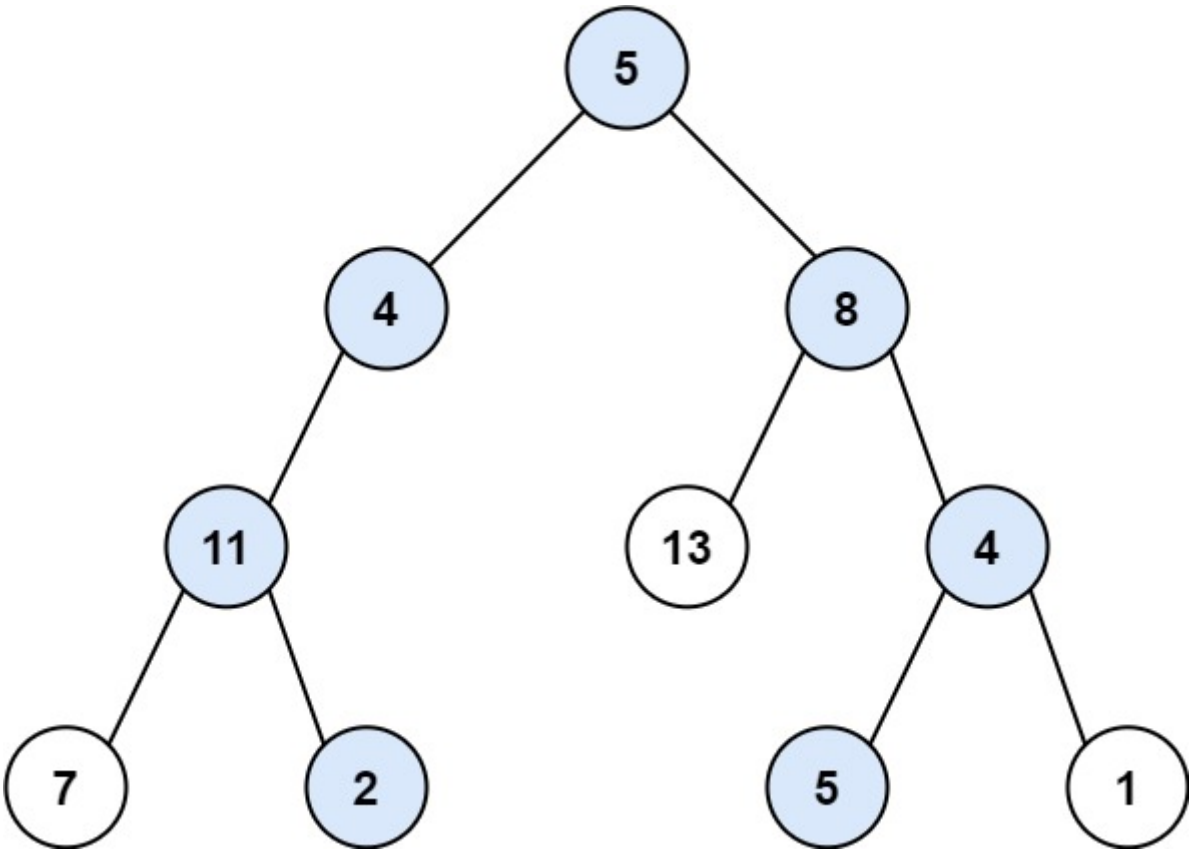
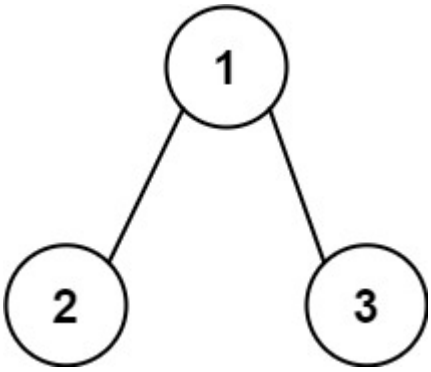
113. Path Sum II

Given the `root` of a binary tree and an integer `targetSum`, return all **root-to-leaf** paths where each path's sum equals `targetSum`.

A **leaf** is a node with no children.



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]]



Input: root = [1,2,3], targetSum = 5
Output: []

```
def pathSum(self, root: TreeNode, targetSum: int) -> List[List[int]]:
        ans = []
```

```
        temp = []
        self.helper(root,ans,temp,targetSum)
        return ans


    def helper(self,root,ans,temp,targetSum):
        if root is None:
            return
        temp.append(root.val)
        if targetSum==root.val and root.left is root.right:
            # temp.append(root.val)
            ans.append(temp[:])
            # return
        self.helper(root.left,ans,temp,targetSum-root.val)
        self.helper(root.right,ans,temp,targetSum-root.val)
        temp.pop()
```

```
def pathSum(self, root: Optional[TreeNode], targetSum: int) ->
List[List[int]]:
        if root is None:
            return []

        ans = []
        self.helper(root,ans,targetSum,[])
        return ans



    def helper(self,root,ans,target,ssf):
        if root is None:
            return
        if root.left is root.right:
            if target == root.val:
                ssf.append(root.val)
                ans.append(ssf)
            return
        self.helper(root.left,ans,target-root.val,ssf+[root.val])
        self.helper(root.right,ans,target-root.val,ssf+[root.val])
```