

 Department of Informatics

School of Computer Science

UNIVERSITY OF PETROLEUM & ENERGY STUDIES,

DEHRADUN - 248007, Uttrakhand

**End-Term Report**

**For**

**SORTING VISUALIZER**

**7th December 2022**

**Mentor: Mr. Amit Singh**

**Year of Graduation: 2024**

Prepared by

|  |  |  |
| --- | --- | --- |
| **Specialisation** | **SAP ID** | **Name** |
| B.Tech. CSE (AI-ML) | 500082404 | Aditi Prasad |
| B.Tech. CSE (AI-ML) | 500083748 | Somyansh Avasthi |
| B.Tech. CSE (AI-ML) | 500085073 | Ananya |
| B.Tech. CSE (AI-ML) | 500082822 | Agnibha Dutta Ray |

**Table of Content**

|  |  |
| --- | --- |
| **S.No.** | **Title** |
|  | Project Title |
|  | Abstract |
|  | Introduction |
|  | Literature Review |
|  | Methodology |
|  | Results |
|  | Conclusion |
|  | Acknowledgement |
|  | References |

**Project Title: Sorting Visualizer**

Visualising all different kinds of sorting algorithms along with time complexity and applications (using python language).

**Abstract**

This paper outlines a study that tested the benefits of animated sorting algorithms for teaching. To visualize all different kinds of sorting algorithms, a web-based animation application was constructed. A visualization of data is implemented as a bar graph, after which a data sorting and algorithm may be applied. The resulting animation is then performed either automatically or by the user, who then sets their own pace. Basically, sorting is a process in which data/items are arranged in an increasing or decreasing manner. We use many techniques and strategies to sort data depending on how many things are there to sort, size of the items and how frequently we use that item. We sort the database to make it easier for the user to look at it and search for an item in a much easier way.

Sorting is an important aspect in computer language and programs, naturally there has been an intensive study of this topic and many algorithms have been devised for our preferred needs. Some algorithms are faster but unstable and some are slower but can sort a huge heap of data perfectly. It is necessary to choose the best algorithm for the required sorting as no one would use quicksort for sorting a database by date even though it is the most used algorithm in all computer programming languages. As there are still more unsolved problems relating to sorting, there are still new algorithms being developed and refined for special purpose applications which will help us in the near future.

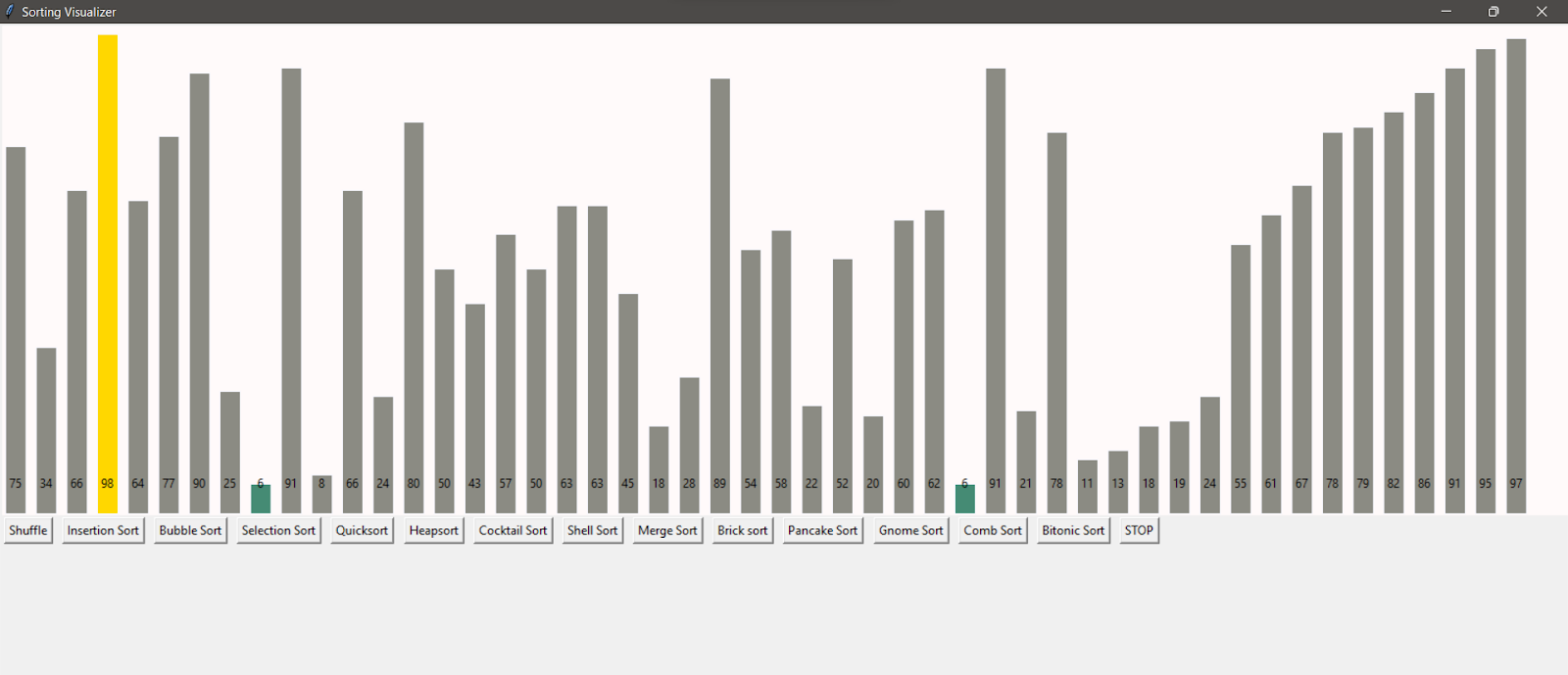
**Keywords**: Sorting Algorithms like Selection Sort, Merge Sort, Bubble Sort, Insertion Sort, Quick Sort, Heap Sort , Cocktail Sort , Pancake Sort , Shell Sort  etc , Time complexity and applications.

