Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №7

# по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «поиск расстояния во взвешенном графе»

**Выполнил:**

студент группы 20ВВ3

Духнов Олег

**Приняли:**

Юрова О.В.

Митрохин М.А.

Пенза 2021

# Название

Поиск расстояний во взвешенном графе.

# 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++.

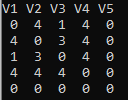
**3.**\* Сгенерируйте (используя генератор случайных чисел) матрицу смежности для ориентированного взвешенного графа *G*. Выведите матрицу на экран и осуществите процедуру поиска расстояний, реализованную в соответствии с приведенным выше описанием.

**Задание 2\***

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

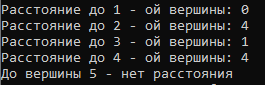
# Результат работы программы

**Задание 1.1.** Результаты работы программы показаны на рисунке 1.



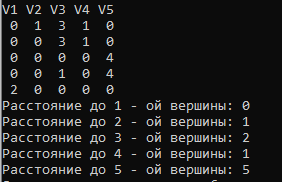
**Рисунок 1 – Результат работы программы**

**Задание 1.2.** Результат поиска расстояния.



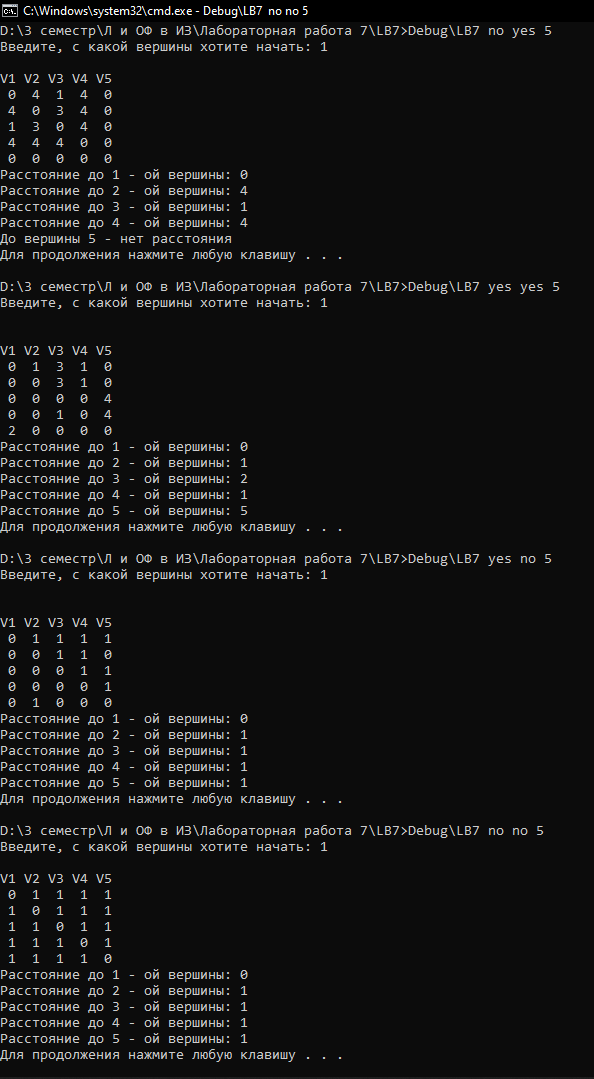
**Рисунок 2 – Результат работы программы**

**Задание 1.3.** Результат поиска расстояния для ориентированного взвешенного графа.



**Рисунок 3 – Результат работы программы**

**Задание 2.1.** Модернизировал программу.



**Рисунок 4 – Результат работы программы**

**Вывод:** яизучил алгоритм поиска расстояний во взвешенном графе.

**Листинг**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <stdbool.h>

#include <conio.h>

#include <time.h>

#include <stack>

#include <queue>

using namespace std;

int\*\* neor\_vz(int n)

{

int s = 1;

int\*\* Arr;

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

srand(time(NULL));

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

{

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

Arr[i][i] = 0;

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

{

if (rand() % 100 > 70)

{

Arr[i][j] = 0;

}

else

{

Arr[i][j] = rand() % n;

}

}

s++;

}

s = 1;

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

{

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

Arr[j][i] = Arr[i][j];

s++;

}

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

printf("V%d ", i + 1);

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

{

printf("\n");

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

{

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

}

}

return Arr;

}

int\*\* orent\_vz(int n)

{

int\*\* Arr2 = (int\*\*)malloc(n \* sizeof(int\*));

srand(time(NULL));

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

{

Arr2[i] = (int\*)malloc(n \* sizeof(int));

