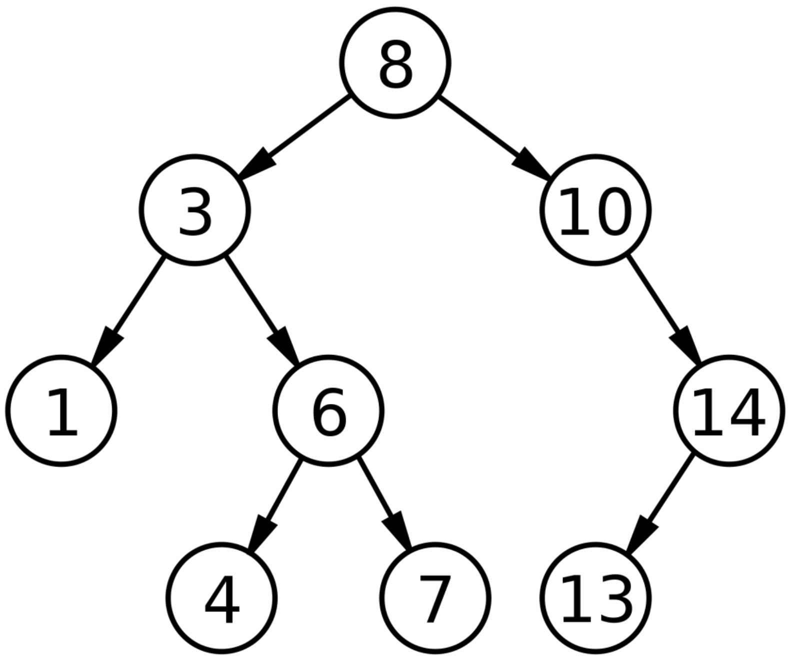
Maximum difference between node and its ancestor in Binary Tree

Given a binary tree, we need to find maximum value we can get by subtracting value of node B from value of node A, where A and B are two nodes of the binary tree and A is an ancestor of B. Expected time complexity is O(n).

For example, consider below binary tree



The above image is taken from wiki.

We can have various ancestor-node difference, some of which are given below :
8 – 3 = 5
3 – 7 = -4
8 – 1 = 7
10 – 13 = -3
. . . .

But among all those differences maximum value is 7 obtained by subtracting 1 from 8, which we need to return as result.

We strongly recommend you to minimize your browser and try this yourself first.

As we are given a binary tree, there is no relationship between node values so we need to traverse whole binary tree to get max difference and we can obtain the result in one traversal only by following below steps :
If we are at leaf node then just return its value because it can’t be ancestor of any node. Then at each internal node we will try to get minimum value from left subtree and right subtree and calculate the difference between node value and this minimum value and according to that we will update the result.
As we are calculating minimum value while retuning in recurrence we will check all optimal possibilities (checking node value with minimum subtree value only) of differences and hence calculate the result in one traversal only.

Below is C++ implementation of above idea.

```
// C++ program to find maximum difference between node
// and its ancestor
#include <bits/stdc++.h>
using namespace std;

/* A binary tree node has key, pointer to left
   child and a pointer to right child */
struct Node
{
    int key;
    struct Node* left, *right;
};

/* To create a newNode of tree and return pointer */
struct Node* newNode(int key)
{
    Node* temp = new Node;
    temp->key = key;
    temp->left = temp->right = NULL;
    return (temp);
}

/* Recursive function to calculate maximum ancestor-node
   difference in  binary tree. It updates value at 'res'
   to store the result.  The returned value of this function
   is minimum value in subtree rooted with 't' */
int maxDiffUtil(Node* t, int *res)
{
    /* Returning Maximum int value if node is not
       there (one child case)  */
    if (t == NULL)
        return INT_MAX;

    /* If leaf node then just return node's value  */
    if (t->left == NULL && t->right == NULL)
        return t->key;

    /* Recursively calling left and right subtree
       for minimum value  */
    int val = min(maxDiffUtil(t->left, res),
                  maxDiffUtil(t->right, res));

    /* Updating res if (node value - minimum value
       from subtree) is bigger than res  */
    *res = max(*res, t->key - val);

    /* Returning minimum value got so far */
    return min(val, t->key);
}

/* This function mainly calls maxDiffUtil() */
int maxDiff(Node *root)
{
    // Initialising result with minimum int value
    int res = INT_MIN;

    maxDiffUtil(root, &res);

    return res;
}

/* Helper function to print inorder traversal of
  binary tree   */
void inorder(Node* root)
{
    if (root)
    {
        inorder(root->left);
        printf("%d ", root->key);
        inorder(root->right);
    }
}

// Driver program to test above functions
int main()
{
    // Making above given diagram's binary tree
    Node* root;
    root = newNode(8);
    root->left = newNode(3);

    root->left->left = newNode(1);
    root->left->right = newNode(6);
    root->left->right->left = newNode(4);
    root->left->right->right = newNode(7);

    root->right = newNode(10);
    root->right->right = newNode(14);
    root->right->right->left = newNode(13);

    printf("Maximum difference between a node and"
           " its ancestor is : %d\n", maxDiff(root));
}
```

Run on IDE

Output :

```
Maximum difference between a node and its ancestor is : 7
```

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

16 Comments Category: Trees

Related Posts:

- 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
- Inorder Non-threaded Binary Tree Traversal without Recursion or Stack
- Check if leaf traversal of two Binary Trees is same?
- Closest leaf to a given node in Binary Tree
- Locking and Unlocking of Resources arranged in the form of n-ary Tree

(Login to Rate and Mark)

2.7 Average Difficulty : **2.7/5.0**
Based on **7** vote(s)

Add to TODO List

Mark as DONE

Like Share 13 people like this.

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