**Introduction**

While using various sorting algorithms like bubble sort, selection sort, insertion sort, quick sort the user sometimes fails to understand the core idea of a particular algorithm.  One of the reasons for the same is the inability to visualize how the algorithm works. Due to the significance of the ability to visualize algorithms, this project is aimed at providing a platform which can be utilized to arrange the user’s data in a systematic and optimized order.

This project is to sort an array and visualize the sorting process.The output will first show a non-sorted array in the form of a bar graph , then it will visualize the process in which the array is being sorted by swapping the bar graph position according to the sorting algorithm. We will do our project by using python language.This project will visualize the sorting process and users will see a visual appearance of algorithms implementation and how they work and sort the arrays .It will decide the appropriate algorithm based upon the array size, the time and space complexity of various sorting algorithms.This software uses both object-oriented and functional programming paradigm show it organises the code.Before the final phase of development, the design was almost completely functional, where only three objects were used: one to control the canvas that displayed the animation, another to represent a piece of data, or “bar”object (blue rectangle with dynamically changing height and position), and a final one to represent the positions that each bar moved to, or “pos” objects. Although this incorporated several function calls.To update the bars to a fresh data configuration, then in Order, Reverse, and Random buttons iterate fast across the array. The bar object is related to the pos object (which is short for position). The region on the canvas that is updated when the SortArea event fires is an x-y pixel coordinate grid.This way we get the visualization and understand the working of this swap function after enter the size of the array , maximum speed and randomised value and then clicking on the submit button ,then click on any sort and see the arrangement of data in  a visualising mode .

After achieving visualization we will get so many things and knowledge about the sorting algorithms and the working and it’s really helpful for the user. It’s really a great animation tool for everyone for visualizing in many fields of work.



**Figure 1:** Main Window of Application

**Literature Review**

The use of sorting algorithms as visual aids has been the subject of numerous studies and papers throughout the years. Some offer thorough explanations on how to make animations and conduct statistical analyses, while others concentrate on various methods designed to improve understanding of an animation of a similar nature. When we say similar, it means that while the animation utilized in two studies might be comparable, the differences in analysis were focused on how the algorithm was applied.

Kerren and J. Stasko's paper "Algorithm Animation" provides a step-by-step tutorial for examining the environment, means, and the available coding techniques to apply a sorting animation. There are many various kinds of software that may be used for animation, and one of which is BALSA, which invented the intriguing event technique.

