Trees
Session 7

Children Sum Parent

Given a Binary Tree. Check whether all of its nodes have the value equal to the sum of their child nodes.

Example 1:

Input:

```
    10
   /
  10
```

Output: 1

Explanation: Here, every node is the sum of its left and right child.

Example 2:

Input:

```
      1
    /   \
   4     3
  /  \
 5    N
```

Output: 0

Expected Time Complexiy: O(N).

Expected Auxiliary Space: O(Height of the Tree).

Constraints:

1 <= N <= 105

1 <= Data on nodes <= 105

```
1.  int isSumProperty(Node *root)
2.  {
3.     int lefts=0;
4.     int rights=0;
5.
6.     if(root==NULL || (!root->left && !root->right))
7.        return 1;
8.     else
9.     {
10.       if(root->left)
11.          lefts=root->left->data;
12.       if(root->right)
13.          rights=root->right->data;
14.       if((lefts+rights==root->data) && (isSumProperty(root->left)) &&
    (isSumProperty(root->right)))
15.          return 1;
16.       return 0;
17.    }
18. }
```

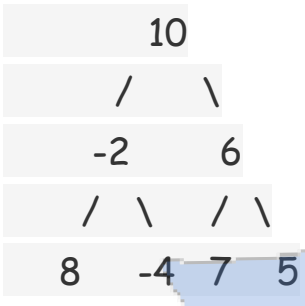
https://practice.geeksforgeeks.org/problems/children-sum-parent/1/?category[]=Tree&category[]=Tree&difficulty[]=0&page=1&query=category[]Treedifficulty[]0page1category[]Tree#

Transform to Sum Tree

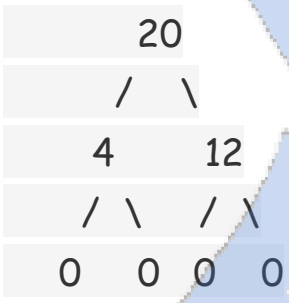
Given a Binary Tree of size N , where each node can have positive or negative values. Convert this to a tree where each node contains the sum of the left and right subtrees of the original tree. The values of leaf nodes are changed to 0.

Example 1:

Input:



Output:



Expected Time Complexity: O(N)

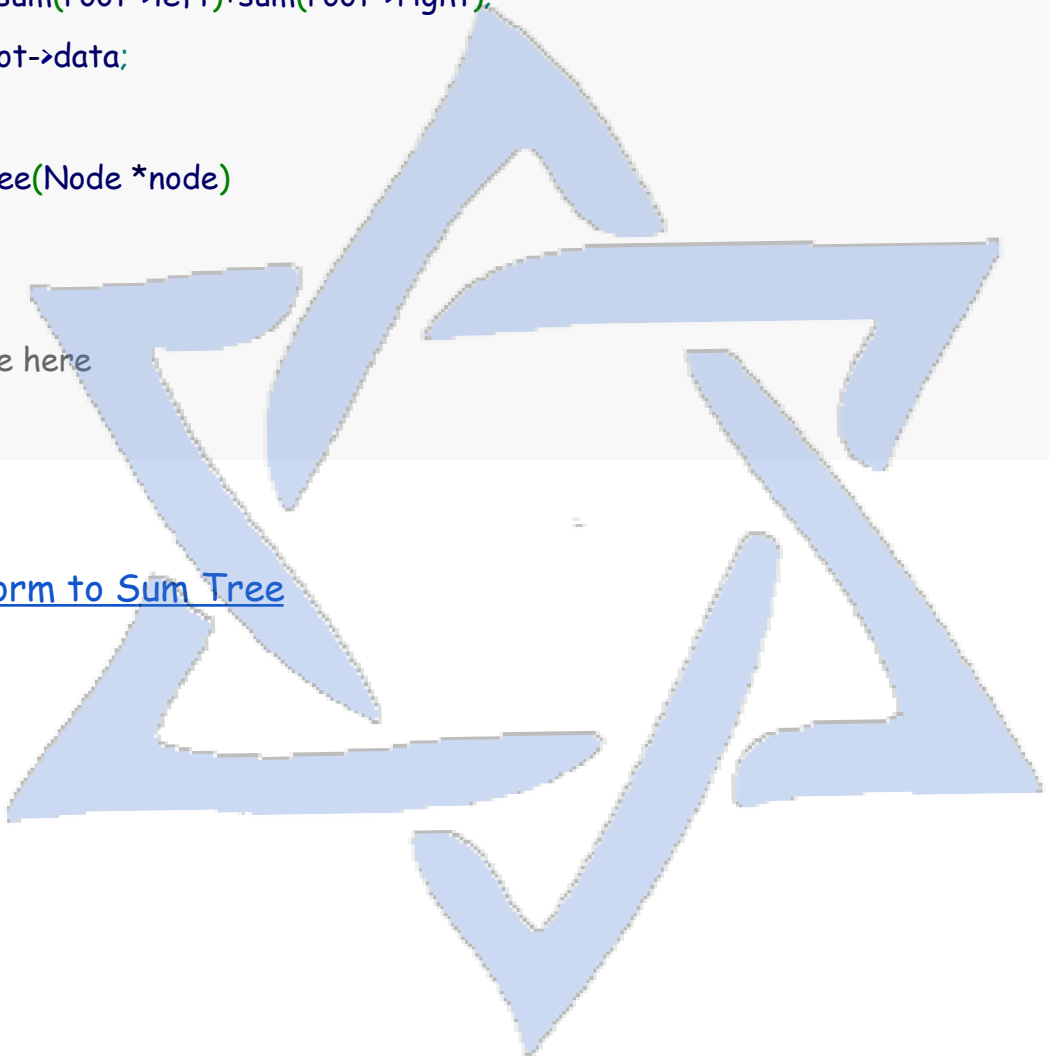
Expected Auxiliary Space: O(height of tree)

