Jennifer Rebella

CS-300

Prof.

10/09/2024

Project One: Pseudocode and Runtime Analysis

Create pseudocode for a menu:

Create integer userInput

userInput equals 0

Create Bid variable

While userInput is not 9;

Print “1. Load data”

Print “2. Print in alphanumerically order”

Print “3. Print the course title and the prerequisites”

Print “9. Close program”

userInput equals input

switch userInput

case 1:

load Bids passed bid

break

case 2:

Print course list

Break

case 3:

Print course

Break

Case 9:

Print “Goodbye”

Exit

Break

Default:

Print “Input not found”

break

Design pseudocode that will print out the list of the courses in the Computer Science program in alphanumeric order:

Create string sorting

Sorting parameter set to string s

Create array character set to length 1

Create string and add to character array

Sort array

Create integer alphabet

Create integer number

Alphabet equals 1

While alphabet is less than 97;

Alphabet ++1

For I in alphabet

If I is less than 97;

Number ++1

Else,

Return s

Evaluate the run time and memory of data structures that could be used to address the requirements:

|  |  |  |  |
| --- | --- | --- | --- |
| Run Time Analysis: | Vector | Hash Table | Binary Tree |
| Loading Data | O(1) | O(1) -O(N) | O(log N) |
| Search | O(n) | O(1)-O(N) | O(log N)- O(N) |
| Run Time | O(N log N) | O(N) | O(N) |

Explain the advantages and disadvantages of each structure in your evaluation:

The three data structures all have their own strengths and weaknesses. While using an unsorted vector and using the append method to load the data is one of the quickest options, sorting the data later takes a very long time leading to vectors having one of the slowest performances. Using a hash table could be as fast as using a vector for loading the data if there were enough room to stop all collisions, but since time and space is limited the hash table would vary in processing times. For the binary tree it does not require the data to be sorted since it can be traveled in any order. This can save time but too much unsorted data while slow down the processing.

Make a recommendation for which data structure you plan to use in your code:

The best data structure to pick is based on how frequency that the data is accessed. Since the data is only read into memory rarely, is printed rarely, but is searched often, the hash table would be the best option. Though the hash table needs to be optimized to avoid collisions.