Chase Verbout
CS163
Program

Efficiency Review 3

1) How well did the data structure selected perform for the assigned application?

The data structure for this given assignment was a hashtable that uses LLL chaining to solve collisions. The application was similar to a database that contained items and their respective information. Wherein users are looking to search up these items by name and observe that information in a timely manner. The data structure adheres to this concept very well because we can find the chain that each item is in using our hash in a matter of a few operations. This applies to all of the ADTs functions as well. With Add, Remove, and Specific Displays we can immediately find the LLL that it should be in and check for matching items. This significantly decreases the runtime of the functions because we don't traverse the whole structure. There are some concerns with the display by type feature that requires a full traversal because type is not the key in this hashtable. And more so if the user wanted to assess any of the information by relative position because a hashtable does not lend itself to that functionality. Though our application does not indicate that the user wants this and so I think it's fair to say this hashtable data structure is adequate. It could be paired with other data structures well to provide more rounded capability.

2) Would a different data structure work better? If so, which one and why...

I think that from what we know, a hashtable works well enough for this ADT. The structure can take on any number of items because of its linear linked lists and a feature that could be added is a resize feature that takes into account when the current chains are too long and prompts the hashtable to increase its array size and re-hash. I think that creating a second hash table organized by type could have been a solution to changing the runtime of display_type. However then you would need to consider the overhead of creating a whole new hashtable just for one function that displays. It may be worth it in that case to just have one slower function. Another aspect to consider is that with chaining we are using more storage with pointers than if we were to use arrays and double hashing. Though that would come with it's own issues such as static array allocation and necessary contiguous memory storage. Ultimately the needs of this application were met by the hashtable so from what I know I am content with it.

Chase Verbout
CS163
Program

3) What was efficient about your design and use of the data structure?

The efficient aspect of my design stems from the hash function itself. A quick and simple math equation that allows us to evenly distribute the items into different chains. Using prime numbers we are able to ensure that our items can land in every possible chain. And so with the hash function we enable ourselves to find items by a given keyword without looking through every possible item. We know exactly where we would store an item if it were to exist and so we look through that chain and that chain alone. This works with add, display item, remove, and retrieve. It also benefits from the ability to add any number of items it wants to a chain. There is no requirement on the size of a chain because we are not using an array but a LLL.

4) What was not efficient?

I would say the inefficiency in this program comes from its memory storage and the display_type function. We know that anytime we use a LLL we are committing to the use of more storage in the way of pointers. So it would be important to know if we have the proper overhead before using this data structure with a large amount of data. As for display_type, as I've said it's an O(n) function that needs to see every single item in the structure because we did not hash by type. This problem would occur if the user wanted to look up anything by a category other than name and so it is not efficient at all in that manner.

5) What would you do differently if you had more time to solve the problem?

Were I to have more time I would play around with this application as a BST now that we are starting to learn what that is. I think that the relative ordering that it maintains would be very useful for a user who was looking to find data with a filter like “ratings between 5 and 10” or “games starting with the letter a” . While the application didn’t ask for that functionality I think a user would want their data to be able to be ordered by more than just one feature as well. So multiple data structures could be something I look to do as well.