Algovis – An Enhanced Way to Visualize Algorithms

Rahul Lanjewar[1], *Nidhish* Mehra[1], *Aashka* Doshi[1,*] and *Yogita* Mane[2]

[1]Bachelor's of Engineering (Information Technology), Universal College of Engineering, Kaman, Vasai, India
[2]Head of Department (Information Technology), Universal College of Engineering, Kaman, Vasai, India

Abstract. It has been observed that learning using visual aids is less complicated than learning using traditional methods. As they say, "Words usually cannot do a picture's justice". Keeping the above fact in mind, we have come up with 'Algovis'. The aim of this project is to build a website that enables a user to visualize the steps and iterations involved in Data Structures and Algorithms using custom input and various animations. The program is menu driven to fit to the convenience of the user, depending on how many inputs he wants to enter, and implement the algorithm of his/her choice. The website also aims to become a convenient pedagogical tool. Algovis helps in visualizing various Data structure algorithms using graphical animations and with 100% accuracy.

1 Introduction

It is broadly seen that algorithm visualizations can give a strong option in contrast to static composed presentations (from course books), or verbal portrayals upheld by illustrations (from lectures). Algorithm Visualizations can improve the understanding of fundamental Data Structures and Algorithms.

Certainly, algorithm visualizations and animations have a long history in the field of computer science. Even so, most of these visualizers have been proven to have a bad user experience. Secondly, most of them have been least useful when it came to teaching, which means they give little to no information of how the algorithm being visualized works and so, they have no value in classrooms.

The purpose of this project will be to tackle these issues by extensive research, understanding and studying of various algorithms and developing a website using MERN stack that provide a superior client experience alongside a wide scope of elements. The venture will likewise plan to end up being a superior instrument for the teaching of Data Structures and Algorithms by utilizing custom visualizations and animations.

1.1 Motivation

We have observed and experienced as well that students struggle to understand the working and implementation of algorithms while learning them in a class. Even if they are successful in applying and running the algorithm practically, they might not necessarily understand what really goes on in the processing part when they are trying to run that algorithm. The dry runs can only be effective to a limit. Also, it is not possible to make dry runs for each iteration of every algorithm. Adding to it, due to the covid pandemic, almost the whole world shifted to online learning mode. Drawing and explaining the algorithms using dry run method was very difficult. Hence, to counter these problems to make learning easier and to encourage digital learning we have come up with this idea.

1.2 Problem statement

Currently, students learning Data Structures and Algorithms in various languages learn mostly through traditional methods like books, lectures, and innumerable dry runs. Through these methods they do understand what the algorithm does, but most of the students have problem imagining the implementation and working of these algorithms.

Secondly, even the traditional 'chalk and board' classrooms are being replaced by modern classrooms that use technological pedagogical aids. To add to the teacher's convenience, there should be a tool that explains by visualization of the processes that go on "behind the scenes" while implementing an algorithm. Though we realize that animations in the form of videos are available on various websites like YouTube, we can't have custom inputs and various use cases tested over there.

In an era where it's possible to experience a whole new virtual world through technology, there should be a way for students to learn with and teachers to teach with the use of these advanced technologies instead of using pen and paper.

2 Review of Literature

In paper [1], the authors have created a wiki of over 350 algorithm visualizers. They have analysed the quality, coverage, and effectiveness of these visualizers. The theoretical establishments for making compelling visualizations give off an impression of being consistently getting to the next level. More papers are seeming regarding successful utilization of visualization in courses, and a collection of work has started to shape with respect to how to foster compelling visualizations in any case. Collectively, the community is learning how to improve. *The disadvantage of this paper is that it is seen that most existing algorithm visualizations are of inferior quality, and the substance inclusion is slanted vigorously toward more straightforward points. There are no*

*Corresponding author: aashkadoshi7@gmail.com

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (http://creativecommons.org/licenses/by/4.0/).

