Sample Input => numCourses = 4
prerequisites = [[1, 0], [2, 1], [3, 2]]

Initialization:
adjList = [[ ], [ ], [ ], [ ]]
indegree = [0, 0, 0, 0]
topOrder = [ ] → Store the final order of courses
queue = deque() → Processing courses with 0 in-degree

STEP 1: Build an adjacency list & compute indegree

1. Prerequisite = [1, 0] in prerequisite list
Course => 1 prereq → 0, update adjList & indegree array
adjList = [[1], [ ], [ ], [ ]] # 0 → 1
indegrees = [0, 1, 0, 0]

② Prerequisite = [2, 1] in given prereq list
Course = 2 prereq = 1, update adjList & indgree array
adjlist = [[1], [2], [ ], [ ]] # 1 → 2
indegrees = [0, 1, 1, 0]

③ Prerequisite = [3, 2] in given prereq list
Course = 3 prereq = 2, update adjlist & indegree array
adjlist = [[1], [2], [3], [ ]] # 2 → 3
indegree = [0, 1, 1, 1]

STEP 2: Initialize queue with 0 indgree course
0 → 1 → 2 → 3
queue = [0] Add '0' to topOrder as well
topOrder = [0]

STEP 3: Process Nodes in BFS manner

① Deque '0' from q queue = [ ]
process neighbor of '0' topOrder = [0]
so decrement indgree of 1 indegree = [0, 0, 1, 1]
- Indegree of 1 is 0 now queue = [1]
So add it to queue

2. • Deque 1 from queue
   • add 1 to toporder
   • process neighbor of '1'
   • Update indegree (decrement)
   • If indegree '0' add to queue.

toporder = [0, 1]
indegrees = [0, 0, 0, 1]
queue = [2]

3. Deque 2 from queue
   • add 2 to toporder
   • process '3' (neighbor) of '2'
   • Update indegree (decrement)
   • If indegree '0' add to q

queue = [2]
toporder = [0, 1, 2]
indegree = [0, 0, 0, 0]
queue = [3]

④ Deque 3 from queue
   • add 3 to toporder
   • Update indegree & queue

queue = []
toporder = [0, 1, 2, 3]
indegree = [0, 0, 0, 0]

Conclusion: length of toporder = 4, which is equal to numCourses. (NO cycles in graph)

Return True

All courses can be finished.

Time ⇒ O(V+E)

Space ⇒ O(V+E)