International Journal of Research in Engineering, Science and Management
Volume 4, Issue 9, September 2021
https://www.ijresm.com | ISSN (Online): 2581-5792

Web-Based Dynamic Algorithm Visualizer Along with Chat Engine SDK for Neophytes

Arijit Goswami[1*], Biswarup Bhattacharjee[2], Rahul Debnath[3], Ankita Sikder[4]
[1,2,3,4]*Student, Department of Computer Science, University of Engineering and Management, Kolkata, India*

***Abstract*: Algorithm Visualization is an online interactive platform that displays algorithms from code. Learning the algorithm is much easier by visualizing it. Algorithm Visualizer is a web application written in React. It contains UI elements and translates commands into visualizations. Algorithms are an attractive use case for visualization. To make an algorithm visualize, we do not simply measure the data in a chart; no primary database. Instead, there are reasonable rules that govern behavior. This may be why visualization of the algorithm is unusual, as designers are trying to create novel forms for better communication. This is a good reason to read them. The main purpose of this is to ensure effective and reliable methods of detecting various algorithms. Using this web application anyone can learn algorithms quickly and easily. Such applications are already available, but the efficiency of the available algorithm visualizers is not achieved thoroughly. This newly developed application proposes to take a step further and visualize all types of algorithms along with a customized chat engine SDK that enhances visual accuracy on a much larger scale.**

***Keywords*: Algorithms, visualization environment, analysis of complexities, web-based learning.**

1. Introduction

This is a web application used to visualize algorithms. This application is created using JavaScript, CSS, SCSS, ReactJS, HTML5, Vercel, Firebase, Chat Engine XDK, nodejs and Formik. [1] The user can access this website with the link provided. First, there is the registration or login area. [2] When the user opens it for the first time he must select the sign-up option. Here he must provide a username, email id, password, password confirmation. If they are already registered, then they must log in and select an email, password, and verify the password. Then enter the main page or home page. Here the user can see his email address and the password provided by firebase.

Here four buttons are provided. VISUAL ALGORITHMS, ALGORITHMS, DISCUSSION, ANSWER. [3] By clicking on VISUALIZE ALGORITHMS the user can view and understand the functionality of certain algorithms. When a user goes to ALGORITHMS they can see 72 complete algorithms. Filtering, Searching, and Other Algorithms. [4] When he clicks on any of them a button to learn more, details will be displayed on the page. The DISCUSSION section will take the user to the login page where they must provide a username and password (password provided with firebase on the home page). [5] It will then be included in the discussion group for hesitation to clear where the developers will answer his or her questions regarding this website. [6] In the feedback section, the user can write how useful this website is or any idea for the development of this website in the comments section. In mathematics and computer science, an algorithm is a limited sequence of computer-generated commands, commonly used on a computer, usually to solve a problem category or to perform calculations.

[7] Algorithms remain obscure and are used as descriptions of mathematical operations, data processing, automated thinking, and other functions. [8] As an effective method, the algorithm can be displayed within a limited amount of space and time, and in a well-defined official language for calculating the task. Starting from the initial state and initial installation (perhaps empty), the commands define a computer calculation that, when done, passes a limited number of well-defined consecutive countries, eventually producing "output" and ending in the end. Here we can see different types of algorithms and their details. The DISCUSSION section will take the user to the login page where they must provide a username and password (password provided with firebase on the home page). [9] The user will then be included in the discussion group for doubt clearing where the developers will answer his or her questions regarding this website. The conversation is essential for any user queries. When developers guide the user properly, it will be very helpful for the user to check the website. [10] Website feedback information is obtained directly from website users - through on-page surveys, response widgets, and other strategies - to help organizations understand what people think (and feel) about their websites and landing pages. This entire web application makes it a good practice of visualization-based algorithms study.

