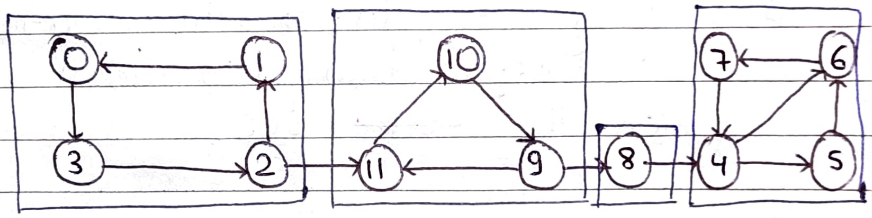
Page :
Date : / /

# Day - 217

## Graph - 21

* **Strongly Connected Component:**

⇒ In a Directed graph, SCC is a subset of vertices where every vertex in the subset is reachable from every other vertex in the same subset by traversing direct edges.




⇒ So, here, we can see that 0 & 3 are strongly connected because we can go from 0 to 3 and from 3 to 0.

⇒ So, in this way, we have four strongly connected component in the group.

⇒ So, if we apply BFS or DFS then we ~~can~~ will traverse all the nodes in one go.

⇒ But we don't want this.

Page:
Date: / /

⇒ So, we can imagine the graph as –


⇒ So, if we change the direction ~~of~~ between two strongly component then we can solve the problem.

⇒ There will be cycle present in every SCC.

⇒ So, if we reverse the direction of the edges then also no impact on the ~~beh~~ SCC.

⇒ So, we will use this property.

⇒ We will follow the order of SCC to apply DFS.

⇒ So, how to achieve the order.

⇒ Order is just saying that SCC1 will be visited before SCC 2.

⇒ So, as we know, we have topological sort that can help in this situation.

Page:
Date: / /

⇒ So, if we use DFS method of T.S then we get ~~an~~ a stack of traversed nodes.

⇒ Also, ~~remember~~ remember that T.S. sort don't give correct answer when cycle is present in the group.

⇒ But here, it can work.

⇒ we use that stack of T.S for finding our answer.

⇒ At the time of T.S, the graph will be normal.

⇒ After that, we reverse the direction of the edges.

⇒ And then T.S stack will help in getting the order.

⇒ This whole algo is known as Kosaraju's Algorithm.

⇒ T.S helps in confirming that (2) will always come before (11).

⇒ After that we start apply DFS and if we get any unvisited node from TS stack then we increase the SCC value

by 1.

=> And apply DFS on this node.

T.C. → O(V+E)
S.C. → O(V+E)