**Project One  
Run-Time and Memory Analysis for Data Structures:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| VectorSort Pseudocode: |  |  |  |  |
| Code | Line Cost | # Times Executes | Total Cost | Runtime |
| for each course | 1 | n | n | O(n) |
| if course.courseNumber == courseNumber | 1 | n | n |  |
| print course information | 1 | 1 | 1 |  |
| for each prerequisite of the course | 1 | n | n |  |
| print prerequisite course information | 1 | n | n |  |
| Total Cost |  |  | 4n+1 | O(n) |

Vector

Advantages:

• Simple implementation

• Sequential memory access can result in good cache performance

• Suitable for small to medium-sized datasets

Disadvantages:

• Insertions and deletions in the middle of the vector are expensive

• Resizing the vector can be costly for large datasets

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Hashtable Pseudocode: |  |  |  |  |
| Code | Line Cost | # Times Executes | Total Cost | Runtime |
| for each line | 1 | m | m | O(m) |
| if courseNumber in coursesHashtable | 1 | 1 | 1 |  |
| print course information | 1 | 1 | 1 |  |
| for each prerequisite of the course | 1 | n | n |  |
| print prerequisite course information | 1 | n | n |  |
| Total Cost |  |  | 2n+2+m | O(m+n) |

Hash Table

Advantages:

* Fast average-case lookup time
* Efficient for large datasets

Disadvantages:

* Hash collisions may degrade performance
* Not suitable for applications requiring ordered data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Binary Search Tree Pseudocode: |  |  |  |  |
| Code | Line Cost | # Times Executes | Total Cost | Runtime |
| for each line | 1 | m | m | O(m) |
| insert course into coursesTree | O(logn) | n | nlogn | O(nlogn) |
| if courseNumber in coursesTree | 1 | 1 | 1 |  |
| print course information | 1 | 1 | 1 |  |
| Total Cost |  |  | m+nlogn+2 | O(m+nlogn) |

Binary Search Tree

Advantages:

* Automatically maintains sorted order
* Efficient for searching and printing in sorted order

Disadvantages

* Can become unbalanced, leading to degraded performance
* Not as efficient for large datasets compared to hash tables

My analysis was conducted considering multiple aspects of what would be needed from the code. After consideration, I would recommend using the Hash table data structure. The Hash table provides a faster average-case lookup time, which is important for efficient retrieval of course information. Additionally, the Hash table can handle large datasets efficiently, making it suitable for the advisor’s program. The Hash table provides the flexibility to maintain ordered data while still benefiting from fast lookup times.