

**Muhammad Tanveer**

Other

PDF generated at: 23 Jul 2024 03:28:51 UTC

[View this report on HackerRank](#)**Score**

28.1% • 90 / 320

scored in CodePath TIP103: Unit 6 Assessment - Summer 2024 in 47 min 32 sec on 22 Jul 2024 19:39:17 PDT

Candidate Information

Email	tanveerm176@gmail.com
Test	CodePath TIP103: Unit 6 Assessment - Summer 2024
Candidate Packet	View
Taken on	22 Jul 2024 19:39:17 PDT
Time taken	47 min 32 sec / 90 min
Personal Email Address	tanveerm176@gmail.com
Invited by	CodePath

Skill Distribution

There is no associated skills data that can be shown for this assessment





Tags Distribution



There is no associated tags data that can be shown for this assessment

Questions

Status	No.	Question	Time Taken	Skill	Score
✓	1	Queue to keep track of vertices Multiple Choice	5 min 4 sec	-	5/5
✓	2	Stack to keep track of vertices Multiple Choice	47 sec	-	5/5
✗	3	Number of operations needed to loop through edges Multiple Choice	5 min 55 sec	-	0/5
✓	4	DFS Graph Traversal Multiple Choice	1 min 8 sec	-	5/5
✗	5	BFS Graph Traversal Multiple Choice	17 sec	-	0/5

	6	DFS Bug Coding	13 min 18 sec	-	15/75
	7	Walls and Gates Coding	6 min 14 sec	-	60/80
	8	Connected Components in Undirected Graph Coding	7 min 19 sec	-	0/70
	9	Graph Valid Tree Coding	7 min 23 sec	-	0/70

1. Queue to keep track of vertices

 Correct

Multiple Choice

Question description

What graph traversal algorithm uses a queue to keep track of vertices which need to be processed?

Candidate's Solution

Options: (Expected answer indicated with a tick)



Breadth-first search



Depth-first search

☐ Level order search☐ None of the above

⚠ No comments.

2. Stack to keep track of vertices

✓ Correct

Multiple Choice

Question description

Which graph traversal algorithm uses a stack to keep of vertices which need to be processed?

Candidate's Solution

Options: (Expected answer indicated with a tick)

☐ Breadth-first search☒ Depth-first search

✓

☐ Level order search☐ None of the above

⚠ No comments.

3. Number of operations needed to loop through edges

✖ Incorrect

Multiple Choice

Question description

What is the expected number of operations needed to loop through all the edges terminating at a particular vertex given an adjacency list representation of the graph?

(Assume n vertices are in the graph and m edges terminate at the desired node.)

Candidate's Solution

Options: (Expected answer indicated with a tick)

☐ $O(m)$



☐ $O(n)$

☐ $O(m^2)$

☒ $O(n^2)$

☐ None of the above

⚠ No comments.

