Leetcode 113.
Path Sum II

#Depth-First Search #Binary Tree #Backtracking #Tree

Problem Description - Word Search

Given the root of a binary tree and an integer **targetSum**, return all **root-to-leaf** paths where the sum of the node values in the path equals targetSum. Each path should be returned as a list of the node **values**, not node references.

[Def] A **root-to-leaf path** is a path starting from the root and ending at any leaf node.

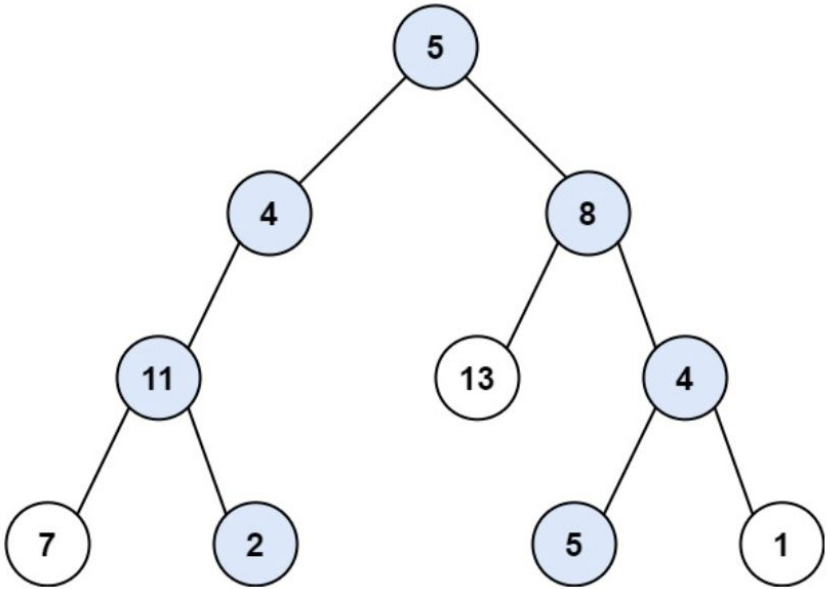
[Def] A **leaf** is a node with no children.

Example 1:

Input:

Input: root = [5,4,8,11,null,13,4,7,2,null,null,5,1],
targetSum = 22

Output: [[5,4,11,2],[5,8,4,5]]

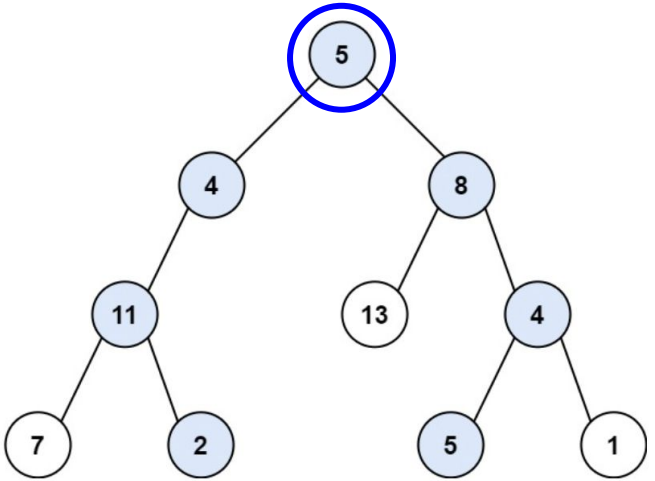


Backtracking - Depth-First Search & Recursion

Backtracking:

- The goal of Backtracking is to use brute force to find all solutions to a problem.
- It needs to decide between multiple alternatives to the next component of the solution, it recursively evaluates every alternative and then chooses the best one

Start from the **root**!



