Sungho Yoon

Professor Lippincott

CS 300

13 April 2025

Project One Evaluation

A linked list provides a simple and efficient way to store course data during file reading and creation. Inserting a course object at the beginning of the list takes constant time, O(1), making it suitable for dynamic file reading operations. However, searching for a specific course requires traversal from the head to the matching node, which results in a worst-case lookup time of O(n). Sorting a linked list, required for displaying courses in alphanumeric order, is also less efficient unless a more complex algorithm like merge sort is used. The primary advantage of a linked list lies in its low memory overhead and ease of dynamic memory allocation. On the other hand, the disadvantage is its poor performance for searching and sorting, especially as the number of courses (n) increases.

The hash table offers excellent performance for both inserting and searching course objects. In the average case, both insertion and lookup operations are performed in O(1) time, which is highly efficient when reading from a file and retrieving course details based on course numbers. This makes it ideal for implementing the "Print Course Details" feature. However, since hash tables do not maintain any specific order, additional logic and temporary storage are required to sort course entries for display. The advantage of a hash table is its fast access time, making it suitable for large datasets where speed is crucial. Its disadvantage is that it requires extra memory for hash buckets and chaining, and sorting operations cannot be performed without auxiliary structures.

A binary search tree (BST), particularly when unbalanced, offers a mix of sorted structure and moderate performance. Inserting and searching in a BST typically take O(log n) in the average case, but can degrade to O(n) in the worst case when the tree becomes skewed. An in-order traversal naturally provides a sorted list of course objects in O(n) time, which directly supports features like displaying sorted course lists. The advantage of a BST is that it maintains sorted order without the need for additional sorting algorithms. However, its disadvantage is that performance may suffer if the tree is not self-balancing, such as an AVL or Red-Black Tree, and searching is slower than in a hash table.

In conclusion, while all three data structures can be used to implement course file reading and creation, the hash table provides the most efficient performance for fast insertions and lookups, which aligns well with the advisor program’s need to retrieve individual course details quickly. Despite its lack of inherent order, the use of a temporary structure to sort and display course data is a reasonable tradeoff. The linked list is simple and memory efficient but lacks search and sort performance. The BST maintains order but is more complex and slower for frequent lookups. Therefore, the recommended structure is the hash table, as it best meets the program's functional and performance requirements.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Data Structure** | **Insert Time** | **Search Time** | **Sorted Output** | **Memory Usage** | **File Load Time** |
| Linked List | O(1) | O(n) | O(n log n) | Low | O(n) |
| Hash Table | O(1) / O(n) worst case | O(1) / O(n) worst case | O(n log n) | Medium | O(n) |
| BST | O(log n) / O(n) worst case | O(log n) / O(n) worst case | O(n) inOrder | Medium | O(n log n) |