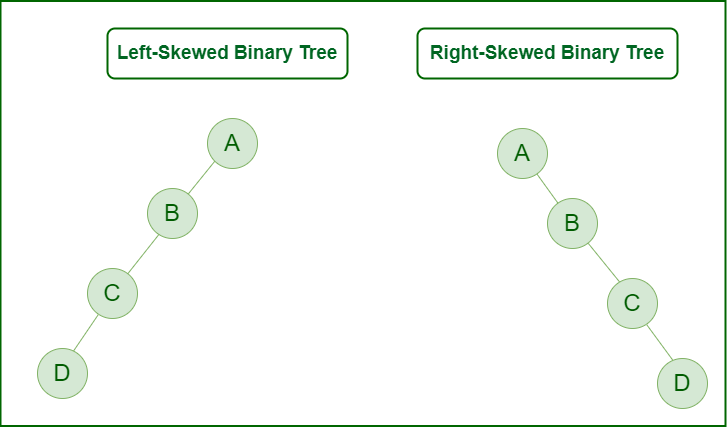
**Need for Self-Balancing Trees.**

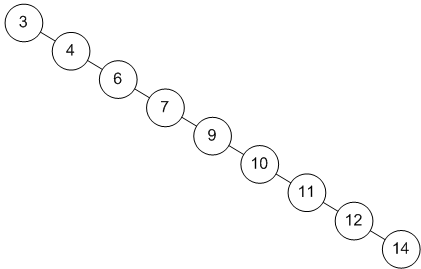
A **Binary Search Tree (BST)** provides efficient search, insertion, and deletion operations, but its performance **depends on its shape**. If a BST becomes **skewed**, it degrades to a **linked list**, leading to poor performance (**O(n) time complexity**).

Self-balancing trees solve this problem by maintaining a **balanced structure**, ensuring that operations stay efficient with **O(log n) time complexity**.



Let's understand the importance of tree balancing with an example of a right-skewed tree.

**Right Skewed Binary tree**



In the above **right-skewed binary tree**, searching for node **14** requires **9 comparisons**. However, if the tree is **balanced**, the number of comparisons is significantly reduced.

In the **Balanced AVL Tree** shown below, only **4 comparisons** are needed to check whether **14** is present. This demonstrates how balancing a tree improves search efficiency.

