Final Project

CS 260

Tonio Thurman

I. There are great structures that could be used to organize data from the ebid site. Starting with **vectors**, this would be a great choice specifically for the arrays. This would allow us to use this special variable to store a list of bids with each item accessible. This is also great for the bid program due to the discrete boundaries. The bids have similar data thus more efficient and accurate data storage for the program. Getting into detail, we could use eclipse to define the vector structure to hold the ebids from the csv file. Then we want to go in and create another structure to hold the ebids from each row to add to the vector. Then the program will need a method call to load the ebids from csv file.

The next structure is **hash tables**, this option would be great for searching for key items on the ebid program. If you were to go on the Nashville bid site or enter the program to look for specific item then this may be a great choice to implement into the program. In this instance each hash table array is called a bucket. Using the item’s key, it makes a bucket index to use. This option uses basic operations like insert, remove, and search to decide the ebids bucket. Something to look out for when choosing this to put in the program are collisions. This happens when a item is being put into the hash table and it was in the same bucket as an existing item in the hash table. There are techniques to use to keep this from happening such as each bucket having a list of items or ebids.

Lastly to discuss are **tree** structure options for the program. This would be a good choice to implement when searching for ebids quickly. This will be a faster search than a list such as a vector. It does this by starting with the root node or current node, then searching the children nodes. So only a portion of the nodes get compared then revealed. A node with no children is called a leaf. And a node with at least one child is called a internal node. The top node is the root.

II. For algorithms my portfolio has a few great options for the ebids. The **search** algorithm has a few good pros when dealing with this type of data. This would allow the program to keep the search simple when looking for certain ebids. Search returns the first node if the data matches that key, such as name or item ID. When working with singly-linked list checks the current node or head node if no match then it points to the next. Another pro is we can used singly-linked list to save memory compared to doubly-linked list. This is because the single uses one node at a time.

Next we have **sorting** algorithms for the program. Sorting would be used for putting the ebids in order ascending or descending etc. The program will simply swap values to put them in order. The selection sort uses two parts one as a sorted and the other as a sorted. Then searches the unsorted and uses the swapping method till the smallest value is found. Using eclipse C++ we first would create a sort logic for the ebids. Then my portfolio will show the selection sort bid logic using the ebid title. Then lastly we want to focus on the timing results from the sorting, so we want to implement a quick sort timing result for the sorting.

**Chaining** could be very useful when dealing with hash tables. Chaining stops collisions by using a list for each bucket. So first it uses the items key to find the bucket then the bucket list. It works for both remove and insert operations. This allows multiple items to map to the same bucket. Collision data will still be put in the bucket list.

III. **My choice** for the best structure and algorithm for this program would be the vector/sorting option. The vector would be great due to the list storing and efficiency. Another pro is the in-place sorting, which would be replacing the bids with the arranged/sorted bids with the same value. This is implemented by using the quicksort function in C++. The vector is also great due to it being able to double the array size. Putting this with the sorting which is great for simplicity and memory we have a great program for the ebids.

IV. In **conclusion** the information for this class is vital. The information is very important for those choosing to create software. Knowing how and what structures/algorithm to use to make a program will save the coder time. Data structures are the organization of the data and how you will management this in the program. For example vector structures are great for lists of data & hash table structures are great for specific items/data. The algorithms are the performance or the instructions for the program. Without algorithms the program would do nothing but hold data. A great example is without the sorting implementation in the vector sorting there would be no logic and the program would be out of order and the bids wouldn't do anything. Comparing lessons learned to my professional life I would first focus on a plan or structure when creating software then focus on the performance and how to manipulate the data to do what I want. This seems a lot smarter than just starting with no plan and just coding. This would most likely lead to a mess a lot of unnecessary debugging.

References

[www.learn.zybooks.com](http://www.learn.zybooks.com)

<https://data.nashville.gov/Business-Development-Housing/eBid-Monthly-Sales/n54t-t7gg/data>