Question 4.1 from CTCI: Checking if Tree is Balanced

Shikha Nair

October 2, 2017

Question

Implement a function to check if a tree is balanced.
A balanced tree is defined to be a tree such that no two leaf nodes differ in distance from the root by more than one.

Explanation and Algorithm

The main thing to understand is that a tree is balanced if the shortest depth (min) and the longest depth (max) differ by no more than one. This is because the difference between max and min is the maximum distance difference possible in a tree.

The algorithm that should be used here is one which uses recursion. You should already have some idea of how to use recursion to find the height of a tree; a similar approach is used. The min and max should be found separately and then you find the difference between the two. If the difference is 0 or 1, the tree is balanced; otherwise, it is not.

Hints

1. Think about this definition of a balanced tree.
2. How do we find the max and min heights of a binary tree?
3. What do we do if the difference between these two values is less than or equal to 1? What do we do if it is not?

Code

```
public static int maxDepth(TreeNode root) {
   if (root == null) {
      return 0;
   }
   return 1 + Math.max(maxDepth(root.left), maxDepth(root.right));
 }

 public static int minDepth(TreeNode root) {
    if (root == null) {
       return 0;
   }
   return 1 + Math.min(minDepth(root.left), minDepth(root.right));
 }

 public static boolean isBalanced(TreeNode root){
   return (maxDepth(root) - minDepth(root) <= 1);
 }
```

Big O analysis

To find both the min and max height of the tree, you will reach each of the n nodes twice. Therefore the the runtime is O(n).

Sources

Question, answer and other material taken from Cracking the Coding Interview 4th edition by Gayle Laakmann McDowell.