Double Tree

Write a program that converts a given tree to its Double tree. To create Double tree of the given tree, create a new duplicate for each node, and insert the duplicate as the left child of the original node.

So the tree…

```
    2
   / \
  1   3
```

is changed to…

```
       2
      / \
     2   3
    /   /
   1   3
  /
 1
```

And the tree

```
            1
          /   \
        2       3
      /  \
    4     5
```


is changed to

```
            1
          /   \
         1     3
        /     /
       2     3
      / \
     2   5
    /   /
   4   5
  /
 4
```

Algorithm:
Recursively convert the tree to double tree in postorder fashion. For each node, first convert the left subtree of the node, then right subtree, finally create a duplicate node of the node and fix the left child of the node and left child of left child.

Implementation:

```
#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;
    struct node* right;
};

/* function to create a new node of tree and returns pointer */
struct node* newNode(int data);

/* Function to convert a tree to double tree */
void doubleTree(struct node* node)
{
  struct node* oldLeft;
```


Popular Posts

```
  if (node==NULL) return;

  /* do the subtrees */
  doubleTree(node->left);
  doubleTree(node->right);

  /* duplicate this node to its left */
  oldLeft = node->left;
  node->left = newNode(node->data);
  node->left->left = oldLeft;
}



/* UTILITY FUNCTIONS TO TEST doubleTree() FUNCTION */
 /* Helper function that allocates a new node with the
    given data and NULL left and right pointers. */
struct node* newNode(int data)
{
  struct node* node = (struct node*)
                      malloc(sizeof(struct node));
  node->data = data;
  node->left = NULL;
  node->right = NULL;

  return(node);
}

/* Given a binary tree, print its nodes in inorder*/
void printInorder(struct node* node)
{
  if (node == NULL)
    return;
  printInorder(node->left);
  printf("%d ", node->data);
  printInorder(node->right);
}


/* Driver program to test above functions*/
int main()
{

  /* Constructed binary tree is
            1
          /   \
        2      3
```

```
        /   \
      4      5
    */
    struct node *root = newNode(1);
    root->left        = newNode(2);
    root->right       = newNode(3);
    root->left->left  = newNode(4);
    root->left->right = newNode(5);

    printf("Inorder traversal of the original tree is \n");
    printInorder(root);

    doubleTree(root);

    printf("\n Inorder traversal of the double tree is \n");
    printInorder(root);

    getchar();
    return 0;
}
```

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

References:
http://cslibrary.stanford.edu/110/BinaryTrees.html

Please write comments if you find any bug in above code/algorithm, or find other ways to solve the same problem.

Recent Comments

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

Backtracking | Set 7 (Sudoku) · 43 minutes ago

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

Flipkart Interview | Set 6 · 1 hour ago

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

Software Engineering Lab, Samsung Interview | Set 2 · 1 hour 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

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.

17 Comments GeeksforGeeks

Sort by Newest

append at the end · 3 hours ago

Neha I think that is what it should return as, in...

Find depth of the deepest odd level leaf node · 3 hours ago

Mukunthan • 8 months ago

Correct me if I am wrong.. Here
Postorder works fine if we can add a duplicate node even to left or right
Inorder works if a duplicate is added to left
Preorder never works

This is because in "Preorder" and "Right duplicate of Inorder" we create a nod
which will result in an infinite loop..

 • Reply • Share ›

code_jazz ➚ Mukunthan • 3 months ago

well u can skip that duplicate node by recursively making calls kike roo
preorder fashion

 • Reply • Share ›

pranjalgupta • 10 months ago

We can also create a double tree by preorder traversal technique.

```
void preorder(tree* root)
{
if(root==NULL)
return;
tree* temp=root->left;
root->left=newnode(root->data);
root->left->left=temp;
free(temp);
preorder(root->left->left);
preorder(root->right);

}
```

Also, this is possible by inorder traversal if we modify the tree's node to have a