Constraints:

1 ≤ N ≤ 104

```
1.  int sum(Node*root)
2.  {
3.     if(root==NULL)
4.        return 0;
5.     int lefts=0,rights=0;
6.     int x=root->data;
7.
8.     root->data=sum(root->left)+sum(root->right);
9.     return x+root->data;
10. }
11. void toSumTree(Node *node)
12. {
13.    sum(node);
14.    // Your code here
15. }
```

Transform to Sum Tree

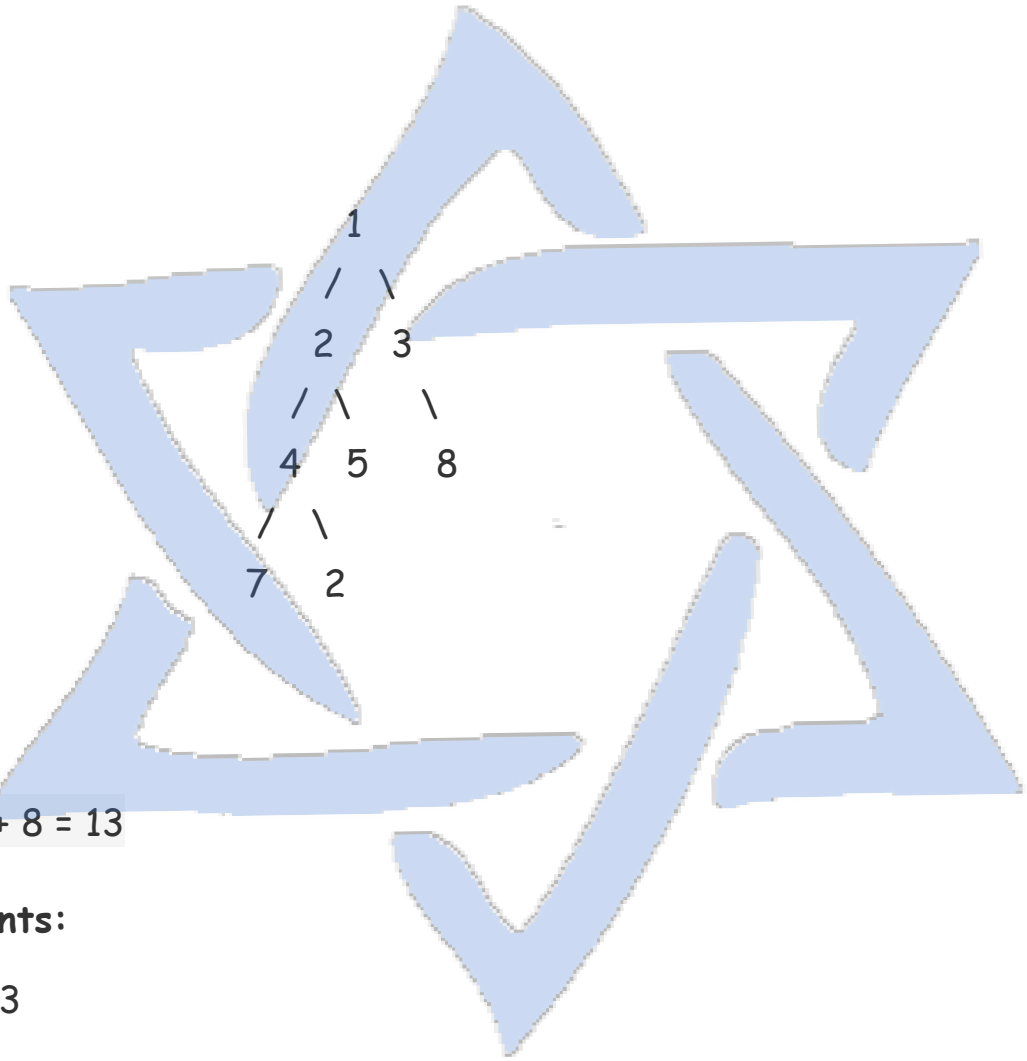


Sum of Leaf Nodes at Min Level

Given a Binary Tree of size **N**, your task is to complete the function **minLeafSum()**, that should return the sum of all the leaf nodes that are at minimum level of the given binary tree.

Example:

Input :



Output :

sum = 5 + 8 = 13

Constraints:

1<=T<=103

0<=N<=103

https://practice.geeksforgeeks.org/problems/sum-of-leaf-nodes-at-min-level/1/?category[]=Tree&category[]=Tree&difficulty[]=0&page=2&query=category[]Treedifficulty[]0page2category[]Tree

Maximum path sum from any node

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

Example 1:

Input:

Output: 32

Explanation: Path in the given tree goes like 10 , 2 , 20 which gives the max sum as 32.

Expected Time Complexity: O(N).

Expected Auxiliary Space: O(Height of the Tree).

Constraints:

1 <= Number of nodes <= 103

1 <= |Data on node| <= 104

```
1.    int path(Node*root,int &res)
2.    {
3.       if(root==NULL)
4.         return 0;
5.
6.         int lefts=path(root->left,res);
7.         int rights=path(root->right,res);
8.
9.         int max_single=max(max(lefts,rights)+root->data,root->data);
10.
11.        int max_top=max(max_single,lefts+rights+root->data);
12.
13.        res=max(res,max_top);
14.
15.        return max_single;
16.   }
17.     //Function to return maximum path sum from any node in a tree.
