PBL

MINI PROJECT

ON

"SORT VISUALIZER"

BACHELOR OF ENGINEERING

Computer Engineering

SUBMITTED BY

Suraj Patil (21049)

Khushi Raval (21053)

Shreya Sandhanshive (21056)

Piyush Shah (21059)

Dhruv Sawant (21058)

Dhruv Saini (21055)

Progressive Education Society's
Modern College of Engineering
Department of Computer Engineering
Shivajinagar, Pune – 411005

2022-2023

Progressive Education Society's

Modern College of Engineering,

ShivajiNagar, Pune- 411005.

Certificate

This is to certify that the project report entitled

"**SORT VISUALIZER**"

Submitted by

Suraj Patil (21049)

Khushi Raval (21053)

Shreya Sandhanshive (21056)

Piyush Shah (21059)

Dhruv Sawant (21058)

Dhruv Saini (21055)

Academic Year: 2022-2023

This is to certify that students from Second Year Computer Engineering have successfully completed their mini project work of **Project Based Learning - II** under the guidance of **Ms. Ruchita Kulkarni** at P.E.S. Modern College of Engineering in the partial fulfillment of the S.E. of Engineering Degree in Computer under of Savitribai Phule Pune University, Pune.

Ms. Ruchita Kulkarni
Internal Supervisor
(Computer Engineering)

Dr.Mrs . S. A. Itkar
Head of Department
(Computer Engineering)

INDEX

CHAPTER 1

INTRODUCTION

1.1 ABSTRACT

The main goal of the thesis was to create a teaching support software with visualization of the most known sorting algorithms and their variations. The application supports a graphic visualization of selected algorithms on randomly generated or manually created array, step-by-step execution possibility.

Our project is a single web application built using React.js that serves as a sorting visualizer. The app allows users to visualize various sorting algorithms by manipulating blocks. Each block represents a data element, and users can observe the sorting process as the blocks move and rearrange themselves. The project provides an interactive and engaging way for users to understand and learn about different sorting algorithms.

To visualize six sorting algorithms, a web-based animation application was constructed. A visualization of data is
Implemented of block moments, after which a data sorting and algorithm may be applied.

Keywords: Sorting Algorithms, Quick Sort, Selection Sort, Merge Sort, Bubble Sort, Insertion Sort, Heap Sort.

1.2 PROBLEM STATEMENT

The existing methods for understanding sorting algorithms are not effective. Traditional explanations and text-based representations make it difficult for users to grasp the step-by-step process of sorting. Without a visual and interactive tool, users struggle to comprehend the mechanics and performance of different sorting algorithms. Therefore, there is a need for a simple and visually engaging sorting visualizer to enhance understanding and proficiency in sorting algorithms.

1.3 MOTIVATION BEHIND PROJECT

The motivation behind the Sorting Visualizer Project is to address the challenges and limitations in understanding sorting algorithms. By developing a sorting visualizer, we aim to provide a tool that promotes interactive learning and comprehension. The visualizer will offer a clear, step-by-step representation of sorting algorithms, enabling users to observe and understand the sorting process in real-time. The project's motivation is to empower users with a visual and interactive tool that enhances their understanding, experimentation, and practical application of sorting algorithms.

1.4 OBJECTIVE

The main objective of the thesis project is to create a web application as a visualization tool. A single-page web application built using modern technology that will visualize the flow and logic of various sorting algorithms. The UI will contain options to select one of the sorting algorithms which were implemented and several items or elements in the data array, control buttons to start, pause, navigate to previous or next steps along with an option for sorting speed and color mode. The data array of the selected size will be filled in with randomly generated unique values. The data set is represented as a vertical bar with the height of their respective values. After the sorting is started, the stepwise arrangement of data in ascending order based on their value will be visualized .

