Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Системы автоматизированного проектирования»

**Отчет**

По лабораторной работе №4

По курсу «Логика и основы алгоритмизации в инженерных задачах»

На тему «Обход графа в глубину»

Выполнил студент гр.20ВВ3

Мирясов Н. А.

Проверили:

Митрохин М. А.

Юрова О. В.

Пенза, 2021

**Цель:** научиться реализовывать алгоритм обхода графа в глубину, представленного матрицей и списком смежности.

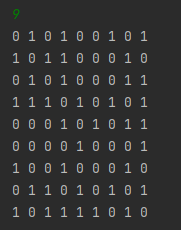
**Задание 1:**

1. Сгенерировали (используя генератор случайных чисел) матрицу смежности для неориентированного графа. Вывели матрицу на экран.

static void rand(int n, int[][] matrix){  
 for(int i = 0; i < n; i++){  
 for(int j = i; j < n; j++){  
 if(i == j)  
 matrix[i][j] = 0;  
 else{  
 matrix[i][j] = (int)(( Math.*random*() \* 2) + 0);  
 matrix[j][i] = matrix[i][j];  
 }  
 }  
 }  
  
 for(int i = 0; i < n; i++){  
 for(int j = 0; j < n; j++){  
 System.*out*.print(matrix[i][j] + " ");  
 }  
 System.*out*.println();  
 }  
}

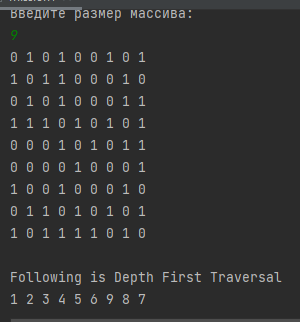
В matrix содержится созданная матрица. В n я храню размер матрицы, задаваемый с клавиатуры. Прохожу два цикла, заполняя граф. После чего повторяю процедуру, выводя на экран инициализированную матрицу.

Получаю такой результат:



1. Для сгенерированного графа провожу процедуру обхода в глубину, реализованную в соответствии с указаниями в задании.

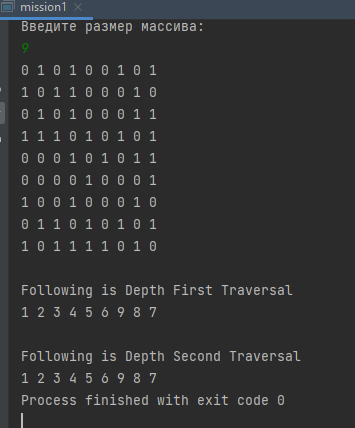
public static void DFS1(int[][] matrix, boolean[] visited,int n, int i){  
 System.*out*.print((i + 1) + " ");  
 visited[i]= true;  
 for (int j = 0; j<n;j++){  
 if(!(visited[j]) && matrix[i][j]==1){  
 *DFS1*(matrix, visited, n, j);  
 }  
 }



1. Реализовал процедуру обхода в глубину для графа, представленного списками смежности.

private int numVertices;  
private LinkedList<Integer> adjLists[];  
private boolean visited[];  
  
mission1(int vertices)  
{  
 numVertices = vertices;  
 adjLists = new LinkedList[vertices];  
 visited = new boolean[vertices];  
  
 for (int i = 0; i < vertices; i++)  
 adjLists[i] = new LinkedList<Integer>();  
}  
  
void addEdge(int src, int dest)  
{  
 adjLists[src].add(dest);  
}  
  
  
void DFS(int vertex)  
{  
 visited[vertex] = true;  
 System.*out*.print(vertex + 1 + " ");  
  
 ListIterator<Integer> ite = adjLists[vertex].listIterator();  
  
 while (ite.hasPrevious()) {  
 int adj = ite.previous();  
 if (!visited[adj])  
 DFS(adj);  
 }  
  
 while (ite.hasNext()) {  
 int adj = ite.next();  
 if (!visited[adj])  
 DFS(adj);  
 }  
}

System.*out*.println("Введите размер массива:");  
Scanner in = new Scanner(System.*in*);  
int n = in.nextInt();  
int[][] matrix = new int[n][n];  
boolean[] visited = new boolean[n];  
*rand*(n, matrix);  
  
for (int i = 0; i < n; i++){  
visited[i] = false;  
}



**Задание 2:**

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

public static void DFS1(int[][] matrix, boolean[] visited,int n, int i, int p){  
 p++;  
 System.*out*.print((i + 1) + " ");  
 visited[i]= true;  
 for (int j = 0; j<n;j++){  
 if(!(visited[j]) && matrix[i][j]==1){  
 if (p == 5)  
 *DFS2*(matrix, visited, n, j);  
 *DFS1*(matrix, visited, n, j, p);  
 }  
 }  
}  
  
public static void DFS2(int[][] matrix, boolean[] visited,int n, int i){  
 int s = 0;  
 for (int j = 0; j < n; j++){  
 for (int p = 0; p < n; p++){  
 if(!(visited[j]) && matrix[i][j]==1) {  
 s++;  
 if (s > n\*n) {  
 System.*out*.print((i + 1) + " ");  
 visited[i] = true;  
 }  
 }  
 }  
 }  
}

При выполнении определенного кол-ва рекурсивных обходов матрицы, вызывается функция, которая заканчивает обход матрицы, но уже без рекурсии.

Получаю такой результат:

