plan

Sunday, 3 October 2021 10:05 PM

Linkedlist → add
→ Dumy
↳ pointr
↳ Mid break

✓ Array and strings
✓ DP
✓ Graph

Revision → Recursion and Back tracking.

→ TSP3 A+B || → Jitu Sir

→ Revision ⟹
↳ TSP1+2

✓ Saturday → ② ✓
✓ Sunday → ③ ✓
✓ Thursday → ① ✓
✓ Saturday → ② ✓
✓ Sunday → ② ✓

R & B ✓

✓ Linked list tree
↳ Remaing for TSP3

Morning. (10th october 2021)

① sliding puzzle
② optimised water distribution
③ Eulerian path and circuit

Evening. (9th october 2021)

✓ ~~①~~ BFS cycle + DFS
✓ ~~②~~ DFS (Directed Graph)
✓ ~~③~~ Topological sort (May be graph is cyclic)
↳ kahn's Algo
~~④~~ Course Schedule - I
~~⑤~~ Course Schedule - II

Evening (11th october 2021)

① Bellmen ford
② Negative weight and cycle detect.
③ Max Edge Removal

[DFS + BFS + cycle detection + DSU + -ve Cycle]

[Amazon] Condensatin

Thursday

① Floyd warshell

② Alien dictionary

③ Min swap to sort an array

④ Bipertite] from LL .

Recursion and Back tracking

2

colouring a border

Reorder on Reverse

0-1 BFS

Rank transformation of matrix

Min. malware spread

" " " 2

Swim in Rising water

Region cut by slashes

Reconstruct Itinerary

③ class

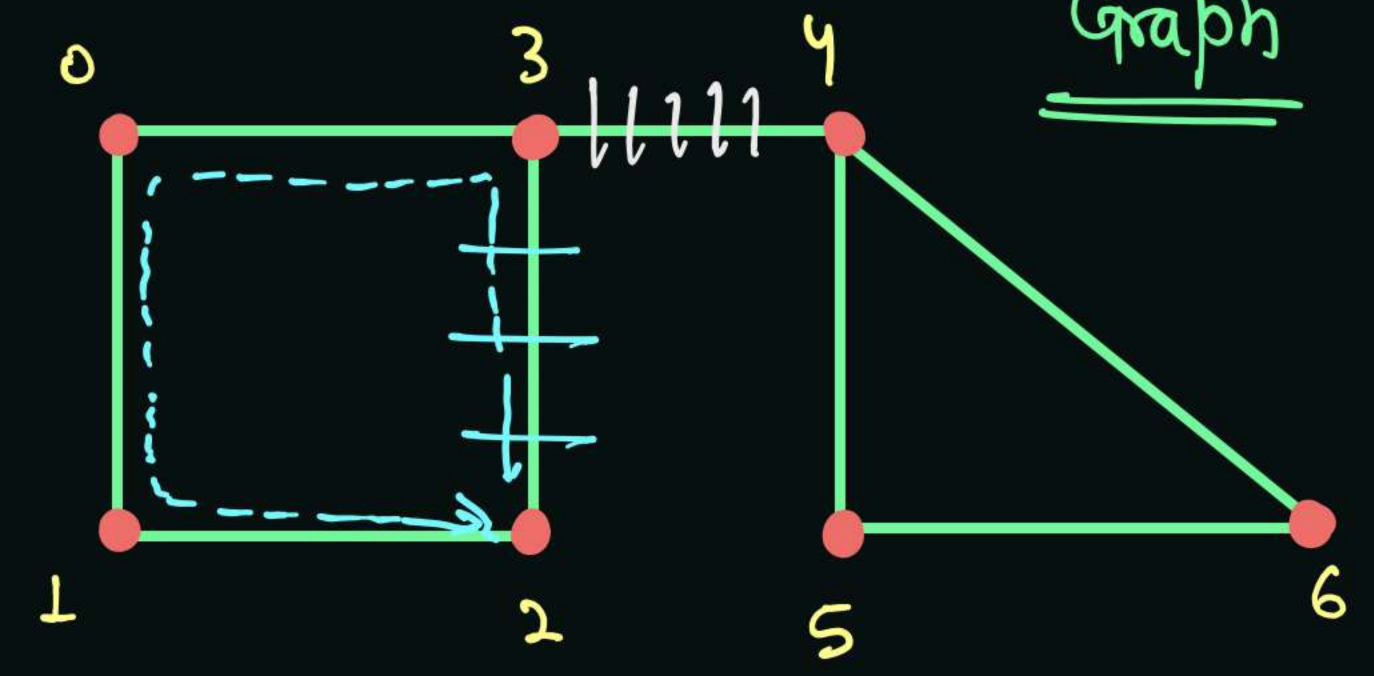
Account Merge

Minimise hamming distance after Swap operation.

DFS + BFS

Saturday, 9 October 2021 7:34 PM

Undirected Graph



Cycle Detection from BFS →

visited booleam array →

0	1	2	3	4	5	6
~~F~~	~~F~~	~~F~~	~~F~~	~~F~~	~~F~~	~~F~~
T	T	T	T			
T	T	T	T	T	T	T

BFS → ① Get + Remove

DS → queue ② Mark *

③ work → (print)