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

Nowadays sorting algorithms are widely used in computer software. For example, if you open file explorer on your PC, you may see files sorted in different ways. Searching in sorted data is more efficient than in not sorted ones. Students of computer science start learning different algorithms in the first year of studies and sorting algorithms are among them. Since I faced the problems of sorting during the course of algorithm design in the first year of my studies, there is an understanding that the visual representation is a vital part of the studying process. During working on the thesis it was very exciting to learn different techniques of sorting algorithms into the depth. The main goal of the thesis was to create a program which would serve as a tool for understanding how most known sorting algorithms work. There was an attempt to make the best possible user experience. The demonstration software is made in a user-friendly and easy-to-use style. To gain maximal benefit from learning you can try each sorting algorithm on your data. The text of the thesis describes principles of the most known sorting algorithms which are demonstrated in the computer program. It might be used as a source for learning algorithms by students. Also, the program might be easily used as a demonstration by lecturers and tutors during classes. Besides, there is programmer documentation and user guide to the provided software. Readers of this text are expected to have some programming experience to know basic data structures such as arrays, lists, trees and understand recursive procedures. Also, knowledge of some simple algorithms and their implementations could be helpful. In order to understand the topic better, knowledge of linear algebra and calculus is involved

2.2 NEED OF THE SYSTEM

The proposed system involves the simulation of the different type of sorting algorithms codes. The scope has its limitations. Only 6 types of sorting algorithms codes are created which are bubble sort, insertion sort, selection sort, heap sort, merge sort and quick sort. Only the software application development began with desktop applications used in this project, it can be used on standalone personal computer only. Once the synthesize and simulation of the codes for the software application have been run, the following animations will show how successfully data sets from distinct unsorted data can be sorted using distinct algorithms.

2.3 Synopsis

The Sorting Visualizer project aims to create an interactive web application using React.js that visually demonstrates various sorting algorithms. The project provides a hands-on learning experience for users to understand and observe the behavior and efficiency of different sorting techniques.

The project includes the following key components and features:

Sorting Algorithms: Implementations of popular sorting algorithms such as Bubble Sort, Insertion Sort, Selection Sort, Merge Sort, Quick Sort, and more. Each algorithm will be designed to work with an array of numbers and will be optimized for performance and accuracy.

Visualization: The sorting process will be visualized using dynamic and interactive charts or animations. The web application will display the initial array and update it step by step to illustrate how the sorting algorithm progresses and rearranges the elements.

User Interface: Develop a user-friendly and responsive user interface that allows users to interact with the sorting visualizer. Users can control the speed of the visualization, choose specific sorting algorithms, and input custom array sizes or generate random arrays for sorting.

Comparison and Analysis: Provide a comparison feature that allows users to compare the performance of different sorting algorithms. Users can observe the differences in algorithmic complexity, execution time, and number of comparisons or swaps required for each algorithm.

Educational Resources: Include educational resources such as tooltips, explanations, and descriptions of each sorting algorithm. These resources will help users understand the principles behind each algorithm, their time complexity, and their best and worst-case scenarios.

Responsive Design: Ensure that the web application is responsive and compatible with different screen sizes and devices. Users should be able to access and use the sorting visualizer on desktops, laptops, tablets, and mobile devices.

The Sorting Visualizer project offers an interactive and visually engaging platform for users to explore sorting algorithms and gain insights into their efficiency and behavior. It serves as an educational tool for students and enthusiasts studying computer science or algorithms and provides a practical way to learn and understand sorting techniques.

2.4 LITERATURE SURVEY

SR.NO	AUTHOR NAME	Research Title	YEAR	Research Objective	Key Findings
1.	Authors: Johnson Smith Lee	"Comparative study of Sorting Algorithms for Educational Purposes"	2017	To compare and evaluate different sorting algorithms in terms of time complexity space complexity, and suitability for educational purposes.	Bubble, Insertion, Selection Sorts are simple and easy to Understand, making them Suitable for educational Purposes. However, more efficient algorithms like Quick , Merge Sorts are better for large data sets.
2.	Authors: Chen Huang Wang	"Visualization Techniques and Sorting Algorithms"	2018	To explore Visualization technique for sorting algorithms and their effectiveness in enhancing understanding and learning.	Animated bar charts and color-coded comparisons help user to understand sorting algorithms better. Interactive features, such as step by step execution and adjustable speed, improve user engagement and learning experience.
3.	Authors: Adams J Thompson Brown M	"Effectiveness of Sorting Visualizations in Computer Science Education"	2019	To investigate the effectiveness of sorting visualizers as teaching tools in computer science education.	Sorting visualizers enhance student understanding of sorting algorithms and improve their problem solving skills. Visualization helps students to grasp the concepts of algorithmic complexity and their impact.

