“ALGORITHM VISUALIZER FOR AOA”

Submitted in partial fulfillment of the requirements of the degree

BACHELOR OF ENGINEERING
IN
COMPUTER ENGINEERING

By

Pratiksha P. Kirolkar (201042001)

Minal R. More (201042005)

Abhishek B. Pawar (201042007)

Rahul R. Chaurasia (201042016)

Name of the Mentor

Prof. A. R. Sonule

Department of Computer Engineering
A.C. Patil College of Engineering Kharghar,
Navi Mumbai
University of Mumbai

(AY 2021-22)

CERTIFICATE

This is to certify that the Mini Project entitled **"Algorithm Visualizer for AOA"** is a bonafide work of **Pratiksha Kirolkar (201042001), Minal More (201042005), Abhishek Pawar (201042007), Rahul Chaurasia (201042016),** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **"Bachelor of Engineering"** in **"Computer Engineering".**

(Prof. A.R. Sonule)
Mentor

(Dr. M. M. Deshpande)
Head of Department

(Dr. V.N. Pawar)
Principal

Mini Project Approval

This Mini Project entitled **"Algorithm Visualizer for AOA"** by **Pratiksha Kirolkar (201042001), Minal More (201042005), Abhishek Pawar (201042007), Rahul Chaurasia (201042016),** is approved for the degree of **Bachelor of Engineering** in **Computer Engineering.**

Examiners

1……
(Internal Examiner Name & Sign)

2……
(External Examiner name & Sign)

Date:

Place:

Contents

ABSTRACT

This paper discusses a study performed on website for algorithms visualize for AOA as a learning aid for classroom instruction. A web-based project was created to visualize many common sorting algorithms: for example Selection Sort, Bubble Sort, Insertion Sort, and Merge Sort,. The website would represent data as a bar-graph and after selecting a data-ordering and algorithm, the user can run an automated animation or step through it at their own pace. Afterwards, a study was conducted with a voluntary student population at A.C. Patil college who were in the process of learning algorithms in their Computer engineering curriculum.

The study consisted of a demonstration and survey that asked the students questions that may show improvement when understanding algorithms. The results and responses are recorded and analyzed in this paper with respect to previous studies.

Algorithm visualization illustrates how algorithms work in a graphical way. It mainly aims to simplify and deepen the understanding of algorithms operation. Within the paper we discuss the possibility of enriching the standard methods of teaching algorithms, with the algorithm visualizations.,

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to several individuals and organizations for supporting us throughout the project.

First, we wish to express our sincere gratitude to our supervisor Prof. A. R. Sonule for his enthusiasm, patience, insightful comments, helpful information, practical advice and unceasing ideas that have always helped us tremendously in our research and writing of this thesis.

The Immense knowledge, profound experience, and professional expertise in quality control has enabled us to complete this project successfully. Without his support and guidance, this project would not have been possible.

Thanks for all your encouragement!

List of Figures

List of Tables:

1.Introduction

1.1 Introduction

Algorithms are the important concept in programming. You can create a website that shows the people how does the actual algorithm work. For Ex: Like array. They can input and it will show them how exactly the numbers getting interchange and how it is sorting the array.

Algorithm visualizer for AOA as an essential part of knowledge in a framework of computer engineering. They have their stable position in every computer professional. A programmer should have the basic knowledge from the area. Our scope here is the higher education in the field of computer engineering. So, within the paper, we discuss the extension of standard methods of teaching algorithms, using the whiteboard or slides, with the algorithm visualizations. According to studies they can be used to attract students 'attention during the lecture, explain concepts in visual terms, encourage a practical learning process, and facilitate better communication between students and instructors. Interactive algorithm visualizations allow students to experiment and explore the ideas with respect to their individual needs. Extensive studies on algorithm visualization effectiveness are available nowadays, and results are quite encouraging.

In addition to the mathematical and empirical analysis of algorithms, there is yet a third way to study algorithms. It is called algorithm visualization and can be defined as the use of images to convey some useful information about algorithms. That information can be a visual illustration of an algorithm's operation, of its performance on different kinds of inputs, or of its execution speed versus that of other algorithms for the same problem. To accomplish this goal, an algorithm visualization uses graphic elements—points, line segments, two- or three-dimensional bars, and so on—to represent some "interesting events" in the algorithm's operation.

The website of algorithm visualization to education seeks to help students learning algorithms. The available evidence of its effectiveness is decisively mixed. Although some experiments did register positive learning outcomes, others failed to do so. The increasing body of evidence indicates that creating sophisticated software systems is not going to be enough. In fact, it appears that the level of student involvement with visualization might be more important than specific features of visualization software. In some experiments, low-tech visualizations prepared by students were more effective than passive exposure to sophisticated software systems.

1.2 Motivation

The motivation behind the decision is detailed and it includes the fact, that the platform is intended to be used as a support tool within the subject analysis of algorithm and algorithms visualization The selection of topics within the scope of the subject is quite wide and it could probably be changed over the time. To cover the scope of the subject, probably more tools would be used, or quite big interventions to selected tool would be required. we believed it was better to start designing and developing in our project. There are some specific issues of analysis and design of algorithm visualization systems, as it was described in Within the section 5 of our paper, we tried to give answers at least to most important questions from the user analysis, needs analysis, task analysis, information analysis and domain analysis point of view.

While I was learning sorting algorithms, I was wondering how each sorting algorithm would look. It was interesting to see how each algorithm looks, and it gave me a better understanding of sorting algorithms. For instance, if a sorting algorithm uses the divide and conquers approach, you can see the pattern of self-imitation.