2. Background of the Work

Planning builds up the basic data structure in the programming curriculum, only a small amount of research focused on investigating student mental models and the difficulty of arranging arrays. [11] According to a survey, loops and topic layouts are two of the three main program topics for novice students. Du Boulay reported student confusion between the same array index and its cell; they have difficulty dealing

*Corresponding author: goswamiarijit2017@gmail.com

with arrangements that contain indices as planned. [12] An unpublished authorship survey of Greek high schools, using a sample of 102 students (K-12), confirmed similar misconceptions and reading difficulties; most students had erroneous or incomplete models of the concept of the same members which led to erroneous assumptions and great difficulty in solving simple algorithmic problems that required the use of a data structure of the same members. [13] In general, there are two main categories of recognition systems in CS education: the recognition system and the algorithm recognition systems. Program Visualization (PV) systems generate specific presentations of program structures and program design components, for example, Jeliot 3 is a well-known PV program that visualizes Java programs; other modern programs, such as Jype, Historic, and Online Python Tutor, Visualization programs in Python. [14] However, the concept behind the algorithm cannot be revealed simply by showing how the flexibility values of the system change. They aim to cover this need by visualizing abstract concepts and expressing the basic concepts of the underlying algorithm, thus helping students to create multiple mental models, link building hierarchies, and creating problem-solving patterns to distinguish the level of communication and visual assessment of learners (e.g. Both conversion capabilities, input data and algorithm code, and various visual representations). [15] The first reference to the animation algorithm was a popular video titled 'Sorting Out Sorting' presented by R. Baecker in 1981 at the SIGGRAPH conference. This 30 min video demonstrates the functionality and performance of nine filter algorithms, using animations and audio comments. Since then, there have been a number of tools being developed as a result of research projects in algorithm visibility. [16] The most popular way to create algorithm images is to specify algorithm code with script commands that generate visibility. [17] The first language-based program was developed by John Stasko and his colleagues and belongs to a wide family of algorithm detection systems (Tango, Polka, Samba, and JSamba). The animation contains a file containing graphic commands corresponding to the key events of the algorithm under sight. [18] One family of programs, such as MatrixPro, Trakla2, and Ville, offers "Algorithm Simulation Exercises", in which the student has to manually perform a given algorithm, usually by dragging elements into new or targeted positions or by pressing buttons to create a specific task. [19] Ville is a new tool for this family; supports multiple programming languages including C ++ and Java. Its built-in editor supports query design and tests that are displayed as pop-up windows. [20] Another AV version of the novel is JHave which helps AV developers easily create animated slideshows. [21] Its specific feature is the 'stop and think' questions and definitions that can be seen at any time during animation, thereby promoting effective interaction with students and algorithm perception. JHave includes a large collection of algorithm recognition and has gained a great deal of interest in education. The latest animation program algorithm is Alvis Live! [22] It is a system development platform that supports the development and interactive presentation of algorithm recognition using the SALSA language. It includes features that support storytelling. [23] A lot of research has been done to investigate the educational value of algorithm detection systems. Overall, the results showed that simple animation or visualization algorithms had little effect on student learning, due to the low level of student engagement. [24] Hundhausen, Douglas, and Stasko conducted systematic tests on the effectiveness of AV systems. Their 24 published research metastudy, related to AV, concluded that a) the way students use visibility is more important than their own perceptions, and b) AV sites only work where students participate in the learning process.

3. Implementation

The first goal of this project visualizes various sorting and searching algorithms, and create a web application to visualize these algorithms. [25] This visualization portion's goal is to create two types of visualization, first is the pathfinding approach and second is sorting visualization approach, In this pathfinding approach's goal is to create simple javascript logic to visualize searching algorithms like Dijkstra algorithm, A* search algorithm, etc and sorting visualization's goal is to generate random arrays with random size and then create some simple javascript logic to sort these arrays, as a result, we can simply visualize sorting algorithms. This project contains three single-page applications. These are nothing but three functionalities of our main project. Another goal of this is to create a backend database in firebase to store the user's email and password, after the store it will fetch the user id which will be provided by firebase. After that pass this provided user-id as password into react-chat-engine SDK, then we will create three react single-page applications and host these applications through netlify. The second goal of our project is that store and show the details of many algorithms to learn easily, and the details are the time complexity of algorithms, and their recurrence relations, logical approach, etc. The third important goal of our project is to create a chat application. By this portion, users will discuss their doubts together. [26] The last portion's goal is basically to create collection contacts, feedbacks in the firestore database which is used as the backend and another goal of this portion is to create a frontend to send feedbacks and contact us portions using bootstrap and sass(syntactically awesome style sheets). After creating a total of three single-page applications and the visualization portion then the next goal is to merge all functionalities and create a multi-page responsive application using vercel development, create a home page of our project after connecting it to the firebase login signup page. This is the overall goal of the project.

