Data Structures and Algorithms using C++

Industry Standard Documentation

1. Project Charter:

- **Project Title: Binary Search Tree implementation**
- **Project Manager: Keshav Yadav**
- **Start Date:** 13-07-2024
- **End Date:** 17-07-2024
- **Objectives:** To implement BST (Binary Search Tree)
- **Scope:**Searching,Sorting: Range queries: Finding elements within a specified range
  Data storage: Databases,Artificial intelligence: Decision trees and rule-based systems
- **Deliverables:** Insights, conclusions, and recommendations.

2. Business Requirements Document (BRD):

- **Business Problem:** Difficulty in efficiently managing and searching sorted data in large datasets, leading to slower performance in data retrieval operations..
- **Business Objectives:** To improve data management and retrieval efficiency by implementing a binary search tree (BST).
- **Functional Requirements:** Implementation of a binary search tree.
  Operations: Insertion, Deletion, Search, Traversal (In-order, Pre-order, Post-order).
  Visualisation of the tree structure.
- **Non-functional Requirements:** Performance: Efficient handling of large datasets.
  Scalability: Ability to handle increasing amounts of data without performance degradation.
  Usability: Easy-to-understand and well-documented code.

3. Technical Requirements Document (TRD):

- **Data Sources:** Input data for the tree (can be dynamically provided by the user or from a predefined dataset).
- **Technologies:** Python, Jupyter Notebook, Matplotlib (for visualisation), NetworkX (for tree visualisation
- **Architecture:** Data input and validation, Implementation of BST operations (Insert, Delete, Search, Traversals),Visualization of the BST structure
- **Data Flow:** Import data,Validate data,Implement BST operations,Perform BST operations (Insertion, Deletion, Search, Traversal),Visualize the BST structure

4. Project Plan:

- **Tasks:** Data collection, Implementation of BST operations (Insert, Delete, Search, Traversals), Visualization of the BST structure, Documentation
- **Risks:** Data quality issues, algorithm performance, visualization limitations

Overview:

The project is a Binary Search Tree (BST) implementation in C++, focusing on key operations such as insertion, searching, and deletion.

Tools and Libraries: Standard C++ libraries.

Features and Functionalities:

1. Insert Elements: Users can insert integers into the BST. The tree maintains its properties by placingsmaller values in the left subtree and larger values in the right subtree.

2. Search Elements: Users can search for a specific integer in the BST. The program will return whetherthe element is found or not.

3. Delete Elements: Users can delete an element from the BST. The deletion process maintains thestructure of the tree, ensuring the BST properties are preserved.

4. Display Elements: The program supports multiple traversal methods to display elements:

- **In-order Traversal:** Outputs elements in sorted order (left, root, right).
- **Pre-order Traversal:** Displays elements in pre-order sequence (root, left, right).
- **Post-order Traversal:** Displays elements in post-order sequence (left, right, root).

5. User-Friendly Menu: A simple text-based menu allows users to interact with the BST and choosevarious operations easily.

Usage Example:

The user interacts with the program through a menu that allows them to perform various operations on the Binary Search Tree (BST):

Select Operations: The user selects operations from the menu: insert, search, delete, display, or exit.

Insertion: The user specifies how many values to insert and enters each value.

For example, if the user wants to insert three values (15, 10, 20), they will input these sequentially, and the program will confirm each insertion.

Searching: The user can specify how many values to search for, entering each one.

The program provides feedback on whether each value is found in the tree or not.

Deletion: The user specifies how many values to delete and enters them.

Feedback is provided for each deletion attempt, indicating success or failure.

Display: The user can display all elements in the tree in sorted order at any time.

Exit: The program continues to operate until the user chooses to exit the menu.

Example Interaction:

Insert Elements: User inserts values: 15, 10, 20.

Outputs: "Inserted 15", "Inserted 10", "Inserted 20".

Search for Elements: User searches for values: 10,

25.**Outputs:** "Found" for 10 and "Not found" for 25.

Delete Elements: User deletes value: 10.

Output: "Deleted node with key 10".

Display Elements: User displays elements in the BST.

- **In-order:**
  - Output: "In-order: 15 20"
- **Pre-order:**
  - Output: "Pre-order: 15 10 20"
- **Post-order:**
  - Output: "Post-order: 10 20 15"

Exit: User chooses to exit the program.

Conclusion:

This project demonstrates the implementation of a Binary Search Tree in C++ with fundamental operations that are crucial for understanding data structures. The user-friendly interface allows for straightforward interaction with the BST. This implementation can serve as a foundational exercise for those learning about data structures and algorithms.