Joaquin Esguerra Jr

CS 499

4-2 Milestone Three: Narrative

Instructor Krupa

September 27, 2024

Enhancement Two: Algorithms and Data Structure - Narrative

**Brief Description of the Artifact:**

The file BST.cpp contains the implementation of a Binary Search Tree (BST) data structure. This data structure organizes nodes in such a way that the nodes in the left subtree have smaller values, while the nodes in the right subtree have larger values. The file includes functions for common BST operations like insertion, deletion, searching, and traversal (e.g., in-order, pre-order, post-order).

BST.cpp was created in the fall of 2021.

**Justification for Inclusion in ePortfolio:**

Incorporating BST.cpp into my ePortfolio is a justified decision to demonstrate my proficiency in algorithms and data structures. The Binary Search Tree (BST) plays a crucial role in computer science, enabling efficient search, insertion, and deletion operations. By including a BST implementation, I aim to showcase my strong algorithmic thinking and deep understanding of data structures, emphasizing my expertise in dynamic memory management and recursive algorithms.

This addition highlights the significance of data structures and algorithms for individuals with a technical focus in their education or professional pursuits. It holds particular relevance for technical interviews and job assessments where understanding BSTs is vital. Furthermore, the implementation demonstrates my proficiency in handling intricate operations such as node insertion, deletion, and searching with efficiency.

Moreover, by prioritizing code quality and optimization in my implementation, including robust error handling, clear comments, and improved performance, I can enhance the artifact. Reorganizing for better readability and efficiency improvements, as well as incorporating additional features like tree balancing mechanisms or a user interface, can further boost the artifact's value.

In order to guarantee the quality of my BST implementation, it is crucial to conduct thorough testing using automated tests and performance benchmarks. By integrating these components into my ePortfolio, I aim to showcase not only my technical expertise but also my problem-solving approach, code quality standards, and dedication to continual improvement.

Including BST.cpp in my ePortfolio highlights my technical skills and demonstrates my expertise in algorithms and data structures. This showcases my readiness for roles that demand a strong understanding of these fundamental concepts in computer science.

**Meeting Course Outcomes:**

To assess my attainment of the course outcomes by improving the BST.cpp artifact and exploring potential updates to my coverage plan, I focused on the following:

The initial objectives of the course encompassed grasping data structures, algorithmic efficiency, and problem-solving skills. Through the implementation and enhancement of the BST.cpp file, I have acquired a deeper understanding of diverse data structures, optimized algorithms, and showcased effective problem-solving methodologies.

During my hands-on experience with BST operations, I gained a deep understanding of data structures and their efficiency. I focused on improving algorithmic efficiency and problem-solving skills, which was reflected in the enhancements made to the insertion, deletion, and search functionalities of the BST.cpp.

I’ll consider enhancing my algorithmic implementation by incorporating more advanced data structures such as AVL Trees or Red-Black Trees. This will demonstrate my proficiency in handling complex algorithms. Additionally, utilizing BST in practical applications like database indexing can showcase the real-world relevance of data structures. To further exhibit my understanding of software development practices, I’ll focus on developing improved testing protocols and documentation. Collaboration on version control systems like GitHub will also highlight my expertise in this area.

By strategically revising my plans to encompass a broader scope of course outcomes, I can showcase a comprehensive skill set that can be advantageous for academic evaluations and prospective career prospects.

**Reflection on the Enhancement Process:**

Reflecting on the process of enhancing and modifying the BST.cpp artifact led to various learning experiences and challenges:

**Learning Experiences:**

Upon delving into the implementation of Binary Search Trees (BST), I gained a deep understanding of key operations such as insertion, deletion, and search. This involved grasping the intricacies of managing nodes within a tree structure. Additionally, exploring traversal techniques including inorder, preorder, and postorder traversals provided valuable insights into diverse methods of accessing and manipulating tree nodes.

Delving into algorithm optimization, I learned to pay close attention to time and space complexities, which underscored the significance of operations such as balancing for efficiency. The exploration of AVL trees and Red-Black trees prompted contemplation of self-balancing algorithms.

Furthermore, in terms of coding practices and refinement, I realized the importance of code refactoring in enhancing readability and maintainability. Moreover, addressing edge cases underscored the significance of comprehensive error handling in bolstering reliability.

In the realm of testing and validation, creating unit test cases proved instrumental in ensuring program functionality correctness across various scenarios. Implementing scripts for testing automation shed light on the efficiencies gained through automation in CI/CD pipelines.

**Challenges Faced:**

When it comes to balancing algorithms, implementing self-balancing BSTs such as AVL trees or Red-Black trees introduces significant complexity and requires diligence. Balancing optimizations with readability and simplicity posed challenges in terms of performance optimization. Managing memory efficiently, especially with manual memory management in C++, is challenging to prevent leaks. Ensuring robustness with edge cases requires thorough testing and understanding. Choosing between recursive and iterative methods posed challenges, especially considering risks like stack overflow. Delving into thread-safety and synchronization mechanisms for concurrent access added complexity in terms of concurrency and thread-safety. Creating clear documentation was critical for code maintenance. Learning effective version control practices for collaborative coding is imperative in terms of documentation and collaboration.

**Final Thoughts:**

Improving the BST.cpp artifact was a valuable learning experience that enhanced my understanding of algorithms. Overcoming challenges underscored the significance of adhering to best practices, thorough testing, and continuous learning for upcoming projects. These insights will inform a more effective approach to my future software development endeavors.