Artificial Intelligence

Asssignment 1 B

Name : Swati Anil Sonone

Div : CS\_D

Roll no. : 53

Problem Statement : Bsf and Dsf

Code :

import java.util.\*;  
  
public class BFS {  
 private static Map<Integer, List<Integer>> *Adjecency\_list* ;  
 public BFS(){  
 *Adjecency\_list* = new HashMap<>();  
 }  
 public void addNode(int node){  
 *Adjecency\_list*.put(node,new ArrayList<>());  
 }  
 public void addEdge(int f,int t){  
 *Adjecency\_list*.get(f).add(t);  
 *Adjecency\_list*.get(t).add(f);  
 }  
 public List<Integer> getNeighbour(int node){  
 return *Adjecency\_list*.get(node);  
 }  
 public static void bfs(int startNode){  
 LinkedList<Integer> q = new LinkedList<>();  
 LinkedList<Integer> visited = new LinkedList<>();  
 q.add(startNode);  
 visited.add(startNode);  
 while(!q.isEmpty()){  
 int current\_node = q.poll();  
 System.*out*.print(current\_node+" ");  
 for(int i: *Adjecency\_list*.get(current\_node)){  
 if(!visited.contains(i)){  
 q.add(i);  
 visited.add(i);  
 }  
 }  
 }  
 }  
 private static void dfs(int node, Set<Integer> visited) {  
 if (!visited.contains(node)) {  
 System.*out*.print(node + " ");  
 visited.add(node);  
 List<Integer> neighbors = *Adjecency\_list*.get(node);  
 if (neighbors != null) {  
 for (int neighbor : neighbors) {  
 *dfs*(neighbor, visited);  
 }  
 }  
 }  
 }  
  
 static void dfsTraversal(int startNode) {  
 Set<Integer> visited = new HashSet<>();  
 *dfs*(startNode, visited);  
 }  
 public static void main(String[] arg){  
 BFS B = new BFS();  
 int[] Array\_list = new int[]{1,3,2,5,7,8,9,4};  
 for(int i:Array\_list){  
 B.addNode(i);  
 }  
 B.addEdge(1,3);  
 B.addEdge(1,7);  
 B.addEdge(3,2);  
 B.addEdge(3,7);  
 B.addEdge(3,8);  
 B.addEdge(2,5);  
 B.addEdge(7,4);  
 B.addEdge(8,9);  
 B.addEdge(9,7);  
  
 for(int node:Array\_list){  
 System.*out*.println("Neighbour of "+ node +" : ");  
 List<Integer> nighbours = B.getNeighbour(node);  
 for(int i:nighbours){  
 System.*out*.println(i);  
 }  
 }  
 *bfs*(1);  
 System.*out*.println();  
 *dfsTraversal*(1);  
 }  
}

Output

