**CS 300: Project One – Pseudocode and Runtime Analysis**

Alejandro Coitinho

CS 300: Project One

October 9th, 2025

# Pseudocode

## Vector Pseudocode

OPEN file  
READ each line in file  
PARSE data into course objects  
STORE each course in vector  
FOR each course in vector:  
 DISPLAY course information  
END FOR

## Hash Table Pseudocode

OPEN file  
READ each line in file  
PARSE data into course objects  
HASH each course number to determine index  
INSERT course into hash table at calculated index  
FOR each entry in hash table:  
 DISPLAY course information and prerequisites  
END FOR

## Binary Search Tree Pseudocode

OPEN file  
READ each line in file  
PARSE data into course objects  
INSERT course into BST using course number as key  
FOR each course in BST (in-order traversal):  
 DISPLAY course title and prerequisites  
END FOR

## Menu Pseudocode

DISPLAY menu options:  
1. Load Data  
2. Print All Courses  
3. Search for a Course  
9. Exit  
  
IF choice == 1 THEN  
 LOAD data from file into data structure  
ELSE IF choice == 2 THEN  
 DISPLAY all courses  
ELSE IF choice == 3 THEN  
 PROMPT user for course number  
 SEARCH for course and DISPLAY details  
ELSE IF choice == 9 THEN  
 EXIT program  
END IF

## Alphanumeric Sort Pseudocode

READ all course objects from data structure  
SORT course list by course number in ascending order (A–Z, 0–9)  
DISPLAY sorted list of courses

# Runtime and Memory Analysis

The following analysis compares the Vector, Hash Table, and Binary Search Tree data structures used in the pseudocode implementation.  
Each was evaluated based on time complexity, space complexity, and overall performance.

## Vector

Time Complexity: O(n) for search, O(1) for access by index.  
Space Complexity: O(n).  
Advantages: Simple implementation and fast access by index.  
Disadvantages: Inefficient for searches and insertions.

## Hash Table

Time Complexity: O(1) average for insert, delete, and search.  
Space Complexity: O(n).  
Advantages: Extremely fast lookup and retrieval when hashing function distributes keys evenly.  
Disadvantages: Potential for collisions, higher memory usage due to resizing.

## Binary Search Tree

Time Complexity: O(log n) average for insert, search, and delete; O(n) in worst case (unbalanced tree).  
Space Complexity: O(n).  
Advantages: Maintains sorted order, allowing efficient traversal.  
Disadvantages: Performance depends on tree balance.

# Recommendation

Based on the analysis, the Hash Table provides the best overall balance between performance and memory efficiency.  
Its average O(1) time complexity for core operations makes it ideal for managing course data efficiently.  
However, a Binary Search Tree remains a suitable alternative for systems that require ordered traversal or sorting by course number.

# References

Baeldung. (2025). Binary Search Trees in Java. https://www.baeldung.com/java-binary-tree

GeeksforGeeks. (2025). Time and Space Complexity of Vector in C++. https://www.geeksforgeeks.org/vector-in-cpp-stl/

Oracle. (2025). HashMap (Java Platform SE 17). https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html