# Max Path Sum

## 124. Binary Tree Maximum Path Sum

Solved

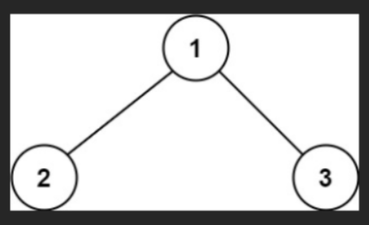
Hard | Topics | Companies

A **path** in a binary tree is a sequence of nodes where each pair of adjacent nodes in the sequence has an edge connecting them. A node can only appear in the sequence **at most once**. Note that the path does not need to pass through the root.

The **path sum** of a path is the sum of the node's values in the path.

Given the `root` of a binary tree, return *the maximum **path sum** of any **non-empty** path*.

**Example 1:**




```
Input: root = [1,2,3]
Output: 6
Explanation: The optimal path is 2 -> 1 -> 3 with a path sum of 2 + 1 + 3 = 6.
```


→ Thus is the path, where we can get maximum Sum = 6

* key points to understand :


if you consider this path, →

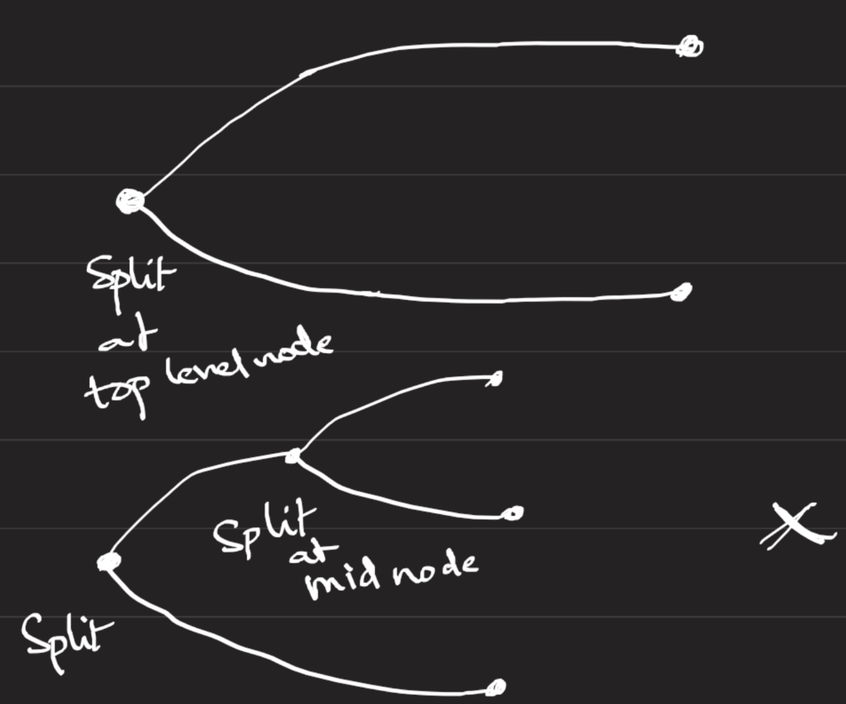
You are actually splitting the path Once... at 3 to {4, 5} ↳ into 2 directions

let us extend above example :


Can i do this ?

→ The answer is No

Because, once split the path at 3, you can split the path again




√

X

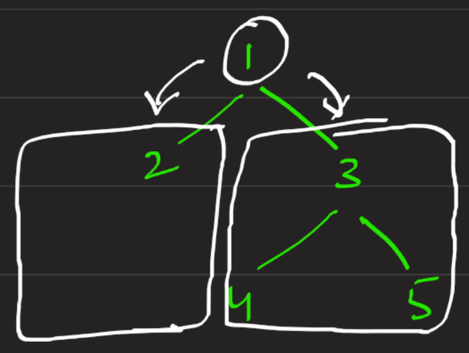
You can't go in a single path, if you split in between

↓

So if you are splitting at a X node

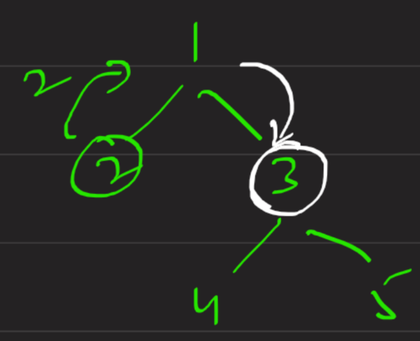
that X need to the top level node

we do simple DFS




In left recursive call, what is the maximum path sum we can get without splitting

