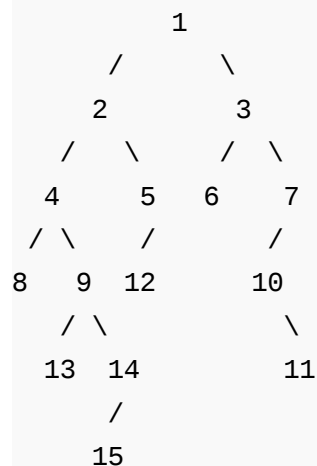


Remove all nodes which don't lie in any path with sum $\geq k$

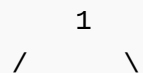
Given a binary tree, a complete path is defined as a path from root to a leaf. The sum of all nodes on that path is defined as the sum of that path. Given a number K, you have to remove (prune the tree) all nodes which don't lie in any path with sum $\geq k$.

Note: A node can be part of multiple paths. So we have to delete it only in case when all paths from it have sum less than K.

Consider the following Binary Tree



For input $k = 20$, the tree should be changed to following (Nodes with values 6 and 8 are deleted)



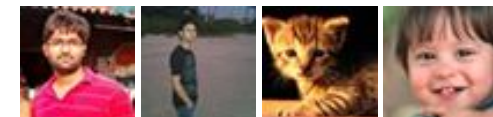
Google™ Custom Search



GeeksforGeeks



52,731 people like [GeeksforGeeks](#).



[Interview Experiences](#)

[Advanced Data Structures](#)

[Dynamic Programming](#)

[Greedy Algorithms](#)

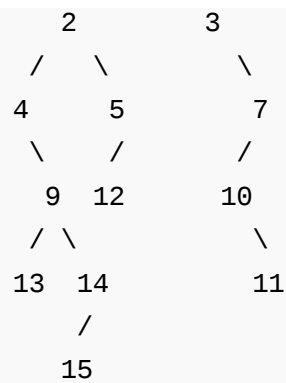
[Backtracking](#)

[Pattern Searching](#)

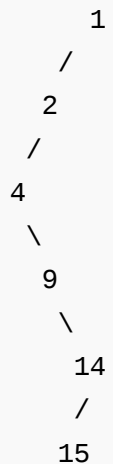
[Divide & Conquer](#)

[Mathematical Algorithms](#)

[Recursion](#)



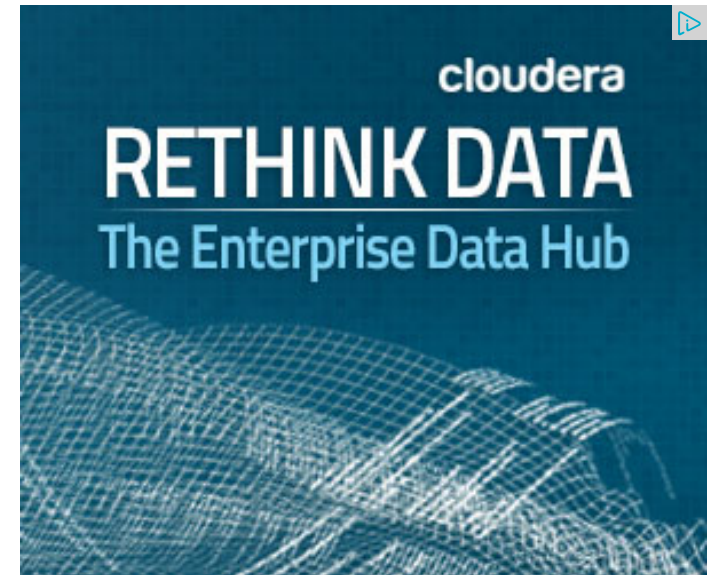
For input $k = 45$, the tree should be changed to following.



