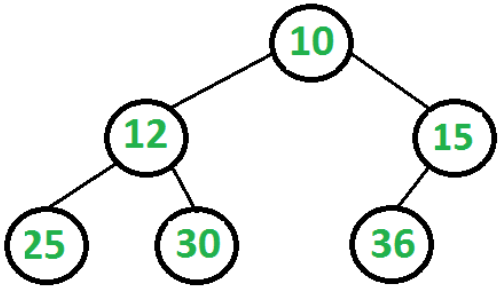
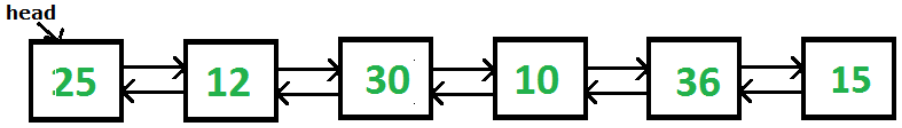
Convert a given Binary Tree to Doubly Linked List | Set 1

Given a Binary Tree (Bt), convert it to a Doubly Linked List(DLL). 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).



I came across this interview during one of my interviews. A similar problem is discussed in this post. The problem here is simpler as we don't need to create circular DLL, but a simple DLL. The idea behind its solution is quite simple and straight.

1. If left subtree exists, process the left subtree
.....**1.a)** Recursively convert the left subtree to DLL.

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

.....**1.b)** Then find inorder predecessor of root in left subtree (inorder predecessor is rightmost node in left subtree).
.....**1.c)** Make inorder predecessor as previous of root and root as next of inorder predecessor.
2. If right subtree exists, process the right subtree (Below 3 steps are similar to left subtree).
.....**2.a)** Recursively convert the right subtree to DLL.
.....**2.b)** Then find inorder successor of root in right subtree (inorder successor is leftmost node in right subtree).
.....**2.c)** Make inorder successor as next of root and root as previous of inorder successor.
3. Find the leftmost node and return it (the leftmost node is always head of converted DLL).

Below is the source code for above algorithm.

```
// A C++ program for in-place conversion of Binary Tree to DLL
#include <stdio.h>

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

/* This is the core function to convert Tree to list. This function fo
  steps 1 and 2 of the above algorithm */
node* bintree2listUtil(node* root)
{
    // Base case
    if (root == NULL)
        return root;

    // Convert the left subtree and link to root
    if (root->left != NULL)
    {
        // Convert the left subtree
        node* left = bintree2listUtil(root->left);

        // Find inorder predecessor. After this loop, left
        // will point to the inorder predecessor
        for (; left->right!=NULL; left=left->right);

        // Make root as next of the predecessor
        left->right = root;
```


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

```
        // Make predecssor as previous of root
        root->left = left;
    }

    // Convert the right subtree and link to root
    if (root->right!=NULL)
    {
        // Convert the right subtree
        node* right = bintree2listUtil(root->right);

        // Find inorder successor. After this loop, right
        // will point to the inorder successor
        for (; right->left!=NULL; right = right->left);

        // Make root as previous of successor
        right->left = root;

        // Make successor as next of root
        root->right = right;
    }

    return root;
}

// The main function that first calls bintree2listUtil(), then follows
//  of the above algorithm
node* bintree2list(node *root)
{
    // Base case
    if (root == NULL)
        return root;

    // Convert to DLL using bintree2listUtil()
    root = bintree2listUtil(root);

    // bintree2listUtil() returns root node of the converted
    // DLL.  We need pointer to the leftmost node which is
    // head of the constructed DLL, so move to the leftmost node
    while (root->left != NULL)
        root = root->left;

    return (root);
}

/* 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)
    {
        printf("%d ", 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 = bintree2list(root);

    // Print the converted list
    printList(head);

    return 0;
}
```

Output:

```
25 12 30 10 36 15
```

This article is compiled by **Ashish Mangla** and reviewed by GeeksforGeeks team. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Recent Comments

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

Backtracking | Set 7 (Sudoku) · 32 minutes ago

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

Flipkart Interview | Set 6 · 52 minutes ago

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

Software Engineering Lab, Samsung Interview | Set 2 · 52 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

You may also like to see Convert a given Binary Tree to Doubly Linked List | Set 2 for another simple and efficient solution.

