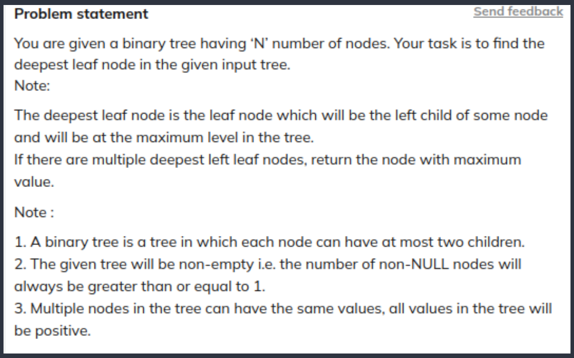
Deepest Left



Problem statement Send feedback

You are given a binary tree having 'N' number of nodes. Your task is to find the deepest leaf node in the given input tree.

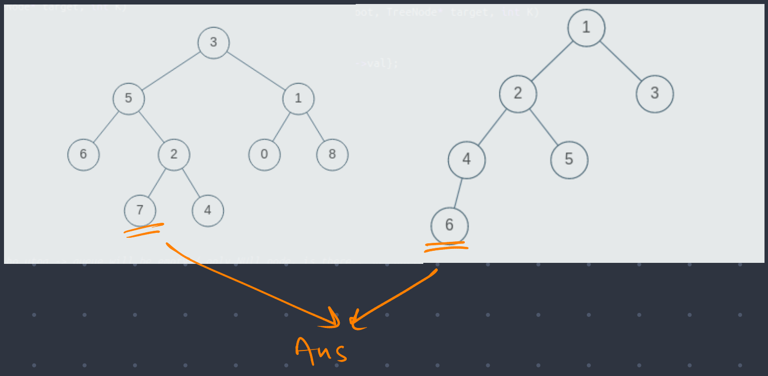
Note:

The deepest leaf node is the leaf node which will be the left child of some node and will be at the maximum level in the tree.

If there are multiple deepest left leaf nodes, return the node with maximum value.

Note :

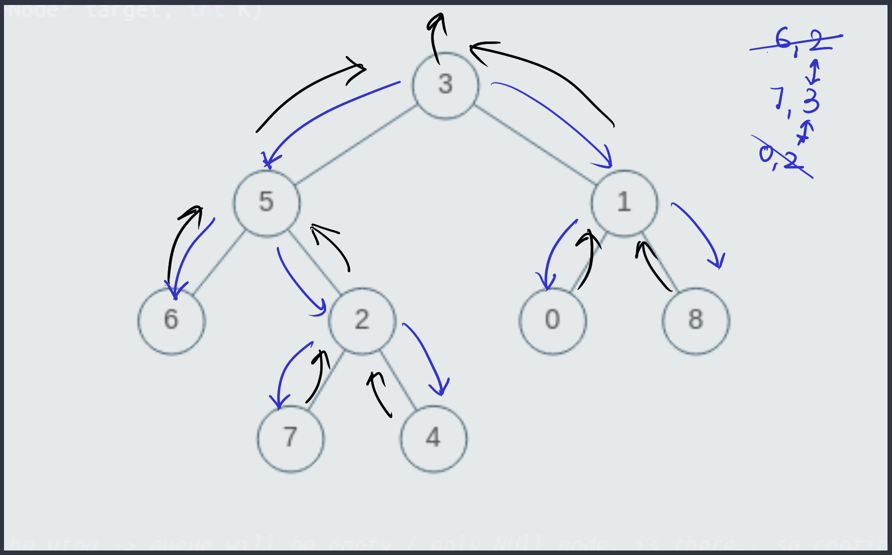
1. A binary tree is a tree in which each node can have at most two children.
2. The given tree will be non-empty i.e. the number of non-NULL nodes will always be greater than or equal to 1.
3. Multiple nodes in the tree can have the same values, all values in the tree will be positive.

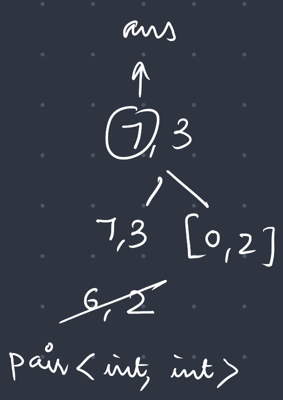


→ Approach:

pair < int, int >

Node data — level





```cpp
#include <bits/stdc++.h>
/*
        Tree Node class.

        class BinaryTreeNode
        {
                T data;
                BinaryTreeNode<T> *left;
                BinaryTreeNode<T> *right;

                BinaryTreeNode(T data) {
                        this->data = data;
                        left = NULL;
                        right = NULL;
                }
        }
*/

void solve(BinaryTreeNode<int> *root, int level, pair<int, int> &p) {
  if (root == NULL)
    return;

  if (root->left) {
    if (p.second < level + 1 && !root->left->left && !root->left->right) {
      p.first = root->left->data;
      p.second = level + 1;
    }
    if (p.second == level + 1 && !root->left->left && !root->left->right) {
      if (p.first < root->left->data) {
        p.first = root->left->data;
      }
    }
    solve(root->left, level + 1, p);
  }
  solve(root->right, level + 1, p);
}

int deepestLeftLeafNode(BinaryTreeNode<int> *root) {
  int level = 0;
  pair<int, int> p;
  solve(root, level, p);
  return p.first;
}
```

Time Complexity : O(n)

Space Complexity : O(n)