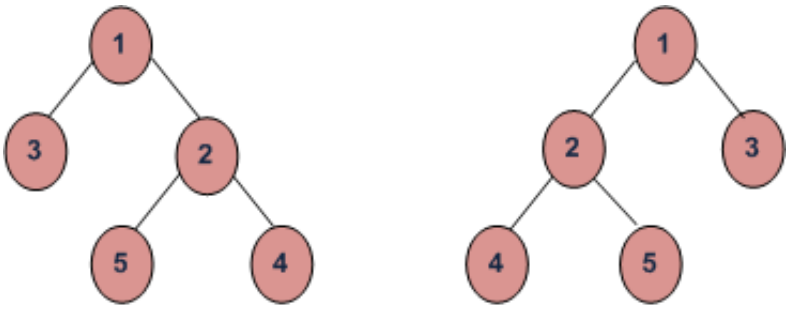
Write an Efficient Function to Convert a Binary Tree into its Mirror Tree

Mirror of a Tree: Mirror of a Binary Tree T is another Binary Tree M(T) with left and right children of all non-leaf nodes interchanged.



Mirror Trees

Trees in the below figure are mirror of each other

Algorithm – Mirror(tree):

```
(1)  Call Mirror for left-subtree    i.e., Mirror(left-subtree)
(2)  Call Mirror for right-subtree  i.e., Mirror(right-subtree)
(3)  Swap left and right subtrees.
          temp = left-subtree
          left-subtree = right-subtree
          right-subtree = temp
```

Program:

C

```
#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;
};

/* 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);
}


/* Change a tree so that the roles of the  left and
    right pointers are swapped at every node.

 So the tree...
       4
      / \
     2   5
    / \
   1   3

 is changed to...
       4
      / \
     5   2
        / \
       3   1
*/
void mirror(struct node* node)
{
  if (node==NULL)
    return;
  else
  {
    struct node* temp;

    /* do the subtrees */
```

```
    mirror(node->left);
    mirror(node->right);

    /* swap the pointers in this node */
    temp        = node->left;
    node->left  = node->right;
    node->right = temp;
  }
}


/* Helper function to test mirror(). Given a binary
   search tree, print out its data elements in
   increasing sorted order.*/
void inOrder(struct node* node)
{
  if (node == NULL)
    return;

  inOrder(node->left);
  printf("%d ", node->data);

  inOrder(node->right);
}


/* Driver program to test mirror() */
int main()
{
  struct node *root = newNode(1);
  root->left        = newNode(2);
  root->right       = newNode(3);
  root->left->left  = newNode(4);
  root->left->right = newNode(5);

  /* Print inorder traversal of the input tree */
  printf("\n Inorder traversal of the constructed tree is \n");
  inOrder(root);

  /* Convert tree to its mirror */
  mirror(root);

  /* Print inorder traversal of the mirror tree */
  printf("\n Inorder traversal of the mirror tree is \n");
  inOrder(root);

  getchar();
  return 0;
}
```

Java

```
// Java program to convert binary tree into its mirror
/* Class containing left and right child of current
 node and key value*/
class Node {

    int data;
    Node left, right;

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

class BinaryTree {

    Node root;

    void mirror() {
        mirror(root);
    }

    void mirror(Node node) {
        if (node == null) {
            return;
        } else {
            Node temp;

            /* do the subtrees */
            mirror(node.left);
            mirror(node.right);

            /* swap the objects/values in this node */
            temp = node.left;
            node.left = node.right;
            node.right = temp;
        }
    }

    void inOrder() {
        inOrder(root);
    }

    /* Helper function to test mirror(). Given a binary
     search tree, print out its data elements in
     increasing sorted order.*/
    void inOrder(Node node) {
```

```
        if (node == null) {
            return;
        }

        inOrder(node.left);
        System.out.print(node.data);

        inOrder(node.right);
    }
    /* testing for example nodes */

    public static void main(String args[]) {
        /* creating a binary tree and entering the nodes */
        BinaryTree tree = new BinaryTree();
        tree.root = new Node(1);
        tree.root.left = new Node(2);
        tree.root.right = new Node(3);
        tree.root.left.left = new Node(4);
        tree.root.left.right = new Node(5);

        /* print inorder traversal of the input tree */
        System.out.println("Inorder traversal of input tree is :");
        tree.inOrder();
        System.out.println("");

        /* convert tree to its mirror */
        tree.mirror();

        /* print inorder traversal of the minor tree */
        System.out.println("Inorder traversal of binary tree is : ");
        tree.inOrder();

    }
}
```