CHAPTER 3

DESIGN AND MODELLING

3.1 System Architecture

The back-end code is comprised of HTML5, CSS, and JavaScript. All three types of code are contained in one .html file and can be run solely from this file. One of the advantages of HTML 5 is that it is not necessary to include different types of web languages in a single file. Therefore, each type could have been separated, making a total of three files (plus the miscellaneous sound and image files). This is good practice for readability and keeping related code together.
As you can see, there are no major components besides the three coding languages. Most websites have tools or scripts that require a server on the back-end (like PHP), but it is not necessary in this case since JavaScript runs right in the user's browser. HTML5 and CSS are used for the interface. The HTML5 communicates with the JavaScript code and vice versa to launch the appropriate algorithms and update the interface accordingly, as seen with a single, bidirectional arrow. Throughout the project, the code for the HTML5 and CSS did not change much. As the JavaScript was modified from a functional programming focus to a more object-oriented one, the parts of the HTML5 that did change were the function calls for each button. All of the back-end interaction is abstracted to the various buttons for selecting algorithms and running the animation.

3.2 Scope of the System

Develop an intuitive user interface for the sorting visualizer. Implement various sorting algorithms accurately. Provide real-time visualization of the sorting process. Include interactive features like speed control, pausing, and stepping.

Allow customization of input data size and simultaneous algorithm comparison. Ensure scalability and performance for large data sets. Create documentation and a user guide for effective usage.

Conduct testing and debugging to ensure correctness. Deploy and distribute the sorting visualizer for user access.

Interactive features, such as step by step execution and adjustable speed, improve user engagement and learning experience.

CHAPTER 4

TECHNICAL SPECIFICATION

4.1 SOFTWARE REQUIREMENTS:

Operating System: The sorting visualizer should be compatible with popular operating systems like Windows, macOS, and Linux.

Programming Languages: The visualizer can be developed using languages such as Python, JavaScript, or Java, depending on the chosen platform and technology stack.

Development Frameworks: Utilize frameworks like Flask, Django, React, or Angular for web-based visualizers, or PyQt, Tkinter, or JavaFX for desktop applications.

4.2 HARDWARE REQUIREMENTS:

Processor: A modern processor (e.g., Intel Core i5 or equivalent) that can handle the computational demands of running sorting algorithms efficiently.

Memory (RAM): Sufficient RAM (e.g., 4GB or higher) to accommodate the data sets used for sorting and ensure smooth execution of the visualizer.

Display: A monitor with a suitable resolution (e.g., 1280x720 or higher) to provide a clear and visually appealing representation of the sorting visualizer.

4.3 IMPLEMENTATION

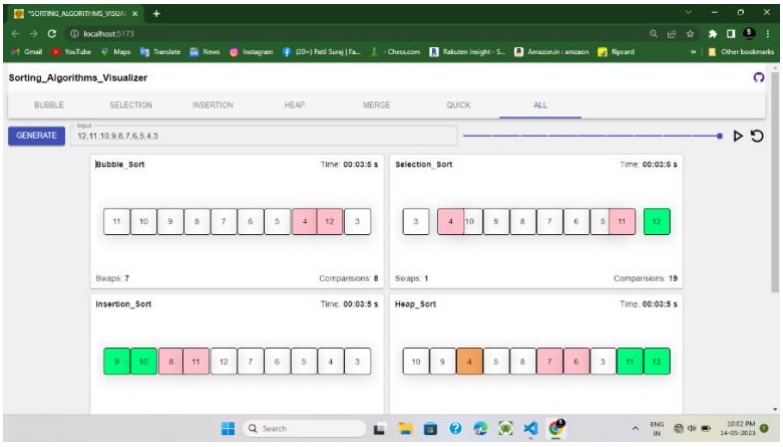
The model is made up of one item, known as the sorter. This object houses the algorithm's code divided into methods. Start method centralizes on an integer constant and uses it to order the algorithm's possible algorithms. This object is directly controlled by the four sorting algorithms shown on the user interface as "Selection Sort, Bubble Sort, Insertion Sort, and Merge/Insertion Sort." The sort algorithm method is invoked as the user selects a sorting technique and clicks on one of the sort algorithm buttons. Then, when the algorithm sorts the data, a trace is created. The steps array, which contains all the movements in the animation, is a two dimensional integer array that is available to methods on the web user interface. The “Start,” “Stop,” and “Step” buttons function as controllers for which sub-array will be displayed on the canvas, after the computational back-end has completed the tracing. If this loads before the user has selected "Start" or "Step", then this represents a user action. When you click the “Step” button, the next step on the canvas loads and animates. An animated sequence of a single bar moving across the others is produced because the timer continually redraws the bars. There's no “Start” button, only a “Step” button that is set to go off on a timer. Using a two-dimensional array gives you the option to view the sorting algorithm's stages within the View.

