CSCI 104L Lecture 23 : AVL Trees

**Question 1.** How would you insert a new value into a (non-balanced) binary search tree?

**Question 2.** For the integers 1 through 7, is there an order you can insert them into an (initially empty) binary search tree such that any search will be efficient?

**Question 3.** For the same set, is there an order that will provide a “bad” binary search tree?

**Deleting from a Binary Search Tree**

Consider the following binary search tree:

```
  44
 /   \
17    62
 |    /   \  \
32   50    78
 |   |       |
48  54   88
```

**Question 4.** Starting from the above tree, what does the tree look like if we remove 32?

**Question 5.** What if we had deleted 17 instead?

**Question 6.** What if we had deleted 44 instead?

**Question 7.** What if we had deleted 62 instead?
We say that a tree is an **AVL Tree** if the following two conditions both hold:

- The **binary search tree** property holds for all nodes.
- For every node $v$ of $T$, the heights of the children of $v$ differ by at most 1. This is referred to as the **height-balance property**.

Note that this means that any subtree of an AVL tree is itself an AVL tree.

![AVL Tree Diagram]

**Question 8.** What is the minimum number of nodes an AVL tree can have if it has height $h$?

**Question 9.** How long does $\text{find}(v)$ take in an AVL tree?

**Insert Operations**

**Question 10.** Consider the example AVL tree from earlier, and consider what would happen if we were to insert the value 14 into it. Is it still an AVL tree?

**Question 11.** Now consider the example AVL tree from earlier (with or without 14 added), and consider what would happen if we were to insert the value 34 into it. If we follow the standard binary search tree insertion algorithm, is it still an AVL tree?
After inserting a node to a binary search tree, we apply the following procedure:

Start at the newly-inserted node and walk up to the root, checking if each node is balanced (the height-balance rule applies to this node). If a node is unbalanced, rotate the subtree rooted at that node. Rotate the following three nodes:

1. Let $z$ be (a pointer to) the first unbalanced node on the way up.
2. Let $y$ be the child of $z$ with greater height (hint: this is always an ancestor of the node you inserted. Why?). Why are ties impossible?
3. Let $x$ be the child of $y$ with greater height (this is always an ancestor of the node you inserted, or the node itself. Why?). Why are ties impossible?

When $x, y, z$ form a zig-zag pattern, we do a **double rotation**. Otherwise we do a **single rotation**. The rotations are pictured on the next page.

**Question 12.** After doing a rotation, the tree is guaranteed to be balanced, and we can stop. Why?

**Question 13.** Starting with the AVL tree that we finished the last example with, insert the value 54. What is the resulting AVL tree?

**Question 14.** Starting with the AVL tree that we finished the last example with, insert the value 100. What is the resulting AVL tree?

**Question 15.** Starting with an empty AVL tree, insert the following values into the tree, in sequence:
1, 2, 3, 12, 9, 13, 7, 4, 6, 5, 8