Andy Fleischer

Prof. Karla Fant

CS163

30 April 2021

Program 2 Efficiency Writeup - CS_Trivia_2

This program was required to store trivia questions and answers in stacks (implemented as a linear linked list of arrays) for a discard pile and correct pile and a queue (implemented as a circular linked list) for the draw pile. In my implementation, both of these classes shared a trivia struct that held a question and an answer. Generally, these data structures did well for what was needed in the program. They had flexible sizes and did well with the small data sets that were needed. If the data sets were bigger, however, some changes would need to be made.

The stack data structure made sense for the given implementation since a pile is basically a stack. However, if shuffling the pile were implemented, it would be complex to make a thorough and efficient algorithm. The circularly linked list queue, on the other hand, made little sense for the draw pile. A draw pile is not inherently circular, and would struggle more if shuffling the questions were to be implemented. Further, a linear linked list of arrays is also more time and memory efficient for traversal than a circular linked list.

If this project were to be expanded, I would suggest keeping the stacks, but making the draw pile another stack of the same class. Being another pile, it makes sense to be a stack, and it would be slightly easier to shuffle. There would be no shifting around and reconnecting of nodes, just the moving of data. Moreover, the partial direct access of a linear linked list of arrays would make traversal a bit faster and reduce memory usage, while still maintaining the flexibility of a linked list. To fully utilize the benefits of a linked list of arrays, I would also increase the maximum size of each node, or at least let the client decide how big they should be, depending on their needs.

To break down the efficiency of both the stacks and the queue, the stacks were slightly more memory efficient, since they did not need as many next pointers. Both ADTs were quite fast for the given functions. Since stacks always add at the head, and they were implemented partially with direct access from arrays, pushing and popping required little or no traversal. The

circular linked lists still only required a couple of movements and allocation/deallocation of memory for enqueuing and dequeuing, but because of that were slightly worse than the stacks. Finally, for displaying the stack was also a bit faster, since iterating over the arrays saves time that is not saved traversing through the entire circular linked list.

Overall, if I had more time I would like to make the change described above by making the draw pile a stack, as well as add some more functions to truly get the benefits of switching. Mainly, I would add a search function and shuffle function, but I could also implement some other functions like removing all data or copying data. If possible, I could even pull from large data sets of trivia questions to make a function to add a random question that does not come from the user. After that, I would also like to go through and clean up parts of the code by making functions for repetitive code, seeing if I could use simpler algorithms, and making sure memory usage is as efficient as possible. Further, I could add some better error handling by using more returns than just 0 or 1, like checking for repeated questions. Finally, outside of the ADT, I could always make a better UI that has stronger input checking and looks nicer.