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.

53 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

Kunal Arora • 17 days ago

I think this question can simply be done by doing Inorder traversal of the binary

Please go through the function which I implemented and Let me know if there required...

```
void binarytreetoDLL(struct tree *root,struct tree **head)
{
static struct tree *prev=NULL;
struct tree *p=root;
if(p==NULL)
return;
else
{
convert(p->left,head);

p->left=prev;
if((*head)==NULL)
(*head)=p;
else
{ prev->right=p; }
prev=p;

convert(p->right,head);
}
}
```

The head parameter is for the head of DLL and is passed NULL initially....

ganeshprabhu1994 • a month ago

I dont find any need for having a base case in bintree2listutil.

• Reply • Share ›

Ravi → ganeshprabhu1994 • a month ago

Try to tun it without base, it would crash.

• Reply • Share ›

ganeshprabhu1994 → Ravi • a month ago

There is no way that node parameter will get the value of null th

• Reply • Share ›

Aniket Thakur • 3 months ago

Java Code with output ---> http://opensourceforgeeks.blog...

• Reply • Share ›

wishall • 3 months ago

```
//easy version
//C
//in main
struct tnode** prev=NULL;
call BST2DLL(root,prev,&root);

//implementation
void convertBST2DLL(struct tnode* root,struct tnode** prev,struct tnode** hea

if(!root)return ;

convertBST2DLL(root->left,prev,head);//call for left subtree

root->left=(*prev);//set left child to inorder predecessor
```

if(*prev)

(*prev)->right=root;//set right child of inorder predecessor to root

else

*head=root;//change the root,,this is only executed for the leftmost leaf

*prev=root;//update prev to its inorder successor

convertBST2DLL(root->right,prev,head);//call for right subtree

}

• Reply • Share ›

GeeksforGeeks → wishall • 2 months ago

Thanks for suggesting this simple approach. We have published this a
http://www.geeksforgeeks.org/c...

• Reply • Share ›

wishall → GeeksforGeeks • 2 months ago

:)

• Reply • Share ›

wgpshashank • 5 months ago

If you carefully examine the DLL , you will see all nodes which are in left subtre
DLL and nodes of right subtree are on right side of root node in DLL , same ap

So in-order Traversal is the solution , as we traverse the tree in-order, we can
its predecessor. We also have to modify the predecessor's right pointer to poi
doubly linked list behavior.

Here is the pseudo code

```
// We Will Keep Three Variable
//root: Current tree node
//pre: this pointer should have the address of in-order predecessor of root
//head: this denoted head of final link list

treeToDoublyList(Node root,Node pre, Node head)
{
if (!root) return;
treeToDoublyList(root->left, prev, head);//call left subtree
```

see more

• Reply • Share ›

samsammy • 5 months ago

Java Implementation without extra space-

```
package abc;

public class BinaryTreeToDLLNew {

Node previous;
boolean first=false;
Node head;

public static void main(String[] args) {

BinaryTreeToDLLNew obj= new BinaryTreeToDLLNew();
obj.dummyStart();

}

public void dummyStart()
{
```

see more

• Reply • Share ›

Mohit Sehgal • 6 months ago

I dont think that above solution will work. You are not finding the inOrder succe
correctly.

1 • Reply • Share ›

Psycho • 7 months ago

We can also store the address of nodes in Inorder traversal in the order of ino
change left pointer to previous element
and right pointer to next element of array. Is it a correct solution?