1.3 Problem Statement & Objectives

An algorithm is a method or a process followed to solve a problem. If the problem is viewed as a function, then an algorithm is an implementation for the function that transforms an input to the corresponding output. A problem can be solved by many different algorithms. A given algorithm solves only one problem (i.e., computes a particular function). OpenDSA modules cover many problems, and for several of these problems we will see more than one algorithm. For the important problem of sorting there are over a dozen commonly known algorithms. It must be correct. In other words, it must compute the desired function, converting each input to the correct output. Note that every algorithm implements some function, because every algorithm maps every input to some output (even if that output is a program crash). At issue here is whether a given algorithm implements the intended function.

Objectives:

- The main objective of this project is to help beginners to be able to visualize the basic algorithms and get a better understanding of the underlying operations.
- And obviously it is needless to say that anyone who is willing to contribute is invited to use their creativity in making the visualizations even better and attractive.
- One can add fresh Algorithms and visualization of their choice too.
- Keeping the user interface clean and simple so the students can focus solely on the visualizations.

1.4 Organization of the Report

Chapter 1

This chapter include initiation of this project where we will analyze problem, project motivation and objectives.

This will be categorized as follows:

1. Introduction
2. Motivation
3. Problem statement
4. Objectives

Chapter 2

This chapter is all about the analysis of existing systems, the problems and flaws of this system, the reference we used to make this project and further reading.

This will be further categorized as:

1. Survey of existing system
2. Limitation of existing system
3. Mini project contribution

Chapter 3

Here, we analyze the architecture and the implementation of our software, it also looks at the possible obstacles and the actual obstacles faced along with our test runs and conclusion.

This will be further categorized into:

1. Introduction
2. Architecture/ Framework
3. Algorithm of application
4. Details on software and hardware used
5. Test criteria
6. Test results
7. Conclusion
8. Further work

2.Literature Survey

The proposed system is organized into several sub sections. The objectives of the proposing the algorithm visualizer for aoa mini project is to be justified by the simplicity of the algorithm. This will allow us to focus on more advanced features and topic like multiusers functionality and interactive learning.

2.1 Survey of Existing System/SRS

The previous work of this already exists. In existing system, they created the web application for only sorting array.

- Over the last decade, several algorithm visualizations tools have been developed and it is still receiving increased interest from both educators and researchers.
- In this project, there is a limitation of user input, which user can want data for operations.
- This project does not tell us more about searching and graph finding algorithms.
- Although there was a great explanation of algorithm, in this paper there was not more focus on user interface where we can visualize.

Sr. No	Title of Review Paper	Year	Review
1	Visualization Tools of Data Structures Algorithms	2014	This study provides better understanding of visualization tool and provides knowledge to the beginners who want to work in visualization tools.
2	Algorithm Visualizations as a Way of Increasing the Quality in Computer Science Education	2016	The group of students using algorithm visualizations within their exercises, could achieve better results in succeeding test than another group, not using them.
3	Visualizing Sequence of Algorithms for Searching and Sorting	2009	In searching algorithms, the user has to give the total numbers of inputs, the set of data and the element to be searched. Then select the particular algorithm from the list and then the visualization of the selected algorithm is shown with the given inputs.

Table 1: Literature Survey

2.2 Limitation Existing System or Research Gap

Therefore, according to the algorithm visualize for aoa is the following research gaps are then summarized:

1. Principles or Guidelines of Algorithm Visualizer for aoa particularly settled on mobile platform are highly rare.
2. Most of the previous studies of Algorithm Visualizer for aoa are much focusing on Desktop and Web Platform.
3. Even though, there are two Algorithm Visualizer for aoa on mobile platform, but they just solely focus on sorting algorithm and also the interactivity level of them are still low.
4. Some design elements and guidelines that should be considered when developing the Algorithm Visualizer for aoa website on mobile platform are not clearly identified in the existing guidelines.
5. The two existing mobile AV study still focus on one mobile operating system only, which is Android

2.3 Mini Project Contribution

Our project is a small attempt for developing an Algorithm visualizer for AOA. Firstly, we generated set of algorithms that we want to implement in this project with the help of many existing system and their limitations. While keeping the design simple and not including too many details a student can easily understand the working of algorithm without having to give their attention elsewhere. It gives the complete satisfaction of knowing that the machine doesn't affect the success of Project. Project will help to enhance the skills.

3.Proposed System (e.g., New Approach of Data Summarization)

3.1 Introduction

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.