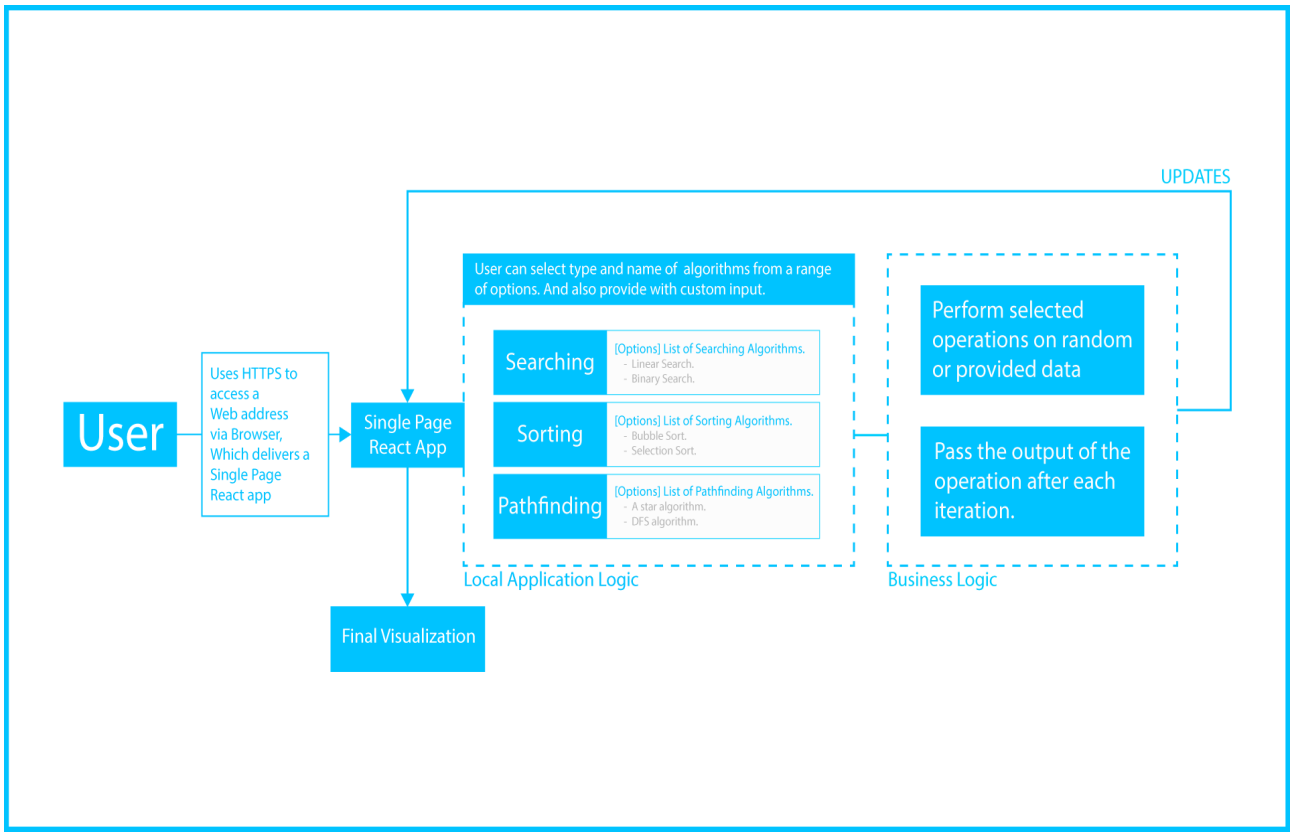

Fig.1. Algovis: A System Architecture

powerful repositories or coordinated assortments of algorithm visualizations as of now accessible.

Algorithm visualization uses computer graphics or interactive media to show the steps of an algorithm. In this research, the algorithm visualization is utilized to assist understudies with understanding the idea of selection sort algorithm and creating coding to picture it. The objective of the paper [2] is to create visualization of Selection Sorting Algorithm. The visualization shows how all data move to the proper position to be sorted. It can be easily understood by students how the algorithm should be implemented while writing actual code. Students easily understand the concept of selection sort by looking at the visualization.

Learning becomes easier when graphic, animation, or video means are used by student. Adobe Flash is a timeline-based authoring and object-oriented programming tools that can be utilized to foster a logical visualization. Adobe Flash with ActionScript programming tools can be utilized to make logic and mathematics operation. *The disadvantage here is that there are no dynamic input options. The visualization does not show which step in the algorithm is being implemented. The design is not very interactive.*

The authors in this paper [3] have analysed different software visualizers and noted down why the development of software visualizations is difficult and what features are lacking in current technologies. The researchers have worked towards the development of generalized algorithm visualizers. *This paper does not show searching, sorting and path finding algorithms.*

Algovis visualizes searching, sorting and pathfinding algorithms with 100% accuracy and the animations are easy to understand. It is an open source webapp whose repository is available on GitHub "https://github.com/RahulLanjewar93/AlgoVisualizer" [4]. It enables the user to give dynamic input and shows which exact step is being implemented currently.

