Homework Assignment 7

CS 430 Introduction to Algorithms

Name : Karthik Mahesh

CWID : A20383027

1.Solution:

Suppose we have matrix A, the adjacency matrix of the graph G, where A(i, j) is 1 iff i has a directed edge to j. If a vertex v is a sink in the graph, all the other vertices have an edge to it and it has no edges to other vertices. This means the row corresponding to vertex v is all 0 in matrix A, and the column corresponding to vertex v in matrix A is all 1 except for A(v, v).

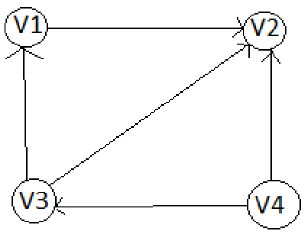
```
Check For Sink(A,k)
Let A be |v|x|v|
For j = 1 to v
    do if a_kj = 1
       then return False
for i ← 1 to |v|
    do if aik = 0 and i != k
       then return False
return True


Sink(A)
let A be |V|x|V|
i = j = 1
while i <= |V| and j <= |V|
    do if a_ij = 1
       then i = i + 1
       else j = j + 1
s ← 0
if i > |V|
    then return 'there is no universal sink'
elseif IS-SINK(A, i ) = FALSE
    then return 'there is no universal sink'
else return i 'is a universal sink'.
```

Sink walks through the adjacency matrix, starting at the upper left corner and always moving either right or down by one position, depending on whether the current entry a_{ij} it is examining is 0 or 1. It stops once either i or j exceeds |V|.
After the while loop terminates, the only vertex that might be a sink is vertex i . The call to Check For Sink then determines whether vertex i is a sink.

Time Complexity : O(|V|)

Proof of correctness :



	V1	V2	V3	V4
V1	0	1	0	0
V2	0	0	0	0
V3	1	1	0	0
V4	0	1	1	0

Here while loop terminates when j reaches V4 and i < |v|
V2 row is all 0 and V2 column is all 1 except (v2,v2)
So it's a Sink.

2. Solution :

A:

DFSN number increases from vertex to its descendants because we start from a vertex and go to its descendants and number them in increasing order. So if vertex w is descendant of v then dfsn(v) should be less than to dfsn(w). If dfsn(w) > dfsn(v), then w is not a descendant of v. So by contradiction
dfsn(v) <= dfsn(w) .. . . (1)

We increase dfsn number by 1 linearly for a vertex's descendants. Dfsn value indicates that the ancestor of that vertex definitely has dfsn(descendant) - dfsn(ancestor) + 1 descendants. It might have other descendants which can increase the number of descendants.
So if w is a descendant of v then,
nd(v) >= dfsn(w) – dfsn(v) + 1 . . . . . (2)

Therefore, from 1 and 2,
dfsn(v) <= dfsn(w) < dfsn(v) + nd(v)

B :

C:

```
Dfs-nd(G,u)

For each vertex u ∈ G.V
   nd(u) = 1
   For each v ∈ G.adj(u)
        Dfs-nd(G,v)
        nd(u) = nd(u) + nd(v)
   next
   return nd(u)
   Number of descendants of vertex u = nd(u)
next
```

Time complexity : O(n^2)

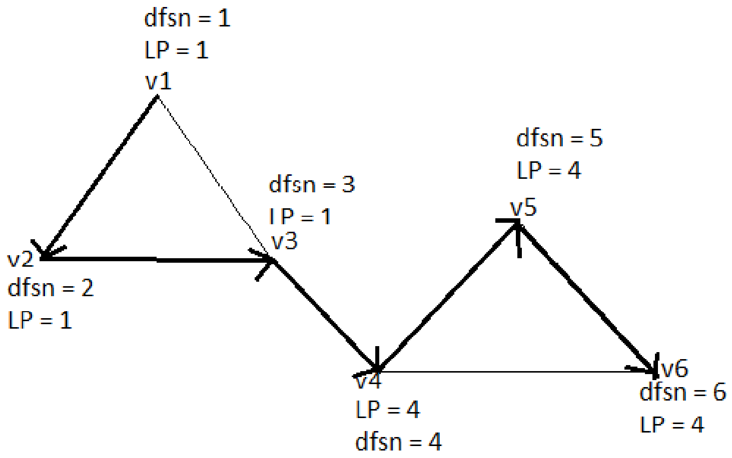
3.Solution:

```
DFS(G,U,V)
Begin
  Mark(u)
  Dfsn(u) ← dfs , dfs++
  Lowpt(u) ← dfsn(u)

  For all w ∈ adj(u)
  Do
   If w is unmarked then
        DFS(G,V,W)
        Lowpt(v) = min(lowpt(v), lowpt(w))
        If lowpt(w) >= dfsn(w)
           "V --- W edge is a Bridge"
   Else
   Lowpt(v) = min(lowpt(v), lowpt(w))
  End
```

Time Comlexity : O(|V|+|E|)

Proof of correctness :

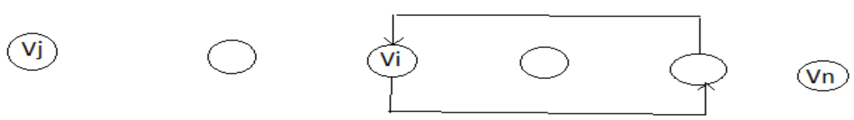


Here Back edges are from V3 to V1 and V6 to V4
Lowpt = Dfsn at v4.
So V3 to V4 is a bridge.
The graph will be disconnected if v3---v4 edge is removed.

4.Solution :

Proof by contradiction

Suppose that G has a topological order v1 , …, vn and that G also has a directed cycle C.
Let's see what happens.
Let V_i be the lowest-indexed node in C, and let V_j be the node just before V_i .
Thus (Vj , Vi) is an edge.
By our choice of i, we have i < j.
On the other hand, since (Vj , Vi) is an edge and v1 , …, vn is a topological order, we must have j < i which is a contradiction.
So a digraph cannot be topologically sorted if it is cyclic.
Therefore ‘digraph can be topologically sorted if and only if it is acyclic’.



References Used :
Analysis and Design of Algorithms texbook
http://web.mst.edu
https://homes.cs.washington.edu