Aysa Binte Masud

Instructor Khalid Mengel

Write up about design choices

12 December 2023

Introduction

The Hotel Finder tool is made to make looking for and handling hotel information in a database easy for everyone. The paper talks about the design decisions that were made while the program was being made, focused on important parts like data structures, and how well the program works.

Data Structures

One of the most important decisions that went into making the Hotel Finder program was choosing the right data structures to store and organize hotel records. Hash Tables and Binary Search Trees (BST) were the main types of data structures used.

Binary Search Trees (BST):

- BSTs were picked so that they could store and look for hotel records quickly by name and place. The BST is organized so that hotels are listed in order by their names and cities. Each node in the BST is a hotel record.

- This was done with a BST because it was important to be able to quickly look for and get hotel records by name and place. For search tasks, BSTs take an average of O(log N) time, which means they can handle this job.
- BSTs were also used to make the findAll process possible, which quickly gets back all the hotels in a certain place. The tiered structure of BSTs helps this process, which is an important part of the program.

Hash Tables:

To store hotel data based on a mix of hotel name and place, hash tables were used. This data format makes it easy to quickly find and access hotel records by their unique IDs.

Hashing functions were carefully picked to make sure that there were as few collisions as possible and that records were spread out evenly across bins. Chaining, in which each box has a linked list of items, was used to handle collisions.

Adding and removing hotel records is done with hash tables. They give these actions an average time complexity of O(1), which makes them good for handling the database.

Efficiency Considerations

Because the Hotel Finder tool could be used with big files, efficiency was the most important thing that went into making it. Several plans were put in place to make sure that data operations ran smoothly.

BST Balancing:

- The program uses AVL tree balance to keep the effectiveness of BSTs high. This self-balancing feature keeps BSTs roughly balanced, which stops trees from breaking down into linked lists, which is the worst thing that could happen.
- When records are added or removed, they are balanced to make sure that the search and findAll processes take the same amount of time.

Hash Table Size:

- The hash table's size was carefully picked to keep clashes to a minimum while making the best use of memory. A prime number was chosen as the hash table's ability to spread out data evenly and stop it from bunching up.

Conclusion

When I designed the Hotel Finder program, the main goals were to make it efficient, easy to use, and durable. The program uses a mix of data structures like BSTs and hash tables, an easy-to-understand command-line interface, and a focus on speed to make a reliable and user-friendly tool for finding and handling hotel records. Because of these design choices, the Hotel Finder app is easy for users to work with the database, which makes it useful for managing hotels and searching for them.