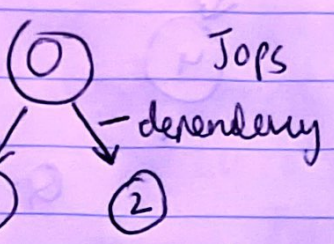
Only works for acyclic graph

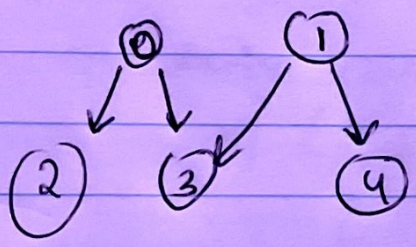
Topological Sortings
- Given a directed acyclic graph

I/O → 

Jops
- dependency

- If there is an edge from 0 to 1 & 0 to 2 that means 0 should finish before 1 & 2
- 1 & 2 can finish after 0 can finish

0 → 0 → 1 → 2 / 0 → 2 → 1

2) 

→ 0 1 2 3 4

- for a Graph, there can be multiple output

BFS based Solution → (Kahn's Algorithm)

① Store indegrees of every vertex
② Create a queue
③ Add all 0 indegree vertices to the q
④ While (q is not empty)
{ a. u = q.pop()
b. Print u
c. for every adjacent v of u
(i) reduce indgree of v
(ii) if indegrees of v is zero & add to q

Time Complexity → $O(V+E)$

DFS Topological Sorting →

① Create an empty stack

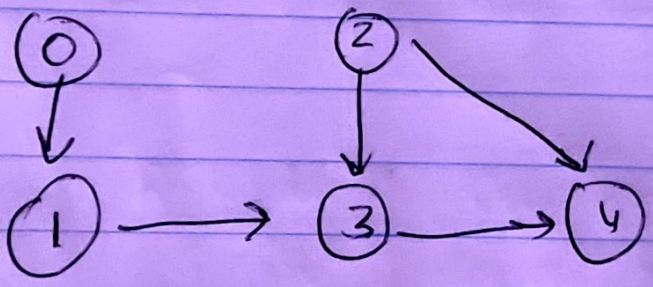
② Run standard DFS → from any vertex
for every vertex v, do following
if (v is not visited)
DFS(v, st)

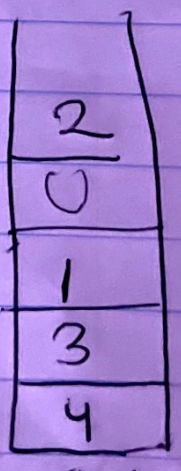
③ while (st is not empty)
pop out item st & print

$O(V)$

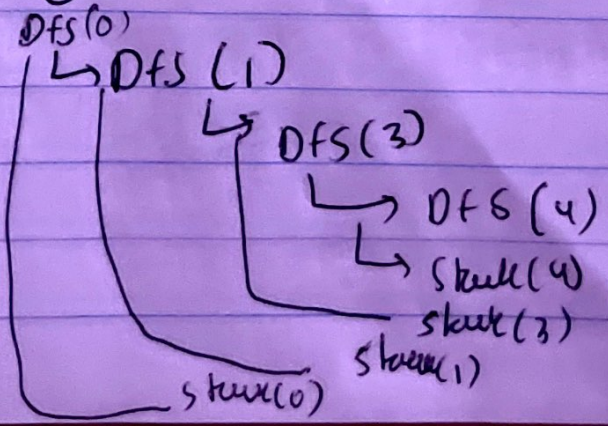
DFS (v, st)
mark v as visited
for every adjacent u of v
if (u is not visited)
dfs (u, st)
Push v to stack

Pushing an element in stack is $O(1)$





Begin from 0 →



- Detect a Cycle in Directed graph :
  Used Topological sorting

- If we process all the vertices, it means the graph is acyclic

no. of vertex == Count