S. B. JAIN INSTITUTE OF TECHNOLOGY, MANAGEMENT & RESEARCH, NAGPUR.
(An Autonomous Institute, Affiliated to RTMNU, Nagpur)
DEPARTMENT OF EMERGING TECHNOLOGIES (AI&ML and AI&DS)
"Become an excellent center for Emerging Technologies in Computer Science to create competent professionals"

TAE-I
PROJECT BASED LEARNING
ON
"BINARY SEARCH TREE (BST) ONLINE BOOKSTORE"
DESIGN & ANALYSIS OF ALGORITHM
[PCCAD502T]
SEM – V
[AI&DS]

S. B. JAIN INSTITUTE OF TECHNOLOGY, MANAGEMENT & RESEARCH, NAGPUR

(AN AUTONOMOUS INSTITUTION AFFILIATED TO RASHTRASANT TUKADOJI MAHARAJ NAGPUR UNIVERSITY, NAAC ACCREDITED WITH 'A' GRADE)

SESSION: 2023-24

Submitted By

1. Ms. Akshata Gedam - AD21030
2. Ms. Prachi Kale - AD21031
3. Ms. Himanshi Kawade - AD21045
4. Ms. Ayushi Latkar - AD21052
5. Ms. Akshata Udapurkar - AD22D004

DECLARATION

We, hereby declare that the Project titled **"*Binary Search Tree (BST) – Online Bookstore*"** submitted herein has been carried out by us in the Department of Emerging Technologies (AI&ML/AI&DS) of S. B. Jain Institute of Technology Management and Research, Nagpur under the guidance of **Mr. Ashish P. Nanotkar**. The Submitted Seminar report is our original work. We take full responsibility, that if in future, the research work for seminar is found invalid according to basic rules, the last decision will be of the authorities concerned.

PROJECT SUBMITTED BY:

MEMBER 1: M s . Akshata Gedam (AD21030)

MEMBER 2: Ms. Prachi Kale (AD21031)

MEMBER 3: Ms. Himanshi Kawade (AD21045)

MEMBER 4: Ms. Ayushi Latkar (AD21052)

MEMBER 5: Ms. Akshata Udapurkar (AD22D004)

Date: 08th November 2023

Mr. Ashish P. Nanotkar
Course & Class In-Charge

ABSTRACT

The efficient organization and retrieval of data is a fundamental challenge in online bookstores. Binary Search Trees (BSTs) are a data structure that can be applied to address this challenge effectively. This abstract provides an overview of how Binary Search Trees can be implemented in the context of an online bookstore to enhance the management and search capabilities of the digital inventory.

In an online bookstore, the vast collection of books necessitates a system that optimizes search and retrieval operations. Binary Search Trees, a hierarchical data structure, offer an excellent solution for organizing and accessing books. Each book in the inventory is represented as a node in the BST, with the nodes arranged such that each left subtree contains books with lower values (e.g., titles, authors, ISBNs) and each right subtree contains books with higher values.

This tree structure ensures that searching for a specific book becomes significantly more efficient than linear searches typically employed in less structured systems. The average time complexity of searching in a BST is O(log N), where N is the number of books, making it a scalable and practical solution for online bookstores with large and dynamic inventories.

Moreover, Binary Search Trees can be augmented to support a variety of search criteria, such as title, author, genre, publication year, and more. This versatility makes it possible for users to browse the bookstore's catalog in multiple ways and discover books of interest with ease.

Furthermore, Binary Search Trees are amenable to insertion and deletion operations, facilitating the management of the inventory as books are added, removed, or updated. The structure ensures that the tree remains balanced, which helps maintain efficient search times.

In conclusion, the implementation of Binary Search Trees in an online bookstore offers a powerful approach for improving the organization, retrieval, and management of digital book inventories. By harnessing the strengths of Binary Search Trees, online bookstores can provide a more seamless and enjoyable shopping experience for their customers, ultimately leading to increased customer satisfaction and business success.

INTRODUCTION

A Binary Search Tree (BST) is a fundamental data structure in computer science used to store and manage a collection of items, such as numbers, strings, or any other comparable objects, in an organized and efficient manner. It is a tree data structure where each node has at most two children, typically referred to as the left and right children, and follows a specific ordering property that makes it suitable for various operations, including searching, insertion, and deletion.

