**Evaluation Final**

Ikechukwu Onuoha

Southern New Hampshire University

CS-300-X6108 DSA

Dr. James O. Webb, Jr

August 6, 2023

**Pseudocode**

Let n be courses stored in the data structure and consider the cost per line of code to be 1, unless a function is called, in which case the cost will be the running time of that function.

1. Pseudocode for Vector Data Structure:

Worst-case Big O Value: O(n^2)

In the pseudocode, the vector data structure is used to store courses. For each line in the file, a course object is created, and its prerequisites are validated. The nested FOR loop inside the loading function iterates through all the courses in the vector to validate prerequisites. This results in a worst-case time complexity of O(n^2) because for each course we iterate through all the courses to check prerequisites.

Advantages:

It is simple, easy to use and memory-efficient for smaller datasets.

Disadvantages:

Slow for large datasets due to the nested FOR loop for prerequisites validation

2. Pseudocode for Hash Table Data Structure:

Worst-case Big O Value: O(n^2)

In the pseudocode, the hash table data structure is used to store courses. The worst-case time complexity occurs when there are hash collisions, leading to a linear search through the hash table. For each line in the file, a course object is created, and its prerequisites are validated by searching the hash table. In the worst case, the search operation for prerequisites becomes O(n)

Advantages:

Fast average-case constant-time retrieval for courses (O(1)) and Ideal for direct access to course objects using unique course numbers.

Disadvantages:

Worst-case time complexity of O(n^2) due to collisions and linear search.

3. Pseudocode for Binary Tree Data Structure:

Worst-case Big O Value: O(n log n)

In the pseudocode, the binary tree data structure is used to store courses. The worst-case time complexity of the binary tree occurs when the tree becomes unbalanced. For each line in the file, a course object is created and inserted into the binary tree. In the worst case, the binary tree could become a linear linked list, leading to a time complexity of O(n).

Advantages:

Efficient searching with an average-case time complexity of O(log n) for a balanced tree and maintains the courses in sorted order based on their course ID.

Disadvantages:

Worst-case time complexity of O(n log n) for unbalanced trees.

Memory Usage:

In terms of memory usage, the vector data structure tends to be memory-efficient for smaller datasets due to its contiguous memory allocation. Hash tables and binary trees consume additional memory for their data structure overhead (hash table buckets or binary tree nodes). However, the difference in memory usage between these structures might not be significant for moderate-sized datasets.

Recommendation:

Based on the Big O analysis results and the evaluation of the three data structures, the hash table data structure seems to be the most appropriate choice for ABCU's advising program.

Justification:

The hash table offers fast average-case constant-time retrieval for courses, providing quick access to course information based on unique course numbers.

Although the worst-case time complexity for the hash table is O(n^2) due to collisions, its average-case performance is more appealing than the nested FOR loop of the vector and the potential unbalanced binary tree.

Hash tables handle collisions efficiently and can be optimized to minimize the likelihood of collisions.

The hash table's ability to handle large datasets with efficient average-case retrieval makes it a suitable choice for ABCU's advising program.

Overall, the hash table data structure provides a good balance between performance, ease of implementation, and memory efficiency for the specific requirements of ABCU's advising program. However, it's important to implement proper collision resolution strategies to further optimize the hash table's performance.