Programming Assignment 3

-Helena Julie Arpudaraj

UCF ID - 4735186

Question: Chapter 3 Question 10

10. A number of art museums around the country have been featuring work by an artist named Mark Lombardi (1951–2000), consisting of a set of intricately rendered graphs. Building on a great deal of research, these graphs encode the relationships among people involved in major political scandals over the past several decades: the nodes correspond to participants, and each edge indicates some type of relationship between a pair of participants. And so, if you peer closely enough at the drawings, you can trace out ominous-looking paths from a high-ranking U.S. government official, to a former business partner, to a bank in Switzerland, to a shadowy arms dealer.

Such pictures form striking examples of *social networks*, which, as we discussed in Section 3.1, have nodes representing people and organizations, and edges representing relationships of various kinds. And the short paths that abound in these networks have attracted considerable attention recently, as people ponder what they mean. In the case of Mark Lombardi's graphs, they hint at the short set of steps that can carry you from the reputable to the disreputable.

Of course, a single, spurious short path between nodes v and w in such a network may be more coincidental than anything else; a large number of short paths between v and w can be much more convincing. So in addition to the problem of computing a single shortest v-w path in a graph G, social networks researchers have looked at the problem of determining the *number* of shortest v-w paths.

This turns out to be a problem that can be solved efficiently. Suppose we are given an undirected graph $G = (V, E)$, and we identify two nodes v and w in G. Give an algorithm that computes the number of shortest v-w paths in G. (The algorithm should not list all the paths; just the number suffices.) The running time of your algorithm should be $O(m + n)$ for a graph with n nodes and m edges.

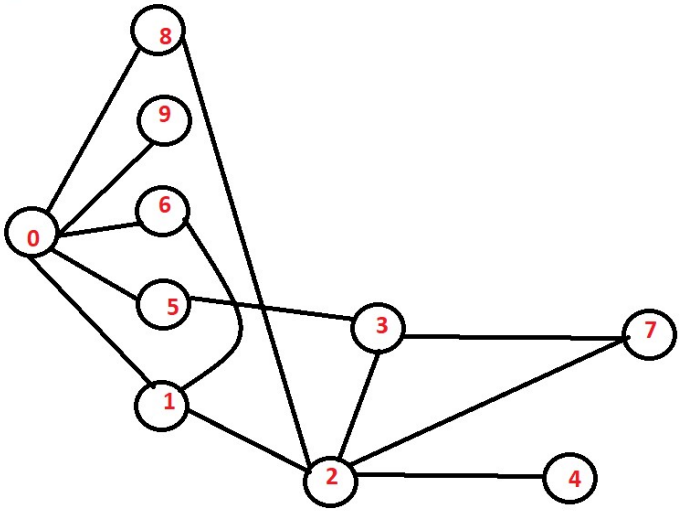
Correctness Validation : Test Cases with Brute Force implementation (screenshots are from the program output for brute force)

- Test Cases for a random network of 10 nodes:

```
The list of adjacent nodes to node  0  is
Node  1
Node  5
Node  6
Node  9
Node  8
The list of adjacent nodes to node  1  is
Node  0
Node  2
Node  6
The list of adjacent nodes to node  2  is
Node  1
Node  3
Node  4
Node  7
Node  8
The list of adjacent nodes to node  3  is
Node  2
Node  7
Node  5
The list of adjacent nodes to node  4  is
Node  2
The list of adjacent nodes to node  5  is
Node  0
Node  3
The list of adjacent nodes to node  6  is
Node  0
Node  1
The list of adjacent nodes to node  7  is
Node  2
Node  3
The list of adjacent nodes to node  8  is
Node  2
Node  0
The list of adjacent nodes to node  9  is
Node  0
```

Output screen shot:-

```
Validation for correctness for nodes 0-9 using brute force
Number of shortest paths from node  0  to node  1  is:  1
Number of shortest paths from node  0  to node  2  is:  2
Number of shortest paths from node  0  to node  3  is:  1
Number of shortest paths from node  0  to node  4  is:  2
Number of shortest paths from node  0  to node  5  is:  1
Number of shortest paths from node  0  to node  6  is:  1
Number of shortest paths from node  0  to node  7  is:  3
Number of shortest paths from node  0  to node  8  is:  1
Number of shortest paths from node  0  to node  9  is:  1
```



Pseudo-code for bfs:

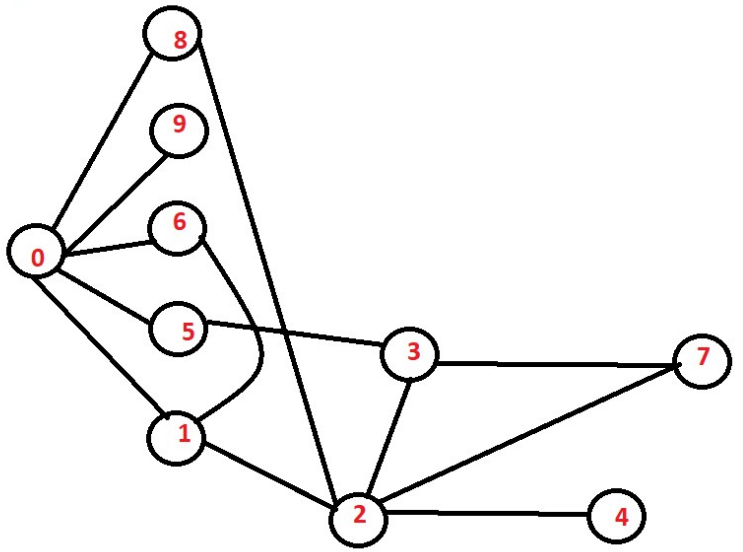
```
Bfs(startNode,endNode
queue=[]
queue.enq(startNode)
startNode.visited=True
startNode.dist=0
startNode.Dist=[0]
while(queue!=empty)
        v=queue.deq()
        for n in v.adjacentnodes
                n.dist=v.dist+1
                n.Dist.enq(n.dist)
                if n.visited=False
                        n.visited=True
                        queue.enq(n)
                if n.dist>0
                        n.dist=min(n.Dist)
shortestlength=min(endNode.Dist)
numOfShortestPaths=endNode.Dist.count(shortestlength)
return numOfShortestPaths
```

Correctness Validation : Test Cases with BFS implementation (screenshots are from the program output for bfs)

- Test Cases for a random network of 10 node

Output screen shot:

```
Validation for correctness for nodes 0-9 using bfs
shortest length from node  0  to  1  =  1
Number of shortest paths from node  0  to node  1  is:  1
shortest length from node  0  to  2  =  2
Number of shortest paths from node  0  to node  2  is:  2
shortest length from node  0  to  3  =  2
Number of shortest paths from node  0  to node  3  is:  1
shortest length from node  0  to  4  =  3
Number of shortest paths from node  0  to node  4  is:  2
shortest length from node  0  to  5  =  1
Number of shortest paths from node  0  to node  5  is:  1
shortest length from node  0  to  6  =  1
Number of shortest paths from node  0  to node  6  is:  1
shortest length from node  0  to  7  =  3
Number of shortest paths from node  0  to node  7  is:  3
shortest length from node  0  to  8  =  1
Number of shortest paths from node  0  to node  8  is:  1
shortest length from node  0  to  9  =  1
Number of shortest paths from node  0  to node  9  is:  1
```



Performance Validation(Time Complexity is O(n+m))