

[Previous \(/articles/lowest-common-ancestor-of-a-binary-tree/\)](#) [Next \(/articles/binary-tree-paths/\)](#)

112. Path Sum [↗](#) (/problems/path-sum/)

Nov. 17, 2018 | 18.4K views

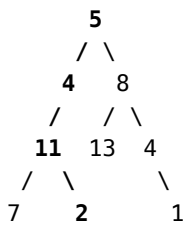
Average Rating: 4.69 (13 votes)

Given a binary tree and a sum, determine if the tree has a root-to-leaf path such that adding up all the values along the path equals the given sum.

Note: A leaf is a node with no children.

Example:

Given the below binary tree and `sum = 22`,



return true, as there exist a root-to-leaf path `5->4->11->2` which sum is 22.

Solution

Binary tree definition

First of all, here is the definition of the `TreeNode` which we would use in the following implementation.

Java

Python

Articles > 112. Path Sum ▼

 Copy

```
1 class TreeNode(object):
2     """ Definition of a binary tree node."""
3     def __init__(self, x):
4         self.val = x
5         self.left = None
6         self.right = None
```

Approach 1: Recursion

The most intuitive way is to use a recursion here. One is going through the tree by considering at each step the node itself and its children. If node *is not* a leaf, one calls recursively `hasPathSum` method for its children with a sum decreased by the current node value. If node *is* a leaf, one checks if the the current sum is zero, *i.e* if the initial sum was discovered.

Java

Python

 Copy

```
1 class Solution:
2     def hasPathSum(self, root, sum):
3         """
4         :type root: TreeNode
5         :type sum: int
6         :rtype: bool
7         """
8         if not root:
9             return False
10
11         sum -= root.val
12         if not root.left and not root.right: # if reach a leaf
13             return sum == 0
14         return self.hasPathSum(root.left, sum) or self.hasPathSum(root.right, sum)
```

Complexity Analysis

- Time complexity : we visit each node exactly once, thus the time complexity is $\mathcal{O}(N)$, where N is the number of nodes.
- Space complexity : in the worst case, the tree is completely unbalanced, *e.g.* each node has only one child node, the recursion call would occur N times (the height of the tree), therefore the storage to keep the call stack would be $\mathcal{O}(N)$. But in the best case (the tree is completely balanced), the height of the tree would be $\log(N)$. Therefore, the space complexity in this case would be $\mathcal{O}(\log(N))$.

