****

**PROJECT REPORT**

**CS2009 - DESIGN AND ANALYSIS OF ALGORITHM**

**FALL 2022**

**MEMBERS: MUHAMMAD FAHAD 20K-0441**

**KULZUM MUJTABA 20K-0220**

**SECTION: BCS-5C**

**TEACHER: SOHAIL AFZAL**

# Abstract:

In this project, we have implemented 10 sorting algorithms which are executable using GUI. The data is read from a file and any sorting algorithm can be selected to sort the data from the file. The data read from file (unsorted) and the algorithm steps are shown in the interface. Furthermore, the time and space complexity of each algorithm is shown, both general and with the input size of the file.

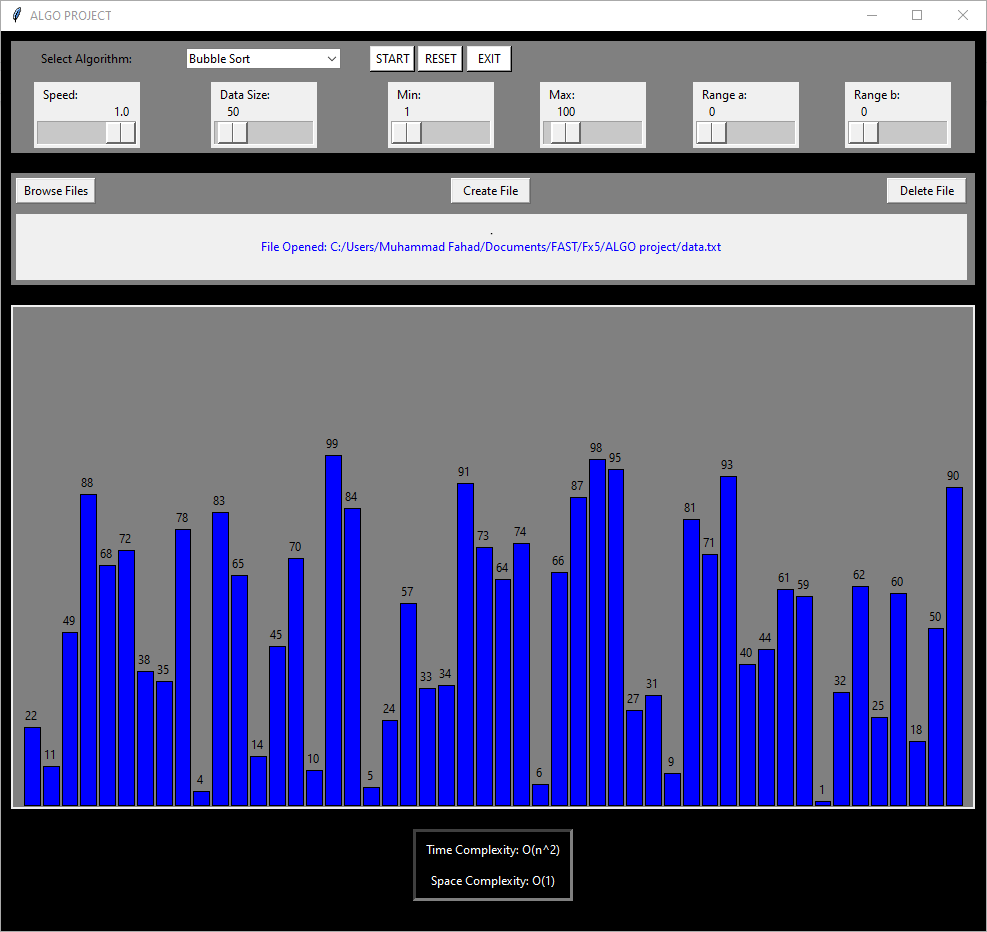
# Introduction:

The goal of this project is to visualize the proposed algorithms on the interface and show steps of how each algorithm sorts the given data. The speed of each algorithm can be slowed to better understand the faster algorithms. Large data files can be generated automatically along with the user providing their own file.

