DESIGN AND ANALYSIS OF ALGORITHM CS-2009

**Report of sorting algorithms**

A picture containing text, gear, metalware

Description automatically generated

SUBMITTED TO:

Miss Anaum Hamid

GROUP MEMBERS:

YUSRA ADAM **20K-0207**

ALIZA HAFEEZ JAHAN **20K-0222**

**ABSTRACT:**

The need of different sorting algorithms to implement them in real life scenarios. Analyzing these sorting algorithms with the help of visualizations to better understand their needs in different situations as per their complexities and efficiency. Providing a user interface here user can select file from the system and can implement any sorting algorithms depending on the requirements. Various sorting algorithms are implemented providing the concept of iterative approach, divide and conquer approach, heap data structure approach. User can also check how many operations any sorting algorithm is taking so that he can look for another algorithm which is better in terms of efficiency and complexity.

**INTRODUCTION:**

Sorting algorithms are used for solving different problem-solving concepts in programming as sorting is a fundamental concept in algorithm designing. Sorting algorithms can be implemented to solve any particular problem more easily like searching and there are certain algorithms that uses sorting technique to further proceed with the process. Sorting techniques gives ideas regarding various approaches that can be used depending on the problem requirements.

**PROGRAMMING DESIGN:**

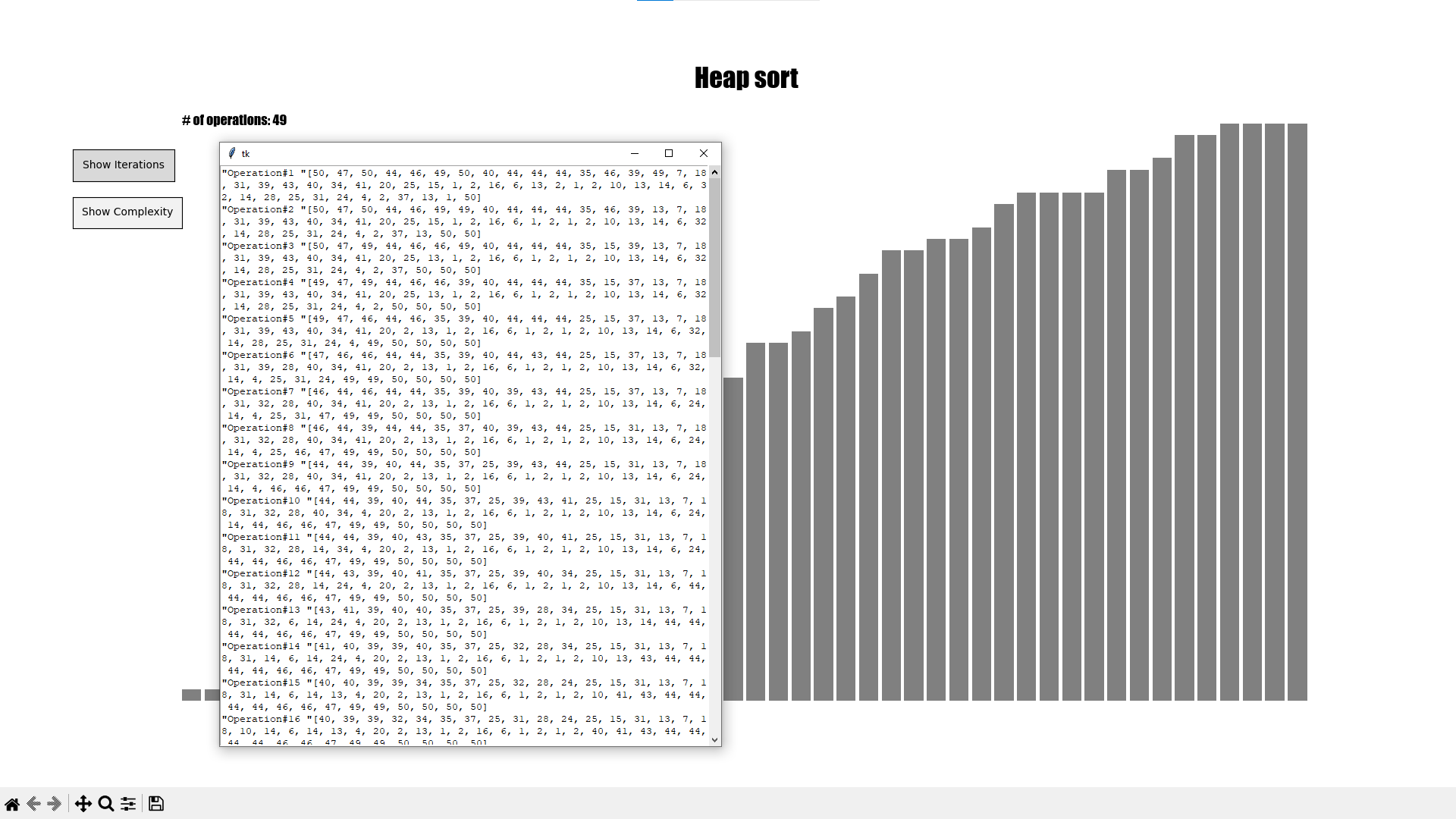
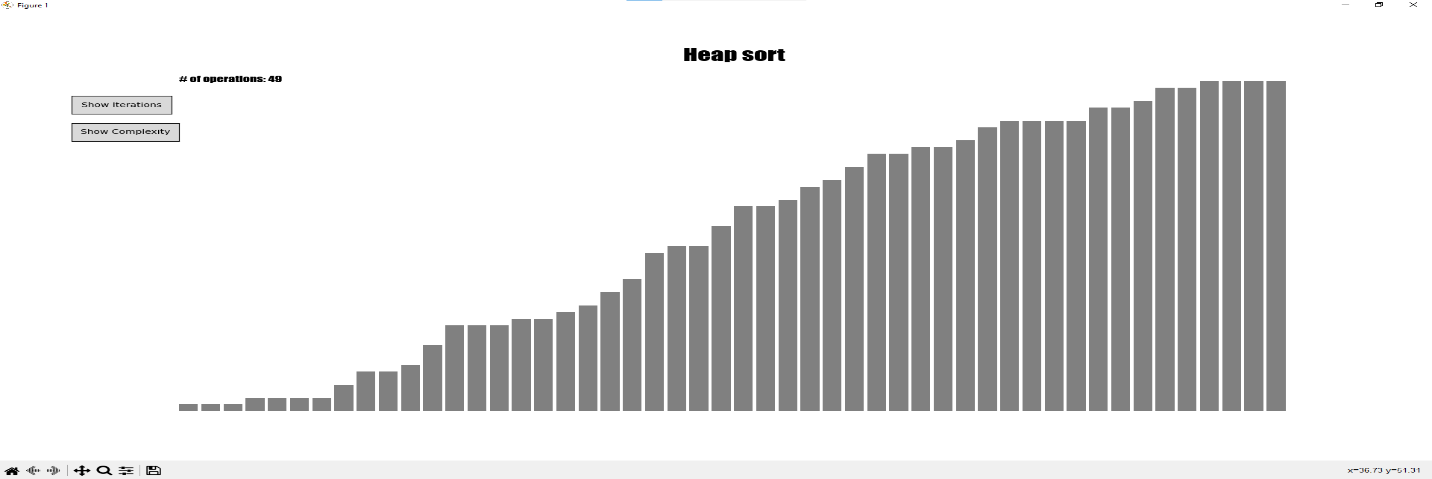
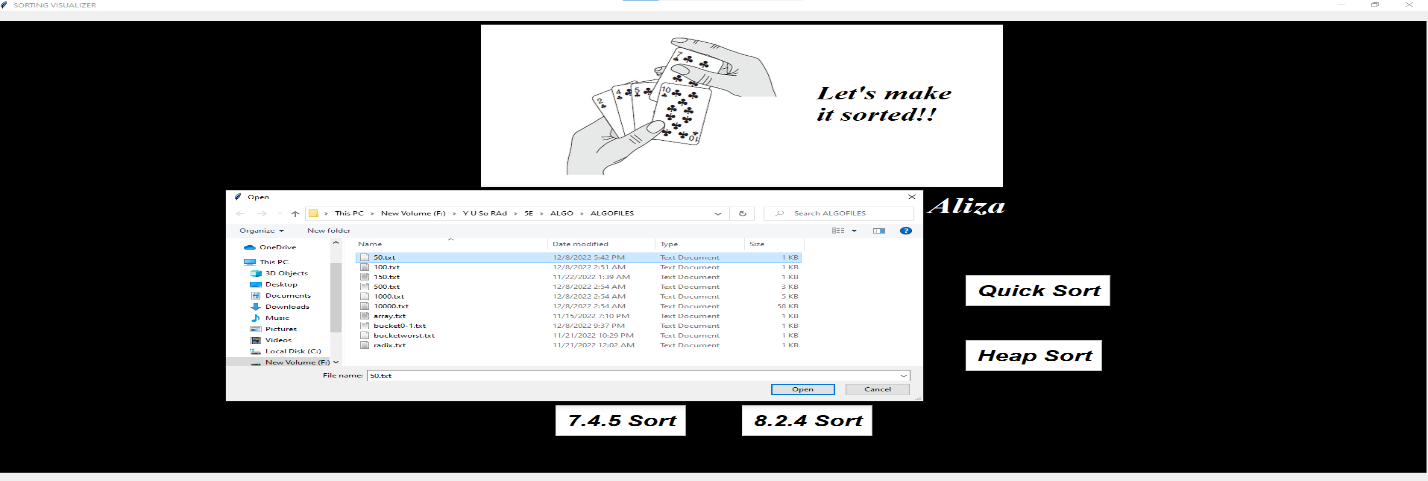
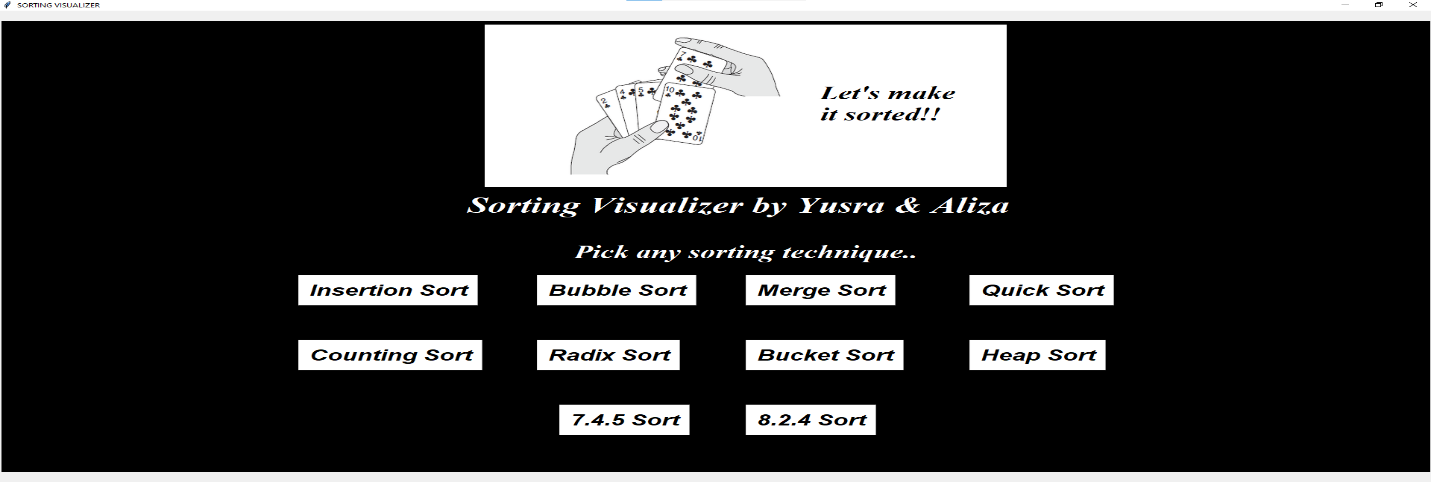
For the implementation of sorting algorithms, we have used Python language. Implemented the front-end user interface through “Canvas” by using “Tkinter”, and various buttons for the selection of any one sorting technique have been made by using “tkinter.Button”.

