Kaleb McCracken

CS 300

Professor Andujar

17 February 2024

Project One

For the vector data structure, the runtime analysis indicates that reading the file and creating course objects have a time complexity of O(n), where n is the number of courses stored in the file. Memory requirements are also proportional to the number of lines in the file since each course object holds data from a single line. Vectors offer simplicity and ease of understanding, making them suitable for small to medium-sized datasets. However, they may become inefficient for frequent insertions and deletions, particularly in larger datasets, due to their linear time complexity for these operations.

Concerning hash tables, the runtime analysis shows that reading the file and creating course objects also have a time complexity of O(n). Memory requirements follow the same pattern as vectors, being proportional to the number of courses. Hash tables offer constant time complexity for insertion, deletion, and retrieval on average, making them efficient for large datasets. However, hash collisions may occur, affecting performance, and ensuring a good hash function is crucial for uniform key distribution.

In the case of trees, the runtime analysis indicates a time complexity of O(n log n) for creating course objects, assuming a balanced tree, while memory requirements depend on the structure of the tree and the number of nodes. Trees provide efficient searching, insertion, and deletion operations, especially for large datasets. They guarantee logarithmic time complexity for these operations if balanced and maintain data in sorted order, facilitating alphabetical listing of courses. However, trees are more complex than vectors and hash tables, requiring balancing for optimal performance and may not be suitable for very large datasets due to memory overhead.

Based on the requirements of printing a list of courses in alphanumeric order and efficiently retrieving course information, a balanced tree data structure is recommended. It offers efficient search operations and maintains sorted order, aligning well with the need to print courses alphabetically and quickly retrieve course information. However, if simplicity and ease of implementation are prioritized, a hash table can also be a viable choice, offering efficient retrieval operations.