Project One Pseudocode & Runtime Analysis

VECTOR

Reading File:

OPEN file using fstream

IF return value == -1

    THEN file not found

ELSE file is found

    WHILE not EOF (End of File)

        READ each line

            IF < 2 values

                return ERROR

            ELSE

                read parameters

            IF 3 or more parameter is first elsewhere

                CONTINUE

            ELSE

                Return ERROR

CLOSE file

Creating Course Objects

Initalize Course vector

Loop through file

WHILE != EOF

    For each line in file

    For first and second value

        Push_back() to add value

    If there's a third value

        Push_back() to add value until new line

Searching Data Structure

PROMPT for user input

LOOP through vector

IF input == course number

PRINT course information

For each prereq

PRINT prereq information

Code	Line Cost	# Times Executes	Total Cost
Open fstream	1	1	1
While != EOF, read each line	1	n	n
IF > 2 values, return error	1	n	n
ELSE continue	1	n	n
Initialize course vector	1	1	1
For each line	1	n	n
ADD value to vector	1	n	n
Push_back() item	1	n	n
Total Cost			7n+2
Runtime			O(n)

HASH TABLE

Read file:

OPEN file using fstream

IF file returns -1

DISPLAY Error message

ELSE

WHILE file != EOF

Read each line

IF < 2 values within line

RETURN Error message

ELSE

Read parameters

IF > 2 parameter

CONTINUE

CLOSE file

Create Course Objects:

Create Hash Table

Insert bid into table

Loop through file

WHILE != EOF

For each line in file

For first and second values

CREATE temp item to hold values

IF there is a third value

ADD to current value

RETURN

Print from Hash Table:

Prompt for input

Assign input to key

IF KEY is found

PRINT course information

For each course preq found

PRINT prereq course info

ELSE

IF key not found

DISPLAY Error message

RETURN

Code	Line Cost	# Times Executes	Total Cost
Open fstream	1	1	1
While != EOF, read each line	1	n	n
IF > 2 values, return error	1	n	n
ELSE continue	1	n	n
CREATE HashTable	1	1	1
INSERT bid	1	n	n
LOOP through file	1	n	n
For each line	1	n	n
CREATE temporary item to hold value	1	n	n
Total Cost			8n+2
Runtime			O(n)

TREE DATA STRUCTURE

Reading File:

Open file with fstream

IF file returns -1

DISPLAY Error message

ELSE

WHILE file != EOF

Read each line

IF < 2 values within line

RETURN error message

ELSE read parameters

IF > 2 parameter

CONTINUE

CLOSE file

Create Course Objects Structure

Create BinarySearchTree

Insert bids within tree

WHILE != EOF

Loop through file

For each line in file

For first and second value

ADD course ID & course name

If a third value exists

Add prereq until newline found

Search & Print from Tree

Prompt for user input

Create a search and print method

IF root != NULL

Traverse left

IF node == course ID

DISPLAY course information

For each course prerequisite

PRINT prereq course info

ELSE

Traverse right

IF node == course ID

DISPLAY course information

For each course prerequisite

PRINT prereq course info

RETURN

Code	Line Cost	# Times Executes	Total Cost
Open fstream	1	1	1
While != EOF, read each line	1	n	n
IF > 2 values, return error	1	n	n
ELSE continue	1	n	n
Create BinarySearchTree	1	1	1
INSERT bids	1	n	n
WHILE != EOF	1	n	n
Loop through file	1	n	n
FOR each line	1	n	n
FOR 1st & 2nd value, ADD courseID & coursename	1	n	n
IF 3rd value exists, add prereq until newline found	1	n	n
Total Cost			10n+2
Runtime			O(n)

MENU

CREATE bid variable

WHILE input != 4;

PRINT 1. Load Data Structure;

PRINT 2. Course List;

PRINT 3. Course;

PRINT 4. Exit

SWITCH (userInput):

Case 1:

loadBids(bid);

break;

```
                Case 2:
                        PRINT Course List;
                        break;
                Case 3:
                        PRINT Course;
                        break;
                Case 4:
                        PRINT Exit Message
                        exit();

// sorting alphanumeric items
SET mid to low + (high - low) / 2
SET pivot to courseName(mid)
        WHILE courseName(low) < pivot
SET low to low + 1
ENDWHILE

WHILE pivot < courseName(high)
SET high = high - 1
ENDWHILE

IF low is >=  high
RETURN
ELSE
CREATE temp to courseName(low)
SET courseName(low) == courseName(high)
```

SET courseName(high) == temp

SET low to low + 1

SET high to high - 1

ENDIF

RETURN high

FUNCTION main()

CALL quicksort(courseName, 0, SIZE - 1)

DISPLAY "Sorted in alphabetical order: "

FOR each course

Advantages & Disadvantages

With a vector, the advantage is that they're the fastest when it comes to reading files and even adding a course it's quick. The disadvantage is if there are a ton of courses, it can be slow in searching through them.

With a hash table, the advantage is that they're quick in searching through a list; however, the disadvantage is that they're difficult to organize data alphabetically or even numerically. Because of this, I wouldn't recommend it for this project.

With a tree, the advantage is that they're faster than a vector, and even easier to use since it's organized numerically. The disadvantage with them is that when it's being organized, the height of the tree increases which means the runtime is increased.

I would recommend personally recommend using a vector to complete this project, based on its speed when it comes to reading and adding objects.