< Question > : Given N courses with pre-requisite of every course.

Check if it is possible to finish all the courses.

N = 5

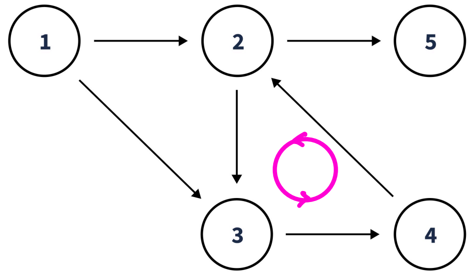
x is a pre-requisite of y $\quad x \rightarrow y$

$1 \rightarrow 2, 3$

$2 \rightarrow 3, 5$

$3 \rightarrow 4$

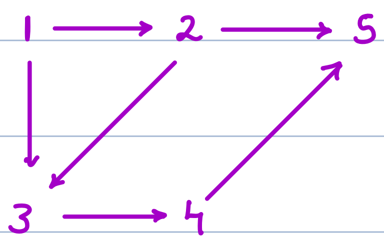
$4 \rightarrow 2$



Ans = false

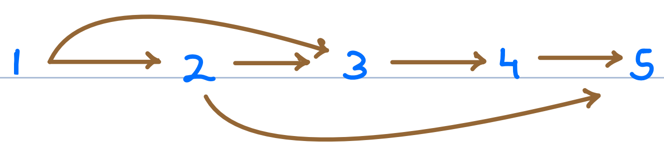
Sol → If cycle is present ⇒ false
else ⇒ true.

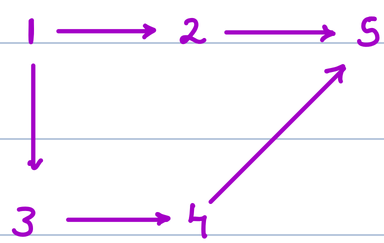
x is a pre-requisite of y $\quad x \rightarrow y$



Ans = true

1 2 3 4 5





1 2 3 4 5
1 3 4 2 5

} multiple order possible.

Topological Sort / Order

Linear ordering of the nodes such that if there is an edge from i to j, then i should be present on L.H.S of j $i \rightarrow j$ DAG [Directed Acyclic Graph]

Find topological order →

N = 7

0 → { 1 , 3 }

1 → { 2 , 3 }

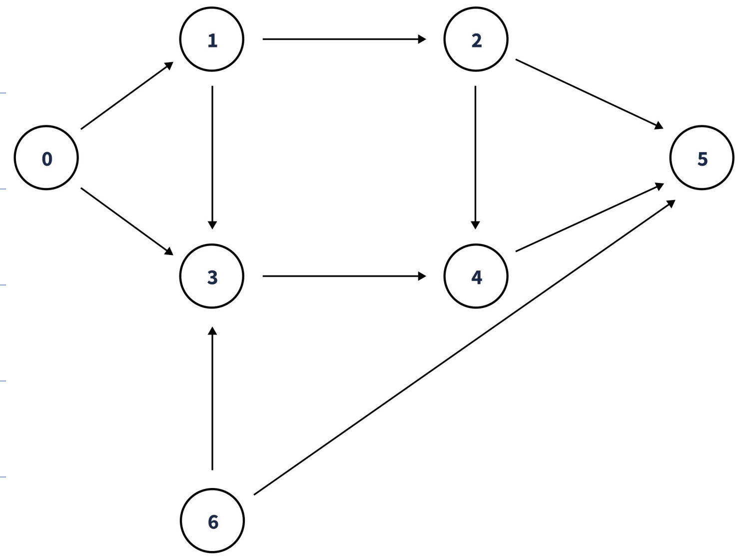
2 → { 4 , 5 }

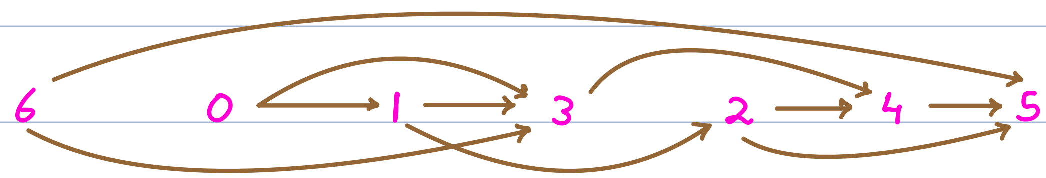
3 → { 4 }

4 → { 5 }

5 → { - }

6 → { 3 , 5 }





Left to Right

How to decide first node ?

