Find if Path Exists in Graph

There is a bi-directional graph with n vertices, where each vertex is labeled from 0 to n - 1 (inclusive). The edges in the graph are represented as a 2D integer array edges, where each edges[i] = [ui, vi] denotes a bi-directional edge between vertex ui and vertex vi. Every vertex pair is connected by at most one edge, and no vertex has an edge to itself.

You want to determine if there is a valid path that exists from vertex start to vertex end.

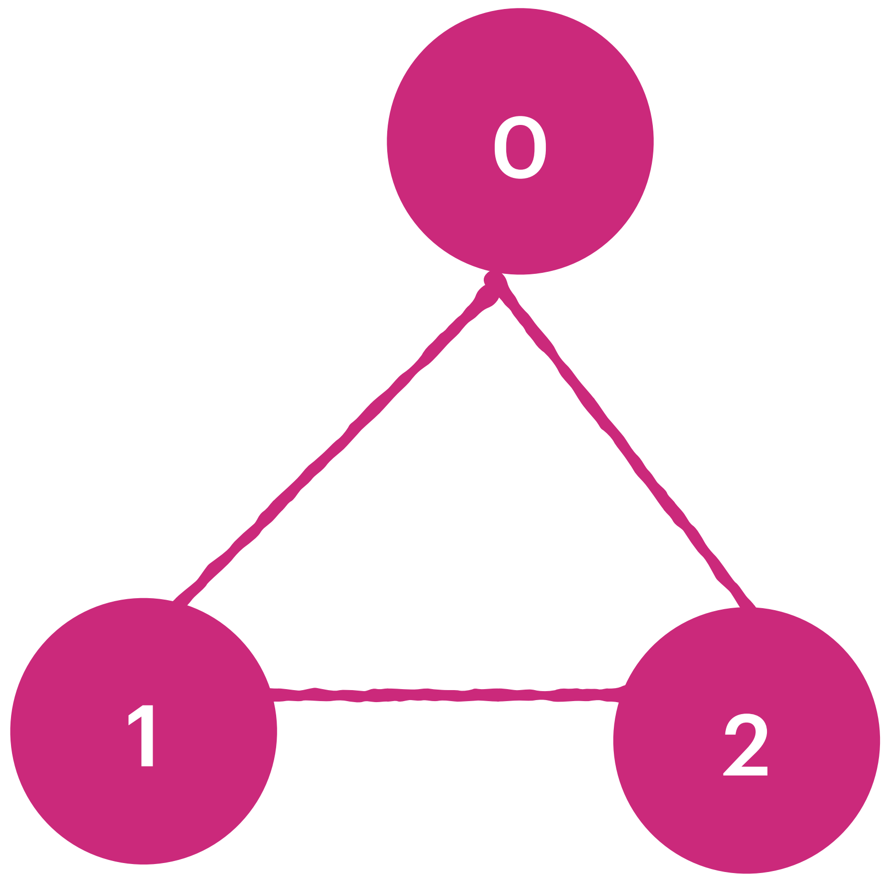
Given edges and the integers n, start, and end, return true if there is a valid path from start to end, or false otherwise.

Input: n = 3, edges = [[0,1],[1,2],[2,0]], start = 0, end = 2
Output: true
Explanation: There are two paths from vertex 0 to vertex 2:
- 0 → 1 → 2
- 0 → 2