2 • Reply • Share ›

Jasprit • 7 months ago

```
void convert(struct node *head, struct node **start, node **tail)
{
if (head == NULL)
return;
convert(head->left, start, tail);

head->left = *tail; // prev
if(*tail == NULL)
{
*start = head;
}
else
{
(*tail)->right = head; // next
}
```

```
*tail = head;
convert(head->right, start, tail);
}
struct node *BTToDLL(struct node *root)
{
struct node *start = NULL, *tail = NULL;
convert(root, &start, &tail);
return start;
}
```

• Reply • Share ›

CODED • 7 months ago

It is as simple as this

```
node* treeToBst(node* root)
{
    static node *pre = NULL;
    node *head = NULL;
    if(!root)
    {
        return NULL;
    }
    if(NULL == (head = treeToBst(root->left)))
    {
        head = root;
    }
    if(pre)
    {
        pre->right = root;
        root->left = pre;
    }
```

```
    pre = root;
    treeToBst(root->right);
    return head;
}
```

• Reply • Share ›

Subrahmanyan Sankaran • 8 months ago

```
// TreeToDLL.cpp : Defines the entry point for the console applicatio
#include "stdafx.h"
#include <iostream>
using namespace std;
struct Node
{
   int data;
   Node *left;
    Node *right;
};
class Tree
{
      static int l;
      static int flip;
      static int leafdepth;
      static Node * prev;
      static Node * previous;
      static int count;
```

see more

1 • Reply • Share ›

Saurabh Verma • 10 months ago

what about this:

struct node* binaryToDLL(struct node* root, struct node* DLL).
{

if(! root)return DLL;

DLL=binaryToDLL(root->right, DLL);.

if(DLL).

DLL->left=root;.

root->right=DLL;.

DLL=root;.

return binaryToDLL(root->left, DLL);.
}
1 • Reply • Share ›

smuralimohan • 10 months ago
All of the trouble can be circumvented by using the code at: http://cslibrary.sta

and then in the final step, convert the circular doubly-linked list to a non-circula
1 • Reply • Share ›

syashi • 10 months ago

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

/*so in this we convert BT to DLL first we set previous pointer.. all the left point
as in inorder
traversal!
it does smthing like inorder traversal n sets left pointer to previous node
A

/ \
B C
/ \ / \
D E F so inorder travesal is D B E A F C.. n D will be the head!!*/
#include<stdio.h>
#include<stdlib.h>

struct node
{
struct node *right;
struct node *left;

see more

• Reply • Share ›

Code Jack • 10 months ago
how does addToDLL() work ??...plz mention
• Reply • Share ›

Tarun Kumar • 11 months ago
traverse(Node* root)
{

if(root == NULL).

return;.

traverse(root->left);.

addTODLL(root);.

traverse(root->right);.
}

What is wrong with this code?

• Reply • Share ›

CODED ➦ Tarun Kumar • 7 months ago

You need to find a way to return the head of the DLL

• Reply • Share ›

vikas • 11 months ago

```
// i think this will work

struct node* bst_dll(struct node * root)
{

if (!root)
return NULL;

static struct node * head=NULL;
static struct node * tail=NULL;

bst_dll(root->left);

if(!head)
{
head=root;
tail=root;
}
else
{
tail->right=root;
tail=root;
}
bst_dll(root->right);
```

}

• Reply • Share ›

KK • 11 months ago

I simply can not understand why we can not do this with inorder.

```
 /* Paste your code here (You may delete these lines if not writing c
void treeToList(node* root,node*& prev, node*& head)
{
        if(root)
        {
                treeToList(root->left,prev,head);
                if(prev)
                        prev->right = root;
                else
                        head = root;
                root->left= prev;
                prev = root;
                treeToList(root->right,prev,head);
        }
}
```

In the calling function do prev->next = nullptr; (It is a not a circular double linke
Otherwise, add pointers between head and prev

5 • Reply • Share ›

Zohreh • 11 months ago

The original solution does not solve the problem in O(n). It is of order O(nlogn)
levels.
// There might be some errors in the code

```
void BTree::TreeToDLL(BTree* n, BTree* &head, BTree* &tail) {
        BTree* head1;
        BTree* head2;
        BTree* tail1;
        BTree* tail2;

        if (n == NULL) {

                return;
        }

        head1 = head;
        tail1 = tail;
        head2 = head;
```

see more

Reply • Share ›

Vikrant • 11 months ago

What is the use of converting a binary tree into doubly Linked List ?

