1. “From Project One, **submit your analysis of the run-time and memory for the data structures**.

Let us perform a Big O analysis for each of the data structures.

A screenshot of a computer

Description automatically generated with low confidenceA picture containing screenshot, text, font, design

Description automatically generated

**Vector:**

**Load Data:** O(n)             n number of courses.

Each course is inserted into vector once.

**Check Line Format:** O(1)           Constant time operation.

Checks format of single line.

**Parse Line and Create Course Object:** O(1)

Constant time operation.

Checks a single course object.

**Search for a Course:** O(n)          n number of courses.

Worst case we may have to scan entire vector to find course.

**Print Course List:** O(n)              n number of courses.

Each course in Vector is printed once.

---------------------------------------------------------------------------------------------------------------------

**A screenshot of a computer

Description automatically generated with low confidenceA screenshot of a computer

Description automatically generated with low confidence**

**Hash Table:**

**Load Data:** O(n)                    n number of courses.

Each course is hashed and inserted into table once.

**Check Line Format:** O(1)      Constant time operation

Checks format of a single line.

**Parse Line and Create Course Object:**  O(1)

Constant time operation

Create a single course object.

**Search for a Course:** O(1) on average with good hash function/handling of collisions.                                In the worst case it could be O(n).

**Print Course List:** O(n)         n number of courses.

Each course in hash table is printed once.

---------------------------------------------------------------------------------------------------------------------

A screenshot of a computer

Description automatically generated with low confidenceA screenshot of a computer

Description automatically generated with medium confidence

**Binary Search Tree (BST):**

**Load Data:**  O(n log n)               n number of courses.

                                             Each insertion into a BST is O(log n) for n times.

**Check Line Format:** O(1)           Constant time operation

Checks format of a single line.

**Parse Line and Create Course Object:** O(1)

Constant time operation

Create a single course object.

**Search for a Course:** O(log n) on average, assuming a balanced tree.

Unbalanced Tree - worst case could be O(n).

**Print Course List:** O(n)              n number of courses.

Each course in BST is printed once.

-------------------------------------------------------------------------------------------------------------------

Advantages and Disadvantages of each data structure with storing and retrieving course information:

**1. Vector:**

**Advantages:**

·         Easy to understand, use, and provides dynamic resizing.

·         Great for accessing elements at specific indices, has O(1) access time.

·         Good for cases where the amount of data is not very large and primary operation is to iterate over the elements.

**Disadvantages:**

·         Not efficient for search operations, needs O(n) time to search for an element.

·         Insertion and deletion of elements at the beginning or in the middle are not efficient, requires shifting of elements.

-------------------------------------------------------------------------------------------------------------------

**2. Hash Table:**

**Advantages:**

·         Fast access to elements based on keys. Can provide O(1) time complexity for search, insert, and delete operations.

·         Good when quick lookups are important.

**Disadvantages:**

·         Hash collisions can occur, where two different keys hash to the same index.

·         The efficiency of hashing depends on the hash function and load factor.

·         Hash tables typically use more memory than other data structures.

-------------------------------------------------------------------------------------------------------------------

**3. Binary Search Tree (BST):**

**Advantages:**

·         Provides a good compromise between efficient search, insert, and delete operations, all of which can be achieved in O(log n) time complexity with balanced tree.

·         In-order traversal of BST gives elements in sorted order without needing to sort the elements again.

**Disadvantages:**

·         The time complexity can degrade to O(n) if tree becomes unbalanced.

·         Requires more memory for storing pointers to child nodes.

·         More complex to implement than vectors and hash tables.

-------------------------------------------------------------------------------------------------------------------

Overall, the choice between these data structures would depend on the specific requirements of the application and the person creating the program.

·         A Hash Table might be best If quick lookups are important and there's an expectation for a large number of courses.

·         A BST might be best If maintaining the courses in sorted order is important.

·         A VECTOR might be best If the number of courses is small and simplicity is a priority.

-------------------------------------------------------------------------------------------------------------------

For this “specific” short list of courses, I would go with a vector since they are good with short lists. I instead chose hash table for a more practical and real-world approach to the problem.

Based on the Big O analysis and the evaluation of the three data structures, I would recommend using the Hash Table for this specific problem. Here's why:

PROS:

**1.**  **Fast Access:** Hash tables provide fast access to data. They offer O(1) average time complexity for search, insert, and delete operations (fastest among the three data structures). This is great if you have a large number of courses and need to frequently look up course information.

**2.**  **Efficient Lookups:** Great for efficient lookups by key.

CON:

**3.**      **Handling Collisions:** While hash collisions can occur, they can be managed with techniques like chaining.

It is important to note that the best data structure often depends on the specific requirements and constraints of the problem. While hash tables provide fast access, they typically use more memory than other data structures. If memory usage is a concern, we might need to consider one of the other data structures. Overall, the hash table makes the most sense for this problem.