Tyler Lafrance

CS-300: DSA Analysis and Design

Southern New Hampshire University

April 4th 2023

Vector Psuedocode

File Opening and Reading Psuedocode (same for all data structures)

```
string fileName
read fileName from ifstream
string lineString
vector prerequisites
while (get current line from fileName and put it in lineString)
        split lineString by comma and put in new vector text
        if text vector is smaller than 2
                print “not enough parameters on line”
                break loop
        add first element of text vector to prerequisite vector
        if text vector is bigger than 2
                if elements after 2nd element are not in prerequisite vector
                        print “no prerequisite found”
                        break loop
close file
```

Object Creation Psuedocode

```
string fileName
read fileName from ifstream
string lineString
vector courses
while (get current line from fileName and put it in lineString)
        split lineString by comma and put in new vector text
        string prerecquisites
        if text vector size is greater than 2 elements
                starting at element 3, loop until vector size -1
                        prerequisites += (“, ” + current element)
        courses append(new course(element 1, element 2, prerequisites))
close file
```

Search and Print Psuedocode

```
string searchValue
for each element in course vector
        if course.code equals searchValue
                print course code, name, and prerequisites
```

Hashtable Psuedocode

Object Creation Psuedocode

```
string fileName
read fileName from ifstream
string lineString
hashtable courses
while (get current line from fileName and put it in lineString)
        split lineString by comma and put in new vector text
        string prerequisites
        if text vector size is greater than 2 elements
                starting at element 3, loop until vector size -1
                        prerequisites += (“, ” + current element)

        courses add (element 1, (element 2, prerequisites))   (course code is the key, course name
                                                                and prerequisites added as a tuple for
                                                                the value)
close file
```

Search and Print Psuedocode

```
string searchValue (a course code)
if searchValue is in course hashtable
                print course code, name, and prerequisites
```

Binary Tree Psuedocode

Object Creation Psuedocode

```
string fileName
read fileName from ifstream
string lineString
binaryTree courses
while (get current line from fileName and put it in lineString)
        split lineString by comma and put in new vector text
        string prerequisites
        if text vector size is greater than 2 elements
                starting at element 3, loop until vector size -1
                        prerequisites += (“, ” + current element)
        create new course object
        set courseID to element 0
        set courseName to element 1
        set coursePrerequisites to prerequisite vector
        add course object to courses tree
close file
```

Search and Print Psuedocode

```
string searchValue (a course code)

foundCourse = courses.search(searchValue) (returns a course)

if foundCourse is not empty

        print foundCourse

else print course not found
```

Menu Psuedocode

```
bool loop = true
int choice  = 0
while bool:
        print the actual menu
        set choice to user input
        switch choice:
                case 1:
                        loadData()
                        break
                case 2:
                        printList()
                        break
                case 3:
                        printCourse()
                        break
                case 9:
                        print “goodbye”
                        set loop to false
                        break
                default:
                        print “invalid choice”
```

Course In Order Printing Psuedocode

Vector

call sort function on courseVector

for each item in courseVector:

    print current vector item (course ID, name, prerequisites)

Binary Search Tree

printAll() (the BST should already be sorted, so it's just a matter of using the built in print function, modified for courses instead of bids)

Hashtable

For the hash table I'm not exactly sure how to do this. Because a hash table in inherently unordered, a way to iterate over the table would need to be added. After the iterator is added you could probably do something like this:

new vector sortVector

for each item in courseTable:

    sortVector add current item

call sort function on sortVector

for each item in sortVector:

    print current vector item (course ID, name, prerequisites)

Runtime Analysis

	Vector	Hash Table	BST
Reading File	n	n	n
Loading Data	n	n	n
Printing Data	n	1	log n

Advantages and Disadvantages of Each Structure

Vector: Easy to sort, resizable, easy to add and remove first/last elements, can access elements by index. But they can be slow to search, and deleting elements from the middle can also be slow.

Hash Table: Quick insertion and reading, but can have problems with hash collision. It can also be slow to resize a hash table if it's required. It's also not good for data that needs to be in a specific order.

Binary Search Tree: Fast searching, data is already sorted. But removing or inserting nodes can be slow depending on how large the tree is.

Recommendation

My recommendation for this project is a binary search tree. While inserting and deleting can sometimes be slow in a BST, the data in this project only needs to be inserted once and it will not be deleted. The fact that it's ordered makes it very easy to print all the courses, and also makes it fast for searching (which I'm assuming would be done fairly frequently to reference ones courses they need).