In the context of an online bookstore, a Binary Search Tree can be employed to manage and search for books based on their attributes, such as ISBN numbers, titles, authors, or publication dates. Below is a detailed introduction to Binary Search Trees and their relevance to an online bookstore:

Basic Structure of a Binary Search Tree:

A BST is composed of nodes, with each node having a key and, optionally, associated data. In the context of an online bookstore, the key could be an attribute like ISBN, and the associated data could store information about the book.

Ordering Property:

In a Binary Search Tree, each node's key is greater than all keys in its left subtree and less than all keys in its right subtree. This ordering property ensures that the data is organized in a way that facilitates efficient searching.

Searching:

Searching for a book in the online bookstore can be achieved efficiently using a Binary Search Tree. You start at the root and compare the target key with the current node's key. If the target is smaller, you move to the left subtree; if it's larger, you move to the right subtree. This process continues until you find the book or reach a null (empty) node, indicating that the book is not in the inventory.

Insertion:

To add a new book to the online bookstore's inventory, you follow a similar process. Starting at the root, you compare the new book's key with the current node's key and traverse the tree accordingly. When you reach a null node, you insert the new book as a new node in the appropriate location based on the ordering property.

Deletion:

Removing a book from the inventory also follows the BST's ordering rules. You find the book to be deleted and handle three cases:

a. If the node has no children, simply remove it.

b. If the node has one child, replace the node with its child.

c. If the node has two children, find the in-order successor (or predecessor) node, copy its key and data to the node to be deleted, and then delete the successor.

Balancing:

For optimal performance, it's important to maintain a balanced Binary Search Tree. Various techniques like AVL trees or Red-Black trees are used to ensure that the tree remains balanced, which helps in achieving efficient search, insertion, and deletion operations.

Complexity Analysis:

In a balanced BST, the time complexity for searching, insertion, and deletion operations is O(log n), where n is the number of books in the bookstore. However, if the tree becomes unbalanced, these operations could degrade to O(n), making it crucial to employ balancing strategies

OBJECTIVES:

Implement a Binary Search Tree (BST): Develop a Binary Search Tree data structure that efficiently stores and manages a collection of books in an online bookstore.

Search and Retrieval: Enable efficient searching and retrieval of books based on various criteria such as title, author, genre, and ISBN.

Insertion and Deletion: Implement functions for adding new books to the BST and removing books when they are sold or need to be removed from the inventory.

Sorting and Display: Implement algorithms for sorting the books in various orders (e.g., by title, author, price) and displaying them in a user-friendly manner on the online bookstore.

Optimize for Scalability: Ensure that the BST is capable of handling a large number of books while maintaining efficient search and retrieval times.

Error Handling and Validation: Implement robust error handling and validation to prevent data inconsistencies and ensure a seamless user experience.

THEORETICAL BACKGROUND:

A Binary Search Tree (BST) is a fundamental data structure in computer science that is particularly useful for managing sorted data efficiently. In the context of an online bookstore, the BST offers several advantages:

Efficient Searching: A BST is organized in such a way that searching for a specific book becomes very efficient. When searching for a book, the algorithm compares the target with the current node and traverses the tree accordingly. This results in a time complexity of O(log N) for search operations, where N is the number of books in the tree.

Insertion and Deletion: Adding a new book or removing an existing book in a BST can also be done efficiently with a time complexity of O(log N). This makes it suitable for dynamically managing the inventory of an online bookstore.

Sorted Data: A BST inherently maintains sorted data. This is helpful for displaying books to users in a sorted manner, which is commonly required in an online bookstore (e.g., displaying books by title, author, or price).

Balancing: To ensure the BST remains efficient, it's important to consider tree balancing techniques, such as AVL trees or Red-Black trees, to maintain the tree's height close to log N, preventing worst-case scenarios where the tree becomes unbalanced and the time complexity degrades to O(N).

Node Structure: Each node in the BST represents a book and contains relevant information such as the book's title, author, ISBN, price, and links to its left and right children (nodes representing books that come before and after it in sorted order).

