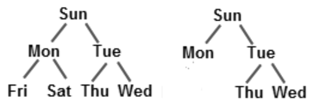
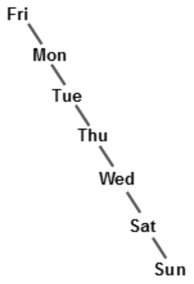
Balanced tree

The tree with n nodes is *balanced* if its height is O(n).



For example, we would all say that the two binary trees on the left are balanced —their height, 2, *is* the minimum possible with 7 or 5 nodes. But the binary tree on the right is not balanced.



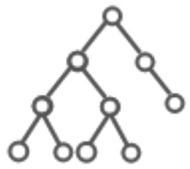
If this is your first introduction to trees, then just stick with this rough idea of a balanced tree and worry about details later. In what you are learning, the precise definition may not be important. Later on, you may want to come back and study the notion of *balanced* tree in more detail.

Below, we (1) give two properties that ensure that a binary tree is, (2) explain why there are several such properties (there are more), and (3) show what happens if a definition is taken out of context.

Height-balanced and weight-balanced binary trees

Here are two definitions:

1. A binary tree is *height-balanced* if for each node the heights of its subtrees differ by at most 1. (Remember, the height of an empty subtree is -1).
2. A binary tree is *weight-balanced* if for each node the numbers of inner nodes in its left subtree and right subtree differ by at most 1.



The tree to the right is height-balanced. It is not weight-balanced because the left subtree of the root has 3 inner nodes but the right subtree of the root has only 1. It has been shown that a weight-balanced tree is also height-balanced.

The key term in both definitions is "for each node". The property being described must hold for *all* nodes, not just the root. To see the need for "for each node", consider the tree whose root has a copy of the tree to the right as its left subtree and another copy as its right subtree. It is not weight-balanced, even though the left and right subtrees of the root have the same number of inner nodes.

Why are there different definitions of *balance*

The important data structure called a Binary Search Tree, or BST, was invented in order to make searching for a value more efficient, say, than searching in an unordered array. However, if the BST tree is unbalanced, as is the BST containing the months in the upper right corner of this page, then search is not efficient. One would rather keep the tree balanced, as is the one containing the months on the upper left. It is difficult and inefficient to keep a BST balanced as values are added and deleted.

Therefore, computer scientists looked for ways to enhance the BST data structure in order to make balancing more efficient as values are added and deleted. This required modifying the BST —by adding more information to each node or by allowing a node to have more than two children. Invented were the AVL tree, the 2-3 tree, the B-tree, the red-black tree, and many more.

With each such new tree data structure, the inventor had to say what property they meant by *balanced.* The goal was the same, of course: if the tree was balanced according to the definition, then the inventors could prove that the height of the tree was O(log size-of-tree).

Don't take definitions of out of context

The wikipedia description of a B-tree (https://en.wikipedia.org/wiki/B-tree) says that "A B-tree is kept balanced by requiring that all leaf nodes be at the same depth." Well, look at the tree of months in the upper right of this page. It has only one leaf, so all leaf nodes are at the same depth! How can they say that it is balanced? The problem is that that tree of months is not a B-tree; it does not satisfy all the requirements of being a B-tree.

Moral: One has to be careful in lifting definitions from one place to be used in another.

©David Gries, 2018