④ add unvisited nbr

Loop → 0 ✓ → BFS

1 → X

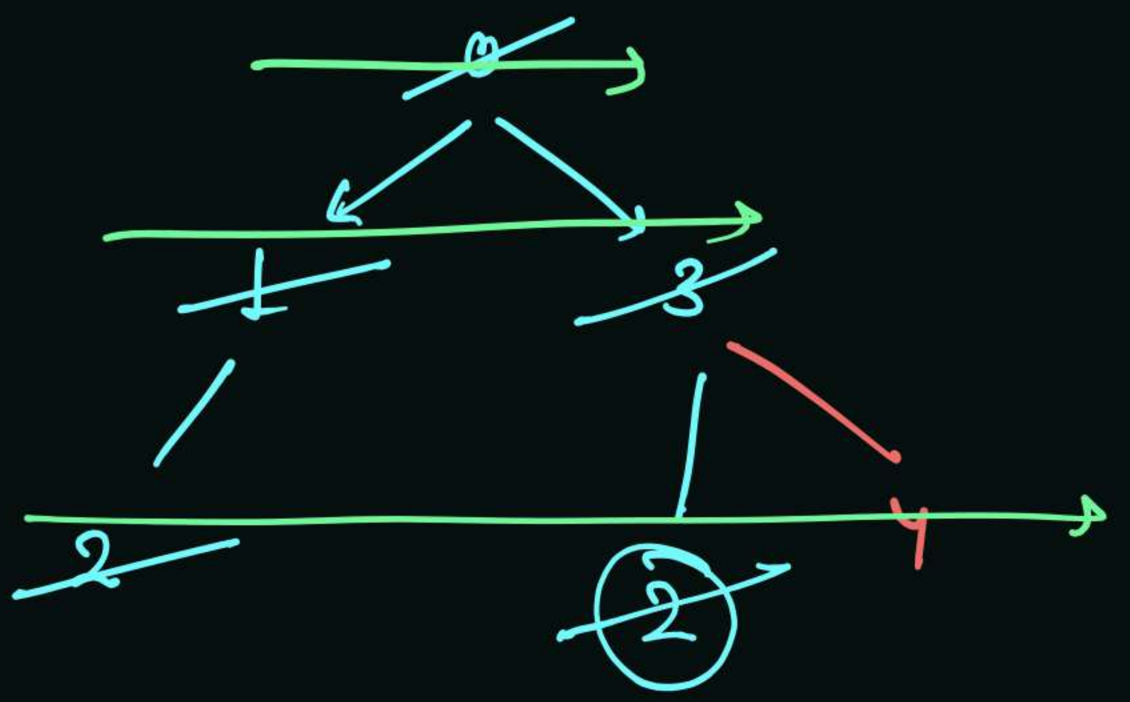
2 → X

3 → X

4 ✓ BFS

5

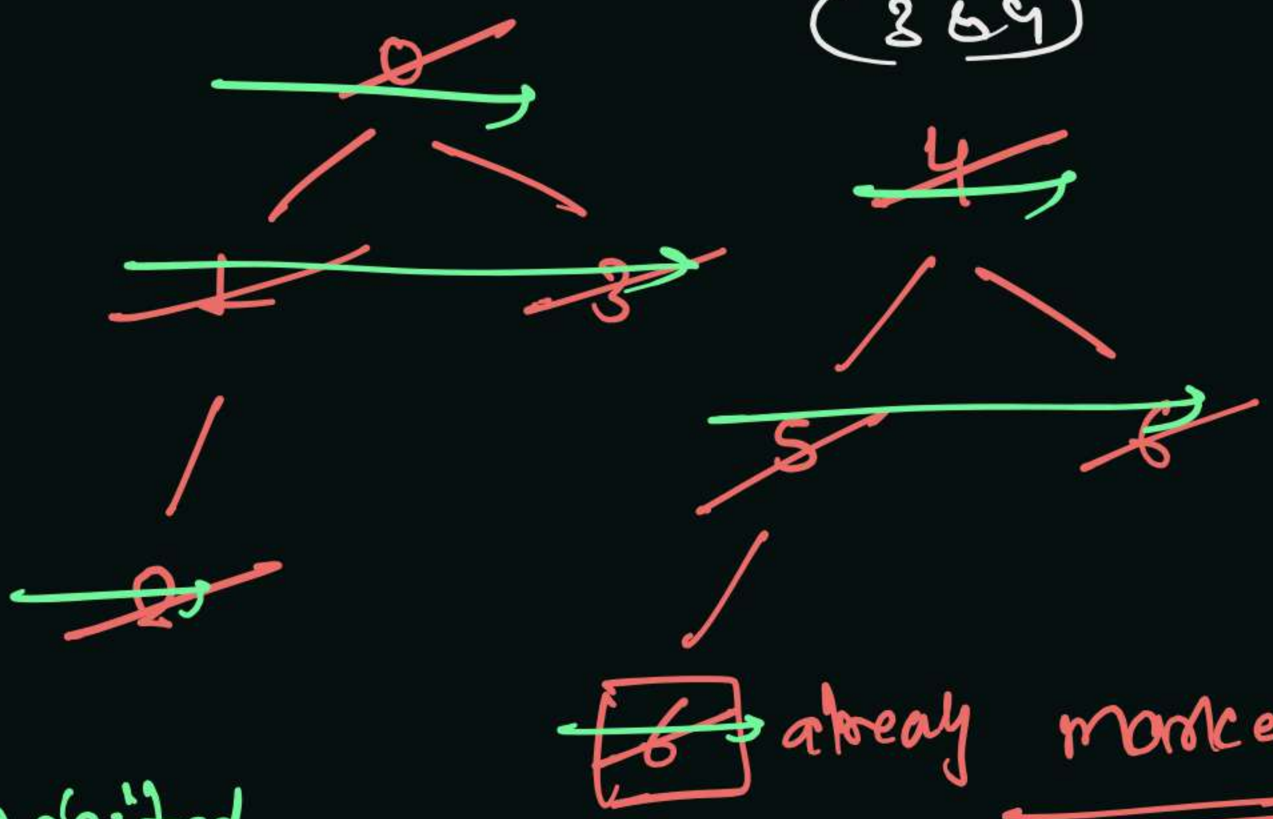
6



return true.