Similarly in right recursive call



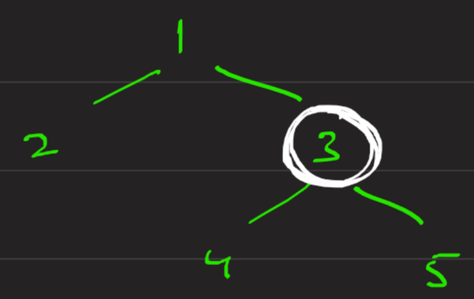

res / 2

On every node, calculate 2 values

(1) what is the max path sum if you are allowed to split ?

(2) what is the max path ↘ left (or) right ↙ ?

I know, this makes you confuse → don't loose pace




Assume :)

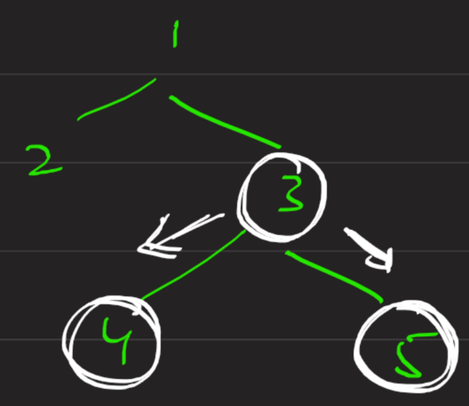
* if somehow, we came to node (3)

→ Think like you have never split your path before coming to node (3)

So, you have choice to split your path from (3)

And you have your decision whether to consider split or take left path or right path

depend upon on sum you got



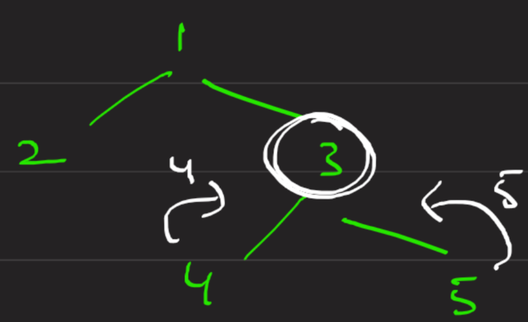

I've decided to split now from 3

Again Same old story, like we have to assume we never split before reaching (4) or (5)

Hold a moment, you'll understand

* Because, 4 is a child node → it returns 4
  5 is a childnode → it return 5

Now game begins




So, previously we assumed from 5, we split or not

if split → the sum will be leftSum + root.val + rightSum

= 4 + 3 + 5 = (12)

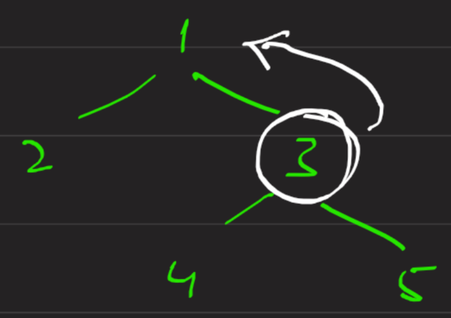
* Create a global variable ans → and store Max of (ans, split val)

ans
0
12

* why particularly store split val in ans?

→ you will clarity, once we go ahead

ans
12



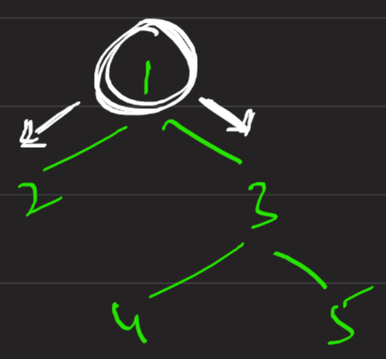

So, we have a path of sum = 12 which came by splitting at (3)

Now what do you return to node (1)

* So, we are going back to node (1)

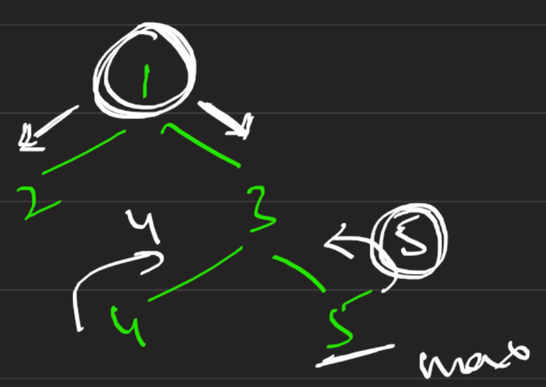
∵ As we thought to split n non-split at every node