Nitesh • 10 months ago

```
 /*Double Tree*/
node* DoubleTree(node *root)
{
      if(root == NULL)
        return NULL;

      node *lNode = root->left;
      node *rNode = root->right;

      root->left = newNode(root->data);
      root->left->left = lNode;
      if(lNode != NULL)
        DoubleTree(lNode);
      if(rNode != NULL)
        DoubleTree(rNode);

      return root;
}
```


abhishek08aug • a year ago

C++ code:

```
 #include <iostream>
#include <stdlib.h>
using namespace std;
```

```
class tree_node {
  private:
    int data;
    tree_node * left;
    tree_node * right;
  public:
    tree_node() {
      left=NULL;
      right=NULL;
    }
    void set_data(int data) {
      this->data=data;
```

see more

Soumya Sengupta • a year ago

A top-down approch.....

```
void doubleTree(struct node* node)
{
struct node* oldLeft;

if (node==NULL) return;

/* do the subtrees */
doubleTree(node->left);
doubleTree(node->right);

/* duplicate this node to its left */
oldLeft = node->left;
node->left = newNode(node->data);
```

```
node->left->left = oldLeft;
}

int main()
```

see more

Reply • Share ›

anantkaushik89 • a year ago

Cant we use preorder here also? I think it should give the same result.

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

Reply • Share ›

Nishant Mishra • 2 years ago

Shouldn't we copy the data field of node to new duplicate node...

Reply • Share ›

Nishant Mishra ➦ Nishant Mishra • 2 years ago

Sorry, about this comment, I didn't see its been passed in Newnode()..

Reply • Share ›

Spock • 2 years ago

The above code is the code for BST. We can do this thing in case of BST by a

I don't know there is some problem with the site so sometimes the code just w

Reply • Share ›

GeeksforGeeks ➦ Spock • 2 years ago

@Spock: Apologies for the trouble. If your comment doesn't appear im
moderation by spam checker tool. Our admins manually approve such

Spock • 2 years ago

Well in case of BST we can simply do this thing with the help of inorder traver
same, please tell if it has some shortcomings.

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

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

struct node *newnode(long int dat) {
       struct node *newone = malloc(sizeof(struct node));
       newone->data = dat;
       newone->left = NULL;
       newone->right = NULL;
       return newone;
```

see more

Spock • 2 years ago

Well in case of BST we can do this just by the inorder traversal of the tree.
Here is the code which uses the inorder traversal of the BST.
Please point out if it has any shortcomings.

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

```
#include<stdlib.h>

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

struct node *newnode(long int dat) {
       struct node *newone = malloc(sizeof(struct node));
       newone->data = dat;
       newone->left = NULL;
```

see more

• Reply • Share ›

k53 • 2 years ago

```
nptr ins_left(nptr root)
{
        if(root==NULL)// boundary check for empty tree
                return NULL;
        if(root->left != NULL) // Left child present
        {
                nptr save;
                save=root->left; // save left child
                root->left=makeNode(root->data); // dup and insert le
                root->left->left=ins_left(save); // recurse with old
        }
        else //no left child
                root->left=makeNode(root->data);// dup and ins left
        if(root->right != NULL)// right child present
```

```
                root->right=ins_left(root->right); // recursively app
      //else -no right child - do nothing
        return root;
}
```


Sunil • 3 years ago

```
void convertToDouble(struct node* root)
{
struct node* temp,*new;
if(root!=NULL)
{
convertToDouble(root->llink);
temp=root->llink;
new=(struct node*)malloc(sizeof(struct node));
new->info=root->info;
new->llink=temp;
new->rlink=NULL;
root->llink=new;
convertToDouble(root->rlink);
}
}
```


neeraj singh • 3 years ago

```
public static void duplicateTheTree(Node n) {
        if (n==null) {
            return;
        }
```

```
        Node dup = new Node(n.value);
        duplicateTheTree(n.left);
        duplicateTheTree(n.right);
        dup.left = n.left;
        n.left = dup;
    }
```

Reply • Share ›

Sangeeta neeraj singh • 2 years ago

nice :)

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

1 Reply • Share ›

Subscribe Add Disqus to your site