Implementation of sorting algorithms and their visualization is done by redirecting the buttons to different functions which implements the sorting algorithm and also gives the respective visualization. Once the user selects any sorting technique, he will be asked to choose file upon which he wants to apply the sorting technique. After that sorting algorithm will be implemented and user can see the visualizations in the form of bars. This visualization is done through “matplotlib” by using various functions of “matplotlib” library. How many numbers of operations are required to sort the file are also maintained iteration wise and is displayed on the user interface.

**EXPERIMENTAL SETUP:**

The experiment and correction phase are done by testing various files from system which different array size and data type. Firstly, to check whether the sorting algorithm sorts the array correctly and secondly to make sure that the operations performed are correct. User can also view the changes in the array after comparison or swapping iteration wise by clicking a button provided in the user interface.

**RESULTS AND DISCUSSION:**

For a demonstration, performing heap sort on an array of 50 length 48. The number of operations required by heap sort are 47 as the time complexity is O(nlog(n)). If this same array would be sorted through insertion sort, it would require a greater number of operations as insertion sort compares each number with other using brute force approach and would take (n2) operations. Using counting sort, the number of operations would be the same as array length because it works with O(n+k) complexity.

**CONCLUSION:**

Concluding it all, implementation of various sorting algorithms has been done to understand the need of these all separately as, one is better than the other in terms of efficiency and complexity. In which situations any sorting algorithm can be implemented like to sort numbers with decimal places we can use bucket sort, problems that require solutions to subproblems can use merge sort.

**REFERENCES:**

* <https://towardsdatascience.com/5-sorting-algorithms-in-python-c7ece9df5dd6>
* <https://github.com/nrsyed/sorts/tree/master/python>