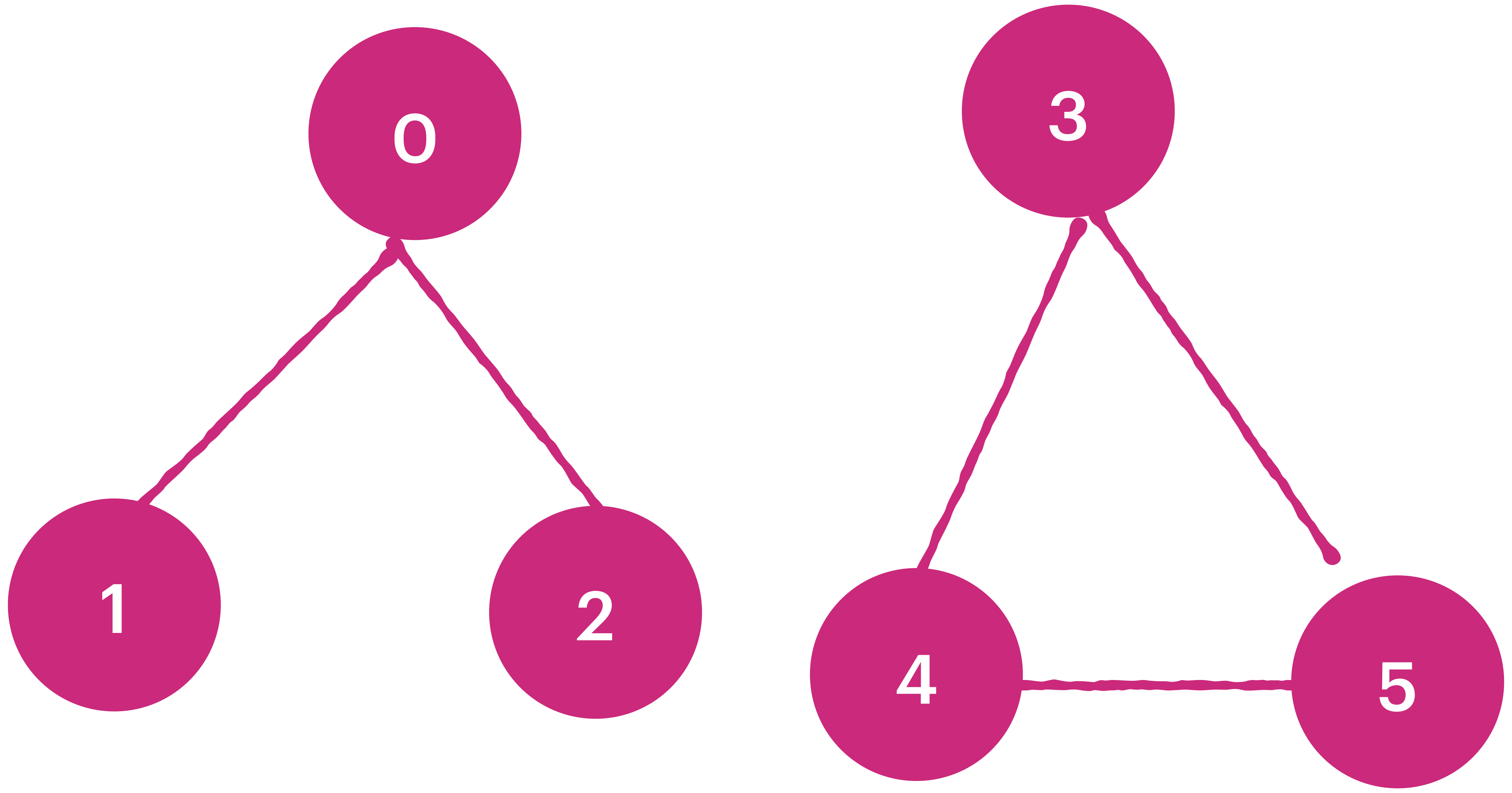
Input: n = 6, edges = [[0,1],[0,2],[3,5],[5,4],[4,3]], start = 0, end = 5
Output: false
Explanation: There is no path from vertex 0 to vertex 5.

Constraints:
$1 <= n <= 2 * 10^5$
$0 <= edges.length <= 2 * 10^5$
edges[i].length == 2
$0 <= u_i, v_i <= n - 1$
$u_i != v_i$
0 <= start, end <= n - 1
There are no duplicate edges.
There are no self edges.

Input: n = 3, edges = [[0,1],[1,2],[2,0]], start = 0, end = 2
Output: true
Explanation: There are two paths from vertex 0 to vertex 2:
- 0 → 1 → 2
- 0 → 2



Solution 1 DisJoint Set : O(1) :
Solution 2. DFS : O(V+E)
Solution 3 BFS : O(V+E)



Input: n = 6, edges = [[0,1],[0,2],[3,5],[5,4],[4,3]], start = 0, end = 5
Output: false
Explanation: There is no path from vertex 0 to vertex 5.

All Paths From Source to Target

Given a directed acyclic graph (DAG) of n nodes labeled from 0 to n - 1,
find all possible paths from node 0 to node n - 1 and return them in any order.
The graph is given as follows: graph[i] is a list of all nodes you can visit from node i
(i.e., there is a directed edge from node i to node graph[i][j]).