At first, we create a project in firebase then this project is set to a web-based application. Basically, firebase is a Backend-as-a-Service (Baas). [27] It provides a variety of tools and services to develop applications. It is built on Google's infrastructure. Firebase is classified as a NoSQL database system, which stores data in documents such as JSON. In our project, we create a custom email password authentication system in firebase. After that we use react library in node js to create login and signup page, to create these forms pages we use the Formik library in

react-js. React makes applications fast, scalable, and simple. After that, we create the main page of our project On this page we use the CSS blob effect using SVG. It literally means Scalable Vector Graphics. Basically, we work within Adobe Illustrator. After creating the main page, we added four functionalities,1st is the visualization portion here we create two subparts. The first is the pathfinder approach and the second is the sorting visualizer approach.

In the pathfinding approach, the authors implement some javascript logic to visualize searching algorithms. Here we visualize seven popular searching algorithms - First Dijkstra's Algorithm (weighted): the father of pathfinding algorithms ,guarantees the shortest path, second A* Search (weighted): arguably the best pathfinding algorithm, uses heuristics to guarantee the shortest path much faster than Dijkstra's Algorithm, third Greedy Best-first Search (weighted): a faster, more heuristic-heavy version of A*, fourth one does not guarantee the shortest path, fifth is Swarm Algorithm (weighted) : a mixture of Dijkstra's Algorithm and A*, does not guarantee the shortest-path, sixth is Convergent Swarm Algorithm (weighted): the faster, more heuristic-heavy version of Swarm; does not guarantee the shortest path, seventh is Bidirectional Swarm Algorithm (weighted): Swarm from both sides; does not guarantee the shortest path, eighth is Breadth-first Search (unweighted): a great algorithm; guarantees the shortest path, ninth is Depthfirst Search (unweighted): a very bad algorithm for pathfinding; does not guarantee the shortest path.

In addition to the pathfinding algorithms listed above, the authors have used the Recursive Division Maze Generation algorithm. Swarm algorithm is actually a combination of Dijkstra's Algorithm and A* Search; precisely, while switching to a pointed node such as A*, it still looks at a few neighboring locations around the original node like Dijkstra. The algorithm distinguishes itself from A* by its use of heuristics: it constantly updates the distance of nodes from the first node while taking into account its estimated distance from the specified node. [28] This effectively "measures" the difference in the total distance between the nodes closest to the first node and the nodes closest to the specified node, resulting in a Swarm Algorithm-like triangle formation. We named the algorithm "Swarm" because one of its applications can be seen in a video game where the character has to track the most important boss (target node), all the while keeping track of neighboring enemies who may be crawling nearby. The second part of the first operation is part of the visualization process here and we used the JavaScript concept visualization to filter the algorithms. [29] The filtering algorithm is used to rearrange a given list or list items depending on the comparison operator in the properties. The comparator operator is used to determine a new order of items in the appropriate data structure. This project contains three one-page programs. These are nothing but three operations of our great project. We create a backend database in the firebase to store the user's email and password, after saving, it will download the user id that will be provided by the firebase. After that pass, this is provided with a user id and password to the reaction-chat SDK engine. Chat Engine is an API that makes it easy to build chat services. Building a conversation from scratch takes a lot of time, code, and is expensive. It is better to use the product than to write it from scratch. We've made it easy to create a conversational concept in minutes. We have created three single-page applications and handle these applications using netlify. [30] Netlify is a web development platform that expands productivity. By integrating modern web-based features, from local development to advanced comprehension, Netlify enables 10x faster access to more efficient, secure, and awesome websites and apps. At the back of this store, it shows the details of many easy-to-read algorithms, and the details are the time-complexity of the algorithms, as well as their repetitive relationships, logical approach, etc. To perform these one-page programs we use the reaction-router-dom.

