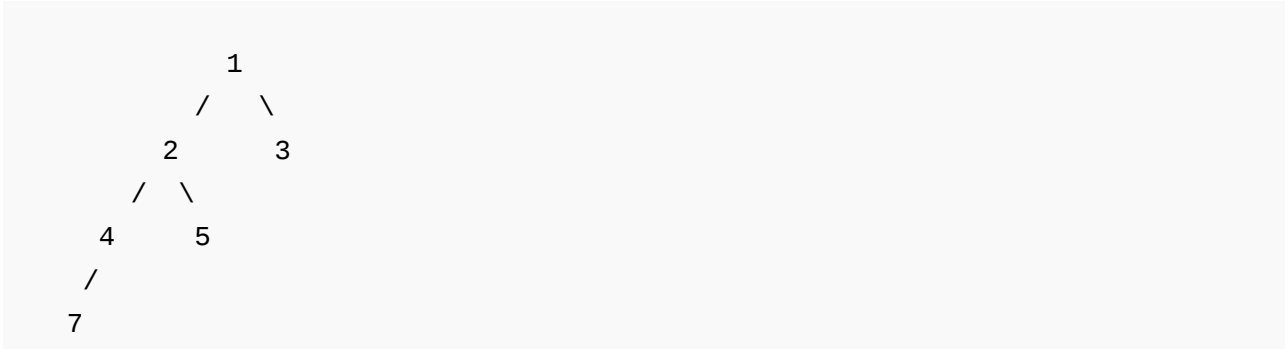
Print Ancestors of a given node in Binary Tree

Given a Binary Tree and a key, write a function that prints all the ancestors of the key in the given binary tree.

For example, if the given tree is following Binary Tree and key is 7, then your function should print 4, 2 and 1.

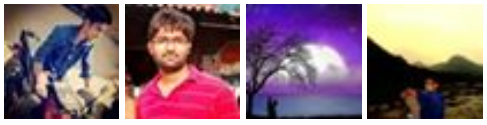


Thanks to Mike , Sambasiva and wgpshashank for their contribution.

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

using namespace std;

/* A binary tree node has data, pointer to left child
   and a pointer to right child */
struct node
{
    int data;
    struct node* left;
    struct node* right;
```

```
};

/* If target is present in tree, then prints the ancestors
   and returns true, otherwise returns false. */
bool printAncestors(struct node *root, int target)
{
  /* base cases */
  if (root == NULL)
     return false;

  if (root->data == target)
     return true;

  /* If target is present in either left or right subtree of this node
     then print this node */
  if ( printAncestors(root->left, target) ||
       printAncestors(root->right, target) )
  {
    cout << root->data << " ";
    return true;
  }

  /* Else return false */
  return false;
}

/* Helper function that allocates a new node with the
   given data and NULL left and right pointers. */
struct node* newnode(int data)
{
  struct node* node = (struct node*)
                      malloc(sizeof(struct node));
  node->data = data;
  node->left = NULL;
  node->right = NULL;

  return(node);
}

/* Driver program to test above functions*/
int main()
{

  /* Construct the following binary tree
              1
            /   \
          2      3
```


Popular Posts

```
          /   \
        4      5
      /
    7
  */
  struct node *root = newnode(1);
  root->left        = newnode(2);
  root->right       = newnode(3);
  root->left->left  = newnode(4);
  root->left->right = newnode(5);
  root->left->left->left  = newnode(7);

  printAncestors(root, 7);

  getchar();
  return 0;
}
```

Output:

4 2 1

Time Complexity: O(n) where n is the number of nodes in the given Binary Tree.

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

695 Subscribe

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

7 Tweet 0 2

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

23 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

aMUchbetteraproach • a month ago

```
#include<iostream>

#include<stdio.h>

#include<stdlib.h>
```

Recent Comments

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

Backtracking | Set 7 (Sudoku) · 43 minutes ago

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

Flipkart Interview | Set 6 · 1 hour ago

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

Software Engineering Lab, Samsung Interview | Set 2 · 1 hour 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 · 3 hours ago

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

Find depth of the deepest odd level leaf node · 3 hours ago

```
using namespace std;

/* A binary tree node has data, pointer to left child

and a pointer to right child */

struct node

{

int data;

struct node* left;

struct node* right;
```

see more

1 · Reply · Share ›

Guest • 7 months ago

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

struct node
{
int data;
struct node* left;
struct node* right;
};

struct node* newnode(int data)
{
struct node* node = (struct node*)
malloc(sizeof(struct node));
```

```
node->data = data;
node->left = NULL;
node->right = NULL;
```

see more

Reply • Share ›

Guest ➔ Guest • 7 months ago

Both recursive and itrative.