Input: graph = [[1,2],[3],[3],[]]
Output: [[0,1,3],[0,2,3]]
Explanation: There are two paths: 0 -> 1 -> 3 and 0 -> 2 -> 3.

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

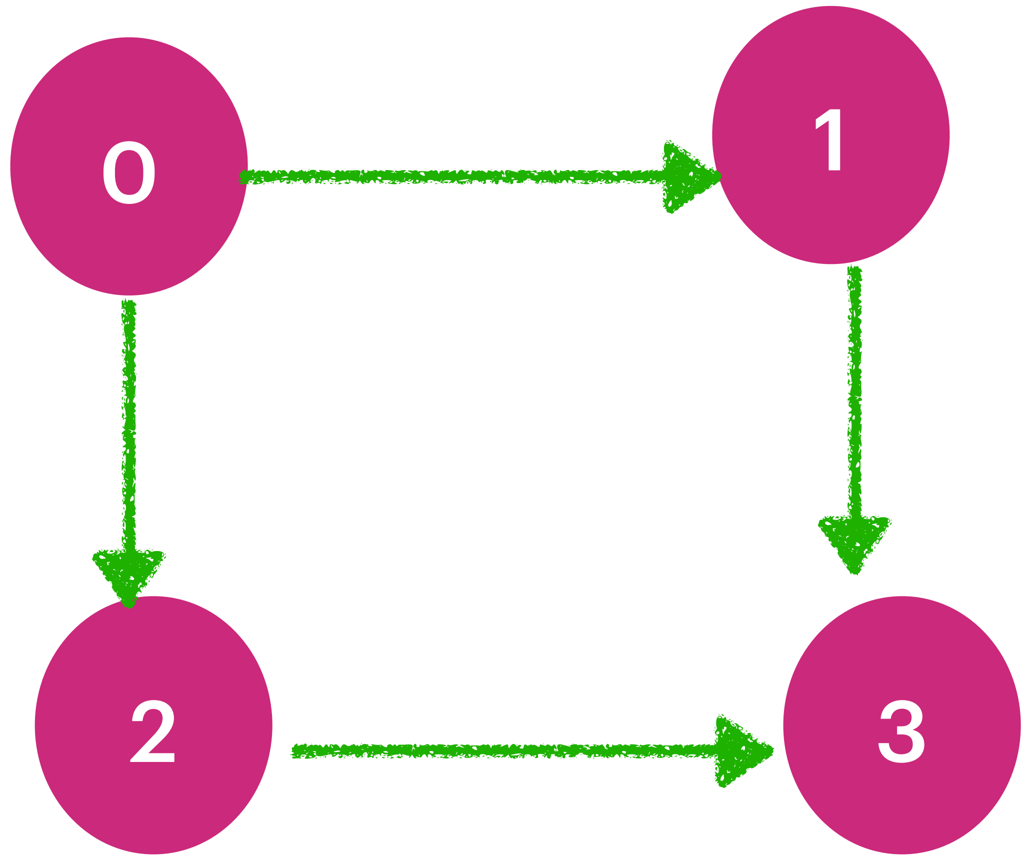
n == graph.length
2 <= n <= 15
0 <= graph[i][j] < n
graph[i][j] != i (i.e., there will be no self-loops).
All the elements of graph[i] are unique.
The input graph is guaranteed to be a DAG.

All Paths From Source to Target

Input: graph = [[1,2],[3],[3],[]] n=4
Output: [[0,1,3],[0,2,3]]
Explanation: There are two paths: 0 -> 1 -> 3 and 0 -> 2 -> 3.

0 1 2 3

graph = [[1,2], [3], [3], []] n=4



Paths From 0 to 3

0 —> 1 —> 3 , 0—> 2 —> 3

[0]

[0,1] [0,2]

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

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

[0,2,3]

Queue: Empty

[0]

[0,1]

[0,2]

[0,1,3]

Last Element is Equals to Target.

[0,2,3]

Last Element is Equals to Target.

