CS2302: Lab 3 – Sorted List

# Introduction

For this lab, a Sorted List Class needs to be created and be able to sort the elements of a list in ascending order. Along with the class, 10 different functions need to be implemented into the class SortedList to be used with the list in the class.

# Design and approach

When approaching this problem, in the instructions, SortedList class needs to assimilate to List class in some aspects, but the only difference is that SortedList needs to sort the elements of the list in ascending order. It was figured perhaps if it is within the constructor, it might be modified after, therefore the functions were written first before the class. Using pre-made function **Node()**, **Append()**, and **Print()** it was possible to create singly-linked lists within the class.

For the **Insert()** function, the element is inserted at the end along the tail.

For **Delete()** function, it was figured that in order to get rid of a node, the next of the next should be linked to the current node through the while loop so it will cut the undesired node, deleting it. After this thinking, a while loop was implemented so that it checks the next of the current node while at the same time considering the current node itself as a “suspect” to be deleted.

For **Merge()** function, it was figured that in order to add another list, it needed to transverse to the end and be the next of the tail. So, a while loop was implemented so that it goes to the end and the head of the new node **M** will be the next of the tail, and the tail of **M** will become the new tail of the new list.

For **IndexOf()** function, a counter was necessary in order to count the index that has been transverse so far. Thus, stopping the loop when it reaches that index and returning the data from that index counted so far.

For **Clear()**, all is needed was to make the next of the tail the head, making the whole list None.

For **Min()** and **Max()**, inside the while loop, a comparison was made in order to see if the current node had the smallest or biggest element so far, and it continued until reaching the end and printing the desired element in each respective function.

For **HasDuplicates()**, it is similar to **Min()** and **Max()** functions, just that it compares if its equal to current node’s data and, If so, it returns true.

# Experiments

For the experiment, the table provided was used to analyze the big O of n of each class. Notice, the ones with **\*.** These are the differencing big O of n between the two classes in these two specific functions.

For the **Min()** function, in the **SortedList** class, it is assumed that the elements are already sorted in a fashion that the smallest element in the list is the head of the list, thus a constant operation is done to return the head of the list as the smallest element. While on the other hand, **List** class uses a linear operation that requires to look for the smallest element in the list.

For **Max()** function, similar to **Min()** function, in **SortedList** class, it is assumed that the list provided is sorted already, thus finding the greatest element at the tail of the list, therefore returning the tail as the max element. Once again, the **List** class loses by having to attempt to find the max element in its list, outputting a big O of n of **N**, losing to its sorted counterpart.

|  |  |  |
| --- | --- | --- |
| **Function** | **SortedList** | **List** |
| Print() | O(n) | O(n) |
| Insert(i) | O(1) | O(1) |
| Delete(i) | O(n) | O(n) |
| Merge(M) | O(1) | O(1) |
| IndexOf(i) | O(n) | O(n) |
| Clear() | O(1) | O(1) |
| Min() | \*O(1) | \*O(n) |
| Max() | \*O(1) | \* O(n) |
| HasDuplicates() | O(n) | O(n) |
| Select(k) | O(n) | O(n) |

**Table 1: Comparation of Big O of n’s of SortedList function and List Function**

# Conclusion

In conclusion, what was leant from this lab was how to use singly-linked list to do a variety of different functions, and also compare two different types of classes with differently sorted lists that provide different outcomes and big O of n for similarly structured functions within each class.

# Source Code Page

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

# Academic Honesty Certification

I certify that this project is entirely my own work, I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provided inappropriate assistance to any student in the class.

Axel E Manzanares – October 08, 2019