The process of adding an algorithm is similar to writing down the trace of the new algorithm, which is then saved in the same location. To complete the algorithm's walkthrough, the View will cycle

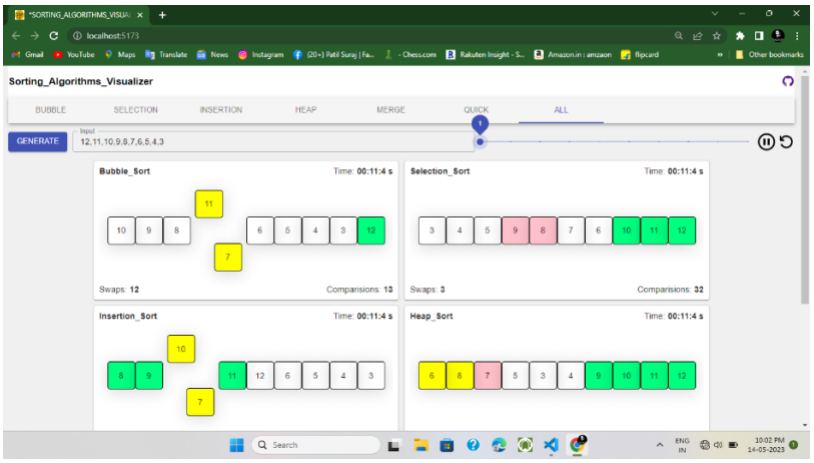
through the data and update the bars in the bar graph to show how the algorithm calculated the steps it took. It's important to note that if the algorithm generated a change in the position of a piece of data, the steps are merely recorded. When sorting the pieces of data using Selection Sort, each piece of data is moved to its final and accurate location after one step, whilst the others require numerous steps to get at their final positions. While this sorting method appears to do the most effort compared to other sorting methods, it finishes sorting the most slowly. As a result, the visualization doesn't provide the correct visual impression of the data comparisons, which is one of the most important aspects of sorting algorithms. Two-dimensional arrays do demand more memory than a one-dimensional array. The size of the array is based on the number of steps that are required to sort the data. We may assess the algorithm's space needs by examining how long it takes. In Computer Science, using Big-Oh analysis is the standard way for determining how long something will take. The notation consists of a capital letter O, which represents the worst-case performance of the algorithm in question, followed by a constraint in parentheses that describes the worst-case performance of the algorithm. When O(n2) is calculated, it is the selection sort and insertion sort's Big-Oh, meaning the time complexity grows at a rate proportional to n2, or the number of items in the collection of data squared. Space required for the steps array may or may not be the same as the space needed for the steps array.

