Project Report On

Algorithm Visualization

“A dissertation submitted in partial fulfilment of the requirements of Bachelor of Technology Degree in Computer Science and Engineering of the Maulana Abul Kalam Azad University of Technology for the year 2019-2023”

Submitted by

Samir Chandra Pramanik (26300120051)

Biswajit Das (26300120035)

Suchismita Pal (26300120034)

Souvik Das (26300120046)

Kazi Moksud Hossain (26300119055)

Under the guidance of

Mr. Atanu Kumar Das

Designation

Dept of Computer Science & Engineering

Regent Education and Research Foundation

Regent Education and Research Foundation

(Affiliated to Maulana Abul Kalam Azad University of Technology, West Bengal)

Barrackpore - 700121, Barrackpore, WB

[IN INSTITUTE LETTER HEAD]

Certificate of Approval

This is to certify that this report of B. Tech. Final Year project, entitled **"Algorithm Visualization"** is a record of bona-fide work, carried out by **Samir Chandra Pramanik, Biswajit Das, Suchismita Pal, Souvik Das, Kazi Moksud Hossain** under my supervision and guidance.

In my opinion, the report in its present form is in partial fulfilment of all the requirements, as specified by the ***Regent Education and Research Foundation*** and as per regulations of the ***Maulana Abul Kalam Azad University of Technology***. In fact, it has attained the standard, necessary for submission. To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report for B. Tech. programme in Computer Science and Engineering in the year 2019-2023.

Guide / Supervisor

____

Mr. Atanu Kumar Das

Department of Computer Science and Engineering

Regent Education and Research Foundation

____

Examiner(s)

____

Head of the Department

Computer Science and Engineering

Regent Education and Research Foundation

ACKNOWLEDGEMENT

We would like to express our profound gratitude to our HOD **Mr. Subhankar Ghosh**, of Computer Science & Technology department, for your contributions to the completion of our project titled Algorithm Visualizer.

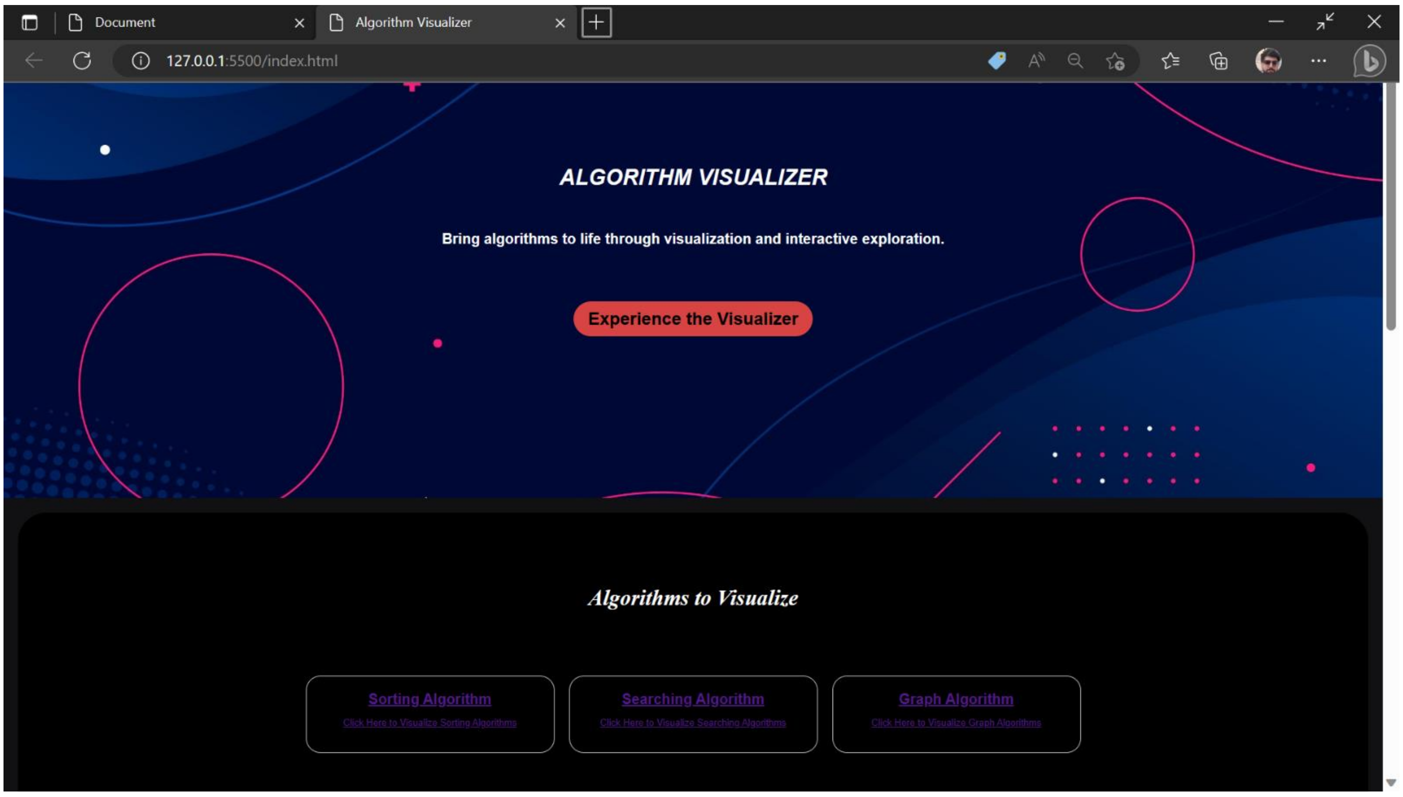
We would like to express our special thanks to our mentor Mr. Atanu Kumar Das, for his time and efforts he provided throughout the year. Your useful advice and suggestions were really helpful to us during the project's completion. In this aspect, we are eternally grateful to you.

Mr./Ms Name ____________

University Roll No and Registration No

Algorithm Visualizer

The goal of the Algorithm Visualizer project is to provide a web application that is interactive and that enables users to visualise and comprehend a variety of algorithms that are frequently used in computer science and programming. For users to investigate the inner workings of algorithms, the project offers a platform that is visually appealing and user-friendly, which encourages improved understanding and learning.



ALGORITHM VISUALIZER

Bring algorithms to life through visualization and interactive exploration.

Algorithms to Visualize

Sorting Algorithm
Click Here to Visualize Sorting Algorithms

Searching Algorithm
Click Here to Visualize Searching Algorithms

Graph Algorithm
Click Here to Visualize Graph Algorithms

Key Features:

1. Algorithm Selection: The application allows users to choose from a wide range of algorithms, including sorting, searching, graph traversal, and more.
2. Interactive Visualization: The algorithms are visualized using animated graphics and step-by-step demonstrations, enabling users to observe the algorithm's execution in real-time.
3. Customization Options: Users can modify input parameters such as array size, data distribution, and initial configurations to observe how the algorithms behave under different scenarios.
4. User-Friendly Interface: The web application boasts a clean and intuitive user interface, ensuring ease of navigation and interaction.

The Algorithm Visualizer project seeks to provide students, educators, and hobbyists with an invaluable learning resource for algorithms and their applications. The project encourages algorithmic comprehension and expertise by offering a fun and interactive platform, which supports the growth of effective problem-solving abilities in the field of computer science.

Here is the project link: Algorithm Visualizer (psamir275.github.io)

CONTENTS

CHAPTER 1 INTRODUCTION

- Objective2
- Scope of the System3
- Feasibility Study
    - Technical Feasibility
    - Operational Feasibility
    - Economic Feasibility

CHAPTER 2 SOFTWARE REQUIREMENT SPECIFICATION (SRS)

CHAPTER 3 SOFTWARE DEVELOPMENT PROCESS MODEL ADOPTED

CHAPTER 4 OVERVIEW

- System Overview
    - Limitation of Existing System
- Proposed System
    - Objectives of the proposed system
    - Users of the Proposed system
    -

CHAPTER 5 ASSUMPTION AND DEPENDENCIES

CHAPTER 6 TECHNOLOGIES

- Tools used in Development
- Development Environment
- Software Interface
- Hardware Used

CHAPTER 7 DESIGN

- Class Diagram
- Data Flow Diagram
- Entity Relationship Diagram

CHAPTER 8 DATA DICTIONARY

CHAPTER 9 TESTING

- **Unit Testing**
- **Integrity Testing**

CHAPTER 10 SNAPSHOTS

CHAPTER 12 CONCLUSION and FUTURE SCOPE

CHAPTER 11 REFERENCES

INTRODUCTION

When we talk about complex subject topics like Algorithms, it becomes extremely necessary for students to have a strong grip over the topic as it would form the foundation of their computational thinking and programming skills. We had observed that through conventional methods of teaching it becomes a little difficult for students to understand the concept and for teachers to explain their thoughts. Motivated by the age-old saying, "a picture speaks more than thousand words", many researchers and educators assume that students would learn an algorithm faster and more thoroughly using algorithm visualization techniques1. So, we developed a method of learning through visualization and hand-on experience over different searching and sorting algorithms which is bound to help the students and teachers.

Definition of Algorithm?

A set of detailed instructions or a step-by-step process for finishing a task or addressing a problem is known as an algorithm. A computer programme follows a clear, unambiguous set of instructions to carry out a particular task or computation. An algorithm's objective is to carry out a task or solve a problem in an effective, accurate, and efficient manner. In many disciplines, such as computer science, mathematics, engineering, and the natural sciences, algorithms are used to carry out operations including data analysis, sorting, searching, optimisation, and decision-making. They are essential to how computers work and how software programmes are created.

Types of algorithms

There are many types of algorithms, some of which are:

1. Sorting algorithms: These algorithms are used to sort a list of elements into a specific order, such as alphabetical or numerical order.

2. Searching algorithms: These algorithms are used to search for a specific item or value within a data set, such as finding a word in a document or a number in an array.

3. Graph algorithms: These algorithms are used to model relationships between data points, such as determining the shortest path between two points or finding the minimum spanning tree of a graph.

4. Computational algorithms: These algorithms are used to perform complex mathematical calculations, such as solving equations or simulating physical systems.

5. Divide and conquer algorithms: These algorithms break a problem into smaller sub-problems, solve each sub-problem independently, and then combine the results to solve the original problem.

6. Greedy algorithms: These algorithms make locally optimal choices at each step in the hope of finding a global optimum solution.

7. Dynamic programming algorithms: These algorithms solve problems by breaking them down into smaller sub-problems, solving each sub-problem only once, and then storing the results to avoid redundant computations.

8. Randomized algorithms: These algorithms use a random element in the algorithm to achieve a specific outcome, such as generating random numbers or sorting data.

Visualization

To facilitate comprehension, analysis, and communication, data or information is represented graphically or visually through visualisation. Complex data sets can be made more understandable, useful, and accessible by using visualisation techniques.

There are many ways to visualise data, including charts, graphs, diagrams, maps, and infographics. It entails choosing data and presenting it in a way that draws attention to patterns, trends, connections, and anomalies. It is frequently simpler to spot patterns and trends, comprehend correlations between variables, and explain difficult information to others by visualising data.

Algorithm Visualisation

The use of visual representations to explain how algorithms behave and operate is known as algorithm visualisation. It entails producing animations, interactive simulations, or other visualisations that show how algorithms function piecemeal.

Making it simpler for people to learn about algorithms and comprehend the principles behind them is the aim of algorithm visualisation. Users can better understand how the algorithm operates and why certain results are produced by using visual aids that allow them to see how data is handled and processed at each level of the process.

Algorithms visualizer

Users can interactively visualise the behaviour and execution of algorithms using the software tool known as an algorithm visualizer. It is a kind of algorithm visualisation that offers a simple user interface for discovering and comprehending algorithms.

Users can often go through the execution of an algorithm using algorithm visualizers, visualising each step as it happens and viewing how data is processed and handled at each stage. Other interactive capabilities offered by certain algorithm visualizers include the ability to alter input parameters, modify the algorithm's settings, and observe how these modifications impact the algorithm's behaviour.

Objective:

Algorithm visualization's goal is to graphically depict an algorithm's behaviour and inner workings in order to aid consumers in understanding it. Various visual representations, including graphs, charts, animations, and interactive simulations, can be used to accomplish this.

There are several situations in which algorithm visualisation is beneficial. It can make complex algorithms easier for students and other learners to understand in a fun and straightforward way. By giving a clearer picture of how algorithms function and where possible problems can occur, it can also aid academics and developers in their efforts to debug and optimise algorithms. Algorithm visualisation is a useful tool for anyone who wants to better understand algorithms and the uses for them.

Scope of the System

The purpose of an algorithm visualisation system is typically to create an interactive and visual depiction of an algorithm's execution. The system should be able to display the algorithm's stages as it processes data and allow users to interact with and edit the data as well as the algorithm itself.
The system should also be able to display the algorithm's runtime complexity, allowing users to see how the algorithm's performance varies as the input data size grows. This can assist users in understanding the algorithm's scalability and making educated decisions about when to utilise it in practise.
Furthermore, the system should be user-friendly and accessible, with intuitive visualisations and clear descriptions of the processes.

Feasibility Study

i. Technical Feasibility

Algorithm visualisation is the act of using graphical or visual representations to better understand the stages involved in addressing a certain problem using an algorithm. The feasibility of constructing an algorithm visualisation is determined by a number of criteria, including the complexity of the algorithm, the availability of appropriate visualisation tools, and the skill of the person doing the visualisation.

In general, most algorithms can be visualised theoretically, but the amount of difficulty varies. Simple algorithms, such as sorting algorithms like bubble sort, selection sort, and insertion sort, can be easily visualised using simple diagrams or animations. More complex algorithms, such as machine learning algorithms like neural networks, may necessitate more advanced visualisation tools and methodologies.

ii. Operational Feasibility

The extent to which an algorithm visualisation tool can be effectively incorporated into an organization's or system's existing operational processes is referred to as operational feasibility. In the context of algorithm visualisation, operational feasibility would entail determining whether the tool can be seamlessly integrated into the existing software development process and whether it is compatible with the existing tools and technology.

User acceptability is a crucial part of operational feasibility. The tool should be user-friendly and simple to use, allowing developers to quickly learn how to use it and integrate it into their workflow. The visualisation tool should also be versatile and flexible, allowing developers to use it in the way that best suits them.

iii. Economic Feasibility

The use of visual representations to help users understand algorithms and their behaviours is referred to as algorithm visualisation (AV). Any software development effort, including algorithm visualisation, must consider economic feasibility. Here are some things to think about:

Development costs vary based on the complexity of the algorithms being visualised, the level of interaction required, and the programming language and tools employed. To assess if the instrument is economically feasible, the development expenses should be compared against the prospective benefits.

Adoption: The success of an antivirus tool is ultimately determined by whether or not people adopt and continue to use it. As a result, it is critical to examine the user experience, convenience of use, and whether the tool is suitable.

SOFTWARE REQUIREMENT SPECIFICATION (SRS)

Algorithm visualization is a useful tool for teaching computer science concepts and improving algorithm design. This document outlines the software requirements for a system that visualizes algorithms.

Purpose: The purpose of this software is to provide a visual representation of algorithms to aid in teaching and understanding computer science concepts.

Scope: The software will provide visualization for a variety of algorithms and allow users to step through each step of the algorithm, observe the state of the data structures, and trace the flow of control. The software will support a variety of programming languages, including C++, Java, and Python.

Functional Requirements:

3.1. User Interface The software will have a graphical user interface that displays the algorithm visualization and controls for stepping through the algorithm. The user interface will be intuitive and easy to use.

3.2. Algorithm Visualization The software will provide visualizations for a variety of algorithms, including sorting algorithms, search algorithms, and graph algorithms. The visualization will display the state of the data structures and the flow of control as the algorithm executes.

3.3. Language Support The software will support a variety of programming languages, including C++, Java, and Python. The user will be able to select the language for the algorithm to be visualized in.

3.4. Code Import The software will allow users to import code from a file or copy and paste code into the software. The software will automatically generate a visualization for the algorithm based on the imported code.

3.5. Code Editing The software will allow users to edit the code of the algorithm within the software. The software will update the visualization in real-time as the user edits the code.

3.6. Step Control The software will allow users to step through each step of the algorithm, observing the state of the data structures and the flow of control.

3.7. Data Structure Inspection The software will allow users to inspect the state of the data structures at any point in the algorithm execution. The user will be able to view the contents of variables and data structures.

Non-functional Requirements:

4.1. Performance The software will be designed for efficient performance, with minimal lag or delay in visualization updates.

4.2. Reliability The software will be reliable and error-free, with robust error handling and recovery mechanisms in place.

4.3. Compatibility The software will be compatible with a variety of operating systems, including Windows, macOS, and Linux.

4.4. Usability The software will be easy to use, with an intuitive user interface and clear documentation for users.

Constraints: The software will require modern web browsers with support for WebGL and JavaScript. The software will not support languages other than C++, Java, and Python.

Future Enhancements: Future enhancements to the software may include additional language support, improved performance, and additional visualization types.

Conclusion: This document has outlined the requirements for a software system that provides algorithm visualization for a variety of programming languages. The software will provide an intuitive user interface and support for inspecting data structures and stepping through the algorithm. The software will be reliable and efficient, with compatibility for multiple operating systems.

SOFTWARE DEVELOPMENT PROCESS MODEL ADOPTED

There are several software development process models that can be adopted for Algorithm Visualization. One popular model is the Agile methodology, which emphasizes continuous delivery and customer collaboration throughout the development process.

In the Agile methodology, development is broken down into small, iterative cycles called sprints. Each sprint typically lasts between 1-4 weeks and involves a set of tasks or user stories that the development team works on. At the end of each sprint, the team delivers a working increment of the software.

Another software development process model that can be adopted for Algorithm Visualization is the Waterfall model. In this model, development progresses in a linear fashion through a set of well-defined stages, starting with requirements gathering and ending with deployment. Each stage must be completed before moving on to the next, with little room for iteration or change once a stage is complete.

There are also hybrid models that combine aspects of Agile and Waterfall methodologies, such as the Spiral model. This model involves a series of iterative cycles, with each cycle involving a set of planning, risk analysis, and prototyping stages.

Ultimately, the software development process model adopted for Algorithm Visualization will depend on the specific needs of the project and the preferences of the development team. It's important to choose a model that supports collaboration, communication, and flexibility in order to ensure the success of the project.

OVERVIEW

- **System Overview**

In algorithm visualization, a system overview provides a high-level view of the different components and processes involved in the algorithm. The system overview typically includes a graphical representation of the algorithm and its key components, such as data structures, functions, and control flow.

The purpose of the system overview is to help users understand how the algorithm works at a high level, without getting bogged down in the details of individual steps. By providing a clear picture of the overall structure of the algorithm, the system overview can help users identify patterns and relationships that might not be immediately apparent from a step-by-step description.

To create a system overview in algorithm visualization, you might use a range of graphical tools, such as flowcharts, diagrams, or interactive visualizations. The system overview should provide a clear and intuitive representation of the algorithm, highlighting the key steps and decision points in the process. You might also include annotations or notes to provide additional context or explanation for each component or process.

Overall, a well-designed system overview can be an invaluable tool for understanding complex algorithms and can help users quickly identify areas where improvements or optimizations might be possible.

○ **Limitation of Existing System**

Algorithm visualization is a powerful tool for teaching and understanding computer algorithms. However, existing systems for algorithm visualization have several limitations:

Limited Interactivity: Most existing systems provide only limited interactivity, such as step-by-step execution or the ability to pause and resume execution. This can make it difficult for learners to explore alternative paths through the algorithm.

Limited Scope: Many algorithm visualization systems only visualize a single algorithm or a small set of algorithms. This can make it difficult for learners to compare and contrast different algorithms.

Limited Customization: Existing systems often do not allow users to customize the visualization to their needs or preferences. This can make it difficult for learners to adapt the visualization to their learning style.

Limited Scalability: Some existing systems can only visualize small instances of algorithms. This can make it difficult for learners to understand the algorithm's behaviour on larger input sizes.

Limited Accessibility: Many algorithm visualization systems are not accessible to learners with disabilities, such as visual or hearing impairments.

Limited Coverage: Many existing systems focus on classic algorithms, neglecting important modern algorithms and data structures.

Overall, while algorithm visualization is a powerful tool for learning and understanding algorithms, existing systems have several limitations that can make it challenging for learners to fully benefit from this technology.

Proposed System

- ### Objectives of the proposed system

Depending on the particular goals and requirements of the system, the objectives of a proposed system for algorithm visualisation may change. However, the following are some general goals that an algorithm visualisation system might pursue:

Enhancing Understanding: The main goal of an algorithm visualisation system is to make complex algorithms and data structures easier for users to understand. An algorithm's behaviour can be learned and understood more easily and interactively through visualisation, which can aid in users' memory retention.

Enhancing Learning: Students, researchers, and developers can all benefit from a visualisation system's improved accessibility and interactivity of algorithms and data structures. The system's ability to provide immediate feedback on an implementation's correctness makes it possible to

- ### Users of the Proposed system

Algorithm visualization systems are software tools that help users to visualize how algorithms work. Users of these systems can be anyone interested in learning about algorithms, from students studying computer science to experienced programmers looking to better understand complex algorithms. Here are some of the potential users of a proposed system for algorithm visualization:

Students: Students who are learning about algorithms in computer science courses can benefit greatly from algorithm visualization tools. These tools help students to better understand how algorithms work by providing a visual representation of the algorithm and its steps.

Educators: Educators who teach computer science can use algorithm visualization tools to enhance their lectures and to help students visualize the concepts being taught. These tools can also be used to create interactive learning activities for students.

Researchers: Researchers in the field of computer science can use algorithm visualization tools to study how people learn algorithms and to evaluate the effectiveness of different visualization techniques.

Programmers: Programmers who work with complex algorithms can use visualization tools to gain a better understanding of how the algorithms work. This can help them to debug code and to optimize algorithms for better performance.

Enthusiasts: Anyone interested in algorithms and computer science can use algorithm visualization tools to explore algorithms in a fun and interactive way. These tools can be used to learn about algorithms that are not typically covered in traditional computer science courses.
Overall, algorithm visualization tools have the potential to benefit a wide range of users, from students to researchers to programmers and enthusiasts. By providing a visual representation of complex algorithms, these tools can help users to better understand how algorithms work and to explore new algorithms in an interactive way.

ASSUMPTION AND DEPENDENCIES

Algorithm visualization is a technique that is used to represent algorithms in a visual form to help people understand the algorithm better. However, when creating an algorithm visualization, there are several assumptions and dependencies that need to be considered. These include:

The algorithm is correctly implemented: The algorithm visualization assumes that the algorithm being represented is correctly implemented and works as intended. If the implementation is incorrect, the visualization may be misleading.

The algorithm has a deterministic behaviour: The algorithm visualization assumes that the algorithm has a deterministic behaviour, i.e., for a given input, the algorithm will produce the same output every time it is executed. If the algorithm has a non-deterministic behaviour, the visualization may not accurately represent the algorithm.

The algorithm is designed for a specific problem: The algorithm visualization assumes that the algorithm is designed to solve a specific problem and that the problem is well-defined. If the problem is not well-defined, the visualization may not be useful.

The user has a basic understanding of programming concepts: The algorithm visualization assumes that the user has a basic understanding of programming concepts like loops, conditionals, variables, etc. If the user does not have this understanding, the visualization may be difficult to understand.

The visualization tool used is reliable: The algorithm visualization depends on the reliability of the visualization tool used. If the tool is unreliable, the visualization may be inaccurate or misleading.
Overall, it is important to consider these assumptions and dependencies when creating an algorithm visualization to ensure that it accurately represents the algorithm and is useful for the intended audience.

TECHNOLOGIES

- ## Tools used in Development.

  i. Code Editor: VS Code

  ii. Integrated Development Environments (IDEs): VS Code

  iii. Debugging and Testing Tools

  iv. Web Development Tools: HTML, CSS, Chrome Developer Tools

- ## Development Environment

An algorithm visualization development environment is a software tool that allows developers to create interactive visualizations of algorithms and data structures. These visualizations can be used to help learners understand complex algorithms and how they work. There are several development environments available for algorithm visualization, including:

a. Programming Languages: JavaScript

b. Integrated Development Environment (IDE): VS Code

c. Debugging and Testing Tools: Chrome Developer Tools, Live Server

d. Data Structures and Algorithms: Searching, Shorting, Graph

e. User Interface (UI) Design: HTML, CSS

- **Software Interface**

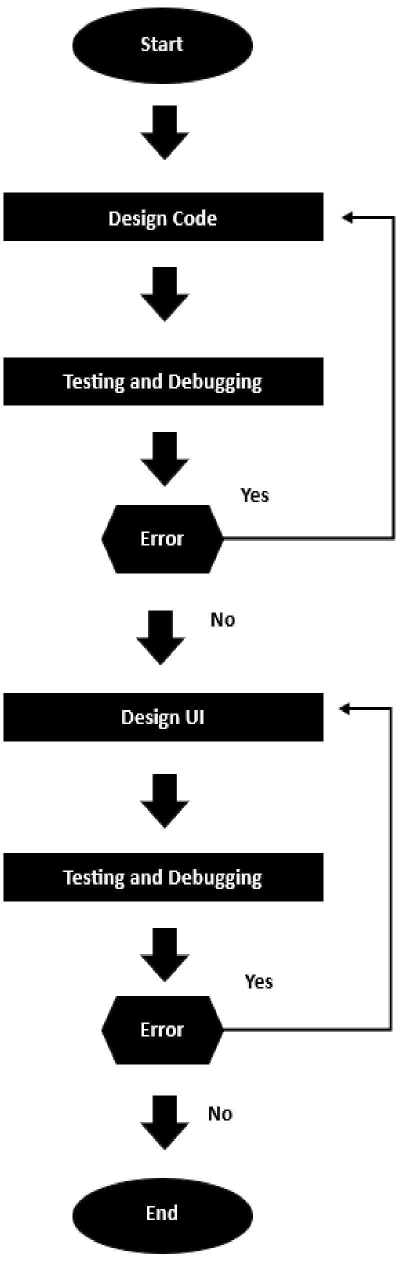
A software interface’s key features for algorithm visualisation include the following:

1. Operating System (OS): Windows 11, Ubuntu 23.04
2. Code Editor: VS Code
3. Testing and Debugging: Google Chrome, Chrome Developer Tools

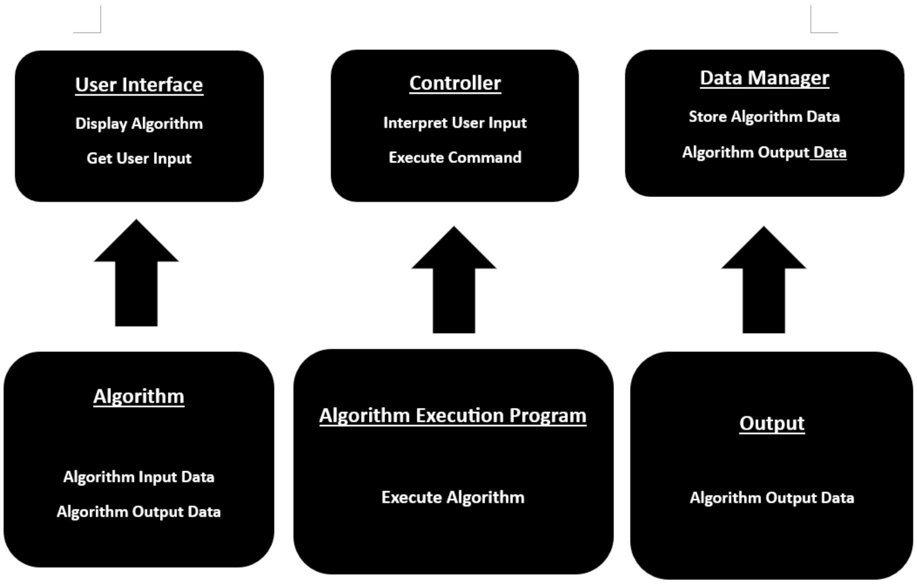
- **Hardware Used**

1. Intel Core I5 Processor
2. 8GB RAM
3. Internet Connectivity

Flowchart of the Algorithm Visualizer



Class Diagram of Algorithm Visualizer



The class diagram shows the different classes involved in the algorithm visualizer.

- User Interface class is responsible for displaying the algorithm to the user and getting user input.

- Algorithm class represents the algorithm being visualized, and it provides methods to get the input, and output.

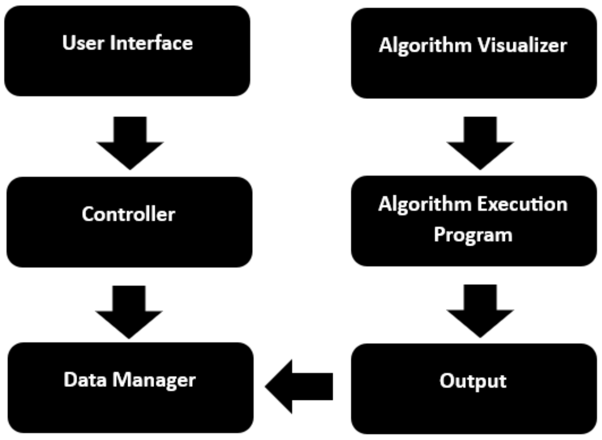
- Controller class is responsible for interpreting the user's input and sending commands to the Algorithm Execution Program class.

- Algorithm Execution Program class is responsible for executing the algorithm and generating the output.

- Data Manager class is responsible for managing the data flow between the Controller and Algorithm Execution Program classes, as well as storing the output generated by the Algorithm Execution Program class.

- Output class represents the output generated by the algorithm, and it provides a method to get the output data.

Data Flow Diagram of Algorithm Visualizer



The diagram shows three main components: User interface, Controller, Data manager.

- User interface: This component is responsible for displaying the algorithm to the user and receiving input from user.
- Controller: This component is responsible for interpreting the user's input and translating it into commands that can be executed by the algorithm execution program.
- Data manager: This component is responsible for managing the data flow between the controller and the algorithm execution engine, as well as storing the output generated by the algorithm.
- Algorithm execution program: This component is responsible for executing the algorithm and generating the output.

DATA DICTIONARY

A data dictionary is a document that provides a detailed description of the data used in an algorithm or program. In algorithm visualization, a data dictionary can be used to help users understand the inputs, outputs, and variables used in the algorithm.

The data dictionary typically includes the following information:
Data element name: The name of the data element being described.
Description: A brief description of the data element, including its purpose, source, and any other relevant information.

Data type: The data type of the element, such as integer, float, or string.

Size: The size of the data element, such as the number of bytes it occupies in memory.

Default value: The value that is assigned to the data element if no other value is specified.

Valid values: A list of all valid values that the data element can take.
Dependencies: A list of other data elements that the current element depends on.

Validation rules: Any rules or constraints that must be met for the data element to be considered valid.

Examples: Examples of typical values for the data element.
By providing this information, the data dictionary can help users understand the data used in the algorithm and how it is processed. It can also be useful for developers who need to maintain and update the algorithm over time.

TESTING

- **Unit Testing**

Unit testing is an important aspect of software development, including algorithm visualization. Algorithm visualization is the process of visually representing how an algorithm works and its step-by-step execution. In this context, unit testing involves testing each component or function of the visualization separately to ensure that it works as expected.

To perform unit testing in algorithm visualization, you can follow these steps:

Identify the components of your visualization that need testing. This could include individual functions, algorithms, and data structures.
Write test cases for each component. Test cases should be designed to cover all possible scenarios and edge cases.

Use a testing framework to automate the testing process. Some popular testing frameworks for JavaScript-based visualization tools include Jest, Mocha, and Jasmine.

Run the tests and analyse the results. If any test cases fail, debug the code and fix the issues.

Repeat the process for each component until all tests pass.
It's important to note that unit testing is just one part of the overall testing process. You should also perform integration testing to ensure that all components of your visualization work together seamlessly, and user acceptance testing to ensure that the visualization meets user requirements.

- **Integrity Testing**

Integrity testing in algorithm visualization is the process of verifying the correctness and accuracy of an algorithm visualization tool. The goal is to ensure that the tool is faithfully representing the algorithm it is visualizing, and that it is not introducing errors or biases that could mislead users.

There are several strategies that can be used to test the integrity of an algorithm visualization tool. One approach is to compare the output of the tool to the expected output of the algorithm for a given input. This can be done by manually executing the algorithm on a small input and comparing the results to the output generated by the visualization tool.

Another approach is to use test cases that cover a wide range of input sizes and edge cases. This can help identify any potential errors or limitations in the tool's implementation. It can also help to identify any performance issues that could cause the tool to run slowly or crash when dealing with large inputs.

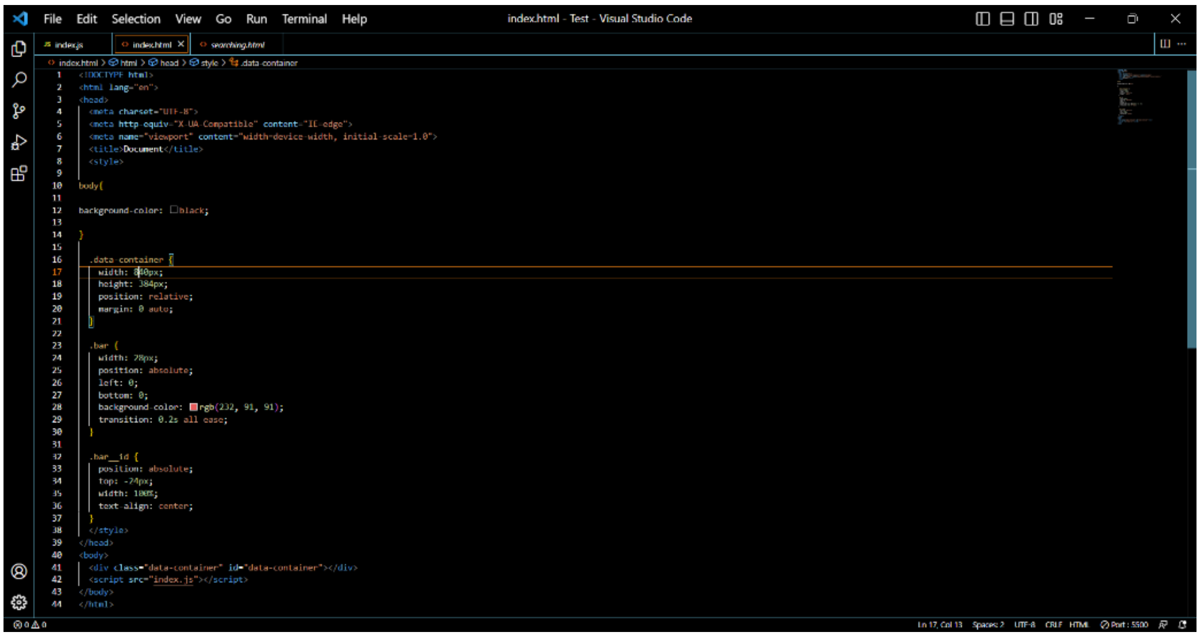
It's also important to consider the usability of the visualization tool. This includes factors such as ease of use, clarity of the visualizations, and the ability to customize and control the visualization parameters. Usability testing can be done by observing users as they interact with the tool and collecting feedback on their experience.