Routing is a process in which a user is redirected to different pages depending on their action or request. [31] ReactJS Router is mainly used to develop Single Page Web Applications. React Router is used to define multiple routes in the app. When a user types a specific URL in a browser, and if this URL path matches any 'path' within the router file, the user will be redirected to that specific route. Provides sync URL in browser with details to be displayed on the web page. Maintains the normal design and behavior of the app and is mainly used to develop single-page web applications. After that, we created a chat app using the pre-discussed response engine. At this stage, users will discuss their doubts together. To do this part of the conversation we use [32] Axios to generate applications in the reaction application. Axios is an HTTP client based on the promise of the browser and Node. js. It can be used in plain JavaScript or in a library such as Vue or React. [33] Consuming REST APIs in React applications can be done in a variety of ways, but we have discussed how to consume REST APIs using the two most popular methods known as Axios (HTTP-based client) and Fetch API (built-in browser) web API).

Axios helps us make HTTP requests. After receiving the data, we add it to the state, so it is ready to use by our system. Finally, we create a collection of contacts, which takes back to the firestore database used as a backend and another purpose of this section is to re-create the frontend for sending feedback and contact us sections using bootstrap and sass (awesome style sheets). Firestore allows for complex ACID transactions against the document data. This provides more flexibility in the way we build our data. Includes HTML and CSS templates design for typography, forms, buttons, tables, navigation, modals, image carousels, etc. Here we use sass to create a frontend, Sass (representing Syntactically awesome style sheets) is a CSS extension that enables to use of things like dynamics, integrated rules, in-line import, and more. It also helps to keep things organized and allows us to create style sheets quickly. [34] Sass is compatible with all types of CSS. After creating a total of three one-and-a-half view applications and combine all functionality and create a multi-page responsive program using vercel development. We start by deploying zero configuration in our global boundary network. Here the main solution to the concept is firebase auth connected to respond to SDK chat engine vercel. Here we set firebase project settings and project private key designed for the react chat-engine into vercel

environment's variables. After deploying this project using vercel deploy. Below in Table 1, we can see an analysis of the time complexity of the algorithms reflected in the initial implementation of our project.

4. Experimental Setup

The architecture of the model is shown in figure 1.

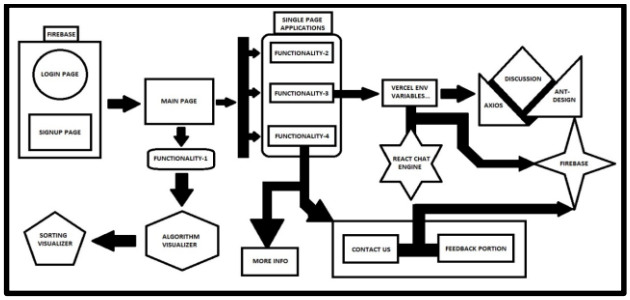

Fig. 1. Architecture of the model

5. Time Complexity Analysis

Table 1
Time complexity analysis of various algorithms

Algorithms	Time Complexities
DIJKSTRA'S ALGORITHM	O(V^2)
BFS ALGORITHM	O(V+E)
DFS ALGORITHM	O(V+E)
A * SEARCH ALGORITHM	O(E)
BUBBLE SORT ALGORITHM	O(n^2)
SELECTION SORT ALGORITHM	O(n^2)
INSERTION SORT ALGORITHM	O(n)
MERGE SORT	O(nlogn)
QUICK SORT	O(nlogn)
HEAP SORT	O(nlogn)

6. Future Scope

1. Students while working on a course have to solve assignments. For solving exercises and finishing assignments students can use this algorithm visualizer.
2. Students like discussion-based study through an interactive platform. So that they will use it.
3. Without searching about algorithms students will get enough information about algorithms. It will save them time.
4. In the age of online study, it is a great opportunity for students of learning algorithms by visualizing the working flow. The visualization-based study is a trend of present days and the way is very efficient for understanding.

7. Conclusion

Algorithm Visualizer is an interactive online platform that visualizes algorithms as per the user's choice. It has been created using JavaScript, CSS, SCSS, ReactJS, HTML5, Vercel, Firebase, Chat Engine XDK, nodejs, and Formik. Users can approach this website through a provided link. First of all, he has a sign-up or login page. If the user is opening it for the first time he has to select the sign-up option. Here he has to give a username, email id, password, verify the password. If he has already signed up, then he has to go to the login option, and then he has to give an email, password, confirm password. Then we have to enter the main page or home page. Here user can see your email id and a firebase-provided password.

