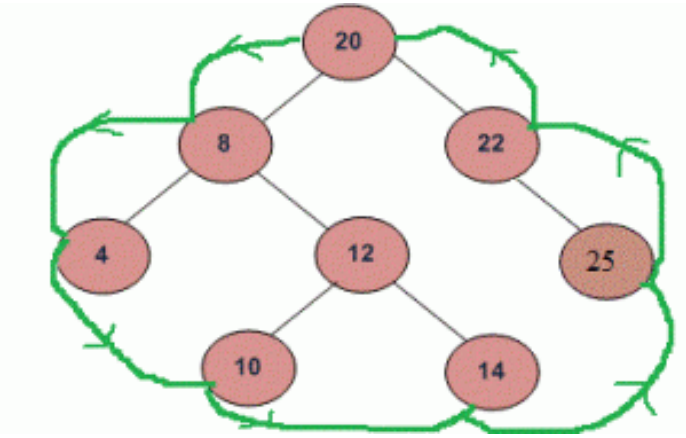
# Boundary Traversal of binary tree



Given a binary tree, print boundary nodes of the binary tree Anti-Clockwise starting from the root. For example, boundary traversal of the following tree is “20 8 4 10 14 25 22″




We break the problem in 3 parts:
**1.** Print the left boundary in top-down manner.
**2.** Print all leaf nodes from left to right, which can again be sub-divided into two sub-parts:
…..**2.1** Print all leaf nodes of left sub-tree from left to right.
…..**2.2** Print all leaf nodes of right subtree from left to right.
**3.** Print the right boundary in bottom-up manner.

We need to take care of one thing that nodes are not printed again. e.g. The left most node is also the leaf node of the tree.

Based on the above cases, below is the implementation:

```
/* program for boundary traversal of a binary tree */
```



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

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

// A simple function to print leaf nodes of a binary tree
void printLeaves(struct node* root)
{
    if ( root )
    {
        printLeaves(root->left);

        // Print it if it is a leaf node
        if ( !(root->left)  &&  !(root->right) )
            printf("%d ", root->data);

        printLeaves(root->right);
    }
}

// A function to print all left boundry nodes, except a leaf node.
// Print the nodes in TOP DOWN manner
void printBoundaryLeft(struct node* root)
{
    if (root)
    {
        if (root->left)
        {
            // to ensure top down order, print the node
            // before calling itself for left subtree
            printf("%d ", root->data);
            printBoundaryLeft(root->left);
        }
        else if( root->right )
        {
            printf("%d ", root->data);
            printBoundaryLeft(root->right);
        }
        // do nothing if it is a leaf node, this way we avoid
        // duplicates in output
    }
```

Geometric Algorithms



## 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

```
}

// A function to print all right boundry nodes, except a leaf node
// Print the nodes in BOTTOM UP manner
void printBoundaryRight(struct node* root)
{
    if (root)
    {
        if ( root->right )
        {
            // to ensure bottom up order, first call for right
            //  subtree, then print this node
            printBoundaryRight(root->right);
            printf("%d ", root->data);
        }
        else if ( root->left )
        {
            printBoundaryRight(root->left);
            printf("%d ", root->data);
        }
       // do nothing if it is a leaf node, this way we avoid
       // duplicates in output
    }
}


// A function to do boundary traversal of a given binary tree
void printBoundary (struct node* root)
{
    if (root)
    {
        printf("%d ",root->data);

        // Print the left boundary in top-down manner.
        printBoundaryLeft(root->left);

        // Print all leaf nodes
        printLeaves(root->left);
        printLeaves(root->right);

        // Print the right boundary in bottom-up manner
        printBoundaryRight(root->right);
    }
}

// A utility function to create a node
struct node* newNode( int data )
```

```
{
    struct node* temp = (struct node *) malloc( sizeof(struct node) );

    temp->data = data;
    temp->left = temp->right = NULL;

    return temp;
}

// Driver program to test above functions
int main()
{
    // Let us construct the tree given in the above diagram
    struct node *root          = newNode(20);
    root->left                 = newNode(8);
    root->left->left           = newNode(4);
    root->left->right          = newNode(12);
    root->left->right->left    = newNode(10);
    root->left->right->right   = newNode(14);
    root->right                = newNode(22);
    root->right->right         = newNode(25);

    printBoundary( root );

    return 0;
}
```

Output:

```
20 8 4 10 14 25 22
```

Time Complexity: O(n) where n is the number of nodes in binary tree.

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



## Recent Comments

## 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.

**61 Comments** GeeksforGeeks

Sort by Newest

hours ago

**AlienOnEarth** • 3 days ago

Another algorithm can be:

1.) print left view of the tree
2.) print all the leaves from left to right
3.) print right view of the tree

  • Reply • Share ›

**Kunal Arora** • 22 days ago

I don't think the above solution will work for skew trees either Left skew or right
wrong.

  • Reply • Share ›

**AlienOnEarth** Kunal Arora • 3 days ago

It will work for all type of trees

  • Reply • Share ›

**mccullum** • 3 months ago

20
\
8
/
4
/ \
10 14
\
22

for the tree above shudn't the ans be 20 10 22 14 8
bt the solution posted gives 20 22 14 4 8

Reply • Share ›

**Rahul** mccullum • 3 months ago

In the boundary traversal as mentioned we have to print .. first left subt
at last right subtree's boundary in bottom-up manner...
In ur example, 10 is neither in left subtree's boundary nor in right subtre
ans. i think should be 20 22 14 4 8

1 Reply • Share ›

**Sriharsha g.r.v** • 6 months ago

Hl..the same idea but code loooks simple here

```
printleft(struct node* node)

{

if (node->left == NULL)

return;

printf("%d ", node->data);

printleft(node->left);

}

printleaves(struct node*root)

{

if(root)
```

see more

Reply • Share ›

**Rahul** ➦ Sriharsha g.r.v • 3 months ago

your code will give incorrect output for this ...

20

/ \

8 22

\ \

12 25

/ \

10 14

the correct output should be . 20 8 12 10 14 25 22

but o/p according to ur code is . 20 10 14 25 22

  • Reply • Share ›

**Gautam** • 6 months ago

Hey i think your code will not properly work for
20->left=8 and right=22; 8->left= 4
and right= 12;
4->left=NULL and Right=NULL;
12->left=10 and Right=14;
10->left=13 and Right=NULL;
14->left=NULL and Right=NULL;
13->left=NULL and Right=15;
22->left=Null and Right=25;
then left boundary will b 20,8,4,10,13 but your code is not giving appropriate ar
plz correct me if i am wrong

**@Ankit** • 8 months ago

i think else if part in printBoundaryLeft and printBoundaryRight function is not n



**sh** • 8 months ago

Another approach could be, use BFS for the tree traversal and enter all the ele seperated by NULL entries.After completing traversal, start taking out element be printed, and entry before NULL will be pushed to stack, rest all the entries a empty, go on popping elements from the stack until it gets empty.

please mention if it is having problem with some cases.

```
/* Paste your code here (You may delete these lines if not writing c
```



**Amit Bgl** • 9 months ago

wow code :D



**punfabi** • 9 months ago

for trees in which root doesn't have left or right child..

```
/* Paste your code here (You may delete these lines if not writing c

// A function to do boundary traversal of a given binary tree
void printBoundary (struct node* root)
{
    if (root)
    {
        printf("%d ",root->data);
```

```
        // Print the left boundary in top-down manner.
        if( root->left)
            printBoundaryLeft(root->left);
        else
            printBoundaryLeft(root->right);

        // Print all leaf nodes
```

see more

• Reply • Share ›

**Akshay Jindal** • 9 months ago

Awesome!!...simplicity at its best..:D

1 • Reply • Share ›

**ushekokar** • 9 months ago

[sourcecode language="C++"]
void boundary(struct node *root1)
{
static struct node *t=root1;
static int leftflag=1;

if(root1->l==NULL&&root1->r==NULL)//dont print leaf nodes
return;

if(leftflag==1&&t->l!=NULL)//check if left of root is not null,if it is null dont go ins
{
cout<<" "<<root1->data;//print data first
if(root1->l!=NULL)
boundary(root1->l);
else if(root1->r!=NULL)

boundary(root1->r);
}

if(root1==t&&leftflag==1)//if left

see more

• Reply • Share ›

**Sunil** • 10 months ago

There is no need of the 'else if' part in both printBoundaryLeft() and printBound
Code works just fine without it.

• Reply • Share ›

**Himanshu** ➚ Sunil • 10 months ago

@Sunil
There is a need of else if in both functions.
Suppose in the above example node 8 has no left child.
Then the output should be 20 8 12 10 14 25 22.If you remove else if fro
20 10 14 25 22

• Reply • Share ›

**Sunil** ➚ Himanshu • 10 months ago

Thank you so much! It needed explanation!

• Reply • Share ›

**denial** • 10 months ago

I think this code solves problem. works in all case. Correct me if I'm wrong. :P

```
 #include <stdio.h>
#include <stdlib.h>
// structure declaration of tree node
struct tnode
{
```