The paper “Do Algorithm Animations Assist Learning? An Empirical Study and Analysis” provides a thorough examination of student’s reactions and responses to sorting animation. It provides an in-depth view for it. In order to compare the outcomes of students who used solely textbook resources to those who also had access to animation and visualization for support, post-test research was conducted. Each student group took the identical post-test, which offered a thorough examination of the subject. The study didn't find any conclusive evidence that an animation would benefit students, with the subject matter much. The article came to the conclusion that while visualizing algorithms may sound appealing, they may not actually produce the desired outcomes.

But since Kerren and Stasko's study focuses on the learnability of data structures by utilizing animation (trees and heaps), not the algorithms itself, their findings are not entirely applicable to the project topic outlined in this synopsis.

Stasko's "Using Student-Built Algorithm Animations as Learning Aids" is another article we came across. It firstly cited the identical error found in the previous article that it doesn’t benefit students that much. However, in an intriguing turn of events, students were given instructions to create the animations themselves, as opposed to using some previously created to aid with understanding. The visual programming tool Samba was explained to the students. Following some introduction assignments were given to them to help them become proficient with the software opportunity to animate and visualize an algorithm.

Positive feedback and a general improvement in comprehension of how the algorithms operated were found in the results. Also, when using the concepts in an animation, several students discovered that they had erroneous and wrong ideas about the subject.

After searching some online videos of algorithms visualization, we got to know about the idea where the animations have sound. In his blog, Timo Bingmann outlines his animation experiments using music to highlight the variations between each sorting method audibly.

Until the animation was complete, a sound would be produced after each iteration of the outer loop. The animation sequence would be heard as a result.

**Methodology**

In order to sort the data/items given by the user by using the different sorting algorithms such as bubble sort, insertion sort, selection sort, heap sort, etc with the functionality of speed control and array size control and then visualizing the sorting process in different sorting algorithms and the final result on the screen in the form of sorted bar graphs and also displaying the time complexity and applications, we have used python 3.9 version and used python programming basic concepts for the coding in python in VS code platform .

The user interface is a basic graph layout which shows the dataset as a bar graph. There are buttons which ask from the users which type of sorting should be done.

We have used programming standards such as - name conventions for variables and functions, indentation, simple coding methodology, sorting of array, creating and generating window using Tkinter module in python. Visualising the sorting process by using animations.

In this sorting visualizer project, we have used python libraries such as Tkinter, random, time, math:

-   *Tkinter* is the standard GUI library for Python.

-  *Random module* is an in-built module in python which is used to generate random numbers.

- *Time module* in python provides many ways of representing time in code, such as the objects, numbers, and strings. It also provides functionality other than representing time, like waiting during code execution and measuring the efficiency of the code.

- *Math* in python is a built-in module that provides standard mathematical constants and functions.

With use of these and coding concepts of python our project visualizes the sorting process of selection sort, insertion sort, bubble sort, quick sort, heap sort, cocktail sort,etc.

The workflow and thorough design processes for the proposed project are outlined below -

1.Taking inputs from the user as normal input (random dataset) or creative input (getting data by the distance of the line) .