Overall, integrity testing is an important step in the development of any algorithm visualization tool. It helps ensure that the tool is accurate, reliable, and useful for its intended audience.

SNAPSHOTS

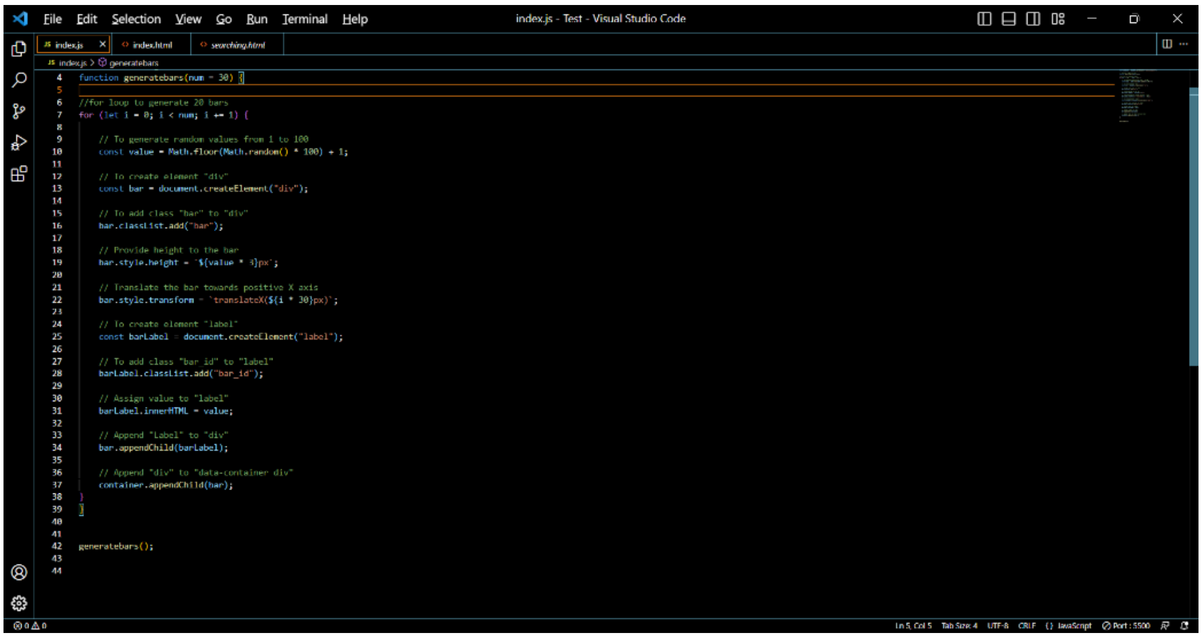
Implementing Bar Graphs for Visualization

Html and css code part


File Edit Selection View Go Run Terminal Help
index.html - Test - Visual Studio Code
index.js
index.html
searching.html
index.html > html > head > style > .data-container
Ln 17, Col 13 Spaces: 2 UTF-8 CRLF HTML Port : 5500

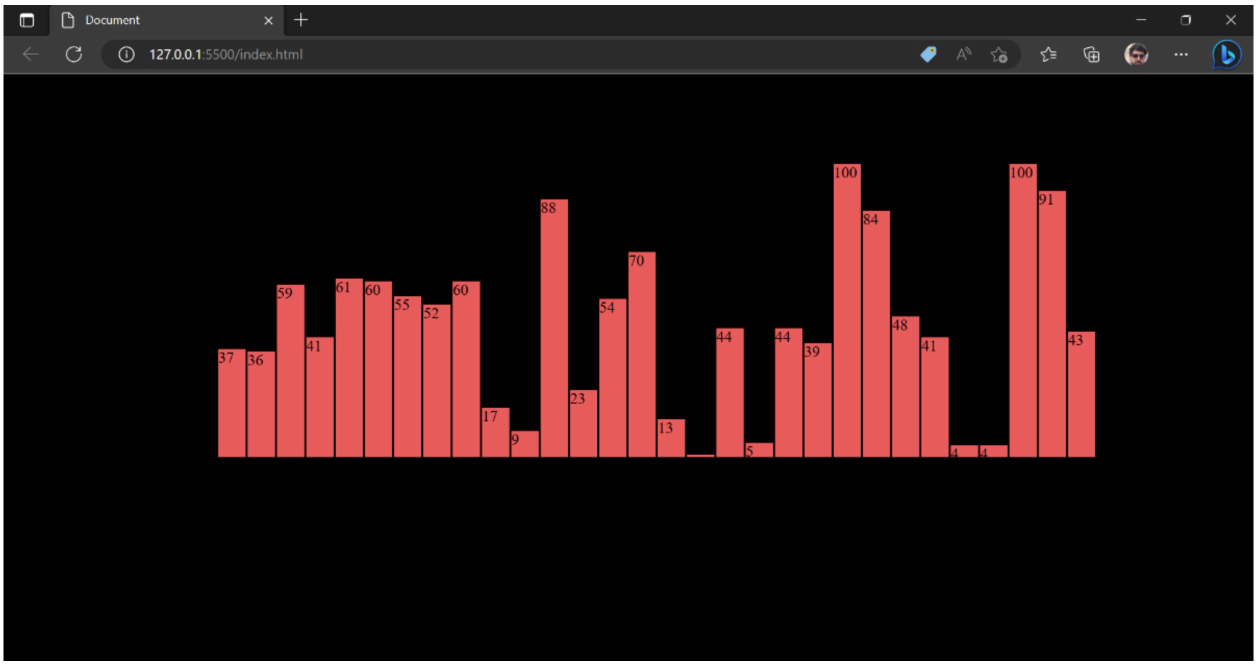
```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>

body{

background-color: black;

}

  .data-container {
    width: 840px;
    height: 384px;
    position: relative;
    margin: 0 auto;
  }

  .bar {
    width: 28px;
    position: absolute;
    left: 0;
    bottom: 0;
    background-color: rgb(232, 91, 91);
    transition: 0.2s all ease;
  }

  .bar__id {
    position: absolute;
    top: -24px;
    width: 100%;
    text-align: center;
  }
  </style>
</head>
<body>
  <div class="data-container" id="data-container"></div>
  <script src="index.js"></script>
</body>
</html>
```

JavaScript Function


File Edit Selection View Go Run Terminal Help
index.js - Test - Visual Studio Code
index.js
index.html
searching.html
index.js > generatebars
Ln 5, Col 5 Tab Size: 4 UTF-8 CRLF {} JavaScript Port : 5500

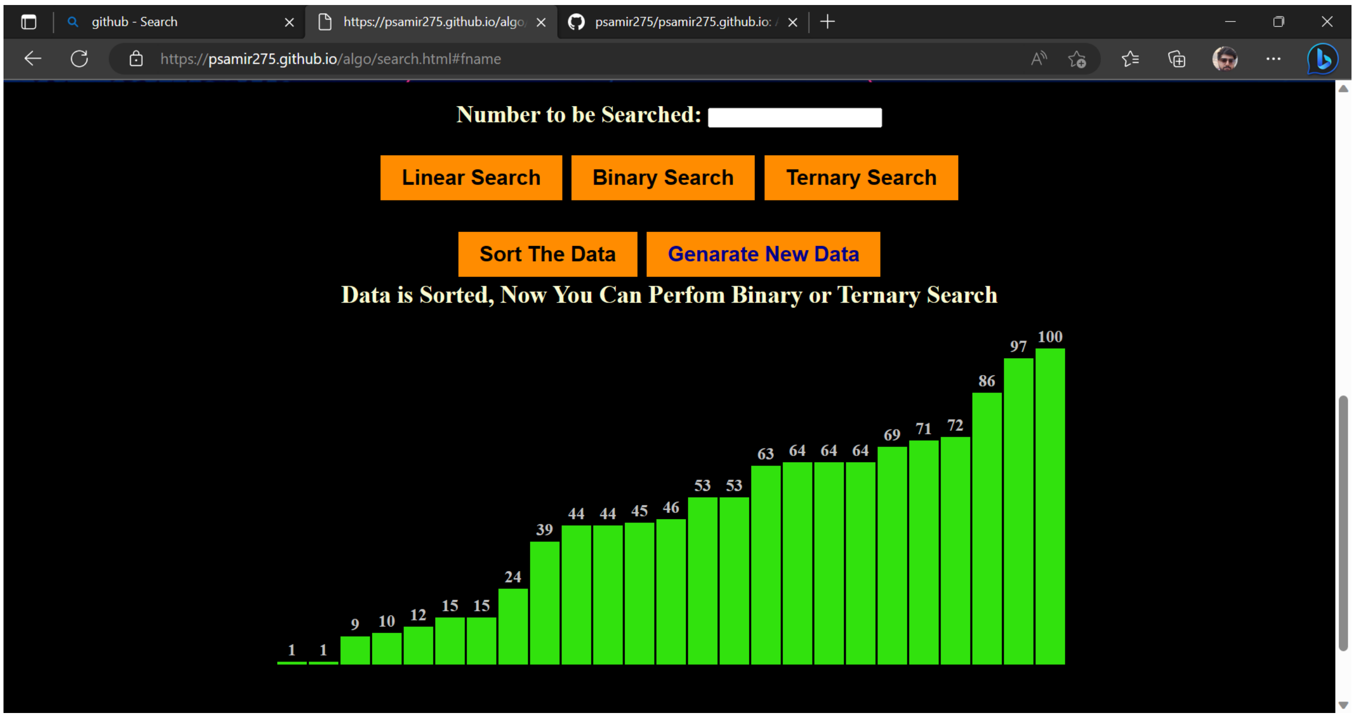
```javascript
function generatebars(num = 30) {

//for loop to generate 20 bars
for (let i = 0; i < num; i += 1) {

    // To generate random values from 1 to 100
    const value = Math.floor(Math.random() * 100) + 1;

    // To create element "div"
    const bar = document.createElement("div");

    // To add class "bar" to "div"
    bar.classList.add("bar");

    // Provide height to the bar
    bar.style.height = `${value * 3}px`;

    // Translate the bar towards positive X axis
    bar.style.transform = `translateX(${i * 30}px)`;

    // To create element "label"
    const barLabel = document.createElement("label");

    // To add class "bar_id" to "label"
    barLabel.classList.add("bar_id");

    // Assign value to "label"
    barLabel.innerHTML = value;

    // Append "Label" to "div"
    bar.appendChild(barLabel);

    // Append "div" to "data-container div"
    container.appendChild(bar);
}
}


generatebars();
```

Output

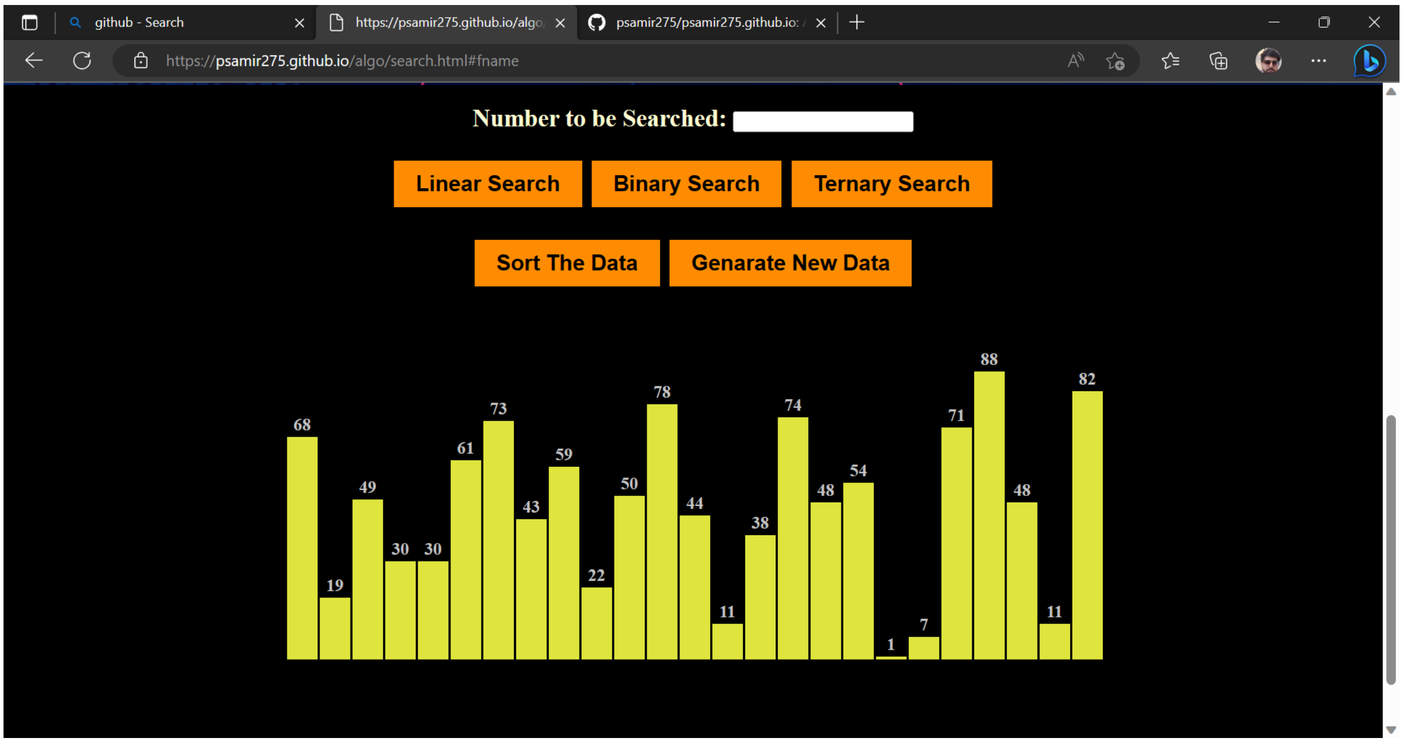

Document
127.0.0.1:5500/index.html
37
36
59
41
61
60
55
52
60
17
9
88
23
54
70
13
44
5
44
39
100
84
48
41
4
4
100
91
43

Generate New Data Function

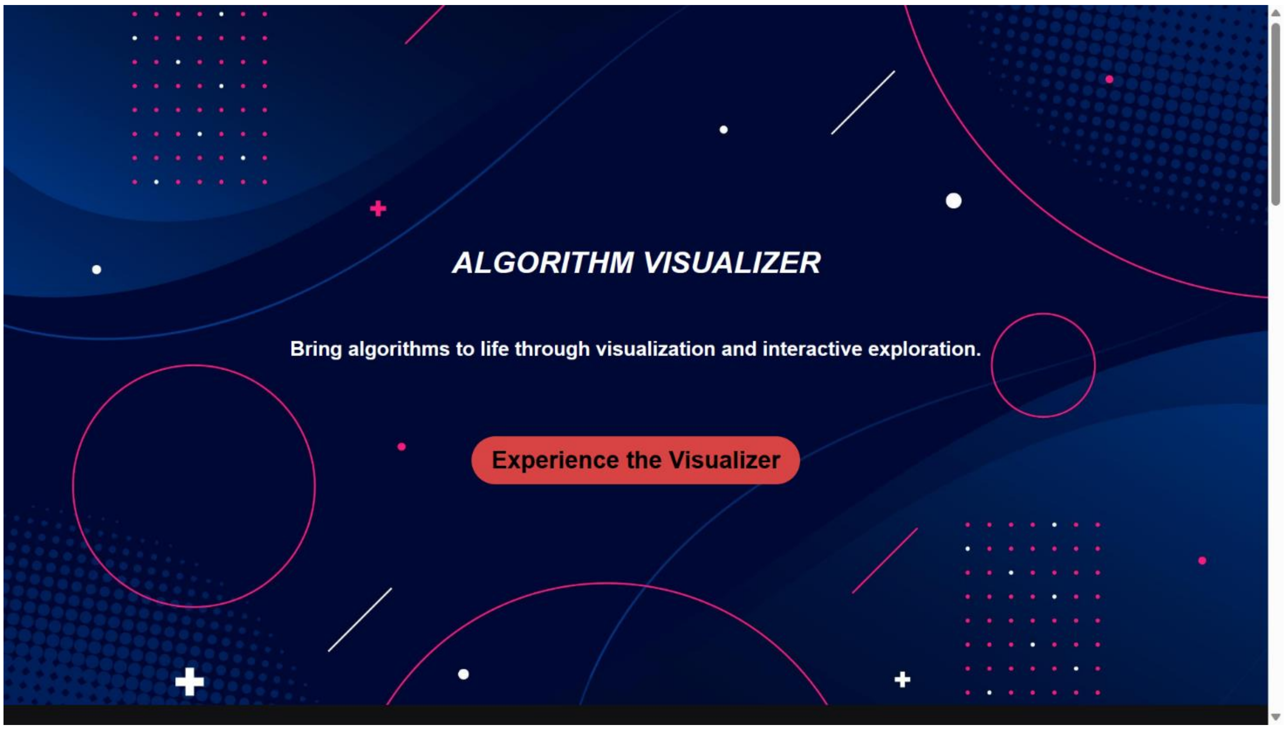
It is a sorted data


github - Search
https://psamir275.github.io/algo
psamir275/psamir275.github.io
https://psamir275.github.io/algo/search.html#fname
Number to be Searched:
Linear Search
Binary Search
Ternary Search
Sort The Data
Genarate New Data
Data is Sorted, Now You Can Perfom Binary or Ternary Search
1
1
9
10
12
15
15
24
39
44
44
45
46
53
53
63
64
64
64
69
71
72
86
97
100

