**Project One**

Edgar I Pena

Southern New Hampshire University

CS-300: Analysis and Design

Professor Aline Yurik

June 22, 2025

# Project One

## Pseudocode

**Vector - Milestone 1**

void searchCourse(Vector<Course> courses, String courseNumber) {

    FOR each course IN courses

        IF course.courseNumber is equal to courseNumber THEN

            EXTRACT digits from course.courseNumber and store in numPart

            PRINT "CS-" + numPart + ": " + course.courseTitle

            IF course.prerequisites is NOT empty THEN

                FOR EACH prerequisite IN course.prerequisites

                    PRINT prerequisite

            END IF

            RETURN

        END IF

    END FOR

    PRINT "Course not found"

}

**Hash Table - Milestone 2**

void searchCourse(HashTable<Course> courses, String courseNumber) {

    GET course from courses using courseNumber

    IF course exists THEN

        EXTRACT digits from course.courseNumber and store in numPart

        PRINT "CS-" + numPart + ": " + course.courseTitle

        IF course.prerequisites is NOT empty THEN

            FOR EACH prerequisite IN course.prerequisites

                PRINT prerequisite

        END IF

    ELSE

        PRINT "Course not found"

    END IF

}

**Binary Search Tree – Milestone 3**

void searchCourse(Tree<Course> courses, String courseNumber) {

FIND course in the tree using courseNumber

IF course is found THEN

EXTRACT digits from course.courseNumber and store in numPart

PRINT "CS-" + numPart + ": " + course.courseTitle

IF course.prerequisites is NOT empty THEN

FOR EACH prerequisite IN course.prerequisites

PRINT prerequisite

END IF

ELSE

PRINT "Course not found"

END IF

}

## Runtime Analysis

***Vector***

|  |  |  |  |
| --- | --- | --- | --- |
| Code Line | Line Cost | # Times Executes | Total Cost |
| open file | 1 | 1 | 1 |
| read each line | 1 | n | n |
| split line into parts | 1 | n | n |
| check number of parts | 1 | n | n |
| for each prerequisite, search courseCodes | n | n | n² |
| create course object | 1 | n | n |
| add course to vector | 1 | n | n |
|  |  |  |  |
| Total Cost |  |  | n² + 5n + 1 |
| Runtime |  |  | O(n²) |

Advantage: Easy to use and simple to set up.

Disadvantage: Slower when checking for things or finding courses.

## *Hash Table*

|  |  |  |  |
| --- | --- | --- | --- |
| Code Line | Line Cost | # Times Executes | Total Cost |
| open file | 1 | 1 | 1 |
| read each line | 1 | n | n |
| split line into parts | 1 | n | n |
| check number of parts | 1 | n | n |
| for each prerequisite, check in hash table | 1 | n | n |
| create course object | 1 | n | n |
| add course to hash table | 1 | n | n |
|  |  |  |  |
| Total Cost |  |  | 6n + 1 |
| Runtime |  |  | O(n) |

Advantage: Very fast at finding courses and checking if something exists.

Disadvantage: Does not keep the courses in order, so sorting takes extra work.

## *Binary Search Tree*

|  |  |  |  |
| --- | --- | --- | --- |
| Code Line | Line Cost | # Times Executes | Total Cost |
| open file | 1 | 1 | 1 |
| read each line | 1 | n | n |
| split line into parts | 1 | n | n |
| check number of parts | 1 | n | n |
| for each prerequisite, search in tree | n | n | n² |
| create course object | 1 | n | n |
| insert course into tree | n | n | n² |
|  |  |  |  |
| Total Cost |  |  | 2n² + 4n + 1 |
| Runtime |  |  | O(n²) |

Advantage: Keeps courses in order, making printing easy.

Disadvantage: If the tree gets unbalanced, it can take longer to add or find courses

## Recomendation

After reviewing all three data structures, the hash table is the best choice for this project because it offers the fastest way to store and check course data. It handles large amounts of information efficiently, keeping the program simple and fast. A vector is easy to use but becomes slower as more courses are added because it has to search through the entire list to validate prerequisites. A binary search tree keeps courses in order, which is helpful for printing, but it can become uneven if the data isn’t balanced, making it harder to manage. This conclusion is based on the Big O analysis and a comparison of each structure’s strengths and weaknesses in this context. Choosing the hash table will help ensure the program meets the academic advisor’s goal of accessing course information quickly and reliably.