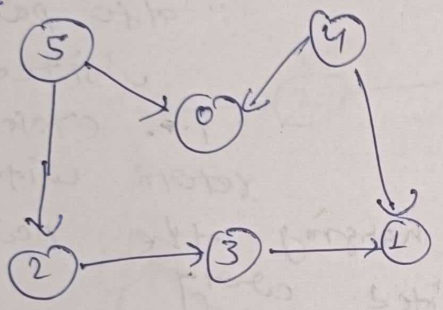
Topological Sorting

Linear ordering of vertices (in DAG) such that if there is an edge (directed) between 'u' & 'v', 'u' appears before 'v' in that ordering. (u→v)

ex:-



Note:- Topological sorting is only applicable for Directed Acyclic Graph (DAG).

Topo sort → 5 4 2 3 1 0
4 5 2 3 1 0

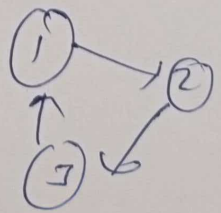
edges
5→0
4→0
5→2
2→3
3→1
4→1

i.e. there can be multiple topological sort / multiple linear ordering available for a DAG.

Que. Why topological sorting is only for DAG?

⇒ Why Directed graph bcz in undirected graph the edge like (1)—(2) i.e (1)→2 & (2)→1 & this scenario you could figure out which should come first

Acyclic



(1)→2
(2)→3
(3)→1 → voilate the condi".

1 2 3

Topological Sorting :-

Topological sorting is a dependency problem in which completion of one task depends upon the completion of several other task whose order can vary.

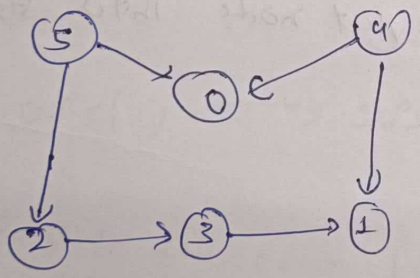
Advantag. of Topological Sorting :-

→ Helps in scheduling tasks/events based on dependencies.

→ Detects cycle in a directed graph.

→ Efficient for solving problems with precedence constraints.

Algo for Topological Sorting →



adj List

0 → {2, 4}
1 → {2, 4}
2 → {3, 4}
3 → {1, 4}
4 → {0, 1, 4}
5 → {0, 2, 4}

// what we are doing in dfs

```
for (i=0 to N)   // N = no. of vertex.
{
  if (!vis[i])
  {
    dfs(i)
  }
}
```

vis

0	0	0	0	0	0
0	1	2	3	4	5

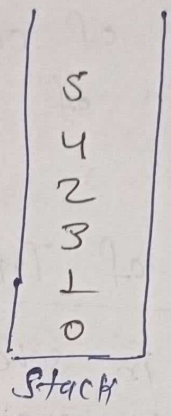
initially marked zero all.

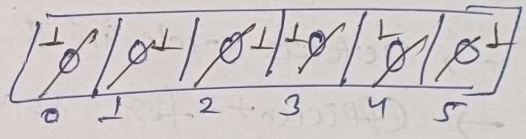
Now for topological sorting take Stak as DS

dfs(0)

// does 0 have any adj nodes/vertex
→ no
so dfs(0) is completed

Note:- while going back from dfs(node) put that vertex into stack





dfs(1) i=1

→ first mark 1 as visited
→ second check the adj nodes of 1's
(there is no adj for 1)
→ while going back from dfs put node into stack

dfs(2)
dfs(3)
(adj = 1 but already visited)
dfs(4)
(adj → 1 & 0)
dfs(5)
(adj 0 & 2)

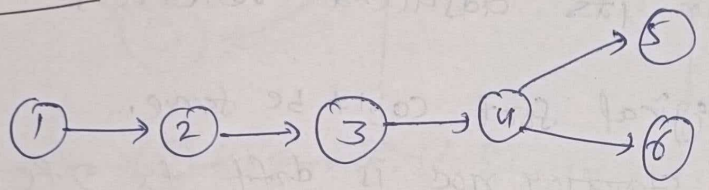
After the completion of DFS take one by one from stack & that will your topo sort

topo sort → { 5, 4, 2, 3, 1, 0 }
→ one of the linear ordering

⇒ Why this approch working?

