Based on the pseudocode I produced in the previous assignments the worst-case running time for each data structure in the algorithm can be measured with the Big O value.

1. **Big O Analysis for Vector**
   1. Worst-case O(n) If the vector needs to resize
2. **Loading file:**
   1. Reading the file: O(1)
   2. Parsing each line: O(1)
3. **Creating course objects**:
   1. Creating course objects: O(n)
   2. Inserting to vector: O(1) with each insertion
4. Printing all courses:
   1. Sorting vector: O(logn)
5. b. Printing each course: O(n)
6. **Big O Analysis for Hash Table**
   1. Worst case: O(n) if many elements collide in the same bucket
7. **Loading file:**
   1. Reading the file: O(1)
   2. Parsing each line: O(1)
8. **Creating course objects**:
   1. Creating course objects: O(n)
   2. Inserting to hash table: O(1) with each insertion
9. **Printing all courses:**
   1. Sorting vector: O(logn)
   2. Converting hash table to list: O(n)

c. Printing each course: O(n)

1. **Big O Analysis for Binary Search Tree**
   1. In the worst case unbalanced tree: O(n)
2. **Loading file:**
   1. Reading the file: O(1)
   2. Parsing each line: O(1)
3. **Creating course objects**:
   1. Creating course objects: O(1)
   2. Inserting to binary search tree: O(1) with each insertion
4. **Printing all courses:**
   1. In-order traversal: O(1)
   2. Post-order traversal: O(1)
   3. Pre-Order traversal: O(1)
5. **Advantages and Disadvantages of each Data Structure**
   1. **Vector advantages**
      1. Dynamic insertion and deletion because vectors automatically resize as an element is added or removed. This helps memory allocation because the initialization of the vector leaves the size unknown, and the elements are sorted sequentially.
      2. The code for vectors is easy to implement and compatible with c++ programming language.
   2. **Vector disadvantages**
      1. Not very efficient working with large sets of data because of the slower sorting time.
   3. **Hash Table advantages**
      1. Searching, inserting, and deleting elements because the mapped key to the index allows direct access to the desired element.
      2. Very flexible. The hash table can store any data type
   4. **Hash Table disadvantages**
      1. If two keys are mapped to the same index it will cause a collision which will disrupt the functions of searching, inserting, and deletion
   5. **Binary Search Tree advantages**
      1. Most efficient data structure for searching, inserting, and deleting because of the sorted parent-child relationship which makes traversal time logarithmic
   6. **Binary Search Tree disadvantages**
      1. An unbalanced tree causes the sorted order to become skewed which can degrade the functions
6. **Recommendations**
   1. Based on the Big O analysis I have found that the Binary Search Tree is the most efficient data structure. However, because the desired function of the code is to print a list of all the computer science courses in alphanumeric order including the given course, title, and prerequisites I would recommend implementing a hash table. Hash tables allow for fast searching when the key is effectively mapped to the correct index, perfect for the course information data. Since the hash table works well with the main data types which allows a flexibility for how to map the data.