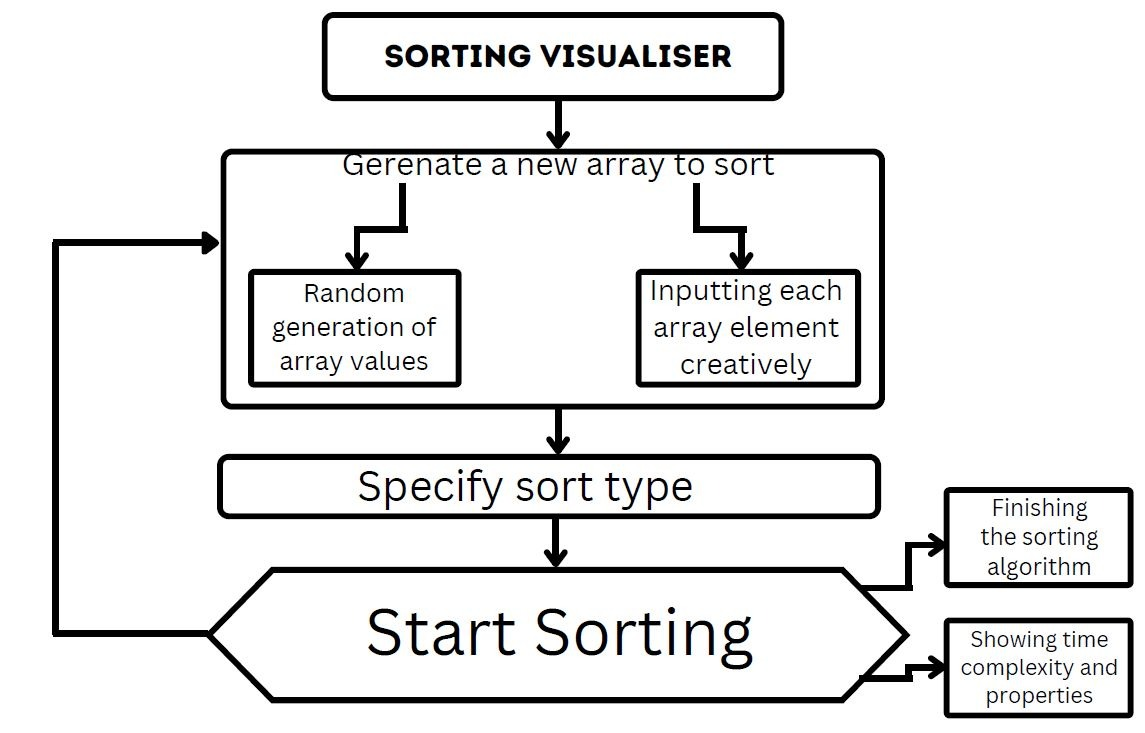
2. The dataset is converted into a bar graph and is shown to the user .

3. User selects a sorting algorithm provided by the buttons to sort the dataset .

4. After selecting the sorting method, users can see and understand how that sorting is working and also their time complexity and applications displayed on the screen .

5. Users can stop the program by selecting the stop button or pressing the shuffle button if the user wants to see another sorting method .

**Flowchart for the sorting visualizer –**

****

The following sorting algorithms have been used in the project –

1.Insertion Sort - The array is divided into sorted and unsorted parts. Values are chosen and assigned to the appropriate positions in the sorted part of the data from the unsorted part.

2.Bubble Sort - the simplest sorting algorithm, which continually switches nearby components if they are out of order. Large data sets should not be used with this approach due to its high average and worst-case time complexity.

3.Selection Sort - An array is sorted by continually selecting the lowest member from the unsorted portion and moving it to the beginning while taking ascending order into consideration.

4.QuickSort - It is a Divide and Conquer algorithm. It chooses an element to act as a pivot and divides the supplied array around it. There are numerous versions of QuickSort that select pivot in various ways.

5.Cocktail Sort - It is a variation of Bubble sort.

The Bubble sort algorithm always transfers the largest element to its proper position in the first iteration, followed by the second-largest in the second iteration, and so forth. Cocktail Sort alternately moves through an array in both directions. Cocktail sort is effective for huge arrays since it skips the extra iterations.

6.ShellSort - For a high value of h, we use Shell sort to produce the array h-sorted. H is gradually decreased until it reaches a value of 1. If all sublists for every element in an array are sorted, the array is said to be h-sorted.

7.Heap Sort - A comparison-based sorting method based on the Binary Heap data structure is called heap sort. It is comparable to the selection sort in which the minimum element is initially located and placed at the start. For the remaining components, follow the same procedure.

8.Merge sort - It is a sorting algorithm based on the Divide and Conquer paradigm . This algorithm divides the array into two equal parts, which are then combined in a sorted way.

9.Pancake sort - Pancake sort is a sorting algorithm in which the only allowed operation is to "flip" one end of the list. It is inplace but not stable .

10.Bitonic Sort - A parallel sorting method called bitonic sort performs comparisons in O(n2log n) time. The items are compared in a predetermined order that must not be dependent on the data being sorted, so even though there are more comparisons than in any other common sorting algorithm, it performs better for the parallel implementation. The predefined sequence is called the Bitonic sequence.