⇒ Intution

ex :-



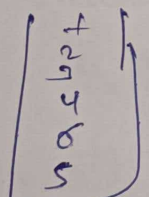
if you are asked to write the linear ordering then

1 2 3 4 5 6
or
1 2 3 4 6 5

How we can get this you call the dfs for 1

Dfs(1) → Dfs(2) → Dfs(3) → Dfs(4)
Dfs(5) Dfs(6)

& when you come back your are putting them into stack



the top of the stack. element say that the DFS for that is complete in the last or that task should be completed in first in topo sort.

Topological sorting vs DFS

In DFS, we are printing a vertex and then recursively call DFS for its adjacent vertices.

In topological sorting, we need to print a vertex before its adjacent vertices.

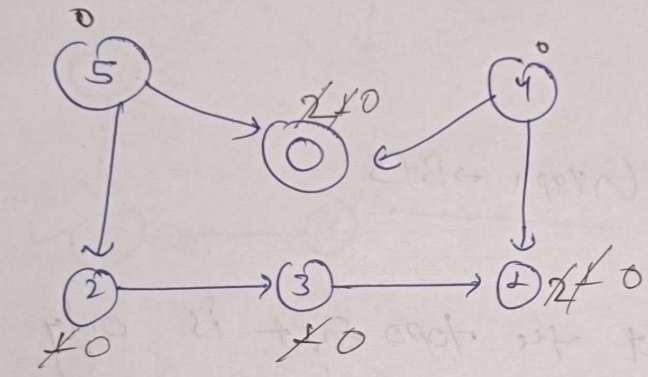
DFS & topological sort can't be same.

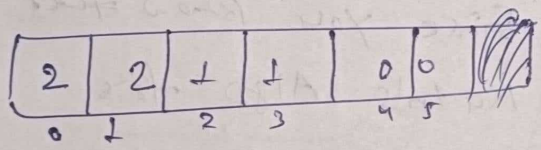
for DFS if starting vertex is diff for DFS is different.

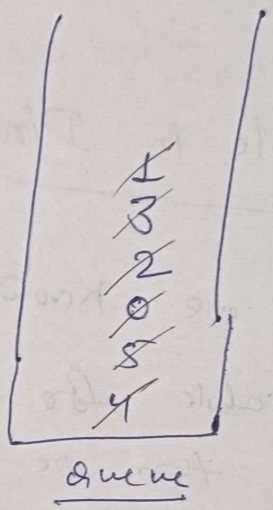
topological sort might be very for a DAG.

Topological Sorting (Kahn's Algorithm / BFS)

Indegree → No of incomming edges.







topo sort → 4 5 0 2 3 1

Since 5 & 4 don't have any incomming edges then you can place 5 & 4 at starting of your linear ordering (either 5 or 4)

Step-1 Insert all the nodes whose indegree are '0' in queue

Step-2: take a node from queue
node = 4
and go the Neighbours of the four ie
(4) → (0) & (4) → (1)
and decrease the indegree of 0 & 1 by 1 bcz 4 is now in our linear ordering

& check if the indegree of 0 & 1 is zero or not.

if zero then place them into queue else skip.

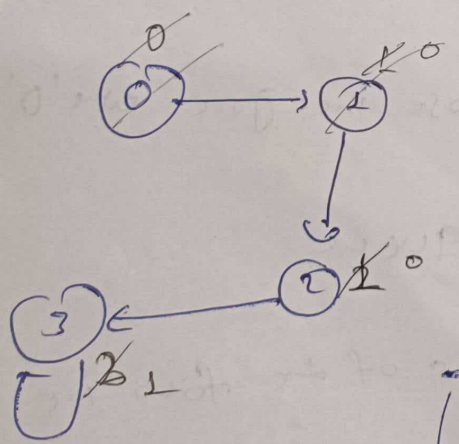
Step-3 go on.

