NAME: Soham Phadke  
CLASS: TY\_CS\_D

BATCH: 2

ROLL NO: 37

PRN NO: 12111222

**ASSIGNMENT 2**

**Problem Statement:** Uninformed search: Breadth first search and Depth first search

**Code:**

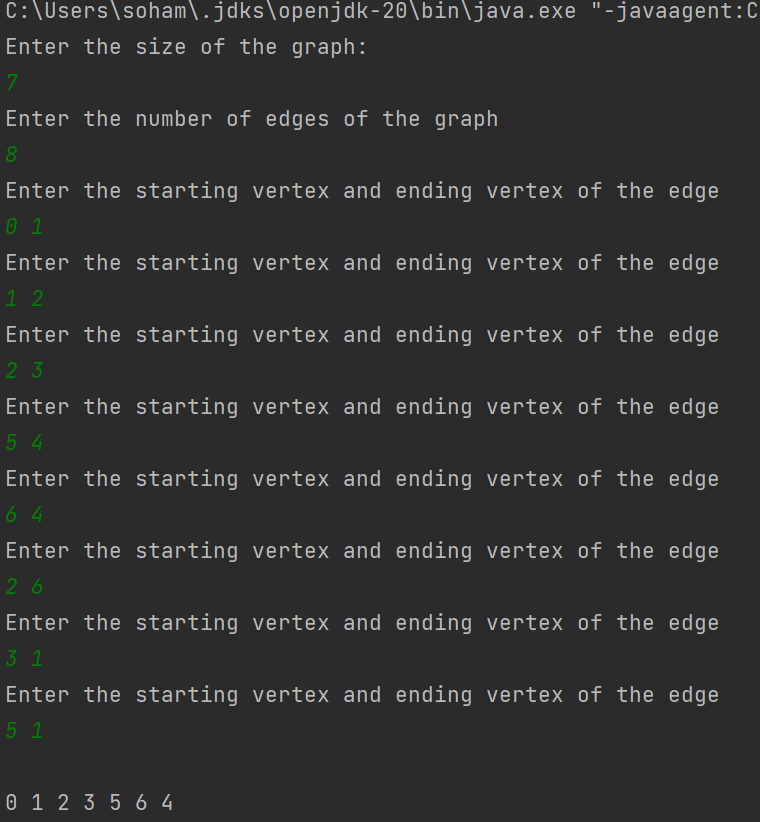
**BFS:**

import java.util.\*;  
  
class Graph{  
 private int val;  
 private LinkedList<Integer> adj[];  
  
 public LinkedList<Integer>[] getAdj() {  
 return adj;  
 }  
  
 Graph(int size){  
 val = size;  
 adj = new LinkedList[val];  
  
 for (int i = 0; i < val; i++) {  
 adj[i] = new LinkedList<>();  
 }  
  
 }  
  
 public void addEdge(int s, int e){  
 adj[s].add(e);  
 adj[e].add(s);  
 }  
}  
  
public class BFS {  
  
 private static ArrayList<Integer> *bfs*;  
 public static void bfsTraversal(LinkedList<Integer> adj[], int size){  
 LinkedList<Integer> q = new LinkedList<>();  
 boolean[] visited = new boolean[size];  
  
 visited[0] = true;  
 q.add(0);  
 *bfs* = new ArrayList<>();  
  
 while (q.size() != 0){  
 int s = q.poll();  
 *bfs*.add(s);  
 for(Integer num:adj[s]){  
 if(visited[num] == false){  
 visited[num] = true;  
 q.add(num);  
 }  
 }  
 }  
  
 System.*out*.println();  
 for(Integer num:*bfs*){  
 System.*out*.print(num + " ");  
 }  
 }  
  
 public static void main(String[] args) {  
 Scanner scan = new Scanner(System.*in*);  
 System.*out*.println("Enter the size of the graph: ");  
 int size = scan.nextInt();  
 Graph g = new Graph(size);  
 int edgeSize=0, s=0,e=0;  
 System.*out*.println("Enter the number of edges of the graph");  
 edgeSize = scan.nextInt();  
 for (int i = 0; i < edgeSize; i++) {  
 System.*out*.println("Enter the starting vertex and ending vertex of the edge");  
 s=scan.nextInt();  
 e=scan.nextInt();  
 g.addEdge(s,e);  
 }  
  
 LinkedList<Integer> adj[] = g.getAdj();  
  