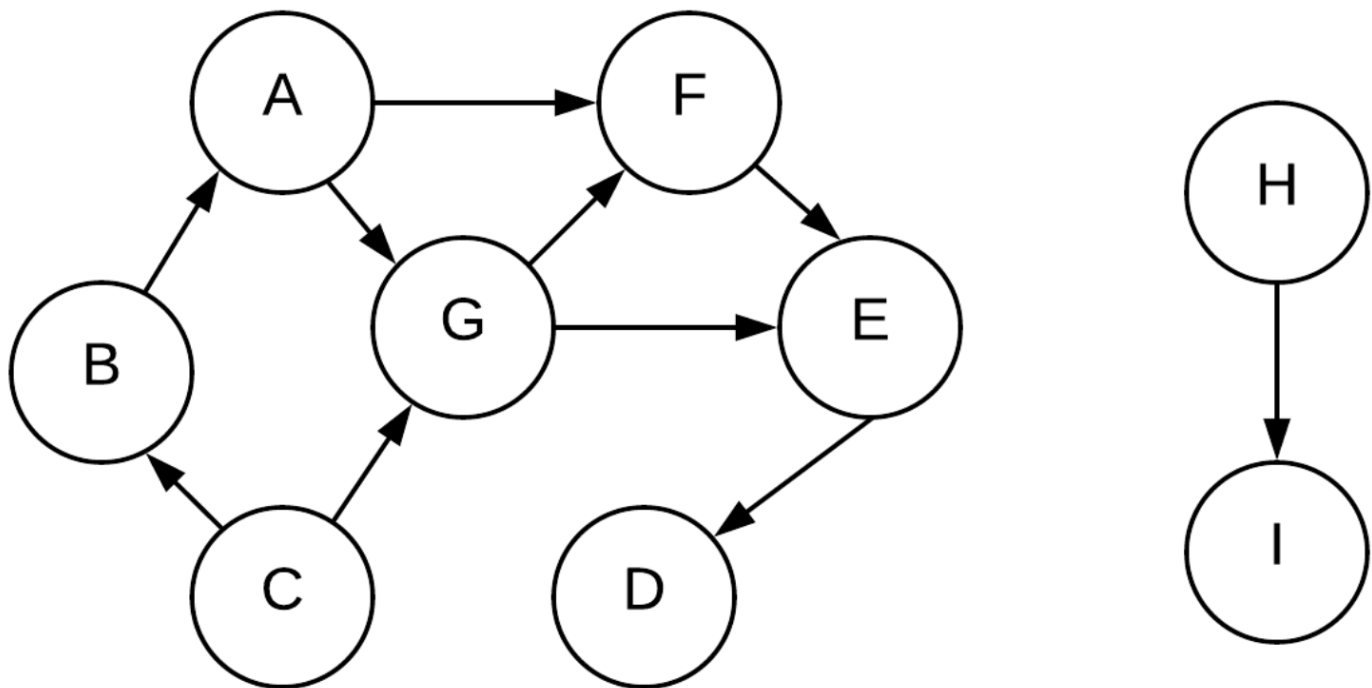
4. DFS Graph Traversal

✓ Correct

Multiple Choice

Question description

Given this graph, answer the following questions:



What is the result of running preorder DFS starting on node C?

*Note: ties are broken alphabetically, so if node A had both node B and node C as neighbors, node B would be visited first.

Candidate's Solution

Options: (Expected answer indicated with a tick)



CBAFEDG



CGFEDBA



CBAGEDF



CGFEDBAHI



No comments.

