CS 260 Module Final Project

In life we have tasks and information that we must store and manage. In the case of our customer eBid Nashville they are no different. The amount of data that eBid must store and access is massive. Storing this data without the use of technology would be an almost unmanageable task. As a result of this eBid Nashville has asked us to come up with a program to manage this data in a fast and effective way possible. In this document I will outline three data structures and their effectiveness at solving the problem, three algorithms and how to best implement the concepts of each, and lastly I will outline my suggestion of data structure and algorithm.

A screen shot of a computer code

Description automatically generatedA screen shot of a computer code

Description automatically generated The first data structure I would like to point out is vectors. In the past I have used Vectors as a storage method for data and I have a good understanding of their usage and operations. Vectors are very interesting data structures that are used in the C++ language. Vectors just like other containers can hold however many data types we want and be accessed by using the “.” operations. For our purposes we used a structure to hold things like the bid id, title, fund, and the amount. We then used the dot operation to add them into our vector for each individual bid. The choice for using vectors is good because vectors can change in size depending on our needs. As our needs change so does our vector. We can use the method “resize” that allows us to grow or shrink the vector. By using a vector we store them in a linear way meaning that as we add a bid into the vector it starts from the first element and then continues down the line. One of the issues with this way is that if our bids are not sorted before we add them to the vector they will stay unsorted. I personally like using vectors because they were one of my first data structures I learned and had a lot of experience good and bad with them.

Adding Bid Items to a vector

Structure Definition

A computer screen with white text

Description automatically generatedA black background with white text

Description automatically generated The next data structure we will talk about is hash tables. I do not have much experience working with hash tables and because of that I find them difficult to work with. The concept of hash tables makes use of the modulus operator which when used divides a number by whatever we want and leaves the remainder. The hash table uses that number as the key that we should add to an array or vector. As an example, if we took the number “11 modulus 10” we would be left with the number “1” as the key so we would store that number into the data structure with that key. One of the challenges of this method is dealing with the collision of bids with the same key. The way we learned to combat this challenge was to move to the next element that was not used and store it there. Dealing with the hash table is not the worst method for storing the elements but for troubleshooting and working with them for myself can become very complex and difficult.

Combat Collision

Key Creation

The last data structure we will look at is the binary search tree.