**ASSIGNMENT NO.1**

**Breadth-first Search Algorithm:-**

import java.io.\*;

import java.util.\*;

public class BFSTraversal

{

private int node; /\* total number number of nodes in the graph \*/

private LinkedList<Integer> adj[]; /\* adjacency list \*/

private Queue<Integer> que; /\* maintaining a queue \*/

BFSTraversal(int v)

{

node = v;

adj = new LinkedList[node];

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

{

adj[i] = new LinkedList<>();

}

que = new LinkedList<Integer>();

}

void insertEdge(int v,int w)

{

adj[v].add(w); /\* adding an edge to the adjacency list (edges are bidirectional in this example) \*/

}

void BFS(int n)

{

boolean nodes[] = new boolean[node]; /\* initialize boolean array for holding the data \*/

int a = 0;

nodes[n]=true;

que.add(n); /\* root node is added to the top of the queue \*/

while (que.size() != 0)

{

n = que.poll(); /\* remove the top element of the queue \*/

System.out.print(n+" "); /\* print the top element of the queue \*/

for (int i = 0; i < adj[n].size(); i++) /\* iterate through the linked list and push all neighbors into queue \*/

{

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

if (!nodes[a]) /\* only insert nodes into queue if they have not been explored already \*/

{

nodes[a] = true;

que.add(a);

}

}

}

}

public static void main(String args[])

{

BFSTraversal graph = new BFSTraversal(6);

graph.insertEdge(0, 1);

graph.insertEdge(0, 3);

graph.insertEdge(0, 4);

graph.insertEdge(4, 5);

graph.insertEdge(3, 5);

graph.insertEdge(1, 2);

graph.insertEdge(1, 0);

graph.insertEdge(2, 1);

graph.insertEdge(4, 1);

graph.insertEdge(3, 1);

graph.insertEdge(5, 4);

graph.insertEdge(5, 3);

System.out.println("Breadth First Traversal for the graph is:");

graph.BFS(0);

}

}

**OUTPUT:-**

Breadth First Traversal for the graph is:

0 1 3 4 2 5