AVL Trees

Observation

1. Imagine you are inserting a sequence of keys into an initially empty AVL tree, but you are only allowed to use left rotations. Is it possible to maintain the AVL property in all cases? If not, describe a scenario where it fails.
2. You've been handed an AVL tree and told it's balanced, but you can't see the node heights. What's the minimum set of node heights you need to check to verify if the tree is indeed AVL balanced?
3. If an AVL tree has a height of 20, what is the minimum number of nodes it can contain? Also, if you never want to give it up, what's the maximum number of nodes it could have?
4. Suppose you have a perfectly balanced Binary Search Tree (BST) and an AVL tree containing the same keys. If both trees perform 100 insertions, which one is more likely to require fewer rotations, and why?
5. Imagine you are tasked with designing a variation of AVL trees that only allows a maximum height imbalance of 1 for nodes at an even level and 2 for nodes at an odd level. Would this new tree structure still maintain logarithmic height? Why or why not?

Execution

1. Imagine you are building an event scheduling system where each event has a unique ID and a start time (represented as an integer). Events are added dynamically, and it's critical that the list of events remains balanced to ensure quick access for upcoming events. To manage this, you decide to store the events in an AVL tree, where each node contains the event ID and start time. Each insertion should maintain AVL balance so that looking up the next event after any insertion is efficient. Implement a function to insert a new event into the AVL tree, ensuring that the tree remains balanced after each addition. Write the necessary AVL rotations (single and double) to handle any imbalances. Also, write a helper function to find the upcoming event with the earliest start time after each insertion.

2. In a magical forest, each tree has a special power determined by the number of fruits it holds. The trees are represented in an AVL tree, and you must maintain the balance of the forest as trees are planted and uprooted. The wizard needs to find the most powerful tree (the one with the most fruits) quickly.

   Task:

   1. Write a function to **plant a new tree** with a specified fruit count.
   2. Write a function to **uproot a tree** by its ID.
   3. Write a function to **find the most powerful tree** (the one with the highest fruit count).