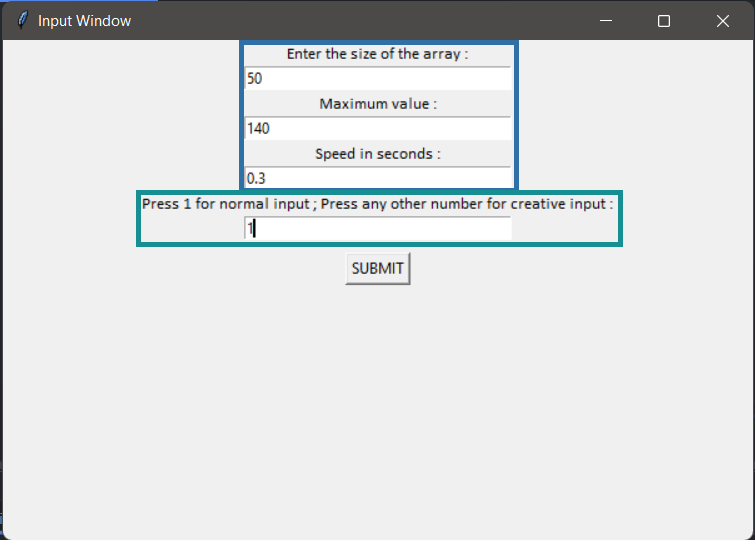
11. Gnome Sort - Gnome sort builds a sorted list one element at a time by swapping each item into its rightful position.

12.Comb Sort - Comb Sort primarily improves Bubble Sort. Always compare adjacent values when using a bubble sort. Thus, each inversion is eliminated one at a time. When compared to Bubble Sort, Comb Sort performs better since it uses gaps larger than 1. The gap has a high initial value and gets smaller by a factor of 1.3 with each iteration until it is equal to 1.

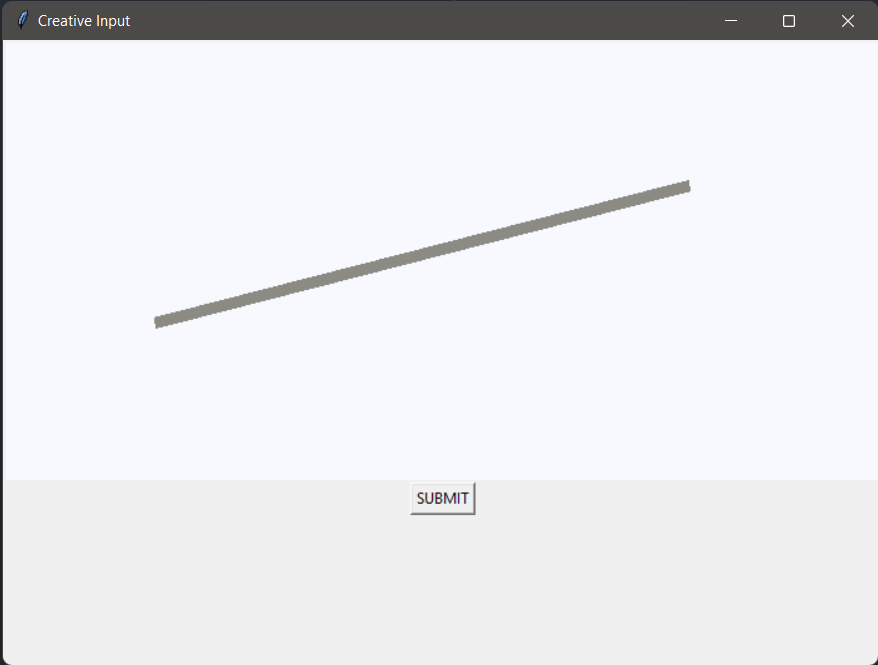
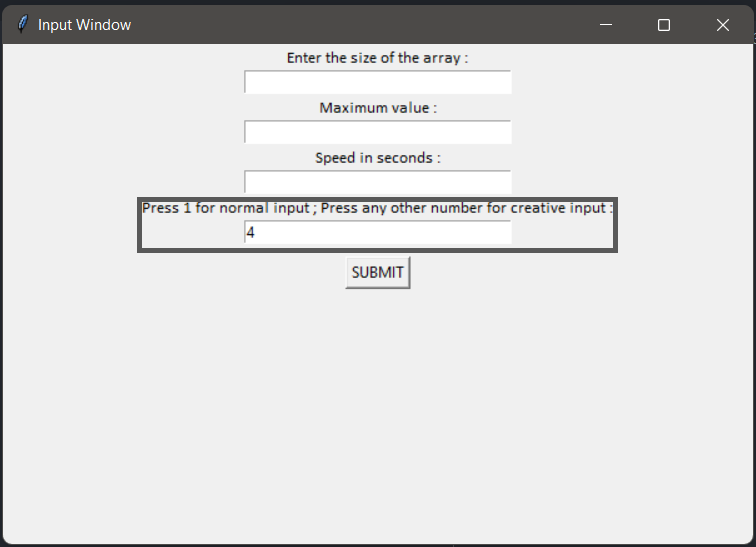
**Result**

The result obtained from the code firstly allows the user to choose the input method if anyone wants to perform. On running the code the user has a choice to either to input the array values randomly or by inputting each value manually.