Reply • Share ›

Jignesh • 10 months ago

```
public boolean ancestor(Node root, int value) {


        if (root == null)
            return false;
        if (root.key == value)
            return true;
        if(ancestor(root.leftChild, value) || ancestor(root.rightChi
            System.out.println("Value is: "+ root.key);
            return true;
        }
        else
            return false;



    }
```

1 Reply • Share ›

Sumit Monga • 10 months ago

A very easy way is to use a "static" variable which keeps track of whether the
variable retains its value between function calls.

```
void ancestor(struct node * root, int key).
{
if(root==NULL)
return;
static int success = 0;.
if(root->data == key).
{
success = 1;.
return;
}
if(success!=1)
ancestor(root->left, key);
if(success!=1)
ancestor(root->right, key);
if(success ==1)
printf("%d ", root->data);
}
```


Tapas Mahanta • 10 months ago

pretty neat

vishal • 11 months ago

// Algorithm
// Current stack Contains all the ancestors nodes

```
 void ancestors(node_t* root , int k)
{
```

```
        if( root)
        {
                push(root -> key); // or node itself can be
                 // stored instead of values
                if( root -> key == k)
                        display_stack(); // and exit
                else
                {
                        ancestors( root -> left , k);
                        ancestors( root -> right ,k);
                        pop();
                }
        }
}
```

▲ | ▼ • Reply • Share ›

shek8034 ➦ vishal • 10 months ago

Why u are using your own stack if you r doing it with recursion. ?
Just use the recursion stack.
Stack is needed if you do it iteratively

▲ | ▼ • Reply • Share ›

vishal ➦ shek8034 • 9 months ago

I am using stack to store all the ancestors . If I use the recursio
I come out of the function everything is lost.

▲ | ▼ • Reply • Share ›

Iterative version(running for all cases):

```
 public void printAncestorIterative(Node root,Node a)
{
      System.out.println();

    int flag=0;
    if (root == null)
    {
        return;
    }

    Queue<Object> q = new LinkedList<Object>();
    q.add(root);
    q.add(root.value + " ");

    while(!q.isEmpty()){
```

see more

• Reply • Share ›

abhishek08aug • a year ago

C++ code:

```
#include <iostream>
#include <stdlib.h>
using namespace std;

class tree_node {
  private:
```

```
    int data;
    tree_node * left;
    tree_node * right;
  public:
    tree_node() {
      left=NULL;
      right=NULL;
    }
    void set_data(int data) {
```

see more

Reply • Share ›

Audu Dan'azumi Pindiga • a year ago

Boko no eary.

Reply • Share ›

Anil arya • 2 years ago

```
#include<stdio.h>

#include<stdlib.h>
struct node
{
        struct node *left;
        struct node *right;
        int data ;
};
int ance[122];
void print_arr(int ance[],int len)
{
```

```
for(i=0;i<len;i++)
{
        printf("%d ",ance[i]);
}
```

see more

Reply • Share ›

rituraj • 2 years ago

My algo :

1. Find the level of the given node say k.
2. Now ,Do a level order traversal up to (k-1)th level and print all nodes encour

PS:I might be wrong ,So plz comment on this post

Reply • Share ›

kg1020 ➦ rituraj • 2 years ago

suppose k= 3 then according to u. print all the nodes up to level 2. then printed whereas only one node from level 2 & level 1 should be printed.

Reply • Share ›

John • 2 years ago

[sourcecode language="C#"]
private void ancestor(Node root,ref bool found,int data)
{
//bool found = false;
if (root != null)
{

if (root.Data > data)
{

```
ancestor(root.Left,ref found,data);
}
else if (root.Data < data)
{
ancestor(root.Right,ref found,data);
}
if (found)
{
Console.WriteLine(root.Data);
}
if (root.Data == data)
{
found = true;
}
}
}
```


guest ➚ John • 2 years ago

your code assumes this is BST

manishj • 3 years ago

Iterative approach(as you do a pre-order tarversal ,stack itself always contains

```
btree* rightvisited[100]= {NULL};
int searchinrightvisited(btree * ptr)
{
        for(int i = 0; i <100;i++)
        {
                if(ptr == rightvisited[i])
```

```
                        return 1;
        }
        return 0;
}
void printpath(btree *root, int key)
{
        btree * current = root;
        stack<btree *> st;
        bool done = false;
```

see more

Reply • Share ›

Ankit Gupta manishj • 2 years ago

Nice. Pushing off the stack when its right branch has been examined f

Reply • Share ›

KC • 3 years ago

1. Do an iterative DFS with root of the tree as the starting vertex.
2. pass the value of the node along with root in each pass such as DFS(root, r
3. When n == root->data, put n on the stack and print the stack.

Reply • Share ›

aimless KC • 3 years ago

can you write the code?

Reply • Share ›

Sandeep KC • 3 years ago

@KC:
This approach looks an iterative version of the approach given in post.
recursion.

Reply • Share ›

wgpshashank • 3 years ago

@geeksfrogeek @Mike The Only extra asked in dat question that we have to p
..although its serious because a node can't b ancestor of itself so if u wants u
node it self

2nd also pass the node instead of int value in printAncester() method e.g pass

  • Reply • Share ›

 Subscribe Add Disqus to your site