IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 11, November 2015

A Polynomial Time Algorithm to Determine Singly Connectivity in Directed Graph

Ishwar Baidari[1], Rashmi Gangadhar[2]

Associate Professor, Dept. of Computer Science, Karnataka University, Dharwad, India[1]

Assistant Professor, Dr D Y Patil Institute of MCA, Akurdi, Pune, Maharashtra, India[2]

Abstract: In this paper, we consider the problem of determining whether or not a directed graph is singly connectedi.e. a directed graph is singly connected if for any pair of vertices there exists at most one simple path connecting them and also undirected graph is singly connected if and only if it is a tree. We have given a straight forward implementation of this problem using DFS algorithm theory which takes polynomial time to check it.

Keywords: DFS; source vertex; cyclic; tree; spanning forest.

I. INTRODUCTION

Graph connectivity has become a widely studied property of graph theory. Graph connectivity theories are essentially in computer communication network applications such as routing, transportation networks, separation edges and vertices correspond to singly point of failure in a network etc.,

In this paperwe investigated another connectivity problem: determining if a graph has at most one distinct path between every pair of vertices. In particular, we say a directed graph is singly connected if there existsat most one simple path between every pair of vertices. If there are two distinct simple paths connecting any two vertices, the graph is not singly connected. An undirected graph is singly connected if and only if it is a tree[2].

1.1 Definitions:

We used graph terminology; see e.g.. [1]. In particular given a directed graph $G = (V, E)$ with $n = |v|$ and $m = |E|$. a path connecting two vertices $u, v \in V$ in a sequence of nodes $(u = u_0, u_1 \dots\dots u_l = v)$ such that $(u_i, u_{i+1}) \in E, 0 \le i < l$. A path is a simple path if $u_i \neq u_j \; \forall \, i \neq j$. Two paths $(u_0 \dots\dots\dots\dots\dots u_k)$ and $(u_0 \dots\dots . u_l)$ distance if either (1) $k \neq 1$ or (2) $k = l$ and $u_i \neq v_i$ for some $0 \le i \le l$. A cycle is path $(u_0 \dots\dots . u_l)$ Such that $u_0 = u_l$ a simple cycle is cycle $(u_0 \dots\dots . u_l)$ Such that $u_i \neq u_j, 0 \le i < j < l$ givien some cycle a chord is a path of arcs none of which is on the cycle connecting two distance node at these cycles

1.2 Definition:

A diagraph $= (V, A)$ is singly connected if and only if for any two verticesu, $v \in v$. there exists at most one simple path connection $u\&v$.

II ALGORITHM FOR SINGLY CONNECTIVITY

In this section we designed an algorithm to determine single connectivity. **A directed graph G= (V , E) is singly connected if u ~> v implies that G contains at most one simple path from u to v for all vertices u,v ϵ V. give an efficient algorithm to determine whether or not a directed graph is singly connected.**

A simple algorithm designed for solving the single connectivity of the graph. In this algorithm we have used the concept of DFS algorithm. We have taken U as a source node, from that node we started to traverse each and every other node of graph to check whether there exists single path between source and other node of a graph by making use of count value.

```
SinglyConnected( G )
1.      U = source
2.      First_loop: for each vertex v ϵ G – { U }
3.          Count=0
4.          For each x ϵ G.Adj(U)
5.              For each vertex g ϵ G
6.      g.color = WHITE
7.      U.color = GRAY
8.              Count=SimplePath(U,V,x)
9.          If Count = = 1|| count = = 0
10.     gotofirst_loop;
11.         Otherwise
12.                 Write " not singly connected"
13.                 Exit from process
14.     Write " The given graph is singly connected"
```

```
SimplePath(U ,V, x)
1.      If x = = V
2.          return ++count
3.      x.color=GRAY
4.      for each y ϵ G.Adj(x)
5.          if y.color = = WHITE
6.                  Count = SimplePath(U,V,y)
7.      return count
```

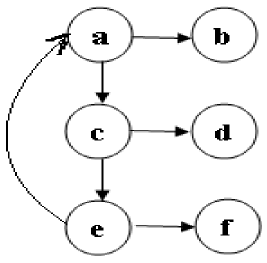
Singly Connected procedure work as follows: In line 1 we assumed U as a source node. In Line 2 considered every node of a graph as destination node except source node U. Lines 3-8 determines number of simple path between the source node U and any other node. Line 3 initializes count to zero, line 4 starts to check the path for each adjacent node of U. Since we need to check the path for every adjacent node of a U, it is necessary to paint all the nodes of a graph as WHITE color which indicates we are not traverses any node of a graphfrom Lines 5-6. As algorithm start to traverse from U, paint GRAY color to U to indicate that node has visited in line 7. Every time Simplepath(U, V, x) procedure is called in line 8, finding a path to destination node V through new adjacent vertex 'x' of U and updated value is recorded in count variable. Lines 4-8 repeats for all adjacent nodes of U. finally count holds the exact value of number of path between U and V.

From lines 9- 13 checking the value of count variable, if the value of count =0 means there is no path to destination node V and count = 1 means then there exist only one path to V. in both the cases control returning back to line 2 to take next destination node. Otherwise we can conclude the given graph is not singly connected because for a particular destination node V there exist more than one path which violates the definition of singly connected and stop to proceed further.

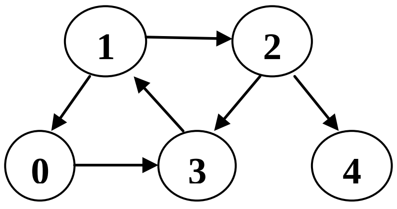
Line 14 executed if and only if there exist exact one path or no path to all other nodes of a graph it means given graph is singly connected.

In each call SimplePath (U, V, x), checking whether the visited adjacent node is destination node or not if so return count value in line 1-2. Line 3 paints x to GRAY. Lines 4-6 examine each vertex y ϵ Adj[x] and recursively visit y if it is WHITE.

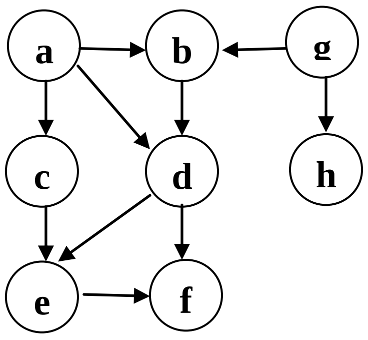
The algorithm traced for the following graph. Graph-1 and Graph-2 are Directed Singly Connected Graph. Graph-3 and Graph-4 are Directed not singly connected Graph.



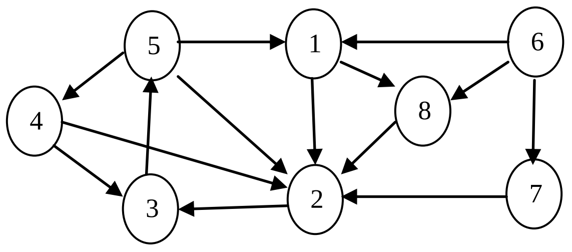
Graph-1



Graph-2



Graph-3



Graph-4

REFERENCES

[1] Introduction to Algorithms, 3rd Edition, The MIT Press Cambridge, Massachusetts London, England. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein [page no. 594].
[2] Adam L.Buchsbaum , Martin C.Carlisle. "Determining Single Connectivity in Directed Graph " , CS-TR-390-92 September 1992.