↳ Already marked [cycle]

if no edge b/w (2 & 3) & no edge b/w (3 & 4)



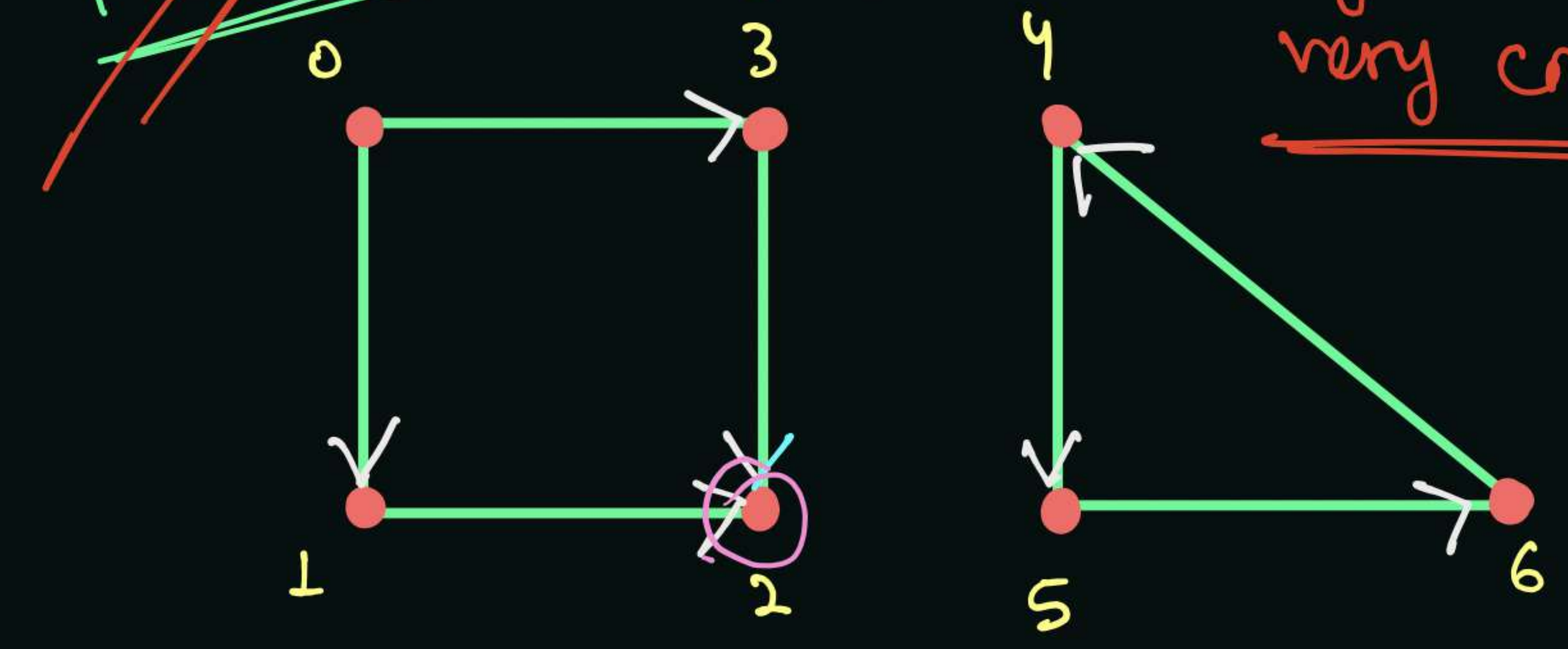
already marked]→ cycle is detected

return

Cycle Detection from DFS]→ if vertex is visited but not parent, then graph is cyclic]

~~in BFS~~
~~(Same as DFS~~
~~Manage mypath~~ → path Menegemet in BFS very complex

* cycle Detection of directed graph using DFS *



For directed graph →

if visited, then mange the path in which vertex is visited.

	0	1	2	3	4	5	6
visited	T~~F~~	~~F~~T	~~F~~T	~~F~~T	~~F~~T	~~F~~T	~~F~~T
Mypath	F	F	F	F	~~F~~T	~~F~~T	~~F~~T

Backtracking

visited → boolean

mypath → boolean

if mypath is not there

then think about complexity.

