Administrative

Team Name: BookWorms #144
Team Members + GitHub Usernames:

- Yuvika Shendge: YuviShendge
- Marcos Laserna: MarcosLaserna
- Aiden Everage: aideneverageuf

Link to GitHub Repo: BookWormsProject
Link to Video Demo: _____

Extended and Refined Proposal

Problem

The main problem we're trying to solve is figuring out which data structure works best for recommending books based on what a user has already liked. With so many books out there, it's hard for users to find new ones they'll enjoy. By testing different data structures, specifically graphs and hash maps, we want to see which one gives faster and more accurate recommendations.

Motivation

This is a problem because people often get stuck scrolling through endless lists of books that don't match their interests. Good recommendation systems, like the ones used by Goodreads, need to handle tons of data quickly and still give personalized results. If the system is too slow or inaccurate, it doesn't really help the user. That's why it's important to figure out which data structure is better at storing and retrieving data to make these recommendations as efficient as possible.

Features Implemented

- **Add Liked Books**: Users can input books they've already enjoyed to build their profile.
- **Get Recommendations**: Based on their input, the system suggests books that they might like next.

Description of Data

The data we're using includes basic book info, like titles, authors, genres, average ratings, and page counts. We have user preference data, which tracks which books people liked

and how they rated them. Together, these datasets let us connect users and books, creating relationships that help with generating recommendations.

Tools/Languages/APIs/Libraries Used

- **Python**: for writing the algorithms and working with the data.
- **Jupyter Notebook**: for writing the algorithms and working with the data.
- **React and JavaScript**: for the user interface.
- **Flask**: to connect the frontend and backend.
- **Pandas**: for handling and analyzing the data.

Algorithms Implemented

1. **Graph**: We created a graph where users and books are nodes, and edges show which books a user liked. By analyzing the graph, we recommend books that similar users enjoyed.
2. **Hashmap**: We used hash maps to quickly store and retrieve user preferences to let us match users to books based on similar data.

Additional Data Structures/Algorithms Used
No additional data structures or algorithms

Distribution of Responsibility and Roles

- **Yuvika**:
    - Worked on the hashmap algorithm.
    - Wrote the extended and refined Proposal portion of the report.
- **Aiden**:
    - Built the frontend using React and connected it to the backend with Flask.
    - Created the video showing off the project.
- **Marcos**:
    - Worked on the graph-based recommendation algorithm.
    - Worked on the Analysis portion of the report.

Analysis

Any changes the group made after the proposal? The rationale behind the changes.
We decided to focus on certain properties of each book, such as the number of pages, author relationships, and general ratings, rather than using every property available. This choice was made to reduce computational cost, which would have been much higher if all attributes were considered.

For the graph-based approach, we incorporated techniques to relate users who share similar liked books to strengthen the recommendation process. In the hash map implementation, we prioritized global book ratings instead of trying to store user relationships, as doing so in hash maps would have been overly complex and inefficient. While the hash map implementation offers faster response times compared to the graph approach, the graph approach may be slightly more precise due to its ability to store richer features and relationships.

Big O Worst-Case Time Complexity Analysis of the Major Functions/Features Implemented

Graph-Based Approach
We analyze each function separately:

- **File Reading and Graph Construction**: The complexity is O(n + m), where n is the number of books and m is the number of users. Reading the book file requires O(n), and adding nodes to the graph for books and users contributes an additional O(m).
- **get_liked_books()**: This function has a complexity of O(k), where k is the number of books input by the user.
- **calculate_bonus(book_data, liked_books_df)**: This function has a time complexity of O(k * n), as it iterates through each liked book's author and then checks all books in the dataset to find matches. Here, k is the number of liked books, and n is the total number of books.
- **find_users(liked_books_df)**: This function iterates through all users and checks their liked books, resulting in a complexity of O(k * m), where k is the number of liked books, and m is the total number of users.
- **recommend_books_for_user(user_id, ranked_users, liked_books_df, G, books)**: The complexity is the sum of the previous functions, with an additional sorting step that requires O(m log m) time in Python. Therefore, the total complexity is O(m log m) + O(n + m) + O(k * m) + O(k * n), which simplifies to O(m log m) + O(k * (n + m)).
- **display_recommendations(user_id, recommended_books)**: The complexity is O(1) since it only displays the top 5 recommendations.

Global Complexity (Graph)
In the worst case, the global complexity is O(m log m) + O(k * (n + m)). Assuming k (the number of input liked books) is relatively small and bounded, the overall worst-case complexity simplifies to O(n + m log m), where n is the total number of books, and m is the total number of users.

Hash Map-Based Approach

- **File Reading and Data Storage**: Reading the dataset has a complexity of O(n), where n is the total number of books. Storing user preferences and iterating over all books for each user results in a complexity of O(n * m), where m is the number of users.
- **Checking the Hash Map**: Each lookup in the hash map has O(1) complexity, with a possible total of O(n) lookups, where n is the number of books.

Global Complexity (Hash Map)
The overall worst-case complexity for the hash map approach is O(n * m) + O(n) + O(1), which simplifies to O(n * m), where n is the total number of books, and m is the number of users.

Reflection

Overall Experience
The project experience overall was positive. We were able to divide the work effectively among group members, and each person contributed to their assigned tasks. It was satisfying to see the project come together at the end, but the process itself was more challenging than we initially expected.

Challenges
One of the main challenges was realizing how much more complex our problem was compared to what we envisioned during the proposal phase. Implementing and optimizing the algorithms to ensure they performed well, especially with the graph-based recommendation system, took more effort than anticipated. Additionally, integrating the different components of the project towards the end required some coordination, and it was difficult to manage everything with limited time.

Potential Changes
If we were to start the project over, we would definitely begin earlier to avoid the time crunch we experienced at the end. Starting sooner would have allowed us to refine our algorithms more and test our systems more thoroughly. It also might have given us room to explore additional features or improvements.

What Each Member Learned

- **Marcos**: I learned about how graphs can be used to model relationships in a dataset and how these relationships can be leveraged for recommendations. Seeing the entire process was valuable. I also found the frontend work interesting and might explore web development in the future.

- **Aiden**: I improved my knowledge of JavaScript and React while working on the frontend. I also learned how to connect the frontend and backend using Flask and Python, which gave me a better understanding of how full-stack applications function.
- **Yuvika**: I focused on implementing hash maps in Python and learned how they can be used to efficiently handle recommendation-related tasks. This project reinforced how important algorithmic efficiency is, especially when working with large datasets. I also developed better teamwork and organizational skills, as balancing group collaboration with technical tasks required clear communication.

In summary, the project had its challenges, but it was a learning opportunity for all of us. We each gained a better understanding of the strengths and limitations of different data structures and how to apply them to real-world problems. While there are things we would do differently, the experience was worthwhile.