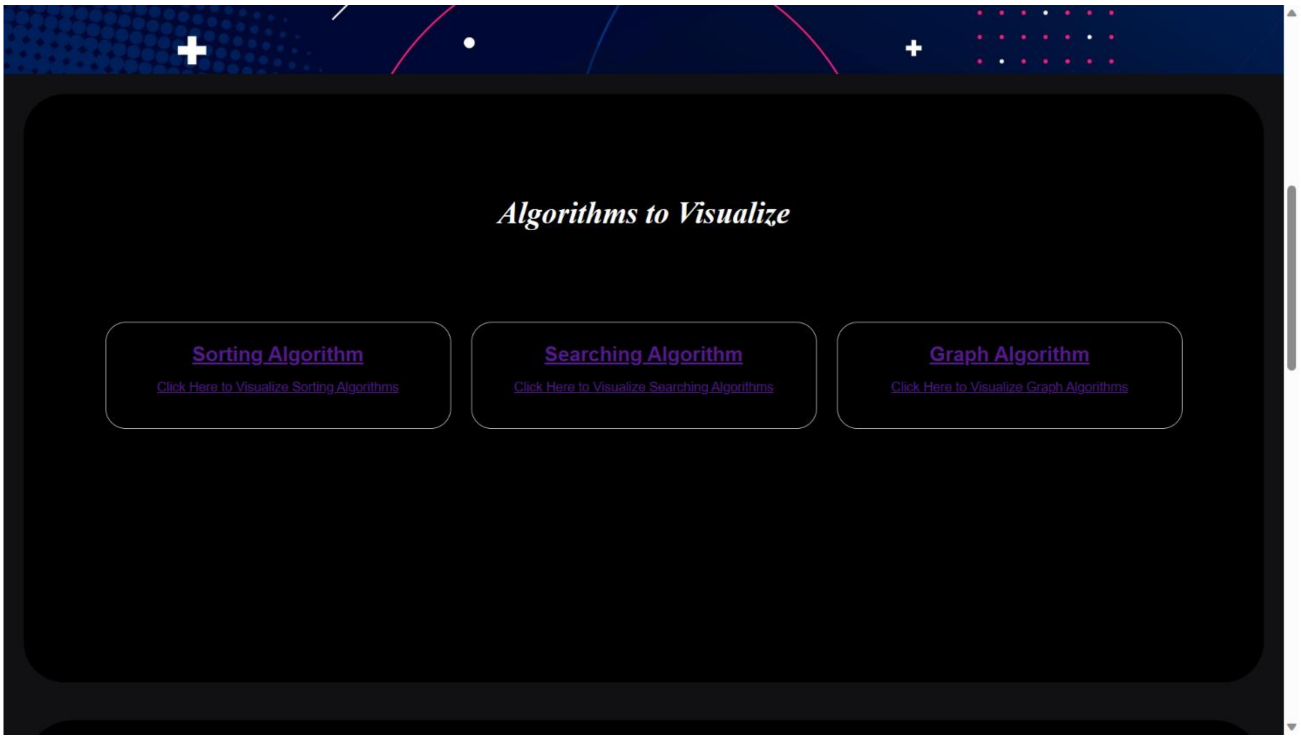
Random data generated



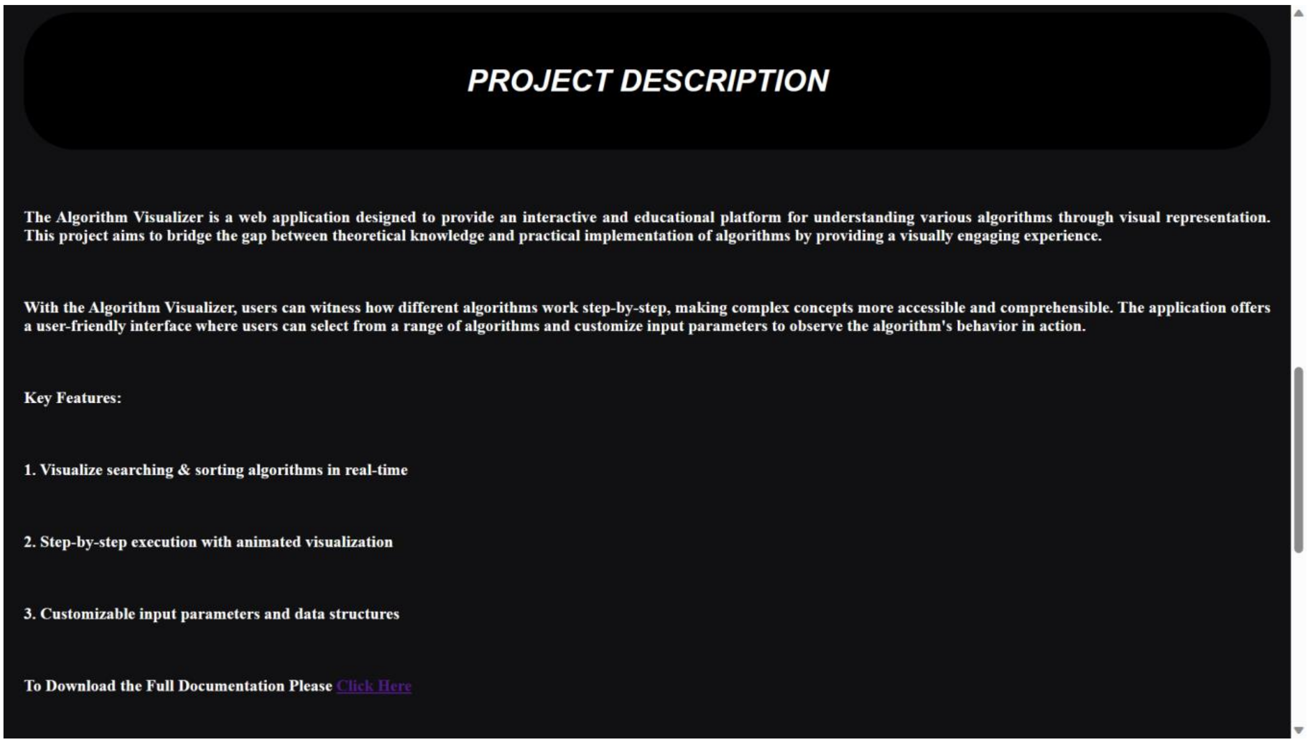
Home Page


ALGORITHM VISUALIZER
Bring algorithms to life through visualization and interactive exploration.
Experience the Visualizer

Algorithmes Visualizer Section


Algorithms to Visualize
Sorting Algorithm
Click Here to Visualize Sorting Algorithms
Searching Algorithm
Click Here to Visualize Searching Algorithms
Graph Algorithm
Click Here to Visualize Graph Algorithms

Project Description Section



PROJECT DESCRIPTION

The Algorithm Visualizer is a web application designed to provide an interactive and educational platform for understanding various algorithms through visual representation. This project aims to bridge the gap between theoretical knowledge and practical implementation of algorithms by providing a visually engaging experience.

With the Algorithm Visualizer, users can witness how different algorithms work step-by-step, making complex concepts more accessible and comprehensible. The application offers a user-friendly interface where users can select from a range of algorithms and customize input parameters to observe the algorithm's behavior in action.

Key Features:

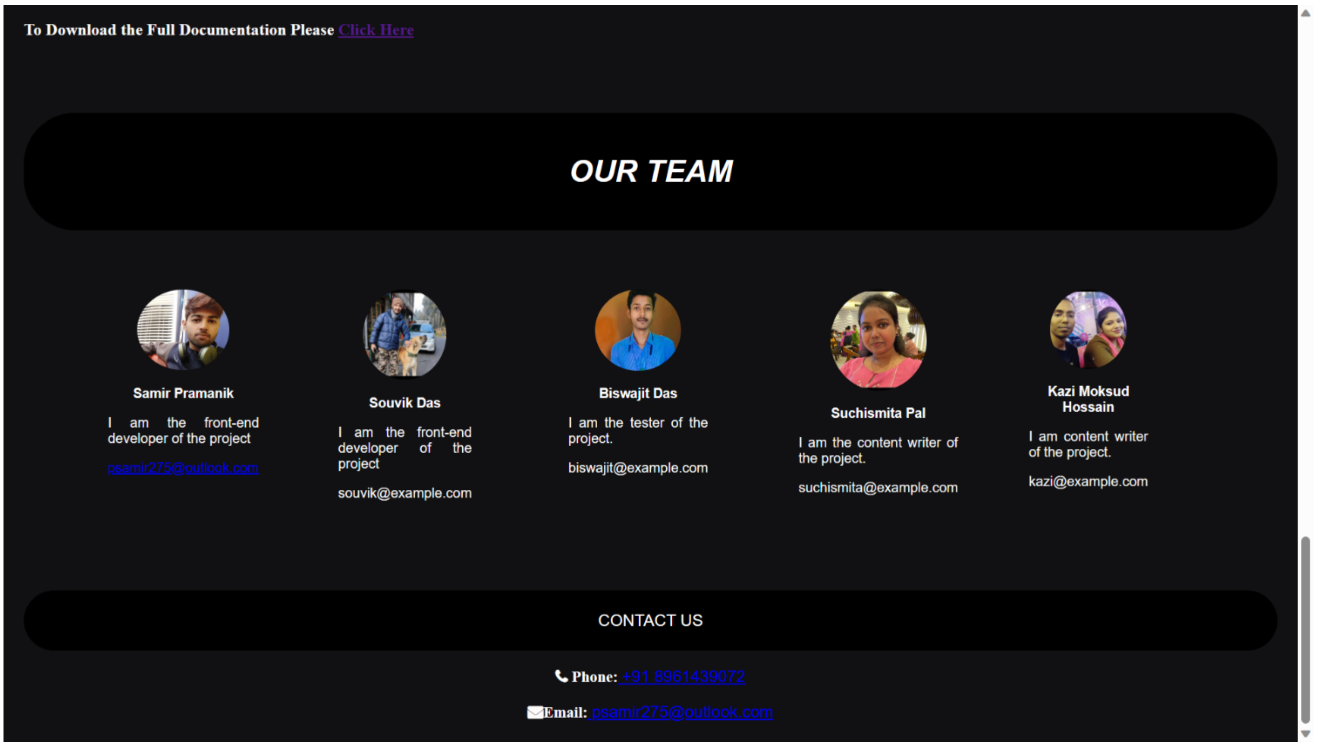
1. Visualize searching & sorting algorithms in real-time

2. Step-by-step execution with animated visualization

3. Customizable input parameters and data structures

To Download the Full Documentation Please Click Here

About Us & Contact Us Section



To Download the Full Documentation Please Click Here

OUR TEAM

Samir Pramanik
I am the front-end developer of the project
psamir275@outlook.com

Souvik Das
I am the front-end developer of the project
souvik@example.com

Biswajit Das
I am the tester of the project.
biswajit@example.com

Suchismita Pal
I am the content writer of the project.
suchismita@example.com

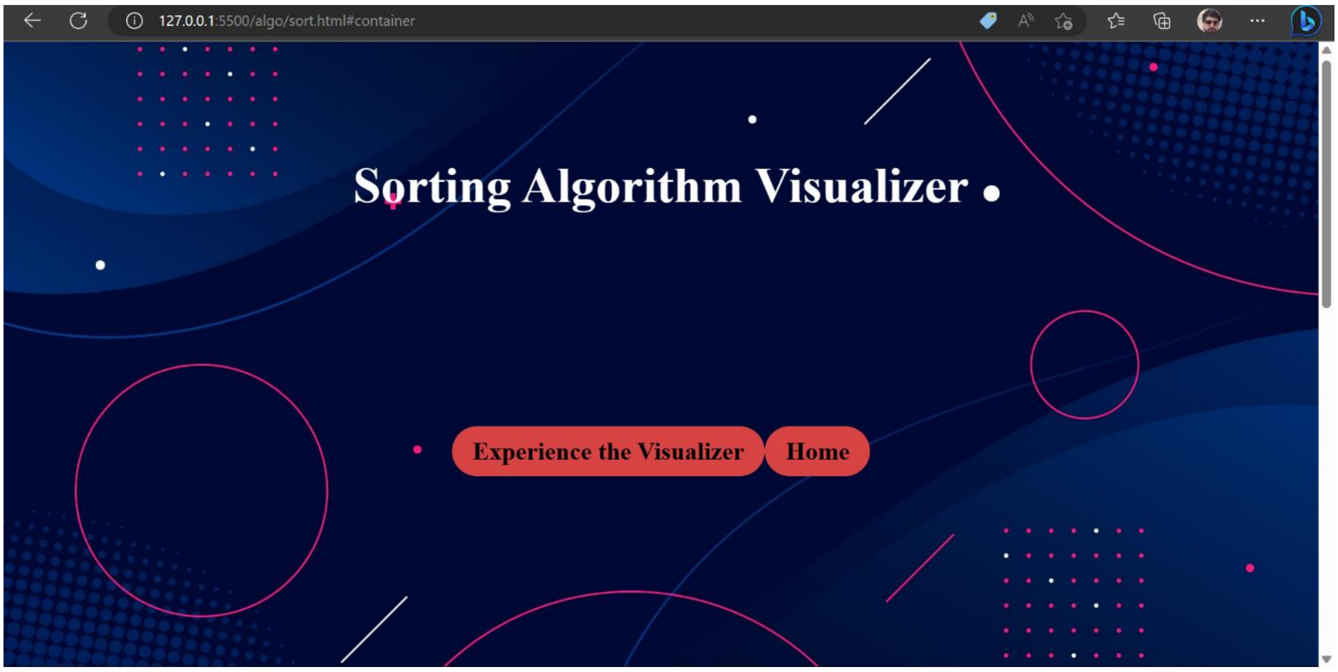
Kazi Moksud Hossain
I am content writer of the project.
kazi@example.com

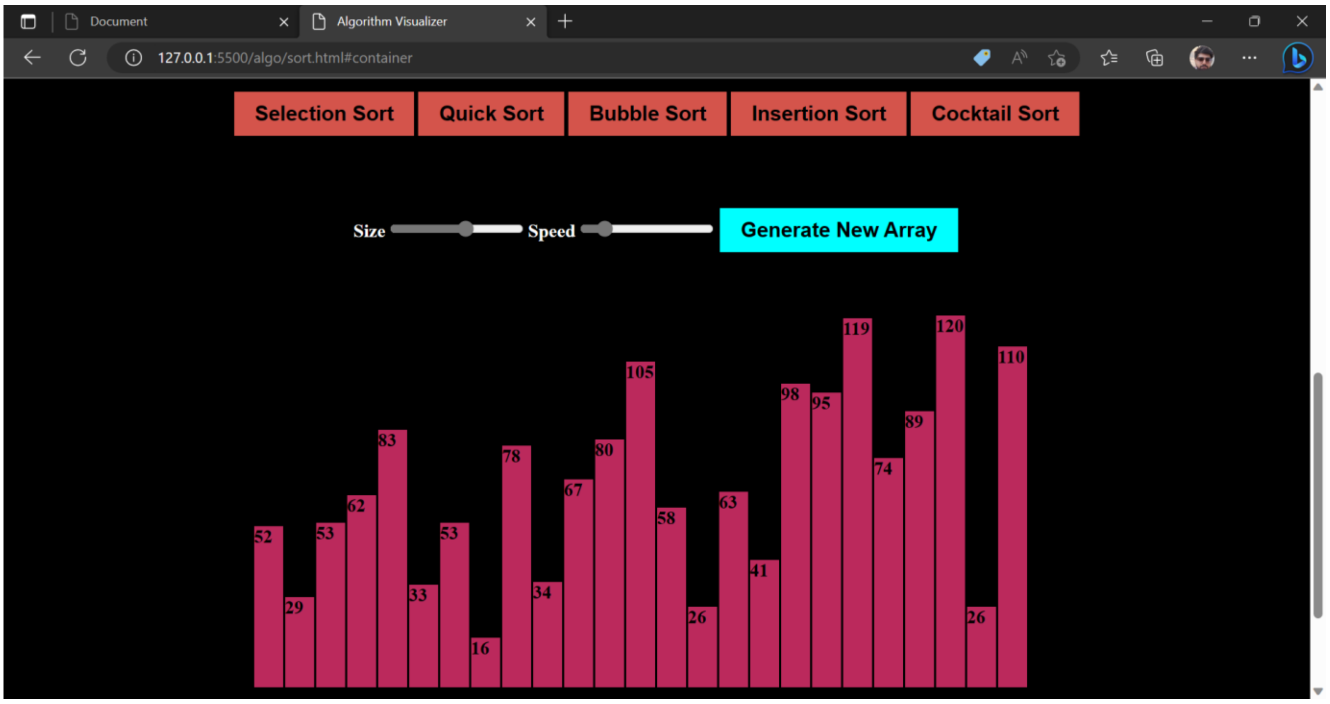
CONTACT US

Phone: +91 8961439072

Email: psamir275@outlook.com

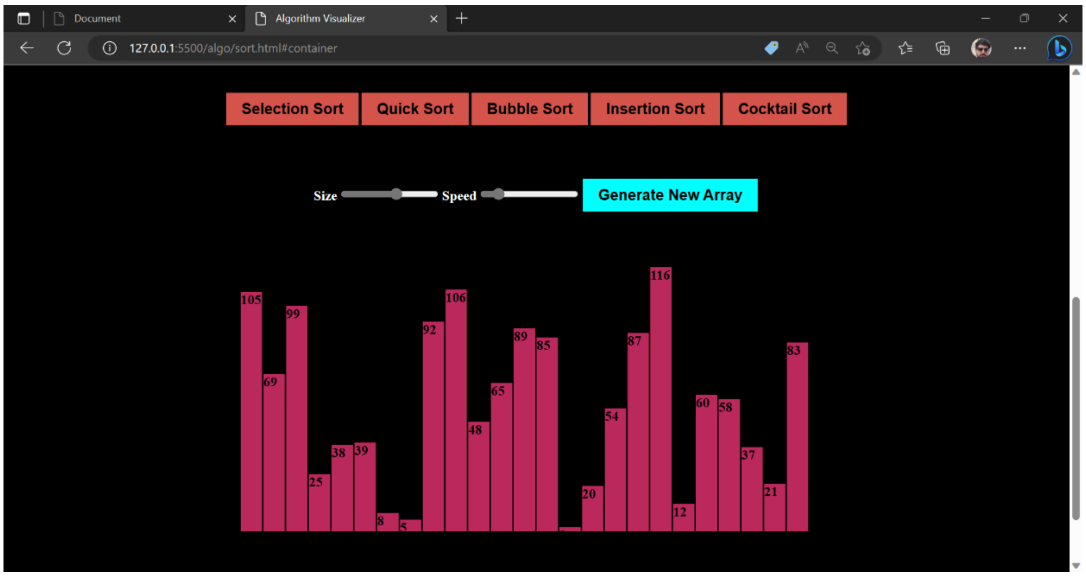
Sorting Algorithm


127.0.0.1:5500/algo/sort.html#container
Sorting Algorithm Visualizer
Experience the Visualizer
Home