3 System Architecture

There are three modules in our system namely searching, sorting and path finding which can be seen in Fig. 1. The user can navigate between these modules through the navbar at the top.

The user can access our web app which is developed using react. With the help of react it is possible to update the DOM (Document object model) effectively. The user can select the algorithm they want to perform on the landing page as well as a navigation bar is provided for ease of access. After selecting the algorithm, the user will be redirected to a dedicated algorithm page which will display the steps and visualization of the algorithm.

In sorting, we have provided options on which type of sorting the user wants to perform. The user can use the random array generated by the app or enter a custom array. After clicking on the start button, the user will be able to see the steps of the algorithm and which step is currently being performed in real-time and the animated graph where user will be able to see how the sorting works. A new array will be formed after each iteration unless it's sorted.

In searching, the user either uses randomly generated array or input a custom array. Then chooses which type of searching must be performed. Then the user enters the number which must be searched and press the start

button. Then the user will be able to see the steps of the algorithm and which step is currently being performed in real-time and the animated graph where user will be able to see how the searching works. The index number on which the entered number is present will be given as a result. If the number is not there, then "Number not found" message will be shown.

In pathfinding, we show a grid having walls and nodes. There are various algorithms from which the user can choose. After selecting the algorithm, the user will be able to see the animation through which he will understand how the algorithm works.

4 Requirements

- Personal Computer/ Laptop.
- Internet Connection.
- Any browser.

5 Performance Analysis

5.1 Bubble Sort

Considering the size of the array and varying speed of users, the entire dry run might take on an average 5 minutes to complete. Apart from that, this method might not be 100% accurate if we keep human errors in mind.

The system that we have developed (Fig. 2) visualizes the bubble sorting of an array input of [41, 6, 43, 50, 23, 23, 57, 47, 32, 52, 22, 1, 20, 35, 15, 27, 36, 16, 44, 61] along with constantly highlighting the step which is currently being performed in approximately a minute. Along with that, the accuracy of the visualization is 100%. [5, 6]

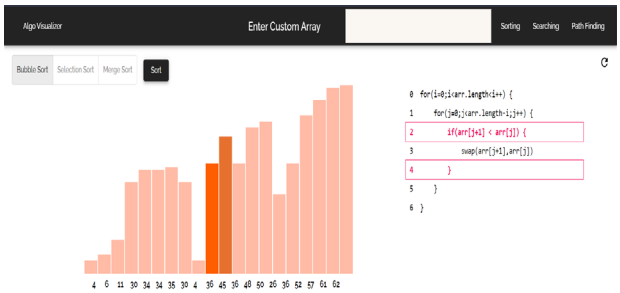

Fig.2. Bubble Sort Animation

Note the total time taken by the algorithm is directly dependent upon the animation speed + $O(n^2)$.

5.2 Selection Sort

Considering the size of the array and varying speed of users, the entire dry run might take on an average 10 minutes to complete. Apart from that, this method might not be 100% accurate if we keep human errors in mind.

The system that we have developed (Fig. 3) visualizes the selection sorting of an array input of [1, 3, 6, 63, 40, 55, 38, 25, 21, 19, 8, 52, 36, 35, 8, 13, 37, 34, 62, 7] along with constantly highlighting the step which is currently being performed in approximately two minutes. Along with that, the accuracy of the visualization is 100%. [5, 6]

Note the total time taken by the algorithm is directly dependent upon the animation speed + $O(n^2)$.

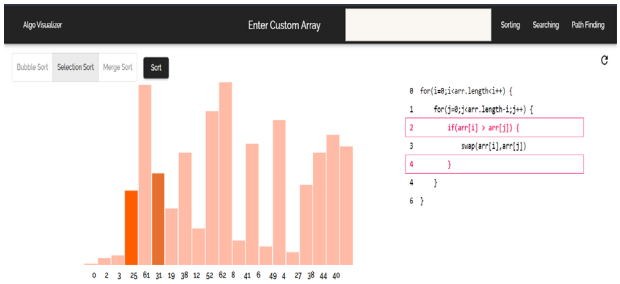

Fig.3. Selection Sort Animation

5.3 Merge Sort

Considering the size of the array and varying speed of users, the entire dry run might take on an average 12 minutes to complete. Apart from that, this method might not be 100% accurate if we keep human errors in mind.

The system that we have developed (Fig. 4) visualizes the merge sorting of an array input of [46, 44, 8, 32, 45, 25, 35, 62, 15, 2, 36, 33, 26,47, 20, 34, 27, 38, 33, 56] along with constantly highlighting the step which is currently being performed in approximately two minutes. Along with that, the accuracy of the visualization is 100%. [5, 6]

Note the total time taken by the algorithm is directly dependent upon the animation speed + O(nlogn).

