CSCI 230 Fall 2023
Instructor: Michael Levet

Homework Two

Oscar Jiang

9/6/2023

1 Homework Two: Honor Code

On my honor, my submission reflects the following:

- Everything I have submitted is my work and in my own words. Everything that I submit is something I understand.
- Any collaborations and external sources have been clearly cited in this document.
- I have not posted to external services including, but not limited to Chegg, Reddit, StackExchange, etc.
- I have neither copied nor provided others solutions they can copy.

Honor Code Agreement Signature. **Oscar Jiang** □

2 Homework Two: Collaborators

The following people I have collaborated with are as listed:

- Matthew Jennings- is my roommate and fellow classmate. He and I discussed how to write code for problem two. Neither of us knew how to write the selection sort outside of the LinkedListADT itself. I discussed with him how I think the selection sort should work for linked lists and for the most part, we agreed upon a much simpler process in contrast to swapping unnecessarily . That was it and did not affect my final product.

3 Homework Two: Problem One Reflection

The concept that this assignment helped me cement is linked list in general. At first, I thought them to be no different to array lists, i.e. they are formed out of arrays. So arraylists would be my go to for all my storing needs. However, after further assessment of linked lists and their functions, I see that they are far superior for data manipulation than array lists would be– however, they cannot access data as efficient as arraylists.

Arraylists are far more process intensive than linked lists when it comes to data manipulation. For linked lists, all nodes have a previous and next node within and they point to addition nodes or null... that is why traversing the linked list is the hardest part. However,when it comes to insertion or addition, you wouldn't have to shift all elements down or up, create a new space/increment length, but rather just attach the next and previous node pointers of (two)certain nodes to that node you would be adding or inserting between. Therefore, not having to move everything everywhere, but only two objects are affected.

4 Homework Two: Problem Two B

Selection sort for arrays and linked lists are different. For arrays, you would have to find the smallest element past the number you're comparing to, which usually starts at i, via an out loop, starting at j, via an inner loop, then swapping them after each outer iteration. Within the array sorting, there would be a temp variable to keep data to prevent loss and that takes up extra memory. In contrast, for linked lists, you have two loops too; there would be one outer and one inner, one for holding the stop point and one for iterating through the list. However, you would just point the two nodes encasing the selected(smallest) node towards each other. Then, you would point the selected node's next node to the node selected by the outer loop, and its previous node to the outer loop node's previous node– only adding four constant steps on top with the conditional for the head at the beginning. Both arrays and linked lists have outer loops consisting of n - 1, which is the number of elements to be compared to, with each making n-i, for each i, comparisons. However, within arrays, you would have to shift all elements from the index of which was swapped down(when removed), then back up when a new element is added(sorted). Depending on how many swaps, it would lead to a far larger time complexity than linked lists would due to the loops it would have to go through each time it is swapped or removed to shift the elements. Therefore, yes, selection sort is a reasonable sorting algorithm for linked lists.