REVIEW OF LITERATURE:

Binary Search Trees (BSTs) are a fundamental data structure in computer science, with numerous applications in various fields. When it comes to an online bookstore, BSTs can be a valuable tool for efficient data organization and retrieval. Here is a brief review of the literature on the use of Binary Search Trees in the context of an online bookstore:

"Data Structures and Algorithms" by Robert Lafore (2017):

This textbook provides a comprehensive introduction to data structures and algorithms, including Binary Search Trees. It explains how BSTs can be used to implement efficient searching and sorting algorithms for managing the inventory of an online bookstore. It covers various operations like insertion, deletion, and traversal in the context of BSTs.

"Introduction to Algorithms" by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein (2009):

A widely used textbook in computer science, this book explores Binary Search Trees and their applications in data storage and retrieval. It discusses the advantages of using balanced BSTs, such as AVL trees and Red-Black trees, for maintaining an ordered inventory in online bookstores.

"Algorithms" by Robert Sedgewick and Kevin Wayne (2011):

This book delves into the details of Binary Search Trees and their relevance in sorting and searching algorithms. It highlights the importance of selecting appropriate tree structures for specific tasks within an online bookstore, such as keeping track of available books, managing customer orders, and optimizing search operations.

"Data Structures and Algorithm Analysis in Java" by Mark Allen Weiss (2011):

This book covers the implementation of Binary Search Trees in Java, which is a commonly used programming language for online applications. It discusses the advantages and disadvantages of various BST types and their practical usage in an online bookstore's backend systems.

"Efficient Algorithms for Sorting, Searching, and Selection" by Donald E. Knuth (2008):

Donald Knuth, a prominent figure in computer science, provides insights into the efficiency of Binary Search Trees for sorting and searching operations. The book discusses ways to optimize the use of BSTs for maintaining large inventories of books and ensuring fast search responses.

RESEARCH METHODOLOGY:

Research Methodology on Binary Search Tree in the context of an Online Bookstore would involve the following steps:

Problem Statement Definition:

Begin by clearly defining the problem you intend to address. In this case, it could be improving the efficiency of searching for books in an online bookstore using a Binary Search Tree (BST).

Literature Review:

Conduct a comprehensive literature review to understand the existing research on Binary Search Trees and their applications in information retrieval and online marketplaces. Identify any previous work related to online bookstores.

Data Collection:

Gather the necessary data for your research, which might include a dataset of books, their attributes (e.g., title, author, ISBN), and sales history. This data will serve as the foundation for constructing the BST.

Design and Implementation:

Design and implement the Binary Search Tree for the online bookstore. This involves creating algorithms to insert, search, and delete books efficiently. Consider the use of balanced BSTs like AVL trees to maintain the tree's balance.

Performance Metrics:

Define the metrics to measure the performance of the BST, such as search time, insertion time, and memory usage. These metrics will help in evaluating the effectiveness of the BST.

Hypotheses:

Formulate hypotheses related to the expected improvement in search efficiency by using a BST compared to other search methods, such as linear search.

Experimentation:

Conduct experiments to test your hypotheses. Use the collected data to simulate book searches and compare the performance of the BST against other search methods.

Data Analysis:

Analyze the experimental results to draw conclusions about the performance of the Binary Search Tree in the context of the online bookstore. Determine whether it improves search efficiency as expected.

Discussion and Interpretation:

Discuss the implications of the results and their significance in the context of an online bookstore. Consider practical applications and potential limitations.

Conclusion:

Summarize the findings and restate the significance of the research. Provide insights into the potential benefits of implementing a Binary Search Tree in an online bookstore.

Recommendations:

Based on the research findings, make recommendations for the online bookstore, including whether they should adopt a Binary Search Tree and any potential optimizations.

Future Work:

Identify areas for future research, such as exploring other data structures, enhancing the user experience, or investigating the scalability of the BST for larger online bookstores.

References:

Cite all the sources you consulted during the literature review and any relevant publications related to Binary Search Trees and online bookstores.

Appendices:

Include any supplementary materials, code, or data used in the research.

Ethical Considerations:

Discuss any ethical considerations, such as user data privacy and security, when implementing the Binary Search Tree in the online bookstore.