```
        int data;
        struct tnode *left;
        struct tnode *right;
};
typedef struct tnode node;
node *newNode(int data)
{
        node *temp=(node *)malloc(sizeof(node));
        temp->data=data;
        temp->left=NULL;
        temp->right=NULL;
```

see more

1 ^ | v • Reply • Share ›

**sid** • 11 months ago

#include

using namespace std;

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

struct node *getnode(int data)
{
struct node *new_node =(struct node *)malloc(sizeof(struct node));
new_node->data=data;
new_node->left=NULL;

return new_node;
}

see more

Reply • Share ›

**koolkeshaw** • 11 months ago

@GeeksforGeeks

your algo will not work for a node satisfying following condition :-
i) it is neither on left boundary nor on right boundary
ii) only one of its child(either left OR right) is null

refer :
http://ideone.com/QtOivl

I think the correct o/p :- 20,8,23,24,25

Reply • Share ›

**koolkeshaw** koolkeshaw • 11 months ago

correct o/p :- 20,8,24,23,25

for correct algo
refer to
http://ideone.com/kaZWyD

Reply • Share ›

**abhinav** • 11 months ago

just level order traversal and print first and last nodes

```
/* Paste your code here (You may delete these lines if not writing c
```

Reply • Share ›

**Kunal Arora** ➦ abhinav • 22 days ago

In the last level you have to print all the values in queue and also the rig
approach.

  • Reply • Share ›

**Krishna 'TottaPhilic' Durai** • 11 months ago

What about trees like:

--------10
-------/
------5
-----/-
----3---7
---/----
--1---4---8

Here, wouldn&#039t the expected solution be: 10, 5, 3, 1, 4, 8, 7?
The solution posted here assumes the root has 2 children.

1   • Reply • Share ›

**KK** • 11 months ago

We can do with level order traversal also. Print left node of each level and stor
done with tree traversal print the stack in reverse order.

```
/* Paste your code here (You may delete these lines if not writing c
```

  • Reply • Share ›

**abhishek08aug** • 11 months ago

Intelligent :D

  • Reply • Share ›

**r...** • a year ago

we can Print all boundary elements in a single iteration O(n)

```
 void Boundary_traversal ( bst_node * node ,int lcnt ,int rcnt )
{
        if(node == NULL)
                return;

        if( !rcnt && !isLeaf(node))
        {
              printf("-> %d ",node->value);
        }

        Boundary_traversal(node->left ,flag,lcnt+1,rcnt);

        if(isLeaf(node))
        {
              printf("-> %d ",node->value );
        }
```

see more

▲ | ▼ • Reply • Share ›

**r...** ➦ r... • a year ago

fixing the recursive caller function arguments (shouldn't have flag as th

```
 void Boundary_traversal ( bst_node * node ,int lcnt ,int rcnt
{
    if(node == NULL)
        return;

    if( !rcnt && !isLeaf(node))
```

```
{
      printf("-> %d ",node->value);
}

Boundary_traversal(node->left ,lcnt+1,rcnt);

if(isLeaf(node))
{
      printf("-> %d ",node->value );
}
```

see more

Reply • Share ›

**Gupt** • a year ago

The above approach seems to be wrong. Consider a tree where the left subtre
right subtree has say 10 complete levels. The left boundary of right subtree als
tree. But it wont be printed in this aproach.

```
/* Paste your code here (You may delete these lines if not writing c
```

1 Reply • Share ›

**Shrey Trivedi** Gupt • 6 months ago

Then in that case we should do the following :
1. Print the left view of the tree (all nodes that you would see from the l
2. Print the leaves of the tree
3. Print the right view of the tree bottom to top

Reply • Share ›

**Anon** Gupt • 8 months ago

this is a valid point.It is a request to please confirm this point!.

**Manish** • a year ago

```
public void boundaryTraversal(){
boundaryTraversal(root);
}
private void boundaryTraversal(Node root){
leftBoundary(root);
leafNodes(root);
rightBoundary(root.right);

}
private void leftBoundary(Node root){
if(root==null)
return;
while(root.left!=null){
System.out.println(root.key);
root=root.left;
}
}
private void rightBoundary(Node root){
```

see more



**amit** • a year ago

I think if we add another node as left of 10 say with value 6, we should get both
wont give 10 in the output
please verify

```
 /* Paste your code here (You may delete these lines if not writing c
#include <stdio.h>
```

```
#include <stdlib.h>

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

// A simple function to print leaf nodes of a binary tree
void printLeaves(struct node* root)
```

see more

• Reply • Share ›

**Gopal** • a year ago

[sourcecode language="JAVA"]
public static void boundaryTraversal(Node root)
{
if(root == null)
return;

boundaryTraversal(root,true,true);
}

private static void boundaryTraversal(Node node,boolean left,boolean right)
{
if(node == null)
return;
//to avoid root to printed twice
boolean isRootPrinted = false;

