Lab 5 – Exploring Binary Search Trees (BSTs)

Luca Novello | DSA456V1A | April 6, 2025

How the BST Works

The BST organizes the data so that each node has a key greater than all keys in its left subtree, and less than or equal to those in the right subtree. The insertion uses recursive comparisons to maintain its structure.

Insertion

Each key is placed recursively according to its value, forming a tree that can be balanced or skewed depending on the insertion order. The insertion function compares the new key to the current node's key, going left for smaller values and right for greater or equal values. This process continues until a null position is found, and the new node is inserted.

Height and Imbalance

- **Height**: Calculated as the longest path from the root to a leaf.
- **Imbalance**: Defined at the root, the absolute difference in height between left and right subtrees.

Example Output

```
Tree 1: Height = 9, Imbalance = 9
Tree 2: Height = 11, Imbalance = 8
Tree 3: Height = 7, Imbalance = 3
Tree 4: Height = 8, Imbalance = 8
...

Summary (Height, Imbalance):
Tree 1: (9, 9)
Tree 2: (11, 8)
Tree 3: (7, 3)
Tree 4: (8, 8)
...
[Finished in 44ms]
```

Observations Across Different Trees

- Tree heights varied from 5 to 7, showing differences due to insertion order.
- Imbalances ranged from 0 to 4.
- Even with random inserts, some tree remained relatively balanced showing that complete imbalance is not common with small random datasets.