3.2 Architecture/ Framework

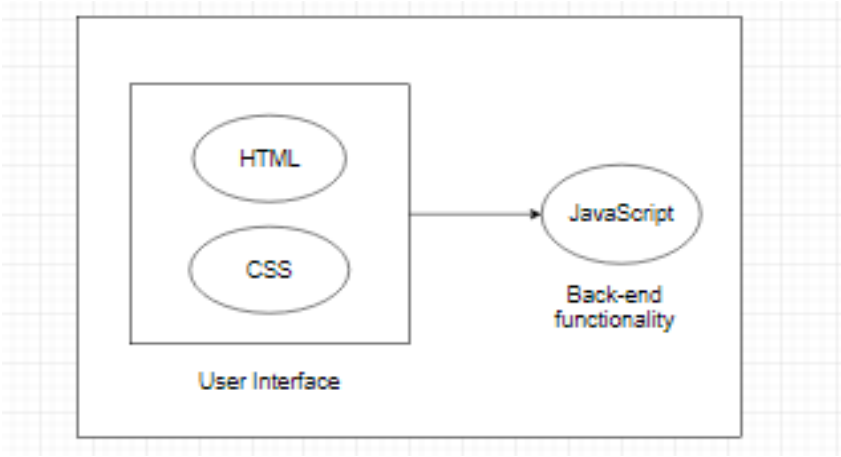

Figure 1: File Architecture Diagram

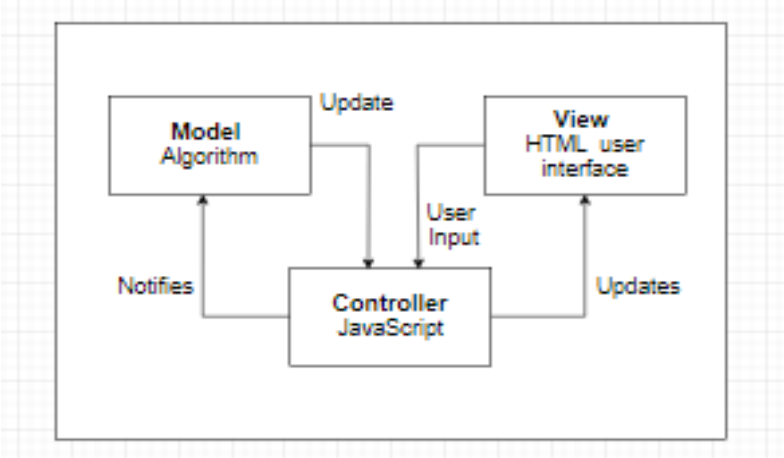

Figure 2: Model-View Controller

3.3 Algorithm and Process Design

Algorithm for Bubble Sort:

Step 1: Look at the first number in the list.
Step 2: Compare the current number with the next number.
Step 3: Is the next number smaller than the current number? If so, swap the two numbers around. If not, do not swap.
Step 4: Move to the next number along in the list and make this the current number.
Step 5: Repeat from step 2 until the last number in the list has been reached.
Step 6: If any numbers were swapped, repeat again from step 1.
Step 7: If the end of the list is reached without any swaps being made, then the list is ordered, and the algorithm can stop.

Algorithm for Selection Sort:

Step 1: Select the first element of the list (i.e., Element at first position in the list).
Step 2: Compare the selected element with all the other elements in the list.
Step 3: In every comparison, if any element is found smaller than the selected element (for Ascending order), then both are swapped.
Step 4: Repeat the same procedure with element in the next position in the list till the entire list is sorted.

Algorithm for Insertion Sort:

Step 1: If it is the first element, it is already sorted. return 1;
Step 2: Pick next element
Step 3: Compare with all elements in the sorted sub-list
Step 4: Shift all the elements in the sorted sub-list that is greater than the value to be sorted
Step 5: Insert the value
Step 6: Repeat until list is sorted

Algorithm for Merge Sort:

Step 1: if it is only one element in the list it is already sorted, return.
Step 2: divide the list recursively into two halves until it can no more be divided.
Step 3: merge the smaller lists into new list in sorted order.

Algorithm for Quick Sort:

Step 1 - Consider the first element of the list as pivot (i.e., Element at first position in the list).
Step 2 - Define two variables i and j. Set i and j to first and last elements of the list respectively.
Step 3 - Increment i until list[i] > pivot then stop.
Step 4 - Decrement j until list[j] < pivot then stop.
Step 5 - If i < j then exchange list[i] and list[j].
Step 6 - Repeat steps 3,4 & 5 until i > j.
Step 7 - Exchange the pivot element with list[j] element.

Algorithm for Radix Sort:

Step 1: Finding the maximum element
Step 2: Count the number of digits of the maximum number
Step 3: Arrange the numbers on the basis of the least significant digit
Step 4: Arrange the numbers according to the next significant digit
Step 5: Keep performing the process until the most significant digit

Algorithm for Breadth First Search:

Step 1: SET STATUS = 1 (ready state) for each node in G
Step 2: Enqueue the starting node A and set its STATUS = 2(waiting state)
Step 3: Repeat Steps 4 and 5 until QUEUE is empty
Step 4: Dequeue a node N. Process it and set its STATUS = 3(processed state).
Step 5: Enqueue all the neighbors of N that are in the ready state (whose STATUS = 1) and set their STATUS = 2 (waiting state)
[END OF LOOP]
Step 6: EXIT

Algorithm for Depth First Search:

Step 1: SET STATUS = 1 (ready state) for each node in G
Step 2: Push the starting node A on the stack and set its STATUS = 2 (waiting state)
Step 3: Repeat Steps 4 and 5 until STACK is empty
Step 4: Pop the top node N. Process it and set its STATUS = 3 (processed state)
Step 5: Push on the stack all the neighbors of N that are in the ready state (whose STATUS = 1) and set their STATUS = 2 (waiting state)
[END OF LOOP]
Step 6: EXIT

