### 958. Check Completeness of a Binary Tree

[My Submissions](https://leetcode.com/contest/weekly-contest-115/problems/check-completeness-of-a-binary-tree/submissions/)[Back to Contest](https://leetcode.com/contest/weekly-contest-115/)

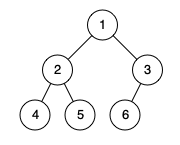
* User Accepted:1533
* User Tried:1814
* Total Accepted:1580
* Total Submissions:4271
* Difficulty:**Medium**

Given a binary tree, determine if it is a *complete binary tree*.

**Definition of a complete binary tree from** [**Wikipedia**](https://en.wikipedia.org/wiki/Binary_tree#Types_of_binary_trees)**:**

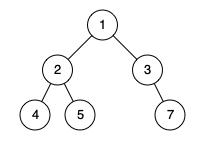
In a complete binary tree every level, except possibly the last, is completely filled, and all nodes in the last level are as far left as possible. It can have between 1 and 2h nodes inclusive at the last level h.

**Example 1:**

****

**Input:** [1,2,3,4,5,6]  
**Output:** true  
**Explanation:** Every level before the last is full (ie. levels with node-values {1} and {2, 3}), and all nodes in the last level ({4, 5, 6}) are as far left as possible.

**Example 2:**

****

**Input:** [1,2,3,4,5,null,7]  
**Output:** false  
**Explanation:** The node with value 7 isn't as far left as possible.

**Note:**

1. The tree will have between 1 and 100 nodes.

/\*\*

\* Definition for a binary tree node.

\* struct TreeNode {

\* int val;

\* TreeNode \*left;

\* TreeNode \*right;

\* TreeNode(int x) : val(x), left(NULL), right(NULL) {}

\* };

\*/

class Solution {

public:

bool isCompleteTree(TreeNode\* root) {

queue <TreeNode\*> q;

q.push(root);

int childcount=1;

bool limitreached=false;

while(!q.empty()){

int n = childcount;

childcount=0;

for(int i=0;i<n;i++){

TreeNode\* curr = q.front();

q.pop();

if(limitreached&&(curr->left!=NULL||curr->right!=NULL)) return false;

else if(curr->left==NULL&&curr->right!=NULL) return false;

else if(curr->left!=NULL&&curr->right==NULL) {

limitreached=true;

q.push(curr->left);

childcount++;

}

else if(curr->left==NULL&&curr->right==NULL) limitreached=true;

else{

cout<<"push both";

q.push(curr->left);

q.push(curr->right);

childcount+=2;

}

}

}

return true;

}

};