Milestone Three

Algorithms and Data Structure

Eric Clancy 3/28/2021

Southern New hampshire university  eric.clancy@snhu.edu

To begin with the portfolio that is attached would be referring to the linked list method. This method would be used for smaller data sets so that the source code can be generated faster. With quick implementation to get the project file up and running when the data structure is not as large. The linked list source code displays good housekeeping variables to keep the list from writing over itself and losing data. Start with the head node and continue to push a new bid into the place of the list. Ability to search for a bid that is wanted to be known. The linked list gives a fast and simple method to get the data up and running for temporary use of the data.

With vector sorting there is a display of two methods to compare to in the source file. This demonstrates the timing involved with the algorithm used between quicksort and selection sort methods. The vector source code can allow visual troubleshooting to obtain the best possible outcome for the data set on hand. Monitors the output to reflect which is a faster result. With the two methods you can find that selection sort can be better if the data structure is redefined in another manner to simplify the sorting process. The quicksort will find the results much faster than that of selection sort when dealing on a larger scale of data. This source code is a great way to troubleshoot between the two methods to prove the capability to obtain data in a faster manner for the user.

A great method of organizing large data sets is reflected in the hash tables source code. In the source code you can find that the hash table is also utilizing the chaining method to store data. The code reflects on how the good housekeeping can find the proper data node when searching a bucket with multiple data nodes due to the hashing function. The hash function creates a structure to simplify the data on hand. Another good advantage of hashing is that a node with a large data set within itself can be simplified into a designation of a hash which can reduce the memory usage when accessing the node. The hash function can be utilized to have the data set be organized in a system that can be easily accessible.

With the binary search tree, you can optimize the search ability on a list of data. The tree structure has a root node, internal nodes, and leaf nodes. Root node being the top node in tier zero. From the root node you have your smaller nodes to the left and the larger nodes to the right branching down until the leaf node is made. This source code shows the structure on how to access the search tree and find the node that is wanted without conflict. Always maintaining good housekeeping with having the node set to nullptr for each search. If no node is found being able to add the node in the proper location in the tree. Breaking down a which path to take the search lessens the process that the computer needs to go through. Instead of having a list to go through the binary search method cuts that down. A list of seven nodes in a normal array will have to go search all nodes until the correct one is found. The same list in a binary tree the search will find the node in three search points. Nodes 1-7 and wanting to find seven all points are not needed to search for the seven. You would hit 4, 6 then 7.

The source codes available in the portfolio are all good examples on how to troubleshoot a data set that is given. There are some standards that need to be implemented when using this code. The standard “-std=c++11” will give functionality to the source codes attached. The hash table usage in the source code can be vital to a large database and can be implemented as shown in the source code. Use of a hashing algorithm can have an impact on performance with a data set. Incorporate code for collisions that may occur can make sure the proper data can be read. The chaining method or the linear probing which uses unused spaces.

The source codes available are reusable with minor changes to the source. They are based of extracting information from an excel database and can change the parameters that are needed for the desired result. Some of the codes provided can correlate to other codes already established. The structure of these codes can show the placement in the original code structure. There are multiple comments added to the code to help in that aspect. They are modular in that concept to be able to use them elsewhere.

In computer science it is vital to know algorithms and how to define data structures. Most work environments want to lean towards going digital in that means to create better flow and environmentally helpful. No need to continually print page after page if you have a database that is accessible and the means to access it whenever from a computer. I can use the source code on hand to generate data that I would normally take around a half hour to compose. Instead, I can organize the data by shipment dates and even incorporate timelines that are needed to meet. This attacks a processing problem to cut back on work hours to focus on other issues at hand. The only obstacle I have is the schedule changes constantly and I cannot change the information at that exact moment.

Linked List Data Structure. (n.d.). Retrieved February 22, 2020, from <https://www.geeksforgeeks.org/data-structures/linked-list/>

Fundamentals of data structures: Hashing. (n.d.). Retrieved February 22, 2020, from <https://en.wikibooks.org/wiki/A-level_Computing/AQA/Paper_1/Fundamentals_of_data_structures/Hash_tables_and_hashing>

Parlante, N. (n.d.). Retrieved February 22, 2020, from <http://cslibrary.stanford.edu/110/BinaryTrees.html>