```
{
System.out.print(node.n + " ");
return;
```

see more

• Reply • Share ›

**Karthik** • a year ago

Using Preorder traversal on left tree and postorder traversal on right

```
[sourcecode language="C++"]
void preLeft(Node* node, bool check = true)
{
if(!node)
return;

if(check || !(node->right || node->left))
cout << node->data << endl;

preLeft(node->left,check && true);
preLeft(node->right,false);
}

void postRight(Node* node, bool check = true)
{
if(!node)
return;
```

see more

• Reply • Share ›

**Sharad Chandra** ➦ Karthik • a year ago

I figured out the same and agree with your idea with some changes. It i

For left tree, pre-order traversal, use the information that backtrack has nodes.

For right tree, post-order traversal, pass the information to recursive fu is part of right boundary.

In both the case, always print leaf node, which can easily be checked.

```
void preorderPrintBoundary (struct node* root, int& backtrack)
{
if (root)
{
if ((! backtrack) || ((!root->left) && (!root->right)))
printf("%d ", root->data);

preorderPrintBoundary(root->left, backtrack);
```

see more

• Reply • Share ›

**Sharad Chandra** Sharad Chandra • a year ago

PrintBoundary requires a correction

```
if (root and root does not have both child node)
{
print root->data
printBoundary(root's child which ever is present)
}
else
{ /* root has both the child node, continue with preorder and pos
}
```

• Reply • Share ›

**Amit** • a year ago

Requirement :
1. Print the left boundary in top-down manner.
2. Print all leaf nodes from left to right, which can again be sub-divided into two
…..2.1 Print all leaf nodes of left sub-tree from left to right.
…..2.2 Print all leaf nodes of right subtree from left to right.
3. Print the right boundary in bottom-up manner.

Solution should be :
1. Root -> Check Not Leaf -> Print -> Recurse to left
2. Inorder Traversal of full tree print only leaves
3. Root -> Recurse to right -> Check Not Leaf -> Print

Am I right looks like simple functions

  • Reply • Share ›

**Dhaval** • a year ago

```
/*@geeksforgeeks
   For the following input...*/
   struct node *root         = newNode(20);
   root->right               = newNode(8);
   root->right->left         = newNode(4);
   root->right->right        = newNode(12);
   root->right->left->left   = newNode(10);
   root->right->right->right  = newNode(14);
   root->right->right->left   = newNode(11);
   root->right->left->right        = newNode(25);
/*    output:20 10 25 11 14 12 8
       20


           8
```

```
        4        12

     10    25  11   14

    4 is not printed...
    I think it should be print 4 as a Left-boundary..
    Correct me if I am wrong..

*/
```

• Reply • Share ›

**AKS** ➦ Dhaval • 9 months ago

@Dhaval, Yes, You are right.I too have same question, let me know if y

• Reply • Share ›

**Sariam** • 2 years ago

why do well call printLeaves() twice, we can just call it once by passing the roo leaves from left to right which we want.

Clarify if I am missing something!

```
/* Paste your code here (You may delete these lines if not writing co
```

• Reply • Share ›

**rahul sihag** • 2 years ago

O(n) in one traversal
Call
printBoundary(root,1,1);

```
void printBoundary(struct node *root,int flagr,int rootflag)
{
        if(!root)
                return;
        if(root->left==NULL&&root->right==NULL)
        {
                printf("%d ",root->data);
                flagl=0;
                return;
        }

        if(flagl)
```

see more

Reply • Share ›

**Vaibhav** rahul sihag • 7 months ago

Your code will not work in below example:
10
\
20
\
30

Reply • Share ›

**Guest** rahul sihag • 7 months ago

Your code will not work in below example:

Reply • Share ›

**Gupt** rahul sihag • a year ago

Could you please explain the code?

```
/* Paste your code here (You may delete these lines if not wri
```

  • Reply • Share ›

**firefist** rahul sihag • 2 years ago

fantastic...

  • Reply • Share ›

**praveen** firefist • 2 years ago

superb

```
/* Paste your code here (You may delete these lines if
```

  • Reply • Share ›

**Nishant Seth** • 2 years ago

What if nodes 10 & 14 have children as well? I think they should still be part of by any of the 3 boundary functions. Let me know if I am wrong.

```
/* Paste your code here (You may delete these lines if not writing c
```

  • Reply • Share ›

**Aashish** Nishant Seth • 2 years ago

If 10 and 14 will have children, they will become internal nodes, so they

  • Reply • Share ›

**Theo** Aashish • 2 years ago

i think 10 and 14 will be still part of the boundary if 10 has only a child(leaf). But your code seems doesn't handling it..the bounda and right most nodes of n-1 levels and all leaves of nth level for