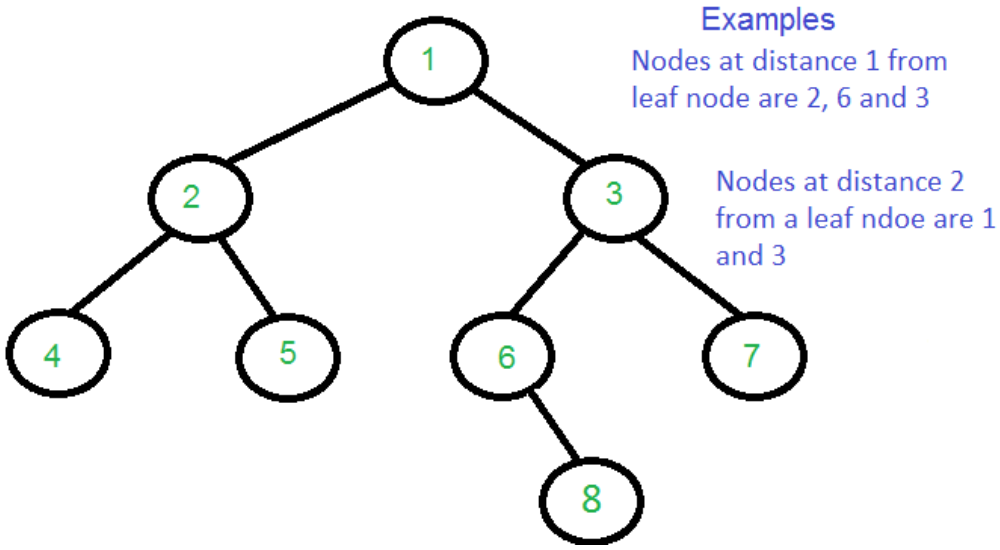
Print all nodes that are at distance k from a leaf node

Given a Binary Tree and a positive integer k, print all nodes that are distance k from a leaf node.

Here the meaning of distance is different from previous post. Here k distance from a leaf means k levels higher than a leaf node. For example if k is more than height of Binary Tree, then nothing should be printed. Expected time complexity is O(n) where n is the number nodes in the given Binary Tree.



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

The idea is to traverse the tree. Keep storing all ancestors till we hit a leaf node. When we reach a leaf node, we print the ancestor at distance k. We also need to keep track of nodes that are already printed as output. For that we use a boolean array visited[].

Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion
Geometric Algorithms

```
/* Program to print all nodes which are at distance k from a leaf */
#include <iostream>
using namespace std;
#define MAX_HEIGHT 10000

struct Node
{
    int key;
    Node *left, *right;
};

/* utility that allocates a new Node with the given key  */
Node* newNode(int key)
{
    Node* node = new Node;
    node->key = key;
    node->left = node->right = NULL;
    return (node);
}
```

```
/* This function prints all nodes that are distance k from a leaf node
   path[] --> Store ancestors of a node
   visited[] --> Stores true if a node is printed as output.  A node m
                 distance away from many leaves, we want to print it o
void kDistantFromLeafUtil(Node* node, int path[], bool visited[],
                          int pathLen, int k)
{
    // Base case
    if (node==NULL) return;

    /* append this Node to the path array */
    path[pathLen] = node->key;
    visited[pathLen] = false;
    pathLen++;

    /* it's a leaf, so print the ancestor at distance k only
       if the ancestor is not already printed  */
    if (node->left == NULL && node->right == NULL &&
        pathLen-k-1 >= 0 && visited[pathLen-k-1] == false)
    {
        cout << path[pathLen-k-1] << " ";
        visited[pathLen-k-1] = true;
        return;
    }

    /* If not leaf node, recur for left and right subtrees */
    kDistantFromLeafUtil(node->left, path, visited, pathLen, k);
```


Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

```
    kDistantFromLeafUtil(node->right, path, visited, pathLen, k);
}

/* Given a binary tree and a nuber k, print all nodes that are k
   distant from a leaf*/
void printKDistantfromLeaf(Node* node, int k)
{
    int path[MAX_HEIGHT];
    bool visited[MAX_HEIGHT] = {false};
    kDistantFromLeafUtil(node, path, visited, 0, k);
}

/* Driver program to test above functions*/
int main()
{
    // Let us create binary tree given in the above example
    Node * root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);
    root->right->left = newNode(6);
    root->right->right = newNode(7);
    root->right->left->right = newNode(8);

    cout << "Nodes at distance 2 are: ";
    printKDistantfromLeaf(root, 2);

    return 0;
}
```

Output:

```
Nodes at distance 2 are: 3 1
```

Time Complexity: Time Complexity of above code is O(n) as the code does a simple tree traversal.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Related Tpoics:

- Print a Binary Tree in Vertical Order | Set 2 (Hashmap based Method)
- Print Right View of a Binary Tree
- Red-Black Tree | Set 3 (Delete)
- Construct a tree from Inorder and Level order traversals
- Print all nodes at distance k from a given node
- Print a Binary Tree in Vertical Order | Set 1
- Interval Tree
- Check if a given Binary Tree is height balanced like a Red-Black Tree

Writing code in comment? Please use **ideone.com** and share the link here.

Recent Comments

affiszerv Your example has two 4s on row 3, that's why it...

Backtracking | Set 7 (Sudoku) · 25 minutes ago

RVM Can someone please elaborate this Qs from above...

Flipkart Interview | Set 6 · 45 minutes ago

Vishal Gupta I talked about as an Interviewer in general,...

Software Engineering Lab, Samsung Interview | Set 2 · 45 minutes ago

@meya Working solution for question 2 of 4f2f round....

Amazon Interview | Set 53 (For SDE-1) · 1 hour ago

sandeep void rearrange(struct node *head) {...

Given a linked list, reverse alternate nodes and append at the end · 2 hours ago

Neha I think that is what it should return as, in...

Find depth of the deepest odd level leaf node · 2 hours ago