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

{

if (rand() % 100 > 70)

{

Arr2[i][j] = 0;

}

else

{

Arr2[i][j] = rand() % n;

}

}

}

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

{

Arr2[i][i] = 0;

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

{

if (Arr2[i][j] > 0)

Arr2[j][i] = 0;

}

}

printf("\n");

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

printf("V%d ", i + 1);

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

{

printf("\n");

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

printf("%2d ", Arr2[i][j]);

}

return Arr2;

}

int\*\* neor\_nevz(int n)

{

int s = 1;

int\*\* Arr;

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

srand(time(NULL));

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

{

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

Arr[i][i] = 0;

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

{

if (rand() % 100 > 70)

{

Arr[i][j] = 0;

}

else {

Arr[i][j] = 1;

}

}

s++;

}

s = 1;

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

{

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

Arr[j][i] = Arr[i][j];

s++;

}

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

printf("V%d ", i + 1);

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

{

printf("\n");

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

{

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

}

}

return Arr;

}

int\*\* orent\_nevz(int n)

{

int\*\* Arr2 = (int\*\*)malloc(n \* sizeof(int\*));

srand(time(NULL));

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

{

Arr2[i] = (int\*)malloc(n \* sizeof(int));

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

{

if (rand() % 100 > 70)

{

Arr2[i][j] = 0;

}

else

{

Arr2[i][j] = 1;

}

}

}

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

{

Arr2[i][i] = 0;

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

{

if (Arr2[i][j] > 0)

Arr2[j][i] = 0;

}

}

printf("\n");

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

printf("V%d ", i + 1);

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

{

printf("\n");

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

printf("%2d ", Arr2[i][j]);

}

return Arr2;

}

void BFSD(int x, int\* dist, int\*\* Arr, int n)

{

queue<int> que;

que.push(x);

dist[x] = 0;

while (!que.empty())

{

x = que.front();

que.pop();

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

{

if (Arr[x][i] > 0 && dist[i] > dist[x] + Arr[x][i])

{

que.push(i);

dist[i] = dist[x] + Arr[x][i];

}

}

}

}

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

{

setlocale(LC\_ALL, "Ru");

int n, \*\* n\_v, x, \*\* o\_v, \*\* n\_n, \*\* o\_n;

char\* d;

//printf("Введите размер матрицы: ");

//scanf("%d", &n);

sscanf(argv[3], "%d", &n);

printf("Введите, с какой вершины хотите начать: ");

scanf("%d", &x);

while (x < 1 || x > n)

{

printf("С этой вершины нельзя начинать! Выберите другю вершину: ");

scanf("%d", &x);

}

x--;

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

if (strcmp("no", argv[1]) == 0 && strcmp("yes", argv[2]) == 0)

{

printf("\n");

n\_v = neor\_vz(n);

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

dist[i] = n;

BFSD(x, dist, n\_v, n);

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

{

if (dist[i] != n)

printf("\nРасстояние до %d - ой вершины: %d ", i + 1, dist[i]);

else

printf("\nДо вершины %d - нет расстояния", i + 1);

}

printf("\n");

}

if (strcmp("yes", argv[1]) == 0 && strcmp("no", argv[2]) == 0)

{

printf("\n");

o\_n = orent\_nevz(n);

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

dist[i] = INT\_MAX;

BFSD(x, dist, o\_n, n);

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

{

if (dist[i] != INT\_MAX)

printf("\nРасстояние до %d - ой вершины: %d ", i + 1, dist[i]);

else

printf("\nДо вершины %d - нет расстояния", i + 1);

}

printf("\n");

}

if (strcmp("no", argv[1]) == 0 && strcmp("no", argv[2]) == 0)

{

printf("\n");

n\_n = neor\_nevz(n);

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

dist[i] = INT\_MAX;

BFSD(x, dist, n\_n, n);

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

{

if (dist[i] != INT\_MAX)

printf("\nРасстояние до %d - ой вершины: %d ", i + 1, dist[i]);

else

printf("\nДо вершины %d - нет расстояния", i + 1);

}

printf("\n");

}

if (strcmp("yes", argv[1]) == 0 && strcmp("yes", argv[2]) == 0)

{

printf("\n");

o\_v = orent\_vz(n);

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

dist[i] = INT\_MAX;

BFSD(x, dist, o\_v, n);

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

{

if (dist[i] != INT\_MAX)

printf("\nРасстояние до %d - ой вершины: %d ", i + 1, dist[i]);

else

printf("\nДо вершины %d - нет расстояния", i + 1);

}

printf("\n");

}

system("pause");

}