Alan García - Binary Tree

Written exercise:

1. What does it mean if a binary search tree is a *balanced tree*?

   A binary tree is considered balanced if the left and right branches have the same height or a height difference no larger than 1.

2. What is the big-Oh search time for a balanced binary tree? Give a logical argument for your response. You may assume that the binary tree is perfectly balanced and full.

   Recall that we could drastically narrow down searches with a sorted array by using binary search. We can apply the same principle in a perfectly balanced tree.

   We would only have to traverse each level h once, because each decision we make will reduce by half the size of our subtree.

   For example: On the first level, when we decide if we go left or right, we are already narrowing down our search by half. On a hypothetical second level we make a decision and that subtree is once again narrowed by half.

   This is exactly the same math we used to calculate binary search of a sorted array, but this time we apply it to a binary search tree. This takes an average big-O of logn.

3. Now think about a binary search tree in general, and that it is basically a linked list with up to two paths from each node. Could a binary tree ever exhibit an O(n) worst-case search time? Explain why or why not. It may be helpful to think of an example of operations that could exhibit worst-case behavior if you believe it is so.

   The easiest example is a perfectly unbalanced tree, where all of the n nodes are located to the left of the previous node.

   This structure defeats the purpose of having a binary tree and it becomes a single linked list. Which will take O(n) to find anything in the worst case, last place.