|  |
| --- |
| import java.util.\*; |
| import java.io.\*; |
| class Graph { |
| ArrayList<ArrayList<Integer>>adjList=new ArrayList<>(); |
|  |
| Graph(int V){ //creating a constructor |
| for(int i=0;i<V;i++){ |
| adjList.add(new ArrayList<Integer>()); //Number of vertices are initialized in constructor |
| } |
| } |
| public void addEdge(int u,int v){ |
| adjList.get(u).add(v); |
| adjList.get(v).add(u); |
| } |
| public void printAdjList(){ |
| for(int i=0;i<adjList.size();i++){ //i-vertex number |
| System.out.println("Adjacency list of vertex :"+i); //print just i th vertex numbers by looping |
| for(int j=0;j<adjList.get(i).size();j++){ //we are looping with j for checking what are the adjacency vertices present inside each arraylist of vertices |
| System.out.println(adjList.get(i).get(j)); //and printing it out |
| } |
| } |
| } |
| public void depthFirstSearch(int V){ //traversal start node is V |
| int v=adjList.size(); //total number of vertices |
| boolean []visited=new boolean[v]; |
| dfs(V,visited); |
| System.out.println(); |
| } |
| public void dfs(int v, boolean []visited){ //traversal start node is v |
| visited[v]=true; |
| System.out.print(v+" "); |
| for(int i=0;i<adjList.get(v).size();i++){ |
| int av=adjList.get(v).get(i); |
| if(!visited[av]) |
| dfs(av,visited); |
| } |
|  |
| } |
| public void bFS(int V){ //traversal start node is V |
| int v=adjList.size(); //total number of vertices |
| boolean []visited=new boolean[v]; |
| visited[V]=true; |
| Queue<Integer> q=new LinkedList<>();//Queue is an interface. we can't instantiate. so, new linkedlist has to be given |
| q.add(V); |
| while(q.size()!=0){ //dequeue one by one |
| int vertex=q.remove(); |
| System.out.print(vertex+" "); |
| for(int i=0;i<adjList.get(vertex).size();i++){ |
| int av=adjList.get(vertex).get(i); //getting vertex th adjacency list and getting ith index of that list. |
| if(!visited[av]){ //if not visited of array's av th index |
| q.add(av); |
| visited[av]=true; |
| } |
| } |
| } |
| System.out.println(); |
| } |
| } |
| public class GraphRepresentation{ //public class required to execute |
| public static void main(String args[]) { |
| Graph g=new Graph(5); //No. of vertices are 5. |
| //representing a graph in adjacency matrix. it is easy |
| // int [][]graph={{1,0,0,1,0} |
| // {0,0,1,1,0} |
| // {1,0,0,1,1} |
| // {1,1,0,1,1} |
| // {1,1,0,1,1} }; |
| //representing a graph in adjacency list method. |
| g.addEdge(0,1); |
| g.addEdge(2,3); |
| g.addEdge(1,2); |
| g.addEdge(3,4); |
| g.addEdge(0,4); |
| g.addEdge(1,4); |
| g.printAdjList(); |
| System.out.print("BFS order: "); |
| g.bFS(0);//traversal start node is 0 |
| System.out.print("DFS order: "); |
| g.depthFirstSearch(0); |
| } |
| } |