# Your programming design:

# Programming Language: Python

# Primary UI Library: Tkinter



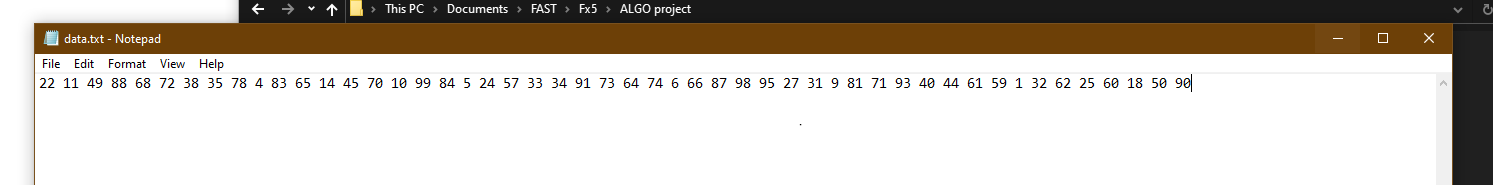
# Experimental Setup:

# The data input from the file is generated through a random number function. The user can select data size to create a new file of numbers and provide a range of numbers for the file. If a file is read from a user it is directly used for the sorting functions. The user interface includes options to slow the speed of each algorithm along with data size and range of numbers, if the user chooses to create a new file. After reading a file, the unsorted numbers are shown on the canvas. The 8.2.4 from the book is implemented using counting sort and the additional sliders in the interface are used to highlight the range input. After sorting, the data can be reset to restore the original unsorted numbers from the file.

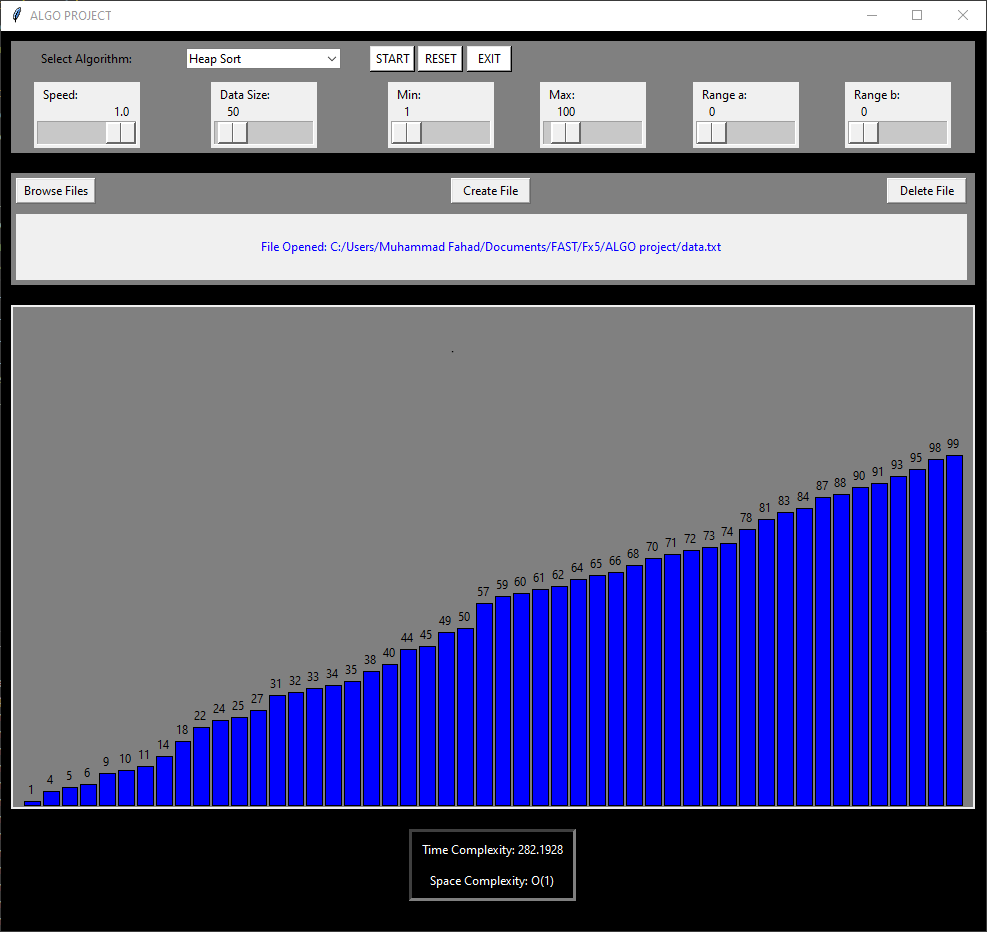
# Below the graph, the time and space complexities of each algorithm can be viewed. Before sorting the general complexities are given in the form of Big O. After choosing a file and initiating the sorting process, new complexities are calculated using the size of data read.

# Results and Discussion:

# Data stored in the input text file:



# The data after sorting is represented as follows:



# Conclusion:

# The project allows us to represent how each algorithm behaves for a set of data. Some are quicker than others, like counting, radix, quick and heap sorts, while others like bubble sort are extremely slower. This can be easily concluded by monitoring the time complexities of every algorithm.

# References:

# Python Documentation: <https://www.python.org/doc/>

# Tkinter Documentation: <https://docs.python.org/3/library/tkinter.html#>