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.

**Question 1.** What is the minimum number of nodes an AVL tree can have if it has height \( h \)?

**Question 2.** How long does \( \text{find}(v) \) take in an AVL tree?

**Insert Operations**

**Question 3.** 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 4.** 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 5.** After doing a rotation, the tree is guaranteed to be balanced, and we can stop. Why?

**Question 6.** Starting with the AVL tree that we finished the last example with, insert the value 54. What is the resulting AVL tree?

**Question 7.** Starting with the AVL tree that we finished the last example with, insert the value 100. What is the resulting AVL tree?

**Question 8.** 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
Deleting from an AVL Tree

To delete from an AVL tree, follow the same procedure as removal from a binary search tree. Then, starting at the node that was removed, move up to the root, recalculating heights if necessary. For each node \( z \), if it is unbalanced, rotate the subtree rooted at that node:

1. Let \( z \) be (a pointer to) the unbalanced node we found.
2. Let \( y \) be the child of \( z \) with greater height (hint: this is never an ancestor of the node you deleted. Why?). Why are ties impossible?
3. Let \( x \) be the child of \( y \) with greater height. In the event of a tie, choose \( x \) so that we’ll have a single rotation instead of a double rotation

Then perform the same rotation you would do for that formation on insertion. Unlike insertion, however, this only fixes the problem locally – it might be unbalanced higher up. You need to continue this until the tree is balanced globally.

![AVL Tree Diagram](Image)

**Question 9.** Starting from the AVL tree above, what does the tree look like if we remove 32?

**Question 10.** What if we had deleted 17 instead?

**Question 11.** What if we had deleted 78 and 88 (in either order) instead?

**Question 12.** Starting with an empty AVL tree, do the following operations, in sequence:

- INSERT: 1, 2, 3, 12, 9, 13, 7, 4, 6, 5, 8
- DELETE: 4, 1
- INSERT: 1, 14, 11
- DELETE: 3, 13, 12, 11, 14, 2, 3, 7, 8, 9
<table>
<thead>
<tr>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Diagram" /></td>
<td><img src="image2.png" alt="Diagram" /></td>
</tr>
<tr>
<td><img src="image3.png" alt="Diagram" /></td>
<td><img src="image4.png" alt="Diagram" /></td>
</tr>
<tr>
<td><img src="image5.png" alt="Diagram" /></td>
<td><img src="image6.png" alt="Diagram" /></td>
</tr>
<tr>
<td><img src="image7.png" alt="Diagram" /></td>
<td><img src="image8.png" alt="Diagram" /></td>
</tr>
<tr>
<td><img src="image9.png" alt="Diagram" /></td>
<td><img src="image10.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>