



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

Difficulty Level : Easy • Last Updated : 09 Jun, 2021

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.



Recommended: Please solve it on "**PRACTICE**" first, before moving on to the solution.

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.

This approach is provided by [Mandeep Singh](#). For **Iterative approach**, simply traverse the tree level by level (level order traversal), store sum of node values in even no. level in evenSum and rest in variable oddSum and finally return the difference.

Below is the simple implementation of the approach.

C++

```
// CPP program to find
// difference between
// sums of odd level
// and even level nodes
// of binary tree
#include <bits/stdc++.h>
using namespace std;

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

```
// returns a new
// tree Node
Node* newNode(int data)
{
    Node* temp = new Node();
    temp->data = data;
    temp->left = temp->right = NULL;
    return temp;
}

// return difference of
// sums of odd level
// and even level
int evenOddLevelDifference(Node* root)
{
    if (!root)
        return 0;

    // create a queue for
    // level order traversal
    queue<Node*> q;
    q.push(root);

    int level = 0;
    int evenSum = 0, oddSum = 0;

    // traverse until the
    // queue is empty
    while (!q.empty())
    {
        int size = q.size();
        level += 1;

        // traverse for
        // complete level
        while(size > 0)
```



```
{
    Node* temp = q.front();
    q.pop();

    // check if level no.
    // is even or odd and
    // accordingly update
    // the evenSum or oddSum
    if(level % 2 == 0)
        evenSum += temp->data;
    else
        oddSum += temp->data;

    // check for left child
    if (temp->left)
    {
        q.push(temp->left);
    }

    // check for right child
    if (temp->right)
    {
        q.push(temp->right);
    }
    size -= 1;
}

return (oddSum - evenSum);
}

// driver program
int main()
{
    // construct a tree
    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);

int result = evenOddLevelDifference(root);
cout << "difference between sums is :: ";
cout << result << endl;
return 0;
}

// This article is contributed by Mandeep Singh.
```

Java



```
// Java program to find
// difference between
// sums of odd level
// and even level nodes
// of binary tree
import java.io.*;
import java.util.*;
// User defined node class
class Node {
    int data;
    Node left, right;

    // Constructor to create a new tree node
    Node(int key)
    {
        data = key;
        left = right = null;
    }
}
```

```
    }  
}  
class GFG {  
    // return difference of  
    // sums of odd level and even level  
    static int evenOddLevelDifference(Node root)  
    {  
        if (root == null)  
            return 0;  
  
        // create a queue for  
        // level order traversal  
        Queue<Node> q = new LinkedList<>();  
        q.add(root);  
  
        int level = 0;  
        int evenSum = 0, oddSum = 0;  
  
        // traverse until the  
        // queue is empty  
        while (q.size() != 0) {  
            int size = q.size();  
            level++;  
  
            // traverse for complete level  
            while (size > 0) {  
                Node temp = q.remove();  
  
                // check if level no.  
                // is even or odd and  
                // accordingly update  
                // the evenSum or oddSum  
                if (level % 2 == 0)  
                    evenSum += temp.data;  
                else  
                    oddSum += temp.data;  
            }  
        }  
    }  
}
```

```
// check for left child
if (temp.left != null)
    q.add(temp.left);

// check for right child
if (temp.right != null)
    q.add(temp.right);
size--;
    }
}
return (oddSum - evenSum);
}


// Driver code
public static void main(String args[])
{
    // construct a tree
    Node root = new Node(5);
    root.left = new Node(2);
    root.right = new Node(6);
    root.left.left = new Node(1);
    root.left.right = new Node(4);
    root.left.right.left = new Node(3);
    root.right.right = new Node(8);
    root.right.right.right = new Node(9);
    root.right.right.left = new Node(7);

    System.out.println("difference between sums is " +
        evenOddLevelDifference(root));
}


// This code is contributed by rachana soma
```




Python3




```
# Python3 program to find maximum product
# of a level in Binary Tree
```



```
# Helper function that allocates a new
# node with the given data and None
# left and right poers.
```



