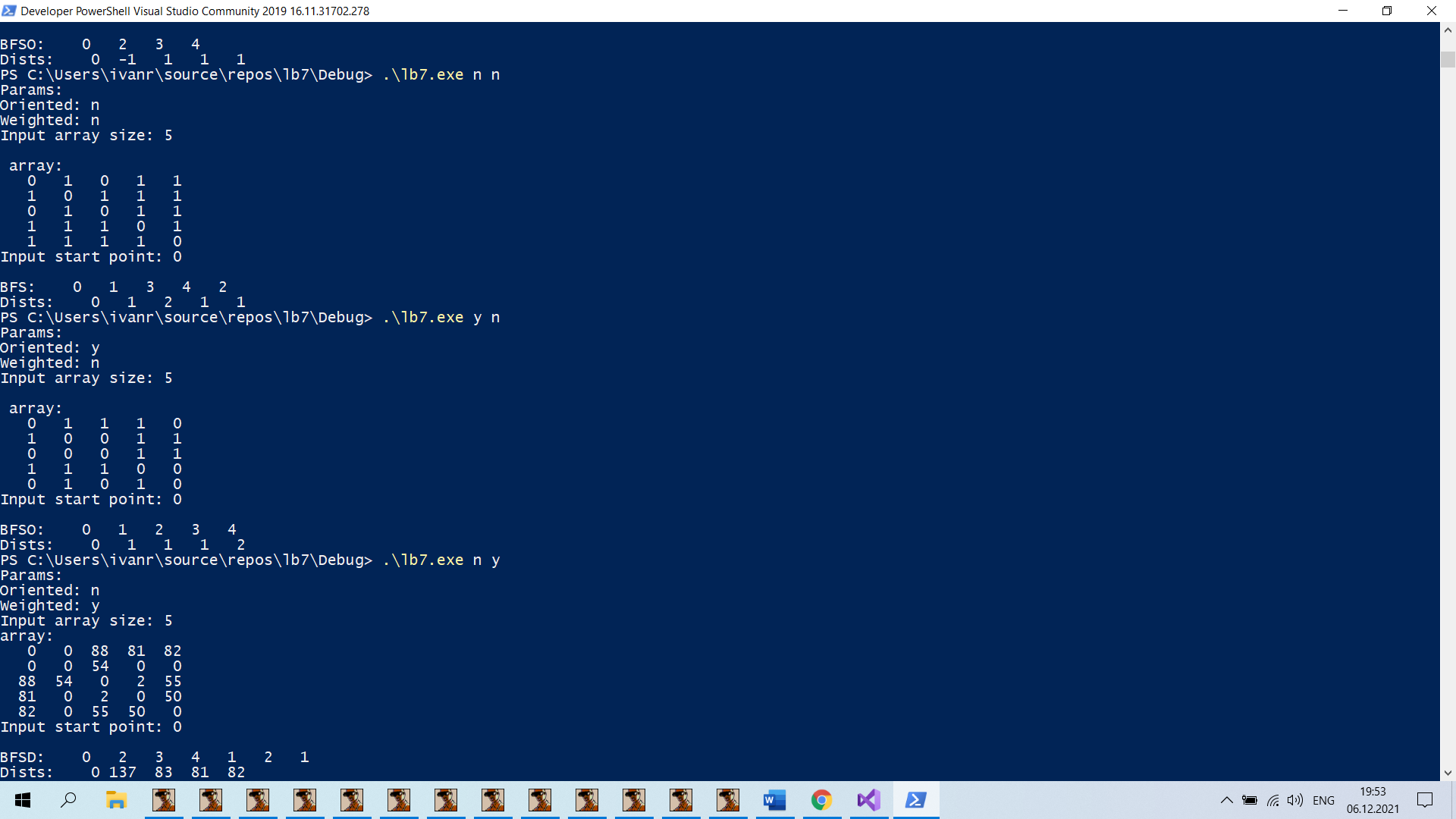
printf("\nBFSO: ");

