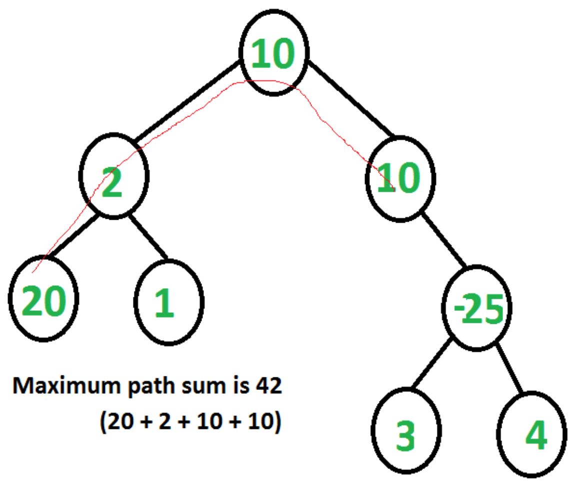
Maximum Path Sum in a Binary Tree

Given a binary tree, find the maximum path sum. The path may start and end at any node in the tree.

Example:

```
Input: Root of below tree
       1
      / \
     2   3
Output: 6

See below diagram for another example.
1+2+3
```



We strongly recommend you to minimize your browser and try this yourself first.

For each node there can be four ways that the max path goes through the node:
1. Node only
2. Max path through Left Child + Node
3. Max path through Right Child + Node
4. Max path through Left Child + Node + Max path through Right Child

The idea is to keep trace of four paths and pick up the max one in the end. An important thing to note is, root of every subtree need to return maximum path sum such that at most one child of root is involved. This is needed for parent function call. In below code, this sum is stored in 'max_single' and returned by the recursive function.

C++

```
// C/C++ program to find maximum path sum in Binary Tree
#include<bits/stdc++.h>
using namespace std;

// A binary tree node
struct Node
{
    int data;
    struct Node* left, *right;
};

// A utility function to allocate a new node
struct Node* newNode(int data)
{
    struct Node* newNode = new Node;
    newNode->data = data;
    newNode->left = newNode->right = NULL;
    return (newNode);
}

// This function returns overall maximum path sum in 'res'
// And returns max path sum going through root.
int findMaxUtil(Node* root, int &res)
{
    //Base Case
    if (root == NULL)
        return 0;

    // l and r store maximum path sum going through left and
    // right child of root respectively
    int l = findMaxUtil(root->left,res);
    int r = findMaxUtil(root->right,res);

    // Max path for parent call of root. This path must
    // include at-most one child of root
    int max_single = max(max(l, r) + root->data, root->data);

    // Max Top represents the sum when the Node under
    // consideration is the root of the maxsum path and no
    // ancestors of root are there in max sum path
    int max_top = max(max_single, l + r + root->data);

    res = max(res, max_top); // Store the Maximum Result.

    return max_single;
}

// Returns maximum path sum in tree with given root
int findMaxSum(Node *root)
{
    // Initialize result
    int res = INT_MIN;

    // Compute and return result
    findMaxUtil(root, res);
    return res;
}

// Driver program
int main(void)
{
    struct Node *root = newNode(10);
    root->left        = newNode(2);
    root->right       = newNode(10);
    root->left->left  = newNode(20);
    root->left->right = newNode(1);
    root->right->right = newNode(-25);
    root->right->right->left   = newNode(3);
    root->right->right->right  = newNode(4);
    cout << "Max path sum is " << findMaxSum(root);
    return 0;
}
```

Java

```
// Java program to find maximum path sum in Binary Tree

/* Class containing left and right child of current
 node and key value*/
class Node {

    int data;
    Node left, right;

    public Node(int item) {
        data = item;
        left = right = null;
    }
}

// An object of Res is passed around so that the
// same value can be used by multiple recursive calls.
class Res {
    public int val;
}

class BinaryTree {

    // Root of the Binary Tree
    Node root;

    // This function returns overall maximum path sum in 'res'
    // And returns max path sum going through root.
    int findMaxUtil(Node node, Res res)
    {

        // Base Case
        if (node == null)
            return 0;

        // l and r store maximum path sum going through left and
        // right child of root respectively
        int l = findMaxUtil(node.left, res);
        int r = findMaxUtil(node.right, res);

        // Max path for parent call of root. This path must
        // include at-most one child of root
        int max_single = Math.max(Math.max(l, r) + node.data,
                                  node.data);


        // Max Top represents the sum when the Node under
        // consideration is the root of the maxsum path and no
        // ancestors of root are there in max sum path
        int max_top = Math.max(max_single, l + r + node.data);

        // Store the Maximum Result.
        res.val = Math.max(res.val, max_top);

        return max_single;
    }

    int findMaxSum() {
        return findMaxSum(root);
    }

    // Returns maximum path sum in tree with given root
    int findMaxSum(Node node) {

        // Initialize result
        // int res2 = Integer.MIN_VALUE;
        Res res = new Res();
        res.val = Integer.MIN_VALUE;

        // Compute and return result
        findMaxUtil(node, res);
        return res.val;
    }

    /* Driver program to test above functions */
    public static void main(String args[]) {
        BinaryTree tree = new BinaryTree();
        tree.root = new Node(10);
        tree.root.left = new Node(2);
        tree.root.right = new Node(10);
        tree.root.left.left = new Node(20);
        tree.root.left.right = new Node(1);
        tree.root.right.right = new Node(-25);
        tree.root.right.right.left = new Node(3);
        tree.root.right.right.right = new Node(4);
        System.out.println("maximum path sum is : " +
                            tree.findMaxSum());
    }
}
```

Output:

```
Max path sum is 42
```

Time Complexity: O(n) where n is number of nodes in Binary Tree.