Time & Space Complexities: This program is similar to traversal of tree space and time complexities will be same as Tree traversal (Please see our Tree Traversal post for details)

127 Comments Category: Trees Tags: Convert to Mirror, Get the Mirror, Mirror Tree, Tree Traveral, Trees

Related Posts:

- Check if removing an edge can divide a Binary Tree in two halves
- Check sum of Covered and Uncovered nodes of Binary Tree
- Lowest Common Ancestor in a Binary Tree | Set 2 (Using Parent Pointer)
- Construct a Binary Search Tree from given postorder
- BFS vs DFS for Binary Tree
- Maximum difference between node and its ancestor in Binary Tree
- Inorder Non-threaded Binary Tree Traversal without Recursion or Stack
- Check if leaf traversal of two Binary Trees is same?

(Login to Rate and Mark)

1.7 Average Difficulty : **1.7/5.0**
Based on **32** vote(s)

Add to TODO List
Mark as DONE

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.

gaurilab • a month ago

```
struct node* mirror(struct node* node)
{
if (node==NULL) return node;

struct node* l = mirror(node->left);
struct node* r = mirror(node->right);

node->left = r;
node->right = l;
return node;

}
```

• Reply • Share ›

Deepak • a month ago

your solution will process leaf nodes also . we do not need to process leaf nodes.

```
void Binary_to_mirror(node *root)
{
if (root == NULL)
return;

if (root->left == NULL && root->right == NULL) // no need to process leaf nodes
return;

Binary_to_mirror(root->left);
Binary_to_mirror(root->right);

node *temp;
temp = root->left;
root->left = root->right;
root->right = temp;
}
```

• Reply • Share ›

Vardaan Sangar • 2 months ago

C version of the code

```
_____________________
-------------------------

struct node* MirrorBinaryTree(struct node* root)

{
```

```
if(root==NULL)

return 0;

root->left=MirrorBinaryTree(root->left);

root->right=MirrorBinaryTree(root->right);

if(root->left==NULL && root->right==NULL)

return root;
```

see more

• Reply • Share ›

Jatin • 4 months ago

www.coder2design.com

Java version

==========

```
private void mirror(Node<integer> node) {
if (node == null) {
return;
}
Node<integer> lNode = node.getLeftChild();
Node<integer> rNode = node.getRightChild();
mirror(lNode);
mirror(rNode);
node.setLeftChild(rNode);
node.setRightChild(lNode);
}
```

• Reply • Share ›

rogermreich • 4 months ago

Does it mind if we do in preorder or postorder?

• Reply • Share ›

algol ➚ rogermreich • 4 months ago

doesn't matter

1 • Reply • Share ›

rogermreich ➚ algol • 4 months ago

rogerfreich ▸ algol • 4 months ago

thank you,I got it

▲ ▼ • Reply • Share ›

SlickHackz • 5 months ago

I see lot of comments suggesting Mirror Operation via Pre-Order Traversal.
How it is possible?
I don't think Pre-order Traversal works for Mirroring the tree.
Can someone please correct me?

▲ ▼ • Reply • Share ›

Pranav Kumar Jha ➦ SlickHackz • 3 months ago

It does work for both pre order and post order.
Why do you think it might not?