5. BFS Graph Traversal

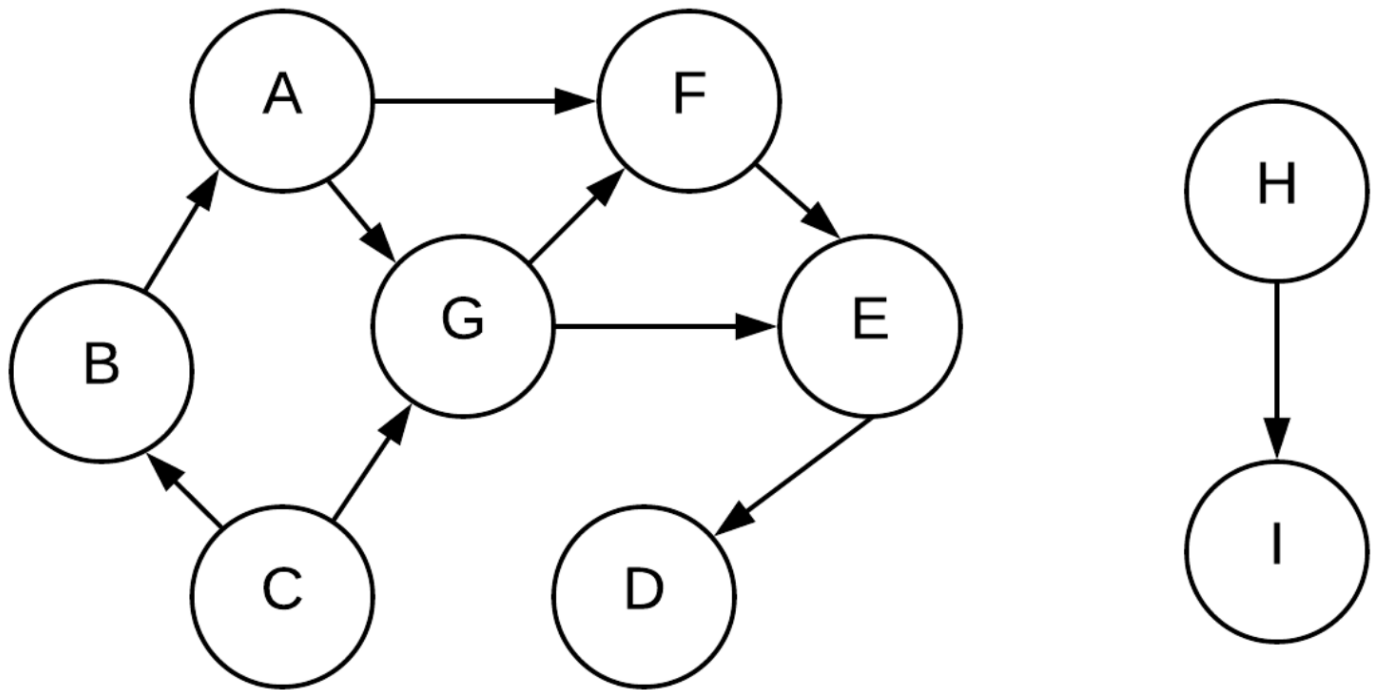


Incorrect

Multiple Choice

Question description

Given this graph, answer the following questions:



What is the result of running BFS, using C as the root node?

*Note: ties are broken alphabetically, so if node A had both node B and node C as neighbors, node B would be added to the stack first

Candidate's Solution

Options: (Expected answer indicated with a tick)

☐ CBGAEFDHI

☐ CBGAEFD



☒ CBGEFDAHI



⚠ No comments.

6. DFS Bug

✅ Partially correct

Coding

Question description

The following code is meant to run a DFS on a directed graph, but there's a bug. Fix this code snippet so that it is a proper DFS function!

If you're trying to understand how the test cases / inputs work, you can analyze the code outside of the function you're trying to implement to see how the input string is parsed to create the graph.