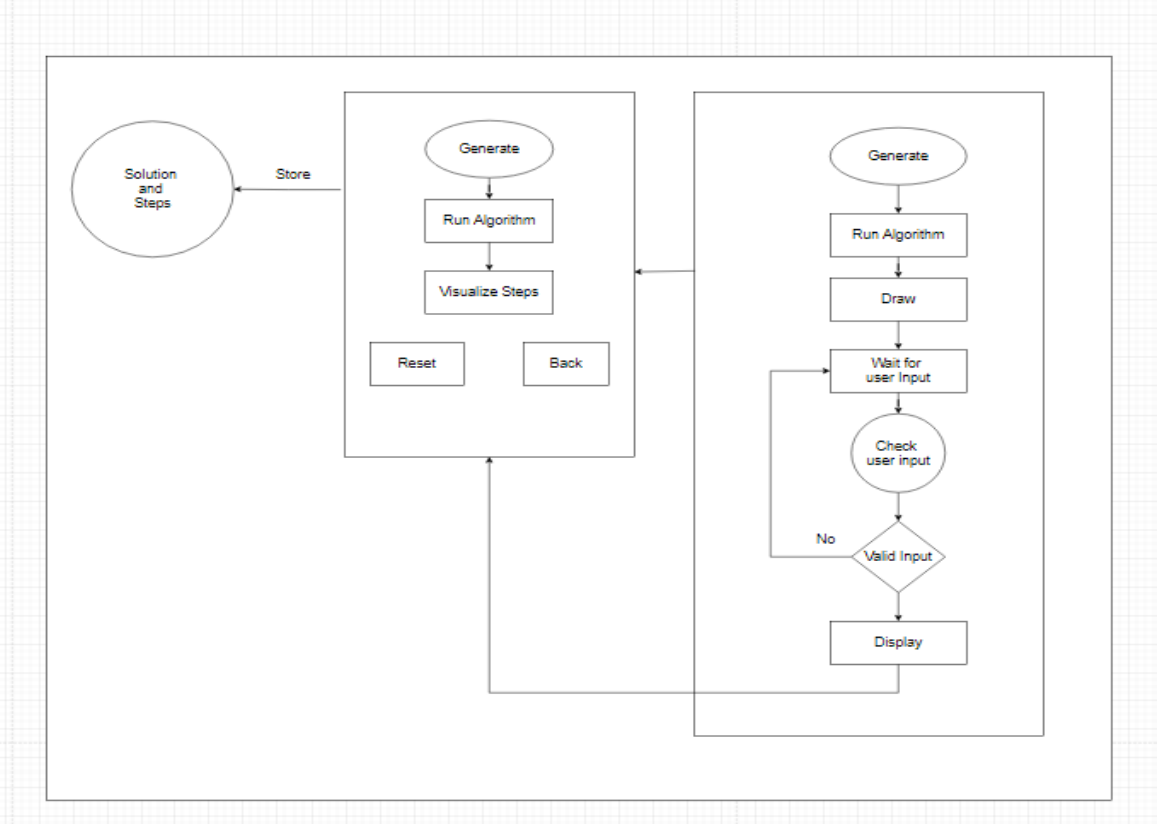

Figure 3: Process Flow Diagram

3.4 Details of Hardware & Software

Software Requirements:

Operating System	- Windows XP, Windows 7, Windows 8/8.1/10, Linux
Language	- HTML, CSS, JavaScript.
Browser	- Mozilla/Chrome (any will do).
Software development kit	- Visual Studio code, Live Browser extension.

Hardware Requirements:

Processor	- Intel core Pentium III 630MHz
RAM	- 4 GB
Hard disk	- 500 GB

3.5 Test Cases (Procedure and Expected Results)

Test case ID	Test Case Objective	Step ID	Test Data	Expected Result	Actual Result	Status
TC 01	To test bubble sort functionality	1)	Click on bubble sort	Bubble sort page should be opened	Bubble sort page opened	Pass
TC 02	To tests bubble sort functionality	1) 2) 3)	Choose array size Select array layout(bar) Enter values	Values should be sorted in bar layout	Values sorted in bar layout	Pass
TC 03	To test bubble sort functionality	1) 2) 3)	Choose array size Select array layout (Array) Enter values	Values should be sorted in array layout	Values sorted in array layout	Pass
TC 04	To test values functionality	1) 2) 3)	Choose array size (5) Select array layout (Array) Enter values= 1,22, 10,15,9,18	Message Should display “invalid data”	Message displayed as “Choose five integers between 0 to 999”	Pass
TC 05	To test reset functionality	1)	Click on reset	Data should be cleared/reset	Data cleared	Pass
TC 06	To test steps	1) 2) 3)	Choose array size Select array layout (Array) Enter values	Steps should change	Steps changed step-by-step	Pass

Table 2: Test Cases

3.6 Code Testing and Result

Testing objectives:

Unit testing is the testing of the individual components (units) of the software. Unit testing is conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

When developing a strategy for unit testing, there are three basic organizational approaches that can be taken. These are top down, bottom up and isolation.
In our case of an Algorithm visualizer for AOA, we simply use a isolation approach.

Here are the essential software testing steps every software engineer should perform before showing their work to someone else.

1. Basic functionality testing

Begin by making sure that every button on every screen works. Also need to ensure that you can enter simple text into each field without crashing the software. If the feature is designed to be accessed by way of an API, you need to run tests to make sure that the basic API functionality works before submitting it for more intensive testing. If your basic functionality testing detects something that doesn't work, that's fine.

2. Code Review

Another pair of eyes looking at the source code can uncover a lot of problems. If your coding methodology requires peer review, perform this step before you hand the code over for testing. Remember to do your basic functionality testing before the code review, though.

3. Static Code Analysis

There are tools that can perform analysis on source code or bytecode without executing it. These static code analysis tools can look for many weaknesses in the source code, such as security vulnerabilities and potential concurrency issues. Use static code analysis tools to enforce coding standards, and configure those tools to run automatically as part of the build.

4. Unit Testing

We will write unit tests to make sure that the unit (be it a method, class, or component) is working as expected and test across a range of valid and invalid inputs. In a continuous integration environment, unit tests should run every time you commit a change to the source code repository, and you should run them on your development machine as well. Some teams have coverage goals for their unit tests and will fail a build if the unit tests aren't extensive enough.

Result:

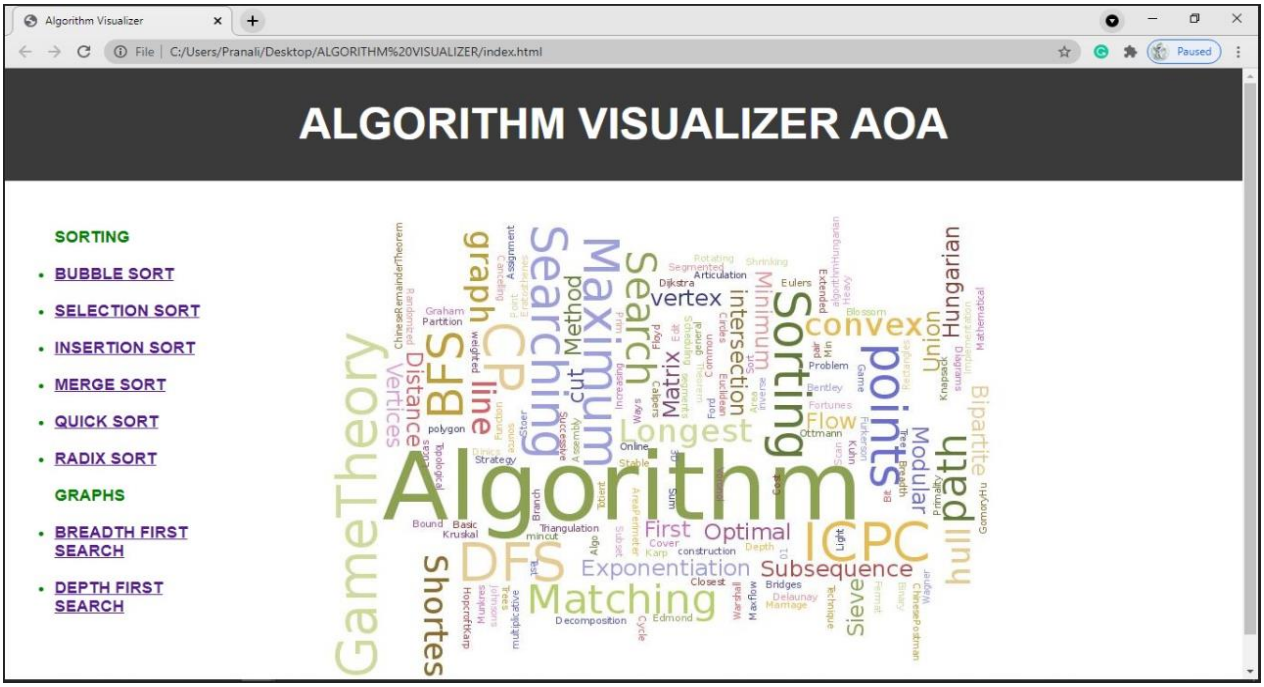

Figure 4: Homepage

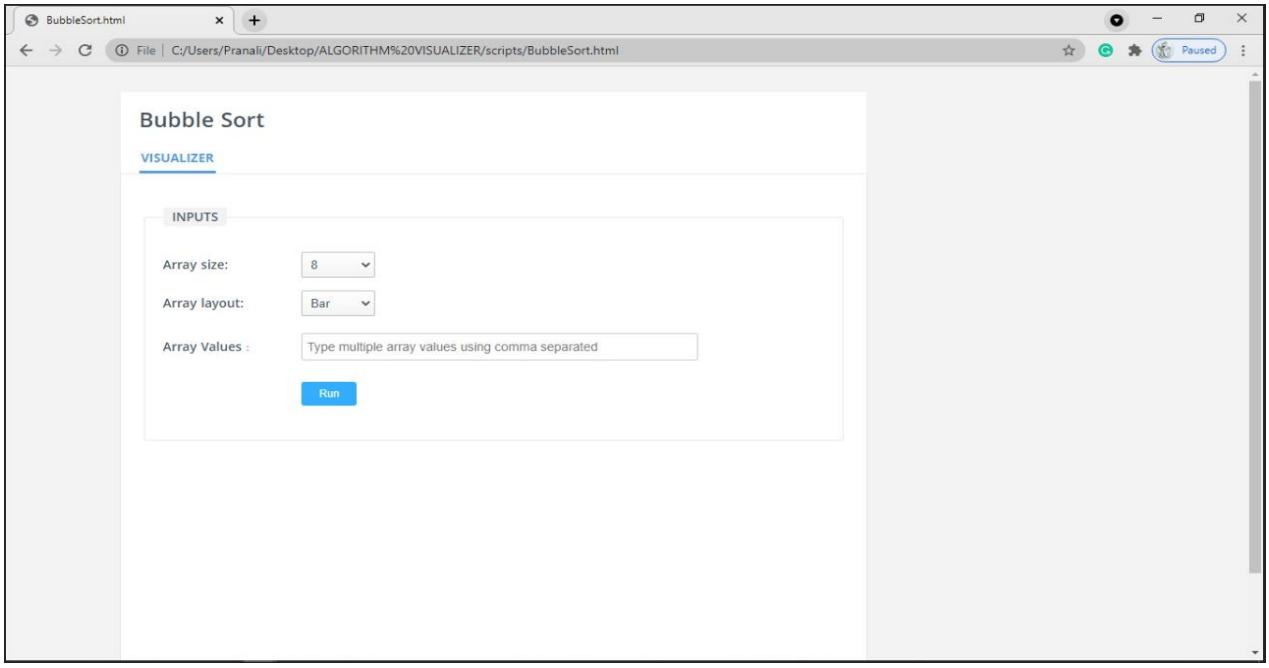

Figure 5: Sorting Page