Reply • Share ›

code_jazz • 11 months ago

```
/************** headers ***************/
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>


/************ data types *****************/
struct node
```

```
{
    int data;
    struct node *left, *right;
};
struct LL{
    int data;
    struct LL *prev,*next;
};
```

see more

• Reply • Share ›

Anuj Prajapati → code_jazz • 9 months ago

we are not allowed to create a new list,
we have to convert the given tree itself into doubly linked list..

• Reply • Share ›

Manik Sidana • 11 months ago

May be I missed out something...Will check once again... Thanks Sandeep !!!

• Reply • Share ›

Sandeep Jain • 11 months ago

It seems to be working for your input tree. Please see http://ideone.com/1b6w

• Reply • Share ›

Deepak Joshi • 11 months ago

kahan appy kar rha hai?

• Reply • Share ›

Manik Sidana • 11 months ago

[Query about the posted solution].

I think this will not work for the below tree.

30 is root.
30->left = 20.
30->right = 40.

20->left =10
20->right = 25.

10->left=NULL
10->right=15

25->right=28
25->left=NULL

40->right=NULL
40->left = 35.

see more

• Reply • Share ›

abhishek08aug • 11 months ago

Intelligent :D

• Reply • Share ›

mukesh2009mit • a year ago

```
 /*Converting a Binary Search Tree to Doubley Linked List */
/*TIME COMPLEXITY=O(N)
STACK SPACE=O(N)*/

#include<stdio.h>
```

```
#include<conio.h>
struct node
{
int x;
struct node *lc;
struct node *rc;
};
void mklist(struct node *p,struct node** lm,struct node** rm)
{
struct node* lm1,*lm2,*rm1,*rm2;
lm1=lm2=rm1=rm2=NULL;
if(p->lc==NULL&&p->rc==NULL)
{
```

see more

• Reply • Share ›

Code1101 • a year ago

```
class Tree {
    Node root;

    Tree(Node root) {
        this.root = root;
    }

    public void postOrder() {
        postOrder(root);
    }

    private void postOrder(Node node) {
        if(node == null) return;
```

```
        postOrder(node.rightChild);
        System.out.println(node.data);
        postOrder(node.leftChild);
    }
```

see more

• Reply • Share ›

Ravi Kesh Singh • a year ago

sorry u can return headDll by changing the function signature or passing the h

• Reply • Share ›

Ravi Kesh Singh • a year ago

void inorder(treeNode* root,treeNode* prev)/*initially prev will be NULL*/

{

treeNode* headDll = NULL;

if(root)

{

inorder(root->left,prev);

if(prev == NULL)

headDll = root;

else

{

prev->right=root;

see more

• Reply • Share ›

Ravikesh • a year ago

```
void inorder(treeNode* root,treeNode* prev)/*initially prev will be NULL*/
{
treeNode* headDll = NULL;
if(root)
{
inorder(root->left,prev);
if(prev == NULL)
headDll = root;
else
{
prev->right=root;
root->left = prev;
}
prev = root;
inorder(root->right,prev);
}
}
```

Point to be noted that headDll prev will be NULL automatically and similarly las

• Reply • Share ›

Ravi Kesh Singh • a year ago

Sorry i haven&#039t seen this, now i am giving sol for time complexity o(n) wit

1 • Reply • Share ›

naren596 • a year ago

reverse postorder traversal gives successor for next level recursion. I guess tl

public class ConvertBTtoDLL

```
{
node head=null;
public node convert_helper(node root)
{
convert(root);
return head;
}

public node convert(node root)
{
if(root!=null)
{
convert(root.right);
root.right=head;
if(head!=null) head.left=root;
head=root;
convert(root.left);
}
return root;
}
}
```

