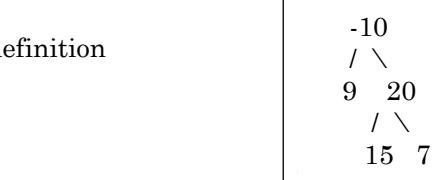
Max path sum in C++

```
#include <iostream>
#include <climits>   // For INT_MIN
#include <algorithm> // For std::max
using namespace std;
// TreeNode structure definition
struct TreeNode {
    int key;
    TreeNode* left;
    TreeNode* right;
    TreeNode(int x) {
        key = x;
        left = nullptr;
        right = nullptr;
    }
};

// Helper function to calculate the maximum path sum going down from a node
int maxPathDown(TreeNode* node, int& maxValue) {
    if (node == nullptr) return 0;

    // Calculate maximum path sums from left and right subtrees
    int left = std::max(0, maxPathDown(node->left, maxValue)); // Ignore negative sums
    int right = std::max(0, maxPathDown(node->right, maxValue)); // Ignore negative sums

    // Update maxValue with the maximum path sum found so far
    maxValue = std::max(maxValue, left + right + node->key);

    // Return the maximum path sum going down from the current node
    return std::max(left, right) + node->key;
}

// Function to find the maximum path sum in a binary tree
int maxPathSum(TreeNode* root) {
    int maxValue = INT_MIN; // Initialize with minimum possible integer value
    maxPathDown(root, maxValue);
    return maxValue;
}

int main() {
    // Constructing the binary tree
    TreeNode* root = new TreeNode(-10);
    root->left = new TreeNode(9);
    root->right = new TreeNode(20);
    root->right->left = new TreeNode(15);
    root->right->right = new TreeNode(7);

    // Finding the maximum path sum in the binary tree
    int answer = maxPathSum(root);
```

Tree Structure

You built this binary tree:



🧠 **Core Logic (Recap)**

1. maxPathDown(node):
    - Gets **max sum** for any path **starting** from the current node and going **downward**.
    - Ignores negative subtrees (max(0, left/right)).
    - Updates the global maxValue if a new candidate sum left + right + node->key is higher.

📋 **Dry Run Table**

Node	Left Subtree	Right Subtree	Local Max (left + right + node)	Return Upward	maxValue Updated
15	0	0	15	15	✅ 15
7	0	0	7	7	❌
20	15	7	42 (=15+7+20)	35	✅ 42
9	0	0	9	9	❌
-10	9	35	34 (=9+35-10)	25	❌

🧠 So the final max path **goes through 15 → 20 → 7 = 42**

✅ **Output:**

The Max Path Sum for this tree is 42

```
    std::cout << "The Max Path Sum for this
tree is " << answer << std::endl;

    // Deallocating memory
    delete root->right->right;
    delete root->right->left;
    delete root->right;
    delete root->left;
    delete root;

    return 0;
}
```

```
The Max Path Sum for this tree is 42
```