Project 1

CS-300

Nathaniel Maplethorpe

Vector

Open file{

Open file = open(filename)

While file is not equal to empty

Read file

Return file

}

Create course{

Integer course number

String course name

Vector <string> prereqs

}

courseSearch

create new course for all courses

if the course number is greater than current put in front

if course number is less than current keep it behind current

print course course

}

HashTable

Open coursefille{

If (file == filename){

Open file

Return file

}

Else {

Print file is not found

}

}

Create course(courses, coursenumber){

Create a key

While key is not 0{

If node is equal to course{

Create course

}

}

Search Course(courses, courseNumber){

Create new key

While key does not equal 0{

If course number equals course number{

Print course information

Print course reqs

}

}

}

Binary tree

Open coursefille{

If (file == filename){

Open file

Return file

}

Else {

Print file is not found

}

}

Create course(courses, coursenumber){

Create a root

While key is not 0{

If node is equal to course{

Create course

}

}

Search Course(courses, courseNumber){

Create root

If root is 0 the current course becomes root

Else if the course number is less than the root add to left

If left is equal to 0 we add course number

Else

if course number is less than the leaf we add it to the left

if course number is greater than the leaf we add it to the right

Else

If course number is greater than the root we add it to the right

If right is not equal to 0 we add the course number

Else

If course number is greater than the leaf we add it to the right

If the course number is less than the leaf we add it to the left

Print course information

Print course reqs

}

}

}

Menu Psuedocode

Key N=Node

While Users input is not equal to 9{

Print : “What would you like to do”

“Press 1 to Load Data”

“Press 2 to Print the available courses.”

“Press 3 to Print a specific course from the list.”

“Press 9 to End the Program”

If users input is equal to 1{

Open course files

Load course num

Load course name

Lod course reqs

}

If users input is equal to 2{

While node\* is not equal to nullPTR{

If node\* is greater than currentN move node to the front

If node\* I less than currentN keep it behind currentN

print courses.

}

If users input is equal to 3{

Ask for user ¡nput course number

If course is equal to course number{

Print course info and course req

}

}

If users input is equal to 9{

Exit program

}

Alphanumeric print

Start at top

While the currentN is not equal to nullPTR

If greater than currentN put in front of currentN

If less than currentN keep behind currentN

Print list from the top to bottom.

}

Reflection

The code has a worst scenario of O(N) as it loops thought all of the files to find one of which it can read so if it has 200 line then it has to read it 200 separate time in order to read the file properly.

For creating a course for all of the trees the worst case is O(N) as well this is because all items are created one after another so when it comes to searching the worst case is O(N) due to the object being searched from the last item.

The main advantage of doing a vector over all of them is the ease to use and ease to code they can add items quick but when it comes to searching it’s not as fast as the other.

A hashtable is a better option when it comes to inserting and searching of items as well as being quicker the biggest draw back to the hashtable is it is much more difficult to implement when compared to a vector and generally takes more time.

A search tree unlike a hashtable is easier to implement and only slightly more difficult than a vector but has the benefit of being quicker than a vector and slower than a hashtable so it’s roughly in the middle of them and easier to implement.

For me I think the binary tree is best for this as there isn’t a lot of items to insert there for the speed wont matter as much and I have found a binary search tree is the easier for me to code and implement.