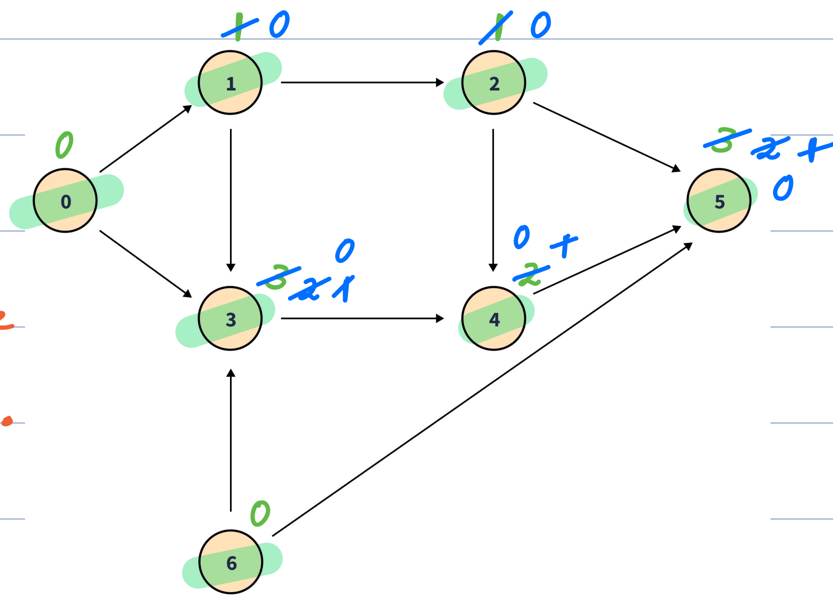
Node with indegree = 0.

Find indegree $\forall$ nodes

$\forall i,\ in[i] = 0$

```
for u → 0 to (N-1) {
    for (v : Adj[u]) {   // u → v
        in[v]++
    }
}
```

TC = O(N + E)



1) Select all nodes with indegree 0 & store in a array/set/queue.

2) Select any one node from array, print it (o/p).

6 0 1 2 3 4 5

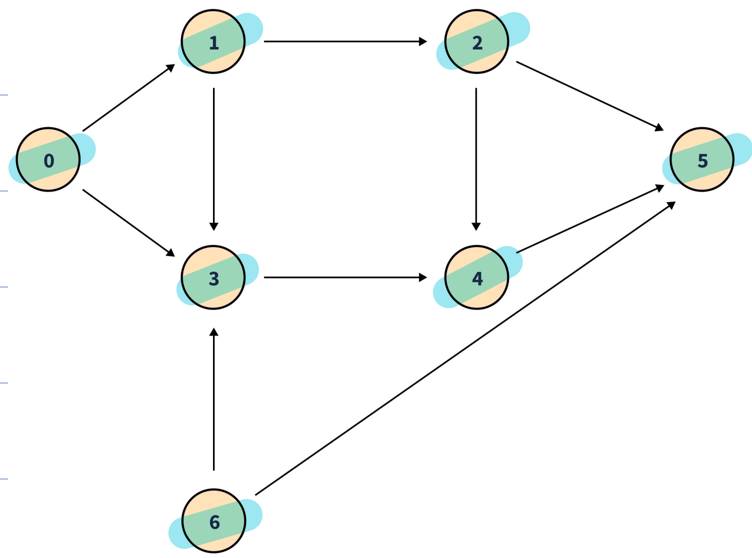
3) Decrease the indegree of adjacent nodes by 1, if updated indegree = 0, insert in array/set, & repeat from step 2 till all nodes are travelled.

TC = O(N + E)

SC = O(N)

Right to Left

Start DFS from any node (till all nodes are visited).



4 → 5

1 → 2, 3

6

0

O/P → 5 4 2 3 1 6 0

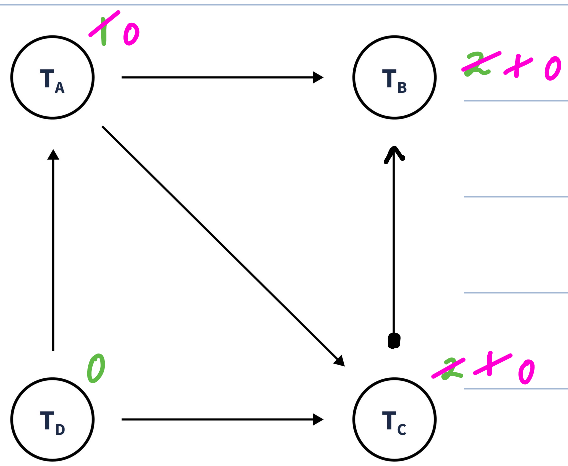
$\forall i$, vst[i] = false

```
for i → 0 to (N-1) {
    if (!vst[i]) dfs(i)
}
```

```
void dfs(u) {
    vst[u] = true
    for (v : Adj[u]) {      u → v
        if (!vst[v]) dfs(v)
    }
    print(u)
}
```

TC = $O(N + E)$ SC = $O(N)$

Quiz :



T_D T_A T_C T_B

a) T_D , T_A , T_C , T_B ✓

b) T_A , T_D , T_C , T_B

c) T_C , T_A , T_D , T_B

Minimum jumps to reach end

You are given a 0-indexed array of integers arr of length n. You are initially positioned at arr[0].

