The height of a height-balanced binary tree

A binary tree is *height-balanced* if, for every node, the heights of its two children differ by at most one. We will prove that the height of a height-balanced tree is O(size-of-tree).

The first tree on the right above is not height-balanced because the root’s empty left subtree has height -1 and its right subtree has height 1. The second tree is height-balanced.

# A height-balanced tree can be heavily weighted to one side. Look at the first tree to the right. It is not balanced: Node 2’s subtrees have heights -1 and 1, and node 0’s subtrees have heights 0 and 2. In the second tree, we add as few nodes as possible, and as far right as possible, to make it balanced: node a and then b. The right subtree (of this height-balanced tree) of the root has twice as many nodes as the left subtree, but the tree is height-balanced.

Suppose we add one more node, c, as to the right; then, to balance the tree, we add nodes d, e, f, and g in that order, as far right as possible. The resulting tree is height-balanced, but it’s heavily “weighted” to the right: The root’s right subtree has almost twice as many nodes as its left subtree.

Nevertheless, we can prove that the height of a balanced tree is O(size-of-tree).

# Below, we use simply *tree* for *binary tree* and *balanced* for *height-balanced.* Also, we use lg n for the base-2 logarithm of n. Finally, we define:

# min(h) is the minimum number of nodes in a balanced tree of height h.

We start by proving the important Lemma 1. It seems backward: Instead of giving an upper bound on the height for a given tree size, it gives a lower bound on tree size given the height. But it works!

**Lemma 1**. For a balanced tree of height h > 0, min(h) > 2h/2 - 1.

**Proof**. The proof is by induction on h.

Case h = 1. One balanced tree of height 1 has 2 nodes. We have: 2 > 21/2 - 1, so the theorem holds for h = 1.

Case h = 2. A balanced tree of height 2 has a minimum of 4 nodes —see the second tree at the top of this page.   
We have: 4 > 22/2 - 1 = 20 = 1, so the theorem holds for h = 2.

Case h ≥ 3. Assume the theorem holds for values in the range 1..h-1 and prove it holds for h. A balanced tree of height h with a minimum number of nodes has a root and two subtrees. Since the tree has a minimum number of nodes, so do it subtrees.

Since the tree has height h, One subtrees has height h-1. Since the subtrees have a minimum number of nodes, the other subtree has height h-2 (see the tree to the right). It can’t have a smaller height because the tree is balanced —for any nod the heights of its subtrees differ by at most 1. From this, we see that:

min(h) = 1 + min(h-1) + min(h-2)

We calculate:

min(h)

= < the above formula>

1 + min(h-1) + min(h-2)

> <arithmetic —delete the 1 and use fact that min(h-1) > min(h-2)>

2 min(h-2)

> <inductive hypothesis>

2 \* 2(h-2)/2 - 1

> <arithmetic>

2h/2 - 1

Q.E.D.

# We prove the theorem based on two lemmas, which are themselves proved after the theorem.

# Theorem: The height of a balanced tree of n nodes is O(lg n).

**Proof**. Let h be the height of a balanced tree. Lemma 2 below proves that

h < 2 (lg min(h)) + 2

Since min(h) is the minimum number of nodes in a balanced tree of height h, we have

h < 2 (lg n) + 2 for *any* balanced tree of height h with n > 0 nodes.

Therefore, h is O(lg n).

Q.E.D. (Quit.End.Done.)

That was the easy part of the proof! We now turn to the hard part, proving a lower bound on the number of nodes in a tree based on its height.

The following Lemma 2 uses Lemma 1 to give the lower bound on the number of nodes in a balanced tree given its height.

**Lemma 2.** h < 2 lg min(h) + 2.

**Proof.** From lemma 1, we have:

min(h) > 2h/2 – 1

= <take lg of both sides>

lg min(h) > h/2 – 1

= <arithmetic>

h /2 < lg min(h) + 1

= <arithmetic>

h < 2 lg min(h) + 2