18.     int findMaxSum(Node* root)
19.     {
20.        int res=INT_MIN;
21.        path(root,res);
22.        return res;
23.        // Your code goes here
24.     }
```

https://practice.geeksforgeeks.org/problems/maximum-path-sum-from-any-node/1/?category[]=Tree&category[]=Tree&problemStatus=unsolved&difficulty[]=1&page=1&query=category[]TreeproblemStatusunsolveddifficulty[]1page1category[]Tree#

Maximum Path Sum between 2 Leaf Nodes

Given a binary tree in which each node element contains a number. Find the maximum possible sum from one leaf node to another.

Example 1:

Input :

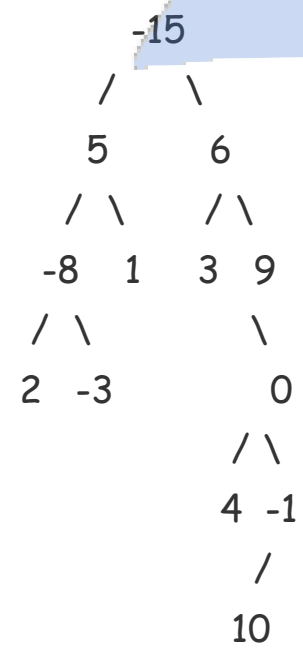
Output: 16

Explanation :

Maximum Sum lies between leaf node 4 and 5. 4 + 4 + 3 + 5 = 16.

Example 2:

Input :



Output : 27

Explanation:

The maximum possible sum from one leaf node
to another is (3 + 6 + 9 + 0 + -1 + 10 = 27)

Expected Time Complexity: O(N)

Expected Auxiliary Space: O(Height of Tree)

Constraints:

1 ≤ N ≤ 10^4

```
1.    int findMaxUtil(Node* root, int &res)
2.    {
3.
4.        if (root == NULL)
5.            return 0;
6.
7.        if (!root->left && !root->right)
8.            return root->data;
9.
10.       int l = findMaxUtil(root->left,res);
11.
12.       int r = findMaxUtil(root->right,res);
13.
```

```
14.         if (root->left && root->right)
15.         {
16.             res = max(res, l + r + root->data);
17.             return max(l, r) + root->data;
18.         }
19.
20.         return (!root->left) ? r + root->data : l + root->data;
21.     }
22.
23.     int maxPathSum(Node *root)
24.     {
25.         int res = INT_MIN;
26.
27.         findMaxUtil(root, res);
28.         return res;
29.     }
30.
```