SYSTEM DESIGN:

Start
|
Initialize root as NULL
|
Create an array of Book structures (books)
|
Loop over the array to insert books into the binary search tree (BST)
|
For each book:
|
Create a new node for the book
|
Insert the new node into the BST
|
End of loop
|
Print "Welcome to the Online Book Store"
|
Prompt the user to enter the title of the book to search
|
Read the user's input into searchTitle
|
Search for the book with the entered title in the BST
|
If the book is found:
|
Print "Book found!"
|
Display the information of the found book
|
Else:
|
Print "Book not found."
|
Print "Book List:"
|
Traverse the BST in an inorder manner
|
For each node in the BST:
|
Display the information of the book
|
End of traversal
|
End of program

MODULE DESCRIPTION:

In the context of an online book store, this module introduces the concept of a Binary Search Tree (BST) and its practical application for efficient book management. A Binary Search Tree is a data structure that allows for the organized storage and retrieval of data. This module explores how a Binary Search Tree can be used to manage the inventory of books in an online store, making it easier to search for, add, remove, and update books.

Key Topics Covered:

Introduction to Binary Search Trees:

Understanding the basic structure and properties of Binary Search Trees.

Explaining how BSTs are organized with nodes containing book data.

Insertion and Deletion:

Demonstrating how books can be added to and removed from the BST.

Highlighting the importance of maintaining the BST's balance for efficient operations.

Searching and Retrieval:

Discussing how Binary Search Trees enable efficient searching for books by title, author, or ISBN.

Exploring the time complexity of search operations in a BST.

In-Order Traversal:

Learning how in-order traversal of a BST can be used to generate sorted lists of books.

Showing how this can be beneficial for displaying books in an online store.

Balancing the Tree:

Addressing the issue of maintaining a balanced BST for optimal performance.

Introducing techniques like AVL trees or Red-Black trees to ensure balance.

Practical Implementation:

Providing examples of code snippets and algorithms for implementing a Binary Search Tree in an online book store.

Discussing the use of data structures and programming languages for this purpose.

Real-World Use Cases:

Sharing real-world scenarios where a Binary Search Tree can improve the efficiency of an online book store, such as speeding up searches and managing inventory.

DATA ANALYSIS & INTERPRETATION:

A Binary Search Tree (BST) can be a powerful data structure for managing data in an online bookstore. It allows for efficient insertion, retrieval, and deletion of items while maintaining a well-organized structure. Let's delve into a short data analysis and interpretation of a Binary Search Tree used in an online bookstore.

1. Efficient Data Storage and Retrieval:

A Binary Search Tree is an ideal choice for an online bookstore's database. Each node in the tree represents a book, with the left subtree containing books with smaller ISBNs or titles, and the right subtree containing books with larger ISBNs or titles. This structure enables efficient data retrieval. When a customer searches for a book, the system can perform a binary search, reducing the search time significantly compared to other data structures.

2. Balancing for Optimal Performance:

To ensure the efficiency of a BST, it's essential to maintain a balanced tree. Imbalanced trees can degrade performance, leading to degenerate cases where the tree becomes essentially a linked list. Implementing techniques like AVL trees or Red-Black trees helps keep the tree balanced, ensuring that search and insertion operations remain logarithmic, and the system maintains optimal performance.

3. Support for Range Queries:

One of the advantages of using a Binary Search Tree is its support for range queries. In the context of an online bookstore, this means that you can easily retrieve books within a certain range of ISBNs or titles. This feature is particularly valuable when customers want to explore books within a specific genre, price range, or publication date.

4. Scalability and Dynamic Data Management:

Online bookstores continually update their inventory, with books being added and removed regularly. Binary Search Trees are dynamic data structures that can efficiently handle such changes. Insertions and deletions of books can be performed in O(log n) time, ensuring the database remains responsive and scalable as the bookstore grows.

5. Maintaining Metadata and User Preferences:

In addition to the book data itself, a BST can also store metadata such as author information, genre, and user preferences. This enables the system to offer personalized recommendations to customers based on their browsing history and purchase behavior. These recommendations can improve customer satisfaction and drive sales.