Candidate's Solution

Language used: Python 3

```
1 import fileinput
2
3 args = []
4 for line in fileinput.input():
5     args.append(line)
6
7 adjList = {}
8 for i in range(1, len(args) - 1):
9     nodes = args[i].split()
10    adjList[nodes[0]] = nodes[1:]
11 start = args[len(args) - 1]
12
13 '''
14 Assuming this adjacency list graph structure and that a node with no outgoing
15 edges will not be included in the graph
16 graph = {'A': ['B', 'C'],
17          'B': ['D', 'E'],
18          'C': ['F'],
```

```

18         'E': ['F']}
19     ...
20
21 def dfs(graph, start):
22     visited, stack = list(), [start]
23     while stack:
24         vertex = stack.pop()
25         if vertex not in visited:
26             if vertex in graph:
27                 neighbors = graph[vertex]
28                 unvisited = [n for n in neighbors if n not in visited]
29                 stack.extend(unvisited)
30                 visited.append(vertex)
31             else:
32                 visited.append(vertex)
33     return visited
34
35 def printArray(arr):
36     print('[', end='')
37     for i in range(len(arr) - 1):
38         print(arr[i], end=', ')
39     print(arr[len(arr) - 1], end='')
40     print(']', end='')
41
42 printArray(dfs(adjList, start))

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample	Wrong Answer	0	0.0304 sec	10.4 KB
Testcase 1	Easy	Hidden	Success	5	0.0372 sec	10.2 KB
Testcase 2	Easy	Hidden	Wrong Answer	0	0.0358 sec	10.3 KB
Testcase 3	Easy	Hidden	Wrong Answer	0	0.0338 sec	10.4 KB

Testcase 4	Easy	Hidden	Wrong Answer	0	0.0306 sec	10.4 KB
Testcase 5	Easy	Hidden	Wrong Answer	0	0.0341 sec	10.2 KB
Testcase 6	Easy	Hidden	Wrong Answer	0	0.0303 sec	10.3 KB
Testcase 7	Easy	Hidden	Wrong Answer	0	0.039 sec	10.3 KB
Testcase 8	Easy	Hidden	Wrong Answer	0	0.0317 sec	10.2 KB
Testcase 9	Easy	Hidden	Success	5	0.0307 sec	10.2 KB
Testcase 10	Easy	Hidden	Wrong Answer	0	0.03 sec	10.4 KB
Testcase 11	Easy	Hidden	Wrong Answer	0	0.0285 sec	10.4 KB
Testcase 12	Easy	Hidden	Success	5	0.0346 sec	10.3 KB
Testcase 13	Easy	Hidden	Wrong Answer	0	0.0313 sec	10.3 KB
Testcase 14	Easy	Hidden	Wrong Answer	0	0.0285 sec	10.3 KB

⚠ No comments.

7. Walls and Gates

✅ Partially correct

Coding

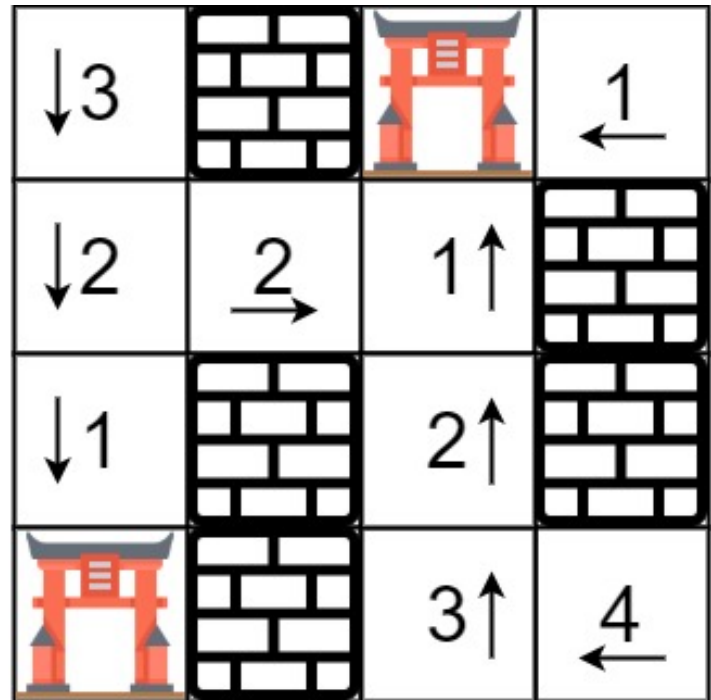
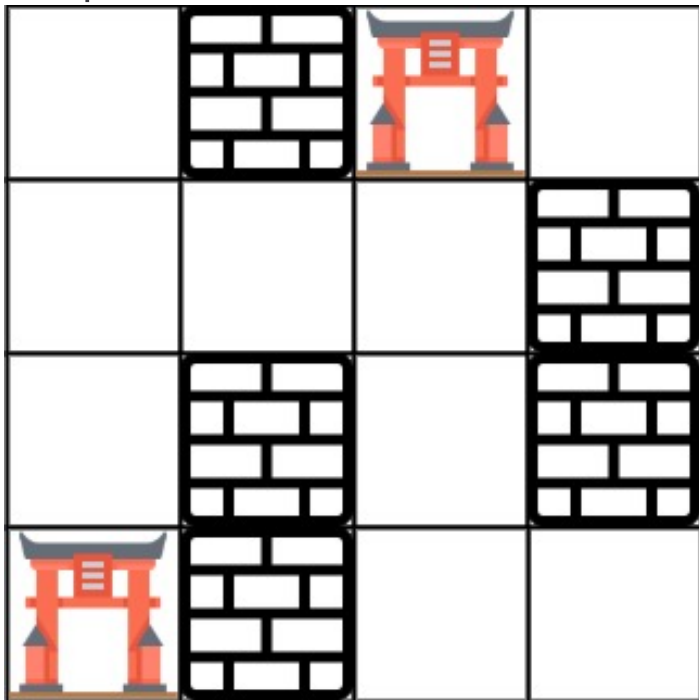
Question description

You are given an $m \times n$ grid rooms initialized with these three possible values.

- -1 A wall or an obstacle.
- 0 A gate.
- INF Infinity means an empty room. We use the value $2^{31} - 1 = 2147483647$ to represent INF as you may assume that the distance to a gate is less than 2147483647.

Fill each empty room with the distance to *its nearest gate*. If it is impossible to reach a gate, it should be filled with INF.

Example 1:



Input: rooms = [[2147483647,-1,0,2147483647],[2147483647,2147483647,2147483647,-1],[2147483647,-1,2147483647,-1],[0,-1,2147483647,2147483647]]
Output: [[3,-1,0,1],[2,2,1,-1],[1,-1,2,-1],[0,-1,3,4]]

Example 2:

Input: rooms = [[-1]]
Output: [[-1]]

Example 3:

Input: rooms = [[2147483647]]
Output: [[2147483647]]

Example 4:

Input: rooms = [[0]]
Output: [[0]]

Candidate's SolutionLanguage used: **Python 3**

```
1 import fileinput
2
3 args = []
4 for line in fileinput.input():
5     args.append(line)
6
7 numOfRows = int(args[0])
8
9 rooms = []
10
11 for line in args[1:]:
12     roomLine = [int(room) for room in line.split()]
13     rooms.append(roomLine)
14
15 from collections import deque
16 def wallsAndGates(rooms):
17
18     if not rooms or not rooms[0]:
19         return
20
```

