Evaluation

Runtime and Memory Analysis

Vector

1. reading the file and creating course objects:
   1. Opening the file: O(1)
   2. Reading each line: O(n)
   3. Parsing each line: O(n)
   4. Checking for formatting errors: O(n)
   5. Creating course objects and adding to vector: O(n)
   6. Validating prerequisites: O(n^2) in the worst case, if each line has multiple prerequisites that must be checked against all courses.

Total worst time complexity: O(n^2)

Memory Complexity:

Storage for course objects: O(n)

Storage for course list (vector): O(n)

Hash Table

1. Reading the file and creating course objects:
   1. Opening the file: O(1)
   2. Reading each line: O(n)
   3. Parsing each line: O(n)
   4. Checking for formatting errors: O(n)
   5. Creating course objects and adding to hash table: O(n)
   6. Validating prerequisites: O(n)

Total worst time complexity: O(n)

Memory complexity:

Storage for course objects: O(n)

Storage for course list (hash table): O(n)

Binary Search Tree

1. Reading the file and creating course objects:
   1. Opening the file: O(1)
   2. Reading each line: O(n)
   3. Parsing each line: O(n)
   4. Checking for formatting errors: O(n)
   5. Creating course objects and adding to BST: O(n log n)
   6. Validating prerequisites: O(n log n)

Total worst time complexity: O(n log n)

Memory complexity:

Storage for course objects: O(n)

Storage for BST: O(n)

Advantages and Disadvantages

Vector

Advantages: simple to implement, direct indexing allows fast access to elements by index.

Disadvantages: inefficient for frequent insertions and deletions due to shifting elements, searching for elements is O(n), which can be slow for large datasets.

Hash Table

Advantages: Fast average case time complexity for insertions, deletions, and lookups (O(1)), efficient handling of large datasets with minimal search times.

Disadvantages: potential for hash collisions which can degrade performance, may require resizing which can be computationally expensive.

Binary Search Tree

Advantages: provides sorted data which is useful for ordered operations, balanced trees performance for insertions, deletions, and lookups.

Disadvantages: more complex to implement than vectors and hash tables, unbalanced trees can degrade to O(n) performance.

Recommendations:

Based on the analysis, the hash table is recommended for this project. The has table provides the best average-case performance for insertions, deletions, and lookups, which aligns well with the requirements of loading course data and searching for course details efficiently. Additionally, hash tables handle large datasets effectively, making them suitable for the expected volume of course information. While vectors are simpler, their O(n) search time is inefficient for large datasets. Binary search trees offer ordered data but are more complex to implement and can suffer from performance issues if not kept balanced. Therefore, the hash table strikes an optimal balance between performance and implementation complexity for the given requirements.