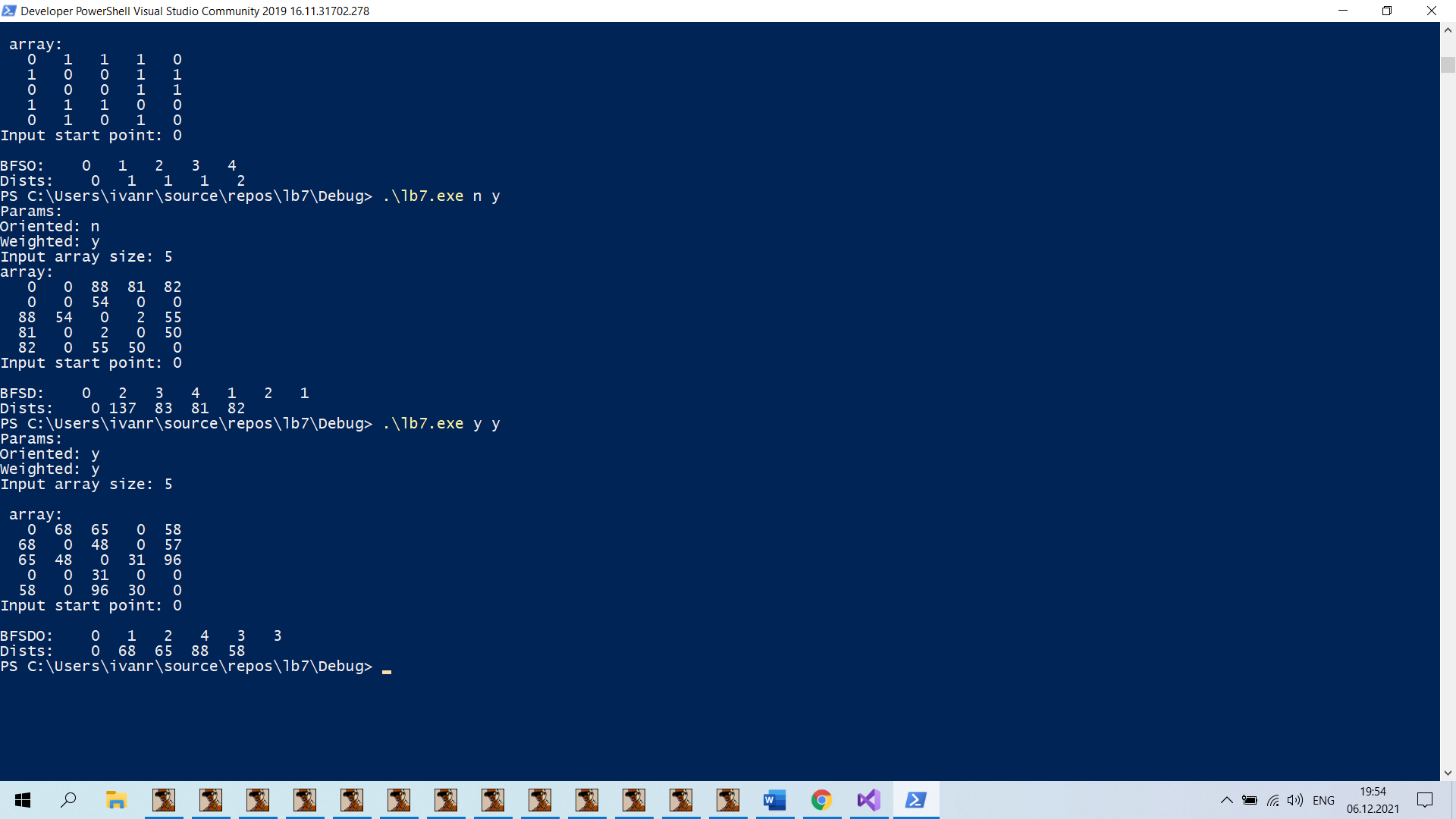
BFSO(choice);

}

}

**Результаты работы программы:**





**Вывод:** научились искать расстояния в взвешенном графе и работать с параметрами командной строки