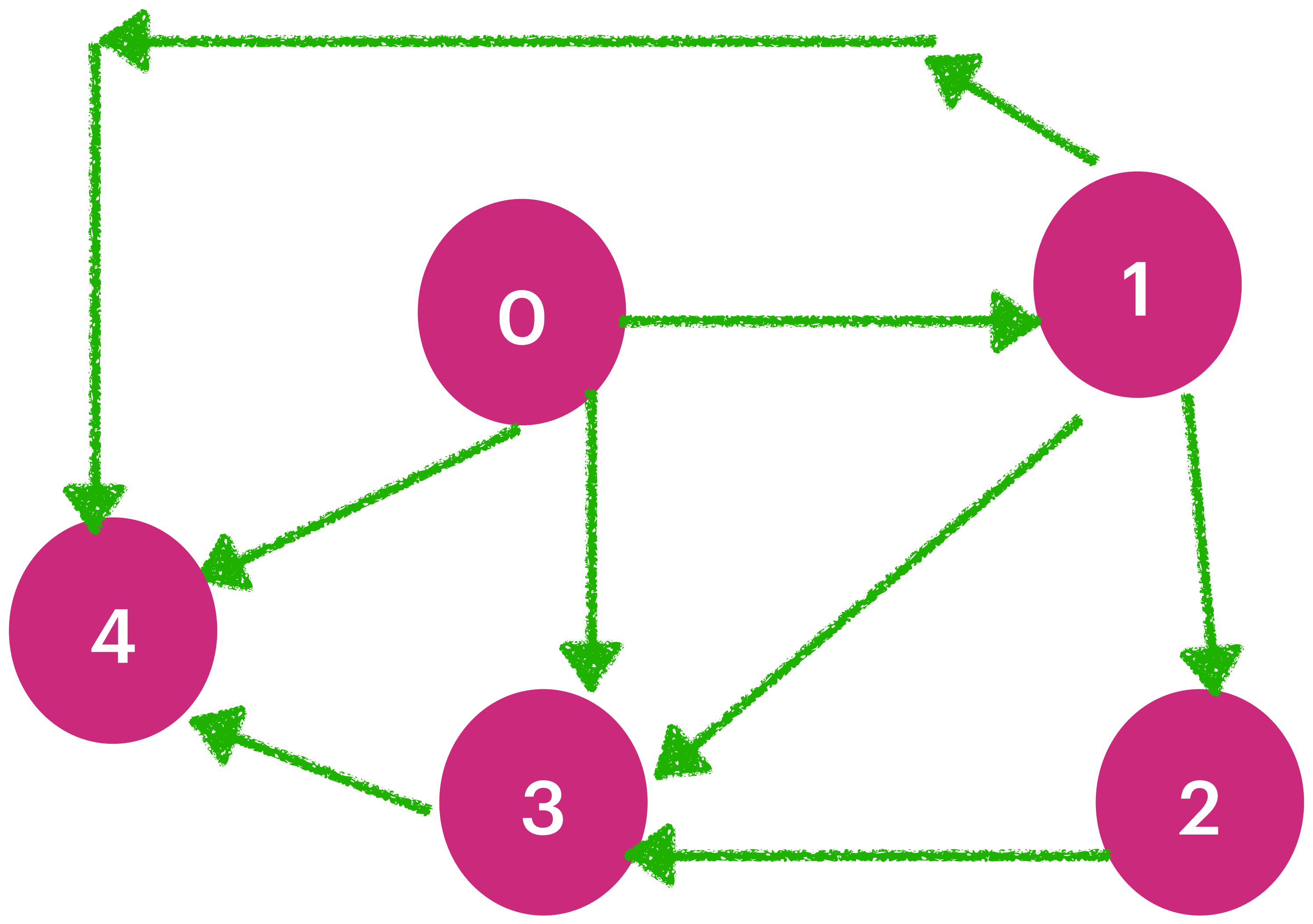
All Paths From Source to Target

n == graph.length
2 <= n <= 15
0 <= graph[i][j] < n
graph[i][j] != i (i.e., there will be no self-loops).
All the elements of graph[i] are unique.
The input graph is guaranteed to be a DAG.

