Advances and Applications in Mathematical Sciences
Volume 21, Issue 5, March 2022, Pages 2721-2736

ALGORITHM VISUALIZATION - MODERN WEB-BASED VISUALIZATION OF SORTING AND SEARCHING ALGORITHMS

ASHISH KUMARI[1], MANAV MITTAL[2], VIPUL JHA[2], ABHISHEK SAHU[2], MANISH KUMAR[2], NEETI SANGWAN[3] and NAVDEEP BOHRA[*,3]

[1]Assistant Professor, [2]Student
Department of Information Technology
Maharaja Surajmal Institute of Technology
New Delhi, India
E-mail: ashishkumari@msit.in
manav12121999@gmail.com
vipuljha1142@gmail.com
abhisheksahu618@gmail.com
officialmanishkr98@gmail.com

[3]Assistant Professor
Department of Computer Science
and Engineering, Maharaja Surajmal
Institute of Technology, New Delhi, India
E-mail: neetisangwan@gmail.com

Abstract

Concepts of Data Structures and Algorithms (DSA) are often difficult for students to understand. To improve understanding of DSA, animation and graphical representation techniques have evolved and are often used to understand the step-by-step workflow of algorithms. Algorithm visualization (AV) is also one such technique. This paper presents an online web-based interactive tool built on modern-day JavaScript to visualize the algorithmic logic of various searching and sorting algorithms. It enables the users to understand by visualizing basic algorithms of sorting and searching. AlgoViz aims to present the core logic and

2020 Mathematics Subject Classification: 68T30.
Keywords: Data Structures, Algorithms, Visualization, Program.
*Corresponding author; E-mail: navdeepbohra@gmail.com
Received September 19, 2021; Accepted November 30, 2021

basic difference between several algorithms present in the industry. It shows the step-by-step process of a algorithms with bar graphical animation and compares them on the basis of their space and time complexity with their average, worst and best case scenarios. AlgoViz also deals with interactivity in AVs by providing various controls and features over the graphical simulation.

1. Introduction

In today's generation computer programming have become a very important skill for students to acquire new age technologies. They need to be well equipped with the ability of critical thinking or algorithmic thinking and problem-solving skills apart from the basic usage of computers.

Data structures and algorithms (DSA) are base and it plays a crucial role in computer science education and programming. DSA helps in creating better technology like FastTTps, etc. [1]. Students are expected to be well versed in several algorithms and data structures (DS) to perform better in their curriculum and in their further career opportunities.

Learning and teaching DSA are both challenging for students as well as teachers. Past researches have shown that the new learners find it difficult to understand the implementation of data structures (DS) and algorithms in programming [2]. Instructors and educators until recently use physical means like pen-paper, textbook diagrams, etc. This requires a great deal of effort to convey the logic of an algorithm. Educators from around the world mentioned the disinterest of the students towards DSA deo to their abstract and dynamic nature. To understand the whole working process of an algorithm, it needs to be broken into steps and it is very difficult to present these steps with static tools like texts and pictures.

There have been many solutions found by the researchers to overcome these barriers like graphical representations, classical animations, explanation videos, etc. These methods enable the learner to visualize the logic but do not provide the diversity to work with different data sets or reuse any existing asset for another similar task with a different approach. Apart from these approaches there evolves an approach that gains popularity amongst the educators in computer science education is Algorithm Visualizer (AV).

An algorithm visualizer is a useful education tool that helps students to

visualize an algorithm by using transition effects and basic animations to understand how an algorithm is working at each step. Many researchers have concluded that the students who are involved in the visualization process of the data structure and algorithms to understand the key working of any algorithm yield a better result in comparison to others who use classical methods because of the dynamic visual representation and interactivity of the algorithm visualizers [3].

Overall, algorithm visualizers help learners to get a better understanding of DSA. It also increases the interest of the user by providing more interactive environment than simple text or classic animations. It solves the problem for teachers to recreate the same presentation for algorithms that uses a similar data structure, as it becomes very easy to change the existing data and base logic of the visualization according to the need.