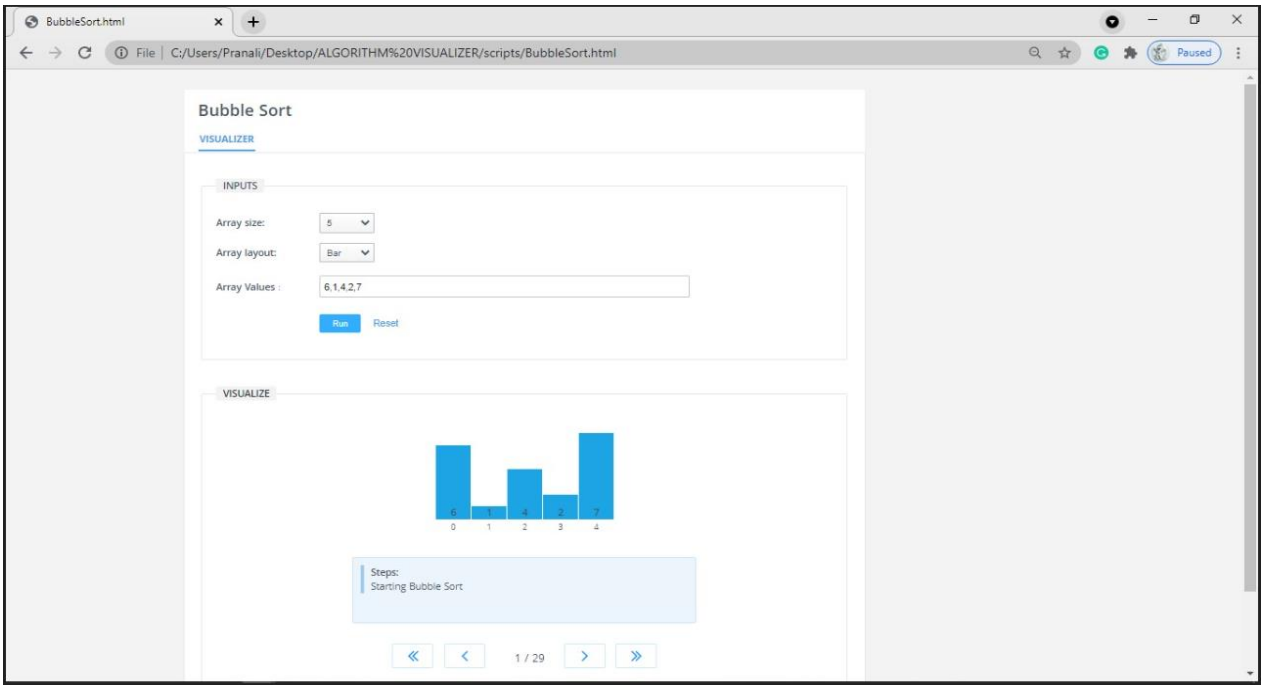

Figure 6: Inserting Values

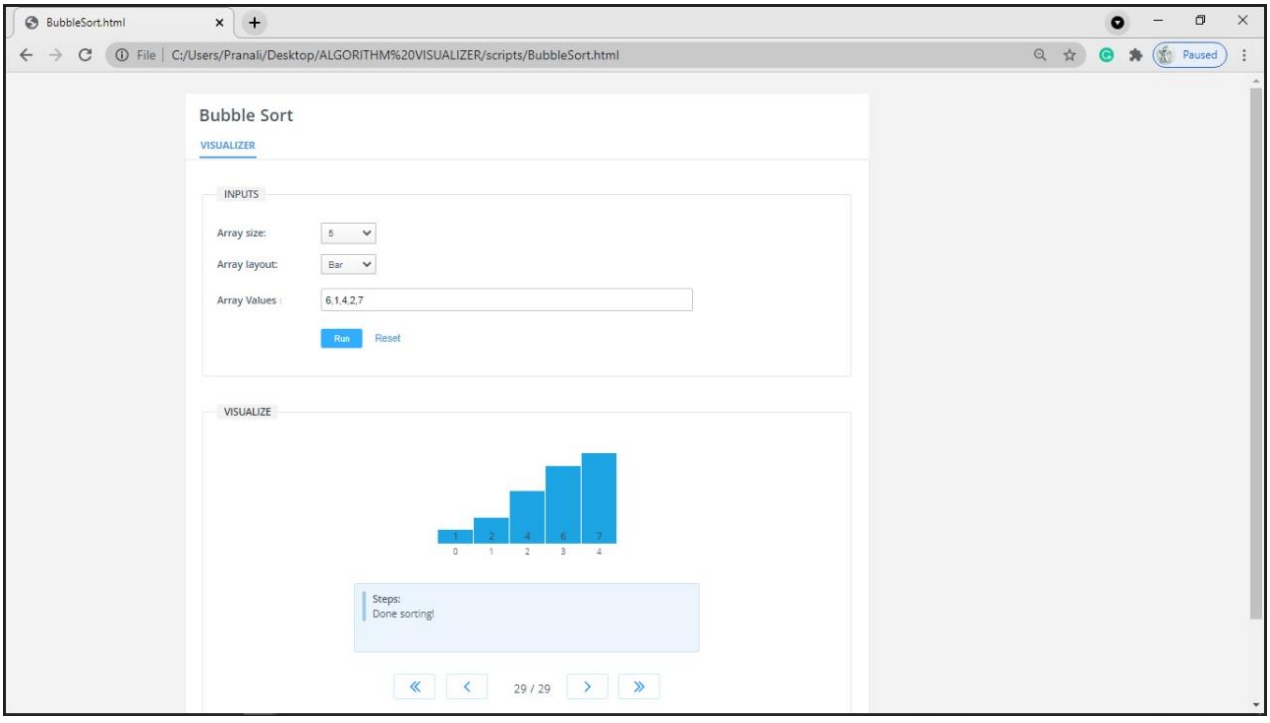

Figure 7: Finished Sorting (Bar Graph)