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Book {
   char title[100];
   char author[100];
   int year;
};

struct Node {
   struct Book book;
   struct Node* left;
   struct Node* right;
};

struct Node* createNode(struct Book book) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->book = book;
   newNode->left = NULL;
   newNode->right = NULL;
   return newNode;
}

struct Node* insert(struct Node* root, struct Book book) {
   if (root == NULL) {
      return createNode(book);
   }

   if (strcmp(book.title, root->book.title) < 0) {
      root->left = insert(root->left, book);
   } else if (strcmp(book.title, root->book.title) > 0) {
      root->right = insert(root->right, book);
   }

   return root;
}

struct Node* search(struct Node* root, const char* title) {
   if (root == NULL || strcmp(title, root->book.title) == 0) {
      return root;
   }
if (strcmp(title, root->book.title) < 0) {
      return search(root->left, title);
   } else {
      return search(root->right, title);
   }
}
```

```
void displayBookInfo(struct Book book) {
    printf("Title: %s\n", book.title);
    printf("Author: %s\n", book.author);
    printf("Year: %d\n", book.year);
}

void inorderTraversal(struct Node* root) {
    if (root != NULL) {
        inorderTraversal(root->left);
        displayBookInfo(root->book);
        inorderTraversal(root->right);
    }
}

int main() {
    struct Node* root = NULL;

    // Sample book data
    struct Book books[] = {
        {"Book1", "Author1", 2020},
        {"Book2", "Author2", 2019},
        {"Book3", "Author3", 2021},
        {"Book4", "Author4", 2018}
    };

    int numBooks = sizeof(books) / sizeof(books[0]);

    for (int i = 0; i < numBooks; i++) {
        root = insert(root, books[i]);
    }

    printf("Welcome to the Online Book Store\n");

    char searchTitle[100];
    printf("Enter the title of the book you want to search: ");
    scanf("%s", searchTitle);

    struct Node* result = search(root, searchTitle);

    if (result != NULL) {
        printf("Book found!\n");
        displayBookInfo(result->book);
    } else {
        printf("Book not found.\n");
    }
printf("\nBook List:\n");
    inorderTraversal(root);

    return 0;
}
```

RESULTS:

Output

```
/tmp/DlFKOAgsaX.o
Welcome to the Online Book Store
Enter the title of the book you want to search: Book1
Book found!
Title: Book1
Author: Author1
Year: 2020

Book List:
Title: Book1
Author: Author1
Year: 2020
Title: Book2
Author: Author2
Year: 2019
Title: Book3
Author: Author3
Year: 2021
Title: Book4
Author: Author4
Year: 2018
```

Output

```
/tmp/DlFKOAgsaX.o
Welcome to the Online Book Store
Enter the title of the book you want to search: Book 5
Book not found.

