

# **SORTING ALGORITHM VISUALIZER**



**UNIVERSITY OF ENGINEERING  
&  
MANAGEMENT, JAIPUR**

# **Sorting Algorithms Visualizer**

Submitted in the partial fulfillment of the degree of

**BACHELOR OF TECHNOLOGY**

In

**COMPUTER SCIENCE & ENGINEERING (AIML)**

Under

**UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR**

BY

**Arya Kumar Johary, Abhay Prasad**

**University Roll no: 12021002026028,12021002026024**

**University Registration no: 204202100200196, 204202100200192**

UNDER THE GUIDANCE OF

**PROF. SAGARIKA GHOSH, PROF. HRIDAY BANERJEE**

**COMPUTER SCIENCE & ENGINEERING(AIML)**

# Approval Certificate

This is to certify that the project report entitled “**Sorting Algorithms Visualizer**” submitted by **Candidate name** (Roll:**12021002026028, 12021002026024**) in partial fulfillment of the requirements of the degree of **Bachelor of Technology in Computer Science & Engineering (AIML)** from **University of Engineering and Management, Jaipur** was carried out in a systematic and procedural manner to the best of our knowledge. It is a bona fide work of the candidate and was carried out under our supervision and guidance during the academic session of 2011-2015.

---

**Prof. SAGARIKA GHOSH, Prof. HRIDAY BANERJEE**

Project Guide, Assistant Professor (CSE)

UEM, JAIPUR

---

**Prof. Mrinal Kanti Sarkar**

HOD (CSE)

UEM, JAIPUR

---

**Prof. A Mukherjee**

Dean

UEM, JAIPUR

# **ACKNOWLEDGEMENT**

The endless thanks goes to Lord Almighty for all the blessings he has showered onto us, which has enabled us to write this last note in our project work. During the period of our project, as in the rest of our life, we have been blessed by the Almighty with some extraordinary people who have spun a web of support around us. Words can never be enough in expressing how grateful we are to those incredible people in my life who made this project possible. We would like an attempt to thank them for making our time during my research in the Institute a period we will treasure. We are deeply indebted to our project supervisor, Prof. Sagarika Ghosh, Prof. Hriday Banarejee for such an interesting project topic. Each meeting with them added valuable aspects to the implementation and broadened my perspective. They have guided us with his invaluable suggestions, lightened up the way in our darkest times and encouraged us a lot in our academic life.

Arya Kumar Johary, Abhay Prasad

# **ABSTRACT**

This project is a Sorting Algorithm Visualizer, a web application that allows users to visualize different sorting algorithms in real-time. Built using HTML, CSS, JavaScript, and the React JS framework, the web application provides users with the option to visualize Bubble Sort, Selection Sort, Insertion Sort, Heap Sort, Merge Sort, and Quick Sort. The application uses a component-based architecture, and each sorting algorithm is implemented as a separate component that communicates with the parent component to update the state of the web application. The visual representation of the sorting algorithm is achieved by animating the sorting process using CSS transitions and transformations. This project is a valuable tool for college students who want to learn more about sorting algorithms and web development.

## Table of Contents

Table of Contents	1
List of Figures	2
1. CHAPTER	
1.1 Introduction	3
1.2 Literature Review	4
1.3 Project Overview	7
1.5 Sorting Algorithms	9
a. Bubble Sort	9
b. Selection Sort	9
c. Insertion Sort	10
d. Heap Sort	10
e. Merge Sort	11
f. Quick Sort	11
2. CHAPTER	
2.1 Technical Implementation	12
2.2 Results	13
Conclusion	17
Objectives	18
Future Scope	19
Bibliography	20

## List of Figures

Figure 1 - Project Flow chart	8
Figure 2 - Front Page	13
Figure 3 - Sorting Visualizers	13
Figure 4 - Bubble Sort	14
Figure 5 - Selection Sort	14
Figure 6 - Insertion Sort	15
Figure 7 - Heap Sort	15
Figure 8 - Merge Sort	16
Figure 9 - Quick Sort	16

# 1. CHAPTER

---

## 1.1 INTRODUCTION

The Sorting Algorithm Visualizer is a web app that provides a visual representation of different sorting algorithms in real-time. Built using HTML, CSS, JavaScript, and the React JS framework, the app allows users to visualize Bubble Sort, Selection Sort, Insertion Sort, Heap Sort, Merge Sort, and Quick Sort. Users generate a set of random numbers to sort and can modify them as needed. During the sorting process, the number of swaps and time taken are displayed. Animations using CSS transitions and transformations represent each algorithm. The Sorting Algorithm Visualizer uses a component-based architecture, with each sorting algorithm implemented as a separate component that updates the state of the web app. This project is a valuable tool for college students to learn about sorting algorithms and modern web development.



