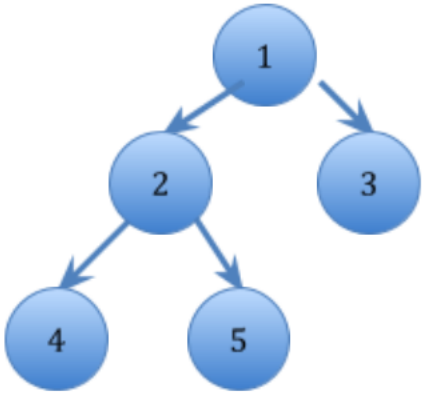
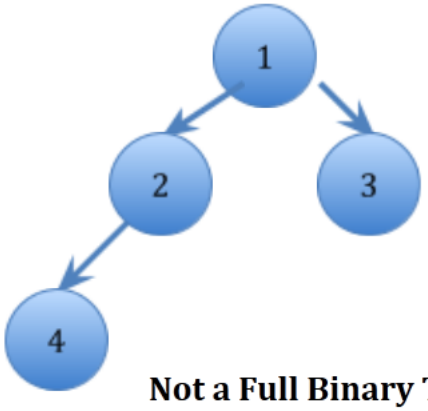
Check whether a binary tree is a full binary tree or not

A full binary tree is defined as a binary tree in which all nodes have either zero or two child nodes. Conversely, there is no node in a full binary tree, which has one child node. More information about full binary trees can be found here.

For Example:



Full Binary Tree



Not a Full Binary Tree

We strongly recommend to minimize your browser and try this yourself first.

To check whether a binary tree is a full binary tree we need to test the following cases:-

1) If a binary tree node is NULL then it is a full binary tree.

2) If a binary tree node does have empty left and right sub-trees, then it is a full binary tree by definition

3) If a binary tree node has left and right sub-trees, then it is a part of a full binary tree by definition. In this case recursively check if the left and right sub-trees are also binary trees themselves.

4) In all other combinations of right and left sub-trees, the binary tree is not a full binary tree.

Following is the implementation for checking if a binary tree is a full binary tree.

C

```
// C program to check whether a given Binary Tree is full or not
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>

/*  Tree node structure */
struct Node
{
    int key;
    struct Node *left, *right;
};

/* Helper function that allocates a new node with the
   given key and NULL left and right pointer. */
struct Node *newNode(char k)
{
    struct Node *node = (struct Node*)malloc(sizeof(struct Node));
    node->key = k;
    node->right = node->left = NULL;
    return node;
}

/* This function tests if a binary tree is a full binary tree. */
bool isFullTree (struct Node* root)
{
    // If empty tree
    if (root == NULL)
        return true;

    // If leaf node
    if (root->left == NULL && root->right == NULL)
        return true;

    // If both left and right are not NULL, and left & right subtrees
    // are full
    if ((root->left) && (root->right))
        return (isFullTree(root->left) && isFullTree(root->right));

    // We reach here when none of the above if conditions work
    return false;
}

// Driver Program
int main()
{
    struct Node* root = NULL;
    root = newNode(10);
    root->left = newNode(20);
```

```
    root->right = newNode(30);

    root->left->right = newNode(40);
    root->left->left = newNode(50);
    root->right->left = newNode(60);
    root->right->right = newNode(70);

    root->left->left->left = newNode(80);
    root->left->left->right = newNode(90);
    root->left->right->left = newNode(80);
    root->left->right->right = newNode(90);
    root->right->left->left = newNode(80);
    root->right->left->right = newNode(90);
    root->right->right->left = newNode(80);
    root->right->right->right = newNode(90);

    if (isFullTree(root))
        printf("The Binary Tree is full\n");
    else
        printf("The Binary Tree is not full\n");

    return(0);
}
```

Run on IDE

Java

```
// Java program to check if binay tree is complete or not
class Node {

    int data;
    Node left, right;

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

class BinaryTree {

    static Node root;

    /* this function checks if a binary tree is full or not */
    boolean isFullTree(Node node)
    {
        // if empty tree
        if(node == null)
        return true;

        // if leaf node
        if(node.left == null && node.right == null )
            return true;

        // if both left and right subtrees are not null
        // the are full
        if((node.left!=null) && (node.right!=null))
            return (isFullTree(node.left) && isFullTree(node.right));

        // if none work
        return false;
```

```
    }


    // Driver program
    public static void main(String args[]) {
        BinaryTree tree = new BinaryTree();
        tree.root = new Node(10);
        tree.root.left = new Node(20);
        tree.root.right = new Node(30);
        tree.root.left.right = new Node(40);
        tree.root.left.left = new Node(50);
        tree.root.right.left = new Node(60);
        tree.root.left.left.left = new Node(80);
        tree.root.right.right = new Node(70);
        tree.root.left.left.right = new Node(90);
        tree.root.left.right.left = new Node(80);
        tree.root.left.right.right = new Node(90);
        tree.root.right.left.left = new Node(80);
        tree.root.right.left.right = new Node(90);
        tree.root.right.right.left = new Node(80);
        tree.root.right.right.right = new Node(90);

        if(tree.isFullTree(root))
            System.out.print("The binary tree is full");
        else
            System.out.print("The binary tree is not full");
    }
}

// This code is contributed by Mayank Jaiswal
```

