CS2302: Lab 2 – Selection problem

# Introduction

In this selection problem, for every given list, the kth smallest element needs to be searched within the list. (If k = 0, the smallest element in the list will be chosen, if k = 1, the second smallest element is chosen, etc.). To do this, three sorting algorithms should be used: Bubble Sort, Quick Sort and a modified version of Quick Sort, where the inputted list is divided in two sub-lists, and depending on where kis located, only one sub-list is sorted and searched on.

# Design and approach

At first approach, bubble sort was the easiest to implement. The algorithm is two for loops that switches the places of L[i] and L[ i+1] each iteration, which is simple enough to implement.

The second part was moderately hard. The quick sort algorithm is needed to complete this method, thus its implementation is needed. So, putting in the method, I had to create a partition method as well as the quick sort method to be able to make a recursion call to itself, using select\_quick\_sort method as a call to quick sort method.

The third part was relatively hard to figure out since I had to understand how quick sort works exactly in order to do as the instructions say and modify it. So, it was figured that one of the calls to quick sort inside the quick\_sort method is partitioning the higher half of the given list, and the other was doing the lower half. So, by logic, it was figured that two other methods were needed in order to get one half of the list or the other half. Using modded\_quick\_sort method, it was needed to implement a for loop so that it went through the list and make sub-lists, one containing the numbers that indexes were smaller than K ,and another sub-list with numbers that indexes were higher than K. So, comparison was needed in order to call different functions of quick sort in which one method only sorts the lower half of the list and the other method sorts the higher part of the list, Thus, quick\_sort\_if\_lower and quick\_sort\_if\_higher methods were implemented onto the code to do that.

Finally, concentration on syntax errors were applied by using an if statement in each “select” method when inputting a k higher than the list, a message was outputted to clarify that the number inputted was too high.

# Experiments

For experiments, to check how fast each method ran, a 3-trial timer was placed to each sorting algorithm to get their average time. Depending on what type of list it is, it runs differently and gives a different time every trial, thus getting the average was needed to find the overall time.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| List | Type of list | Bubble Sort Average | Quick sort average | mod quick sort average |
| l | Random sorted list | 0.000665 | 0.001329 | 0.000665 |
| L1 | Decreasing sorted list | 0.002998 | 0.000665 | 0.000659 |
| L2 | Ascending sorted list | 0.0 | 0.000665 | 0.000665 |
| l3 | Empty List | 0.0 | 0.000334 | 0.00067 |
| l4 | One-element list | 0.000996 | 0.000665 | 0.0 |

**Table 1: Bubble Sort, Quick Sort, and Modded Quick Sort Average Time**

Next, it was required to find the Big O of N function for each method, which results in the big O of N of their respective algorithm.

|  |  |  |
| --- | --- | --- |
| Method | Big o of n function | simplified function |
| select\_bubble | N2+3N+1 | O(N2) |
| select\_quick | 2N\*N/2+9 | O(N\*logN) |
| select\_modified\_quick | N\*N/2+9 | O(N\*logN) |
|  |  |  |

**Table 2: Method’s Big O notation functions**

# Conclusion

In conclusion, what was learned from this project was how to analyze and compare sorting algorithm better, how to work with lists using methods, and how to find the big O of N through analyzing through the code and figuring out all the major features to form the function to Big O of N.

# Source Code Page

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

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 – September 20, 2019