## 1.2 LITERATURE REVIEW

One of the most common problems in computer science education is the way to explain and communicate non-material terms and conditions. As a result the complexity increases when the concept is not visible, as it is fundamental in computer science, such as algorithms. Various methods of support for teaching algorithms have been proposed, including the use of graphical and oral presentations of algorithms. The beautiful AVs make the algorithms come to life by clearly representing their various regions and highlighting the transformation between those regions. They display data structures in natural, invisible ways instead of focusing on memory addresses and activity calls. AVs are naturally appealing to teachers, almost worldwide, who look good and are consistently “liked” by students. They can be used to attract students' attention during lectures, explain ideas in concrete terms, promote a practical learning process, and facilitate better communication between students and teachers. Interactive algorithm viewing allows students to evaluate and evaluate ideas regarding their individual needs. The purpose of this paper is to review the Visualization Algorithm textbooks, highlighting the work done so far in this area and the scope of development. We present the findings of the field of Visualization Algorithm (AV) based on our analysis. We regard the recognition of the algorithm as a priority and a point of view for further research and the application of its results in computer science education today.

Brown University Algorithm Simulator and Animator (BALSA) BALSA [Brown et al., 1985] was one of the first algorithm simulations designed to help students understand computer algorithms. The program has served as an example of the animation of many algorithms that were later developed [Wiggins, 1998]. It was developed in the early 1980s at Brown University to achieve a number of goals. Students could use it to look at the performance of algorithms and thus gain a better understanding of their performance. Students do not have to write code but ask for someone else's code. The teachers would use it to prepare the material for the students. The algorithm designers will use the resources provided by the system to obtain a dynamic image display of their operating systems to fully understand the algorithms. Cartoons using state-of-the-art resources provided by BALSA will design and implement programs that will be featured in production. The program was written in C and the PASCAL programs were animated. BALSA has provided users with a few services. Users can control the display features of the system and thus be able to create, delete, resize, move, zoom in, and filter algorithm views. Users are given different views of the algorithm at the same time. The system allowed several algorithms to be used and displayed simultaneously. Users are also able to interact with the animation of the algorithm. For example they can start, stop, slow down, and run the algorithm backwards. After the algorithm worked once, the entire history of the algorithm was saved so that readers could refer to it and restart it. Users can save their window settings and use it to restore algorithm views later. The original version of the system introduced black and white animation.

Tango and XTango The animation system of the Xtango algorithm was developed under the direction of Drs. John Stasko at Georgia Tech as a follower of the animation system of the Tango algorithm. XTango "supports color enhancement, real-time, 2 & 1/2 dimensional, smooth graphics of algorithms and programs" [Wiggins, 1998]. The system uses the

transformation paradigm to achieve smooth animation [Stasko, 1992]. XTango is used in the X11 window system. It is distributed through sample animation programs (e.g., matrix replication, Fast Fourier Transform, Bubble Filtering, Two Stacks, AVL Trees) [Wilson et al., 1996]. The system is designed for easy use. It is intended to be useful for those who are not computer-assisted in the use of motion pictures [Wiggins, 1998]. The package can be used in two ways: Users can embed data structures and library animation packages in C-format or any other programming language that can generate a tracking file. The embedded dial system is then integrated into the Xtango and X11 window libraries. Another way to use the animation package is to write a command text file read by a downloadable animated system translator along with the Xtango package. A text file can be created with a text editor or as a result of printed statements in a user simulation program.

## 1.3 PROJECT OVERVIEW

The Sorting Algorithm Visualizer is a web application that provides users with the ability to visualize different sorting algorithms. The web application is built using HTML, CSS, and JavaScript programming languages, and the React JS framework. The project is divided into two parts: the landing page and the sorting visualizer.

The landing page contains a simple webpage with a 'Generate' button. When the user clicks on the 'Generate' button, it redirects them to the main sorting visualizer page. The sorting visualizer provides users with the option to visualize six sorting algorithms, including Bubble Sort, Selection Sort, Insertion Sort, Heap Sort, Merge Sort, and Quick Sort. Additionally, users can choose to visualize all these sorting algorithms simultaneously.

The Sorting Algorithm Visualizer works as follows: After selecting the sorting algorithm to visualize, the user clicks the 'Generate' button, which generates a set of random numbers to be sorted. The user can then modify the numbers in the array bar. Once the user has configured the input data, they can click the 'Start' button, and the sorting algorithm will start sorting the data. During the sorting process, the number of swaps and time taken are displayed to the user.