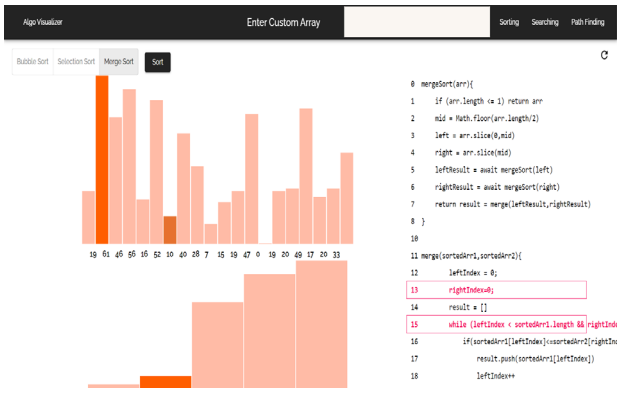

Fig.4. Merge Sort Animation

5.4 Linear Search

Considering the size of the array and varying speed of users, the entire dry run might take on an average 5

minutes to complete. Apart from that, this method might not be 100% accurate if we keep human errors in mind.

The system that we have developed (Fig. 5) visualizes the linear searching of an array input of [16, 35, 55, 37, 25, 27 ,46, 53, 36, 62, 37, 3, 57, 54, 51, 60, 54, 46, 47, 32] along with constantly highlighting the step which is currently being performed in approximately under a minute. Along with that, the accuracy of the visualization is 100%. [5, 6]

Note the total time taken by the algorithm is directly dependent upon the animation speed + O(n).

Fig.5. Linear Search Animation

5.5 Binary Search

Considering the size of the array and varying speed of users, the entire dry run might take on an average 10 minutes to complete. Apart from that, this method might not be 100% accurate if we keep human errors in mind.

The system that we have developed (Fig. 6) visualizes the binary searching of an array input of [0, 7, 8, 13, 15, 15, 16, 18, 18, 24, 29, 36, 37, 37, 47, 50, 51, 58, 58, 63] along with constantly highlighting the step which is currently being performed in approximately under a minute. Along with that, the accuracy of the visualization is 100%. [5, 6]

Note the total time taken by the algorithm is directly dependent upon the animation speed + O(logn).

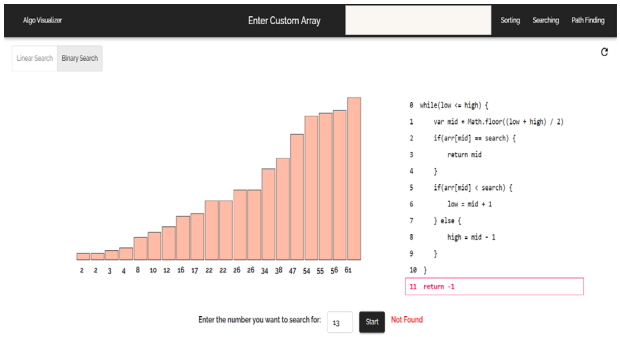

Fig.6. Binary Search Animation

5.6 A star path finding algorithm

Considering the number of nodes and varying speed of users, the entire dry run might take on an average 15 minutes to complete. Apart from that, this method might not be 100% accurate if we keep human errors in mind.

The system that we have developed (Fig. 7) visualizes the A star path finding while constantly highlighting the step which is currently being performed in approximately under a minute. Along with that, the accuracy of the visualization is 100%. [5, 6]

Note the total time taken by the algorithm is directly dependent upon the animation speed.

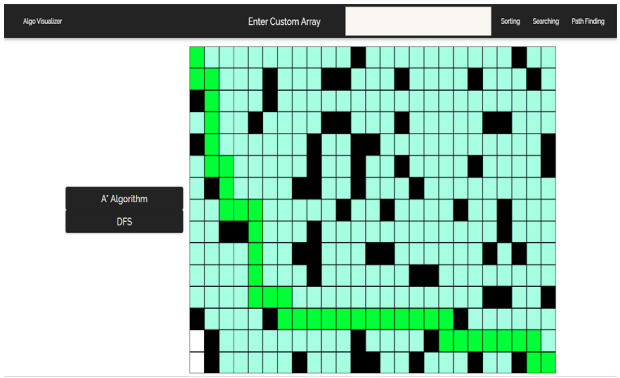

Fig.7. A star path finding Algorithm Animation

5.7 Depth first search algorithm

Considering the number of nodes and varying speed of users, the entire dry run might take on an average 12 minutes to complete. Apart from that, this method might not be 100% accurate if we keep human errors in mind.

