ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Лабораторная работа по дисциплине

«Л и ОА в ИТ»

на тему

## «Обход графа в ширину»

Выполнил студент группы 19ВВ1:

Петухов М. Д.

Принял:

Митрохин М. А.

Пенза 2020

**Теоретическая часть:**

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

Основная идея такого обхода состоит в том, чтобы посещать вершины по уровням удаленности от исходной вершины. Удалённость в данном случае понимается как количество ребер, по которым необходимо прейти до достижения вершины. Например, если для графа на рисунке 1 начать обход из первой вершины, то вершины 3, 6 и 2 будут находиться на уровне удаленности в 1 ребро, а вершины 5 и 4 на уровне удаленности в 2 ребра.

Тогда при обходе этого графа в ширину, мы сначала посетим вершины первого уровня удаленности (с номерами 2, 3 и 6), и только после того, как закончатся не посещенные вершины на этом уровне, мы перейдем к следующему. На втором уровне мы посетим все вершины, которые удалены от исходной на 2 ребра (вершины 4 и 5).

Так, алгоритм обхода в ширину продолжает осматривать уровень за уровнем, пока не пройдет все доступные вершины.

Чтобы не заходить повторно в уже пройденные вершины, они помечаются, как и в алгоритме обхода в глубину.

Для того, чтобы проход осуществлялся по уровням необходимо хранить информацию о требуемом порядке посещения вершин. Вершины, которые являются ближайшими соседями исходной вершины (из которой начат обход) должны быть посещены раньше, чем соседи соседей и т.д. Такой порядок позволяет задать структура данных «очередь». Просматривая строку матрицы смежности (или список смежности) для текущей вершины мы помещаем всех её ещё не посещенных соседей в очередь. На следующей итерации текущей вершиной становится та, которая стоит в очереди первой и уже её не посещенные соседи будут помещены в очередь. Но место в очереди они займут после тех вершин, которые были помещены туда на предыдущих итерациях.

Таким образом, можно предложить следующую реализацию алгоритма обхода в ширину.

**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) = = 1**И** NUM[i] = = False

2.10. **ТО**

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

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

Реализация состоит из подготовительной части, в которой все вершины помечаются как не поcещенные (п.1.1) и осуществляется запуск процедуры обхода для вершин графа (п.1.2, 1.3).

В самой процедуре обхода сначала создается пустая очередь (п. 2.1), в которую помещается исходная вершина, из которой начат обход (п.2.2).

Далее итерационно, пока очередь не опустеет, из нее извлекается первый элемент, который становится текущей вершиной (п. 2.5, 2.6). Затем в цикле просматривается **v**-я строка матрицы смежности графа G(v,i). Как только алгоритм встречает смежную с **v** не посещенную вершину (п.2.9), эта вершина помещается в очередь (п.2.11) и помечается как посещенная (п.2.12). После просмотра строки матрицы смежности алгоритм делает следующую итерацию цикла 2.4 или заканчивает работу, если очередь пуста.

**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++.

**Практическая часть:**

**Листинг:**

#include "stdafx.h"

#include <conio.h>

#include <stdlib.h>

#include <queue>

int G[6][6], Z[6];

using namespace std;

queue <int> Q;

void Wit(int k)

{

for (int l=0;l<6;l++)

if (G[k][l]==1 && Z[l]==0)

{

Q.push(l);

Z[l]=1;

}

if (Q.empty())

{}

else{

k=Q.front();

printf("%d ",k);

Q.pop();

Wit (k);

}

}

int main()

{

int i,j;

for (i=0;i<6;i++)

{

for (j=i;j<6;j++)

{

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

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

}

G[i][i]=0;

}

for (i=0;i<6;i++)

{

for (j=0;j<6;j++)

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

printf("\n");

}

printf("\n");

scanf("%d", &i);

Z[i]=1;

printf("\n");

printf("%d ",i);

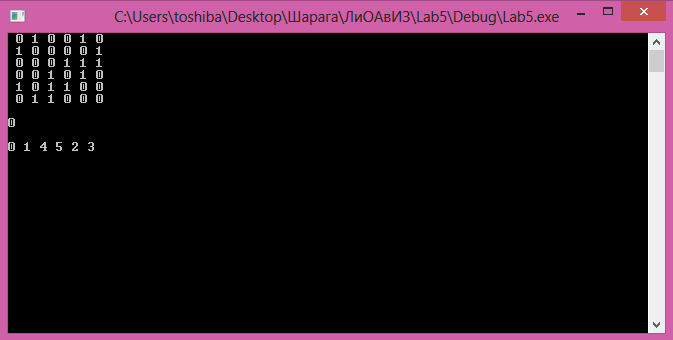
Wit(i);

getch();

return 0;

}

**Задание:**



**Вывод:** Я благополучно написал и использовал программу производящую обход матрицы в ширину.