Experiment 7

Ques 1) Topological Sort of a directed acyclic graph (DAG)

Steps 1) Create the graph by calling addEdge(a,b)

2) Call the topologicalSort()

(a) Create a stack and a boolean array named as visited[]

(b) Mark all the vertices as not visited ie., initialize visited[] with 'false' value.

(c) Call the recursive helper function topologicalSortUtil() to store Topological Sort starting from all vertices one by one.

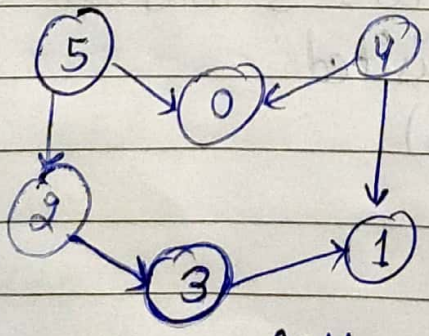
3) def topologicalSortUtil(int v, bool visited[], stack<int> &Stack)

a) Mark the current node as visited.

b) Recur for all the vertices adjacent to this vertex.

c) Push current vertex to stack which stores result.

4) Atlast after return from the utility function, print contents of stack.



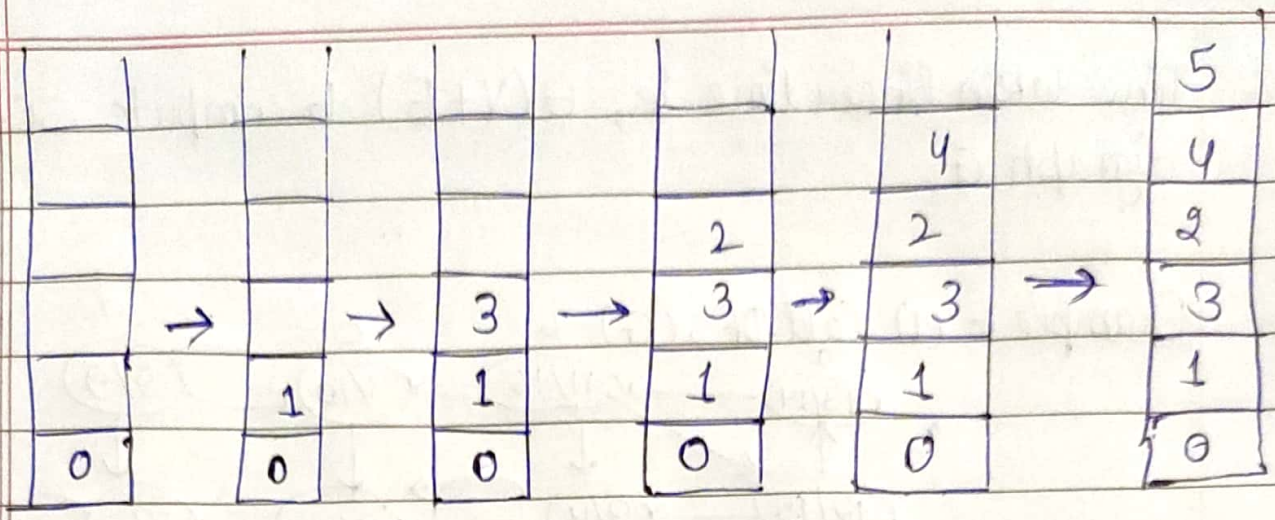
Graph

1) visited[] = | F | F | F | F | F | F |
0 1 2 3 4 5

2) i=0 | T | F | f | F | F | F |
0 1 2 3 4 5

Similary for other values of i

3)



4) Content of stack is pop out one by one and output is
5 4 2 3 1 0

Complexity :- Time Complexity a) Best $\Theta(|V|+|E|)$
b) Average $\Theta(|V|+|E|)$
c) Worst $\Theta(|V|+|E|)$

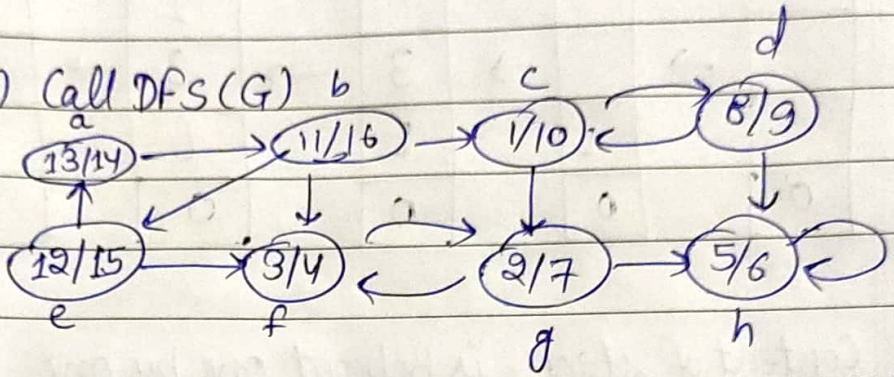
Space Complexity $\Theta(|V|)$

Ques 2) Compute the strongly connected components of a directed graph $G = (V, E)$ using two DFSs one on G and one on G^T.

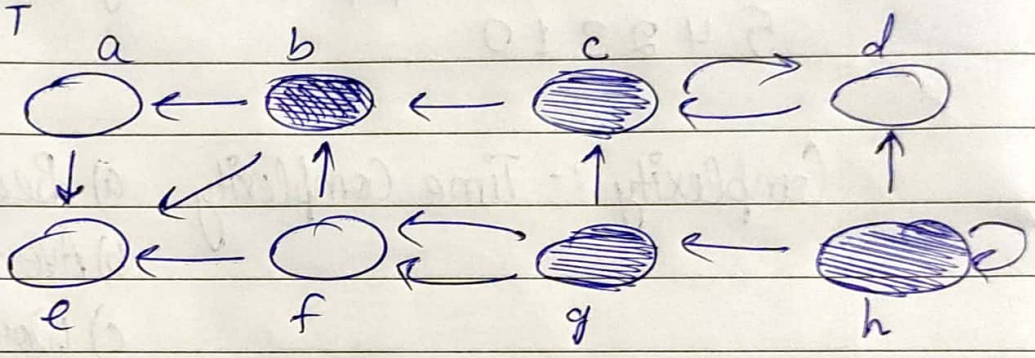
Steps 1) Call DFS(G) to compute finishing times $f[u]$ for all u.
2) Compute G^T
3) Call DFS(G^T) but in the main loop, consider vertices in order of decreasing $f[u]$
4) Output the vertices in each tree of the depth first forest formed in second DFS as a separate SCC.

Time taken linear time ie, $\Theta(V+E)$ to compute SCC of a digraph G.

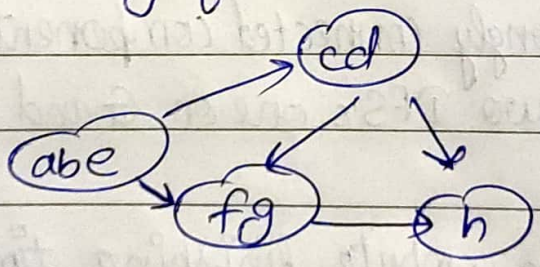
Example :- (1) Call DFS(G)



2) Compute G^T



3) Call DFS(G^T) but this time consider the vertices in order to decreasing finish time.



4) Output :- {a, b, e}, {c, d}, {f, g}, {h}