• Reply • Share ›

Praveen naren596 • a year ago

+1

• Reply • Share ›

Praveen • a year ago

```
package Btree;

import java.util.LinkedList;
import java.util.Queue;
```

```
public class BtreeFromDoublyLinked {

        public Bnode ConvertToDoubly(Bnode root){
                Queue<Bnode> q = new LinkedList<Bnode>();
                if(root==null)
                return null;
                q.add(root);
                Bnode start = root;
                Bnode lbnode=null;
                Bnode current=null;
                while(q.peek()!=null){
                        if(q.peek().left!=null){
                                q.add(q.peek().left);
```

see more

• Reply • Share ›

Sambasiva • a year ago

```
Node * Bst2Dbl(Node *t, Node **prev) {

        if (t == NULL) {
                return NULL;
        }

        Node *root = Bst2Dbl(t->left, prev);

        if (*prev) {
                (*prev)->right = t;
                t->left = *prev;
        }
```

```
        *prev = t;

        Bst2Dbl(t->right, prev);

        return root ? root : t;
}
```

• Reply • Share ›

Ramesh.Mxian • a year ago

We can do it with O(N) easily. We will process the Nodes in inorder traversal a

1. If left subtree exists then Create a DLL for the left subtree and return the hea
2. If right subtree exists then Create a DLL for the right subtree and return the
3. Insert the current node between tail of the DLL from left subtree and head of
4. Return new head as the head from left subtree and tail as tail from right sub

We can make use of the same Node structure to return head and tail of DLL b

Following is the procedure

Node CreateDLL(Node node){
Node Result, LDLL,RDLL.

// If there is no left subtree then current node will be head of the resulting DLL
// If there is no right subtree then current node will be tail of the resulting DLL
Result.right = node
Result.left = node;

if(node.left != null){

see more

• Reply • Share ›

mohitk ➦ Ramesh.Mxian • a year ago

True.
had the same in my design, but just returning the head...
However, can also use the principle as given above by Ramesh.

```
public Node<T> tree2DLL(Node<T> root) {
                if (root == null) return null;
                return tree2DLLHelper(root, root, root);
        }

        /** Params: All are initially equal to node.
         *  Returns: Head of the new DLL to which the binary tr
        private Node<T> tree2DLLHelper(Node<T> node, Node<T> he
                // Left-subtree
                if (node.left != null) {
                        Node<T> headL, tailL;
                        headL = tailL = node.left;

                        // Update head of the combined DLL
```

see more

• Reply • Share ›

???????? ??? • a year ago

ya sure it will work but the question is in-place conversion that is without creati
given tree itself

• Reply • Share ›

Ravi Kesh Singh • a year ago

The simple approach would be modify the inorder traversal instead pf printing
>data) to insert the new element in DLL.

We will keep one tail pointer which will contain the last node of DLL.so in inser
and do tail->next = new_node;.
new_node->prev = tail;tail = new_node;tail->next = NULL; simple and optimal
complexity.

• Reply • Share ›

sri • a year ago

```
void modify_to_DLL(node*p, node*&prev,node*&head)
{
if(!p)return ;
modify_to_DLL(p->left,prev,head);

p->left=prev;
if(prev)
prev->right=p;
else
head=p;

node* right=p->right;
head->left=p;
p->right=head;
prev=p;//updating the prev node

modify_to_DLL(right,prev,head);
}
node* prev=new node;
node* head=new node;
prev=head=NULL;

modify_to_DLL(root,prev,head)
```

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

• Reply • Share ›

surabhiremix • a year ago

Can we go the either way.i mean from doubly linked list to binary tree?

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

• Reply • Share ›

GeeksforGeeks ➦ surabhiremix • a year ago

Please see following post:

http://www.geeksforgeeks.org/in-place-conversion-of-sorted-dll-to-bala

• Reply • Share ›

Max • a year ago

Segfault in bintree2list when root is null.

• Reply • Share ›

Load more comments

Subscribe Add Disqus to your site