Binary Tree Maximum Path Sum

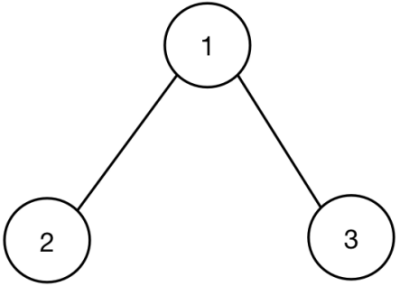
*Description:*

Given a non-empty Binary Tree, try to find the maximum path sum. *A path is defined as any sequence of nodes from any node in the tree along the parent - child connections.*

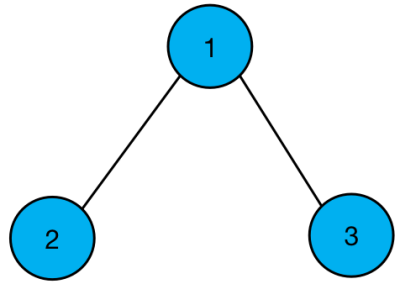
The path must contain at least one node and it’s not necessary to go through the root node.

*Example 1:*

Given the Binary Tree with all tree nodes [1, 2, 3] as below:

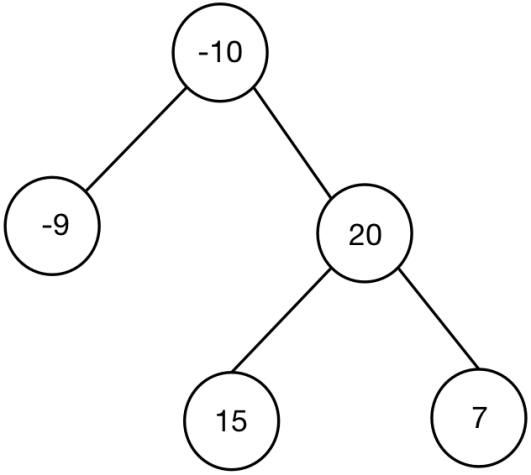


Output is 6, and the final path route with maximum path sum is [2, 1, 3].

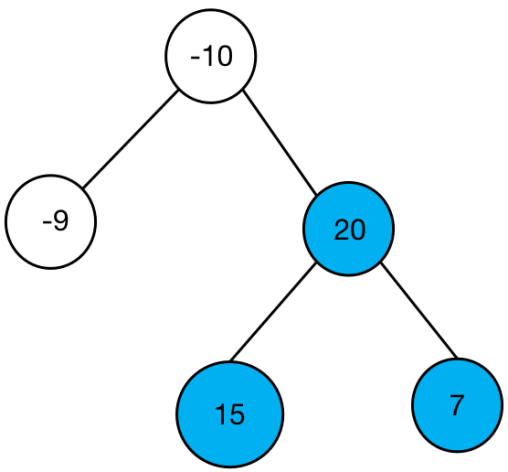


*Example 2:*

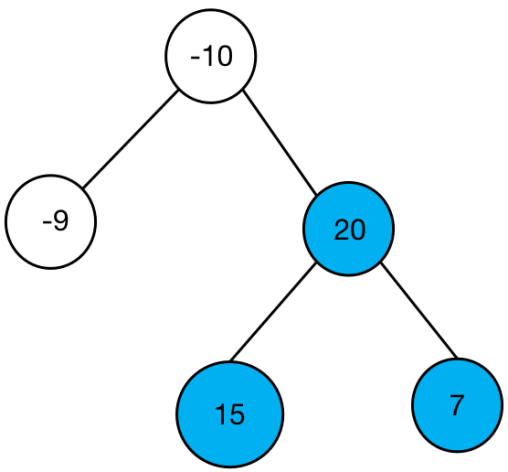
Given the Binary Tree with all tree nodes [-10, 9, 20, null, null, 15, 7] as below:



Output is 42, and the final path route with maximum path sum is [15, 20, 7].



*Analysis:*



In this example, choose LRD algorithm to visit all nodes in the Binary Tree.

*Algorithm Outline:*

After visiting the left node in Binary Tree, then visit the right node in the Binary Tree. Finishing visiting all nodes in the left sub - tree and the right sub – tree, calculating the maximum sum from the root node.

* left\_node->value > 0, then add it to node->value.
* right\_node->value > 0, then add it to node->value.
* save the sum value of current sub – tree and compare it with the global variable MAX\_VALUE.
* return the bigger left sub – tree sum value or the bigger right sub – tree sum value in each round.

*Code:*

struct TreeNode {

int value;

TreeNode \* left;

TreeNode \* right;

TreeNode (int x) : value(x), left(NULL), right(NULL) {}

};

class Solution {

public:

int maxPathSum (TreeNode \* root) {

MAX\_VALUE = INT\_MIN; // (INT\_MIN is the minimum integer value.)

DFS (root);

return MAX\_VALUE;

}

int DFS (TreeNode \* root) {

if (root == NULL) return 0;

int left = DFS(root->left);

int right = DFS(root->right);

int sum = root->value;

if (left > 0)

sum = sum + left;

if (right > 0)

sum = sum + right;

MAX\_VALUE = max(MAX\_VALUE, sum);

return max(left, right) > 0? max(left, right) + root->value : root->value;

}

private:

int MAX\_VALUE;

};