After the sorting process is complete, the user can click the 'Generate' button to generate a new array or click the 'Redo' button to sort the current array again.

## 1.4 PROJECT FLOW CHART

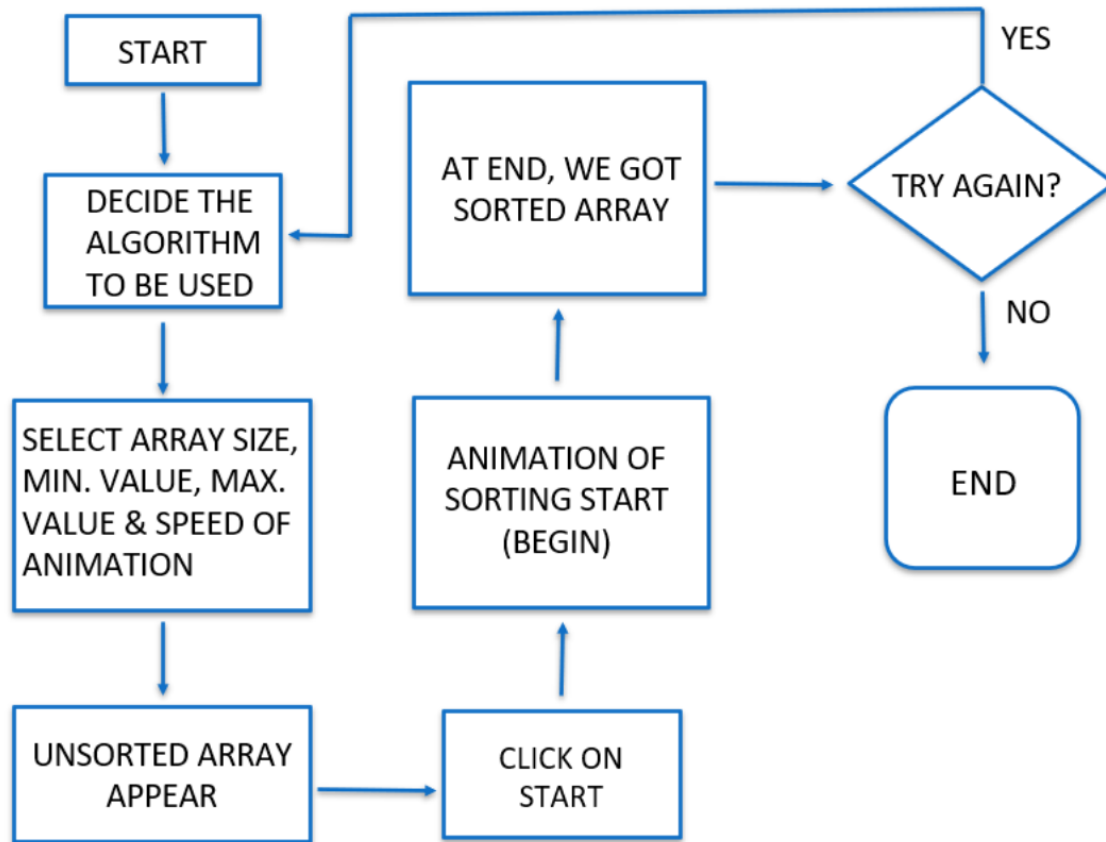


Figure 1 - Project Flow chart

## 1.5 SORTING ALGORITHMS

Here are explanations for each of the sorting algorithms mentioned in the Sorting Algorithm Visualizer project:

### a. Bubble Sort

Bubble Sort is a simple sorting algorithm that works by repeatedly swapping adjacent elements that are in the wrong order. The algorithm iterates through the array multiple times until it is sorted. In each iteration, the algorithm compares adjacent elements in the array and swaps them if they are in the wrong order. During the first iteration, the largest element bubbles up to the end of the array. During the second iteration, the second-largest element bubbles up to the second-last position, and so on until the array is completely sorted.

**Time Complexity:-** Although Bubble Sort is easy to implement, it is not very efficient and has a time complexity of  $O(n^2)$  in the worst-case scenario, where  $n$  is the number of elements to be sorted.

### b. Selection Sort

Selection Sort is a simple sorting algorithm that works by repeatedly finding the minimum element from the unsorted part of the array and placing it at the beginning. The algorithm divides the array into two parts: sorted and unsorted. In each iteration, it searches the unsorted part for the minimum element and swaps it with the first element in the unsorted part. This process continues until the entire array is sorted.