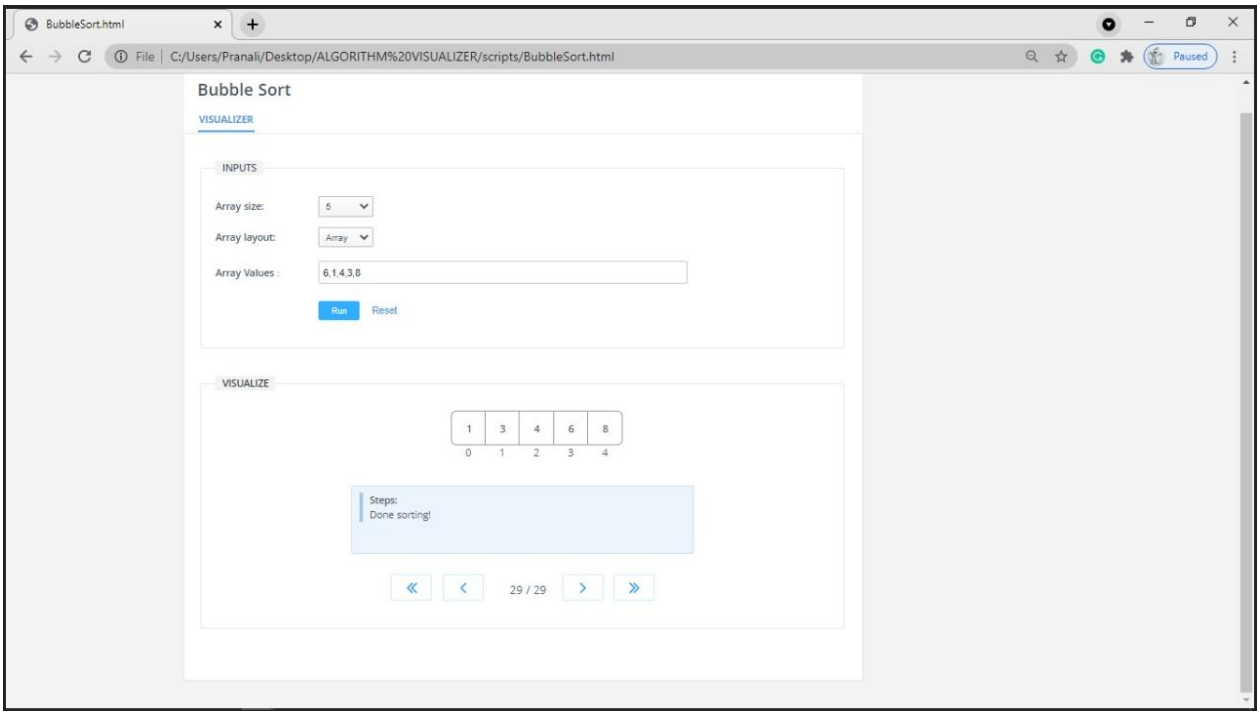

Figure 8: Finished Sorting (Array)

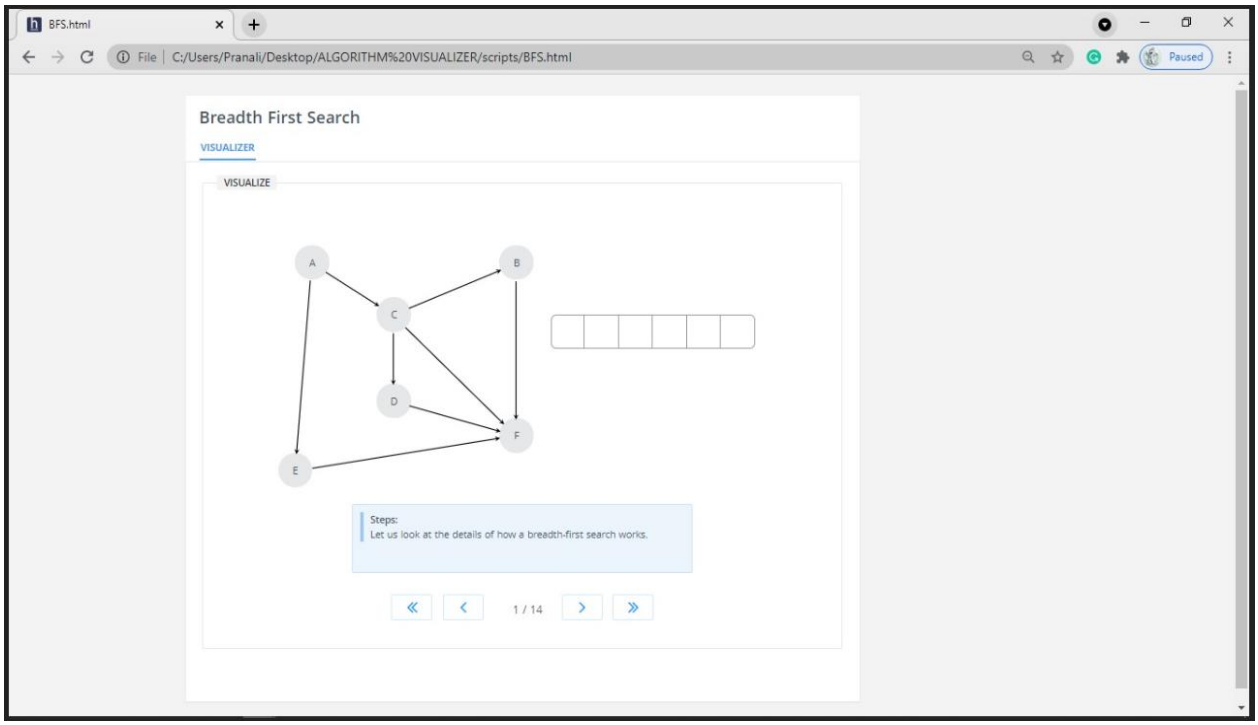

Figure 9: Graph Tree (Start)