→ unmarked visite-

0 ✓ DFS cycle

1 → x

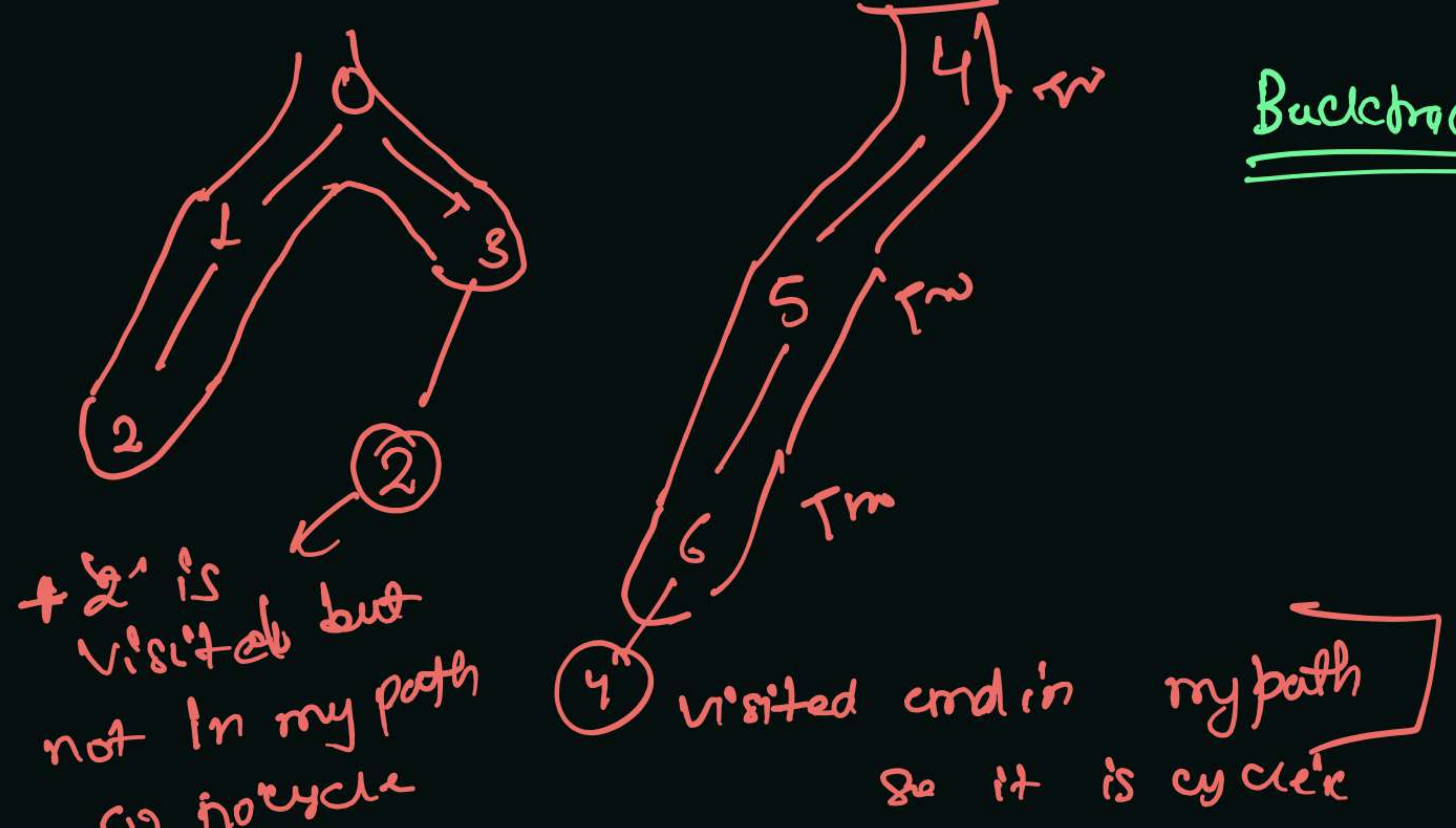
2 → x

3 → x

4 ✓ DFS cycle

5

6



* '2' is visited but not in my path so nocycle

visited and in mypath so it is cycle

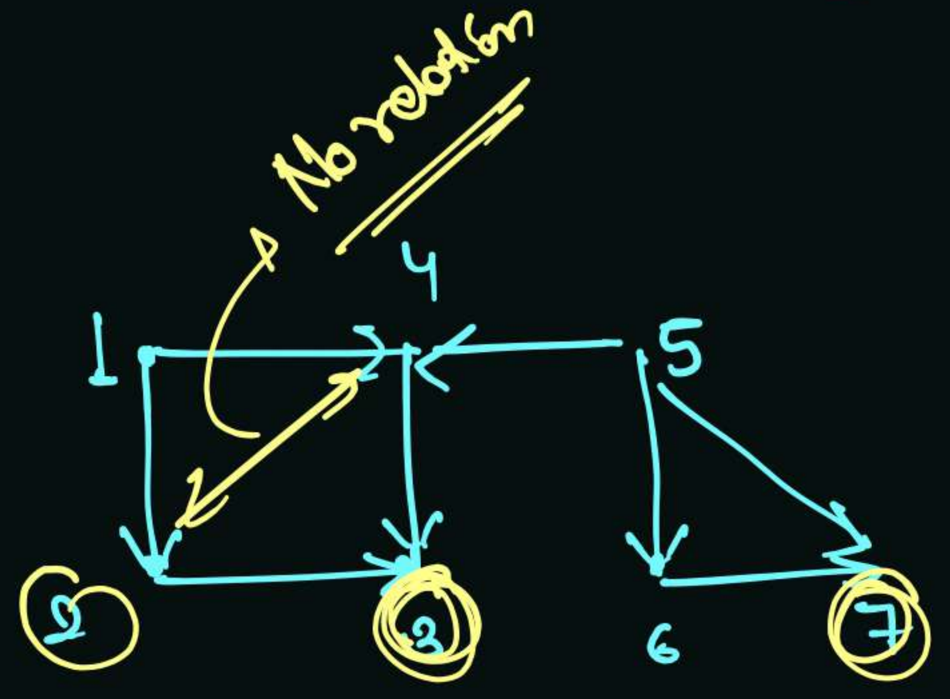
Kahn's Algorithm

Saturday, 9 October 2021 7:34 PM

[It helps in Topological Sort] It can manage cycle detection in directed graph.

Topological Sort → order of dependencies

Ex:



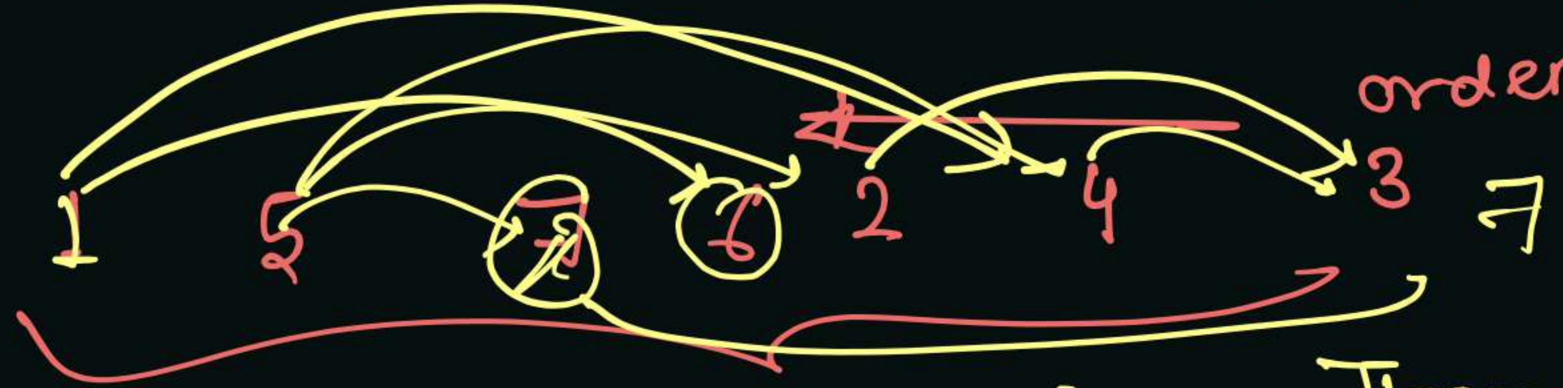
Consider Edge 2, 3 → it represented that '2' is depends on '3'

