Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №5

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

на тему «Обход графа в ширину»

**Выполнили:**

студенты группы 20ВВ3

Халилов Алмаз

Китаев Андрей

Новиков Иван

**Приняли:**

Юрова О.В.

Митрохин М.А.

Пеза 2021

# **Название**

Оценка времени выполнения программ

**Цель работы** – изучение обхода графов в ширину.

# **Методические указания**

Обход графа в ширину – еще один распространенный способ обхода графов.

Вход: G – матрица смежности графа.

Выход: номера вершин в порядке их прохождения на экране.

Алгоритм ПОШ

1.1. для всех i положим NUM[i] = False пометим как "не посещенную";

1.2. ПОКА существует "новая" вершина v

1.3. ВЫПОЛНЯТЬ BFS (v).

Алгоритм BFS(v):

2.1. Создать пустую очередь Q = {};

2.2. Поместить v в очередь Q.push(v);

2.3. пометить v как "посещенную" NUM[v] = True;

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) = = 1И NUM[i] = = False

2.10. ТО

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

2.12. пометить v как "посещенную" NUM[v] = True;

# **Лабораторное задание**

**Задание 1:**

1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного графа *G*. Выведите матрицу на экран.
2. Для сгенерированного графа осуществите процедуру обхода в ширину, реализованную в соответствии с приведенным выше описанием. При реализации алгоритма в качестве очереди используйте класс **queue** из стандартной библиотеки С++.

3**.**\* Реализуйте процедуру обхода в ширину для графа, представленного списками смежности.

**Ход Работы**

**Листинг**

#define \_CRT\_NONSTDC\_NO\_WARNINGS

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <locale.h>

#include <string.h>

#include <windows.h>

#include <queue>

using namespace std;

void matrix(int\*\* g, int size)

{

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

{

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

{

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

}

}

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

{

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

{

if (i == j)

g[i][j] = 0;

else

g[i][j] = g[j][i];

}

}

printf("Матрица: \n");

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

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

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

}

printf("\n");

}

}

void F(int\*\* g, int\* NUM, int v, int size)

{

queue <int> q;

q.push(v);

NUM[v] = 1;

while (!q.empty())

{

v = q.front();

q.pop();

printf("%d ", v);

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

{

if ((g[v][i] == 1) && (NUM[i] == 0))

{

q.push(i);

NUM[i] = 1;

}

}

}

}

void main()

{

setlocale(LC\_ALL, "RUS");

int i = 0;

int j = 0;

int size = 0;

int\*\* g = 0;

int v = 0;

setlocale(LC\_ALL, "RUS");

srand(time(NULL));

printf("Введите размер матрицы: ");

scanf("%d", &size);

g = (int\*\*)malloc(size \* sizeof(int\*));

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

{

g[i] = (int\*)malloc(size \* sizeof(int));

}

matrix(g, size);

printf("Точка входа: ");

scanf("%d", &v);

int\* NUM = (int\*)malloc(size \* sizeof(int));

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

{

NUM[i] = 0;

}

F(g, NUM, v, size);

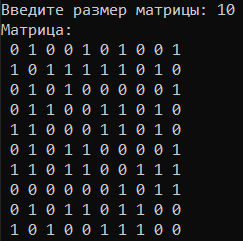
printf("\n");

system("pause");}

**Задание 1:**

1. Сгенерировали (используя генератор случайных чисел) матрицу

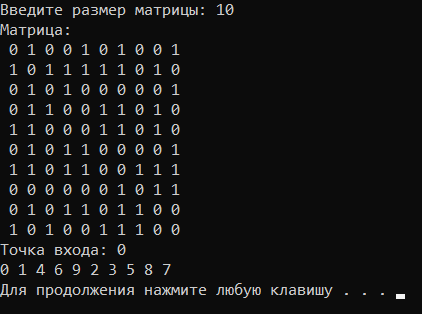
смежности для неориентированного графа G и вывели матрицу на экран.



*Рисунок 1. Генерация массива*

2. Для сгенерированного графа осуществили процедуру обхода в

ширину, реализованную в соответствии с приведенным выше описанием.



*Рисунок 2. Ввод точки входа и результат выполнения программы.*

**Вывод:** изучили обход графа – одну из наиболее распространенных операций с графами. Написали и протестировали программу, которая создает граф определенного размера и обходит его.