Graph Valid Tree

You have a graph of n nodes labeled from 0 to n - 1. You are given an integer n and a list of edges where edges[i] = [ai, bi] indicates that there is an undirected edge between nodes ai and bi in the graph. Return true if the edges of the given graph make up a valid tree, and false otherwise.

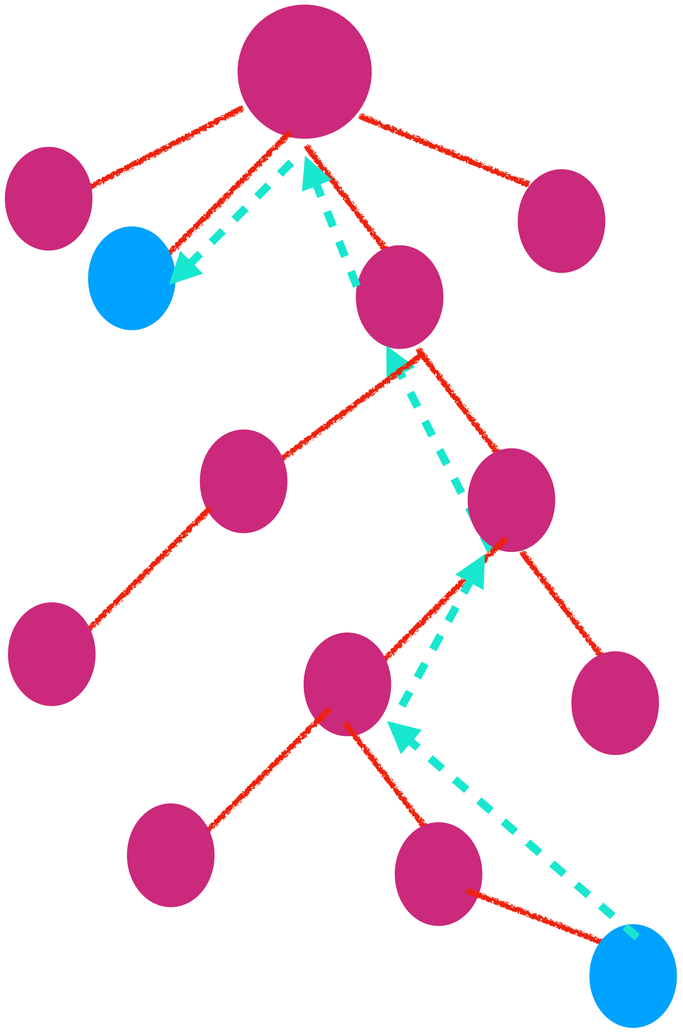
Input: n = 5, edges = [[0,1],[0,2],[0,3],[1,4]]
Output: true

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

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

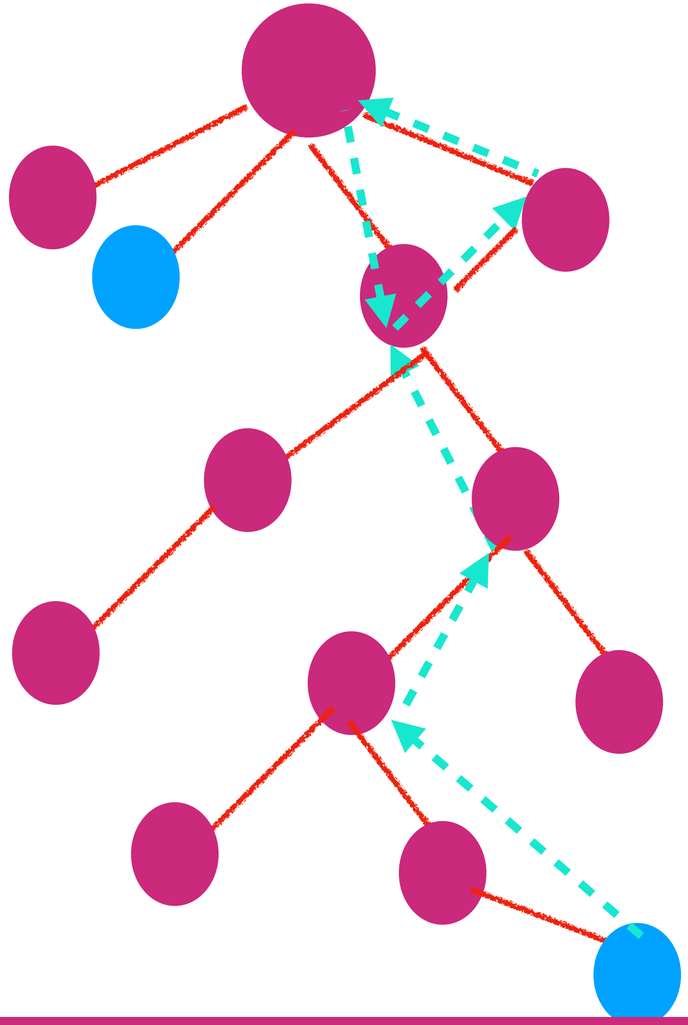
Constraints :
1 <= n <= 2000
0 <= edges.length <= 5000
edges[i].length == 2
0 <= ai, bi < n
ai != bi (No Self loop)

