Chris Trimmer

CS-300-T1159 DSA: Analysis and Design

Milestone 3-3: Vector Data Structure Pseudocode

22EW1 – 09/13/2022

The purpose of this document is to provide a pseudocode and algorithm analysis of code for a course planner that we are going to design for ABC University (ABCU). The pseudocode will consist of functions that pertain file parsing, creating objects and storing them in a vector data structure, searching the data structure, and printing the data. Where applicable, a runtime analysis we be documented for a function.

The main object used in the assignment is a Course class. Each course will contain a string value to represent its id, a string value for its title, and a vector of Course objects that will store prerequisites. I use a vector to hold prerequisite courses because a course can have more than one prerequisite. A Schedule object will hold a vector of Courses. At a high-level, a schedule will contain multiple courses, and each course in the schedule can have zero or more prerequisite courses.

File Opening and Parsing

```
Vector<Course> LoadSchedule(string filepath) {

  Create infile ifstream object
  Use infile object to open the filepath

  If infile is not null
    Create string object to hold a line read from the file
    Create a char object and set to a comma, as the delimiter
    Create a string object to hold a word in the stringstream object, parsed using the delimiter
    Create a vector of strings that will hold words of each line

    Loop through infile object and store each line in the line object
      Pass each line as an object to stringstream object and parse using delim

      Loop through stringstream and push back each word in the line to temp vector

      // check file format
      If the line contains less than two words, then this is an incomplete record
        Print output to user regarding invalid file format
        Return to caller

      // verify that prereqs are valid
      For each word after the first two
        If the word is in not the course list
          Skip the word

      // Creating and storing the object is covered in AddCourse function below
      Call AddCourse and pass the vector of valid words
```

Clear the courseLine vector before starting the next loop

} // end LoadSchedule

Code	Line Cost	Execution Times	Total Cost
Create ifstream object	1	1	1
Open filepath	1	1	1
if infile is not null	2	1	2
Create 5 local variables	5	1	5
for each line	1	n	n
create sstream object	1	1	1
for each sstream object	1	n	n
if courseLine less than 2	2	1	2
return to caller	1	1	1
for each word after 2	1	n	n
if the word not in course list	1	1	1
skip the word (continue)	1	1	1
add word to vector	1	1	1
call AddCourse	2n + 5	1	2n + 5
		Total Cost:	5n + 21
		Runtime	O(n^2)

Creating and Storing Objects

Void AddCourse(vector<string>& line, vector<Course>& courses) {

Instantiate Course object

For each word in line
Set course id to line[0]
Set course title to line[1]

For each additional word in line
Push back word into the prereq vector stored in the Course object

// the course now has all its data
Push back the course object into the vector of courses

}

Code	Line Cost	Execution Times	Total Cost
Create course object	1	1	1
for each word in lines	1	n	n
set course id to line[0]	1	1	1
set course title to line[1]	1	1	1
for each additional word in lines	1	n	n
push back word to prereq vector	1	1	1
push back completed Course	1	1	1
		Total Cost:	2n + 5
		Runtime	O(n)

Search for a Course and display information

Void Search(vector<Course> schedule, string key) {

For each letter in key
convert to uppercase

For each course in schedule
If the key matches the course.id
Print the course information

If prerequisite count is greater than 0
For each prerequisite
Print the prerequisite information

}

Code	Line Cost	Execution Times	Total Cost
For each letter in key	1	n	n
Convert to upper	2	n	2n
for each course in schedule	1	n	n
if key == courseId	1	n	n
Print course information	1	1	1
if prereq count greater 0	1	n	n
For each prereq	1	n	n

Print prereq information	1	1	1
		Total Cost:	7n + 2
		Runtime	O(n)

Print Courses

```
Void DisplayCourses() {

  For each Course in Schedule vector
    Print course information (id and title)
    For each Course p in c.prereq vector
      Print prereq information (id only)

}
```

Code	Line Cost	Execution Times	Total Cost
For each course	1	n	n
Print course information	1	1	1
For each prereq	1	n	n
Print prereq information	1	1	1
		Total Cost:	2n + 2
		Runtime	O(n)

Get Prereq Count

```
Int numPrerequisiteCourses(vector<Course> courses, Course c) {
  Set sum to 0
  Set totalPrereqs = prerequisites in c
  For each prereq in totalPrerequisite
    Increment sum

  Return sum
}
```

Code	Line Cost	Execution Times	Total Cost
Set sum to 0	1	1	1

Set totalPrereq to Prereqs of C	1	1	1
For each prereq in totalPrereq	1	n	n
increment sum	1	1	1
return sum	1	1	1
		Total Cost:	n + 4
		Runtime	O(n)