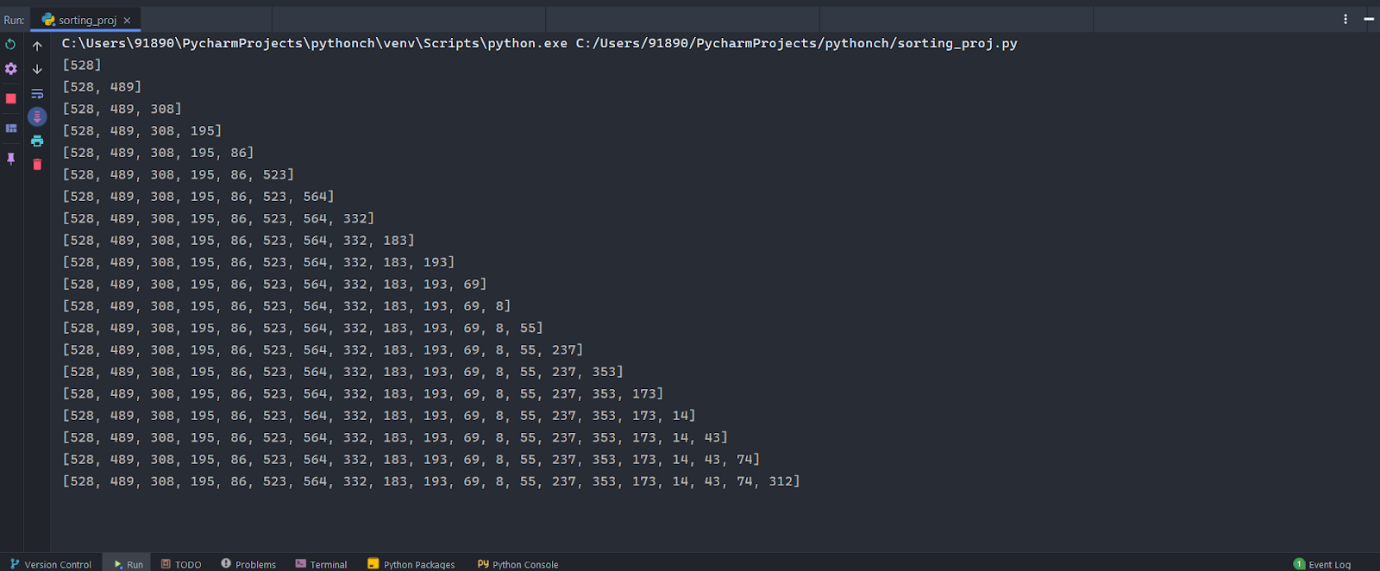
For the normal (randomised) input method, the output allows the user to firstly input the array size, secondly the set the range upto which the values has to be marked, and lastly there is an attribute where the user can set the speed of the sorting for the convenience of the user. For confirming that the user wants the randomized input option he has to enter 1 in the last option.



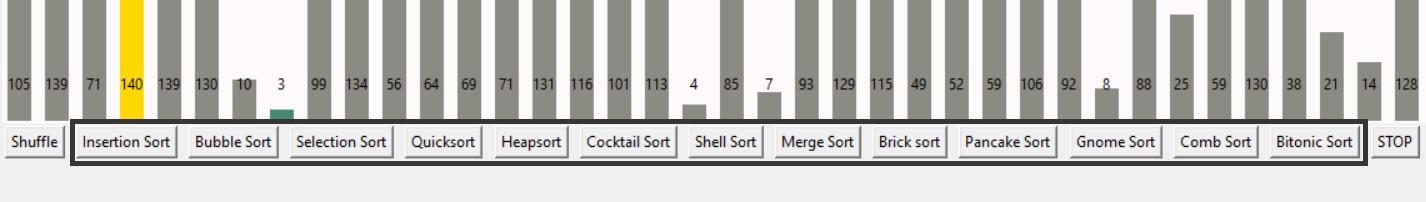
For the creative input we have allowed the user to enter the array elements manually. The user has the convenience to enter the elements according to the needs. For making it more interesting the input is fetched by using the Euclidean distance formula. The user has to use mouse clicks to make lines on the input window. The longer the line is drawn the larger the value is fetched.



