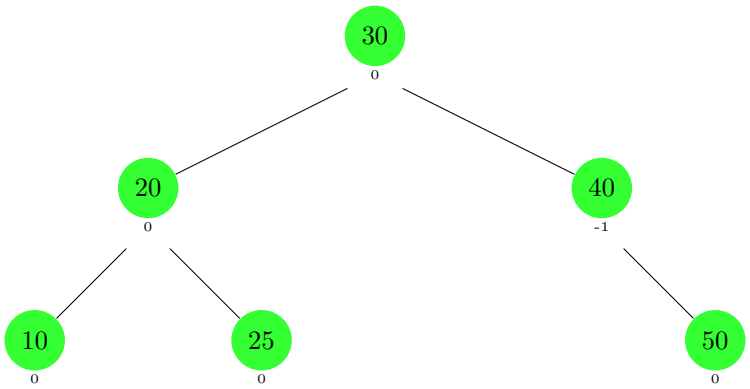
AVL Tree Algorithm Overview

An AVL tree is a self-balancing binary search tree where the height difference between the left and right subtrees of any node is at most one. This balance ensures that the tree remains approximately balanced, leading to $O(\log n)$ time complexity for search, insertion, and deletion operations.

Key Concepts

- **Height**: The height of a node in an AVL tree is the length of the longest path from the node to a leaf. This height is used to determine the balance of the node.
- **Balance Factor**: For any node in the AVL tree, the balance factor is calculated as the height of the left subtree minus the height of the right subtree. It helps in determining if the node is balanced, left-heavy, or right-heavy.
- **Rotations**: To maintain balance in the AVL tree after insertions and deletions, rotations are performed. There are four types of rotations:
  - **Right Rotation (Single Rotation)**: Applied when a left-heavy subtree needs balancing.
  - **Left Rotation (Single Rotation)**: Applied when a right-heavy subtree needs balancing.
  - **Left-Right Rotation (Double Rotation)**: Applied when a left subtree has a right-heavy child.
  - **Right-Left Rotation (Double Rotation)**: Applied when a right subtree has a left-heavy child.

Initial AVL Tree



Final AVL Tree