1-4
1-2
2-3
4-3
5-4
5-6
5-7

Real life Application →

(1) Dependent work

- course schedule
- compilation order
- Methods arrangement



order of compilation

Corder ⟶ There are many possibilities for same graph.

Level 1

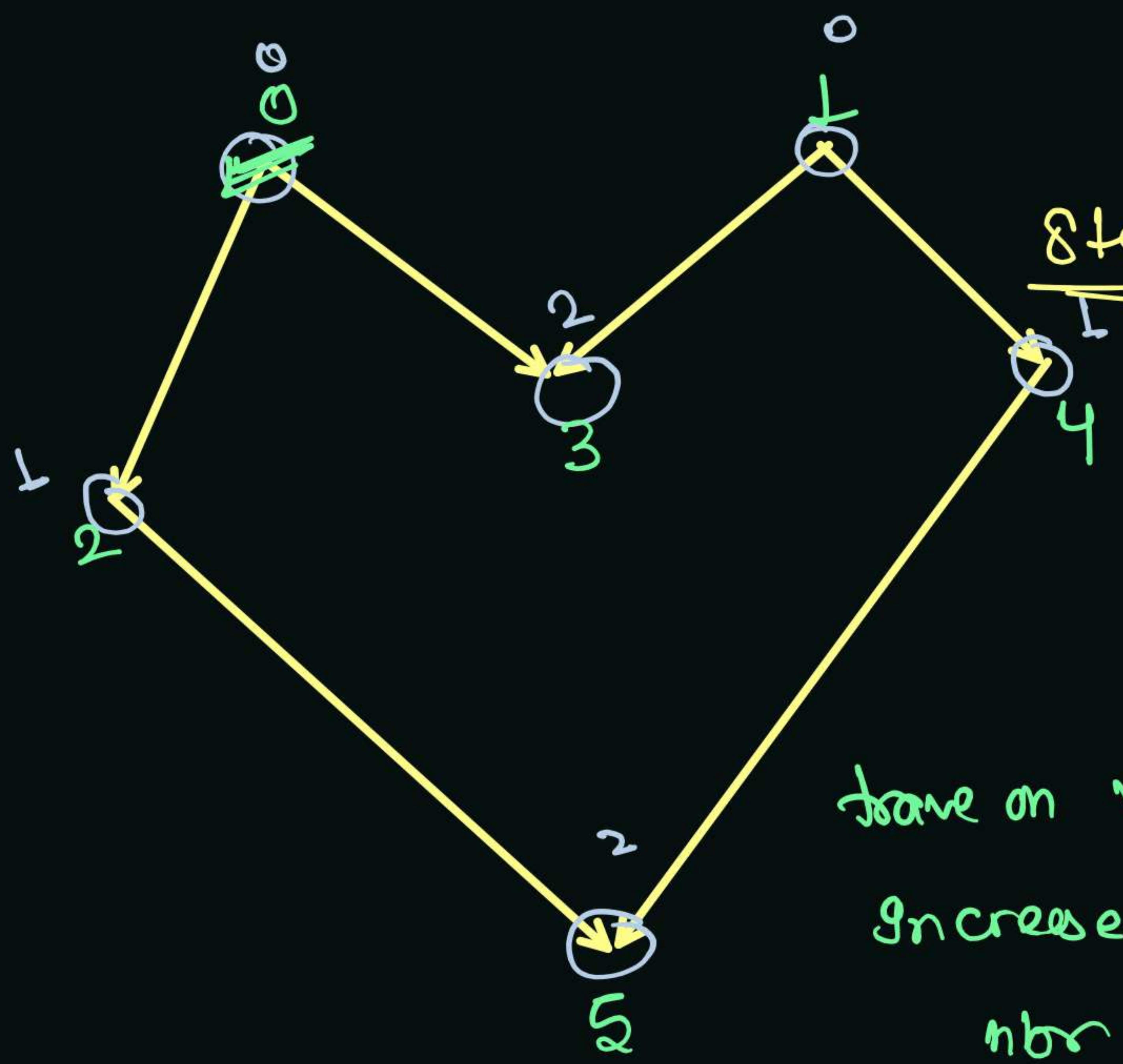
Apply DFS, add element in Stack in Post area.

Topological sort ⟶

Topological sort ⇒ Graph must be directed and Acyclic] why??

Discuss it later.

Kahn's Algo



Step ① Indegree → No. of edges come inward.