**Time Complexity:-** Selection Sort has a time complexity of  $O(n^2)$  in the worst-case scenario, where  $n$  is the number of elements to be sorted.

### **c. Insertion Sort**

Insertion Sort is a simple sorting algorithm that works by dividing the array into two parts: sorted and unsorted. The algorithm iterates through the unsorted part of the array, comparing each element with the elements in the sorted part and inserting it in its correct position in the sorted part. The sorted part starts as a single element, the first element of the array, and grows with each iteration.

**Time Complexity:-** Insertion Sort has a time complexity of  $O(n^2)$  in the worst-case scenario, where  $n$  is the number of elements to be sorted.

### **d. Heap Sort**

Heap Sort is a sorting algorithm that works by using the binary heap data structure to sort the array. A binary heap is a complete binary tree where each node satisfies the heap property, which means that the parent node is either greater than or equal to (in a max heap) or less than or equal to (in a min heap) its child nodes. The algorithm begins by building a max heap from the array to be sorted. It then repeatedly extracts the maximum element from the heap and places it at the end of the array, reducing the heap size by one. This process continues until the entire array is sorted.

**Time Complexity:-** Heap Sort has a time complexity of  $O(n \log n)$  in the worst-case scenario, where  $n$  is the number of elements to be sorted.

#### **e. Merge Sort**

Merge Sort is a divide-and-conquer sorting algorithm that works by recursively dividing the array into halves until each subarray contains a single element, then merging the subarrays in a sorted manner to produce the final sorted array. The merging process involves comparing the first elements of each subarray and placing the smaller one into the final array. The process continues until all elements are sorted and merged.

**Time Complexity:-** Merge Sort has a time complexity of  $O(n \log n)$  in the worst-case scenario, where  $n$  is the number of elements to be sorted.

#### **f. Quick Sort**

Quick Sort is another divide-and-conquer sorting algorithm that works by partitioning the array into two subarrays based on a pivot element. The pivot element is chosen, and the array is partitioned so that all elements smaller than the pivot are placed in the left subarray, and all elements greater than or equal to the pivot are placed in the right subarray. This process is then recursively applied to each subarray until the entire array is sorted. The choice of pivot element can greatly affect the performance of the algorithm. A good choice for the pivot element is the median, which can result in a balanced partitioning of the array.

**Time Complexity:-** Quicksort has a time complexity of  $O(n \log n)$  in the average case, where  $n$  is the number of elements to be sorted. However, in the worst-case scenario, Quick Sort can have a time complexity of  $O(n^2)$  if the pivot is chosen poorly.



## 2. CHAPTER

---

### 2.1 TECHNICAL IMPLEMENTATION

The Sorting Algorithm Visualizer is a web application developed using React JS, a popular front-end JavaScript framework. React JS provides a powerful and flexible library for building user interfaces. It allows the web application to handle complex states and user interactions in a streamlined manner.

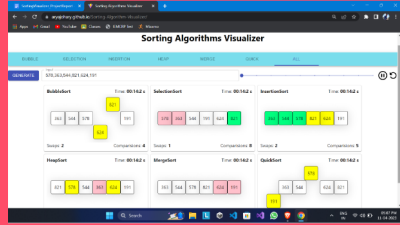
The Sorting Algorithm Visualizer uses a component-based architecture. Each sorting algorithm is implemented as a separate component that communicates with the parent component to update the state of the web application. The array bar is also implemented as a separate component that communicates with the sorting algorithm components to display the sorted data in real-time.

The visual representation of the sorting algorithm is achieved by animating the sorting process using CSS transitions and transformations. Each sorting algorithm is assigned a specific color, making it easier for the user to differentiate between them.

## 2.2 RESULTS

# Get the visual representation of Algorithms.

Generate



### Easy to use.

So easy to use, and works efficiently.

### Comprehensive

Multiple Sorting Algorithms supported.

### Guaranteed to work.

Analyse all the algorithms easily with our project.