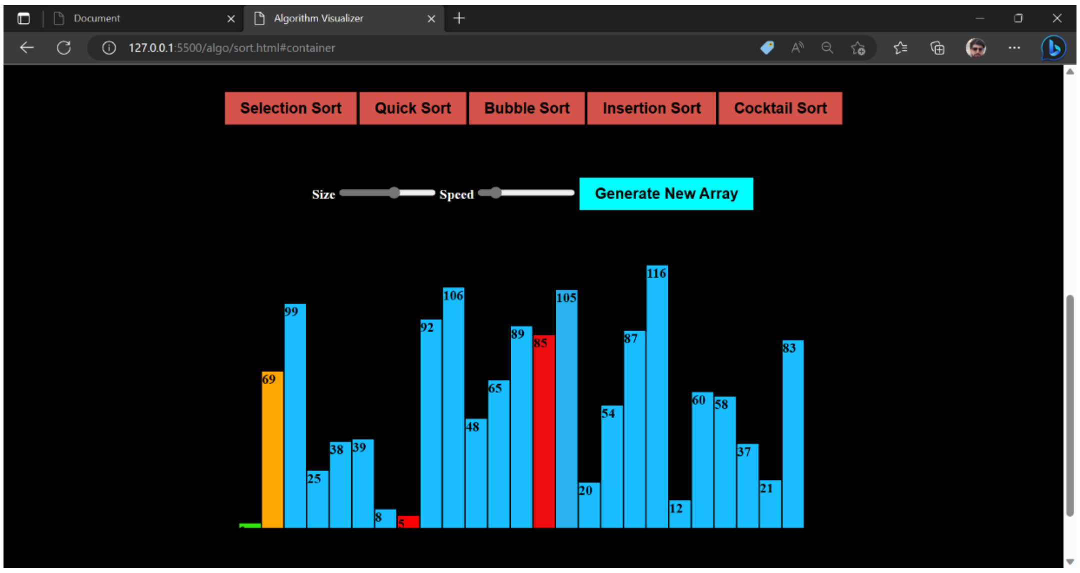


Selection Sort

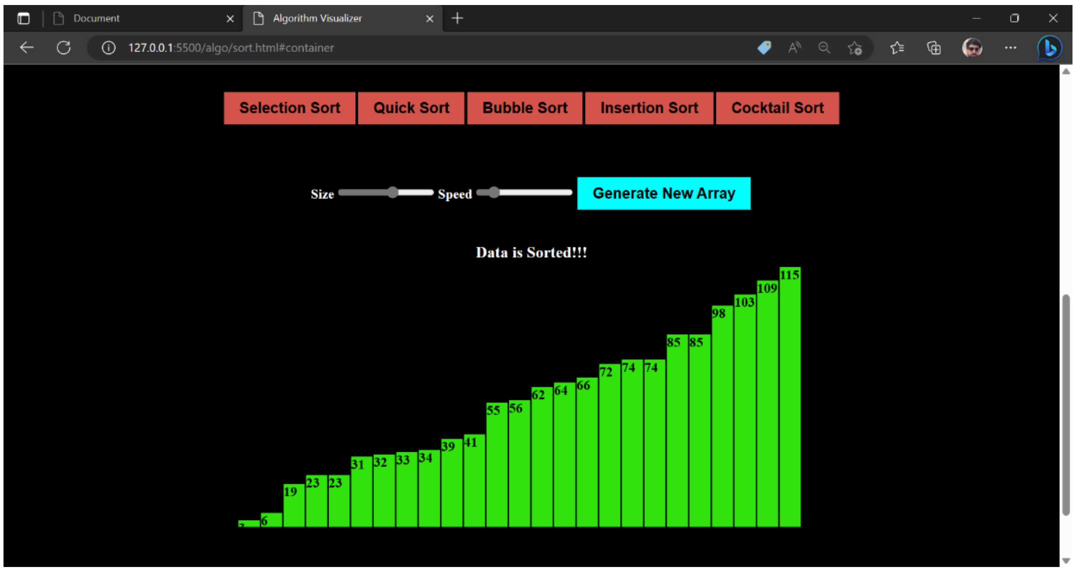
Generate New Data



Performing Selection Sort

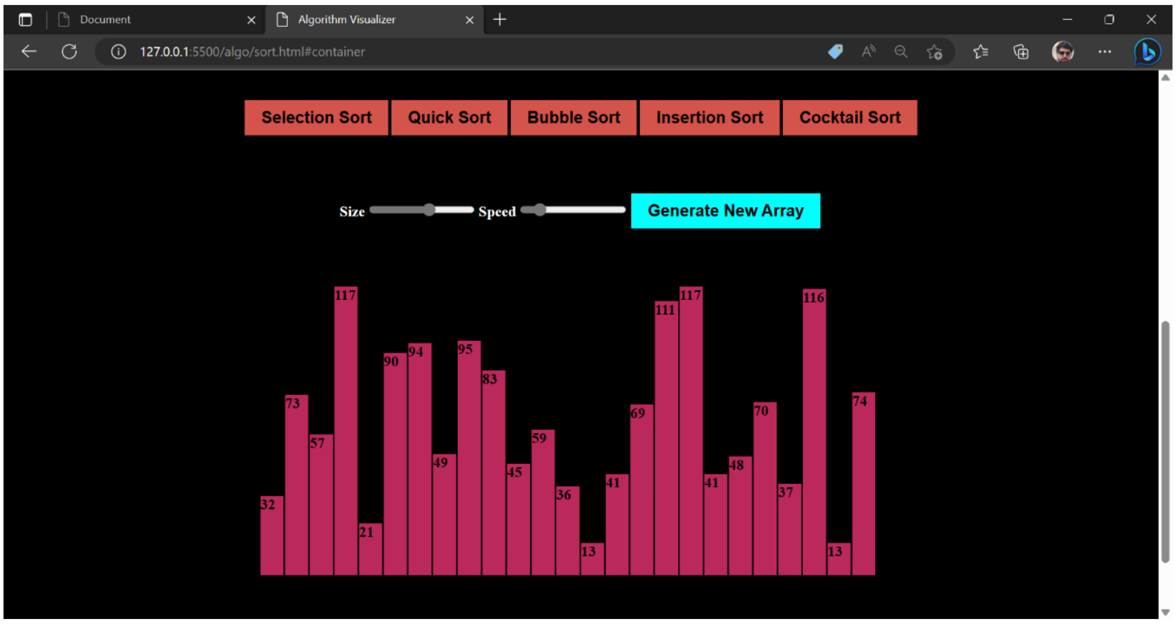


After the Sorting

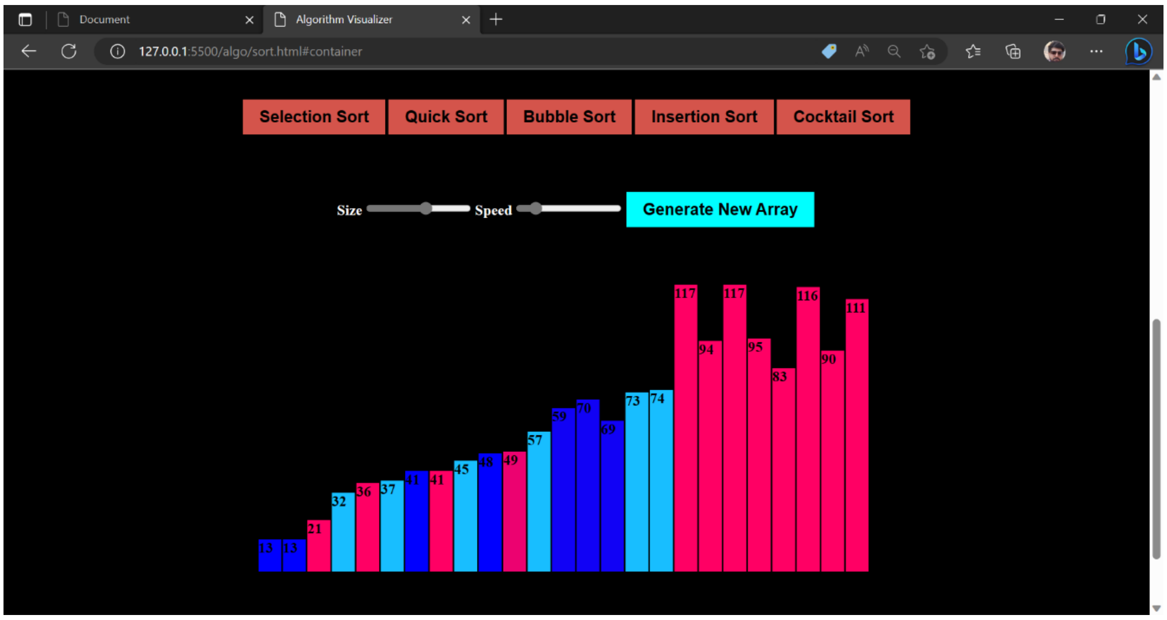


Quick Sort

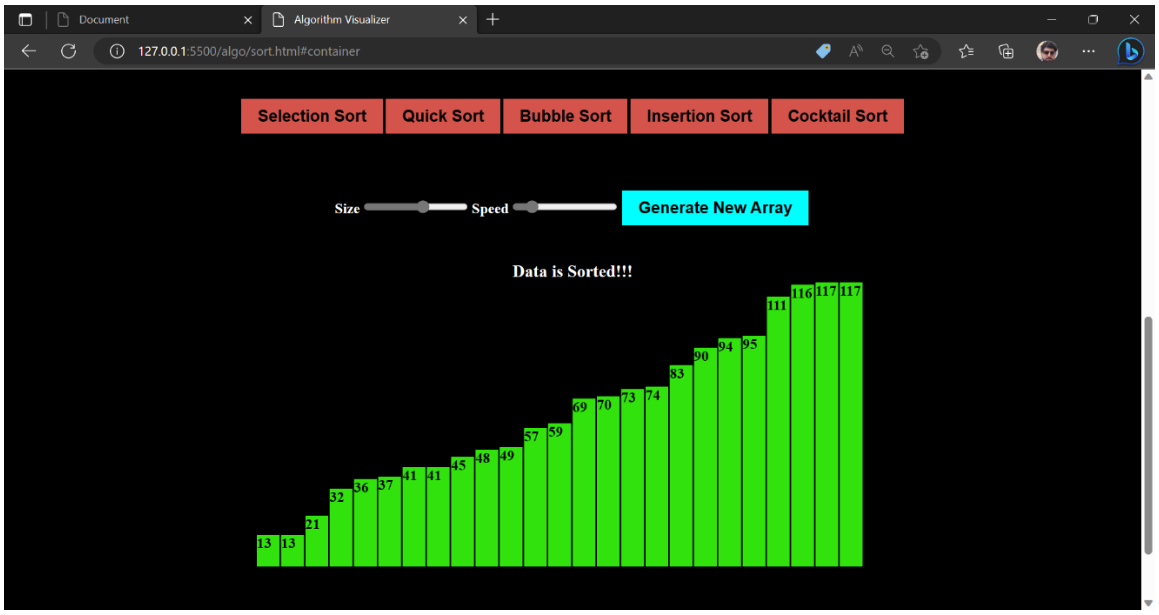
Generate New Data



Performing Quick Sort

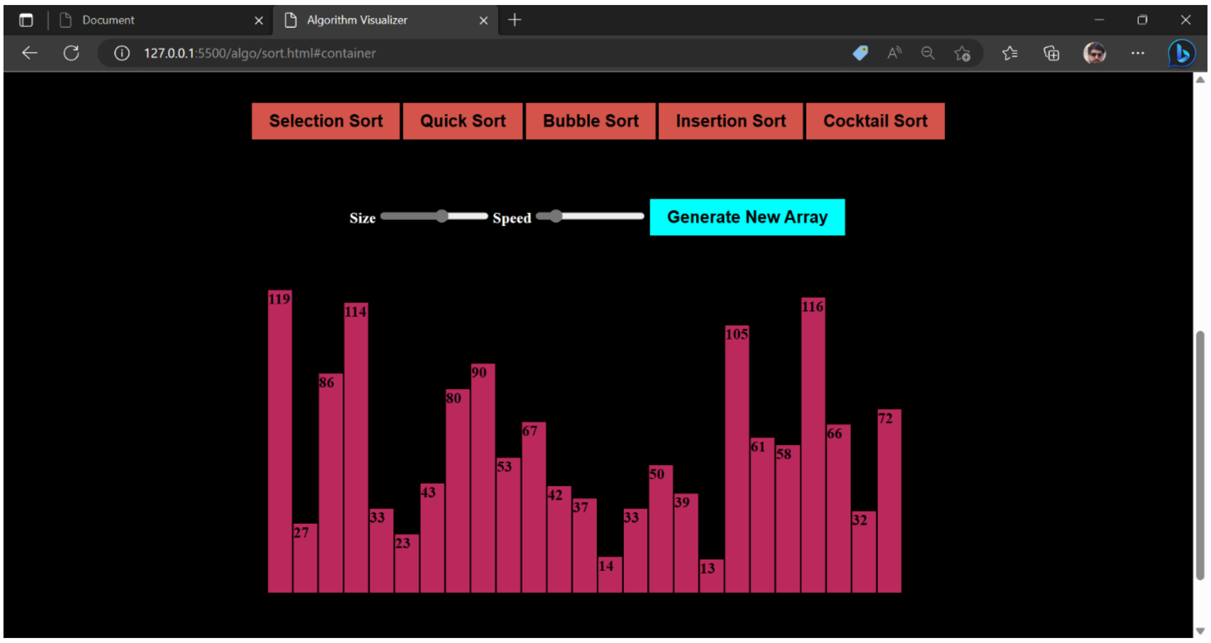


After the Sorting

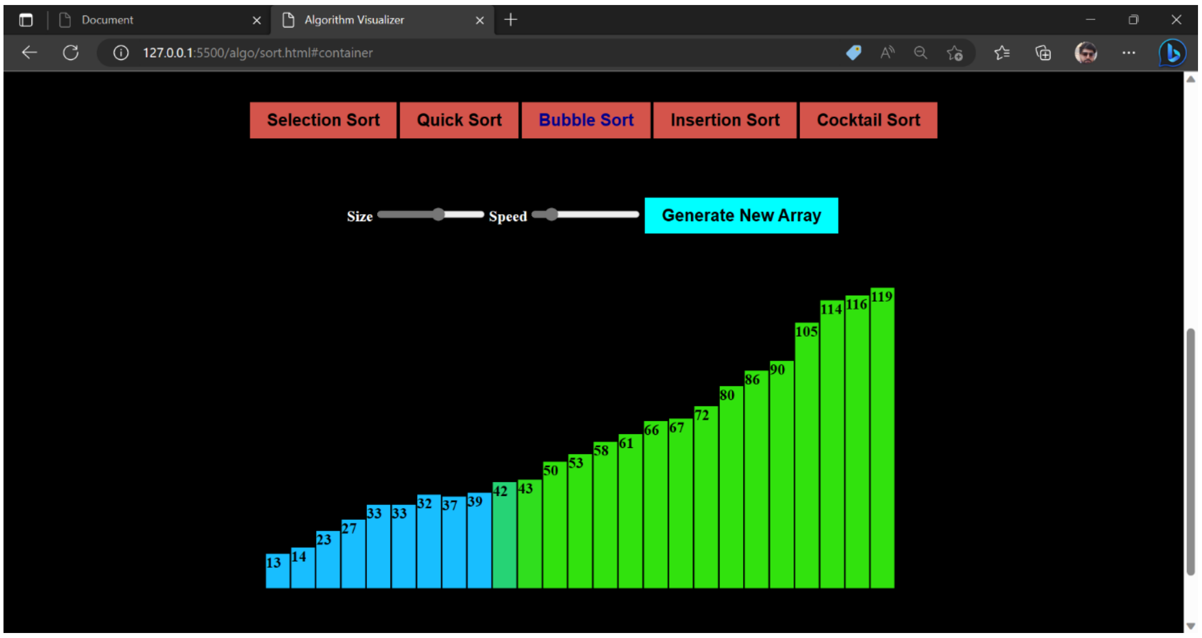


Bubble Sort

