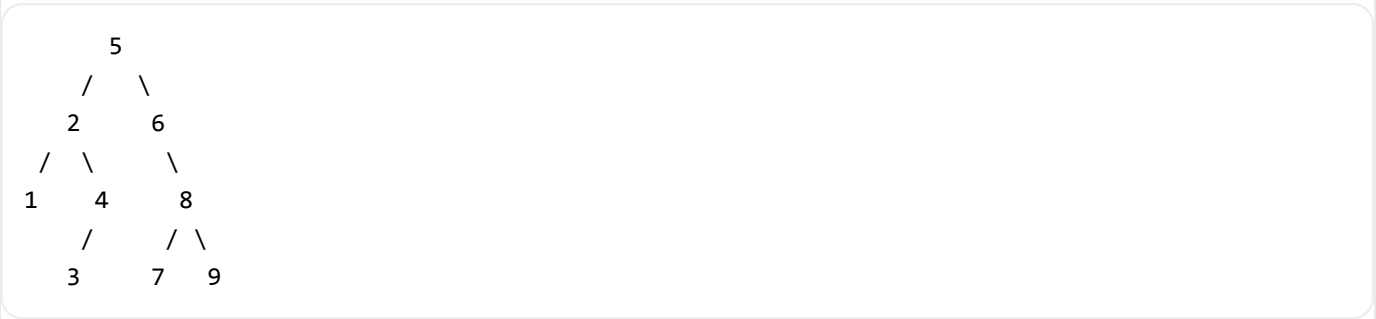
Difference between sums of odd level and even level nodes of a Binary Tree

Given a a Binary Tree, find the difference between the sum of nodes at odd level and the sum of nodes at even level. Consider root as level 1, left and right children of root as level 2 and so on.

For example, in the following tree, sum of nodes at odd level is (5 + 1 + 4 + 8) which is 18. And sum of nodes at even level is (2 + 6 + 3 + 7 + 9) which is 27. The output for following tree should be 18 – 27 which is -9.



A straightforward method is to **use level order traversal**. In the traversal, check level of current node, if it is odd, increment odd sum by data of current node, otherwise increment even sum. Finally return difference between odd sum and even sum. See following for implementation of this approach.

C implementation of level order traversal based approach to find the difference.

The problem can also be solved **using simple recursive traversal**. We can recursively calculate the required difference as, value of root's data subtracted by the difference for subtree under left child and the difference for subtree under right child.

Following is the implementation of this approach.

C

```
// A recursive program to find difference between sum of nodes at
// odd level and sum at even level
#include <stdio.h>
#include <stdlib.h>
```

```
// Binary Tree node
struct node
{
    int data;
    struct node* left, *right;
};

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

// The main function that return difference between odd and even level
// nodes
int getLevelDiff(struct node *root)
{
   // Base case
   if (root == NULL)
         return 0;

   // Difference for root is root's data - difference for left subtree
   // - difference for right subtree
   return root->data - getLevelDiff(root->left) - getLevelDiff(root->right);
}

// Driver program to test above functions
int main()
{
    struct node *root = newNode(5);
    root->left = newNode(2);
    root->right = newNode(6);
    root->left->left  = newNode(1);
    root->left->right = newNode(4);
    root->left->right->left = newNode(3);
    root->right->right = newNode(8);
    root->right->right->right = newNode(9);
    root->right->right->left = newNode(7);
    printf("%d is the required difference\n", getLevelDiff(root));
    getchar();
    return 0;
}
```

Run on IDE

Java

```
// A recursive java program to find difference between sum of nodes at
// odd level and sum at even level

// A binary tree node
class Node {

    int data;
    Node left, right;

    Node(int item) {
        data = item;
        left = right;
```

```
    }
}

class BinaryTree {

    // The main function that return difference between odd and even level
    // nodes

    static Node root;

    int getLevelDiff(Node node) {
        // Base case
        if (node == null) {
            return 0;
        }

        // Difference for root is root's data - difference for left subtree
        // - difference for right subtree
        return node.data - getLevelDiff(node.left) - getLevelDiff(node.right);
    }

    // Driver program to test above functions
    public static void main(String args[]) {

        BinaryTree tree = new BinaryTree();
        tree.root = new Node(5);
        tree.root.left = new Node(2);
        tree.root.right = new Node(6);
        tree.root.left.left = new Node(1);
        tree.root.left.right = new Node(4);
        tree.root.left.right.left = new Node(3);
        tree.root.right.right = new Node(8);
        tree.root.right.right.right = new Node(9);
        tree.root.right.right.left = new Node(7);
        System.out.println(tree.getLevelDiff(root) + " is the required difference");

    }
}

// This code has been contributed by Mayank Jaiswal
```

Run on IDE

Python

```
# A recursive program to find difference between sum of nodes
# at odd level and sum at even level

# A Binary Tree node
class Node:

    # Constructor to create a new node
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

# The main function that returns difference between odd and
# even level nodes
def getLevelDiff(root):

    # Base Case
```

```
    if root is None:
        return 0

    # Difference for root is root's data - difference for
    # left subtree - difference for right subtree
    return (root.data - getLevelDiff(root.left)-
        getLevelDiff(root.right))

# Driver program to test above function
root = Node(5)
root.left = Node(2)
root.right = Node(6)
root.left.left = Node(1)
root.left.right = Node(4)
root.left.right.left = Node(3)
root.right.right = Node(8)
root.right.right.right = Node(9)
root.right.right.left = Node(7)
print "%d is the required difference" %(getLevelDiff(root))

# This code is contributed by Nikhil Kumar Singh(nickzuck_007)
```

Run on IDE

Output:

```
-9 is the required difference
```

Time complexity of both methods is O(n), but the second method is simple and easy to implement.

This article is contributed by **Chandra Prakash**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

46 Comments Category: Trees

Related Posts:

- Check if removing an edge can divide a Binary Tree in two halves
- Check sum of Covered and Uncovered nodes of Binary Tree
- Lowest Common Ancestor in a Binary Tree | Set 2 (Using Parent Pointer)
- Construct a Binary Search Tree from given postorder
- BFS vs DFS for Binary Tree
- Maximum difference between node and its ancestor in Binary Tree
- Inorder Non-threaded Binary Tree Traversal without Recursion or Stack
- Check if leaf traversal of two Binary Trees is same?