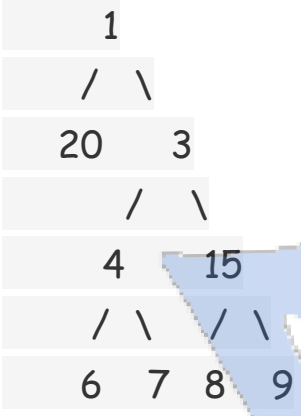
https://practice.geeksforgeeks.org/problems/maximum-path-sum/1/?category[]=Tree&category[]=Tree&difficulty[]=2&page=1&query=category[]Treedifficulty[]2page1category[]Tree#

Paths from root with a specified sum

Given a Binary tree and a sum **S**, print all the paths, starting from root, that sums upto the given sum. Path may not end on a leaf node.

Example 1:

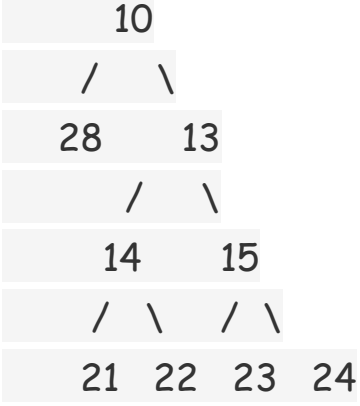
Input :
sum = 8,
Root of tree



Output :
1 3 4
Explanation : Sum of path 1, 3, 4 = 8.

Example 2:

Input :
sum = 38,
Root of tree



Output :

10 28

10 13 15

Explanation :

Sum of path 10, 28 = 38 , and,

Sum of path 10, 13, 15 = 38.

Expected Time Complexity : O(N)

Expected Time Complexity : O(N)

Your Task :

1<=N<=10^5

1<=sum<=10^6

```
1.    void path(Node*root,int sum , int &now_sum,vector<int>&v,vector<vector<int>>&V)
2.      {
3.         if(root==NULL)
4.            return;
5.         v.push_back(root->key);
6.         now_sum+=root->key;
7.         if(now_sum>=sum)
8.         {
9.            if(now_sum==sum)
10.              V.push_back(v);
11.           v.pop_back();
12.           now_sum=now_sum-root->key;
13.           return;
14.        }
15.        if(root->left)
16.           path(root->left,sum,now_sum,v,V);
```

```
        if(root->right)
            path(root->right,sum,now_sum,v,V);
        now_sum=now_sum - root->key;
        v.pop_back();
    }
    vector<vector<int>> printPaths(Node *root, int sum)
    {
        vector<vector<int>>V;
        vector<int>v;
        int now_sum =0;
        path(root,sum,now_sum,v,V);
        return V;
        //code here
    }
```

https://practice.geeksforgeeks.org/problems/paths-from-root-with-a-specified-sum/1/?category[]=Tree&category[]=Tree&problemStatus=unsolved&difficulty[]=1&page=2&query=category[]TreeproblemStatusunsolveddifficulty[]1page2category[]Tree#

Check if two Nodes are Cousins

Given the binary Tree of and two-node values. Check whether the two-node values are cousins of each other or not.

Example 1:

Input:

a = 2, b = 3

Output: 0

Example 2:

Input:

a = 5, b = 4

Output: 1

Explanation: Here, nodes 5 and 4 areat the same level and have different parent nodes. Hence, they both are cousins

Expected Time Complexity: O(N).

Expected Auxiliary Space: O(Height of the Tree).

Constraints:

1<=Number of Nodes<=1000

```
1.  int sibling(Node*root,int a , int b)
2.  {
3.     if(root==NULL)
4.        return 0;
5.     return ((root->left && root->right && root->left->data==a && root->right->data==b) ||
6.           (root->left && root->right && root->left->data==b && root->right->data==a) ||
7.           sibling(root->left,a,b)||sibling(root->right,a,b));
8.  }
9.  int level(Node*root,int a ,int l)
10. {
11.    if(root==NULL)
12.       return 0;
13.
14.    if(root->data==a)
15.       return l;
16.
17.    int lev=level(root->left,a,l+1);
18.    if(lev)
19.       return lev;
20.    level(root->right,a,l+1);
21. }
22.bool isCousins(Node *root, int a, int b)
23.{
24.    if(level(root,a,1)==level(root,b,1) && !sibling(root,a,b))
25.       return true;
26.    return false;
27.  }
```

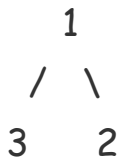
https://practice.geeksforgeeks.org/problems/check-if-two-nodes-are-cousins/1/?category[]=Tree&category[]=Tree&difficulty[]=0&page=2&query=category[]Treedifficulty[]0page2category[]Tree

Reverse alternate levels of a perfect binary tree

Given a complete binary tree, reverse the nodes present at alternate levels.