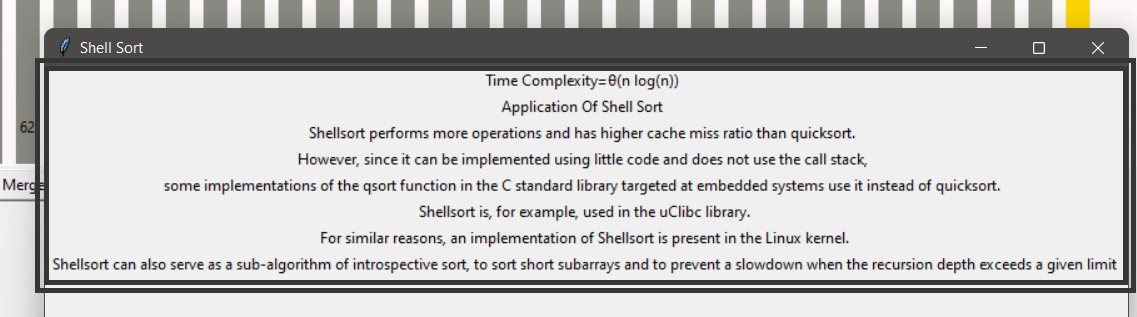
Here along with the array element values displayed on the sorting window, the user is also provided with the element values on the terminal of the code.



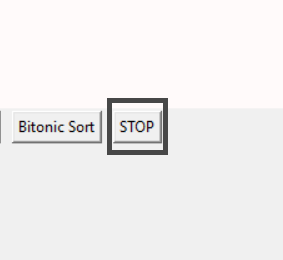
After inputting the values, the user has the option to choose from the several sorting algorithms from which anyone wants to sort the data.



Once done with the sorting, the user is provided with a window where all the attributes of that particular sorting algorithm are displayed. The attributes mainly include time complexity of the algorithm and application of that algorithm.



Lastly, the user can stop the program using the STOP button.



**Conclusion**

All in all, the project focuses on one of the major aspects of the IT industry and helps in making and visualising the data sorted using several sorting algorithms. Also this application can be used in various software domains which can be associated with banking sector, teaching sector, the finance sector, e-commerce sector, etc.

**Acknowledgment**

We wish to express our deep gratitude to our mentor Prof.Amit Singh, for all advice, encouragement, and constant support he hasgiven us throughout our project work. This work would not have been possible without his support and valuable suggestions.We sincerely thanks our cluster head, Prof. (Dr.) T.P Singh, for his great support in doing our project at SOCS. We would like to thank all faculties from UPES for their help and constructive criticism during our project work.

**References**

 [1][Sorting Visualiser  dev community](https://dev.to/dharshak/sorting-visualizer-2bga)

[2][SORTING ALGORITHM VISUALIZER-Shubham Nath\*1, Jatin Gupta\*2, Abhinav Gupta\*3, Teena Verma\*4](https://www.irjmets.com/uploadedfiles/paper/volume3/issue_7_july_2021/14236/1628083553.pdf)

[3]<https://digitalcommons.ric.edu/cgi/viewcontent.cgi?article=1129&context=honors_projects>

[4]<https://www.theseus.fi/bitstream/handle/10024/507342/Bikram_Karki.pdf;jsessionid=898A8374114942395546323B3B139DB9?sequence=2>

[5]<http://faculty.tamuc.edu/dcreider/csci520/Note520/Note%206.htm>

[6]<https://www.interviewkickstart.com/learn/time-complexities-of-all-sorting-algorithms#:~:text=a%20simplified%20form.-,Space%20Complexity%3A,complexity%20is%20O(1).>

[7]<https://panthema.net/2013/sound-of-sorting/>

[8]<https://dl.acm.org/doi/10.1145/169059.169078>

[9]<https://www.geeksforgeeks.org/time-complexities-of-all-sorting-algorithms/>