Description | Editorial | Solutions (4.9K) | Submissions

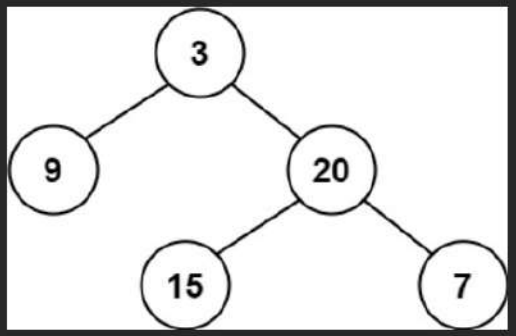
110. Balanced Binary Tree

Easy | 9.1K | 509

Companies

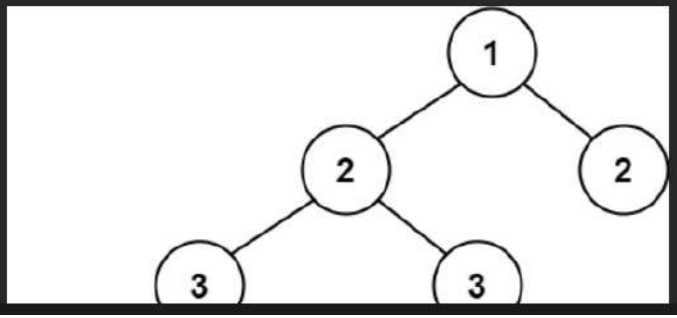
Given a binary tree, determine if it is **height-balanced**.

Example 1:




```
Input: root = [3,9,20,null,null,15,7]
Output: true
```

Example 2:



Java | Auto

```
 */
class Solution {
    public boolean isBalanced(TreeNode root) {
        return dfsHeight(root) != -1;
    }

    public static int dfsHeight(TreeNode root){
        if(root == null){
            return 0;
        }
        int leftHeight = dfsHeight(root.left);
        if(leftHeight == -1){
            return -1;
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

root =
[3,9,20,null,null,15,7]

Output

true

Expected

true

Console | Run | Submit

Description | Editorial | Solutions (3.7K) | Submissions

744. Find Smallest Letter Greater Than Target

Hint

Easy | 3.6K | 2.1K

Companies

You are given an array of characters `letters` that is sorted in **non-decreasing order**, and a character `target`. There are **at least two different** characters in `letters`.

Return *the smallest character in* `letters` *that is lexicographically greater than* `target`. If such a character does not exist, return the first character in `letters`.

Example 1:

```
Input: letters = ["c","f","j"], target = "a"
Output: "c"
Explanation: The smallest character that is lexicographically greater
than 'a' in letters is 'c'.
```

Example 2:

```
Input: letters = ["c","f","j"], target = "c"
Output: "f"
Explanation: The smallest character that is lexicographically greater
than 'c' in letters is 'f'.
```

Example 3:

```
Input: letters = ["x","x","y","y"], target = "z"
Output: "x"
Explanation: There are no characters in letters that is
lexicographically greater than 'z' so we return letters[0].
```

Java | Auto

```java
class Solution {
  public char nextGreatestLetter(char[] letters, char target) {
  int left = 0;
  int right = letters.length-1;
  while(left<=right){
      int mid = left +(right-left)/2;
          if(letters[mid]>target){
           right=mid-1;
          }
            else{
              left = mid+1;
          }
      }
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

letters =
["c","f","j"]

target =
"a"

Output

"c"

Console | Run | Submit