

## Path With Given Sequence (medium)

We'll cover the following ^

- Problem Statement
- Try it yourself
- Solution
- Code
  - Time complexity
  - Space complexity

### Problem Statement #

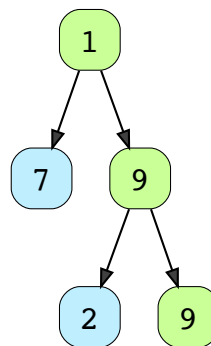
Given a binary tree and a number sequence, find if the sequence is present as a root-to-leaf path in the given tree.

#### Example 1:

Sequence: [1, 9, 9]

Output: true

Explanation: The tree has a path 1 -> 9 -> 9.



#### Example 2:

Sequence: [1, 0, 7]

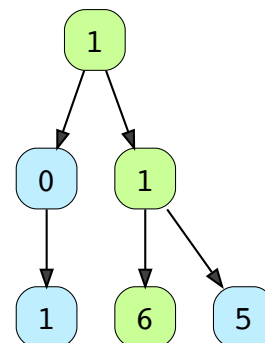
Output: false

Explanation: The tree does not have a path 1 -> 0 -> 7.

Sequence: [1, 1, 6]

Output: true

Explanation: The tree has a path 1 -> 1 -> 6.



### Try it yourself #

Try solving this question here:

educative

Python3

JS

C++

```
1 class TreeNode:
2     def __init__(self, val, left=None, right=None):
3         self.val = val
4         self.left = left
5         self.right = right
6
7
8 def find_path(root, sequence):
9     # TODO: Write your code here
10    return False
11
12
13 def main():
14
15     root = TreeNode(1)
16     root.left = TreeNode(0)
17     root.right = TreeNode(1)
18     root.left.left = TreeNode(1)
19     root.right.left = TreeNode(6)
20     root.right.right = TreeNode(5)
21
22     print("Tree has path sequence: " + str(find_path(root, [1, 0, 7])))
23     print("Tree has path sequence: " + str(find_path(root, [1, 1, 6])))
24
25
26 main()
27
```

## Solution #

This problem follows the Binary Tree Path Sum

(<https://www.educative.io/collection/page/5668639101419520/5671464854355968/5642684278505472/>) pattern. We can follow the same **DFS** approach and additionally, track the element of the given sequence that we should match with the current node. Also, we can return `false` as soon as we find a mismatch between the sequence and the node value.

## Code #

Here is what our algorithm will look like:

Java

Python3

C++

JS

```
1 class TreeNode:
2     def __init__(self, val, left=None, right=None):
3         self.val = val
4         self.left = left
5         self.right = right
6
7
8 def find_path(root, sequence):
9     if not root:
10        return len(sequence) == 0
11
12    return find_path_recursive(root, sequence, 0)
```

educative

```
12     return find_path_recursive(root, sequence, 0)
13
15 def find_path_recursive(currentNode, sequence, sequenceIndex):
16
17     if currentNode is None:
18         return False
19
20     seqLen = len(sequence)
21     if sequenceIndex >= seqLen or currentNode.val != sequence[sequenceIndex]:
22         return False
23
24     # if the current node is a leaf, add it is the end of the sequence, we have found a path
25     if currentNode.left is None and currentNode.right is None and sequenceIndex == seqLen - 1:
26         return True
27
28     # recursively call to traverse the left and right sub-tree
    currentNode.left = find_path_recursive(currentNode.left, sequence, sequenceIndex + 1)
    currentNode.right = find_path_recursive(currentNode.right, sequence, sequenceIndex + 1)
    return True
```

### Time complexity #

The time complexity of the above algorithm is  $O(N)$ , where 'N' is the total number of nodes in the tree. This is due to the fact that we traverse each node once.

### Space complexity #

The space complexity of the above algorithm will be  $O(N)$  in the worst case. This space will be used to store the recursion stack. The worst case will happen when the given tree is a linked list (i.e., every node has only one child).

← Back

Sum of Path Numbers (medium)

Next →

Count Paths for a Sum (medium)

☒ Mark as Completed

- Report an Issue

Ask a Question  
([https://discuss.educative.io/tag/path-with-given-sequence-medium\\_\\_pattern-tree-depth-first-search\\_\\_grokking-the-coding-interview-patterns-for-coding-questions](https://discuss.educative.io/tag/path-with-given-sequence-medium__pattern-tree-depth-first-search__grokking-the-coding-interview-patterns-for-coding-questions))