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

The idea is to traverse the tree and delete nodes in bottom up manner. While traversing the tree, recursively calculate the sum of nodes from root to leaf node of each path. For each visited node, checks the total calculated sum against given sum “k”. If sum is less than k, then free(delete) that node (leaf node) and return the sum back to the previous node. Since the path is from root to leaf and nodes are deleted in bottom up manner, a node is deleted only when all of its descendants are deleted. Therefore, when a node is deleted, it must be a leaf in the current Binary Tree.

Following is C implementation of the above approach.



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](#)

```

#include <stdio.h>
#include <stdlib.h>

// A utility function to get maximum of two integers
int max(int l, int r) { return (l > r ? l : r); }

// A Binary Tree Node
struct Node
{
    int data;
    struct Node *left, *right;
};

// A utility function to create a new Binary 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;
}

// print the tree in LVR (Inorder traversal) way.
void print(struct Node *root)
{
    if (root != NULL)
    {
        print(root->left);
        printf("%d ", root->data);
        print(root->right);
    }
}

/* Main function which truncates the binary tree. */
struct Node *pruneUtil(struct Node *root, int k, int *sum)
{
    // Base Case
    if (root == NULL) return NULL;

    // Initialize left and right sums as sum from root to
    // this node (including this node)
    int lsum = *sum + (root->data);
    int rsum = lsum;

    // Recursively prune left and right subtrees
    root->left = pruneUtil(root->left, k, &lsum);

```

Shouldn't
you expect
a cloud with:

**ONE-CLICK
DEPLOYMENTS**

Experience the
Managed Cloud
Difference

TRY TODAY ►

 **rackspace**
the open cloud company

```

root->right = pruneUtil(root->right, k, &rsum);

// Get the maximum of left and right sums
*sum = max(lsum, rsum);

// If maximum is smaller than k, then this node
// must be deleted
if (*sum < k)
{
    free(root);
    root = NULL;
}

return root;
}

// A wrapper over pruneUtil()
struct Node *prune(struct Node *root, int k)
{
    int sum = 0;
    return pruneUtil(root, k, &sum);
}

// Driver program to test above function
int main()
{
    int k = 45;
    struct 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->left->left->left = newNode(8);
    root->left->left->right = newNode(9);
    root->left->right->left = newNode(12);
    root->right->right->left = newNode(10);
    root->right->right->left->right = newNode(11);
    root->left->left->right->left = newNode(13);
    root->left->left->right->right = newNode(14);
    root->left->left->right->right->left = newNode(15);

    printf("Tree before truncation\n");
    print(root);

    root = prune(root, k); // k is 45

```

Recent Comments

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

[Backtracking | Set 7 \(Sudoku\)](#) · 27 minutes ago

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

[Flipkart Interview | Set 6](#) · 47 minutes ago

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

[Software Engineering Lab, Samsung Interview | Set 2](#) · 47 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

AdChoices 

[► Java Programming](#)

[► Binary Tree](#)

```
printf("\n\nTree after truncation\n");
print(root);

return 0;
}
```

Output:

Tree before truncation


8 4 13 9 15 14 2 12 5 1 6 3 10 11 7

Tree after truncation

4 9 15 14 2 1

Time Complexity: $O(n)$, the solution does a single traversal of given Binary Tree.


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.

AdChoices 

[▶ Java Tree](#)

[▶ Java to C++](#)

[▶ Remove Tree Root](#)

AdChoices 

[▶ Java Source Code](#)

[▶ Prune Tree](#)

[▶ Java Array](#)



Better Than Hadoop.

HPCC Systems is Big Data Processing and Analytics
Open Source. Proven. Trusted.

 LexisNexis® [Learn More !\[\]\(16d8d87f3f3b2bd45357825eb854fccf_img.jpg\)](#)

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](#)



17



Tweet

3



1

Writing code in comment? Please use [ideone.com](#) and share the link here.

@geeksforgeeks, **Some rights reserved**

Contact Us!

Powered by **WordPress** & **MooTools**, customized by geeksforgeeks team