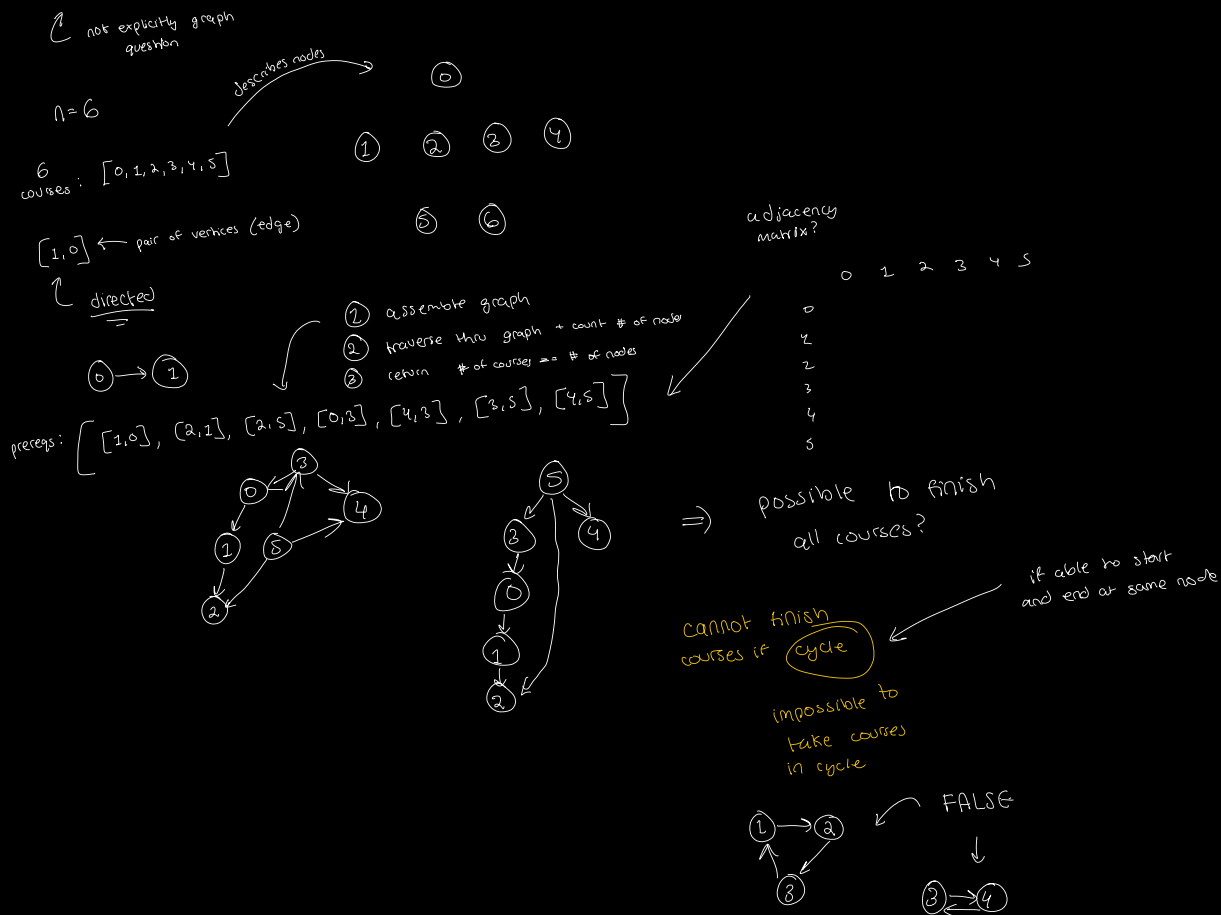


Problem: Total of  $n$  courses ( $0 - n-1$ ), some courses have prereqs

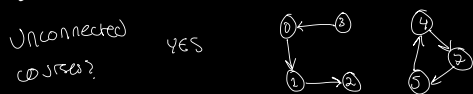
Ex:  $[1, 0] \rightarrow$  course: 1, prereq: 0

Given total number of courses and an array of prerequisite pairs, return if possible to finish all courses.



Intro:

• Verify Constraints



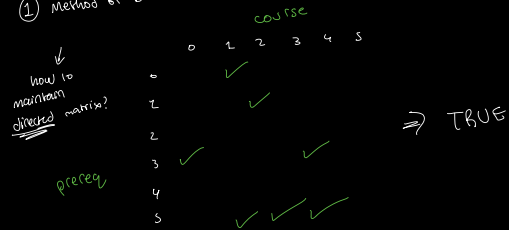
check no cycle  
in any component

- Create Testcases

## Brute Force:

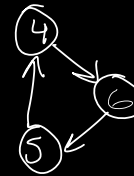
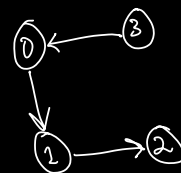
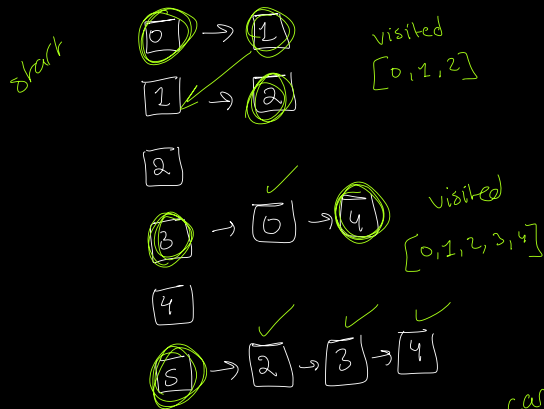
- Brainstorming & Pattern Observations

