Name of the Student: Aditya Raut

Class: SY-06

Enrollment Number: ADT24SOCB0082

Topic: Tree

Binary Search Tree (BST) Visualization and Traversals

AI tool Used: Cursor (GPT-5 coding assistant)

Concept Statement

A Binary Search Tree (BST) is a node-based binary tree where every node satisfies the BST property: for a node with value v, all values in its left subtree are less than v, and all values in its right subtree are greater than v. This structure enables efficient search, insert, and delete operations with average-case time complexity of O(log n).

Explanation of the Concept

**BST properties**

* Each node stores a value and pointers to left and right children.
* Inorder traversal of a BST yields a sorted (non-decreasing) sequence.
* Average time complexities: search/insert/delete O(log n); worst-case O(n) for skewed trees.

**Core operations implemented**

* **Insert**: Compares from root, branching left/right until an empty spot is found.
* **Search**: Compares the target at each step, branching left if smaller and right if larger until found/not found.
* **Traversals**:
  + **Inorder (Left → Root → Right)**: returns values in sorted order.
  + **Preorder (Root → Left → Right)**: useful for serialization or copying.
  + **Postorder (Left → Right → Root)**: useful for deletions/evaluations.

**Visual behavior**

* Pre-built BST values: 50, 25, 75, 15, 35, 65, 85, 10, 20, 30, 40, 60, 70, 80, 90.
* Click-to-search: clicking a node highlights the search path from root to that node.
* Traversal visualization: pressing Inorder/Preorder/Postorder shows sequence and highlights nodes step-by-step.
* Color coding: Green for search path; Purple for traversal sequence; smooth hover effects on nodes.

**Implementation overview**

* Tech stack: HTML, Tailwind CSS, JavaScript (ES6 classes), D3.js v7.
* Classes: BSTNode and BinarySearchTree; conversion to D3 hierarchy for rendering.
* D3 tree layout for automatic node spacing and redrawing.
* Modular files: index.html, style.css, script.js.

Brief Intro about the AI tool

Cursor with GPT-5 assists coding by generating modular code, visualizations, and UI; refactoring into separate CSS/JS; resolving UX issues (e.g., hover behavior); and iterating on requirements (pre-built tree, click-to-search, simplified controls) rapidly.

Snapshots of the Output

* Default tree view with blue circular nodes and connecting edges.
* Click-to-search example for 35: 50 → 25 → 35.
* Click-to-search example for 90: 50 → 75 → 85 → 90.
* Inorder traversal output (sorted sequence) displayed in the text area.

(Insert screenshots here in Word: Tree Visualization, Search Path panel, Traversal output.)

Link for the Implementation

**Hosted**: GitHub/Pages link: https://dsa-jury.pages.dev/

**Local run**: Open index.html in a modern browser.

**Files**: index.html, style.css, script.js