Graph with Valid Tree



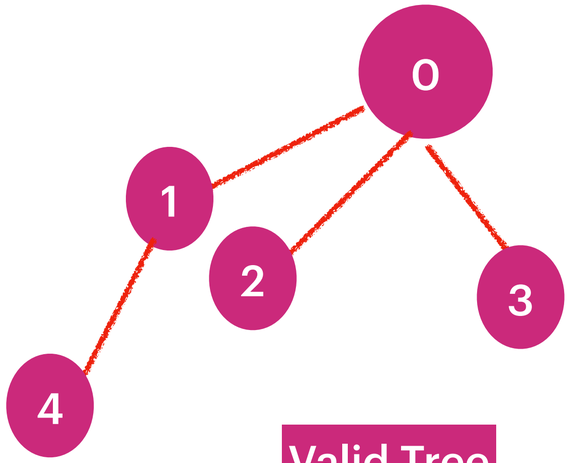
Its a valid tree because
From one branch we can move to another branch

Graph with Invalid Tree



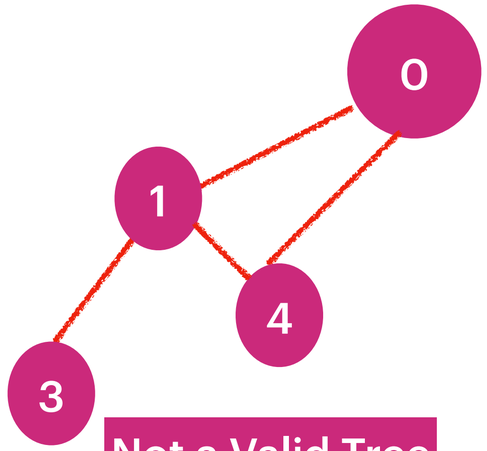
Its not a valid tree because
, a child node has more than one immediate parent so it causes cycle.

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



Valid Tree

Input: n = 5,
edges = [[0,1],[0,4],[1,4],[1,3]]
Output: false

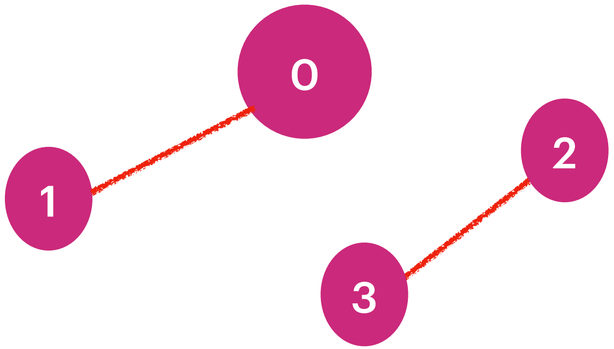


Not a Valid Tree

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

For n vertexes we can have at max n-1 edges.
Here n = 5 and edges = 5 so Not a Valid Tree.

n=4 [[0,1], [2,3]]



Its not a Valid Tree because

[[0,1],[0,4],[1,4],[1,3]] n = 5

0[0]
1[1]
2[2]
3[3]
4[4]

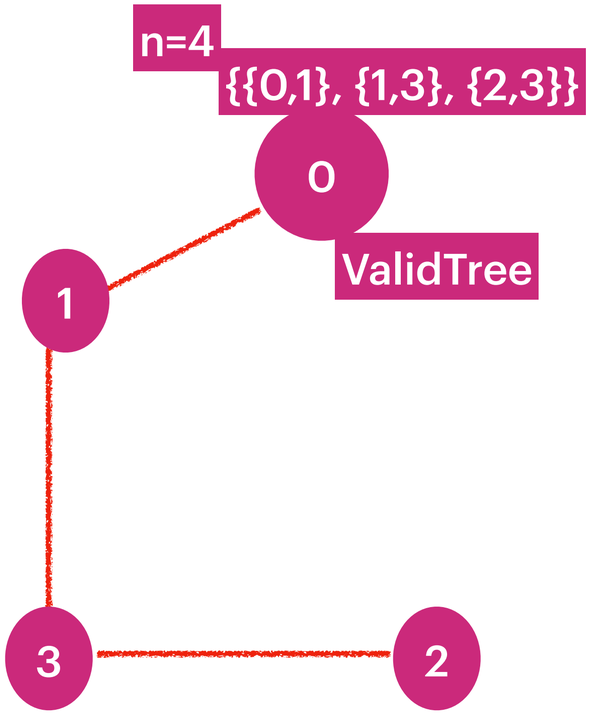
[0,1] => 0[0]-1[0]