Here four buttons are given. VISUALIZE ALGORITHMS, ALGORITHMS, DISCUSSION, FEEDBACK. By clicking on VISUALIZE ALGORITHMS users can watch and understand the working of some specific algorithms. If the user goes to ALGORITHMS he can see a total of 72 algorithms. Sorting, Searching, and Others algorithms. If he clicks any of them in learn more button, then their details will be shown on a page. The DISCUSSION part will take the user to a login page where he has to give a username and password (firebase-provided password on the home page). Then he will be added to a discussion group for doubt clearing where the developers will answer his questions regarding this website. In the feedback, portion the user can write about how much this website is useful or any idea for improvement in this website in the comment section.

Applications of Algorithm Visualizer are various like the following:

1. Learning an algorithm gets much easier with visualizing it. Algorithm Visualizer lets users visualize different types of algorithms. User can understand the algorithms and their working flow easily. It contributes three types of algorithms. They are sorting algorithms, searching algorithms, and others algorithms.
2. This web application has a descriptive portion of 72 algorithms which is very useful for students of Computer Science and Mathematics subjects. Here time complexity, space complexity, definition, and other information are gathered for students so that they don't have to search all information in google spending a lot of time.
3. Students or anyone who wants to learn algorithm they can interact with the developers or any other user for any query, related to this web app. This idea of interaction and a safe chatting area provides users with a nice way of learning.
4. This web app can be useful for students, interns, office workers, or anyone who want to learn algorithms and their working flow. This online interactive platform base study is trending these days.

References

[1] McGettrick A, Boyle R, Ibbett R, Lloyd J, Lovegrove L, Mander K. Grand challenges in computing education – A summary. The Computer Journal 2005;48(1):42-48.

[2] Rößling G. The Animal algorithm animation tool. 5th Annual SIGCSE/SGCUE Conference on Innovation and Technology in Computer Science Education, ITiCSE'00. Helsinki, Finland; 2000. pp. 37-40.

[3] Pierson W, Rodger S. Web-based animation of data structures using JAWAA. 29th SIGCSE Technical Symposium on Computer Science Education. 1998. pp. 267-271.

[4] Sorva J, Sirkiä T. UUhistle: a software tool for visual program simulation. In: Proceedings of the 10th Koli Calling International Conference on Computing Education Research (Koli Calling '10). New York: ACM Press; 2010. pp. 49-54.

[5] Fouh E, Akbar M, Shaffer C. The role of visualization in computer science education. Computers in the Schools 2012;29:95-117.

[6] Baecker R. Sorting out Sorting. Narrated colour videotape, 30 minutes, Presented at ACM SIGGRAPH'81, 1981.