targetSum = 22

Backtracking - Depth-First Search & Recursion

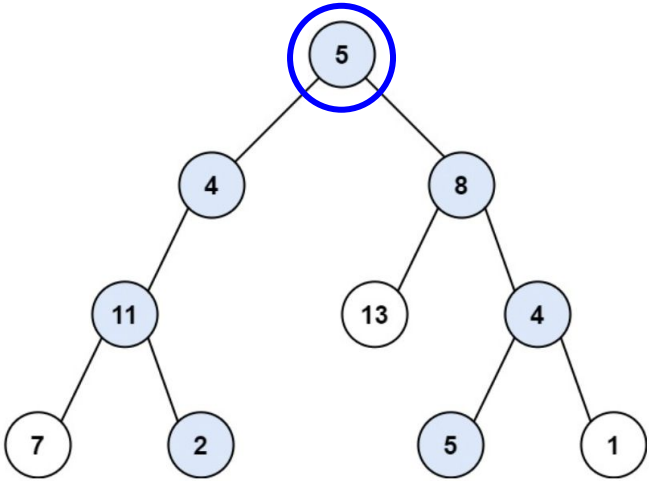
Base case:

- Current is a leaf node (cur.right, cur.left is none)
- The remaining sum should be 0 (targetSum should be achieved)

What we have to keep track:

[1] Current node we are visiting
[2] path so far (containing each node values)
[3] remaining sum

Start from the **root**!

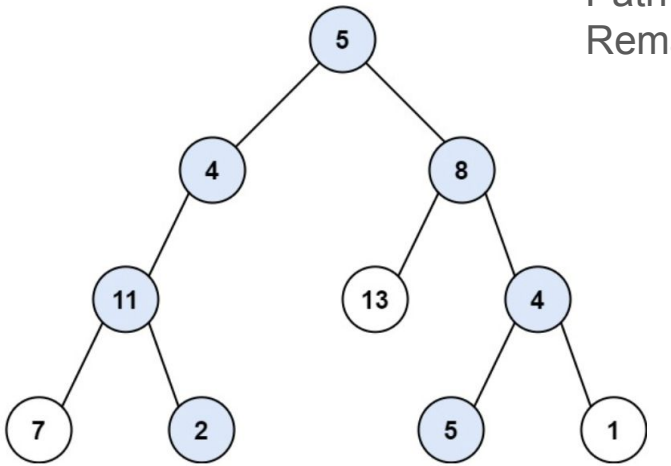


targetSum = 22

Backtracking - Depth-First Search & Recursion

Start from the **root**!

cur = 5
Path = [5]
Remaining_sum: 22-5=17



[1] Current node we are visiting
[2] path so far (containing each node values)
[3] remaining sum

targetSum = 22

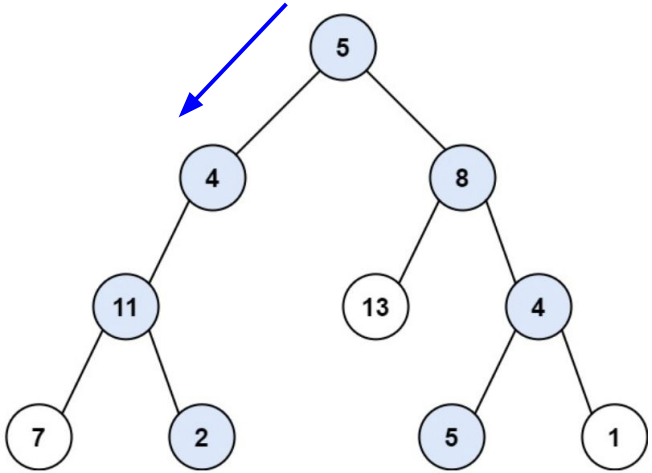
Backtracking - Depth-First Search & Recursion

cur.left:

cur: 4

path so far: [5, 4]

remaining_sum: 17-4=13



[1] Current node we are visiting
[2] path so far (containing each node values)
[3] remaining sum

targetSum = 22

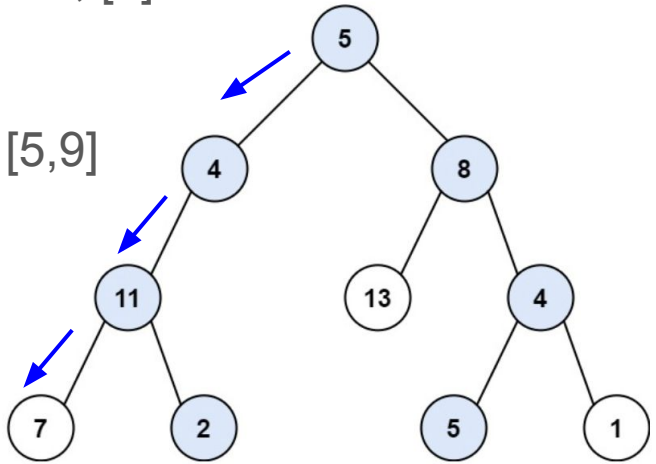
Backtracking - Depth-First Search & Recursion