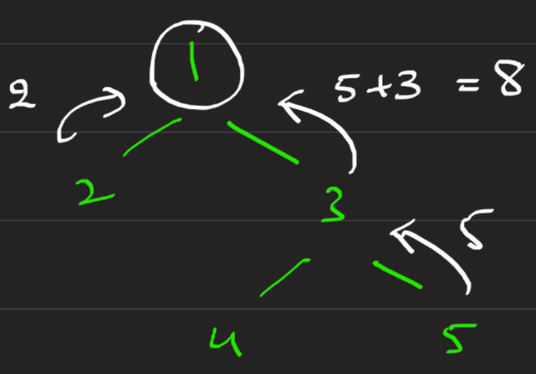
→ we somehow went to 3, as we assumed same split scenario on node (1) aswell




So, if we decide to split from (1) → we are not suppose to split at (3)

So, we pick the path which gives max from (3) (Either left or right)




, Now if you check a path that passes through (1) when split happens

→ leftSum + root.val + rightSum

= 2 + 1 + 8 = (11)

But our ans is already having a max sum of (12)

So, we wont update it

∴ ans = 12


→ this path is not possible

as there are 2 splits at Node ① & ②

that's why we use a global variable to update inner splits & just return max path on Either left Subtree (or) rightSubtree

Assuming split is already happened.

```java
class Solution {
    int ans;
    public int dfs(TreeNode root){
        if(root == null)return 0;
        int x = dfs(root.left);
        int y = dfs(root.right);
        y = Math.max(0,y);
        x = Math.max(0,x);

        //with split
        ans = Math.max(ans,root.val+x+y);

        //without split
        return root.val+Math.max(x,y);
    }
}
```