 *bfsTraversal*(adj, size);  
 }  
}

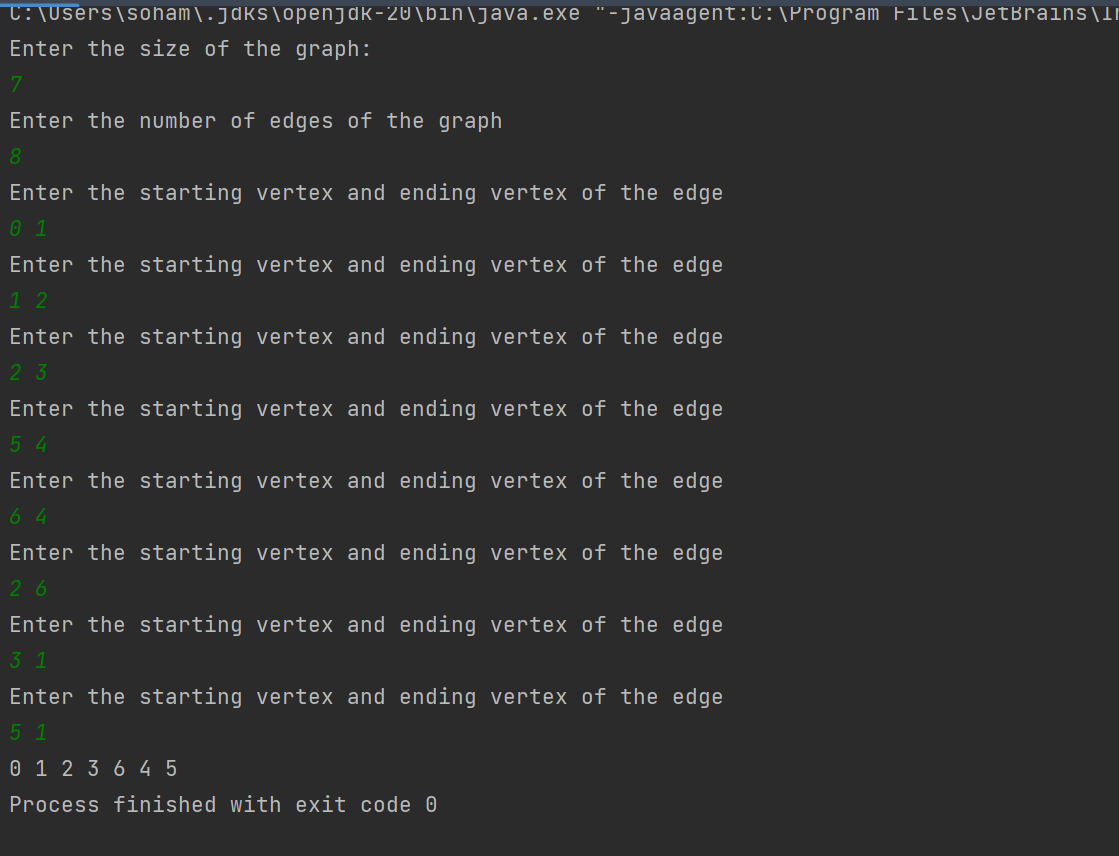
**DFS:**

import java.util.\*;  
  
public class DFS {  
 private static ArrayList<Integer> *bfs*;  
 static void dfsTraversal(int v, boolean visited[], LinkedList<Integer> adj[])  
 {  
 // Mark the current node as visited and print it  
 visited[v] = true;  
 System.*out*.print(v + " ");  
  
 // Recur for all the vertices adjacent to this  
 // vertex  
  
 for(Integer num:adj[v]){  
 if(!visited[num])  
 *dfsTraversal*(num, visited, adj);  
 }  
 }  
  
 public static void main(String[] args) {  
 Scanner scan = new Scanner(System.*in*);  
 System.*out*.println("Enter the size of the graph: ");  
 int size = scan.nextInt();  
 Graph g = new Graph(size);  
 int edgeSize=0, s=0,e=0;  
 System.*out*.println("Enter the number of edges of the graph");  
 edgeSize = scan.nextInt();  
 for (int i = 0; i < edgeSize; i++) {  
 System.*out*.println("Enter the starting vertex and ending vertex of the edge");  
 s=scan.nextInt();  
 e=scan.nextInt();  
 g.addEdge(s,e);  
 }  
  
 LinkedList<Integer> adj[] = g.getAdj();  
 boolean[] visited = new boolean[size];  
  
 *dfsTraversal*(0, visited, adj);  
 }  
}

**Output:  
BFS:**

****

**DFS:**

****