Each element arr[i] represents the maximum length of a forward jump from index i. In other words, if you are at arr[i], you can jump to any arr[i + j] where:

* 0 <= j <= arr[i]
* i + j < n

Return the minimum number of jumps to reach arr[n - 1]. The test cases are generated such that you can reach arr[n - 1].

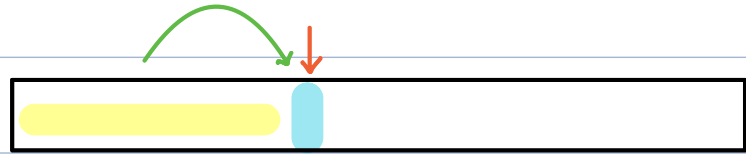
Example 1
Input: arr = [2,3,1,1,4]
Output: 2
Explanation: The minimum number of jumps to reach the last index is 2. Jump 1 step from index 0 to 1, then 3 steps to the last index.

Example 2
Input: arr = [2,3,0,1,4]
Output: 2

$$\begin{array}{cccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ A = & [\,2 & 3 & 1 & 2 & 0 & 5 & 1\,] \end{array}$$

Ans = 4

Next element from A[i] ⇒ A[i] <= element <= A[i + A[i]]



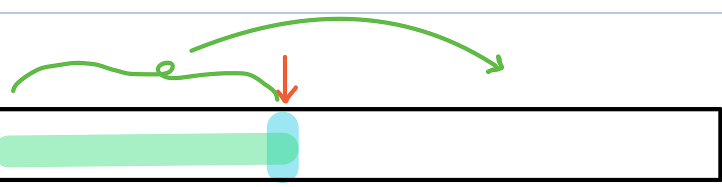
Sol. 1 → ∀ nodes, find min jumps to reach the node.

	0	1	2	3	4	5	6
A =	[2	3	1	2	0	5	1]
jump →	0	1	1	2	2	3	4

Ans = 4

```
if (A[0] == 0) return INT_MAX
∀ i, jump[i] = INT_MAX
jump[0] = 0
for i → 1 to (N-1) {
    for j → 0 to (i-1) {
        if ( j + A[j] >= i  &&  jump[j] != INT_MAX )
            jump[i] = min (jump[i], jump[j]+1)
    }
} return jump[N-1]
```

TC = $O(N^2)$ SC = $O(N)$



Sol. 2 → $\forall i$, find max node we can reach considering nodes from 0 to i.

$$i \longrightarrow i + A[i]$$

```
d[0] = A[0]
for i → 1 to (N-1) {
    d[i] = max( i + A[i], d[i-1] )
}
```

	0	1	2	3	4	5	6
A =	2	3	1	2	0	5	1
d →	2	4	4	5	5	10	10

Ans = 4

```
ans = 0
i = 0
while (i < N) {
    ans++
    i = i + d[i]
}
return ans
```

TC = O(N) ✓

SC = O(N) ⟶ O(1)

d[] ⟶ A[]

Max profit from stock prices

Given an array A where the i-th element represents the price of a stock on day i, the objective is to find the maximum profit. We're allowed to complete as many transactions as desired, but engaging in multiple transactions simultaneously is not allowed.

b s b s . ✓ .   b b s s ✗

	0	1	2	3	4	5	6
A =	2	3	1	2	0	5	1
min on left →	2	2	1	1	0	0	0

Only 1 transaction → $\text{Ans} = \max\limits_{\forall i} (A[i] - \text{min on left})$

Multiple transactions

	0	1	2	3	4	5	6
A =	2	3	1	2	0	5	1

1 + 1 + 5 = 7 (Ans)

	0	1	2	3	4	5	6	7	8	9
A =	5	2	3	5	7	6	2	4	1	0

5 + 2 = 7 (Ans)

	0	1	2	3	4	5	6	7	8	9
A =	5 ✗	2 ✓	3 ✓	5 ✓	7 ✗	6 ✗	2 ✓	4 ✗	1 ✗	0

1 + 2 + 2 = 5   2

```
profit = 0
for i → 1 to (N-1) {
    if ( A[i-1] < A[i] )    profit += A[i] - A[i-1]
}

return profit
```

TC = O(N) SC = O(1)