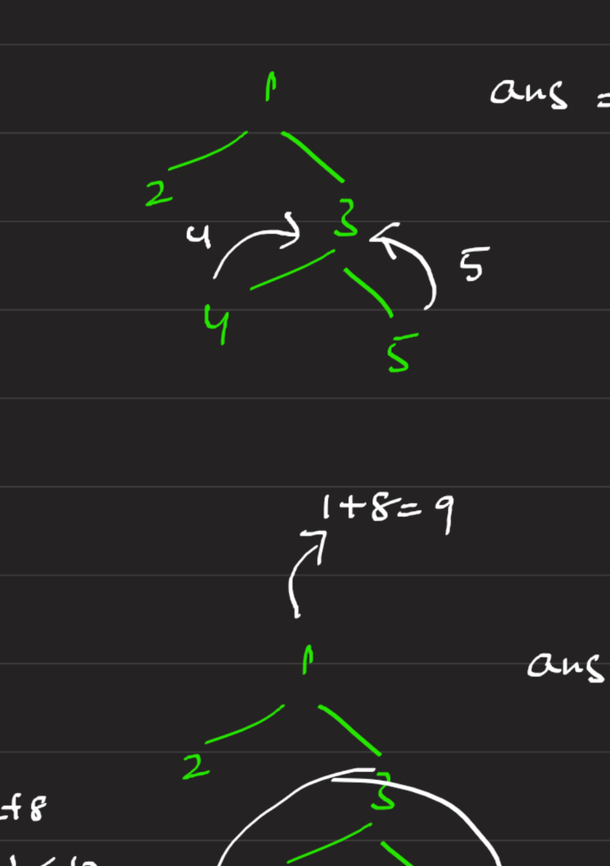
→ global, to update inner split maxSum

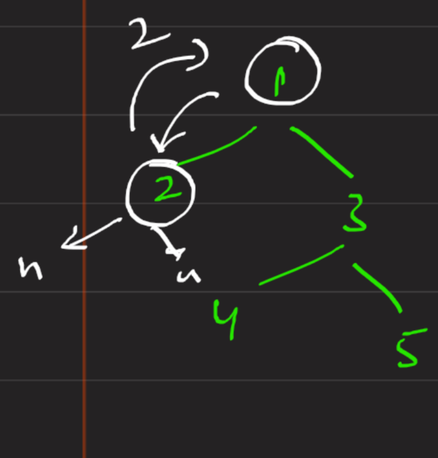
Make sure, you never carry negative Sum

leftSum

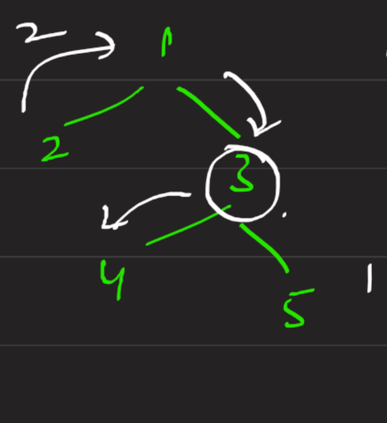
rightSum

without split




ans =



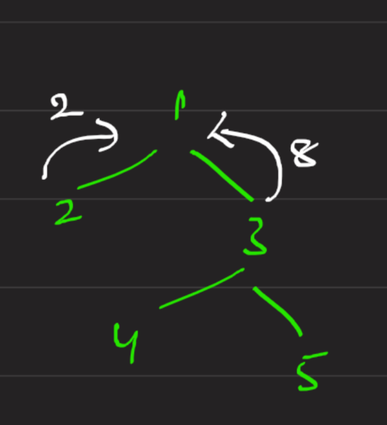

ans =




ans = ∅

3+4+5 = 12


ans =




ans = 12

SplitSum = 1+2+8

= 11 < 12

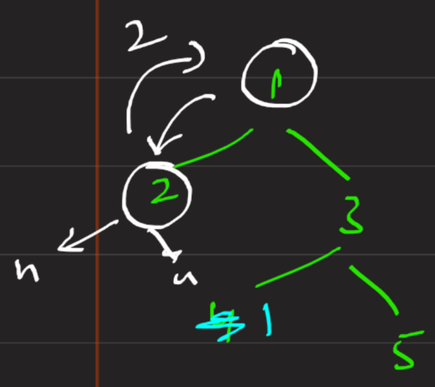
No update



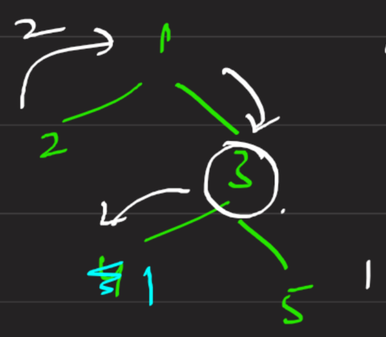

ans = 12

this is max

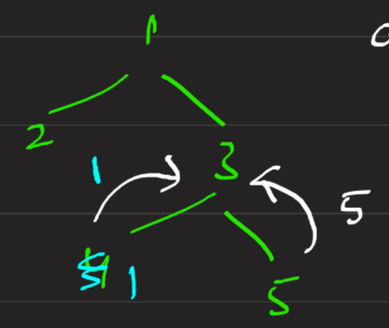
If suppose → node (4) is replaced by some value (1)



ans =

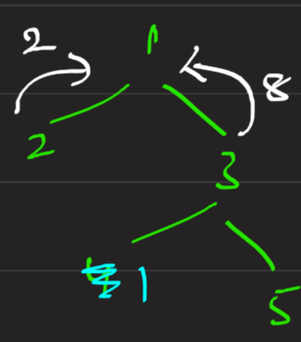


ans =



ans = ~~9~~ 3+1+5 = 9

ans = 9



ans = ~~9~~

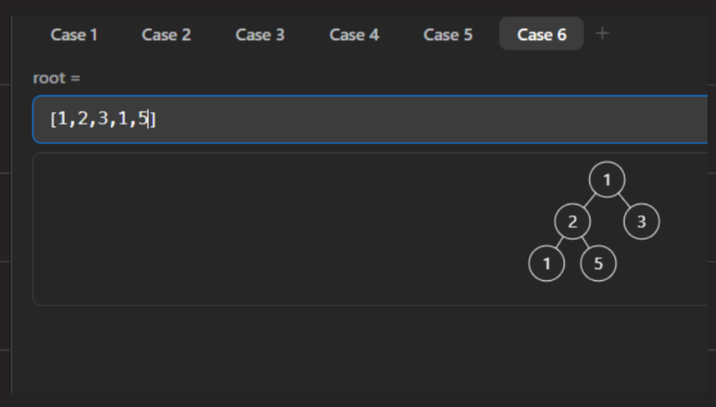
SplitSum = 1+2+8
= 11 < 29

No update

ans = 11

this is max

new ans, which is greater
So, update ans




→ I have entered a new testcase same as above example & check our ans = 11 is true or not