Run on IDE

Python

```
# Python program to check whether given Binary tree is full or not

# Tree node structure
class Node:

    # Constructor of the node class for creating the node
    def __init__(self , key):
        self.key = key
        self.left = None
        self.right = None

# Checks if the binary tree is full or not
def isFullTree(root):

    # If empty tree
    if root is None:
        return True

    # If leaf node
    if root.left is None and root.right is None:
        return True

    # If both left and right subtress are not None and
    # left and right subtress are full
    if root.left is not None and root.right is not None:
        return (isFullTree(root.left) and isFullTree(root.right))
```

```
    # We reach here when none of the above if condiitions work
    return False

# Driver Program
root = Node(10);
root.left = Node(20);
root.right = Node(30);

root.left.right = Node(40);
root.left.left = Node(50);
root.right.left = Node(60);
root.right.right = Node(70);

root.left.left.left = Node(80);
root.left.left.right = Node(90);
root.left.right.left = Node(80);
root.left.right.right = Node(90);
root.right.left.left = Node(80);
root.right.left.right = Node(90);
root.right.right.left = Node(80);
root.right.right.right = Node(90);

if isFullTree(root):
    print "The Binary tree is full"
else:
    print "Binary tree is not full"

# This code is contributed by Nikhil Kumar Singh(nickzuck_007)
```

Run on IDE

Output:

```
The Binary Tree is full
```

Time complexity of the above code is O(n) where n is number of nodes in given binary tree.

This article is contributed by **Gaurav Gupta**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

16 Comments Category: Trees

Related Posts:

- 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?
- Closest leaf to a given node in Binary Tree

Dev Kumar • 9 months ago
Check whether a binary tree is a full binary tree or not without recursion.
http://ideone.com/fyg6yA
1 • Reply • Share ›

Yu Yu Aung • 10 months ago
3) If a binary tree node has left and right sub-trees, then it is a part of a full binary tree by definition. In this case recursively check if the left and right sub-trees are also ****full***** binary trees themselves.
1 • Reply • Share ›

Mr. Lazy • a year ago
3 lines of code :D
http://ideone.com/8Ywyzx
• Reply • Share ›

Pushkar Kr Raijada Mr. Lazy • 9 months ago
u can write all funtions in one line to make it single line of code....
• Reply • Share ›

Mr. Lazy Pushkar Kr Raijada • 9 months ago
Thanks for letting me know! learned something new :3 Will surely do it next time :P single line code.
• Reply • Share ›

Pushkar Kr Raijada Mr. Lazy • 8 months ago
LOL.....
• Reply • Share ›

Guest • a year ago
one more way
http://ideone.com/VzOERS
• Reply • Share ›

Mohammed El-Afifi • a year ago
Since the condition for a full binary tree can be checked locally at each node, any search method can be used(BFS or DFS). The first node that violates the condition will make the tree not full, otherwise if all nodes satisfy the condition then the tree is full. Check my implementation here http://ideone.com/xqK9wy.
• Reply • Share ›

Ajitesh Mandal • a year ago

Ajitesh Mandal • a year ago

Another Easy Solution is

```
#include<stdio.h>

#include<stdlib.h>

#include<stdbool.h>

struct Node

{

int key;

struct Node *left, *right;

};

struct Node *newNode(char k)

{
```

see more

• Reply • Share ›

Avata This comment was deleted.

vixir Guest • a year ago

flag this please I can't delete my own comment

1 • Reply • Share ›

Guest • a year ago

Dude,
A full binary tree (sometimes proper binary tree or 2-tree) is a tree in which every node other than the leaves has two children.

what you have shown in the image 1 is a "Complete Binary Tree".

http://web.cecs.pdx.edu/~shear...

4 • Reply • Share ›

Guest Guest • a year ago

u are just rephrasing the definition...the above given tree is a full binary tree as well as complete...

3 • Reply • Share ›

passionatecoder • a year ago

Alternative way is to check if each level is Full or not. Level 'k' should contain pow(2,K) nodes(assuming root is at level 0) . This can be done by using BFS/level order traversal.

• Reply • Share ›

Teo Chirileanu • a year ago

thanks!

• Reply • Share ›

kanu • a year ago

int isFullTree(struct Node*p)

{

if(p==NULL)

return 1;

if((((p->left==NULL)&&(p->right==NULL))||((p->left!=NULL)&&(p>right!=NULL)))&&isFullTree(p->left)&&isFullTree(p->right))

return 1;

}

check my implementation

• Reply • Share ›

chmod ➚ kanu • a year ago

After the 2nd if you need else false

• Reply • Share ›