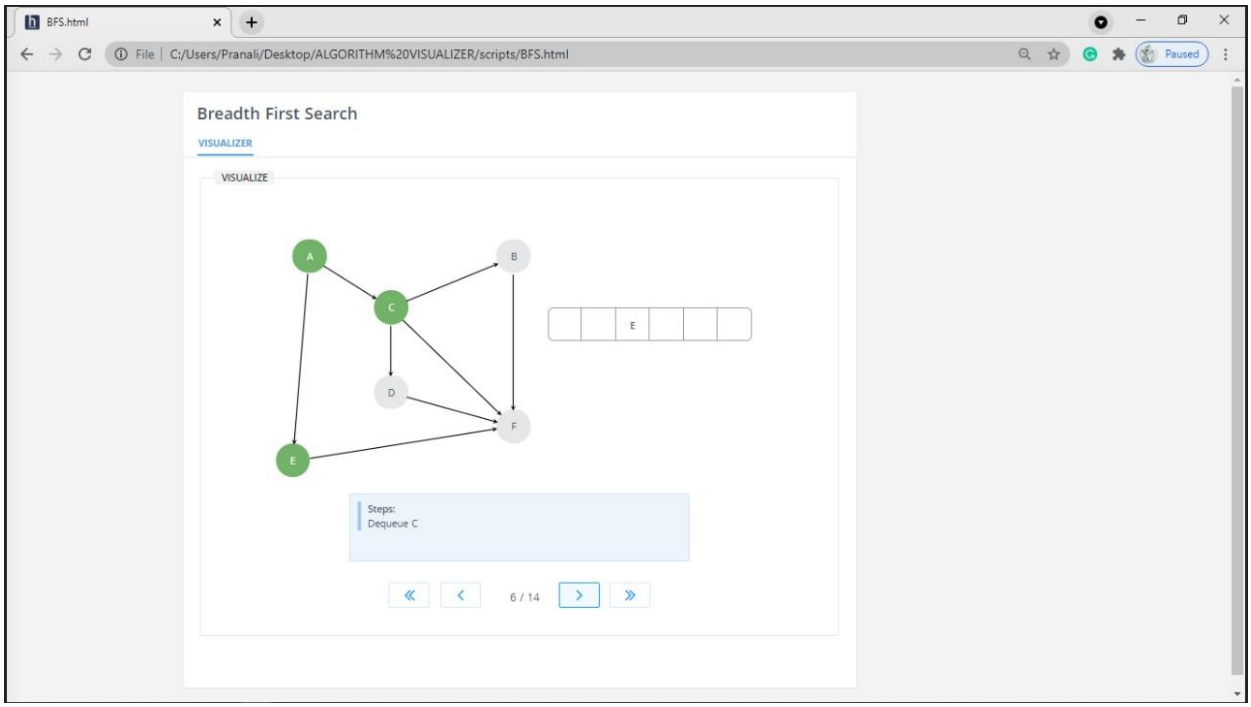

Figure 10: Graph Tree (Step)

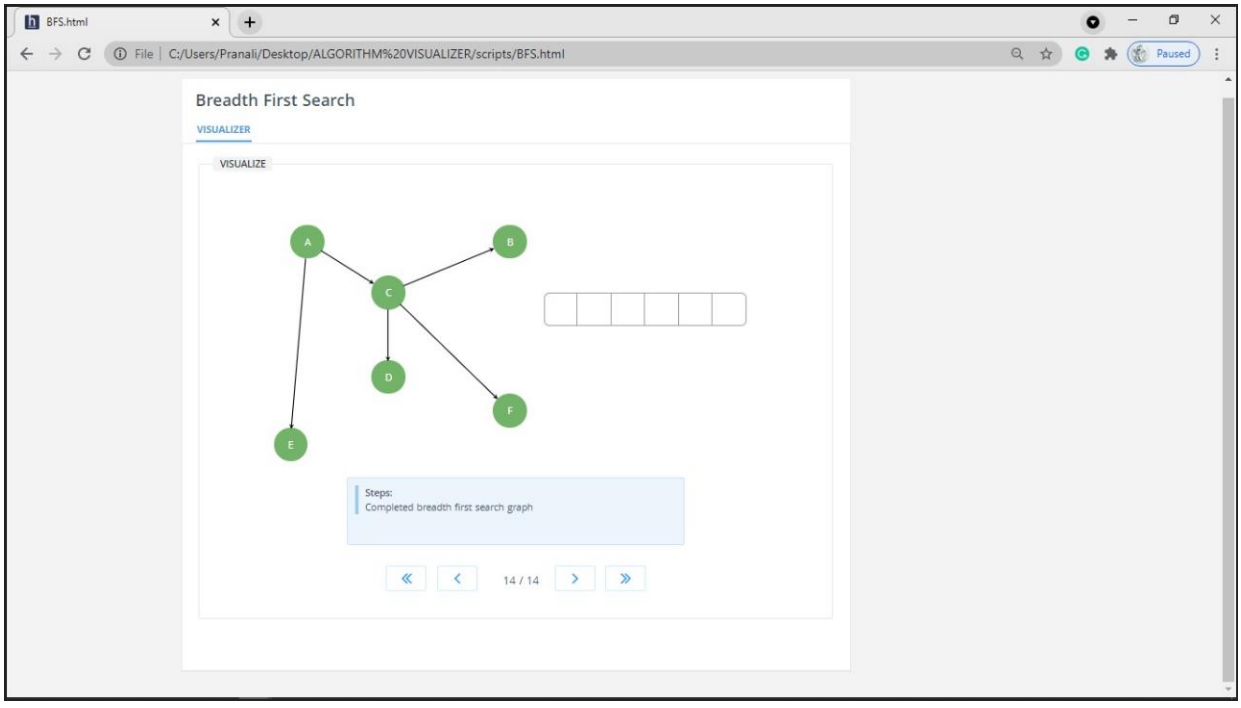

Figure 11: Graph Tree (End)

3.7 Conclusion and Future work

According to our findings, algorithm visualization can be seen as a valuable supporting tool, used in addition to standard ways of education in the field of computer science. The results seem to indicate that a simple visualization which focuses mainly on the procedural steps of the algorithm may help participants understand and notice the procedural steps in a better way. Using the algorithm visualizer, the learning curve and time spent learning and understanding should be significantly reduced. While more fundamental research on how to develop and use algorithm visualizations is necessary, we also simply need more implementors to produce more quality visualizations. And we need to encourage them to provide visualizations on under-represented topics.

Future Scope:

Project will be able to implement with better features in future after making some changes and Modifications.
Further implementations are as follows:
1) More Algorithms can be implemented
2) Problems and tutorials can be implemented
3) Simultaneous comparison of different algorithm

References

- algorithm-visualizer.org
- visualgo.net
- Geeksforgeeks.org
- sortvisualizer.com
- w3schools.com
- “Introduction to Algorithms” by Thomas