```
class newNode:
```



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

# return difference of sums of odd
# level and even level
def evenOddLevelDifference(root):

    if (not root):
        return 0

    # create a queue for
    # level order traversal
    q = []
    q.append(root)

    level = 0
    evenSum = 0
    oddSum = 0

    # traverse until the queue is empty
    while (len(q)):

        size = len(q)
        level += 1

        # traverse for complete level
```



```
while(size > 0):

    temp = q[0] #.front()
    q.pop(0)

    # check if level no. is even or
    # odd and accordingly update
    # the evenSum or oddSum
    if(level % 2 == 0):
        evenSum += temp.data
    else:
        oddSum += temp.data

    # check for left child
    if (temp.left) :

        q.append(temp.left)

    # check for right child
    if (temp.right):

        q.append(temp.right)

    size -= 1

return (oddSum - evenSum)

# Driver Code
if __name__ == '__main__':

    """
    Let us create Binary Tree shown
    in above example """
    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)

result = evenOddLevelDifference(root)
print("Difffence between sums is", result)
```

```
# This code is contributed by
# Shubham Singh(SHUBHAMSINGH10)
```

C#



```
// C# program to find
// difference between
// sums of odd level
// and even level nodes
// of binary tree
using System;
using System.Collections.Generic;

// User defined node class
public class Node
{
    public int data;
    public Node left, right;

    // Constructor to create a new tree node
    public Node(int key)
    {
        data = key;
        left = right = null;
    }
}
```

```
public class GFG
{
    // return difference of
    // sums of odd level and even level
    static int evenOddLevelDifference(Node root)
    {
        if (root == null)
            return 0;

        // create a queue for
        // level order traversal
        Queue<Node> q = new Queue<Node>();
        q.Enqueue(root);

        int level = 0;
        int evenSum = 0, oddSum = 0;

        // traverse until the
        // queue is empty
        while (q.Count != 0)
        {
            int size = q.Count;
            level++;

            // traverse for complete level
            while (size > 0)
            {
                Node temp = q.Dequeue();

                // check if level no.
                // is even or odd and
                // accordingly update
                // the evenSum or oddSum
                if (level % 2 == 0)
                    evenSum += temp.data;
                else
```



```
        oddSum += temp.data;

        // check for left child
        if (temp.left != null)
            q.Enqueue(temp.left);

        // check for right child
        if (temp.right != null)
            q.Enqueue(temp.right);
        size--;
    }
}
return (oddSum - evenSum);
}





// Driver code
public static void Main(String []args)
{
    // construct a tree
    Node root = new Node(5);
    root.left = new Node(2);
    root.right = new Node(6);
    root.left.left = new Node(1);
    root.left.right = new Node(4);
    root.left.right.left = new Node(3);
    root.right.right = new Node(8);
    root.right.right.right = new Node(9);
    root.right.right.left = new Node(7);

    Console.WriteLine("difference between sums is " +
        evenOddLevelDifference(root));
}

// This code is contributed by 29AjayKumar
```



Javascript



```
<script>
// Javascript program to find
// difference between
// sums of odd level
// and even level nodes
// of binary tree

// User defined node class
class Node
{
    constructor(key)
    {
        this.data = key;
        this.left = this.right = null;
    }
}

// return difference of
// sums of odd level and even level
function evenOddLevelDifference(root)
{
    if (root == null)
        return 0;

    // create a queue for
    // level order traversal
    let q = [];
    q.push(root);

    let level = 0;
    let evenSum = 0, oddSum = 0;

    // traverse until the
    // queue is empty
```

```
while (q.length != 0) {
    let size = q.length;
    level++;

    // traverse for complete level
    while (size > 0) {
        let temp = q.shift();

        // check if level no.
        // is even or odd and
        // accordingly update
        // the evenSum or oddSum
        if (level % 2 == 0)
            evenSum += temp.data;
        else
            oddSum += temp.data;

        // check for left child
        if (temp.left != null)
            q.push(temp.left);

        // check for right child
        if (temp.right != null)
            q.push(temp.right);
        size--;
    }
}

return (oddSum - evenSum);
}

// Driver code
let root = new Node(5);
    root.left = new Node(2);
    root.right = new Node(6);
    root.left.left = new Node(1);
    root.left.right = new Node(4);
    root.left.right.left = new Node(3);
```



```
root.right.right = new Node(8);  
root.right.right.right = new Node(9);  
root.right.right.left = new Node(7);  
  
document.write("diffence between sums is " +  
               evenOddLevelDifference(root)+"<br>");  
  
// This code is contributed by rag2127  
</script>
```

Output:

```
diffence between sums is -9
```

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.