The system that we have developed (Fig. 8) visualizes the Depth First Search algorithm while constantly highlighting the step which is currently being performed in approximately under a minute. Along with that, the accuracy of the visualization is 100%. [5, 6]

Note the total time taken by the algorithm is directly dependent upon the animation speed.

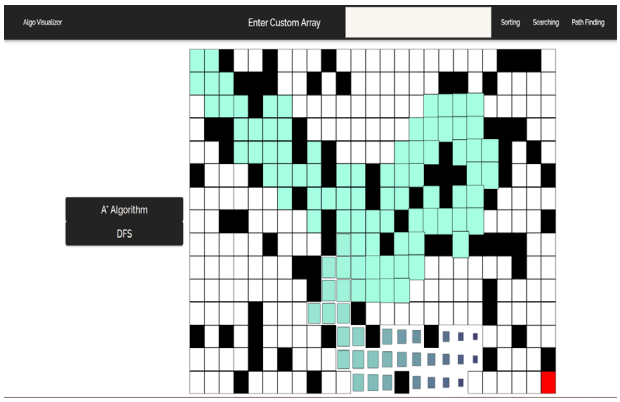

Fig.8. Depth First Search Algorithm Animation

Table 1 shows performance analysis of Algovis.

Algorithm	Time	Accuracy
Bubble Sort	Animation speed + $O(n^2)$	100%
Selection Sort	Animation speed + $O(n^2)$	100%
Merge Sort	Animation speed + O(nlogn)	100%
Linear Search	Animation speed + O(n)	100%
Binary Search	Animation speed + O(logn)	100%
A* Algorithm	Animation speed + $O(b^d)$	100%
DFS	Animation speed + O(V)	100%

Table 1. Performance Analysis

6 Comparative Analysis

	algorithm - visualizer [7]	Visualgo [8]	UFSCA [9]	Proposed work
Good UI	No	No	Yes	Yes
Custom Input	No	No	No	Yes
Real-time algorithm steps	Yes	Yes	No	Yes

Table 2. Comparative Analysis

Table 2 gives brief idea about comparisons of different tools available online.

7 Conclusion and Future Plans

Our future plan is to add more data structures' visualizations. Along with that we aim to improve the UI/UX of the website.

Apart from that we will add various features that will ease the work of students as well as teachers. Also, we aim to remove as many bugs as possible.

In conclusion, Algovis is a free, open-source pedagogical tool that enables the teachers to teach data structures conveniently and the learners to learn data structures easily while promoting remote and digital learning.

References

1. C. Shaffer, M. Cooper, S. Edwards,Algorithm visualization: a report on the state of the field, SIGCSE Bull. 39, 1 (March 2007), 150-154 (2007)
2. H. Sutopo, selection sorting algorithm visualization using flash, Int. J. of Multimedia & it's Applications, vol. 3, no. 1, (February 2011)
3. V. Karavirta, Effortless creation of Algorithm Visualization (2002)
4. "Source Code", Accessed: 12th October 2021 [Online], Available: https://github.com/RahulLanjewar93/AlgoVisualize
5. "Algorithms", Accessed: 18th October 2021 [Online], Available: https://www.geeksforgeeks.org/fundamentals-of algorithms/?ref=shm
6. "Algorithms", Accessed: 18th October 2021 [Online], Available: https://www.tutorialspoint.com/data_structures_algorithms/index.htm
7. "Algorithm-Visualizer", Accessed: 27th February 2022 [Online], Available: https://algorithm visualizer.org/
8. "Algorithm-Visualizer", Accessed: 27th February 2022 [Online], Available: https://algorithm visualizer.org/
9. "Algorithm-Visualizer", Accessed: 27th February 2022 [Online], Available: https://algorithm visualizer.org/
10. "USFCA", Accessed: 27th February 2022 [Online], Available: https://www.cs.usfca.edu/~galles/visualization/Algorithms.htm
11. "DSA", Accessed: 27th February 2022 [Online], Available: https://www.programiz.com/dsa/
12. "Learn DSA using Animations", Accessed: 27th February 2022 [Online], Available: https://www.youtube.com/playlist?list=PLmYemBEXGmVjAlx4K9JkyVTiFFayzDNwi