① Method of "building nodes"



prereqs:  $\left[ \begin{matrix} \text{course} \\ \text{row} \end{matrix} \right] \left[ \begin{matrix} \text{preq} \\ \text{col} \end{matrix} \right] \left[ [1,0], [2,1], [2,5], [0,3], [4,3], [3,5], [4,5] \right]$

① There is ALWAYS at least



detect cycle

can you reach stopping point?

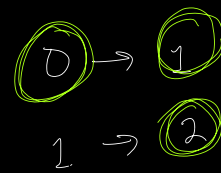
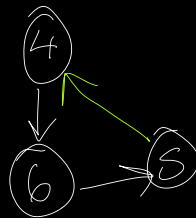
How to detect a cycle?

bfs

completed [0,1,2]

completed [0,1,2,3,4]

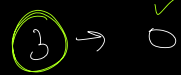
dfs



0,1,2,3,4,5,6

2 - stopped

visited [0,1,2]



visited [0,1,2,3]



visited

[0,1,2,3,4,6]

5  $\rightarrow$  4

6  $\rightarrow$  5

completed

[0,1,2]

[0,1,2,3]

visited [4,6,5]

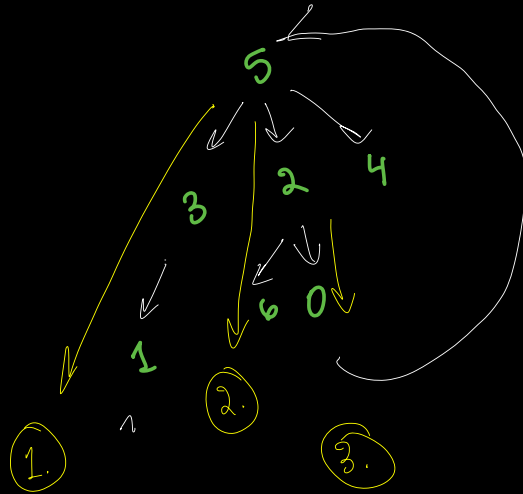
failed

8

- Pseudocode
- Write code
- Run through testcases
- Analyze time and space complexity

Optimal:

- Brainstorming & Pattern Observations
- Pseudocode
- Write code
- Run through testcases
- Analyze time and space complexity



dfs

Queue: [0]

Visited:

Q: [2, 5, 6]

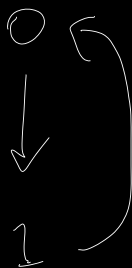
V: [0, 1, 3, 4]

Queue: [0, 1, 3, 4]

Visited: [0]

Q: [0]

V: [0, 1, 3, 4, 2, 5, 6]



return false

Q: 0

V:

Q: 1

V: 0

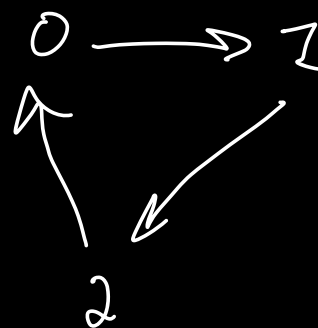
Q: 0  
V: [0, 1] False



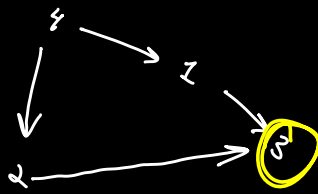
Course

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

	0	1	2
0			✓
1	✓		
2		✓	



$[1, 0], [2, 1],$   
 $[0, 2]$



4: 0

[1, 1]

[2, 1]

[3, 2]

① ⑤

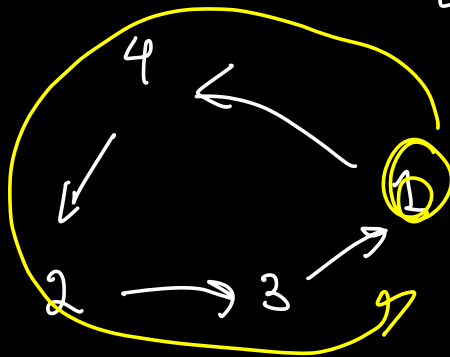
Q: 1      S: 0, 1

Q: 3      S: 0, 1, 3

Q: 2

course  
if 3 in queue:

[      ]



Q: 1      S: 0, 1

Q: 4      S: 0, 1, 4

Q: 2      S: 0, 1, 4, 2

Q: 3

S: 0, 1, 4, 2, 1

separate seen array?