Below 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 <bits/stdc++.h>
using namespace std;

// Binary Tree node
class node
{
public:
    int data;
    node* left, *right;
};

// A utility function to allocate
// a new tree node with given data
node* newNode(int data)
{
    node* Node = new 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(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 code
int main()
{
    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);
    cout<<getLevelDiff(root)<<" is the required difference\n";
    return 0;
}

// This code is contributed by rathbhupendra
```

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;
}
```


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
    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(tree.root) +
                                   " is the required difference");
    }
}
```

}

// This code has been contributed by Mayank Jaiswal

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)
```

C#



```
using System;
```



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



```
// A binary tree node
public class Node
{
    public int data;
    public Node left, right;

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

```
public class BinaryTree
{
    // The main function that return difference
    // between odd and even level nodes
```



```
public Node root;

public virtual 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);
    Console.WriteLine(tree.getLevelDiff(tree.root)
        + " is the required difference");
}

}

// This code is contributed by Shrikant13
```



Javascript



```
<script>
// A recursive javascript program to find difference between sum of nodes at
// odd level and sum at even level

// A binary tree node
class Node
{
    constructor(item)
    {
        this.data = item;
        this.left = this.right = null;
    }
}

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

function getLevelDiff(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
root = new Node(5);
```



```
root.left = new Node(2);
root.right = new Node(6);
root.left.left = new Node(1);
root.left.right = new Node(4);
root.left.right.left = new Node(3);
root.right.right = new Node(8);
root.right.right.right = new Node(9);
root.right.right.left = new Node(7);
document.write(getLevelDiff(root) +
               " is the required difference");

// This code is contributed by avanitrachhadiya2155
</script>
```

Output:

-9 is the required difference

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



Difference between sums of odd level and even level nodes of a Binary T...



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.

Attention reader! Don't stop learning now. Get hold of all the important DSA concepts with the [DSA Self Paced Course](#) at a student-friendly price and become industry ready. To complete your preparation from learning a language to DS Algo and many more, please refer [Complete Interview Preparation Course](#).

In case you wish to attend live classes with industry experts, please refer [Geeks Classes Live](#)



Looking for math superheroes!



Like 18

Next >

Find maximum level sum in Binary Tree

RECOMMENDED ARTICLES

Page : [1](#) [2](#) [3](#)

Difference between sums of odd level and

Print even positioned nodes of odd levels in

01 even level nodes in an N-ary Tree
07, Sep 20

02 Difference between sums of odd position and even position nodes for each level of a Binary Tree
30, Apr 19

03 Modify Binary Tree by replacing all nodes at even and odd levels by their nearest even or odd perfect squares respectively
22, Feb 21

04 Print odd positioned nodes of odd levels in level order of the given binary tree
05, Sep 19

05 level order of the given binary tree
06, Sep 19

06 Print odd positioned nodes of even levels in level order of the given binary tree
06, Sep 19

07 Print even positioned nodes of even levels in level order of the given binary tree
29, Apr 19

08 Difference between sum of even and odd valued nodes in a Binary Tree
09, Jul 20



Article Contributed By :



Vote for difficulty

Current difficulty : [Easy](#)

Easy

Normal

Medium

Hard

Expert

Improved By : [shrikanth13](#), [rachana soma](#), [SHUBHAMSINGH10](#), [29AjayKumar](#), [rathbhupendra](#), [rag2127](#),
[avanitrachhadiya2155](#)

Article Tags : [Amazon](#), [Tree](#)

Practice Tags : [Amazon](#), [Tree](#)


Improve Article

Report Issue

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments



 5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305

 feedback@geeksforgeeks.org



Company[About Us](#)[Careers](#)[Privacy Policy](#)[Contact Us](#)[Copyright Policy](#)**Learn**[Algorithms](#)[Data Structures](#)[Languages](#)[CS Subjects](#)[Video Tutorials](#)**Practice**[Courses](#)[Company-wise](#)[Topic-wise](#)[How to begin?](#)**Contribute**[Write an Article](#)[Write Interview Experience](#)[Internships](#)[Videos](#)

@geeksforgeeks , Some rights reserved