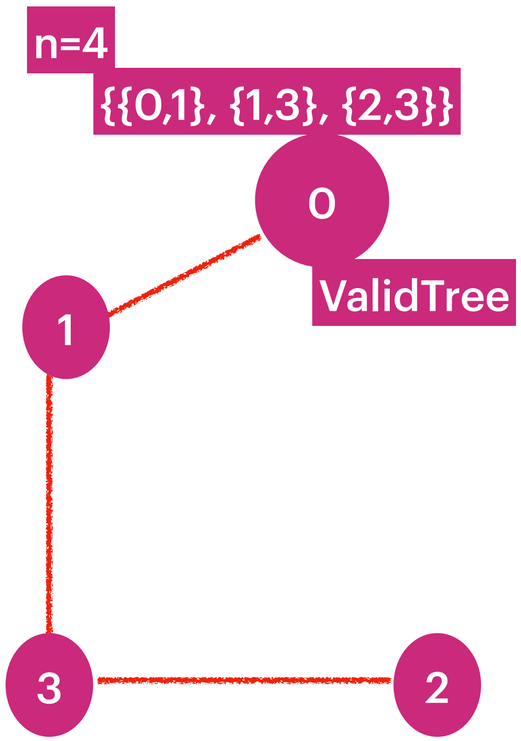
[0,4] => 0[0]-1[0]-4[0]

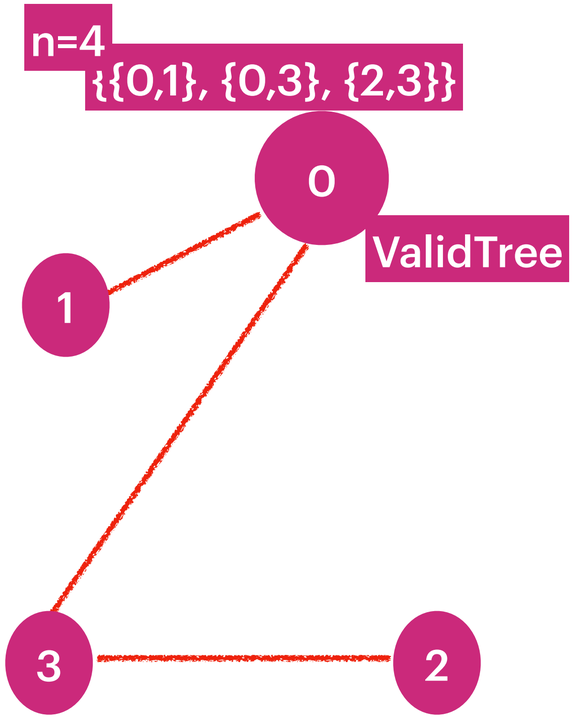
[1,4] => 1[0] - 4[0] They are already connected : Its a loop return false

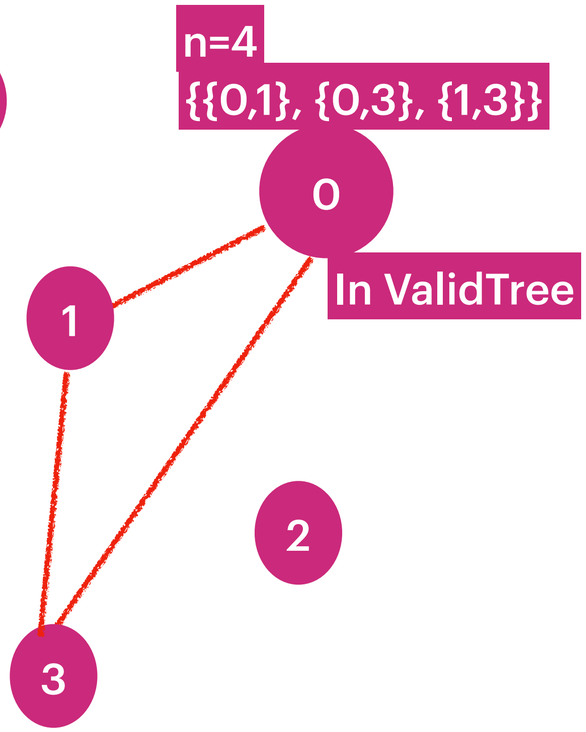
Input: n = 5, edges = [[0,1],[0,2],[0,3],[1,4]]

0[0]
1[1]
2[2]
3[3]
4[4]
[0,1] => 0[0]-1[0]
[0-2] => 0[0]-1[0]- 2[0]
[0-3] => 0[0]-1[0]- 2[0]-3[0]
[1-4] =>0[0]-1[0]- 2[0]-3[0]-4[0] Its a valid Tree : return true


n=4
{{0,1}, {1,3}, {2,3}}
0
ValidTree
1
3
2


n=4
{{0,1}, {1,3}, {2,3}}
0
ValidTree
1
3
2


n=4
{{0,1}, {0,3}, {2,3}}
0
ValidTree
1
3
2


n=4
{{0,1}, {0,3}, {1,3}}
0
In ValidTree
1
2
3

Number of Connected Components in an Undirected Graph

You have a graph of n nodes. You are given an integer n and an array edges where edges[i] = [ai, bi] indicates that there is an edge between ai and bi in the graph.

