

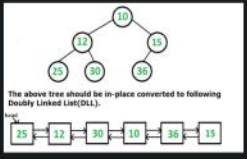
## Convert Binary Tree to DLL (asked in Meta) .~

**Binary Tree to DLL**

Difficulty: Hard Accuracy: 53.38% Submissions: 168K+ Points: 8 Average Time: 80m

Given a Binary Tree (BT), convert it to a Doubly Linked List (DLL) in place. The left and right pointers in nodes will be used as previous and next pointers respectively in converted DLL. The order of nodes in DLL must be the same as the order of the given Binary Tree. The first node of **inorder traversal** (leftmost node in BT) must be the head node of the DLL.

**Note:** h is the tree's height, and this space is used implicitly for the recursion stack.

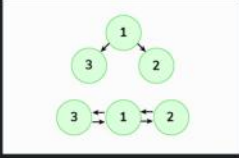


The above tree should be in-place converted to following Doubly Linked List (DLL).

**Examples:**

**Input:**  
 1  
 / \  
 3 2

**Output:**  
 3 1 2  
 2 1 3



**Explanation:** DLL would be 3<->1<->2

**Input:**  
 10  
 / \

### Approach :-

1. If the left subtree exists, process the left subtree
  1. Recursively convert the left subtree to DLL.
  2. Then find the inorder predecessor of the root in the left subtree (it is the rightmost node in the left subtree)
  3. Make the inorder predecessor as the previous root & the root as the next inorder predecessor.
2. If the right subtree exists, process it
  1. Recursively convert the right subtree to DLL.
  2. Then find the inorder successor of the root in the right subtree (it's the leftmost node in the right subtree)
  3. Make the inorder successor as the next root & the root as the previous inorder successor.
3. Find the leftmost node & return it (it is always the head of a converted

DLCL

```
/* Structure for tree and linked list
struct Node
{
    int data;
    struct Node* left;
    struct Node* right;

    Node(int x){
        data = x;
        left = right = NULL;
    }
};

// This function should return head to the DLL
class Solution {
public:
    Node* bToDLL(Node* root) {
        // code here
        if(!root) return root;
        root = bintree2listUtil(root);
        while(root->left != NULL) root = root->left;
        return root;
    }

    Node* bintree2listUtil(Node* root){
        if(!root) return root;
        if(!root->left) return root;
        Node* left = bintree2listUtil(root->left);
        root->left->right = root;
        root->left = left;
        left->right = root;
        root->left = left;
        if(!root->right) return root;
        Node* right = bintree2listUtil(root->right);
        root->right->left = root;
        root->right = right;
        right->left = root;
        root->right = right;
        return root;
    }
};
```

Another approach :-

Algorithm :-

1. Traverse the tree in inorder fashion.
2. While visiting each node, keep track of DLL's head & tail pointers, insert each visited node to the end of DLL using tail pointer.
3. Return the head of the list

```
1. // Driver Code Ends
2.
3. /* Structure for tree and linked list
4.
5. struct Node
6. {
7.     int data;
8.     struct Node* left;
9.     struct Node* right;
10.
11.     Node(int x){
12.         data = x;
13.         left = right = NULL;
14.     }
15. };
16.
17. // This function should return head to the DLL
18.
19. class Solution {
20. public:
21.     Node* prev=NULL;
22.     Node* bToDLL(Node* root) {
23.         // code here
24.         Node* head=NULL;
25.         bToDLLUtil(root, head);
26.         return head;
27.     }
28.
29.     void bToDLLUtil(Node* root, Node* head){
30.         if(!root) return;
31.         bToDLLUtil(root->left, head);
32.         if(!prev){
33.             head=root;
34.         }
35.         else{
36.             root->left=prev;
37.             prev->right=root;
38.         }
39.         prev=root;
40.         bToDLLUtil(root->right, head);
41.     }
42. };
43.
44. // Driver Code Ends
45.
```