Finding Maximum Subtree Sum in a Binary Tree

```javascript
class Node {
    constructor(data) {
        this.data = data;
        this.left = null;
        this.right = null;
    }
}

class Solution {
    constructor() {
        this.maxSum = Number.MIN_SAFE_INTEGER;
    }

    maxSubtreeSum(root) {
        if (root === null) return 0;
        const leftSum = this.maxSubtreeSum(root.left);
        const rightSum = this.maxSubtreeSum(root.right);
         const subtreeSum = root.data + leftSum + rightSum;


        this.maxSum = Math.max(this.maxSum, subtreeSum);


        return subtreeSum;
    }


    findMaxSubtreeSum(root) {
        this.maxSubtreeSum(root);
        return this.maxSum;
    }
}

const solution = new Solution();
// const root1 = new Node(1);
// root1.left = new Node(2);
// root1.right = new Node(3);
// root1.left.left = new Node(4);
// root1.left.right = new Node(5);
// root1.right.left = new Node(6);
// root1.right.right = new Node(7);
// console.log(solution.findMaxSubtreeSum(root1)); // Output: 28

const root2 = new Node(1);
root2.left = new Node(-2);
root2.right = new Node(3);
root2.left.left = new Node(4);
root2.left.right = new Node(5);
root2.right.left = new Node(-6);
root2.right.right = new Node(2);
console.log(solution.findMaxSubtreeSum(root2));
```