DATA STRUCTURES AND ALGORITHMS

LAB ASSIGNMENT-6

NAME – KAPAROTU VENKATA SURYA THARANI

COURSE – AIDE

SECTION - “A”

USN ID – 22BTRAD018

Write a JAVA program to implement Depth First Search and Breadth First Search.

# Java program to implement Depth First Search: Code (Recursive):

**package** p1;

**import** java.io.\*; **import** java.util.\*; **public class** DFS\_R {

## private int V;

**private** LinkedList<Integer> x[]; @SuppressWarnings("unchecked") DFS\_R(**int** v)

{

V = v;

x = **new** LinkedList[v];

**for** (**int** i = 0; i < v; ++i)

x[i] = **new** LinkedList();

}

**void** addEdge(**int** v, **int** w)

{

x[v].add(w);

}

**void** DFSUtil(**int** vertex, **boolean** nodes[])

{

nodes[vertex] = **true**; System.***out***.print(vertex + " "); **int** a = 0;

**for** (**int** i = 0; i < x[vertex].size(); i++)

{

a = x[vertex].get(i);

**if** (!nodes[a])

{

DFSUtil(a, nodes);

}

}

}

**void** DFS(**int** v)

{

**boolean** already[] = **new boolean**[V]; DFSUtil(v, already);

}

**public static void** main(String args[])

{

DFS\_R g = **new** DFS\_R(6);

g.addEdge(0, 1);

g.addEdge(0, 2);

g.addEdge(1, 0);

g.addEdge(1, 3);

g.addEdge(2, 0);

g.addEdge(2, 3);

g.addEdge(3, 4);

g.addEdge(3, 5);

g.addEdge(4, 3);

g.addEdge(5, 3);

System.***out***.println(

"Following is Depth First Traversal: ");

g.DFS(0);

}

}

**Output:**



**Code (Non-Recursive):**

**package** p1;

**import** java.util.\*;

**public class** DFS\_NR {

**int** V; LinkedList<Integer>[] x;

DFS\_NR(**int** V)

{

**this**.V = V;

x = **new** LinkedList[V];

**for** (**int** i = 0; i < x.length; i++) x[i] = **new** LinkedList<Integer>();

}

**void** addEdge(**int** v, **int** w)

{

x[v].add(w);

}

**void** DFS(**int** n)

{

**boolean** nodes[] = **new boolean**[V]; Stack<Integer> stack = **new** Stack<>();

stack.push(n);

**int** a = 0;

**while**(!stack.empty())

{

n = stack.peek(); stack.pop();

**if**(nodes[n] == **false**)

{

System.***out***.print(n + " "); nodes[n] = **true**;

}

**for** (**int** i = 0; i < x[n].size(); i++)

{

a = x[n].get(i);

**if** (!nodes[a])

{

stack.push(a);

}

}

}

}

**public static void** main(String[] args)

{

DFS\_NR g1 = **new** DFS\_NR(6);

g1.addEdge(0, 1);

g1.addEdge(0, 2);

g1.addEdge(1, 0);

g1.addEdge(1, 3);

g1.addEdge(2, 0);

g1.addEdge(2, 3);

g1.addEdge(3, 4);

g1.addEdge(3, 5);

g1.addEdge(4, 3);

g1.addEdge(5, 3);

System.***out***.println("Following is the Depth First Traversal (Without recursion):");

g1.DFS(0);

}

}

**Output:**



**Java program to implement Breadth First Search: Code:**

**package** p1;

**import** java.io.\*;

**import** java.util.\*;

**public class** BFS {

## private int V;

**private** LinkedList<Integer> x[];

**private** Queue<Integer> queue;

BFS(**int** v)

{

V = v;

x = **new** LinkedList[v]; **for** (**int** i=0; i<v; i++)

{

x[i] = **new** LinkedList<>();

}

queue = **new** LinkedList<Integer>();

}

**void** addEdge(**int** v,**int** w)

{

x[v].add(w);

}

**void** BFS(**int** n)

{

**boolean** nodes[] = **new boolean**[V]; **int** a = 0;

nodes[n]=**true**; queue.add(n);

**while** (queue.size() != 0)

{

n = queue.poll(); System.***out***.print(n+" ");

**for** (**int** i = 0; i < x[n].size(); i++)

{

a = x[n].get(i);

**if** (!nodes[a])

{

nodes[a] = **true**; queue.add(a);

}

}

}

}

**public static void** main(String args[])

{

BFS g1 = **new** BFS(6);

g1.addEdge(0, 1);

g1.addEdge(0, 3);

g1.addEdge(0, 4);

g1.addEdge(4, 5);

g1.addEdge(3, 5);

g1.addEdge(1, 2);

g1.addEdge(1, 0);

g1.addEdge(2, 1);

g1.addEdge(4, 1);

g1.addEdge(3, 1);

g1.addEdge(5, 4);

g1.addEdge(5, 3);

System.***out***.println("The Breadth First Traversal of the graph is as follows :");

g1.BFS(0);

}

}

**Output:**