Figure 2 - Front Page

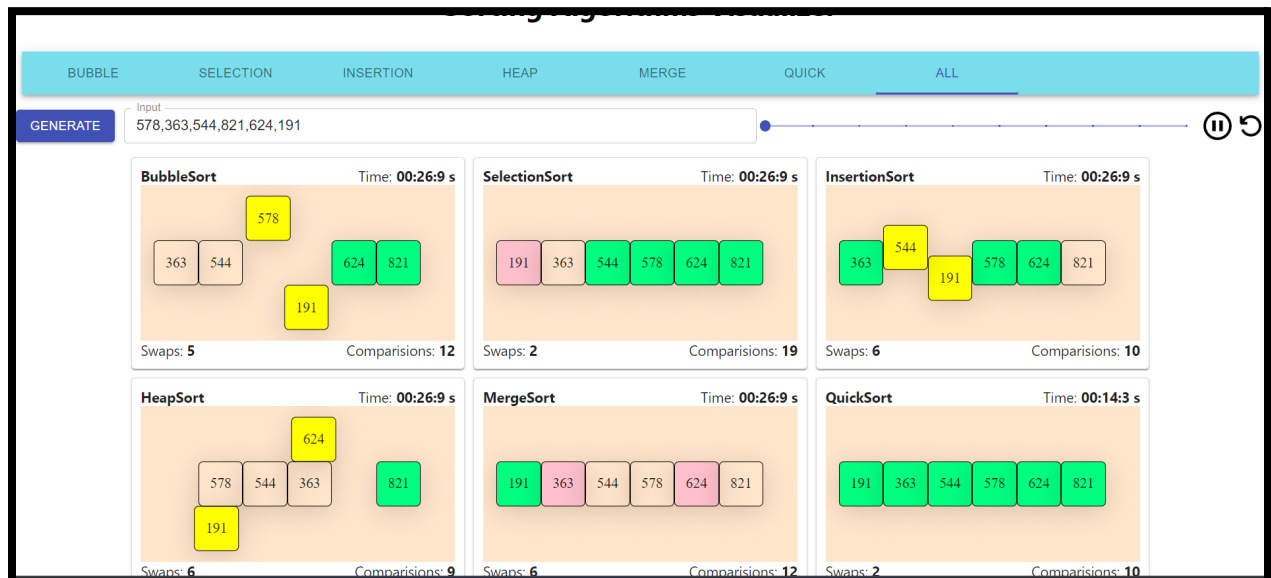


Figure 3 - Sorting Visualizers

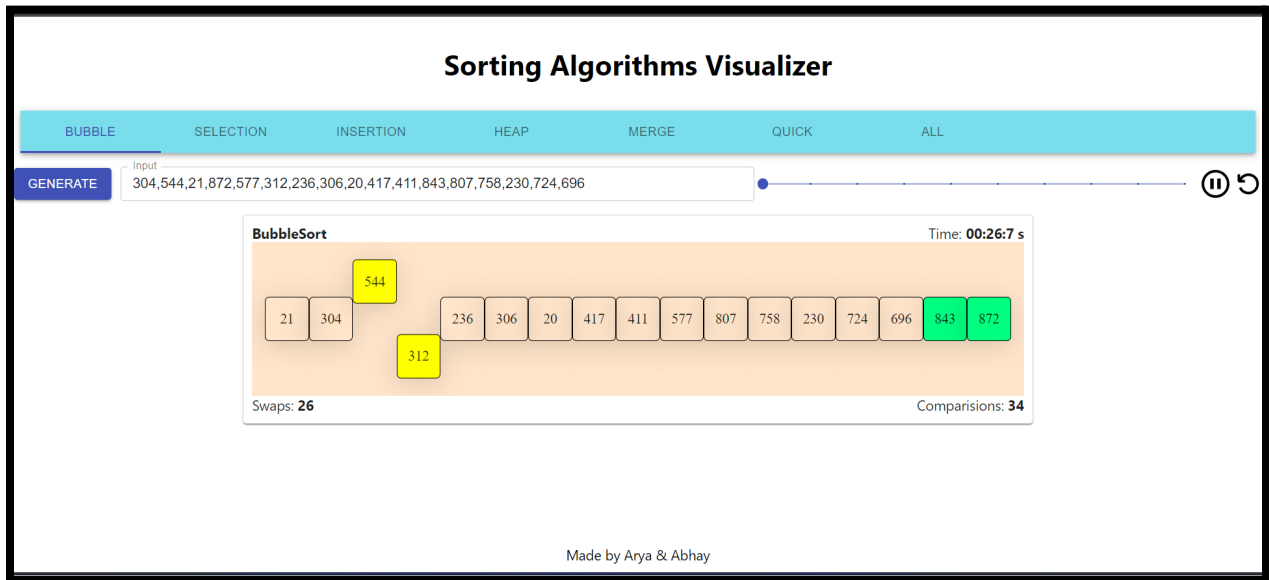


Figure 4 - Bubble Sort

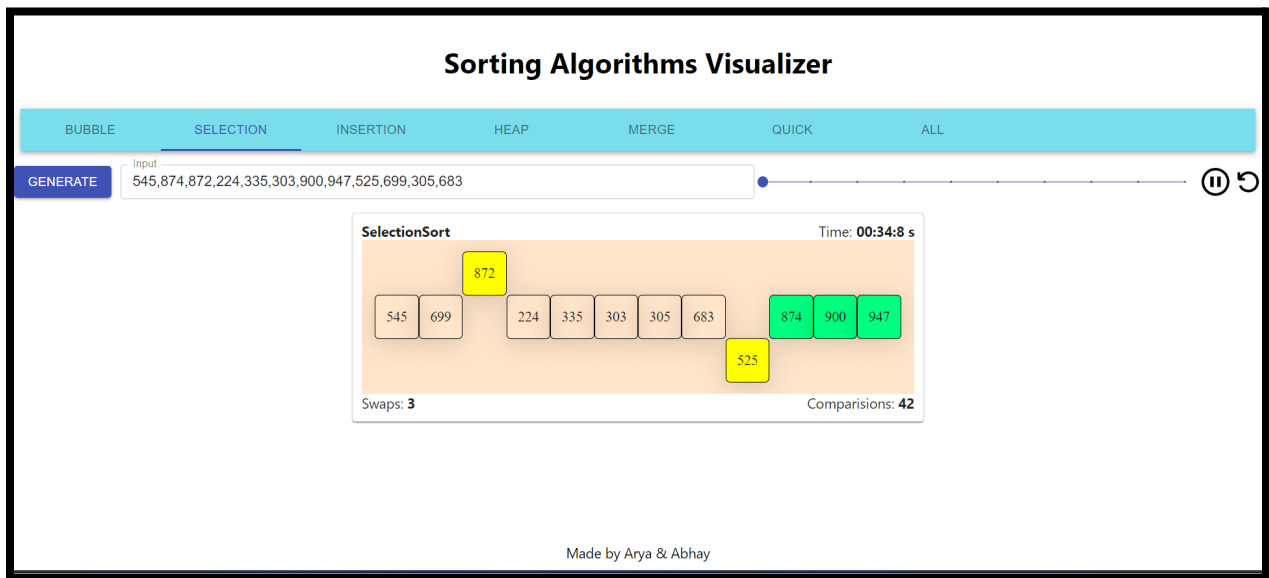


Figure 5 - Selection Sort

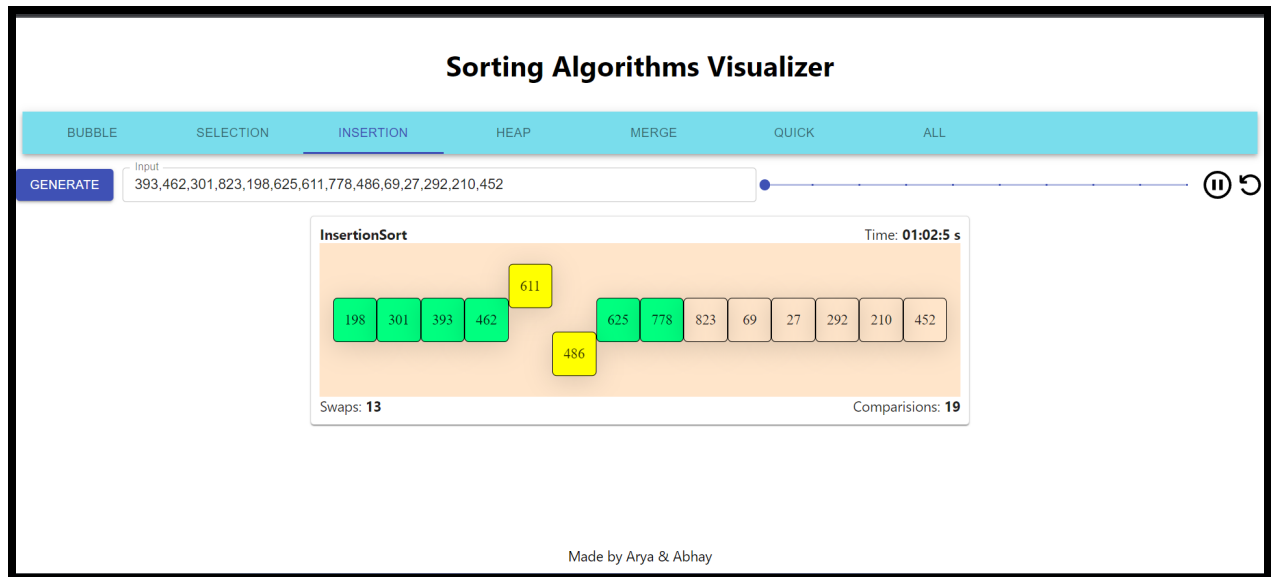


Figure 6 - Insertion Sort

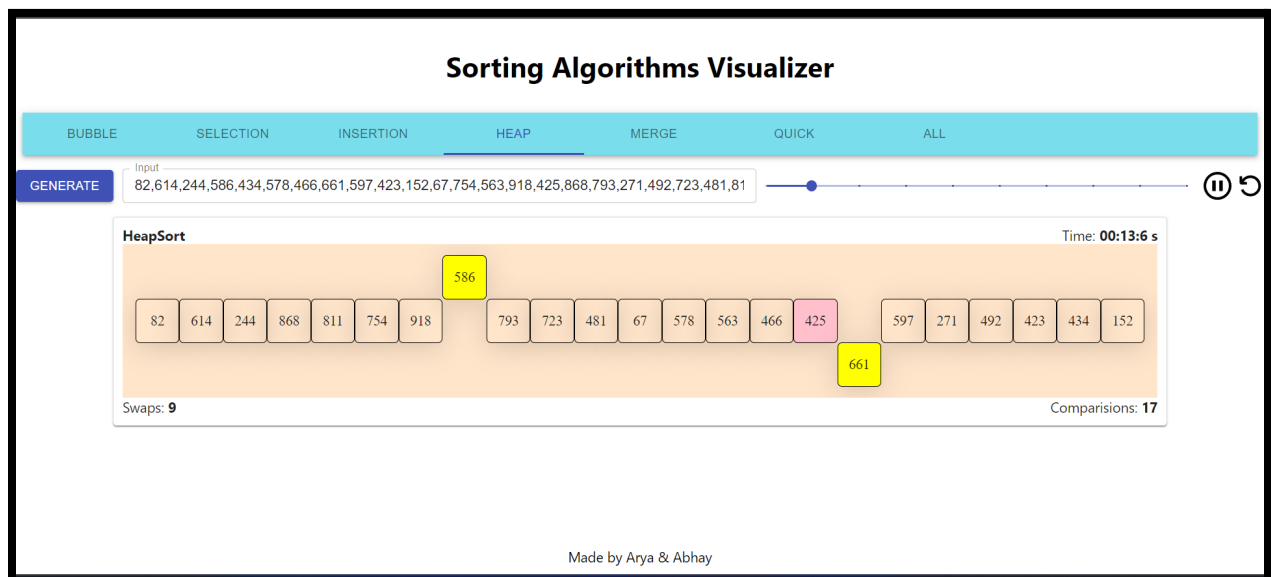


Figure 7 - Heap Sort

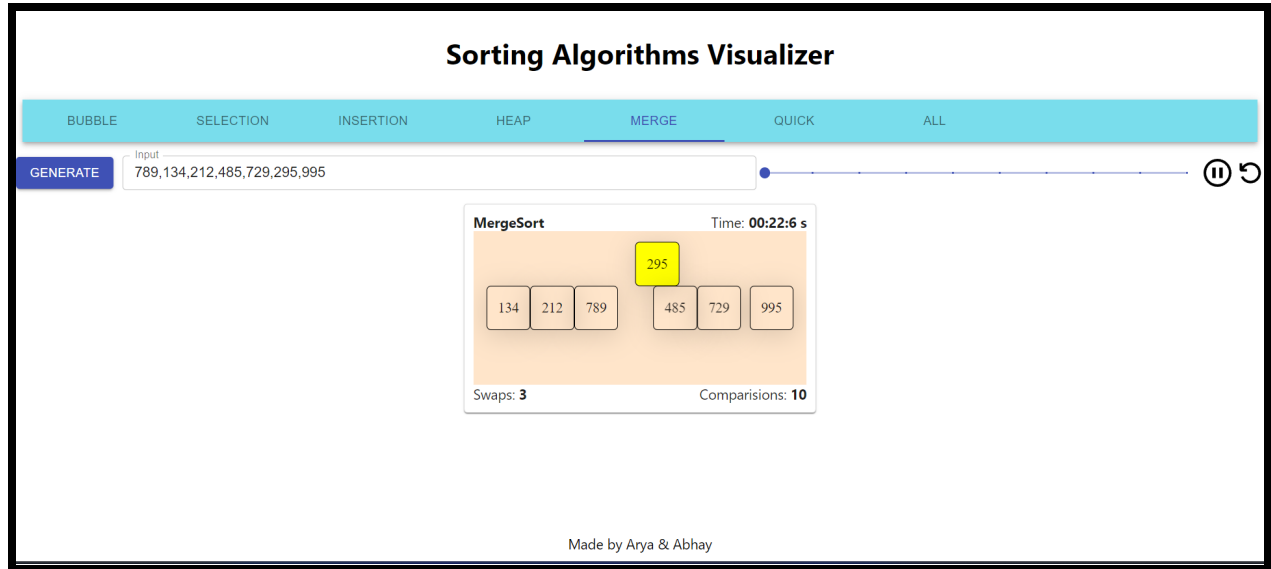


Figure 8 - Merge Sort

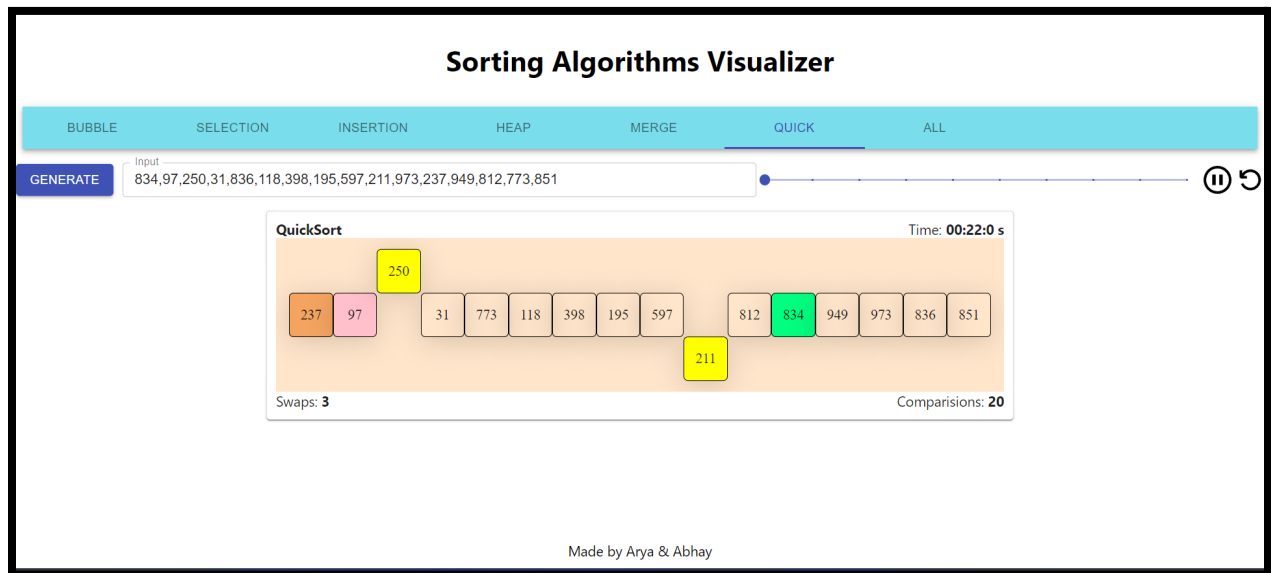


Figure 9 - Quick sort