visited =

1  
↓  
0

Q: 0

S: 0

Q: 1

S: 0, 1

neighbor

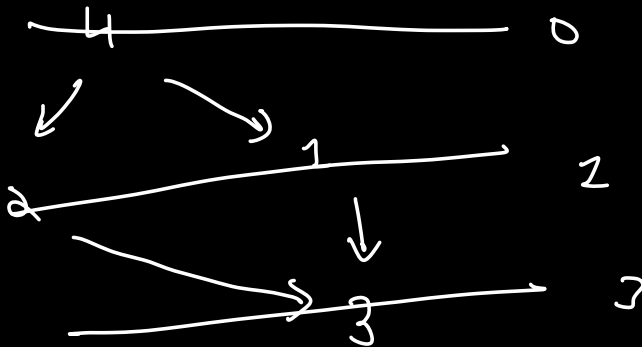


reset at every new "island"

seen  $[i] =$

$[False, -1]$

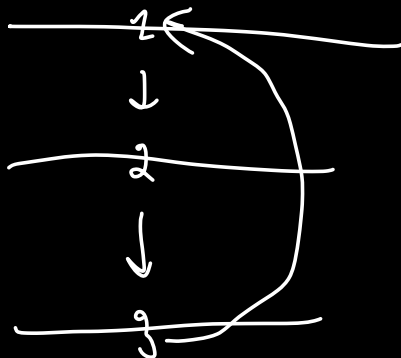
every node starts at its own level



Q: 4

S: 4

if  $= -1$  level  $++ 1$   
 $[True, 0]$

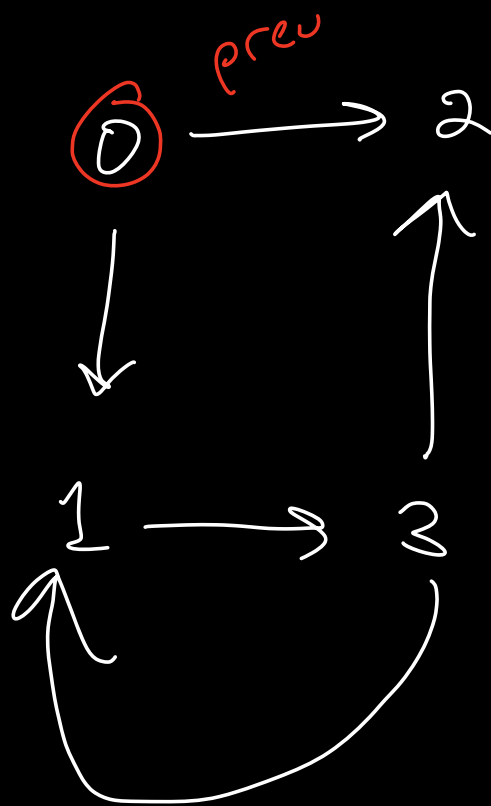


4's children:  
 $(1, 2)$

1: seen  $[False, 1]$

2: seen  $[False, 1]$





3: seen [False, 2]

if course in queue:

if seen[course][1]

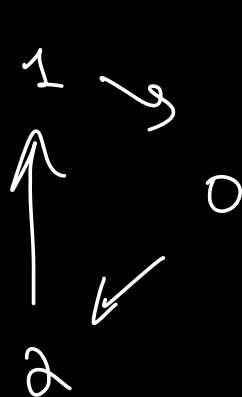
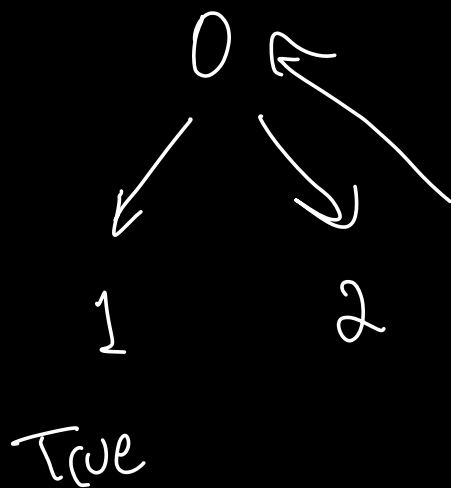
z = level : return

Q: 0 S: 0 <sup>prev</sup>

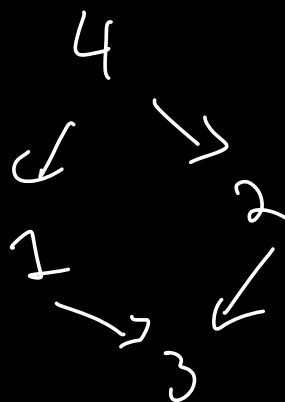
Q: 0, 1, 2 S: 0, 1, 2

Q: 3 S: 0, 1, 2, 3

Q: 1, 2 S: 0, 1, 2, 3



$[0, 2, 1]$   
false



$[1, 3]$

$[2, 3]$

$[4, 1, 2, 3]$