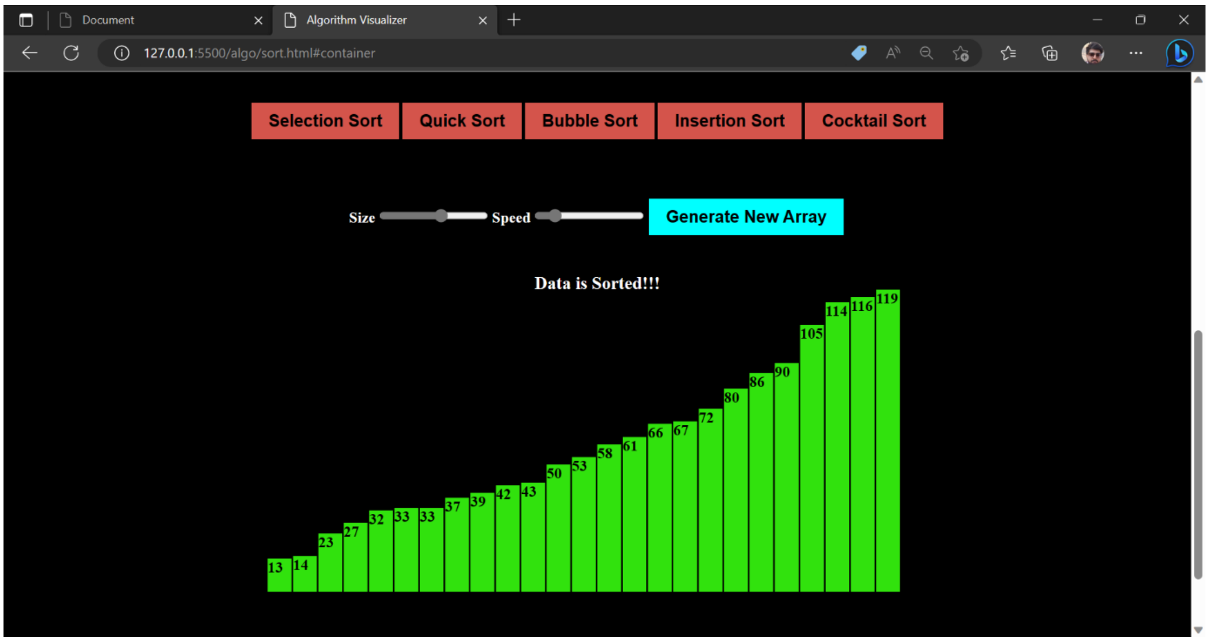
Generate New Data



Performing Bubble Sort



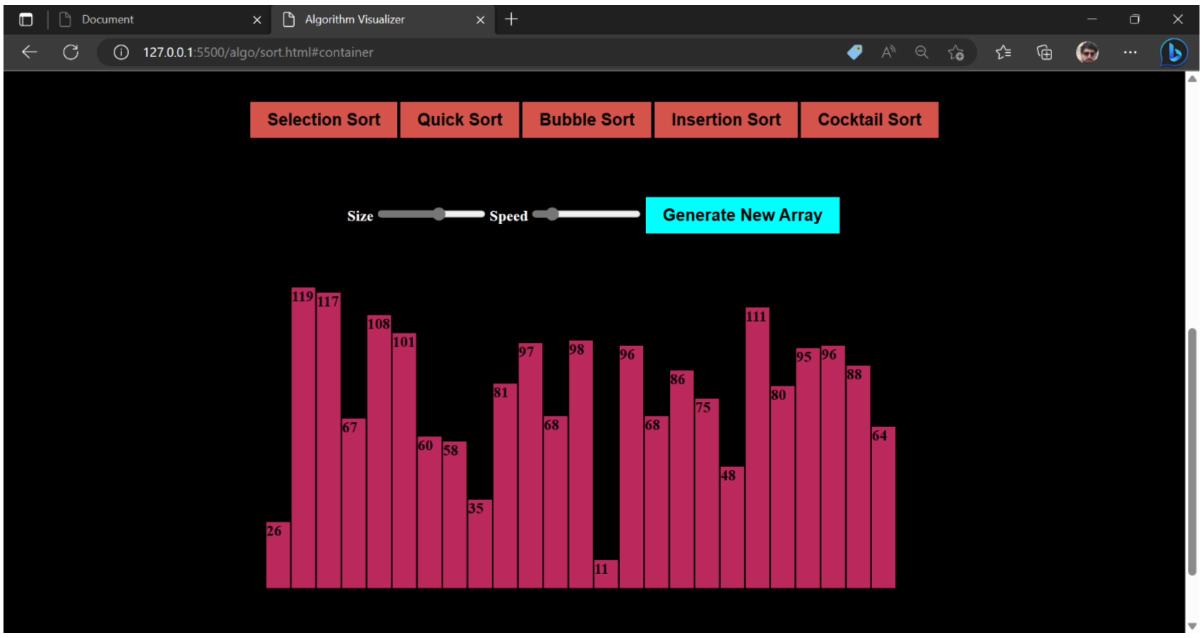
After the Sorting



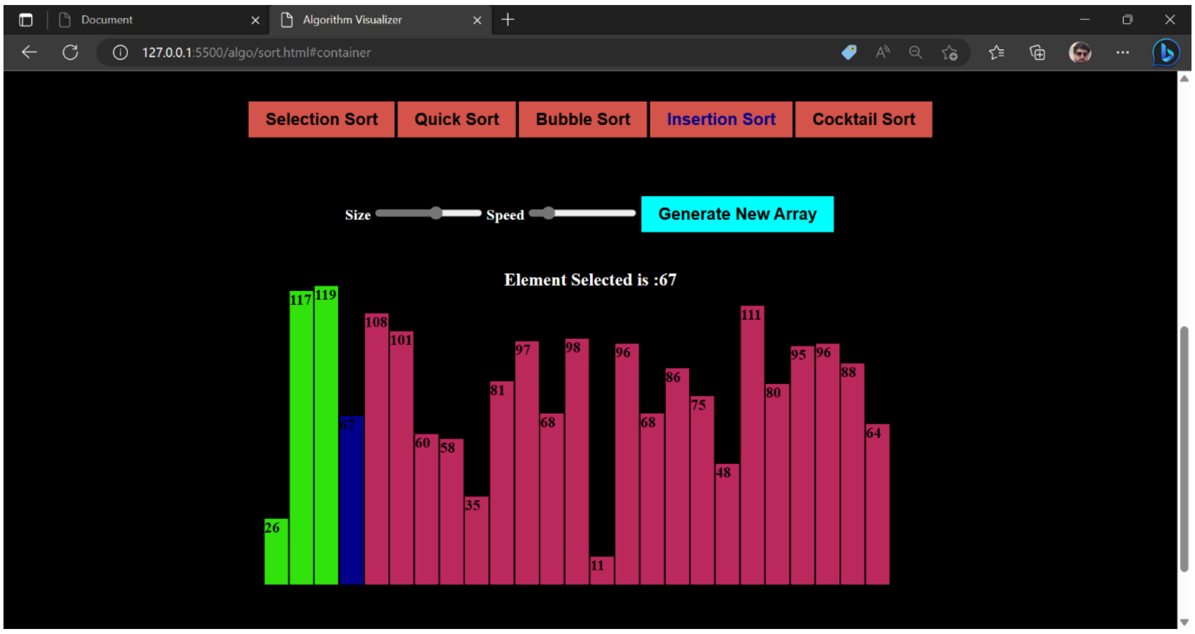
Data is Sorted!!!

Insertion Sort

Generate New Data

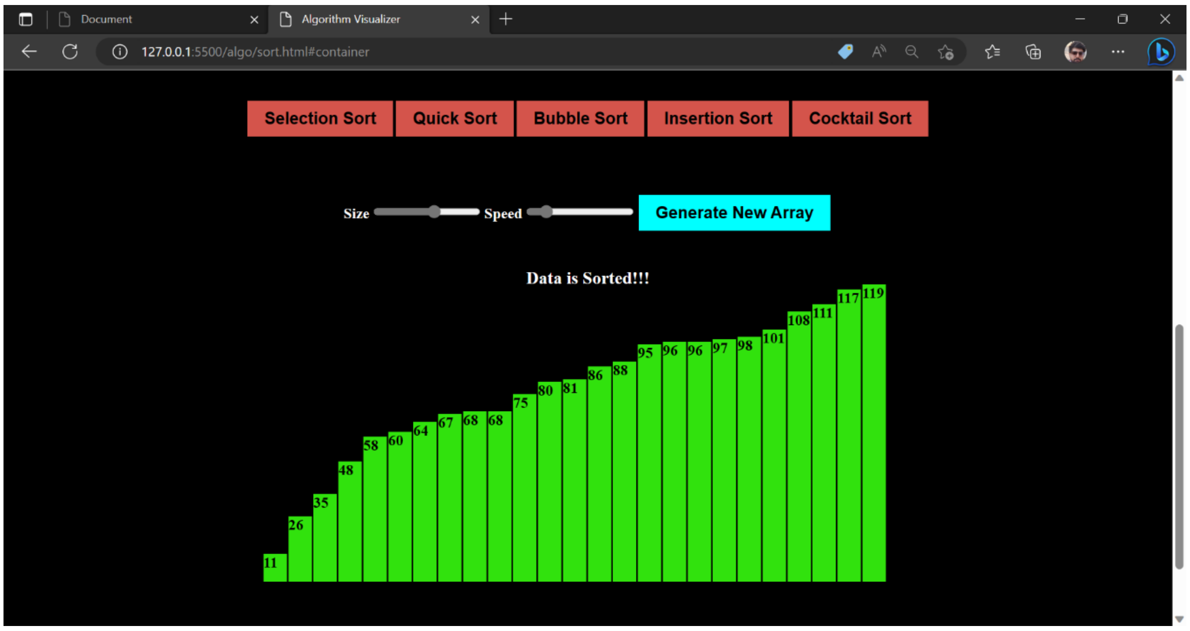


Performing Insertion Sort



Element Selected is :67

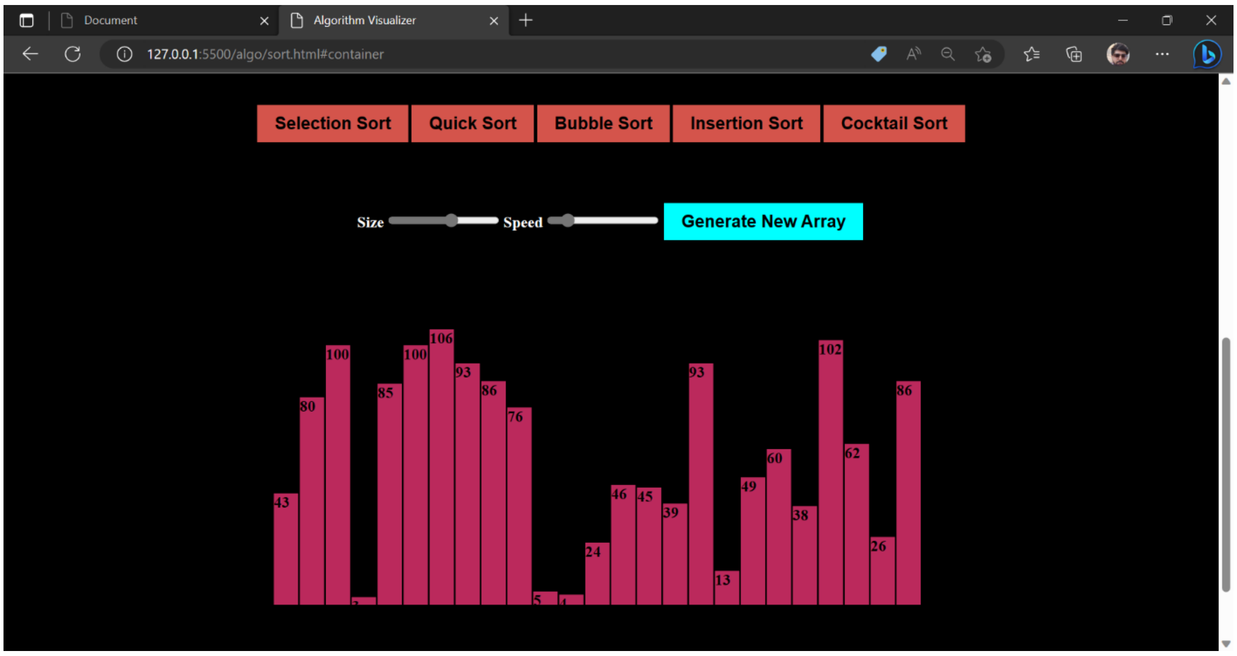
After the Sorting



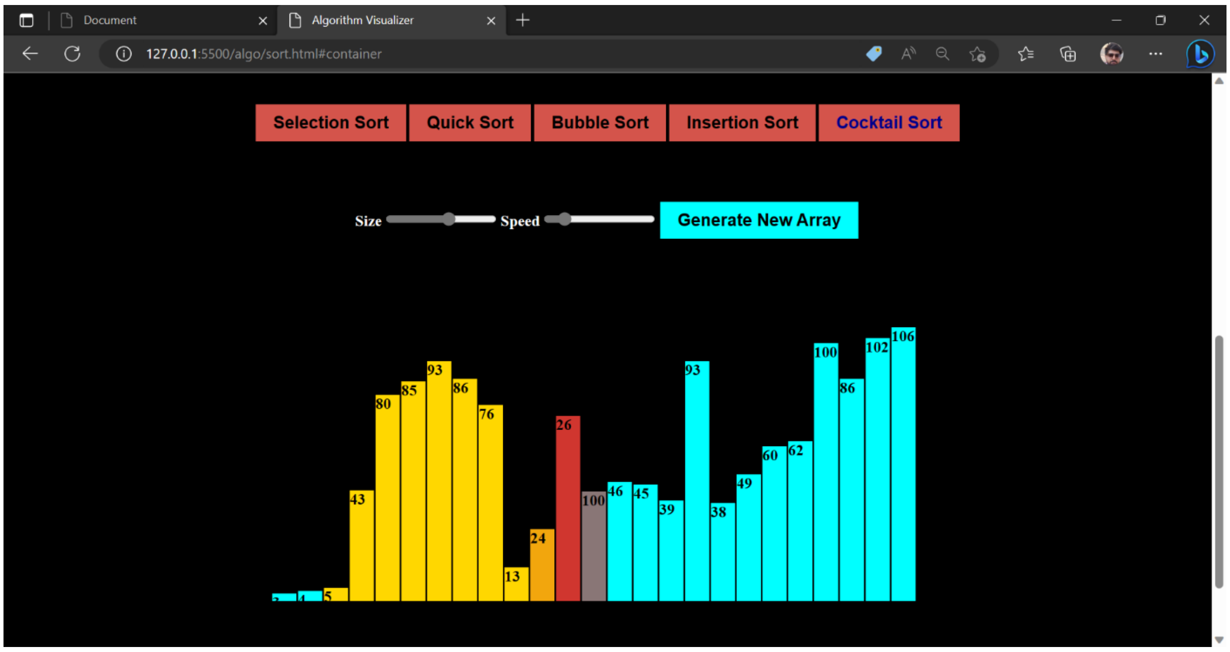
Data is Sorted!!!