0 1 2 3 4

graph = [[4,3,1], [3,2,4], [3], [4], []] n=5

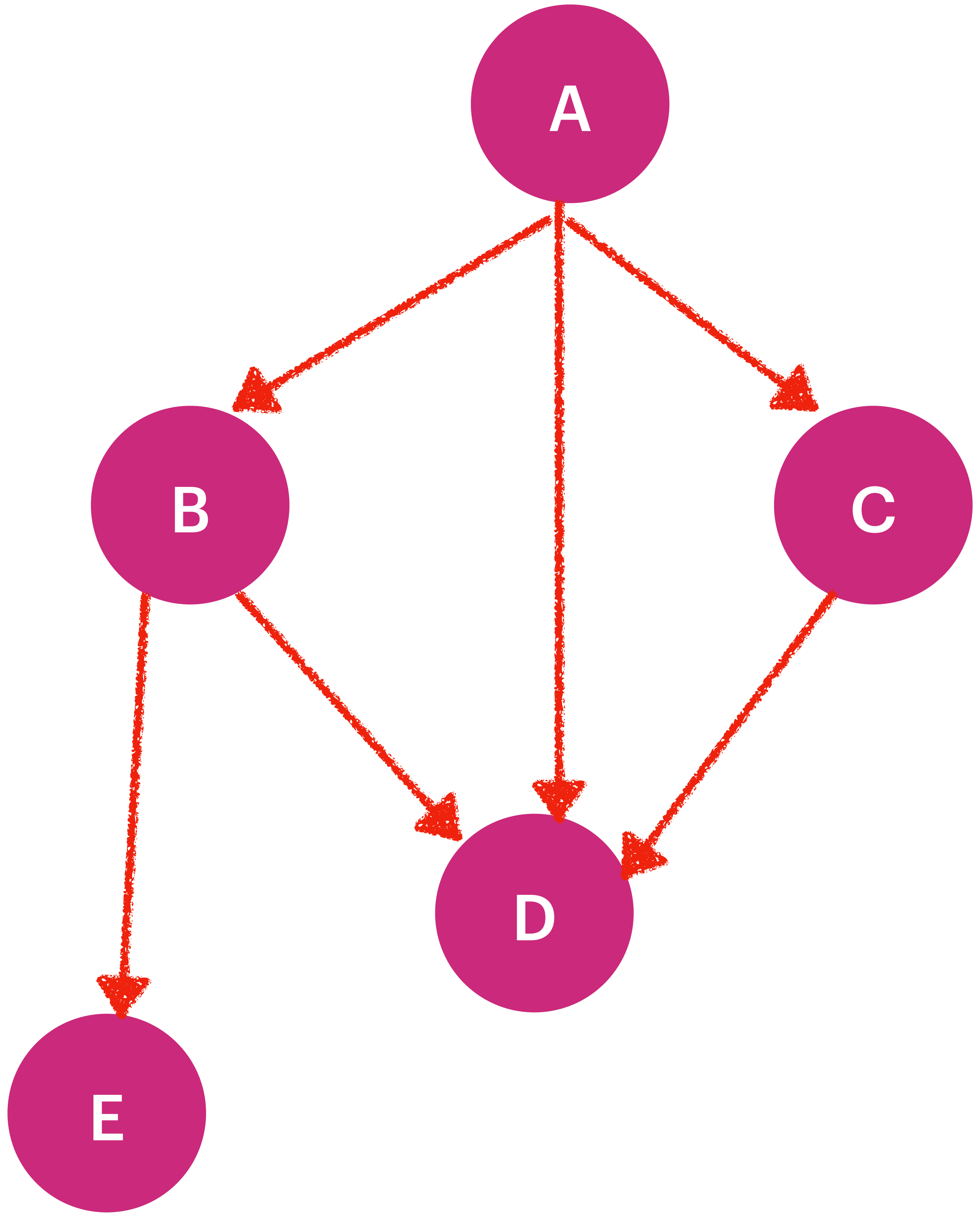
Input: graph = [[4,3,1],[3,2,4],[3],[4],[]] n=5
Output: [[0,4],[0,3,4],[0,1,3,4],[0,1,2,3,4],[0,1,4]]



Paths From 0 to 4

0->4 , 0->3->4, 0->1->3->4, 0->1->2->3->4, 0->1->4

BFS
Source : A
Target : D



[A] [A,B] [A,C] [A,D] [A,B,D] [A,B,E] [A,C,D]

Last Element is Equals to Target.

Last Element is Equals to Target.

Last Element is Equals to Target.