Indegree array →

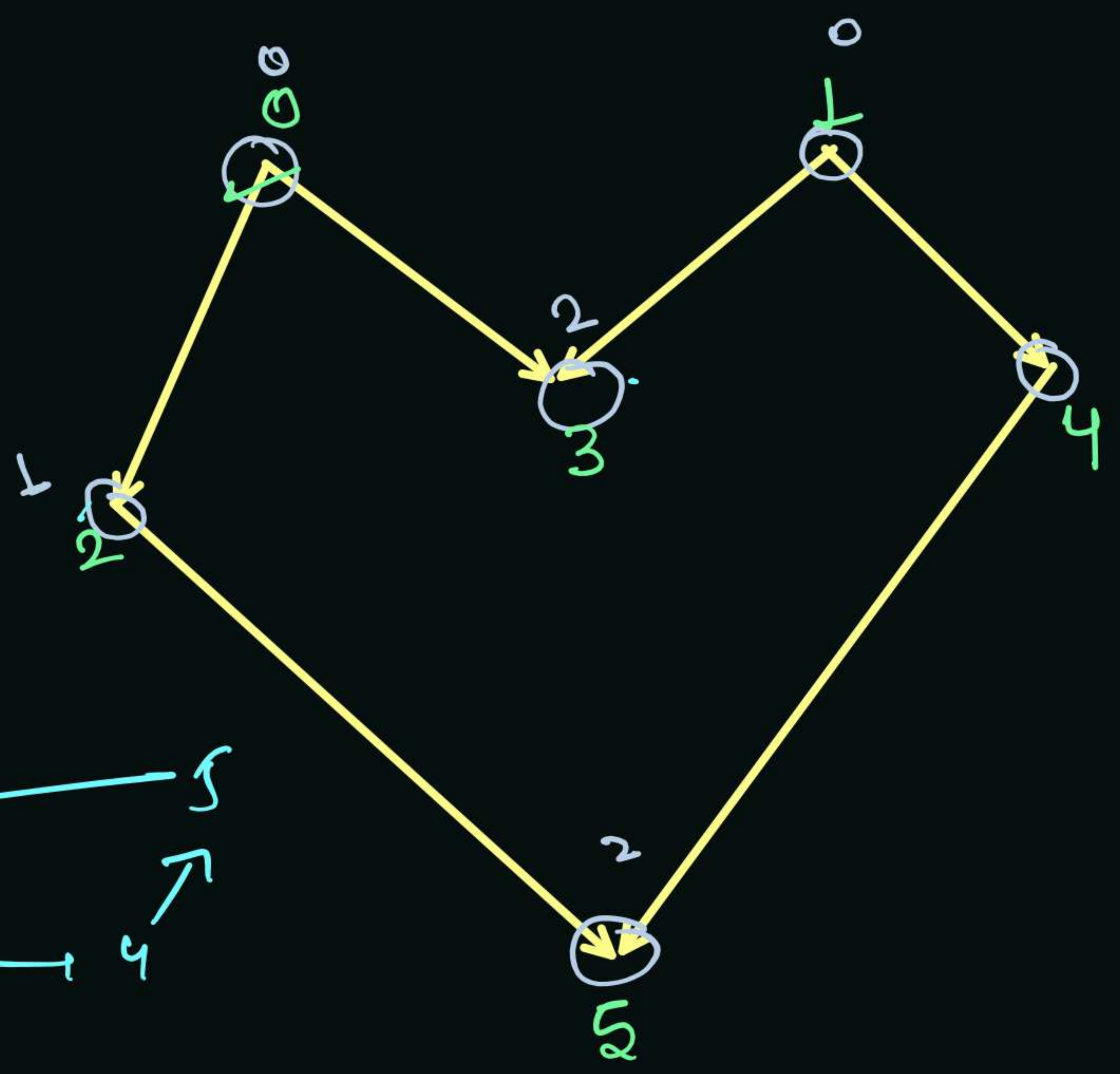
0	1	2	3	4	5
0	0	1	2	1	2

trave on vertex and increase Indegree of nbr vertex

0 → ✓
1 → ✓
2 → ✓
3 → ✓
4 → ✓
5 → ✓

If there is no element present where Indegree is zero, then topological sort is not possible

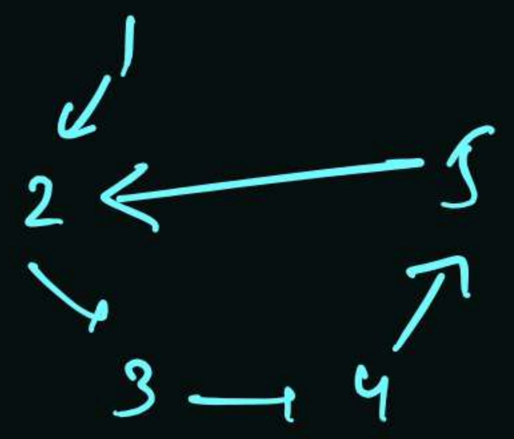
NOTE: if There is no zero then definitely there is a cycle. but vice-versa is not necessary.



Indegree array →	~~0~~	~~1~~	2	3	4	5
	0	0	~~1~~	~~2~~	~~1~~	~~2~~
			0	~~1~~	0	~~1~~
				0		0

Step 1 - Indegree

Step2 → Add vertex in queue which have Indegree '0'

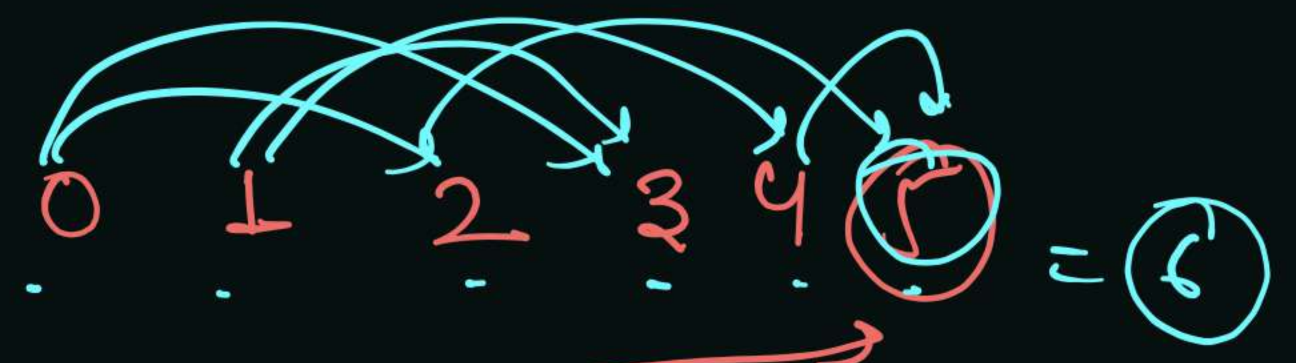


~~0~~ | ~~1~~ | ~~2~~ | ~~3~~ | ~~4~~ | ~~5~~

① remove ✓

② print ✓

③ decrease Indegree of nbr and if nbr have '0' Indegree after removal, then add it in queue.