Cocktail Sort

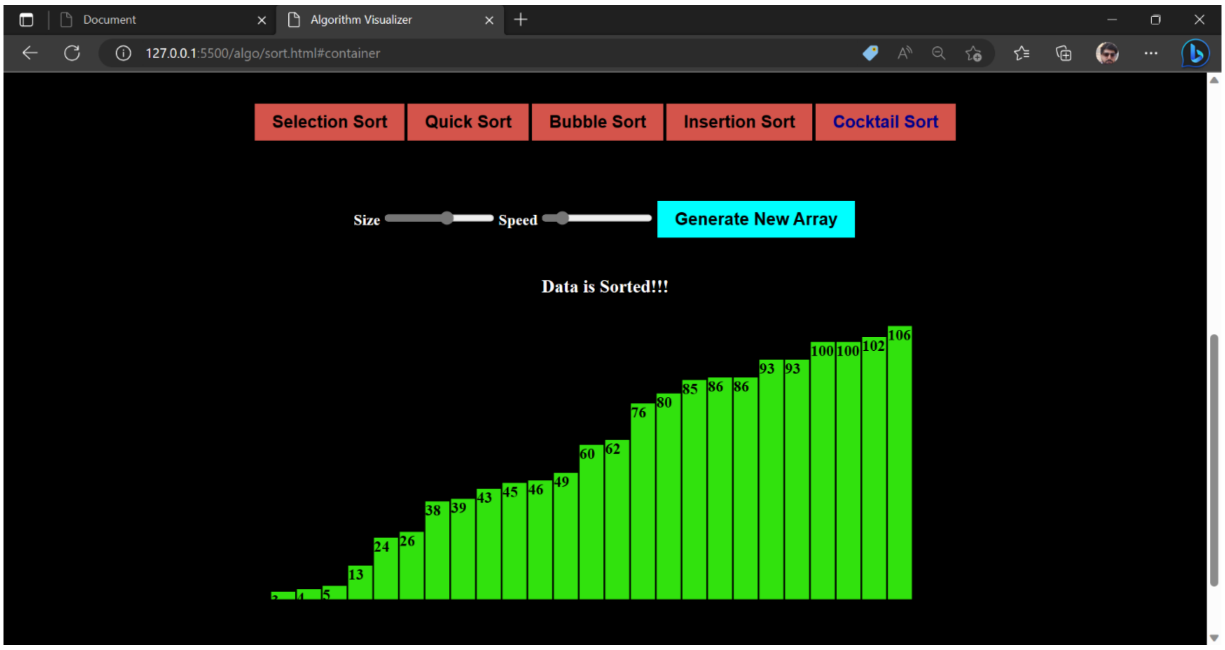
Generate New Data



Performing Cocktail Sort

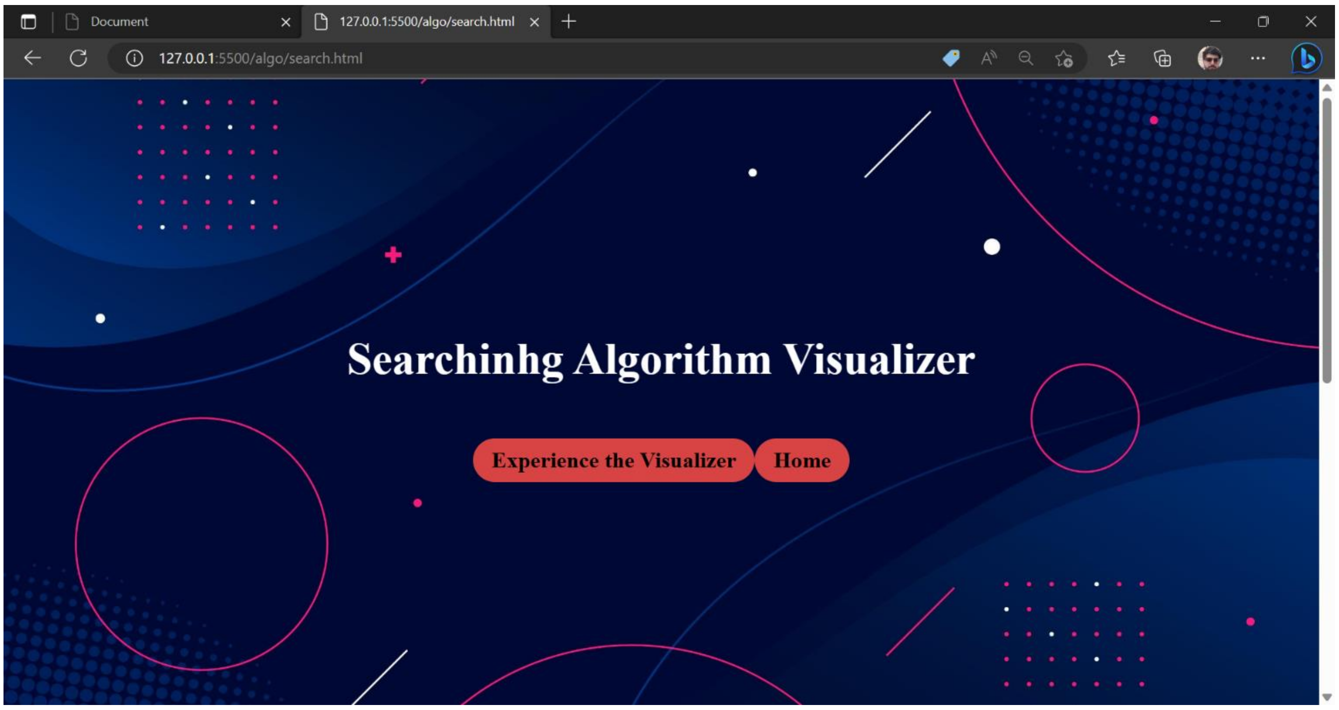


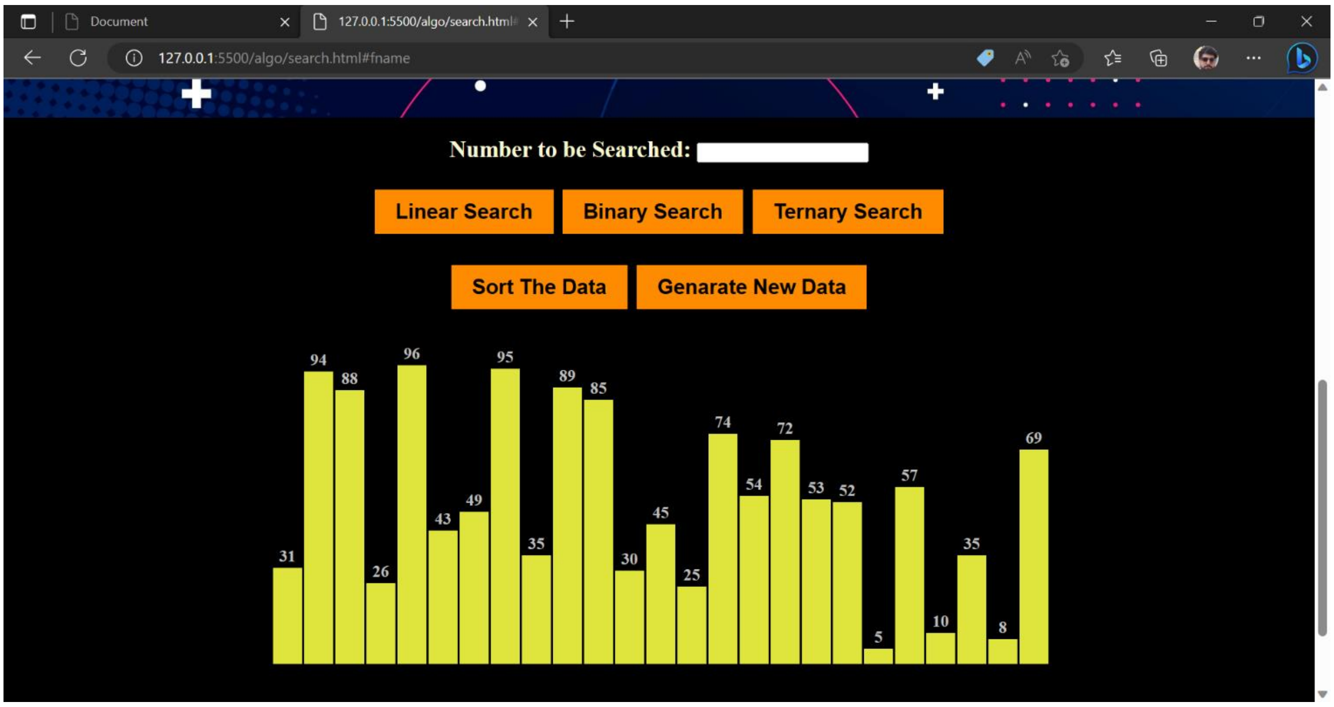
After the Sorting



Data is Sorted!!!

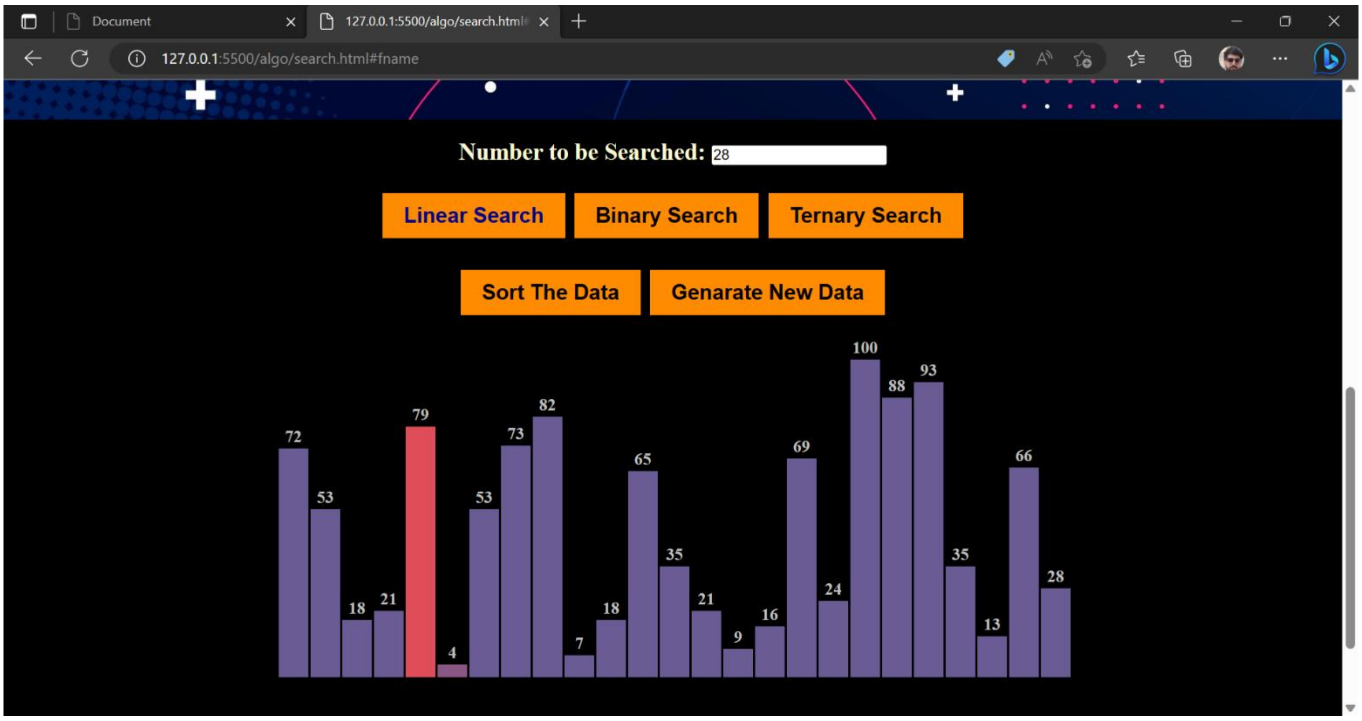
Searching Algorithm

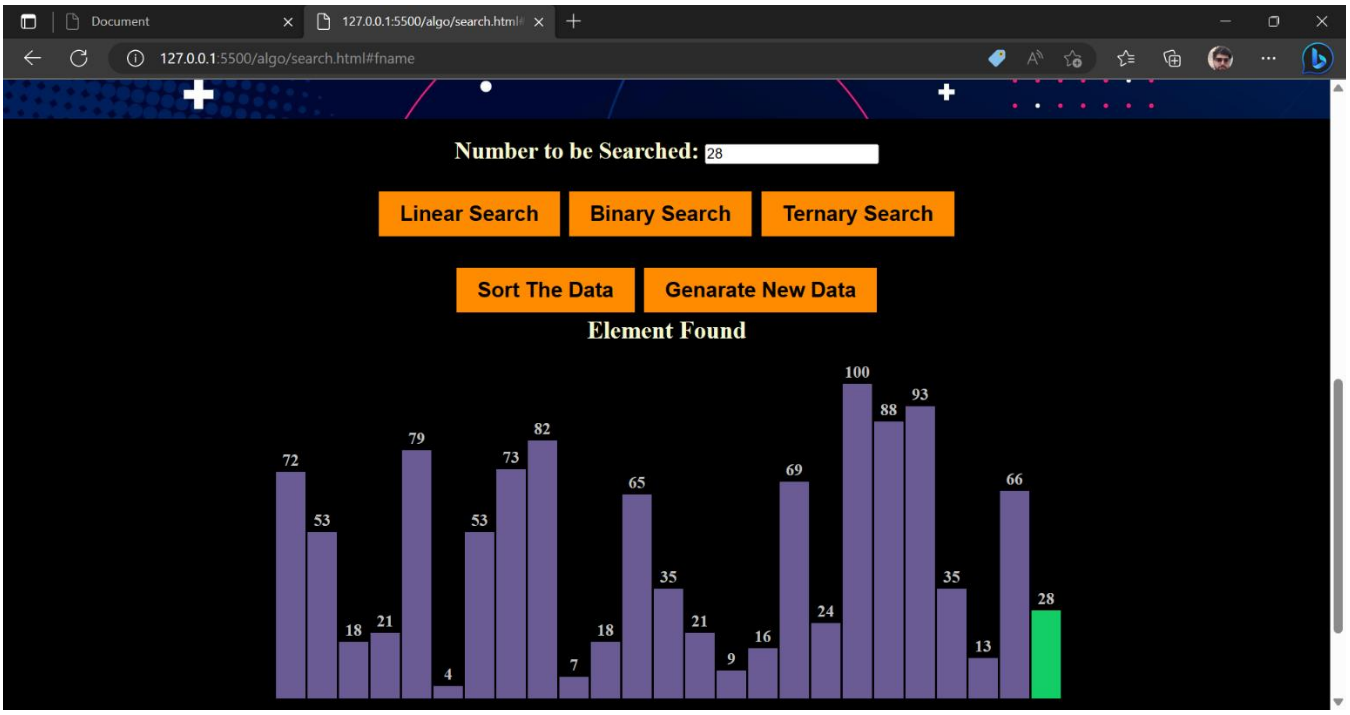

Document
127.0.0.1:5500/algo/search.html
127.0.0.1:5500/algo/search.html
Searchinhg Algorithm Visualizer
Experience the Visualizer
Home