▲ ▼ • Reply • Share ›

Manglam Singh • 5 months ago

(1) Swap left and right subtrees.
temp = left-subtree
left-subtree = right-subtree
right-subtree = temp
(2)Call Mirror for left-subtree i.e., Mirror(left-subtree)
(3) Call Mirror for right-subtree i.e., Mirror(right-subtree)
will it work ?

▲ ▼ • Reply • Share ›

gotham_ka_raja ➦ Manglam Singh • 3 months ago

the given algo does it in bottom up manner yours does it in top down manner .. answer will be same .. complexity O(n) in both cases

▲ ▼ • Reply • Share ›

algol ➦ Manglam Singh • 4 months ago

yes . its more of a level wise swap . you still would have to traverse all levels . O(logn) which is better than O(n)

▲ ▼ • Reply • Share ›

Vivek Mathur ➦ algol • 4 months ago

time complexity will be same i.e. O(n) in both the cases as u have to traverse every node.

▲ ▼ • Reply • Share ›

sid ➦ Manglam Singh • 5 months ago

Yes, It'll.

1 ▲ ▼ • Reply • Share ›

Deepak Singhal • 5 months ago

```
public void Mirror(Node t1, Node t2){
if(t1==null)
return;
else{
Node n=new Node(t1.data);
t2=n;
Mirror(t1.getLeft(), t2.getRight());
Mirror(t1.getRight(), t1.getLeft());
}
}
```


AlienOnEarth • 6 months ago

Preorder:

```
void mirror(struct node* node)

{

if (node==NULL)

return;

// swap
struct node* temp = node->left;

node->left = node->right;

node->right = temp;

/* do the subtrees */

mirror(node->left);

mirror(node->right);

}
```


prokilogram • 6 months ago

No need to have a temporary variable :

```
node->right = mirror(node->left);
node->left = mirror(node->right);
```

yoyo ➚ prokilogram • 5 months ago
You can't do this because mirror doesn't return any value.
• Reply • Share ›

Ishani Karmakar • 6 months ago
if we modify the condition to (node->left==NULL||node->right==NULL)
it decreases the number of recursions by 1. That should make it more efficient.
1 • Reply • Share ›

aman dhapola • 8 months ago
cout<<root->left->left->data<<endl; gives="" segmentation="" fault.="" why?="">
• Reply • Share ›

Pranav Kumar Jha ➚ aman dhapola • 3 months ago
Hmmm.
Try this tree, would you?
1
/ \
2 3

or this one
1
\
2
or this
1

Got the point?
You try to access data of node which doesn't even exist in the tree, you get SEGMENTATION FAULT.
• Reply • Share ›

ayush1gupta ➚ aman dhapola • 8 months ago
Your code will access data of NULL.
• Reply • Share ›

Sarthak Garg • 8 months ago
Geeks, in the algorithm above, please update it to Mirror(right-subtree)

(2) Call Mirror for right-subtree i.e., Mirror(left-subtree)
• Reply • Share ›

GeeksforGeeks Mod ➚ Sarthak Garg • 6 months ago

Thanks for pointing this out. We have corrected the typo.
• Reply • Share ›

deepak • 8 months ago
http://code.geeksforgeeks.org/
• Reply • Share ›

deepak deepak • 8 months ago
why isn't working?
• Reply • Share ›

Hari Prasath • 9 months ago
There is a mistake in algorithm
(2) should be(mirror->right)
• Reply • Share ›

GeeksforGeeks Mod Hari Prasath • 6 months ago
Thanks for pointing this out. We have corrected the typo
• Reply • Share ›

Shantanu • 9 months ago
you have done it using postorder fashion
we can also use preorder
• Reply • Share ›

V_CODER Shantanu • 9 months ago
i think he have done using POSTORDER(Left->Right->swap).
1 • Reply • Share ›

