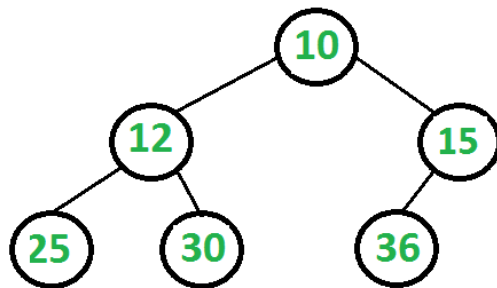
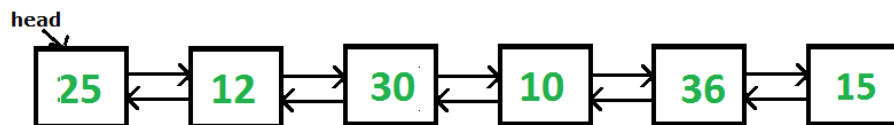


## Convert a given Binary Tree to Doubly Linked List | Set 3

Given a Binary Tree (BT), convert it to a Doubly Linked List(DLL) In-Place. The left and right pointers in nodes are to be used as previous and next pointers respectively in converted DLL. The order of nodes in DLL must be same as Inorder of the given Binary Tree. The first node of Inorder traversal (left most node in BT) must be head node of the DLL.



The above tree should be in-place converted to following Doubly Linked List(DLL).



Following two different solutions have been discussed for this problem.

[Convert a given Binary Tree to Doubly Linked List | Set 1](#)

[Convert a given Binary Tree to Doubly Linked List | Set 2](#)

In this post, a third solution is discussed which seems to be the simplest of all. The idea is to do inorder traversal of the binary tree. While doing inorder traversal, keep track of the previously

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](#)

[Geometric Algorithms](#)

visited node in a variable say *prev*. For every visited node, make it next of *prev* and previous of this node as *prev*.

Thanks to rahul, wishall and all other readers for their useful comments on the above two posts.

Following is C++ implementation of this solution.

```
// A C++ program for in-place conversion of Binary Tree to DLL
#include <iostream>
using namespace std;

/* A binary tree node has data, and left and right pointers */
struct node
{
    int data;
    node* left;
    node* right;
};

// A simple recursive function to convert a given Binary tree to Doubly
// Linked List
// root --> Root of Binary Tree
// head --> Pointer to head node of created doubly linked list
void BinaryTree2DoubleLinkedList(node *root, node **head)
{
    // Base case
    if (root == NULL) return;

    // Initialize previously visited node as NULL. This is
    // static so that the same value is accessible in all recursive
    // calls
    static node* prev = NULL;

    // Recursively convert left subtree
    BinaryTree2DoubleLinkedList(root->left, head);

    // Now convert this node
    if (prev == NULL)
        *head = root;
    else
    {
        root->left = prev;
        prev->right = root;
    }
    prev = root;
}
```



Integrated  
**Desktop & Mobile Device**  
Management

ManageEngine  
Desktop Central

Download 

## 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

```

// Finally convert right subtree
BinaryTree2DoubleLinkedList(root->right, head);
}

/* Helper function that allocates a new node with the
   given data and NULL left and right pointers. */
node* newNode(int data)
{
    node* new_node = new node;
    new_node->data = data;
    new_node->left = new_node->right = NULL;
    return (new_node);
}

/* Function to print nodes in a given doubly linked list */
void printList(node *node)
{
    while (node!=NULL)
    {
        cout << node->data << " ";
        node = node->right;
    }
}

/* Driver program to test above functions*/
int main()
{
    // Let us create the tree shown in above diagram
    node *root      = newNode(10);
    root->left       = newNode(12);
    root->right      = newNode(15);
    root->left->left  = newNode(25);
    root->left->right = newNode(30);
    root->right->left = newNode(36);

    // Convert to DLL
    node *head = NULL;
    BinaryTree2DoubleLinkedList(root, &head);

    // Print the converted list
    printList(head);

    return 0;
}

```

Output:

Shouldn't you  
expect a  
cloud with:

# SYSTEM MONITORING

Plus the experts  
to help run it?

TRY MANAGED CLOUD ►

 **rackspace**  
the open cloud company



Note that use of static variables like above is not a recommended practice (we have used static for simplicity). Imagine a situation where same function is called for two or more trees, the old value of *prev* would be used in next call for a different tree. To avoid such problems, we can use double pointer or reference to a pointer.

Time Complexity: The above program does a simple inorder traversal, so time complexity is  $O(n)$  where  $n$  is the number of nodes in given binary tree.

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](#)

## Recent Comments

[affizerv](#) 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

AdChoices 

[▶ Tree Diagram](#)

[▶ Binary Tree](#)

[▶ Java Tree](#)

- [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](#)



10



Tweet

3



2

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

14 Comments

GeeksforGeeks

Sort by Newest ▾



Join the discussion...



**Pranav Sawant** · 22 days ago

Don't we need to set the left of head to null??

^ | ▾ · Reply · Share ›



**AlienOnEarth** → Pranav Sawant · 3 days ago

It has been taken care as initially \*prev points to NULL

^ | ▾ · Reply · Share ›



**GOPI GOPINATH** · 25 days ago

why should we do inorder traversal alone for this ?? cant we do a level order tr

^ | ▾ · Reply · Share ›



**Jonathan** → GOPI GOPINATH · 25 days ago

With level-order traversal we would get the wrong DLL.

The instructions say we should end up with a list that is the same as a order. Thus, we do an inorder traversal.

^ | ▾ · Reply · Share ›

AdChoices ▸

► [Convert C++](#)

► [Linked List](#)

► [Convert DLL](#)

AdChoices ▸

► [Convert XML Data](#)

► [Convert Int](#)

► [Convert Java](#)



**Alien** · a month ago

Amazingly simple solution !!! Thank you geeksforgeeks

1 ^ | v · Reply · Share ›



**Abhishek Kumar** · a month ago

this is an awesome for conversion of BT to DLL..!!!!

1 ^ | v · Reply · Share ›



**Aj** · 2 months ago

The way I tried to do it was maintaining the tree structure while creating the linked list and my node looked is like

struct node

{

int data;

node \*right;

node \*left;

node \*next; // I'm not touching right and left pointer for sake of maintaining tree linked list.

};

So if you want to traverse the tree, pass the root node to inorder routine and if the head pointer to it.

Code

-----

#include <stdio.h>

#include <stdlib.h>

[see more](#)

^ | v · Reply · Share ›



**ravindra dingankar** · 2 months ago



If we are implementing this in java and cant use double pointer, then we have 1

1 ^ | v • Reply • Share ›



**Ravi** → ravindra dingankar • a month ago

In java, your root's reference is wrapped inside a class tree. When you reference to tree object. So you can change root.

^ | v • Reply • Share ›



**Siva Krishna** • 2 months ago

nice and simple

^ | v • Reply • Share ›



**Vinay Singh** • 2 months ago

this is the best way to do this problem...just simply awesome :)

^ | v • Reply • Share ›



**www.asktogeek.com** • 2 months ago

great tutorial, hard work definitely. thanks.

^ | v • Reply • Share ›



**wishall** • 2 months ago

gud wrk,,,,gfg,,:)

^ | v • Reply • Share ›



**fateh\_singh** • 2 months ago

awsme ...:)

^ | v • Reply • Share ›



Subscribe



Add Disqus to your site