```

21     # rows = len(rooms)
22     # cols = len(rooms[0])
23     # queue = deque()
24
25     # # init queue with all gates pos
26     # for r in range(rows):
27     #     for c in range(cols):
28     #         if rooms[r][c] == 0:
29     #             queue.append((r, c))
30
31     # # directions for up, down, left, right
32     # directions = [(0,1), ]
33 wallsAndGates(rooms)
34
35 for i in range(len(rooms)):
36     printLine = ""
37     for j in range(len(rooms)):
38         printLine += str(rooms[i][j]) + " "
39     print(printLine)

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Hidden	Wrong Answer	0	0.0486 sec	10.4 KB
Testcase 1	Easy	Sample	Success	10	0.0332 sec	10.3 KB
Testcase 2	Easy	Sample	Success	10	0.0362 sec	10.3 KB
Testcase 3	Easy	Hidden	Success	10	0.0272 sec	10.5 KB
Testcase 4	Easy	Hidden	Wrong Answer	0	0.03 sec	10.5 KB

Testcase 5	Easy	Hidden	Success	10	0.029 sec	10.3 KB
Testcase 6	Easy	Hidden	Success	10	0.0395 sec	10.3 KB
Testcase 8	Easy	Hidden	Success	10	0.0337 sec	10.4 KB

! No comments.

8. Connected Components in Undirected Graph

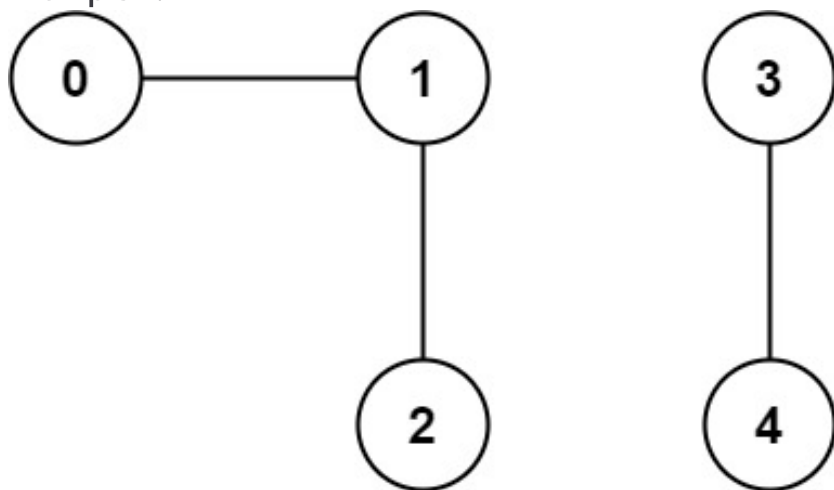
✖ Incorrect

Coding

Question description

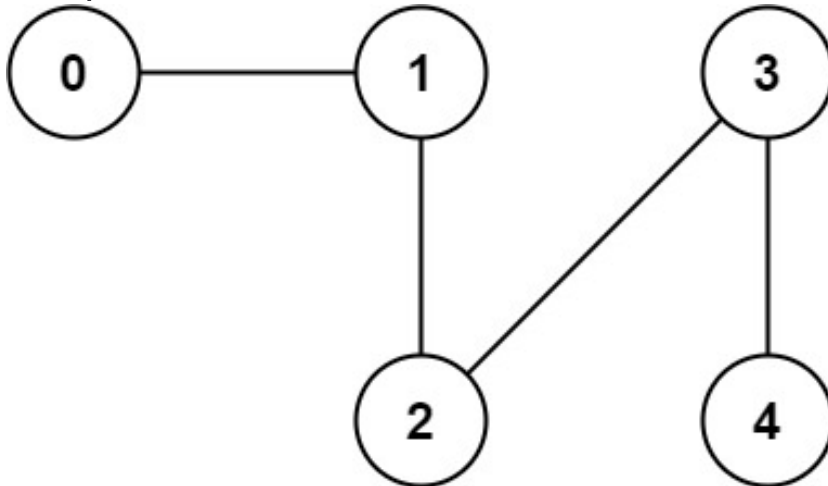
You have a graph of n nodes. You are given an integer n and an array `edges` where `edges[i] = [ai, bi]` indicates that there is an edge between a_i and b_i in the graph.
Return *the number of connected components in the graph*.