Linear Search

Performing Linear search to find element 28

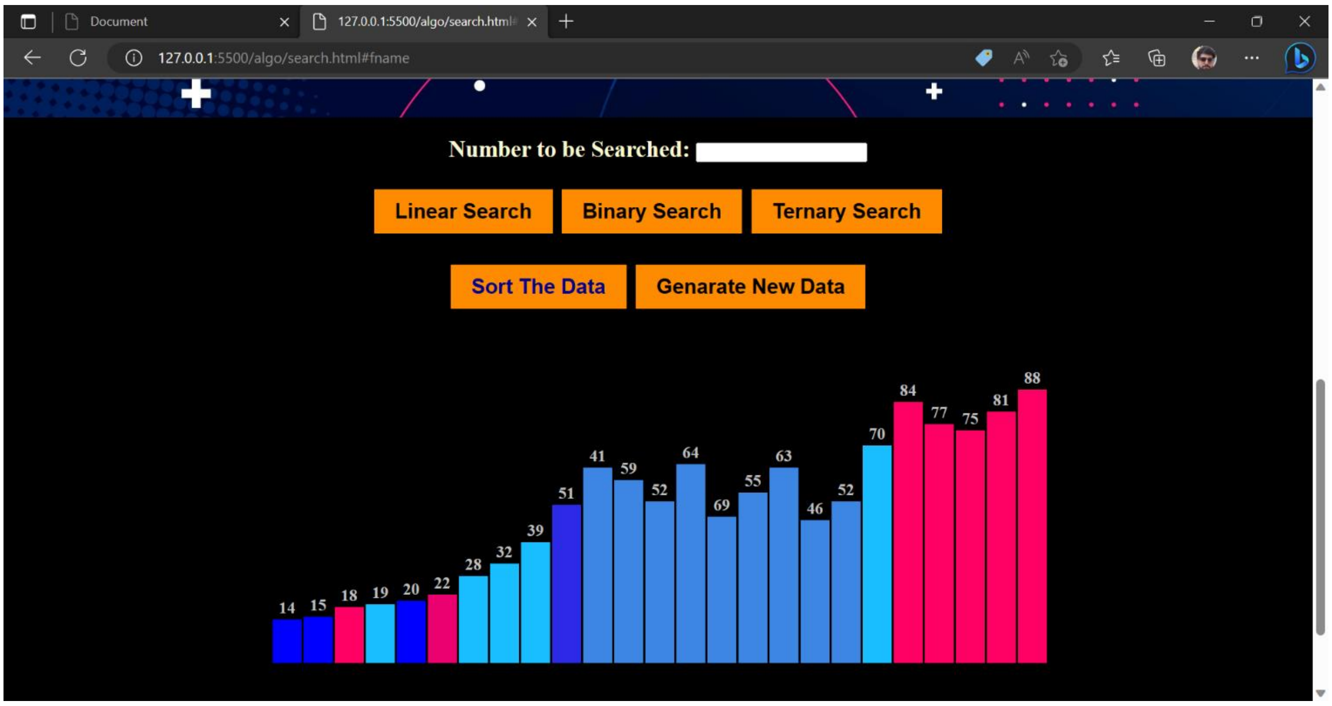


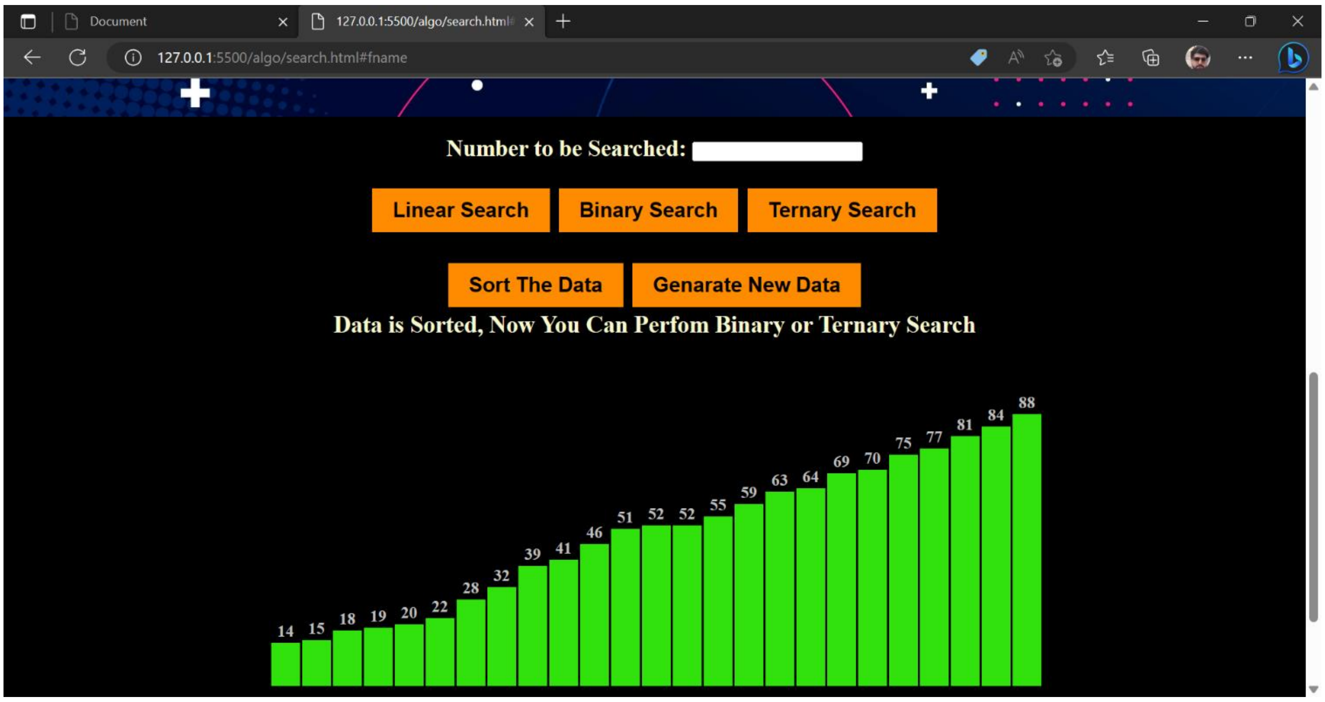


Element Found

Sort function for Binary or Ternary Search

Performing Quick Sort

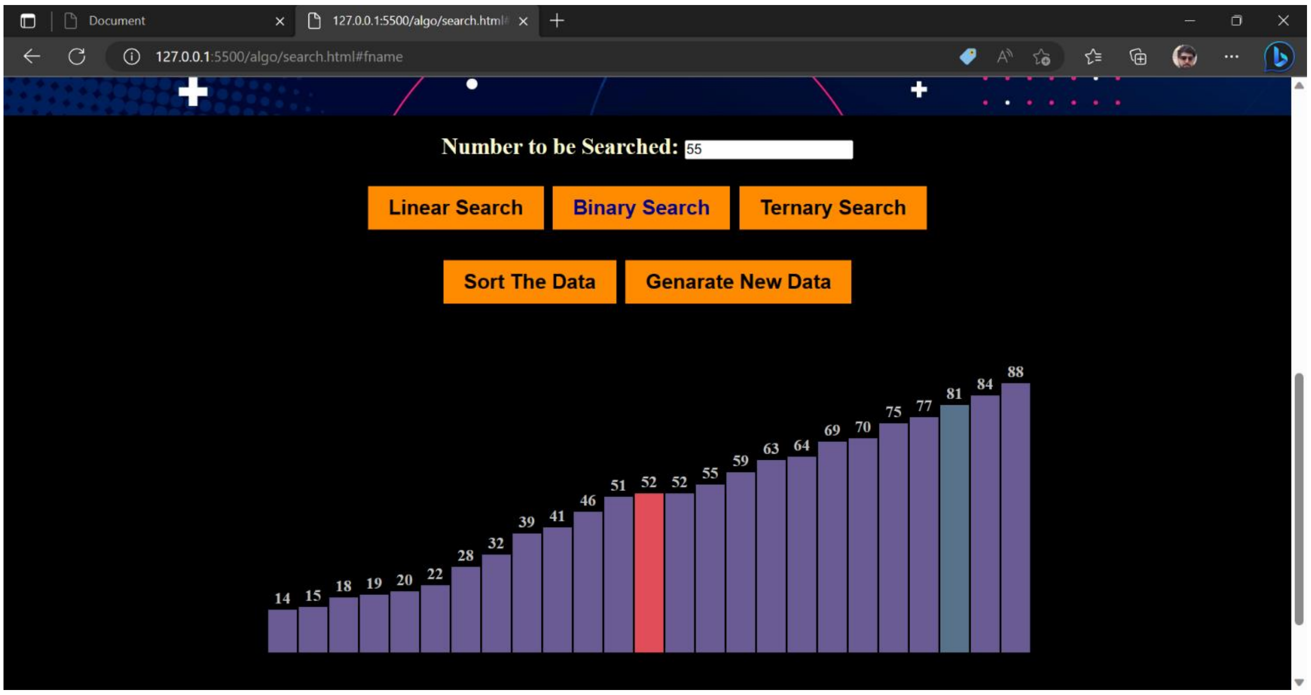


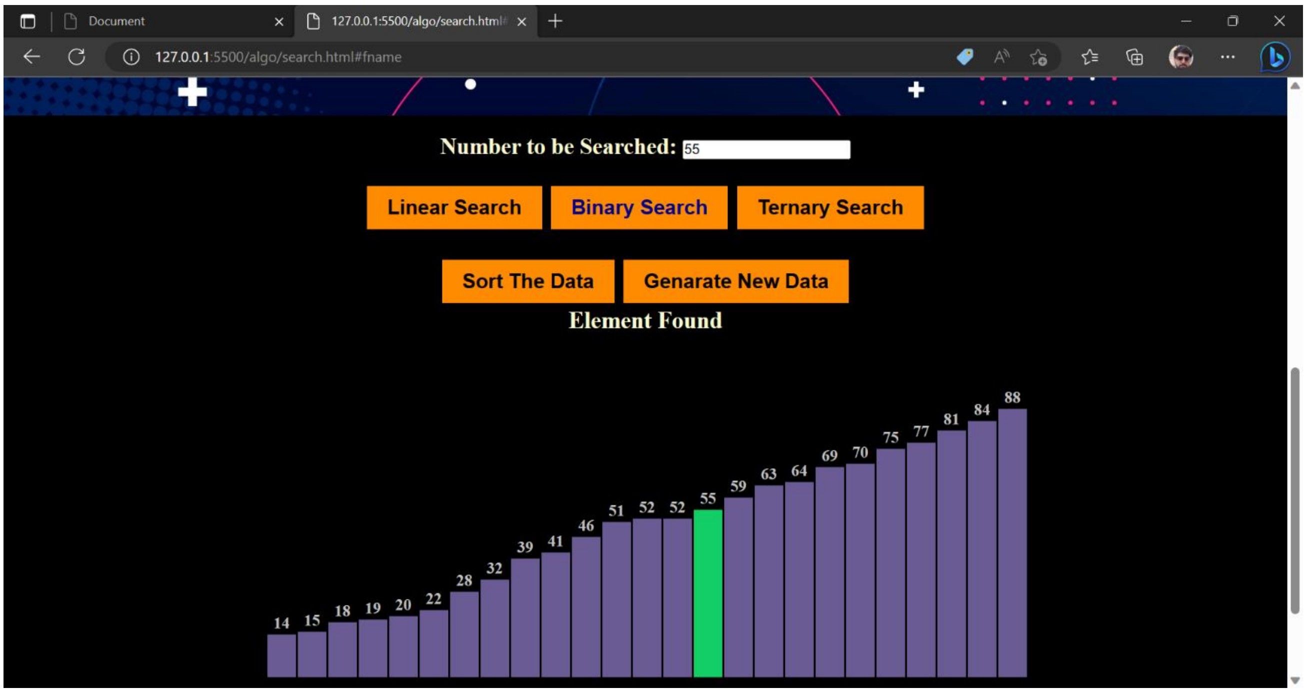


Data is Sorted, Now You Can Perfom Binary or Ternary Search

Binary Search

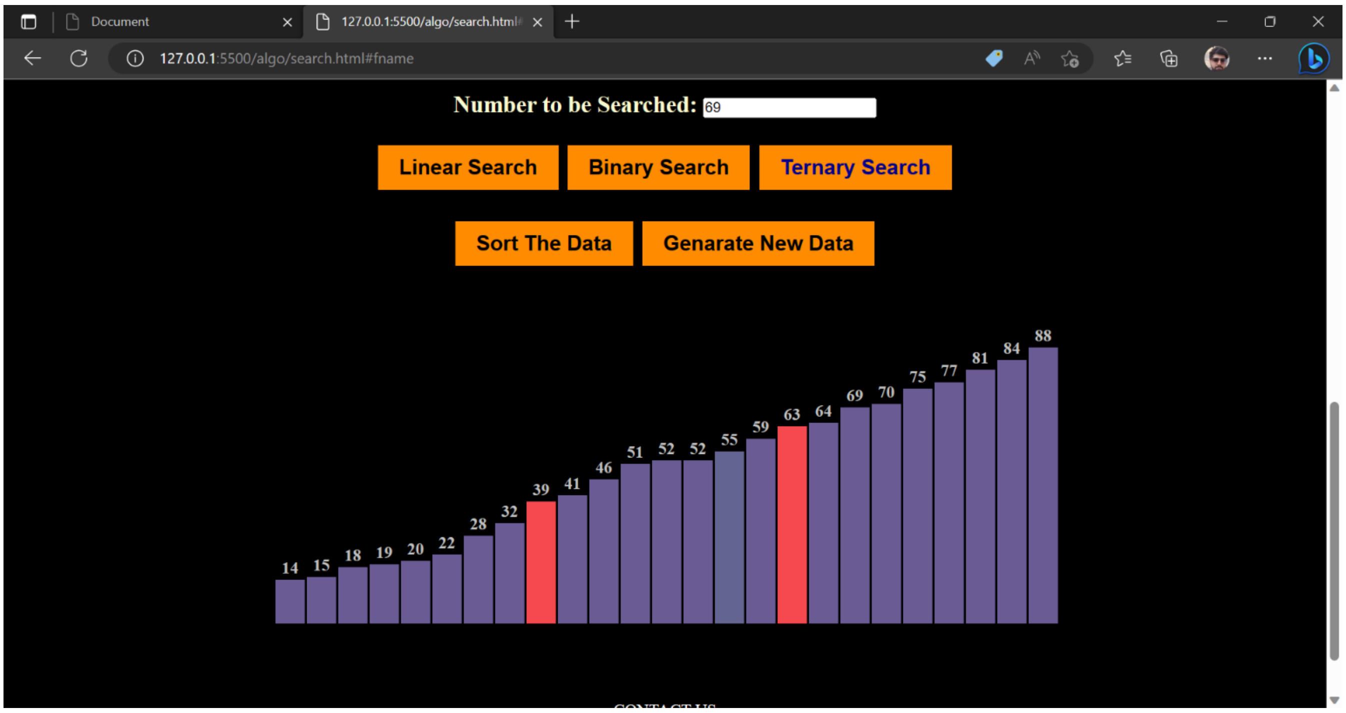
Performing Binary search to find element 55

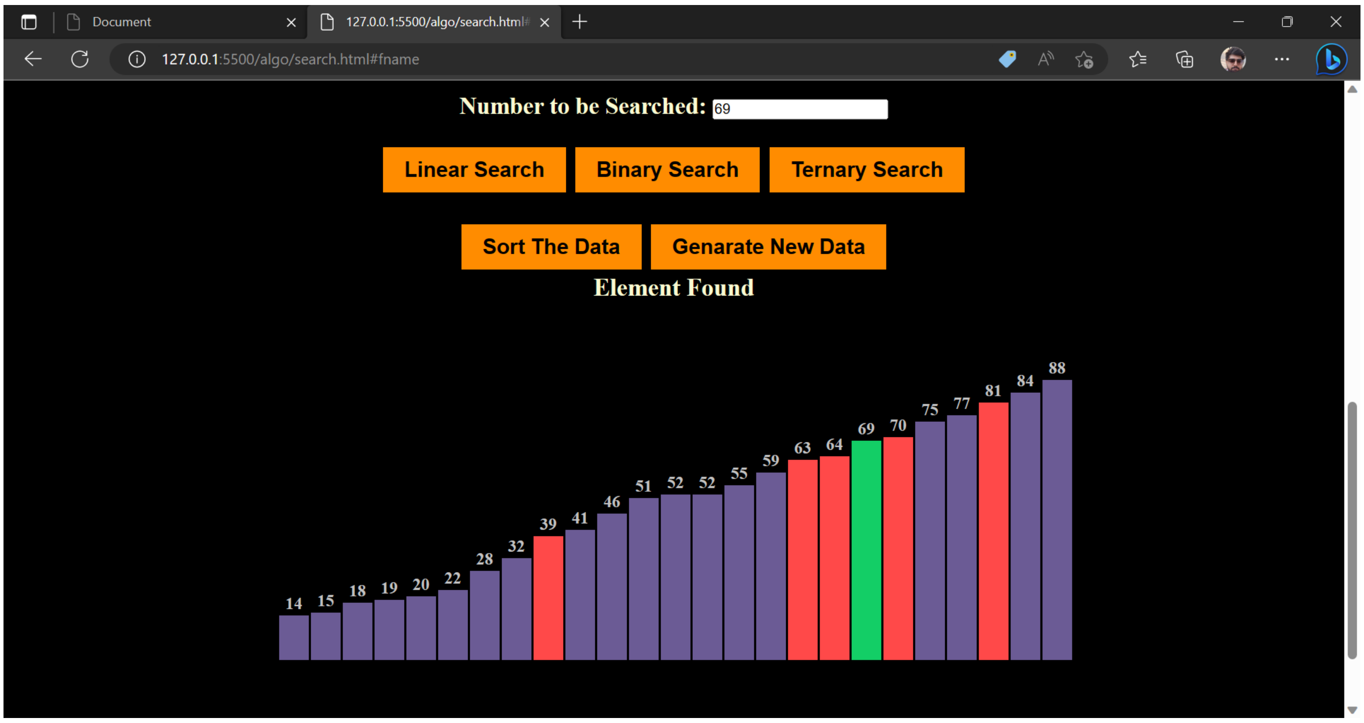




Ternary Search

Performing Ternary search to find element 69





CONCLUSION and FUTURE SCOPE

Algorithm visualization has become an essential tool for computer science students and researchers to learn and understand complex algorithms. It is a powerful technique that helps learners to visualize and understand the working of algorithms, which is crucial for their success in the field of computer science.

In conclusion, algorithm visualization is a rapidly evolving field, and the future scope looks promising. With the advent of new technologies and advancements in the field of computer science, there is a lot of potential for algorithm visualization to become more interactive and engaging. Moreover, with the increasing demand for online learning, algorithm visualization tools can be utilized to develop more effective online courses that cater to the needs of a broader audience.

In the future, algorithm visualization could also be used in other fields like mathematics and engineering to help learners understand complex concepts. Additionally, with the rise of machine learning and artificial intelligence, algorithm visualization could play a crucial role in understanding and explaining how these algorithms work.

Overall, algorithm visualization has come a long way, and with its many benefits, it is set to become an even more important tool for teaching and learning in the field of computer science and beyond.

REFERENCES

1. E-learning Tool for Visualization of Shortest Paths Algorithms” by Daniela Borissova and Ivan Mustakerov, ResearchGate, July 2015.

2. “Algorithm Visualization: The State” of the Field by Clifford A. Shaffer, Matthew L. Cooper, Alexander Joel D. Alon, Monika Akbar, Michael Stewart, Sean Ponce and Stephen H. Edwardsacm Transactions on Computing Education, Vol. 10, No. 3, Article 9, Pub. date: August 2010.

3. “Visualizing sorting algorithms” by Brian Faria, Rhode Island College, 2017.

4. Geeks for Geeks (https://www.geeksforgeeks.org)

5. Google (https://www.google.co.in)

6. Tutorials Points (https://www.tutorialspoint.com/)

7. Open AI (https://openai.com/)

8. Stack Overflow (https://stackoverflow.com)