Cycle in Directed Graph → BFS

As we know that the topo sort is only applicable for DAG & since you know this then we apply the Kahn's Algo for directed graph. if there is no cycle then

topo sort. size = V

else topo sort. size ≠ V



topo sort = 0 1 2

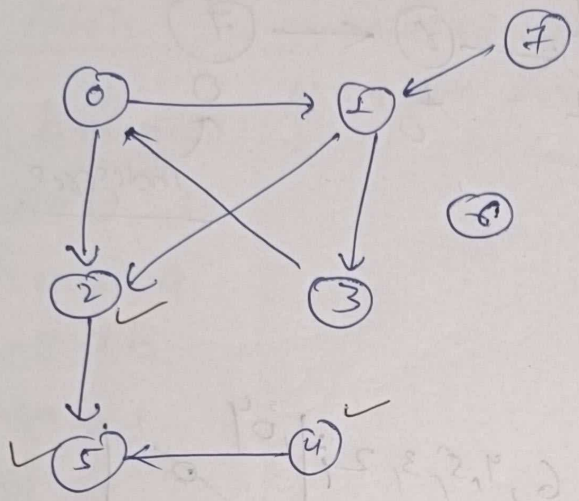
Q is empty &

topo sort. size ≠ V → cycle

T.C. = O(V+E)

S.C. = O(V).

Find Eventual Safe State – Bfs



Safe-Node?

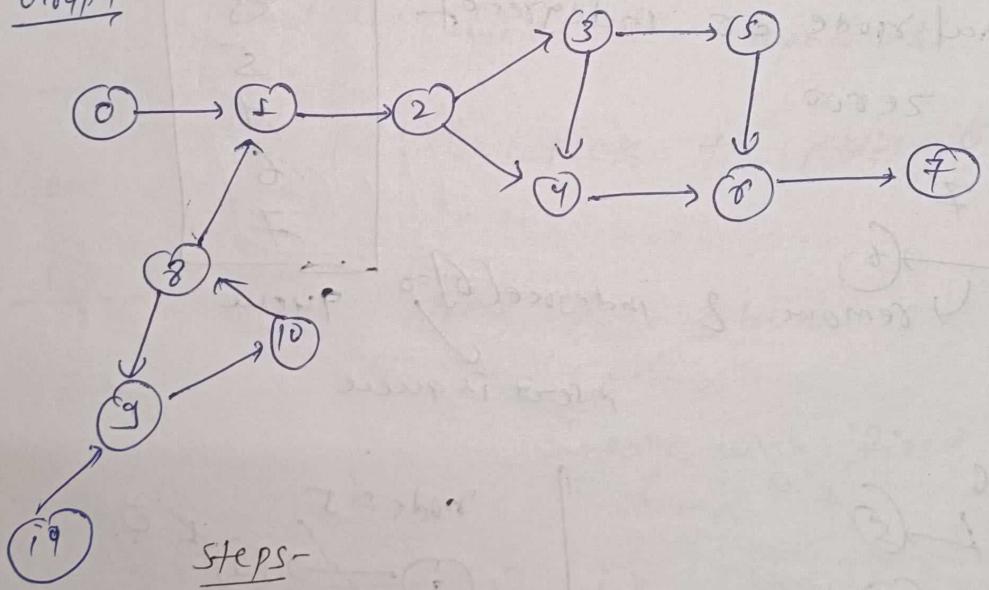
↓

terminal Node

safe Node → 2, 4, 5, 6

Soln.