## **CONCLUSION**

The Sorting Algorithm Visualizer is a powerful tool that provides users with a visual representation of how sorting algorithms work. It allows users to learn about sorting algorithms in a more engaging and interactive way. Additionally, the web application is built using modern web development technologies, making it scalable and efficient. The Sorting Algorithm Visualizer is an excellent project for college students who want to learn more about sorting algorithms and web development.

## **OBJECTIVES**

1. Develop a web application to visualize popular sorting algorithms.
2. Provide users with the ability to visualize sorting in real-time, with animations that demonstrate how the algorithms work.
3. Provide users with real-time statistics, such as the number of swaps and time taken to sort.
4. Improving problem-solving skills through algorithmic thinking.

## **FUTURE SCOPE**

1. Adding more sorting algorithms to the visualizer, such as Radix Sort and Counting Sort.
2. Expanding the visualizer to include other types of algorithms, such as searching algorithms or graph algorithms.
3. Integrating machine learning to automatically recommend the best sorting algorithm based on the dataset's characteristics.
4. Developing a teaching module or curriculum that incorporates the visualizer to enhance students' understanding of sorting algorithms.



## **BIBLIOGRAPHY**

Full Stack Web Development Course by Angela Yu - [Udemy.com](https://www.udemy.com/)

Sorting Visualizer Tutorial by Clément Mihailescu - [Youtube.com](https://www.youtube.com/)

React JS Tutorial - [w3schools.com](https://www.w3schools.com/)