Given **G=(V,E)** is a directed graph. Define a directed graph to be nice if it satisfies the property that either **u ---> v** or **v ---> u** or **both** for any two vertices **u** and **v**.

For any vertex x, define **subtree(x)** as the set of vertices that are visited if we call **DFS(x)** on **G** before calling **DFS**() on any other vertex.

## **16.(a)**

Given that either **u ---> v** or **v ---> u** or **both** for every pair **(u,v)** of vertices. **…(1)**

Choose a vertex **w** such that **subtree(w)** has the maximum number of vertices (choose anyone arbitrarily if there are multiple such vertices).

Let **subtree(w)** does not contain some vertex **k** **∈ V => k** is not reachable from **w.**

Based on the given fact **(1)**, then there must be a path from **k** to **w.**

Hence, **w** will be in **subtree(k),** i.e. **subtree(k)** will have all the vertices in **subtree(w)** + the vertex **w.** So, **subtree(k)** has more nodes than **subtree(w)**. This contradicts the claim that **subtree(w)** has the largest size.

If we continue claiming that **subtree(k)** will have the largest size over **subtree(w)**, eventually there will exist one such **k,** such that **subtree(k)** will have all the vertices. Hence **k** has a path to every other vertex. **[Q.E.D.]**

**16.(b)**

The above claim will still be **TRUE**, even if **G** is undirected.

Since the graph has become undirected, then **u ---> v** means there will be an edge from (**v** to **u)** & (**u** to **v)**. Hence all the paths in the directed graph still hold in the corresponding undirected graph also. Hence the above claim still holds in the undirected graph.

Moreover, if earlier there was one vertex that had a path to every other vertex, then now in the undirected graph there would be paths from all vertices to all vertices.

**16.(c)**

Given that for every vertex **v** there is some **u**!=**v** such that **v -!-> u**.

**To prove:** There must be some pair of vertices x and y such that x -!-> y and y -!-> x.

Let's assume the contradiction of the given statement i.e. for all **(x,y)** either **x ---> y** or **y ---> x** i.e. there does not exist any pair of vertices **(x,y)** which are mutually not reachable from each other.

Earlier in part(a), I proved that then there will exist one such **k,** such that **subtree(k)** will have all the vertices. Hence **k** has a path to every other vertex.

This contradicts our given condition that for every vertex **v** there is some **u**!=**v** such that **v -!-> u**.

Thus our assumption is false i.e. there must be some pair of vertices x and y such that x -!-> y and y -!-> x.

**16.(d) A graph G is nice if and only if** there exists some super source and SCC of the graph has only one topological sorting order.

**16.(e)**

**def find\_supersourse( G ):**

Liz =[]

for i in all vertex(G):

if dfs(G) = vertex(G):

Liz.add(i)

return Liz

**def fun(G):**

Supersourese = find\_supersourse(G)

if length(Supersourese )==0:

return “not nice”

scc=kosaraju(G)

result=topologicalSort(scc) // return all topological sort as list.

if length( result ) !=1:

return “not nice”

return “nice”

**Time complexity :-**

Find\_supersourse usage |V| DFS, and each DFS take O(V+E) time so total time complexity us O(V(V+E)) = O(V2 +VE).

Then kosaraju requires and topologicalSort requires DFS which is again O(V+E).

Total Time Complexity - O(V2 +VE).

**Space complexity:-**

Since we are using only DFS which only requires O(V) space hence Total Space complexity is O(V).