Graph



Steps-

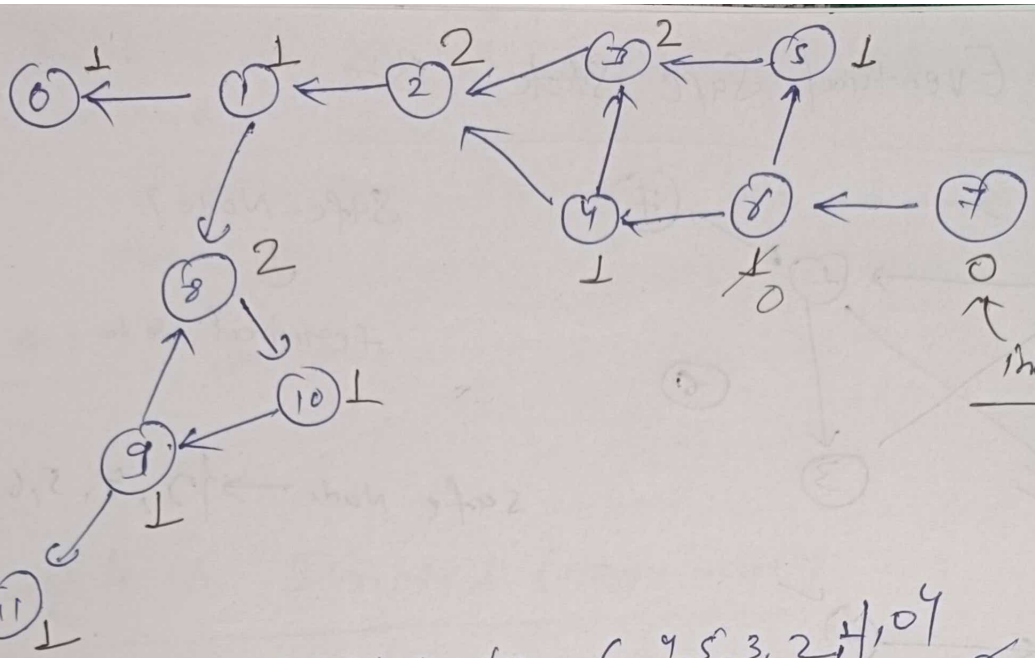
(1) Reverse all the edges in above graph.

(2) Apply topo sort

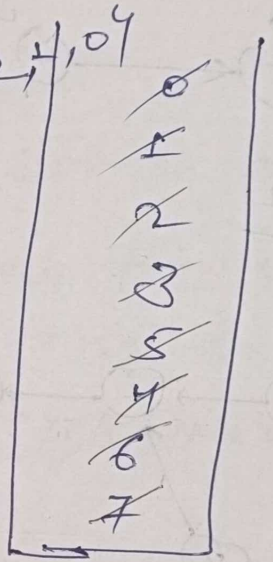
↳ write the indegree of nodes in reversed graph

↳ get all node whose indegree is zero i.e. terminal Nodes

→ do a removal of edge & reduce the indegree.



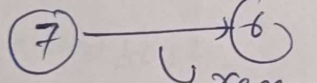
SafeState = {7, 6, 4, 5, 3, 2, 1, 0}



toposort

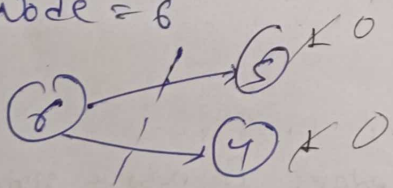
initially 7 is queue bcz it is a terminal node or indegree of 7 is zero

node = 7

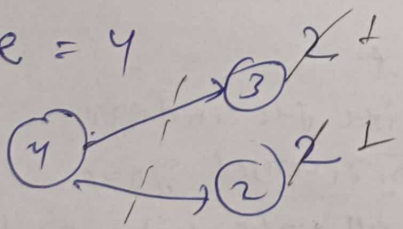


↳ remove & indegree[6] = 0 → insert in queue

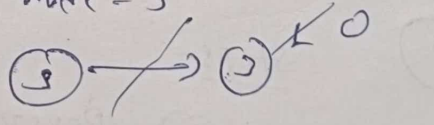
node = 6



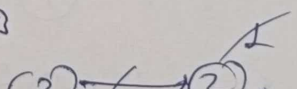
node = 4



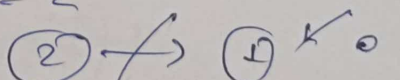
node = 5



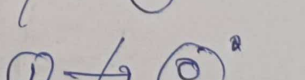
node = 3



node = 2



node = 1



Alien Dictionary

Alien

$N = 5 \leftarrow$ Words

$K = 4 \leftarrow$ first four alphabets of english letters.

b a a
a b c d
a b c a
c a b
c a d

a	b	c	d
0	1	2	3

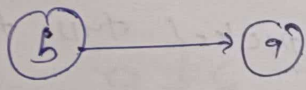
To solve this problem first we make a directed graph & after that we find the topo sort.