Example 1:

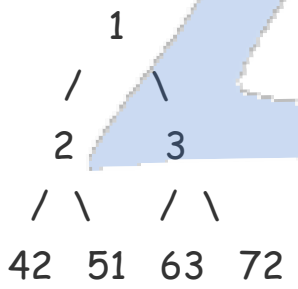
Input:



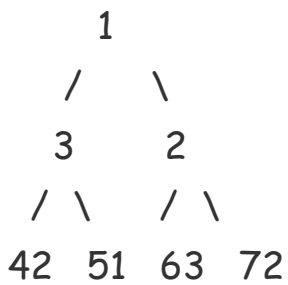
Output:

Example 2:

Input:



Output:



Explanation:
Nodes at level 2 are reversed.Level 1 and 3 remain as it is.

Expected Time Complexity: O(N)

Expected Auxiliary Space: O(height of tree)

Constraints:

1 ≤ N ≤ 10^4

```
1.  void preorder(struct Node *root1, struct Node*root2, int lvl)
2. {
3.       if (root1 == NULL || root2==NULL)
4.        Return;
5.
6.     if (lvl%2 == 0)
7.        swap(root1->data, root2->data);
8.
9.      preorder(root1->left, root2->right, lvl+1);
10.       preorder(root1->right, root2->left, lvl+1);
11.}
12.    void reverseAlternate(Node *root)
13.    {
14.       preorder(root->left, root->right, 0);
15.    }
```

1. https://practice.geeksforgeeks.org/problems/reverse-alternate-levels-of-a-perfect-binary-tree/1/?category[]=Tree&category[]=Tree&difficulty[]=0&page=2&query=category[]Treedifficulty[]0page2category[]Tree
2. https://practice.geeksforgeeks.org/problems/exchange-the-leaf-nodes/1/?category[]=Tree&category[]=Tree&difficulty[]=0&page=2&query=category[]Treedifficulty[]0page2category[]Tree

3. https://practice.geeksforgeeks.org/problems/transform-to-sum-tree/1/?category[]=Tree&category[]=Tree&difficulty[]=0&page=1&query=category[]Treedifficulty[]0page1category[]Tree

4. https://practice.geeksforgeeks.org/problems/max-and-min-element-in-binary-tree/1/?category[]=Tree&category[]=Tree&difficulty[]=0&page=1&query=category[]Treedifficulty[]0page1category[]Tree
5. https://practice.geeksforgeeks.org/problems/children-sum-parent/1/?category[]=Tree&category[]=Tree&difficulty[]=0&page=1&query=category[]Treedifficulty[]0page1category[]Tree#
6. https://practice.geeksforgeeks.org/problems/sum-of-right-leaf-nodes/1/?category[]=Tree&category[]=Tree&difficulty[]=0&page=1&query=category[]Treedifficulty[]0page1category[]Tree
7. https://practice.geeksforgeeks.org/problems/sum-of-left-leaf-nodes/1/?category[]=Tree&category[]=Tree&difficulty[]=0&page=1&query=category[]Treedifficulty[]0page1category[]Tree
8. Diameter of Binary Tree
9. https://practice.geeksforgeeks.org/problems/sum-of-leaf-nodes-at-min-level/1/?category[]=Tree&category[]=Tree&difficulty[]=0&page=2&query=category[]Treedifficulty[]0page2category[]Tree
10. https://practice.geeksforgeeks.org/problems/check-if-two-nodes-are-cousins/1/?category[]=Tree&category[]=Tree&difficulty[]=0&page=2&query=category[]Treedifficulty[]0page2category[]Tree

11. https://practice.geeksforgeeks.org/problems/reverse-alternate-levels-of-a-perfect-binary-tree/1/?category[]=Tree&category[]=Tree&difficulty[]=0&page=2&query=category[]Treedifficulty[]0page2category[]Tree
12. https://practice.geeksforgeeks.org/problems/bbt-counter4914/1/?category[]=Tree&category[]=Tree&difficulty[]=0&page=2&query=category[]Treedifficulty[]0page2category[]Tree
13. https://practice.geeksforgeeks.org/problems/maximum-sum-of-non-adjacent-nodes/1/?category[]=Tree&category[]=Tree&problemStatus=unsolved&difficulty[]=1&page=1&query=category[]TreeproblemStatusunsolveddifficulty[]1page1category[]Tree