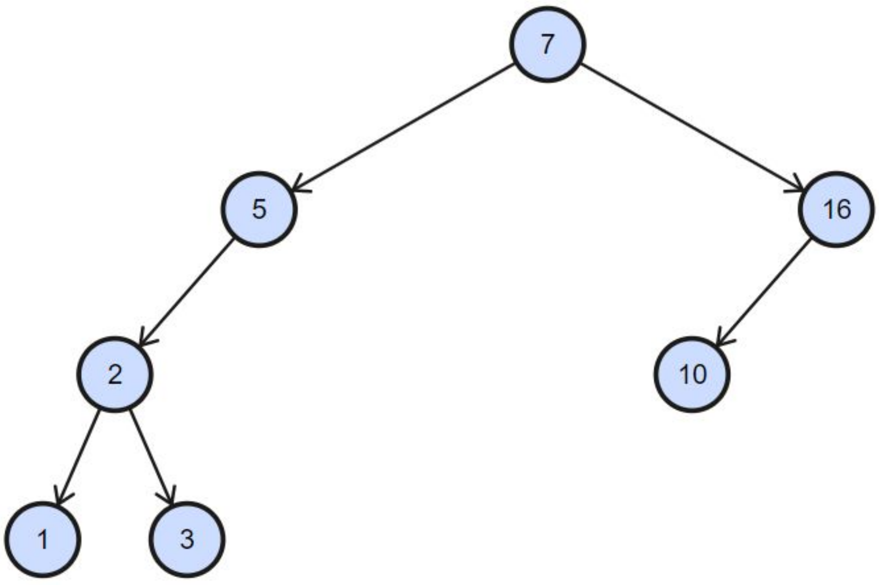
CMS 460
Algorithms
Sprint 5
Dynamic Programming
Kenneth Shortrede, Alex Wehr, Olliver Luo, Sanghun Park

Trees - Sum of the children of even nodes

Introduction

In the following problem, we are given a tree with different integer values that (of course) are either even or odd. Then we need to find those nodes that are even, and sum up the values of their immediate children. To give a simple example, let's look at the following graph.

Example



1) We first look at the even nodes in the graph. They are: 2, 16, and 10.

2) Second, we find if they have any immediate children nodes, and we find **1, 3, 10**

3) Finally, we add their values up, and we get that the result for this tree is: **14.** Notice that we don't care about whether the children are odd or even, as we just look at the parent being even.

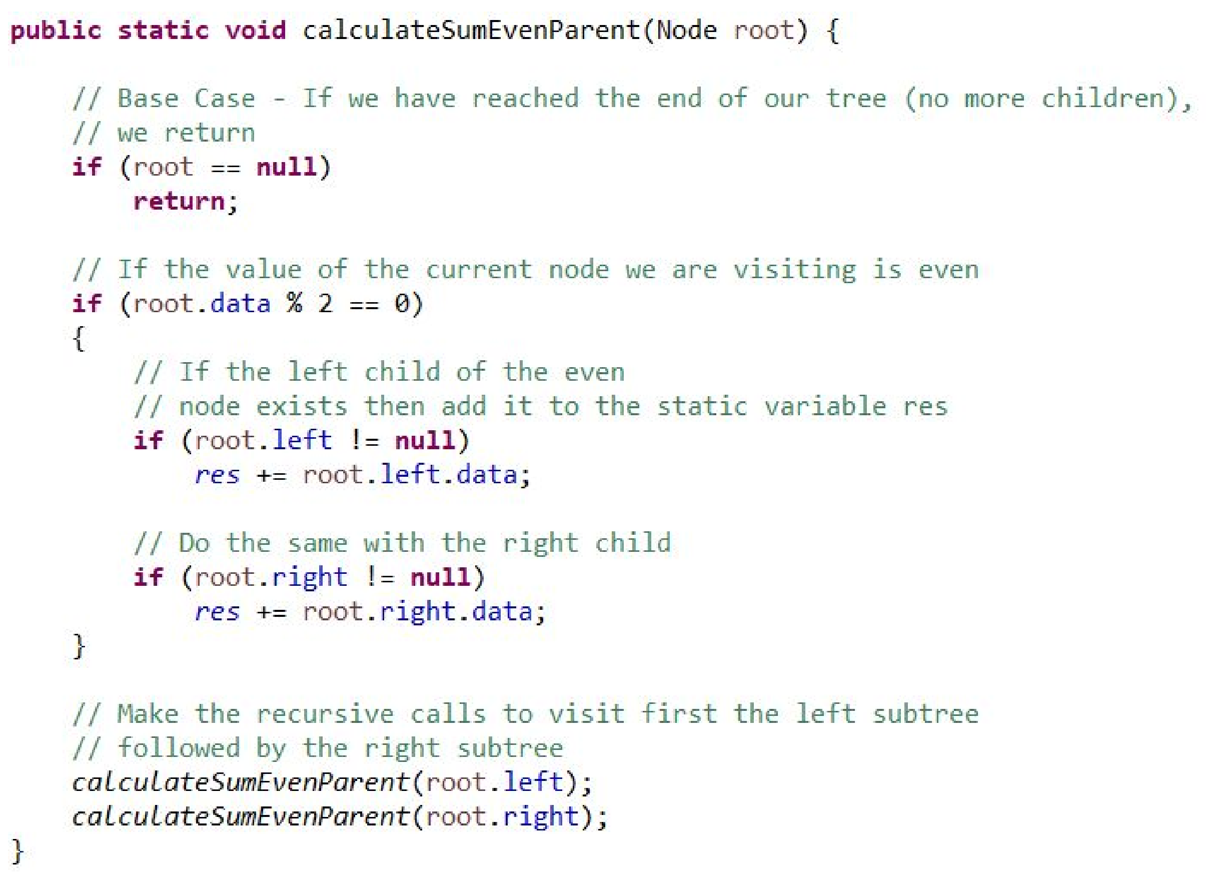
Solution

This problem presents a straightforward solution by traversing the tree as a whole. The main idea is to move through the entire tree until we reach the base case.

1) First, we set up a counter that starts at 0 and that is going to keep the total count of the nodes that we find on our way that have an even parent.
2) We choose a base case that will tell us that the problem has been finished, as there is not anymore leaves to visit.
3) We check if the node that we are currently visiting is even, we are going to obtain the values of all the children (maximum 2 since it is a binary tree) and add those to the total summation.
4) Then, we are going to make the recursive call to go towards the left child node (this will repeat all the steps)
5) Finally, we make the recursive call to go towards the right child node (this will repeat all the steps as well)

After out program has traversed through every possible node, our static variable sum will have the total value that we need.
Below we can see a representation of the program.

Programming Solution to the problem:



```java
public static void calculateSumEvenParent(Node root) {

    // Base Case - If we have reached the end of our tree (no more children),
    // we return
    if (root == null)
        return;

    // If the value of the current node we are visiting is even
    if (root.data % 2 == 0)
    {
        // If the left child of the even
        // node exists then add it to the static variable res
        if (root.left != null)
            res += root.left.data;

        // Do the same with the right child
        if (root.right != null)
            res += root.right.data;
    }

    // Make the recursive calls to visit first the left subtree
    // followed by the right subtree
    calculateSumEvenParent(root.left);
    calculateSumEvenParent(root.right);
}
```

Review:

We went over a simple implementation of the tree problem, where we used recursive calls to visit all the nodes in the tree, and then check if they were even in order to sum up their children. We also utilized a global or static variable in order to keep the total count.

Some questions to consider:

- Can you think of other possible ways of traversing the tree?
- Would you change something if the tree was not binary, or if it was a binary search tree?