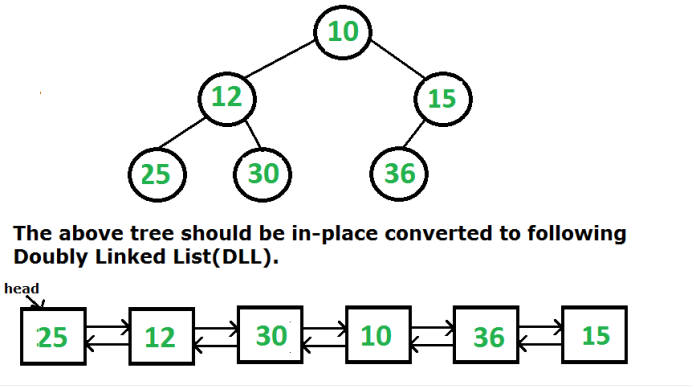
Convert a given Binary Tree to Doubly Linked List | Set 4

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.



Below three 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

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

In the following implementation, we traverse the tree in inorder fashion. We add nodes at the beginning of current linked list and update head of the list using pointer to head pointer. Since we insert at the beginning, we need to process leaves in reverse order. For reverse order, we first traverse the right subtree before the left subtree. i.e. do a reverse inorder traversal.

C++

```
// C++ program to convert a given Binary
// Tree to Doubly Linked List
#include <stdio.h>
#include <stdlib.h>

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

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

    // Recursively convert right subtree
    BToDLL(root->right, head_ref);

    // insert root into DLL
    root->right = *head_ref;

    // Change left pointer of previous head
```

```
    if (*head_ref != NULL)
        (*head_ref)->left = root;

    // Change head of Doubly linked list
    *head_ref = root;

    // Recursively convert left subtree
    BToDLL(root->left, head_ref);
}

// Utility function for allocating node for Binary
// Tree.
Node* newNode(int data)
{
    Node* node = new Node;
    node->data = data;
    node->left = node->right = NULL;
    return node;
}

// Utility function for printing double linked list.
void printList(Node* head)
{
    printf("Extracted Double Linked list is:\n");
    while (head)
    {
        printf("%d ", head->data);
        head = head->right;
    }
}

// Driver program to test above function
int main()
{
    /* Constructing below tree
                5
              /   \
             3     6
            / \     \
           1   4     8
          / \       / \
         0   2     7   9  */
    Node* root = newNode(5);
    root->left = newNode(3);
    root->right = newNode(6);
    root->left->left = newNode(1);
    root->left->right = newNode(4);
    root->right->right = newNode(8);
    root->left->left->left = newNode(0);
    root->left->left->right = newNode(2);
    root->right->right->left = newNode(7);
    root->right->right->right = newNode(9);

    Node* head = NULL;
    BToDLL(root, &head);

    printList(head);

    return 0;
}
```

Java

```

// Java program to convert a given Binary Tree to
// Doubly Linked List

/* Structure for tree and Linked List */
class Node
{
    int data;
```

```
    int data;
    Node left, right;

    public Node(int data)
    {
        this.data = data;
        left = right = null;
    }
}

class BinaryTree
{
    // 'root' - root of binary tree
    Node root;

    // 'head' - reference to head node of created
    //double linked list
    Node head;

    // A simple recursive function to convert a given
    // Binary tree to Doubly Linked List
    void BToDLL(Node root)
    {
        // Base cases
        if (root == null)
            return;

        // Recursively convert right subtree
        BToDLL(root.right);

        // insert root into DLL
        root.right = head;

        // Change left pointer of previous head
        if (head != null)
            (head).left = root;

        // Change head of Doubly linked list
        head = root;

        // Recursively convert left subtree
        BToDLL(root.left);
    }

    // Utility function for printing double linked list.
    void printList(Node head)
    {
        System.out.println("Extracted Double Linked List is : ");
        while (head != null)
        {
            System.out.print(head.data + " ");
            head = head.right;
        }
    }

    // Driver program to test the above functions
    public static void main(String[] args)
    {
        /* Constructing below tree
               5
             /   \
            3     6
           / \     \
          1   4     8
         / \       / \
        0   2     7   9  */

        BinaryTree tree = new BinaryTree();
        tree.root = new Node(5);
        tree.root.left = new Node(3);
        tree.root.right = new Node(6);
        tree.root.left.right = new Node(4);
        tree.root.left.left = new Node(1);
```

```
        tree.root.right.right = new Node(8);
        tree.root.left.left.right = new Node(2);
        tree.root.left.left.left = new Node(0);
        tree.root.right.right.left = new Node(7);
        tree.root.right.right.right = new Node(9);

        tree.BToDLL(tree.root);
        tree.printList(tree.head);
    }
}

// This code has been contributed by Mayank Jaiswal(mayank_24)
```

Output :

```
Extracted Double Linked list is:
0 1 2 3 4 5 6 7 8 9
```

Time Complexity: O(n), as the solution does a single traversal of given Binary Tree.