Andy Fleischer
Prof. Karla Fant
CS163
16 April 2021

Program 1 Efficiency Writeup - CS_Trivia

This program was required to store trivia questions with questions, answers, and categories. Overall, the structure of a linear linked list of linear linked lists lent itself well to this task, and only had minor places where efficiency could be improved without damaging the efficiency of other areas. The reason a nested linear linked list worked well was because the trivia questions could easily be organized by category, improving the traditionally terrible traversal efficiency of linear linked lists. Further, the nature of the problem suggests that the structure likely will not hold too much data, so the downsides of linked list efficiency are minimal. I could have used a linear linked list of linear linked lists of arrays, but this depth of sorting is not completely necessary for the scope of this program. I would likely not use a different data structure, only make some minor modifications as will be described below.

The most efficient functions in comparison to how fast any data structure would be are the display functions. Since you need to traverse through the entire structure for any display, the display all was about as efficient as possible. The displaying a full category was more efficient than just one massive linked list because the questions were sorted by category, so the program did not have to check every trivia question. One massive linked list might save slightly on memory (no category nodes), but would be much less efficient for displaying by category. The other fairly efficient function was removing (entire categories). Again, because the data was organized by category, the program only needed to traverse the category nodes list, and then the list of questions for deletion if a match was found.

The least efficient function by far was adding. Because I did not use a tail pointer for either category nodes or trivia nodes, adding required going through the entire linked list just to get to the end. Also, the selection algorithm was somewhat inefficient. After searching for the category (which was easier because of the sorting by category), the program chooses a random question from the list. If that question has been asked already, the program walks forward from that point to the first unasked question, looping from end to front if needed. If it comes back to the first question it tried to ask, the questions must have all been asked and the function returns 2

to show that. This was the most efficient algorithm that I could come up with, but the worst case requires two entire traversals. The randomly chosen question had to be the last in the list, and all the questions had to have been asked. Luckily, this worst-case is pretty rare, especially on larger sets of questions.

A way to improve the inefficiencies of those functions is to add a tail pointer to both linked lists. I would also consider making the trivia nodes a circularly linked list, to aid in the wraparound of my selection algorithm. The categories can remain linear, since there is no need to allocate more memory for it to be doubly-linked and no inherent circular/stack/queue nature for the categories. Moreover, if I had more time, I would definitely think about my selection algorithm and see if there was a more efficient way to select an unused question. Also, I could have expanded the cs_trivia class a bit more, possibly by adding checks for repetitive questions or more error handling/more unique returns than just 0 and 1. I additionally could have added more functions to the class, such as deleting individual questions, deleting the whole structure, revealing the answer to the last asked question, and so on. Finally, unrelated to the abstract data type, I could always spend extra time on creating a better user interface.