Byron Laferriere

CS-260

28 FEB 2021

Final Project

**I. Data Structures**

**A. Vectors**

The file that best exemplifies my understanding of the vector data structure type would be Lab2-1. In this assignment I was tasked with creating a vector data structure that would hold bids that were being placed or had already been placed. Much like an array, a vector can be used to store an ordered list of data items, with having direct access to each of the items. Once this vector was created, I created a data structure within the vector to hold data from each row entry. Four variables were created within the vector to store the values entered by the program. When tested in Powershell, the program successfully stored and displayed the entered information. This assignment also required learning to loop through the vector to display all the bids stored. The program successfully stored the entered items and was able to search and locate them upon request. This exemplified my understanding of how to implement and initialize vectors successfully.

**B. Hash tables**

The document in the portfolio that best exemplifies my knowledge and understanding of hash tables and how they work would be the file named Lab5-1. In this assignment, we were tasked with creating a structure that could take unordered information and store each item by mapping it to a location. I displayed my understanding of how hash tables work by creating structures to hold all bid information and then implementing code to calculate the hash value assigned for the bid information into a way that the table could be read quickly. The modulo operator was used to calculate a hash value determined by the size of the table. After this, I also implemented logic to insert new bids into the hash table, while the modulo operator was constantly updating the table as necessary. Lastly, in this assignment, I implemented the code to remove items from the hash table. Using the logic hash() and ‘atoi’ or askew to integer allowed for this to successfully happen. This lab clearly exemplifies that I have a keen understanding of how to operate a hash table.

**C. Tree structures**

The document that best exemplifies my mastery and knowledge of tree structures and how they operate with roots, parents, internals, and leaf’s would be the assignment from week 6 titled Lab6-2. In this assignment I was tasked with creating a binary search tree (BST) and filling nodes with new and current bids from the sample bids file. The code begins by implementing the root node and pointers going left and right. After this I created a left and right variable that was assigned to 0 to begin. The second task completed in this assignment worked with inserting new bids into the BST. Using an if loop, the program was easily able to work through filling in the tree by determining if the root was equal to zero. Elsewise, it would fill in the bid into the proper node accordingly by working through left and right comparisons in the tree. I also created a while loop to create a search method through the BST in this week’s assignment. Within the while loop I had an if loop and a nested if loop that controlled the search parameters through the tree. I believe that this verifies that this assignment best exemplifies my understanding and knowledge of tree structures.

**II. Algorithms:**

1. **Search**

The document that best exemplifies my mastery of how to implement the search algorithm in C++ would be the assignment from Week 3 in the folder Lab3-2. This assignment required us to complete many tasks within a linked list that was creating. Of these tasks, the implemented search logic was created using a while loop that checked for the entered value against 0 using the compare() command. From there, it looped over each node looking for a match. When tested in Powershell this piece of the program ran very smoothly, proving that this assignment best displays my understanding of the search algorithm.

1. **Sort**

The assignment that best exemplifies my understanding and knowledge of the sort algorithm is the assignment from Week 4 found in the file Lab4-2 that implements the sort algorithm over a vector. This assignment required us to implement and invoke both the quick sort algorithm and the selection sort algorithm to a vector of bids. The quick sort algorithm was implemented by using multiple while loops to check against the pivot to determine whether to decrement or increment. Selection sort was completed by using if and a nested if that used compare() to sort bids. This assignment showed the speed of the quick sort method over the selection sort and exemplifies my mastery of the algorithm.

1. **Hash/Chaining**

The assignment that best displays my understanding and knowledge of the hash/chaining algorithm would again be from Week 5 in the file Lab5-1 where we used chaining to handle collisions in the hash table by creating temporary buckets. In this assignment I created a variable named ‘temp’ to store these values as collisions happened during inserts and reconfigurations. This lab best exemplifies my mastery of this algorithm.

**III. Student’s Choice:**

The document in the portfolio that was my favorite program was the assignment from Week 5 in the file Lab5-1 that required the creation of a data structure called a hash table and also combined the chaining algorithm. This program created a hash value for each bid by using the modulo operator to continuously convert values according to the table size. The program was able to do so successfully upon testing Powershell, when adding new bids into the table and seeing the chaining algorithm create a temporary bucket to store bids as they were passed through the hash table. There is well-written in-line comments that lay out the decisions that were made while writing the code for this program. This is most evident in the design of the structure Node() under FIXME1. The code is concise and easy to read, which makes it reusable and easy to edit as needed. The area of code which best highlights this fact is the implementation of the erase() and resize() logic. No matter how the table changes, the code is prepared to adapt. Modular composition is achieved throughout the code as it has been spaced out into easy to comprehend modules of code according to the needs of the program.

**IV. Conclusions:**

Data structures play a very important role in computer programming because of their foundational properties that they supply to a program. Programs need data structures to collect and organize data within the program, in order to run successfully. Depending on the needs of the program, different data structures can be more appropriate for each individual case needing to be satisfied. The assignments in this course required us to explore vectors, hash tables, and different types of trees. Throughout the semester, we learned how these different types of structures were both similar, yet different in many ways at the same time. Algorithms play an important role in interacting with the chosen data structures. Program effectiveness relies on this combination of algorithm and data structure to handle large amounts of data for processing purposes when necessary. This semester, we covered different algorithms that do this like search, sort, and hash/chaining. The assignments had us combine different techniques like vectors and sorting and showed the power of using quick sort over a vector, when compared to selection sort combined with a vector. There was a 14 second differential between the two when only dealing with 78k files. I can only imagine the impact this type of decision could have on a large program’s processing speeds. This assignment highlighted the importance of properly combining data structures and algorithms to avoid slowing down the program. This can interestingly enough be compared to the balance that is needed in our daily lives to ensure efficiency. Going out drinking every night, will never mix with professional needs and will only slow you down in the long run. All while blocking personal ability to perform as you should be able to and slowing down your own personal ‘program.’

**Reference:**

Unknown. (n.d.). *CS: 260 Data Structures and Algorithms*. Zybooks.