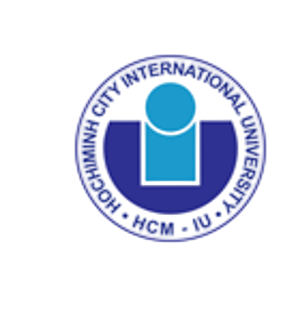
**International University Algorithms & Data Structures**

**School of Computer Science and Engineering**

**VIETNAM NATIONAL UNIVERSITY – HOCHIMINH CITY THE INTERNATIONAL UNIVERSITY**

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

****

**Algorithms & Data Structures IT013IU**

**Final Report**

**Topic: Library Management System**

By:  
 Nguyễn Việt Thảo ITCSIU23058

Instructor: Thái Trung Tín

**1) Data Structure**

The **Library Management System** utilizes a **Binary Search Tree (BST)** as its primary data structure for managing the collection of books. Each book is represented as a node in the tree, which contains the following attributes:

* **ISBN (String)**: Unique identifier for each book, used as the key for organizing the BST.
* **Title (String)**: The title of the book.
* **Author (String)**: The author's name.
* **Borrowed (boolean)**: A flag indicating whether the book is currently borrowed.
* **Left and Right (Book references)**: Pointers to the left and right children in the BST.

The root of the BST represents the starting point for storing and searching books. Books are organized such that:

* Nodes in the left subtree have ISBN values smaller than the parent.
* Nodes in the right subtree have ISBN values greater than the parent.

This hierarchical structure ensures efficient management and retrieval of books.

**2) Algorithms**

**a) Adding a Book**

The **addBook** method inserts a new book into the BST. The algorithm works recursively to find the correct position for the new book based on its ISBN:

1. Start at the root.
2. Compare the ISBN of the new book with the current node:
   * If smaller, traverse to the left subtree.
   * If larger, traverse to the right subtree.
3. Insert the new book at the first null position encountered.

**b) Searching for a Book**

The **searchBook** method searches for a book by its ISBN. The algorithm recursively traverses the BST:

1. Start at the root.
2. Compare the target ISBN with the current node:
   * If they match, return the current node.
   * If smaller, search the left subtree.
   * If larger, search the right subtree.
3. If a null node is reached, the book does not exist.

**c) Deleting a Book**

The **deleteBook** method removes a book from the BST while maintaining its structure:

1. Locate the book to be deleted.
2. Handle the following cases:
   * **Leaf Node**: Simply remove it.
   * **One Child**: Replace the node with its child.
   * **Two Children**: Replace the node with its in-order successor (the smallest node in the right subtree), then recursively delete the successor.

**d) Displaying Books in Order**

The **displayBooksInOrder** method performs an in-order traversal of the BST:

1. Recursively traverse the left subtree.
2. Process the current node (e.g., print its details).
3. Recursively traverse the right subtree.

**e) Borrowing and Returning Books**

* **Borrow Book**: Search for the book by ISBN. If found and not already borrowed, mark it as borrowed.
* **Return Book**: Search for the book by ISBN. If found and currently borrowed, mark it as available.

**f) Counting Books**

The **countBooks** method calculates the total number of books in the BST by recursively counting nodes:

1. If the current node is null, return 0.
2. Otherwise, return 1 plus the counts of the left and right subtrees.

**3) Time Complexity**

**a) Adding a Book**

* **Best Case**: (balanced tree).
* **Worst Case**: : O (n)

**b) Searching for a Book**

* **Best Case**: (balanced tree).
* **Worst Case**: : O (n)

**c) Deleting a Book**

* **Best Case**: (balanced tree).
* **Worst Case**: : O (n)

**d) Displaying Books in Order** : O (n)

* : Each node is visited once.

**e) Borrowing and Returning Books:** : O (n)

* **Search Operation**

**f) Counting Books :** : O (n)

* : Each node is visited once.

**4) References**

1. Blackboard. (n.d.). *Lecture 09-10 - Tree*. Retrieved from <https://blackboard.hcmiu.edu.vn/bbcswebdav/pid-94194-dt-announcement-rid-5581038_1/courses/IT013IU1_2024-2503/Lecture%2009-10%20-%20Tree.pdf>
2. freeCodeCamp.org. (n.d.). *Binary Search Tree (BST) Explanation* [Video]. YouTube. Retrieved from <https://www.youtube.com/watch?v=DdBIFZphmck&list=PLJfV3mV9d1ZVJJUTckluxRSXMlcWpS8gQ>
3. GeeksforGeeks. (n.d.). *Library Management System*. Retrieved from <https://www.geeksforgeeks.org/library-management-system/>
4. CodeWithHarry. (n.d.). *Library Management System Project Explanation* [Video]. YouTube. Retrieved from <https://www.youtube.com/watch?v=FjQ0uWyXTuo&list=PLuji25yj7oILToHnJepnIjSGufSK9qU_d>
5. Muneer, H. (n.d.). *Library Management System in Java* [GitHub Repository]. GitHub. Retrieved from <https://github.com/harismuneer/Library-Management-System-JAVA/tree/master/Project/build/classes/LMS>
6. Patel, R. (n.d.). *Library Management System Using Binary Search Tree* [GitHub Repository]. GitHub. Retrieved from <https://github.com/rajvi-patel-22/Library-Management-System-Searching-catalogues-in-library-using-binary-search-tree/blob/master/library_management.java>
7. Imtiaz, M. (n.d.). *Library Management System in Java* [GitHub Repository]. GitHub. Retrieved from <https://github.com/minaahilimtiaz/Library-Management-System-Java>