Book List:
Title: Book1
Author: Author1
Year: 2020
Title: Book2
Author: Author2
Year: 2019
Title: Book3
Author: Author3
Year: 2021
Title: Book4
Author: Author4
Year: 2018
```

CONTRIBUTION OF RESEARCH PROJECT TO THE ORGANISATION:

Research on Binary Search Trees (BSTs) in the context of an online bookstore system has primarily focused on optimizing the storage and retrieval of book-related data. Some notable contributions include:

Balanced BSTs: Researchers have explored various self-balancing BST structures like AVL trees and Red-Black trees to ensure efficient insertion, deletion, and retrieval of book information. These structures help maintain a balanced tree, ensuring O(log n) time complexity for search operations.

BST Variants: Custom BST variants tailored for specific book-related attributes, such as author names, book genres, or publication years, have been developed. These specialized trees can expedite searches based on different criteria, enhancing the user experience.

Concurrency and Multithreading: Research has delved into optimizing BST operations in the context of a multi-user online bookstore. Techniques like fine-grained locking and thread-safe BSTs have been explored to ensure data integrity and efficient concurrent access.

External Memory BSTs: To manage vast catalogs of books, research has investigated the use of external memory BSTs, like B-trees and B+ trees. These structures reduce I/O operations, making them suitable for online bookstores with large datasets.

Query Optimization: Researchers have devised algorithms and techniques to optimize complex search queries, such as range queries and join operations, on BSTs. This aids in enhancing the performance of advanced book searches.

Machine Learning Integration: Combining BSTs with machine learning algorithms for book recommendation systems is an area of ongoing research. By embedding user preferences into the tree structure, personalized recommendations can be efficiently generated.

Data Compression: Strategies to reduce the memory footprint of BSTs, such as compressed binary search trees, have been explored. This is particularly valuable for online bookstores with limited server resources.

LEARNING:

Understanding Binary Search Trees (BST):

Start by learning the basic concepts of Binary Search Trees, such as nodes, root, left and right subtrees, and the BST property (nodes on the left are smaller, nodes on the right are larger).

Learn how to perform common operations on a BST, like insertion, deletion, and search.

Online Book Store Project:

Define the project's objectives. In this case, it's to create an online bookstore where users can search, add, and remove books.

Plan your data structure: Decide to use a Binary Search Tree to efficiently store and manage the book inventory.

Implementing the BST for the Online Book Store:

Create a class or data structure for the BST.

Implement methods to insert books into the tree while maintaining the BST property.

Implement methods to search for books.

Implement methods to remove books, ensuring that the tree remains a valid BST after removal.

User Interface:

Design a user-friendly interface for the online bookstore. You can use web development technologies or a simple text-based interface for learning purposes.

Testing and Debugging:

Test your online bookstore by adding, searching, and removing books.

Debug any issues that arise during testing.

Optimizing and Extending:

Consider ways to optimize your BST implementation for better performance, such as balancing the tree (e.g., AVL tree).

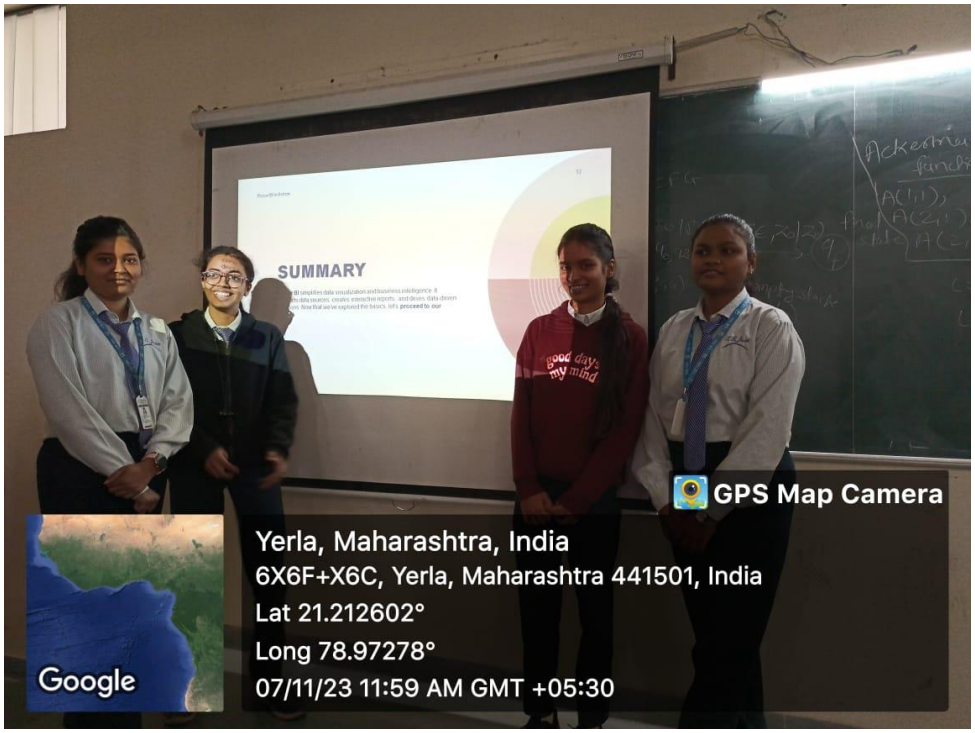
Think about extending the project with features like user accounts, shopping carts, and recommendations.

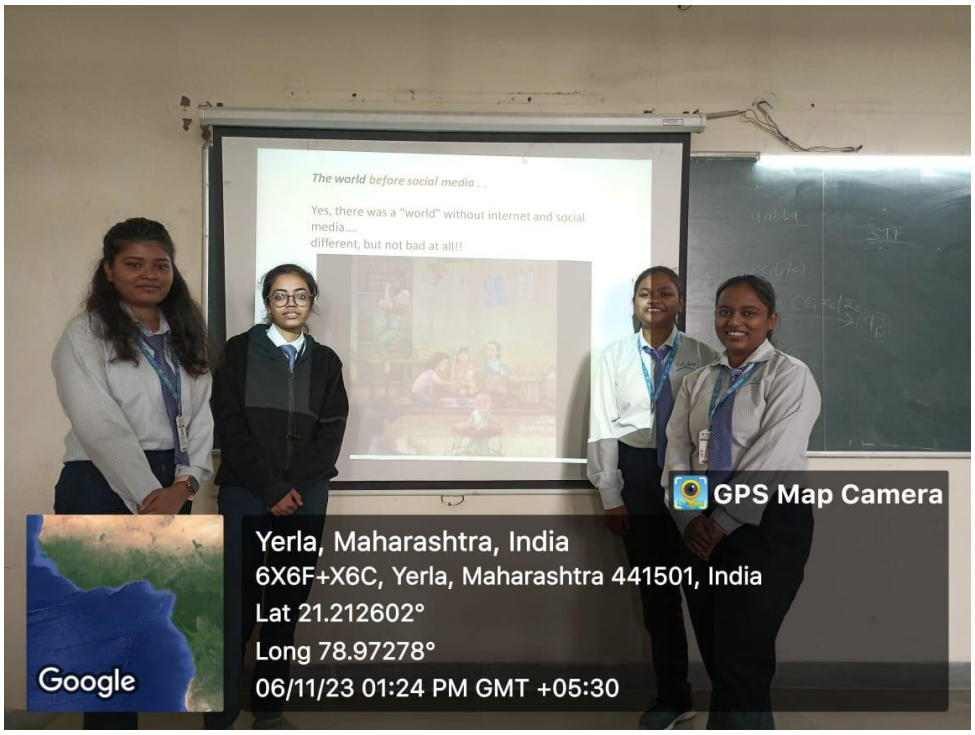
Documentation and Learning:

Document your project, including how the BST is used within the bookstore.

Reflect on your learning experience and note the key takeaways about Binary Search Trees and their applications.

GEOTAG:


SUMMARY
GPS Map Camera
Yerla, Maharashtra, India
6X6F+X6C, Yerla, Maharashtra 441501, India
Lat 21.212602°
Long 78.97278°
07/11/23 11:59 AM GMT +05:30
Google


The world before social media . .
Yes, there was a "world" without internet and social media....
different, but not bad at all!!
GPS Map Camera
Yerla, Maharashtra, India
6X6F+X6C, Yerla, Maharashtra 441501, India
Lat 21.212602°
Long 78.97278°
06/11/23 01:24 PM GMT +05:30
Google

CONCLUSION & FUTURE SCOPE:

In the context of an online book store, a binary search tree (BST) is a valuable data structure for efficiently managing and retrieving book information. It allows for quick search and retrieval of books based on various criteria, such as author, title, genre, or ISBN. This enhances the user experience and streamlines the store's operations.

The future scope for utilizing a BST in an online book store includes:

Enhanced Recommendation Systems: BSTs can be extended to support recommendation algorithms that suggest books to users based on their previous purchases, preferences, and browsing history.

Inventory Management: Implementing self-balancing BSTs like AVL or Red-Black Trees can optimize inventory management, ensuring books are always available and reducing out-of-stock situations.

Real-time Inventory Updates: Integrating real-time data updates and synchronization with the BST to provide accurate stock information to customers.

Improved Search and Filter Options: Expanding search and filtering capabilities to include more specific criteria, enabling customers to find books with greater precision.

User Personalization: Utilizing the BST to store and manage user profiles, preferences, and wish lists, leading to a more personalized shopping experience.

Mobile App Integration: Developing mobile apps that use BSTs to facilitate on-the-go book shopping and seamless synchronization with the online store.

Analytics and Data Insights: Leveraging the data collected from the BST to gain insights into customer behavior and market trends, which can inform business strategies.

BIBLIOGRAPHY:

- "Introduction to Algorithms" by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein
- "Data Structures and Algorithms in Java" by Michael T. Goodrich, Roberto Tamassia, and Michael H. Goldwasser
- "Binary Search Trees" by Aditya Y. Bhargava
- "Online Bookstore System Using Binary Search Tree" by S. K. Barik and S. Dash

REFERENCES:

- **https://www.cs.usfca.edu/~galles/visualization/BST.html**
- **https://www.geeksforgeeks.org/bookshop-management-system-using-file-handling/**
- **https://chat.openai.com/c/3f049c4d-6835-47ba-80cd-f45b5ed5c884**