Approach 2: Iterations

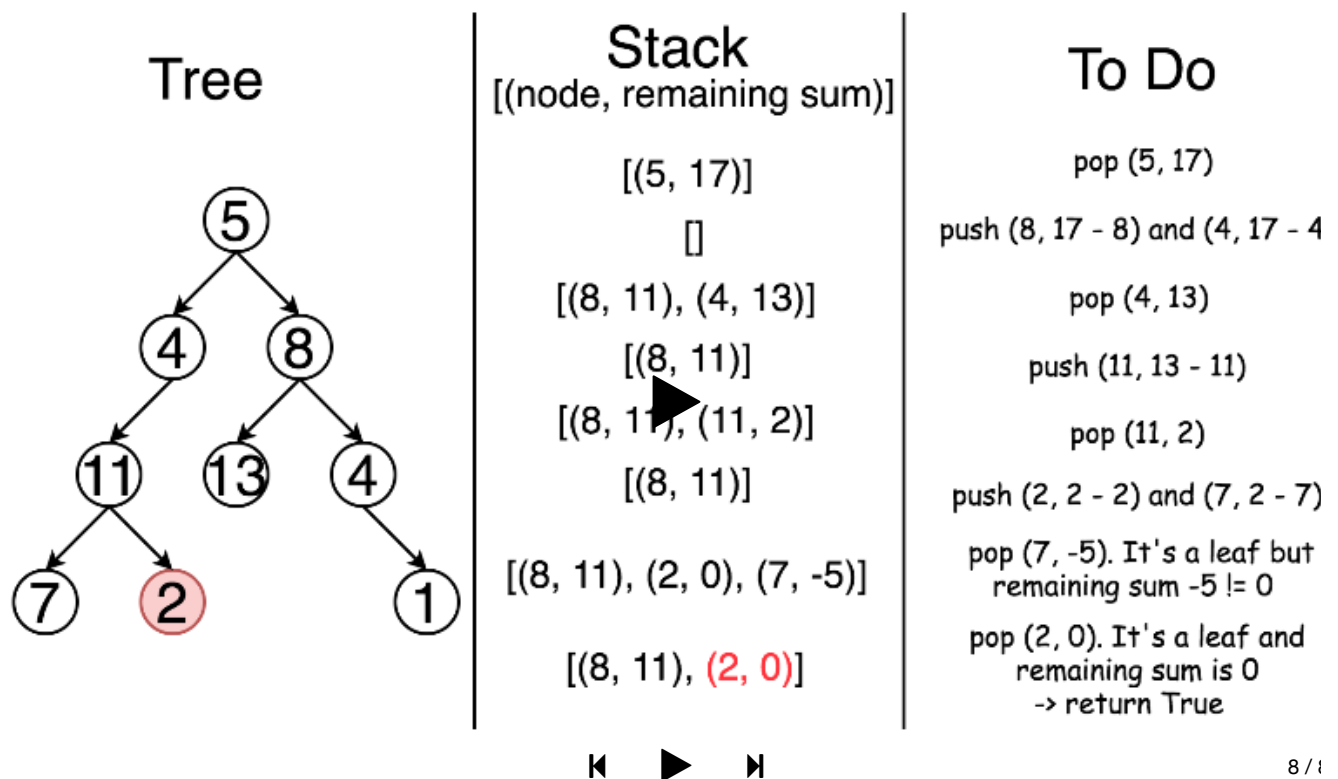
Articles > 112. Path Sum ▼

Algorithm

We could also convert the above recursion into iteration, with the help of stack. DFS would be better than BFS here since it works faster except the worst case. In the worst case the path $\text{root} \rightarrow \text{leaf}$ with the given sum is the last considered one and in this case DFS results in the same productivity as BFS.

The idea is to visit each node with the DFS strategy, while updating the remaining sum to cumulate at each visit.

So we start from a stack which contains the root node and the corresponding remaining sum which is $\text{sum} - \text{root.val}$. Then we proceed to the iterations: pop the current node out of the stack and return `True` if the remaining sum is `0` and we're on the leaf node. If the remaining sum is not zero or we're not on the leaf yet then we push the child nodes and corresponding remaining sums into stack.



8 / 1

Java

Python

Articles > 112. Path Sum ▼

Copy

```

1 class Solution:
2     def hasPathSum(self, root, sum):
3         """
4         :type root: TreeNode
5         :type sum: int
6         :rtype: bool
7         """
8         if not root:
9             return False
10
11         de = [(root, sum - root.val), ]
12         while de:
13             node, curr_sum = de.pop()
14             if not node.left and not node.right and curr_sum == 0:
15                 return True
16             if node.right:
17                 de.append((node.right, curr_sum - node.right.val))
18             if node.left:
19                 de.append((node.left, curr_sum - node.left.val))
20         return False

```

Complexity Analysis

- Time complexity : the same as the recursion approach $\mathcal{O}(N)$.
- Space complexity : $\mathcal{O}(N)$ since in the worst case, when the tree is completely unbalanced, e.g. each node has only one child node, we would keep all N nodes in the stack. But in the best case (the tree is balanced), the height of the tree would be $\log(N)$. Therefore, the space complexity in this case would be $\mathcal{O}(\log(N))$.

Analysis written by @liaison (<https://leetcode.com/liaison/>) and @andvary (<https://leetcode.com/andvary/>)

Rate this article:

Previous (</articles/lowest-common-ancestor-of-a-binary-tree/>)

Next (</articles/binary-tree-paths/>)

Comments: 7

Sort By ▼



Type comment here... (Markdown is supported)

Preview

Post



hongyu9310 (hongyu9310) ★ 19 🕒 December 27, 2018 11:28 PM
Articles > 112. Path Sum ▼

I'm pretty sure this is DFS

(/hongyu9310)

```
node, curr_sum = de.pop()
if node.right: de.append(node.right, curr_sum - node.right.val)
if node.left: de.append(node.left, curr_sum - node.left.val)
```

Read More

2 ^ v | Share | Reply



(/nideesht)

NideeshT (nideesht) ★ 172 🕒 June 23, 2019 10:11 PM

Java Code + Whiteboard Youtube Video Explanation accepted - <https://www.youtube.com/watch?v=zfsRQGzM7iE> (<https://www.youtube.com/watch?v=zfsRQGzM7iE>) (clickable link)

Leetcode 112 - Binary Tre...

Read More

0 ^ v | Share | Reply



(/s961206)

s961206 (s961206) ★ 316 🕒 July 5, 2019 5:56 PM

For iteration version, it can also use pair:

```
public boolean hasPathSum(TreeNode root, int sum) {
    LinkedList<Pair<TreeNode, Integer>> stack = new LinkedList<>();
    stack.add(new Pair(root, sum));
```

Read More

0 ^ v | Share | Reply



(/hanw3)

hanw3 (hanw3) ★ 3 🕒 January 11, 2019 12:24 AM

I think the second iterative solution has a space complexity of $O(1)$ since everytime you push at most two node into stack and pop out one so the element in stack would be always under 2.

0 ^ v | Share | Reply

SHOW 1 REPLY



(/pidouki)

pidouki (pidouki) ★ 6 🕒 January 5, 2019 1:23 PM

If a stack there then it is dfs
Bfs should use a queue

0 ^ v | Share | Reply



(/yyc0340)

yyc0340 (yyc0340) ★ 0 🕒 November 22, 2018 9:21 PM

Correct me from wrong. The iterative solution seems to be BFS rather than DFS.

0 ^ v | Share | Reply

SHOW 2 REPLIES



(/arihant95)

arihant95 (arihant95) ★ 5 🕒 November 23, 2018 1:12 PM
Articles > 112. Path Sum ▼

Even I agree with @yyc0340 (<https://leetcode.com/yyc0340>) the code is BFS and animation os for dfs. We have to convert if condition to elif condition to make it dfs. Correct me if I am wrong?

-1 ▲ ▼ ⏏ Share ↩ Reply

SHOW 1 REPLY