Shantanu V_CODER • 9 months ago
(y)
• Reply • Share ›

Dipankar Bhardwaj • 9 months ago
http://code.geeksforgeeks.org/...
• Reply • Share ›

Mohammed Raqeeb • 9 months ago
In method mirror(), we do not need 'else' clause as we are returning from the function when node is NULL.
• Reply • Share ›

Pankaj Kushwaha • 10 months ago

there is no need to go till down of tree in order to swap , we can first swap then can traverse through tree, consider blow example ,

void mirror(struct node* node)

{

if (node==NULL)

return;

else

{

struct node* temp;

temp = node->left;

node->left = node->right;

see more

8 • Reply • Share ›

Pankaj Kushwaha • 10 months ago

@admin:
Tree which is constructed in main function , is not the one given in comment in programm, please change it , its confusing...

• Reply • Share ›

Mihaela mitkova • 10 months ago

If i want to create a new tree with different numbers and the mirror it how it would look like ?

• Reply • Share ›

Raj Kumar Chauhan ➚ Mihaela mitkova • 8 months ago

it depends upon your tree structure.!

• Reply • Share ›

Siya ➚ Mihaela mitkova • 10 months ago

what do u mean by different numbers? not able to understand your question.

• Reply • Share ›

Ashish kulkarni • a year ago

Hi GeekforGeeks,

Please correct the algorithm since it has two lines as

(1) Call Mirror for left-subtree i.e., Mirror(left-subtree)
(2) Call Mirror for right-subtree i.e., Mirror(left-subtree)

Both are left-subtrees

• Reply • Share ›

Sunil Sharma • a year ago

No need to swap the pointers when left and right child are NULL.

simple function we can write

```
void Mirror(node * root)
{
if(root==NULL||(root->left==NULL&&root->right==NULL)
return;

Mirror( root->left );
Mirror( root->right );

node * temp = root->left;
root->left = root>right;
root->right=temp;
}
```

3 • Reply • Share ›

jas • a year ago

we an also swap first and call mirror later. That will also work.

```
void mirror( node * root)
{
if(root == null)
return;
node * ptr = root->left;
root->left = root->right;
root->right = ptr;
mirror(root->left);
mirror(root->right)
}
```

1 • Reply • Share ›

ramesh kanth ➦ jas • 10 months ago

how we can swap with out using temporary node?...

• Reply • Share ›

Mohammed Raqeeb ramesh kanth • 9 months ago
I think he is using a temp node i.e, ptr. So above should work.
• Reply • Share ›

Joey jas • a year ago
Yeah even I thought so.No issue with this it seems.
• Reply • Share ›

natasha • a year ago

```
void TreeFuncLib::MirrorTree(struct node** node)

{

if(*node == NULL)

return;

MirrorTree(&((*node)->left));

MirrorTree(&((*node)->right));

if(node != NULL && ((*node)->left != NULL || (*node)->right != NULL))

{

struct node* temp = (*node)->left;

(*node)->left = (*node)->right;

(*node)->right = temp;

}

}
```

• Reply • Share ›

jas natasha • a year ago
It doesnt cover the condition if there is only one child of a node.
• Reply • Share ›

AllergicToBitches • a year ago
pls correct the typo-
Call Mirror for right-subtree i.e., Mirror(left-subtree)

It should be right->subtree
• Reply • Share ›

Ashish Jaiswal • a year ago

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

typedef struct node
{
int data;
struct node*left;
struct node*right;
}Node;

Node*createnode(int d)
{
Node*newnode=(Node*)malloc(sizeof(Node));
newnode->data=d;
newnode->left=newnode->right=NULL;
return newnode;
}

void mirror(Node*node)
```

see more

• Reply • Share ›

Gohired.in • a year ago

video explanation code is discussed : https://www.youtube.com/watch?...

• Reply • Share ›

Gohired.in • a year ago

video explanation Mirror of Tree's both method and code is discussed : https://www.youtube.com/watch?...

• Reply • Share ›

Load more comments