Example 1:



Input: $n = 5$, $\text{edges} = [[0,1],[1,2],[3,4]]$
Output: 2

Example 2:



Input: $n = 5$, $\text{edges} = [[0,1],[1,2],[2,3],[3,4]]$
Output: 1

Candidate's Solution

Language used: Python 3

```
1 import fileinput
2
3 args = []
4 for line in fileinput.input():
5     args.append(line)
6
7 n = int(args[0])
8 edges = []
9 for line in args[2:]:
10     preReqLine = [int(course) for course in line.split()]
11     edges.append(preReqLine)
12
13 def countComponents(n, edges):
14     """
15     :type n: int
16     :type edges: List[List[int]]
17     :rtype: int
18     """
19
20     print(countComponents(n, edges))
```


TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample	Wrong Answer	0	0.0301 sec	10.3 KB
Testcase 1	Easy	Hidden	Wrong Answer	0	0.0458 sec	10.4 KB
Testcase 2	Easy	Hidden	Wrong Answer	0	0.0307 sec	10.3 KB
Testcase 3	Easy	Hidden	Wrong Answer	0	0.0308 sec	10.2 KB
Testcase 4	Easy	Hidden	Wrong Answer	0	0.0299 sec	10.3 KB
Testcase 5	Easy	Hidden	Wrong Answer	0	0.0267 sec	10.4 KB
Testcase 6	Easy	Hidden	Wrong Answer	0	0.03 sec	10.3 KB

🚫 No comments.

9. Graph Valid Tree

⊖ Not attempted

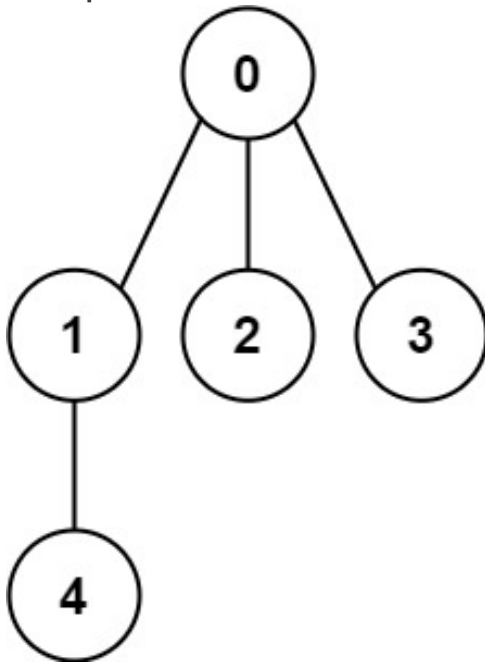
Coding

Question description

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 $\text{edges}[i] = [a_i, b_i]$ indicates that there is an undirected edge between nodes a_i and b_i in the graph.

Return `true` if the edges of the given graph make up a valid tree, and `false` otherwise.

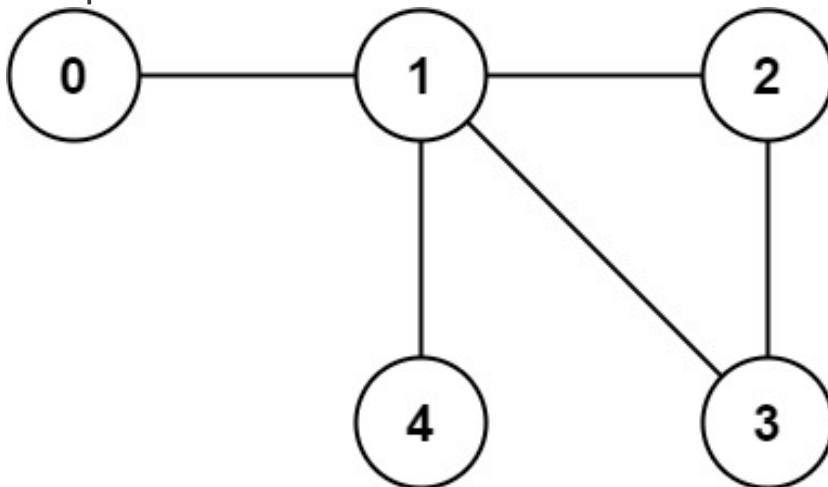
Example 1:



Input: $n = 5$, $\text{edges} = [[0,1],[0,2],[0,3],[1,4]]$

Output: `true`

Example 2:



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

Output: false

Candidate's Solution

Language used: **Python 3**

⚠ No answer was submitted for this question. Showing compiled/saved versions.

```
1
2 def validTree(n, edges):
3     """
4     :type n: int
5     :type edges: List[List[int]]
6     :rtype: int
7     """
8
```

⚠ No comments.