count

if

Topological sort

order of compilation

0 - 2 ✓

0 - 3 ✓

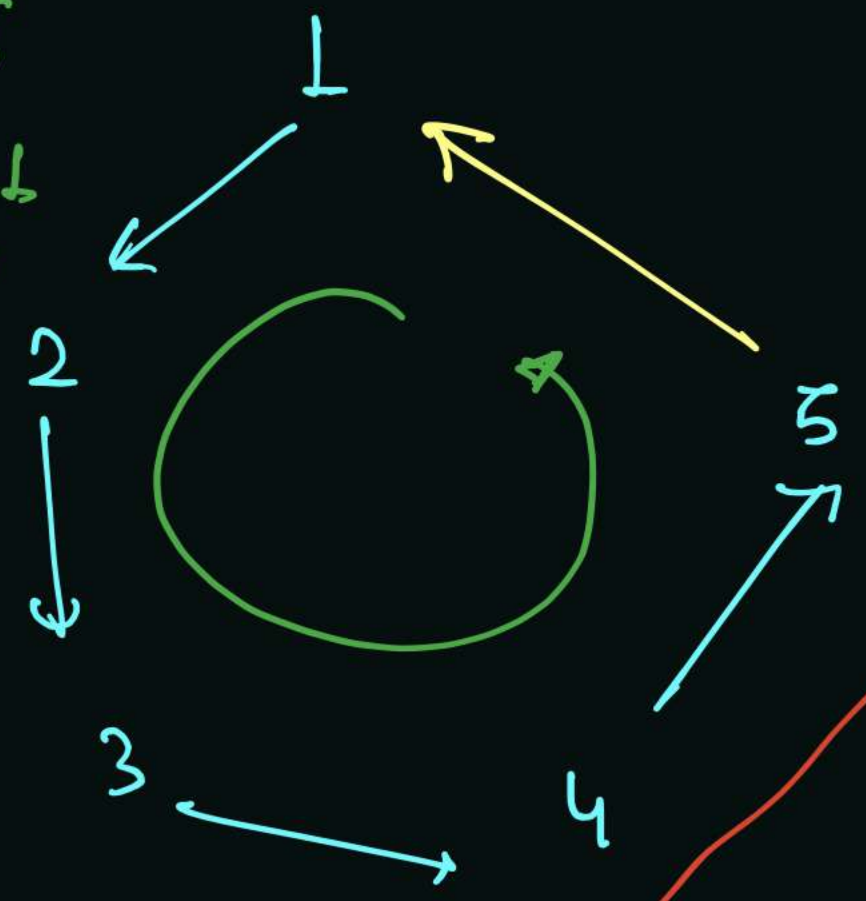
1 - 3 ✓

1 - 4 ✓

2 - 5 ✓

4 - 5 ✓

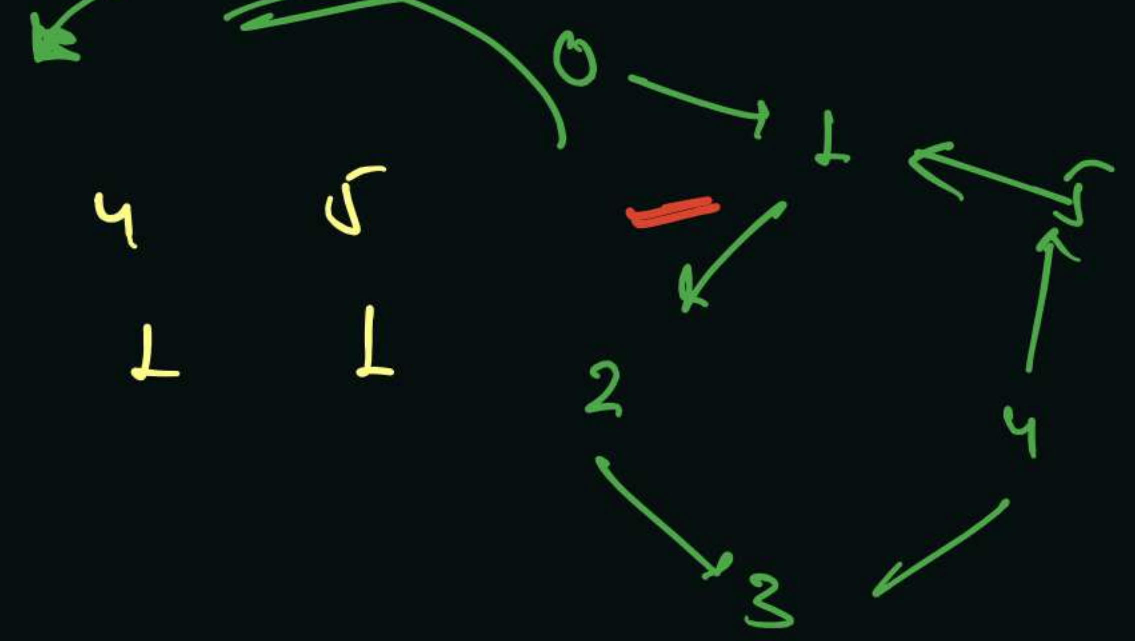
1	2	3	4	5
1	1	1	1	1



Indegree

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

graph

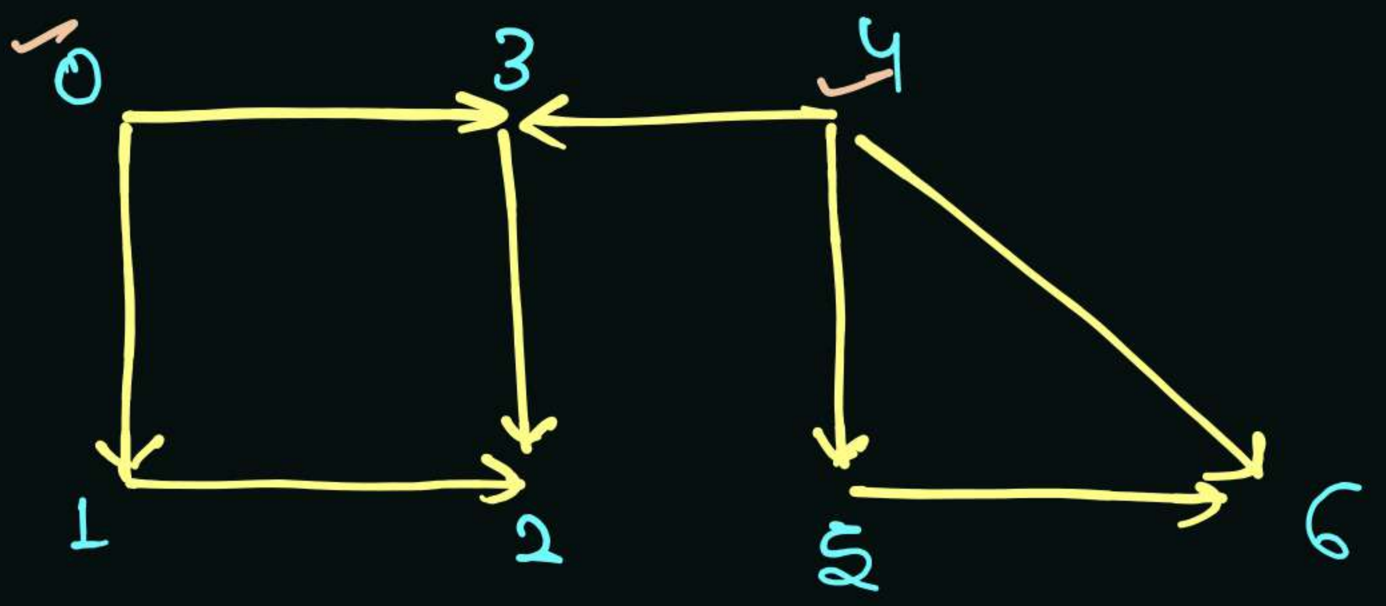


~~0~~ | ~~1~~

0 1

count = 2

count must be equal to 6 for perfect topological sort.



Indegee →	0	1	2	3	4	5	6
	0	~~1~~ 0	~~2~~ ~~1~~ 0	~~2~~ ~~1~~ 0	0	~~1~~ 0	~~2~~ ~~1~~ 0

add Element in queue having indegee '0'