[7] Stasko JT. Using student-built algorithm animations as learning aids, In: 28th SIGCSE Technical Symposium on Computer Science Education; 1997. pp. 25-29.
[8] Karavirta V, Korhonen A, Malmi L, Stålnacke K. MatrixPro: A tool for on-the-fly demonstration of data structures and algorithms. In: Proceedings of the Third Program Visualization Workshop; 2004. pp. 26-33.
[9] Hundhausen D, Brown J. What you see is what you code: A 'live' algorithm development and visualization environment for novice learners. Journal of Visual Languages and Computing 2007;18(1):22-47.
[10] AlgoViz.org Bibliography. Annotated bibliography of AV literature. http://algoviz.org/biblio, 2011.
[11] Ma L, Ferguson J, Roper M, Wood M. Investigating and improving the models of programming concepts held by novice programmers. Computer Science Education 2011;21(1):57-80.
[12] Guo JP. Online Python Tutor: Embeddable Web-Based Program Visualization for CS Education. In: Proceedings of the 44th SIGCSE Technical Symposium on Computer Science Education. New York: ACM. 2012. pp. 579-584.
[13] Vrachnos E, Jimoyiannis A. Dave: A Dynamic Algorithm Visualization Environment for novice learners. 8th IEEE International Conference on Advanced Learning Technologies 2008. pp. 319-323.
[14] Garner S, Haden P, Robins A. My program is correct but it doesn't run: a preliminary investigation of novice programmer's problems. In: ACE'05: Proceedings of the 7th Australasian conference on Computing education. 2005. pp. 173-180.
[15] Du Boulay B. Some difficulties of learning to program, In: Soloway E, Spohrer JC, editors. Studying the Novice Programmer, Hillsdale, NJ: Lawrence Erlbaum Associates; 1986. pp. 238-299.
[16] Dale NB. Most difficult topics in CS1: results of an online survey of educators. SIGCSE Bull. 2006;38(2):49-53.
[17] Robins A, Rountree J, Rountree N. Learning and teaching programming: A review and discussion, Computer Science Education 2003;13(2):137-172.
[18] Soloway E., Spohrer, JC. Studying the novice programmer. NJ: Lawrence Erlbaum; 1989.
[19] Jimoyiannis A. Using SOLO taxonomy to explore students' mental models of the programming variable and the assignment statement. Themes in Science and Technology Education 2011;4(2):53-74.
[20] Danielsiek H. Detecting and understanding student's misconceptions related to algorithms and data structures. Proceedings of the SIGCSE 2012 Technical Symposium on Computer Science Education. Raleigh, North Carolina: ACM Press; 2012. pp. 197-201.
[21] Brown, M., Najork, M., and Raisamo, R. 1997. A java-based implementation of collaborative active textbooks. In Proceedings of the IEEE Symposium on Visual Languages (VL'97), pp. 372-379.
[22] Galles, D. 2006. Data structure visualization. http://www.cs.usfca.edu/galles/visualization/.
[23] Jarc, D. J., Feldman, M. B., and Heller, R. S. 2000. Assessing the benefits of interactive prediction using Web-based algorithm animation courseware. In Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education (SIGCSE'00), pp. 377-381.
[24] Lawrence, A. W., Stasko, J., and Badre, A. 1994. Empirically evaluating the use of animations to teach algorithms. In Proceedings of the IEEE Symposium on Visual Languages (VL'94), pp. 48-54.
[25] Naps, T., Rössling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., and Ángel Velázquez-Iturbide, J. 2002. Exploring the role of visualization and engagement in computer science education. In Proceedings of the Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education (ITiCSE-WGR'02), pp. 131-152.
[26] Rössling, G., Naps, T., Hall, M., Karavirta, V., Kerren, A., Leska, C., Moreno, A., Oechsle, R., Rodger, S., Urquiza-Fuentes, J., and Ángel Velázquez-Iturbide, J. 2006. Merging interactive visualizations with hypertextbooks and course management. SIGCSE Bull, vol. 38, no. 4, pp. 166-181.
[27] Rössling, G., Schüer, M., and Freisleben, B. 2000. The animal algorithm animation tool. In Proceedings of the 5th Annual SIGCSE/SIGCUE ITiCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE'00), pp. 37-40.
[28] Grissom, S., McNally, M., and Naps, T. 2003. Algorithm visualization in CS education: Comparing levels of student engagement. In Proceedings of the ACM Symposium on Software Visualization (SoftVis'03), pp. 87-94.
[29] Hundhausen, C. and Douglas, S. 2000. Using visualizations to learn algorithms: Should students construct their own, or view an expert's? In Proceedings of the IEEE Symposium on Visual Languages (VL'00), pp. 21-28.
[30] Hope College. 2001. Complete collection of algorithm visualizations. http://www.cs.hope.edu/~dershem/ccaa/ccaa.
[31] Brown, M. 1992. An introduction to Zeus. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'92), pp. 663-664.
[32] Crescenzi, P., Gambosi, G., and Grossi, R. 2006. Strutture di Dati e Algoritmi. Pearson Education Addison-Wesley.
[33] Byrne, M. D., Catrambone, R., and Stasko, J. T. 1996. Do algorithm animations aid learning? Tech. rep. GIT-GVU-96-18, Georgia Institute of Technology.
[34] Diehl, S. 2007. Software Visualization: Visualizing the Structure, Behavior, and Evolution of Software. Springer.