Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе № 4

по дисциплине: "Логика и основы алгоритмизации в инженерных задачах"

на тему: "Обход графа в глубину"

Выполнили:

студенты группы 20ВВ2

Макарова А. Ю.

Тельнова А. Д.

Принял:

д.т.н., профессор

Митрохин М. А.

к.т.н., доцент

Юрова О. В.

Пенза, 2021

**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 глубину, реализованную в соответствии с приведенным выше описанием.

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

**Описание работы программы:**

**1-2.** Объявляются переменные int N – количество вершин графа, visit – массив для хранения информации о посещенных вершинах, G – матрица смежности графа G.

Количество вершин графа вводится с клавиатуры. Затем все N элементов массива visit инициализируются нулями. С помощью генератора случайных чисел заполняется значениями массив G[N][N]. Он представляет собой симметричную матрицу, элементы главной диагонали которой – нули.

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

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

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

srand(time(NULL));

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

for (int j = i; j < N; j++)

if (i == j)

G[i][j] = 0;

else

{

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

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

}

Полученная матрица выводится на экран.

Вводится номер вершины num, с которой следует начать обход графа в глубину. Рекурсивная функция DFS получает значение num-1, так как индексация элементов массива начинается с 0.

void DFS(int num)

{

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

visit[num] = 1;

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

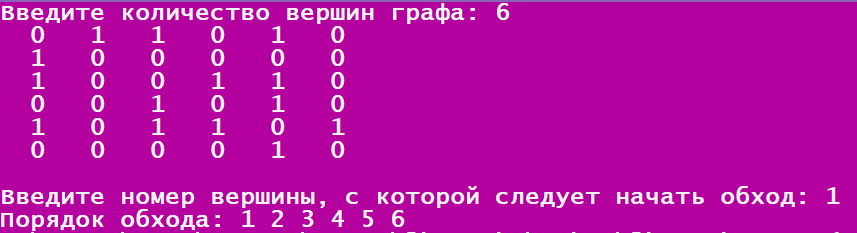
if (G[num][i] == 1 && visit[i] == 0)

DFS(i);

}

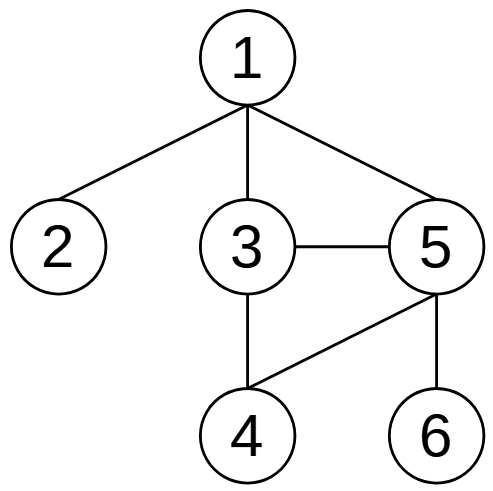
Эта функция реализует обход графа в глубину. Полученный на вход номер вершины выводится на экран, элемент в массиве visit с соответствующим индексом получает значение 1, т. е. вершина отмечается как посещенная. Затем производится поиск не посещенных вершин, смежных текущей: если в строке матрицы с номером текущей вершины находится 1 и элемент массива visit с индексом равным индексу столбца, где нашлась 1, равен 0, найденная вершина становится текущей, а её номер передается функции DFS.

**Результаты работы программы:**



Проверка:

Сгенерированной матрице смежности соответствует граф следующего вида:



Порядок обхода в глубину такого графа: 1 – 2 – 3 – 4 – 5 – 6.Этот результат соответствует результату работы программы.

**3.\*** Перед началом обхода в глубину графа, представленного списками смежности, необходимо обнулить массив visit и создать списки смежности.

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

visit[i] = 0;

Для преобразования представления графа из матрицы смежности в списки смежности используется функция CreateList.

void CreateList(int\*\* matrix, Node\*\* head, int N)

{

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

{

AddFirst(&head[i], i);

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

if (matrix[i][j] == 1)

AddLast(head[i], j);

}

}

С помощью функции AddFirst добавляются элементы в начало списка. После добавления очередного элемента в список, проверяется условие на наличие связи между вершинами графа с индексами i и j. Если эта связь есть, то в конец списка смежности i-ой вершины, добавляется вершина j с помощью функции AddLast.

Вводится номер вершины num, с которой следует начать обход графа в глубину. Рекурсивная функция DFSlist реализует обход графа в глубину.

void DFSlist(Node\*\* list, int num)

{

printf("%2d ", num + 1);

visit[num] = 1;

Node\* tmp\_node = list[num];

while (tmp\_node)   
{

if (visit[tmp\_node->vertex] == 0)

DFSlist(list, tmp\_node->vertex);

tmp\_node = tmp\_node->next;

}

}

Полученный на вход номер вершины выводится на экран, элемент в массиве visit с соответствующим индексом получает значение 1, т. е. вершина отмечается как посещенная. Затем производится поиск вершин, смежных с текущей. Если вершина не посещена, передаем ее адрес и номер, вызовом DFSlist делаем ее текущей. Если же вершина посещена, просматриваем следующую вершину.

Порядок обхода графа в глубину выводится на экран.

**Задание 2\*:**

1. Для матричной формы представления графов выполните преобразование рекурсивной реализации обхода графа к не рекурсивной.

**Описание работы программы:**

Перед началом обхода в глубину с помощью стека графа, представленного матрицей смежности, необходимо обнулить массив visit.

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

visit[i] = 0;

Вводится номер вершины num, с которой следует начать обход графа в глубину. Функция DFSstack получает значение num-1, так как индексация элементов массива начинается с 0 и реализует обход графа в глубину при помощи стека.

void DFSstack(int num)

{

stack <int> stack;

stack.push(num);

while (!stack.empty())  
{

if (visit[[stack.top](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fstack.top" \t "_blank)()] == 0)

{

visit[[stack.top](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fstack.top" \t "_blank)()] = 1;

printf("%d ", [stack.top](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fstack.top" \t "_blank)() + 1);

num = [stack.top](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fstack.top" \t "_blank)();

stack.pop();

for (int i = N - 1; i >= 0; i--)

if (G[num][i] == 1 && visit[i] == 0)

stack.push(i);

}

else

stack.pop();

}

}

На вершину стека кладется вершина с номером num. Пока стек не пуст, проверяем массив visit на наличие не посещенных вершин. Если вершина не посещена, то элемент в массиве visit с соответствующим индексом получает значение 1, т. е. вершина отмечается как посещенная. Переменной num присваивается значение верхнего элемента стека. Посещенная вершина извлекается из стека. Условие i = N - 1; i >= 0; i-- позволяет выводить на экран список посещенных вершин в прямом порядке. Если найдена смежная с текущей вершиной не посещенная вершина, то она добавляется в стек. Когда все вершины будут отмечены, как посещенные, стек очищается.

Полученный порядок обхода графа в глубину выводится на экран.

**Вывод:** Написали программу, осуществляющую генерацию матрицы смежности неориентированного графа с использованием генератора случайных чисел. Для графа, представленного матрицей смежности, а также списками смежности, реализовали процедуру обхода в глубину. Для матричной формы представления графов выполнили преобразование рекурсивной реализации обхода графа к не рекурсивной.