Benjamin Leanna

Pseudocode Project One – Page One

CS300 02-09-2023

I broke down the pseudocode pages into their respective cpp files for better organization and readability (to me at least). Page 1 here is where the database function (asking which to use) and the main function of the program will be. Page 2 will be the Vector database, page 3 will be the HashTable database, and page 4 will be the Binary search tree database.

INCLUDE all database cpp files (vector, hashtable, tree)

INCLUDE all necessary general functions (algorithm, climits, iostream, string, namespace)

DEFINE DATABASE function

GET Database wanted // Ask user choice with option to return to main

CASE 1 : VECTOR

LOAD Vector file function

CASE 2 : HASHTABLE

LOAD Hashtable file function

CASE 3 : TREE

LOAD Tree file function

CASE 4 : MENU

LOAD main function to return to main

DEFINE MAIN function // switch function to ask user to chose what they would like to do

SWITCH function

CASE 1

LOAD DATABASE function //either vector, hash, or tree

CASE 2

PRINT PrintAll function from specified database

CASE 3

PRINT PrintCourse function from specified database

CASE 4

EXIT Program

Evaluation:

Here I will evaluate the different best-case and worst-case time complexities of different data structures for search and access (printing). We have three different structures (vector, hash table, and a BST). To start with the vector, the best time complexity would be O(1) with the worse being either O(1) (accessing) or 0(N) (searching). For the hash table, the best would be O(log n) and the worse being O(N). Also, for the binary search tree, the best case is also O(log n) and worst case being O(N).

In comparisons, the hash table are more efficient that search trees or other data structures and provide constant time for searching, insertion, and deletion operations. They can also resize themselves and are easy to use. A disadvantage to them would be that they become inefficient when there are many collisions and have a limited capacity and will eventually fill up. Unlike BST or vectors, they also cannot allow null values. BST’s, however, can automatically sort elements as they are inserted, so the elements are always stored in a sorted order which makes them efficient because they only store the elements and do not require additional memory for pointers or other data structures.

In my code, I will implement a Hash Table. The reason I chose this one is that building a hash table, in my opinion, is one of the basic tasks any developer should know how to do and implement. I’ve seen it asked on different job interview whiteboard questions and feel that it is widely used for so many things. While the others also have their advantages, I want to practice and build a functional hash table.

Sources:

*Time Complexities of Different Data Structures*. (2022, November 22). Geeks4Geeks. <https://www.geeksforgeeks.org/time-complexities-of-different-data-structures/>

*Applications, Advantages and Disadvantages of Hash Data Structure*. (2023, January 16). Geeks4Geeks. https://www.geeksforgeeks.org/applications-advantages-and-disadvantages-of-hash-data-structure/