17, [5]

13, [5,9]

2, [5,9,11]

-5, [5,9,11, 7]



base case condition satisfied:

Is the node a leaf node? ✅
Is the remaining sum = 0? ❌

→ pop cur node from the path

[1] Current node we are visiting
[2] path so far (containing each node values)
[3] remaining sum

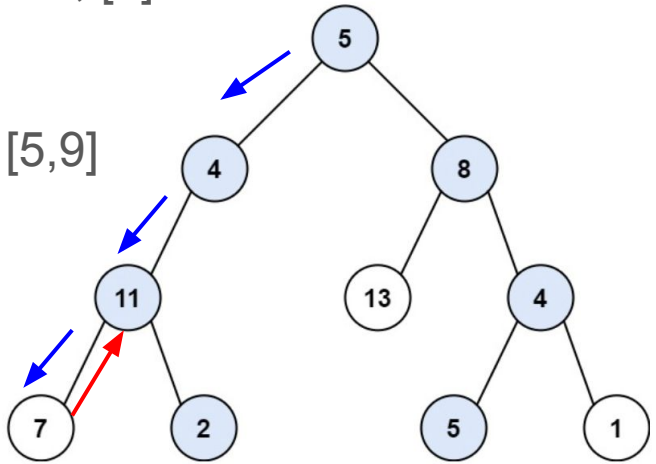
targetSum = 22

Backtracking - Depth-First Search & Recursion

17, [5]

13, [5,9]

2, [5,9,11]



base case condition satisfied:

Is the node a leaf node? ✅
Is the remaining sum = 0? ❌

→ pop cur node from the path

[1] Current node we are visiting
[2] path so far (containing each node values)
[3] remaining sum

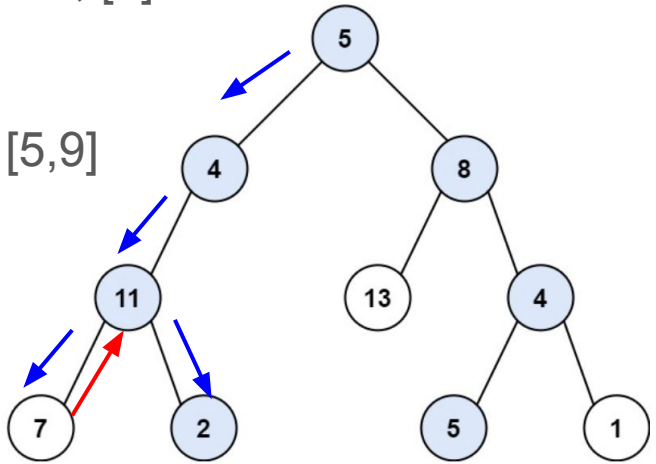
targetSum = 22

Backtracking - Depth-First Search & Recursion

17, [5]

13, [5,9]

2, [5,9,11]



base case condition satisfied:

- Is the node a leaf node? ✅
  - Is the remaining sum = 0? ✅ → Add to paths

0, [5,9,11, 2]

[1] Current node we are visiting
[2] path so far (containing each node values)
[3] remaining sum

targetSum = 22

Code Solution

```python
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
```

Time Complexity: O(N^2),
where N is number of tree nodes

Memory Complexity: O(N)

```python
class Solution:
    def pathSum(self, root, targetSum):

        if not root:
            return []

        paths = []

        def backtrak(cur, path, remaining_sum):
            if cur is None:
                return

            # append current node's value
            path.append(cur.val)

            # Base case
            if not cur.left and not cur.right and remaining_sum == cur.val:
                paths.append(path.copy()) #  create a shallow copy of the current path list
                return

            backtrak(cur.left, path, remaining_sum - cur.val)
            backtrak(cur.right, path, remaining_sum - cur.val)

            path.pop()


        backtrak(root, [], targetSum)
        return paths
```

감사합니다!

THANK YOU