The forthcoming parts of this paper discuss the previous researches and different AV applications developed. It also gives a detailed view of how they come into the picture and what are the existing algorithms visualizers in the field of computer education for a very long time. Post that, the project introduces our proposed work with all of it's design framework along with its visualization interface and functionalities. Thereafter the different use cases of the project are discussed followed by the conclusion.

2. Related Research

Algorithm Visualizers have a long history in computer science education as they are being used as an effective learning tool and have received increasing interest from both students and educators. There have been a lot of advancements in visualization. Since starting, hundreds of Algorithm Visualizers have been implemented and provided to the computer science education community. ANIMAL [5], JAWAA [6], Jeliot [7], JHAVÉ [8], TRAKLA2 [9], JSAV [10] and various other projects on smaller scales are previously built Algorithm Visualizers that gained popularity.

2.1 Algorithm visualizer trends. As numerous visualization systems have been built over these years, the computer science education community has praised this form of learning and teaching [8, 11]. It is out of the scope of this paper to introduce each one. We will just get to know about some of the most popular and effective algorithm visualizer tools or systems.

One of the popular techniques used for creating and designing the algorithm visualizers in the early days is by annotating the algorithm code by using commands for scripting programming to generate the visualization. One of the popular Algorithm Visualizers that used this technique for visualization purposes is JAWAA [6] which uses scripting language for generating animations of the data structure and algorithms by simply adding its commands in place of the output of the program. Since it is a scripting type language based on Java, a layman in the computer who has just started to learn may not be able to take the advantage it offers.

To overcome this, another type of drawing tool like ANIMAL (A New Interactive Modeler for Animations in Lectures) is designed [5], which is a general-purpose animation tool with a focus on algorithm animation. Other tools like JHAVÉ (Java-hosted Algorithm Visualization Environment), are designed and practiced for a very long time due to their capability to provide code changing of the algorithms [8]. Although it is not a complete Algorithm Visualization tool, instead, it provides a supportive environment for various other visualizers to work. It is dubbed as an algorithm visualizer engine in the field of this work. Similar to ANIMAL, this tool is based on a desktop environment and needs the installation of software and different packages.

These AVs had quirks and were not able to grow with the development of the industry. With the rise of browser and internet, Algorithm visualizers also started to adopt this trend and projects like DAVE [11], a web-based AV used to animate various algorithms that are applicable to the array data structure. It uses Java applets to run inside a browser that supports them.

Another project that uses Java and Java applets is JSAV [10]. The use of Java applets comes with its own challenges in modern web technologies. The support for Java applet has ended in major browsers [12] and most of the new browsers does not allow the tools due to security and privacy. With modern technologies like HTML and JavaScript, it provides a system for programmers to design almost any kind of algorithm visualizer.

2.2 Ideal algorithm visualizer features. Since the introduction of Algorithm Visualizer, it has been expected to produce a substantial difference in the field of study. However, there are conflicting results on the effectiveness of visualization methods over traditional methods. [13]. One of the reasons for not achieving desired results is the lack of simplicity and

availability to wider audiences. We drafted some main features that an AV should have in order to achieve the goal effectively. Those features are following listed.

F1: Ease of usability for novice programmers: It should be easy to use even for a layman in the computer field.

F2: Platform Independence: In the modern world, the software has to be platform independent in order to reach a mass audience.

F3: No programming bounds: An ideal AV should not require programming from the end-user to visualize a particular algorithm.

F4: Interactive animation: Users should be given control of the animation for better engagement.

F5: Fast and responsive: It should not lag or slow in its working. It should also be small in package size to load up faster.

We do not state that this list of required features is complete and an AV should only follow these features but present our intake on necessary features that an AV should have in order to help our case of teaching environment.

2.3 Evaluation of existing algorithm visualizers

We evaluated some of the existing and popular algorithm visualizers on the condition of having features that are stated in the previous sections. We have chosen and evaluated JAWAA, ANIMAL, JHAVÉ, DAVE and JSAV.