To make a DG pick the pair of string one by one.

like 1st

baa & abcd

compare when first difference is found

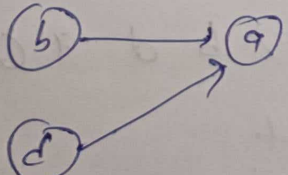


i.e. there is an edge b/w b to a

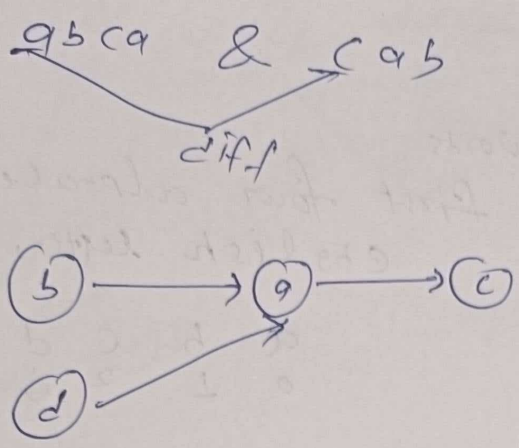
2nd

a b c d & a b c a

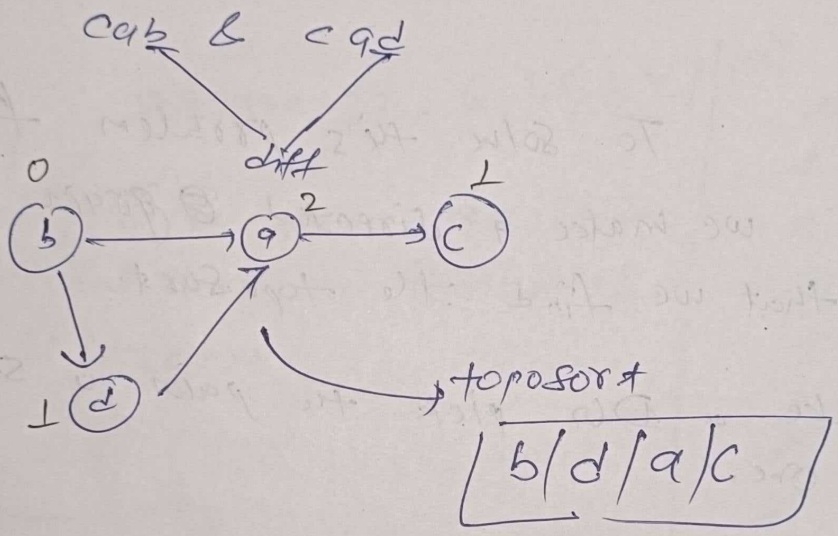
difference



2nd



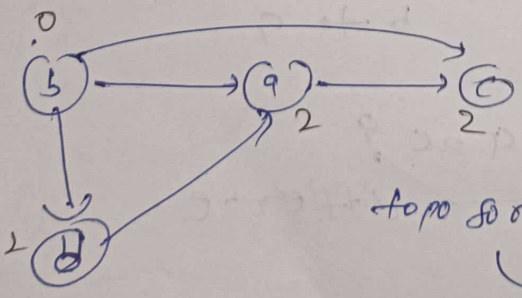
4th



Note why we are not picking up

baa & cab

bcz you know that the words are sorted & if you picked this. then



topo sort

↳ b d a c

No change in topo sort

Steps →

1. make an adj list of k vertex
2. pick the string in pair & find the first difference & create an edge
3. apply the topo sort algo on DAG
4. convert the digits to character & make the ans string.

Note :-

when the order is not possible

(1) abcd → i.e. when the sorter string
abc is after the larger string
then order is ~~not~~ not possible

(2) when there is cyclic dependency.

a b c]→ (a) → (b)
b a f]→ (b) → (a) X
a d e]→

dictionary wrong.