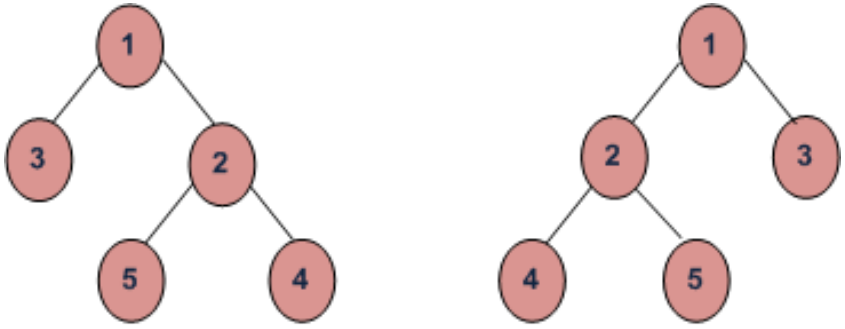
Write an Efficient C Function to Convert a Binary Tree into its Mirror Tree

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(left-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)
{
```


Popular Posts

```
  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;
}
```

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)

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

Recent Comments

karthik it should have been max_wrap= max_wrap -...
Maximum circular subarray sum · 1 minute ago

affiszerv Your example has two 4s on row 3, that's why it...
Backtracking | Set 7 (Sudoku) · 45 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 append at the end · 3 hours ago

- 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.