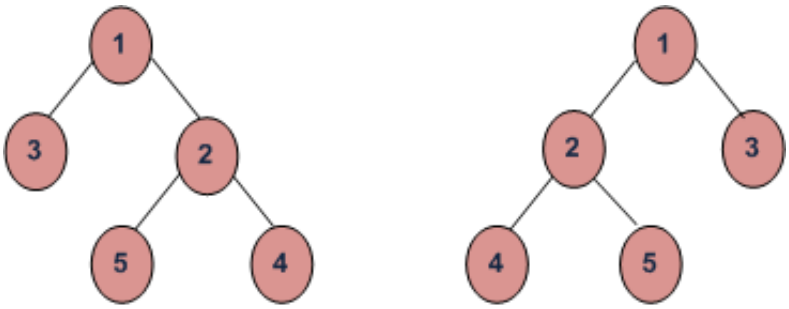
Write an Efficient C Function to Convert a Binary Tree into its Mirror Tree

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:

```
#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;
}
```

Run on IDE

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)

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

Related Posts:

- Check if a given array can represent Preorder Traversal of Binary Search Tree
- Mirror of n-ary Tree
- Succinct Encoding of Binary Tree
- Construct Binary Tree from given Parent Array representation
- Symmetric Tree (Mirror Image of itself)
- Find Minimum Depth of a Binary Tree
- Maximum Path Sum in a Binary Tree
- Expression Tree

0 Average Difficulty : **0/5.0**
No votes yet.

(Login to Rate)

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

115 Comments GeeksforGeeks 1 Login

Recommend Share Sort by Newest

Join the discussion…

SlickHackz • 19 days 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 ›

Manglam Singh • a month 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 ›

sid ➚ Manglam Singh • 17 days ago
Yes, It'll.
1 • Reply • Share ›

Deepak Singhal • a month 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());
}
}
```

• Reply • Share ›

AlienOnEarth • a month 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);

}
```

1 • Reply • Share ›

prokilogram • a month ago

No need to have a temporary variable :

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

• Reply • Share ›

yoyo prokilogram • 9 days ago

You can't do this because mirror doesn't return any value.

• Reply • Share ›

Ishani Karmakar • 2 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.

• Reply • Share ›

aman dhapola • 4 months ago

cout<<root->left->left->data<<endl; gives="" segmentation="" fault.="" why?="">

• Reply • Share ›

ayush1gupta aman dhapola • 3 months ago

Your code will access data of NULL.

• Reply • Share ›

Sarthak Garg • 4 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 • a month ago

Thanks for pointing this out. We have corrected the typo.

• Reply • Share ›

deepak • 4 months ago

http://code.geeksforgeeks.org/

• Reply • Share ›

deepak deepak • 4 months ago

why isn't working?

• Reply • Share ›

Hari Prasath • 4 months ago

There is a mistake in algorithm
(2) should be(mirror->right)

• Reply • Share ›

GeeksforGeeks Mod Hari Prasath • a month ago

Thanks for pointing this out. We have corrected the typo

• Reply • Share ›

Shantanu • 4 months ago

you have done it using postorder fashion
we can also use preorder

• Reply • Share ›

V_CODER Shantanu • 4 months ago

i think he have done using POSTORDER(Left->Right->swap).

1 • Reply • Share ›

Shantanu ➔ V_CODER • 4 months ago

(y)

• Reply • Share ›

Dipankar Bhardwaj • 4 months ago

http://code.geeksforgeeks.org/...

• Reply • Share ›

Mohammed Raqeeb • 4 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 • 5 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 • 5 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 • 6 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 • 4 months ago

it depends upon your tree structure.!

• Reply • Share ›

Siya ➚ Mihaela mitkova • 5 months ago

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

• Reply • Share ›

Ashish kulkarni • 6 months 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 • 6 months 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 • 8 months 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 • 5 months ago

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

• Reply • Share ›

Mohammed Raqeeb ramesh kanth • 4 months ago

I think he is using a temp node i.e, ptr. So above should work.

• Reply • Share ›

Joey jas • 6 months ago

Yeah even I thought so.No issue with this it seems.

• Reply • Share ›

natasha • 8 months 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 • 8 months ago

It doesnt cover the condition if there is only one child of a node.

• Reply • Share ›

Aditya Goel • 8 months ago

pls correct the typo-
Call Mirror for right-subtree i.e., Mirror(left-subtree)

It should be right->subtree

• Reply • Share ›

Ashish Jaiswal • 9 months 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 • 9 months ago

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

• Reply • Share ›

Gohired.in • 9 months ago

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

• Reply • Share ›

anil • 10 months ago

Using Queues
void mirror(Node * root)
{
queue<node*> s;
Node *top=new Node();
s.push(root);
while(!s.empty())
{

swap(root->left,root->right);
if(top->left!=NULL)

s.push(top->left);

if(top->right!=NULL)
s.push(top->right);

s.pop();
}
}

• Reply • Share ›

radek anil • 6 months ago

Since it is queue better use enque and deque as the function names for push and pop..

• Reply • Share ›

StrictMath • a year ago

If you want to create a new mirror tree, instead of doing it in-place, here is the code - http://ideone.com/VE5Pbw

• Reply • Share ›

srikant • a year ago

struct node* newNode(int data) funtction sucks up

• Reply • Share ›

ankits • a year ago

#include
using namespace std;

class A
{
public:
static int a;
A()
{ cout<

1 • Reply • Share ›

Udbhav • a year ago

I was writing the same function but instead of using temp pointer. I tried this:

void swap(struct node** left,struct node** right)

{

struct node* temp = *left;

*left = *right;

*right = temp;

free(temp);

}

I even tried the same function without pointer to pointer. Instead :

void swap(struct node* left,struct node* right)

{

see more

• Reply • Share ›

SANKHYA Udbhav • 10 months ago

YOU ARE FREEING THE TEMP POINTER WHICH IS DE ALLOCATING THE TREE NODES, SO THIS MAKES THE OTHERS DANGLING POINTERS. AND DANGLING POINTERS SHOWS NO COMPILE TIME ERROR IN C.

• Reply • Share ›

Udbhav Udbhav • a year ago

I mean not using temp pointer in the mirror function itself.

• Reply • Share ›

Anand Barnwal Udbhav • 7 months ago

If you want want to write the swap function then you need to pass the addresses of left node and right node as you tried first. But you should not

addresses of left node and right node as you tried first. But you should not free "temp" node as it is pointing to node of the tree.

• Reply • Share ›

rihansh • a year ago

```
void *mirror2(node *root){

if(root){

swap(root->left,root->right);

mirror2(root->left);

mirror2(root->right);

}

}
```

1 • Reply • Share ›

rihansh rihansh • a year ago

i think this is the simplest way of producing mirror image following preorder traversal. Although, i have not find any code making use of pre order traversal.

• Reply • Share ›

sachin soni • a year ago

@geeksforGeeks
I have one improvement Item in your code inside a function mirror(struct node *) following line of this function must be under a condition

Improve Code:-

```
mirror(node->left);
mirror(node->right);
// when left & right child pointer both are not present Do not execute unnecessary following swapping

/* swap the pointers in this node only if child node are present */
if(node->right != NULL && node-> left !=NULL)
{
temp = node->left;
node->left = node->right;
node->right = temp;
}
```

• Reply • Share ›