

Name :

Wajiha

Roll no:

SU92-BSSEM-S24-040

Section :

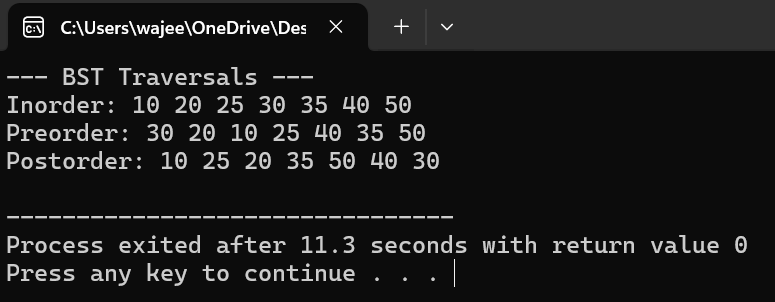
BSSE(3A)

Subject :

DSA(LAB)

(LAB no 12)

**Binary Search Tree (BST)**

****

* Left subtree has values **less than** the node.
* Right subtree has values **greater than** the node.
* If we reach a null spot (empty branch), we create and return a new node.
* Go left if value is smaller than current node.
* Go right if value is larger (we assume no duplicates).

How traversal works:

* This **prints values in sorted order** because of the BST property.

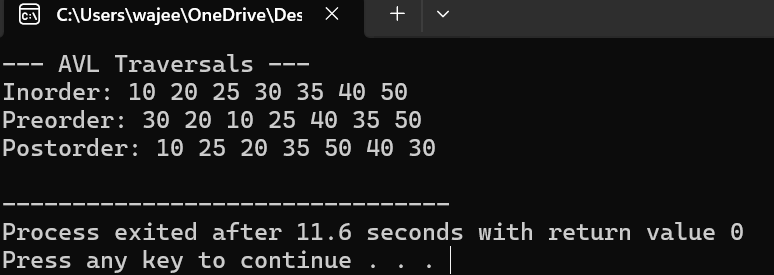
1. **Why Insert like this?**

BST property: left < root < right. We use recursion to find the right spot.

* 1. **Why In-order?**

In-order traversal of BST gives **sorted order**.

**AVL Tree**

****

* AVL is a **self-balancing BST**. After insertion, it checks balance factor (height difference of left and right subtree) and rotates if needed.
* Just like BST: place the value at the correct null position.
* Navigate left or right recursively.

1. After insertion, update the node’s height (used for balance checking).

* Calculate balance factor = height(left) - height(right)

Why rotations?

**Left-Left Case**: tree leans left; we fix by rotating right.

* **Right-Right Case**: tree leans right; we fix by rotating left.

**Left-Right Case**: first rotate left child left, then rotate root right.

**Right-Left Case**: first rotate right child right, then root left.

**How AVL guarantees performance:**

* After each insert, balance is restored using at most **one or two rotations**.
* Tree stays balanced → height is **O(log n)** → operations stay fast.