Data Structures and Algorithms

CSE2001

Lab - 7 - Assignment - 1

Yashwanth Reddy
19BCE7362
Date- 20thJuly2021

Problem : Write a Program to check the connectivity of a graph using BFS

Code

```
import java.util.LinkedList;
import java.util.Queue;
import java.util.Scanner;

public class Connectivity_BFS
{
    private final int      vertices;
    private int[][]        adjacency_matrix;
    private Queue<Integer> queue;

    public Connectivity_BFS(int v)
    {
        vertices = v;
        adjacency_matrix = new int[vertices + 1][vertices + 1];
        queue = new LinkedList<Integer>();
    }
```

```java
public void makeEdge(int to, int from, int edge)
{
    try
    {
        adjacency_matrix[to][from] = edge;
        adjacency_matrix[from][to] = edge;
    } catch (ArrayIndexOutOfBoundsException index)
    {
        System.out.println("The vertices does not exists");
    }
}

public int getEdge(int to, int from)
{
    try
    {
        return adjacency_matrix[to][from];
    } catch (ArrayIndexOutOfBoundsException index)
    {
        System.out.println("The vertices does not exists");
    }
    return -1;
}

public void bfs(int source)
{
    int number_of_nodes = adjacency_matrix[source].length - 1;
    int[] visited = new int[number_of_nodes + 1];
    int i, element;
    visited[source] = 1;
```

```
    queue.add(source);
    while (!queue.isEmpty())
    {
        element = queue.remove();
        i = 1;// element;
        while (i <= number_of_nodes)
        {
            if (adjacency_matrix[element][i] == 1 && visited[i] == 0)
            {
                queue.add(i);
                visited[i] = 1;
            }
            i++;
        }
    }

    System.out.print("The source node " + source + " is connected to: ");
    int count = 0;
    for (int v = 1; v <= number_of_nodes; v++)
        if (visited[v] == 1)
        {
            System.out.print(v + " ");
            count++;
        }

    if (count == number_of_nodes)
        System.out.print("\nThe Graph is Connected ");
    else
        System.out.print("\nThe Graph is Disconnected ");
}
```

```java
public static void main(String args[])
{
    int v, e, count = 1, to = 0, from = 0;
    Scanner sc = new Scanner(System.in);
    Connectivity_BFS graph;
    System.out.println("The Undirected Graph Connectivity Test");
    try
    {
        System.out.println("Enter the number of vertices: ");
        v = sc.nextInt();
        System.out.println("Enter the number of edges: ");
        e = sc.nextInt();

        graph = new Connectivity_BFS(v);

        System.out.println("Enter the edges: <to> <from>");
        while (count <= e)
        {
            to = sc.nextInt();
            from = sc.nextInt();

            graph.makeEdge(to, from, 1);
            count++;
        }
        System.out.println("The adjacency matrix for the given graph is: ");
        System.out.print("  ");
        for (int i = 1; i <= v; i++)
            System.out.print(i + " ");
        System.out.println();

        for (int i = 1; i <= v; i++)
```

```
        {
            System.out.print(i + " ");
            for (int j = 1; j <= v; j++)
                System.out.print(graph.getEdge(i, j) + " ");
            System.out.println();
        }
        System.out.println("Enter the Source Node: ");
        int sourceNode = sc.nextInt();
        graph.bfs(sourceNode);

    } catch (Exception E)
    {
        System.out.println("Something went wrong");
    }
    sc.close();
  }
}
```

Output

```
C:\WINDOWS\system32\cmd.exe

C:\Users\yashw\Desktop\Summer\Labs>java Connectivity_BFS
The Undirected Graph Connectivity Test
Enter the number of vertices:
4
Enter the number of edges:
2
Enter the edges: <to> <from>
1 2
1 4
The adjacency matrix for the given graph is:
  1 2 3 4
1 0 1 0 1
2 1 0 0 0
3 0 0 0 0
4 1 0 0 0
Enter the Source Node:
3
The source node 3 is connected to: 3
The Graph is Disconnected
C:\Users\yashw\Desktop\Summer\Labs>
```