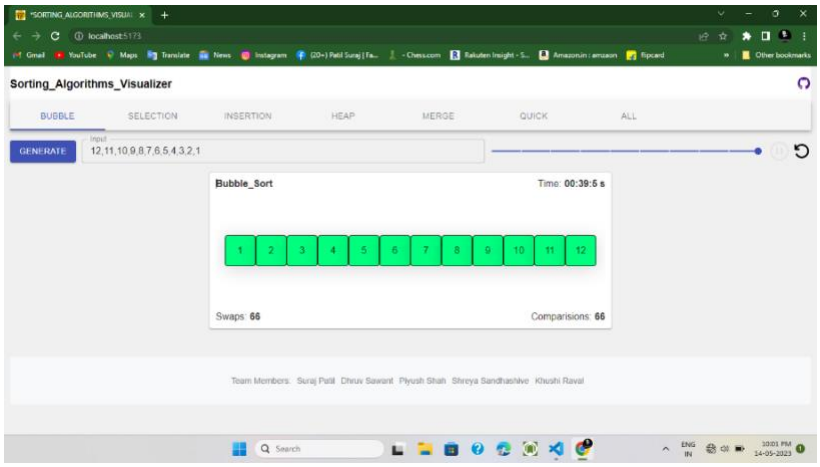
4.4 RESULTS

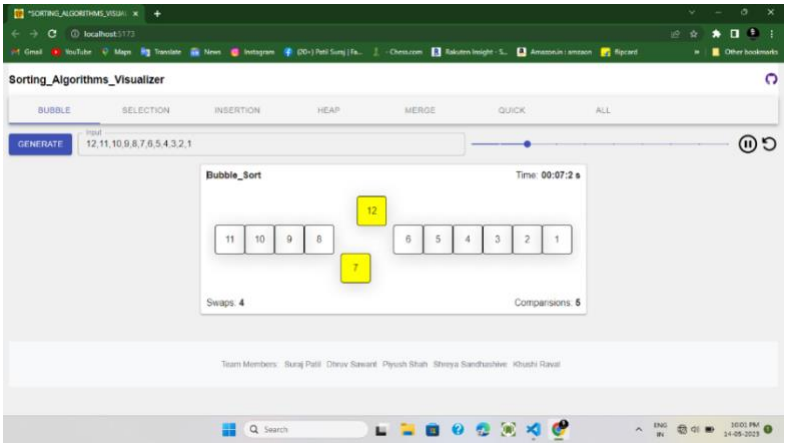
The result of the sorting visualizer project is a fully functional application that allows users to input unsorted data and visualize sorting algorithms. It accurately implements multiple sorting algorithms, provides real-time visualization, and allows user interaction and customization. The project documentation and report explain the purpose, methodology, implementation details, and future improvements of the sorting visualizer



SORT VISUALIZER







CHAPTER 5

CONCLUSION AND FUTURE SCOPE

As the main goal of this project, that was created for teaching support application which visualizes the most known sorting algorithms. The application allows stepping forward and backward through each represented algorithm. User may run sorting on a random or custom array. During the demonstration run, the application visualizes pseudocode and current information about some variables. We tried to create high-quality software with a user-friendly and easy-to-use interface, which could be used by lecturers, tutors, and students. Possible next improvement of the applications is extension it by other algorithms. The first part of the thesis text is more theoretical. It tells about algorithms in general and the algorithms represented in the application. The second part is focused on the application itself.

This web-based animation tool for viewing the following sorting algorithms functions in great part.

In spite of its memory overhead, the feedback given to it was mostly good from the students that worked with it. This is consistent with prior research, which revealed that there was no substantial difference in learning the content. The project holds there is a great need to investigate and produce animated presentations to enhance education in the classroom. This would be excellent, as it would enable a form of comparison study.

CHAPTER 6

ACKNOWLEDGMENT

We would like to express sincere gratitude to all those who have contributed to the successful completion of this Sorting Visualizer project.

First and foremost, we would like to thank our supervisor/mentor [Prof Ruchita Kulkarni] for her guidance, support, and valuable insights throughout the project. Her expertise and encouragement have been instrumental in shaping this project and enhancing its quality.

We would also like to extend thanks to the research papers and articles authors whose work provided valuable insights and information for the literature survey. Their contributions have helped to establish a strong foundation for this project.

We are grateful to the React.js community for developing and maintaining such a powerful and flexible JavaScript library. Their efforts have made it possible to create dynamic and interactive web applications like this sorting visualizer.

We would like to acknowledge our friends and classmates who have provided feedback, suggestions, and assistance during the development process. Their input and support have been invaluable in refining the project and overcoming challenges.

We would like to acknowledge all those who have directly or indirectly contributed to the successful completion of this Sorting Visualizer project. Your support and involvement are greatly appreciated.

Thank you.

CHAPTER 7

REFERENCES

[1] T. Bingmann. “The Sound of Sorting - ‘Audibilization’ and Visualization of Sorting Algorithms.” Panthemanet Weblog. Impressum, 22 May 2013. Web. 29 Mar. 2017.

[2] < http://panthema.net/2013/sound-of-sorting/>

[3] Bubble-sort with Hungarian (“Cs´ang´o”) Folk Dance. Dir. K´atai Zolt´an and T´oth L´aszl´o. YouTube

Sapientia University, 29 Mar. 2011. Web. 29 Mar.2017.

<https://www.youtube.com/watch?v=lyZQPjUT

SORT VISUALIZER