0 → 7 5 6 0 1 3 2

1 ✓

2 ✓

3 ✓ → get + remove

4 ✓ → print

5 ✓

6 ✓ → dec. in Indegee of nbr, and if indegee become 0 add it in quee.

~~0~~ | ~~4~~ | ~~1~~ | ~~3~~ ~~5~~ | ~~2~~ | ~~6~~ |

Interchangable.

0 4 1 3 5 2 6

Count = ~~0~~ 7

Topological Sort

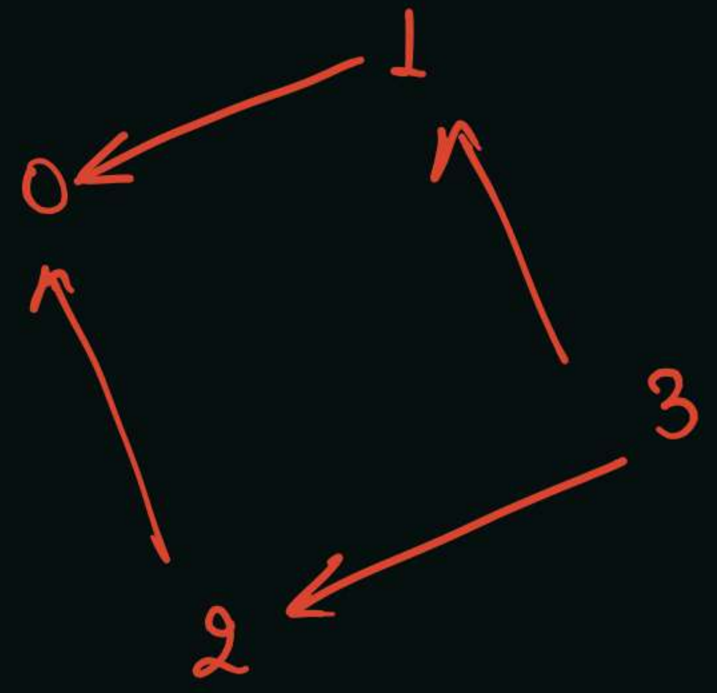
4 0 1 3 5 2 6

Both are correct.

✓ ✓ ✓ ✓

[[1,0],[2,0],[3,1],[3,2]]

graph →



Indegree

0	1	2	3
~~2~~	~~1~~	~~1~~	0
~~1~~	0	0	
0			

~~3~~ | ~~1~~ | ~~2~~ | ~~0~~ |

stack ⊃ add in stack

↳ Pop and add in

res1 — ans

3	1	2	0

←