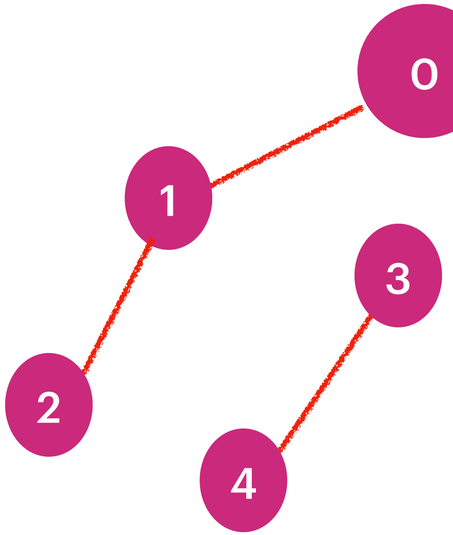
Return the number of connected components in the graph.

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

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

Constraints : 1 <= n <= 2000
1 <= edges.length <= 5000
edges[i].length == 2
0 <= ai <= bi < n
ai != bi
There are no repeated edges.

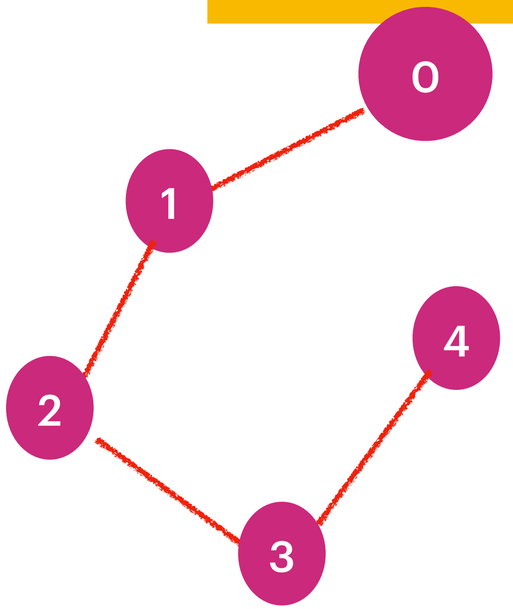
Input: n = 5, edges = [[0,1],[1,2],[3,4]]
Output: 2



edgeCount = 5

[0,1] = edgeCount - - = 4
[1,2] = edgeCount - - = 3
[3,4] = edgeCount - - = 2

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



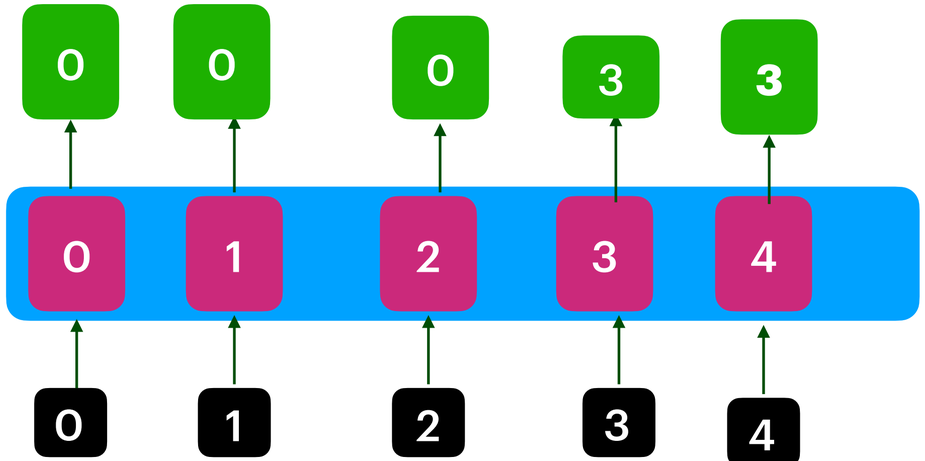
edgeCount = n = 5
edgeCount
[0,1] = edgeCount- - = 4

[1,2] = edgeCount- - = 3

[2,3] = edgeCount- - = 2

[3,4] = edgeCount- - = 1

Connected Components / Paths. = 2



Connected Components / Paths. = 1