After evaluating these algorithm visualizers strictly on the condition of our list of features, we found some major problems that are necessary to overcome for us. Out of all these Algorithm Visualizers, only JSAV is built with a modern technology framework. It uses the JavaScript programming language and runs on web browser. All of the others are using Java programming and run either on desktop environment or in browser using Java applets. In the past, Java was one of the best options available when comes to platform independence, but in modern computer era, we have more devices to support and the web browser is one such environment that is supported in most devices. Java applets are now not supported in many popular browsers [12] that makes these AVs lack feature F2.

JAWAA (Java and Web-based Algorithm Animation) is also written in java and it provides an interface through which users can write their own animations as well as display these animations in a browser-based environment with the help of Java Applets. The commands are easy to understand but more sophisticated for the end-users as we need to use exact coordinates to plot the data in display frames to animate. It is old and not much work has been done to match the technological advancement since then. Therefore, it lacks features F1, F2, and F3.

The ANIMAL (animation in lectures) is also a Java-based Algorithm Visualizer tool in which animations are generated using visual editors, scripting and API calls. It works using API calls which is a network-intensive task. This makes the process slow and delays the process of rendering the visualization. It is an AV system and needs scripting to perform visualization. Thus, because of the above-mentioned reasons, it lacks the features F2 and F5.

JHAVÉ (Java-hosted Algorithm Visualization Environment), works on Java in the form of plug-ins. It is mainly focused on increasing user engagement in the process of visualization. It has an animation engine that provides VCR-like controls to control the animation, information and pseudo-code windows to view the needed information and actual code that is being rendered currently. In-between the user gets some questions as an exercise regarding the current algorithm from the education and learning perspective. It lacks features F1, F2, F3, and F4.

DAVE is one of the most updated from the pool of AVs. It utilizes the features provided by the modern-day browser by integrating the tool with Web technologies. It gives the facility to experiment by doing modifications in the code and data of existing array algorithms in the project. It is written in Java and uses Java Applets to render visualizations in a web browser [11]. One of the major drawbacks is that it provides too much control over the interactive options for a novice programmer. The control over each bit of the AV serves as a challenge for a novice programmer who is not familiar with the basics of an algorithm, making the learning curve difficult. DAVE heavily incorporates the idea of interactive exercises to learn a particular algorithm. It lacks features F1 and F2.

JSAV is built using modern web technologies like JavaScript and

HTML5. It runs on modern age browsers, making it platform-independent. It lacks feature F3, as it is an AV system that is used by developers to build an AV but not by students.

These researched AVs focus on one or two features from the mentioned features list instead of implementing all the features to boost the process of learning through Algorithm Visualizers. Having listed features lacked by popular algorithm visualizers, we decided to create our own AV with an attempt to include all the listed features.

3. Proposed work

In this section, we are introducing AlgoViz, A modern web-based visualization software for sorting and searching algorithms.

3.1 Improvements over evaluated algorithm visualizers

This application is developed to adapt all the listed features in previous sections. We have used a new and different approach by using modern visual design curated for the user of the modern era. We have used bars to represent the different data values present in the data structures. This helps in visually differentiating each element present in the data set. The application is made easy for the end-user to use. It has been toned down to an extent where a user who has little or no knowledge can also operate and understand the visualization process. It can be easily controlled by play and stop buttons from the control panel. It uses modern web technologies for platform independence by using JavaScript and HTML5 in-browser environments. The uprising of HTML5 and web technologies with increasing capabilities of modern web browsers graphics have enabled the developers of these algorithm visualizers to build sophisticated applications or tools for web browsers also [13]. In the modern world, smart devices like mobiles and tablets are also becoming capable enough to execute such complex tasks as graphics rendering and running web applications. AlgoViz utilizes the power of the modern web to showcase our application using JavaScript. AlgoViz provides a comparison chart between different sorting and searching algorithms. It shows a time and space complexity comparison graph with a defining parameter as the number of elements. AlgoViz does not require a user to code in any form. It presents a graphical user interface with control buttons to interact. This makes it easy to use and free from code anxiety for

novice programmers. It completes the features F1 and F3. AlgoViz has a very small package size and is delivered through fast CDN (Content Delivery networks), making it faster to load on the browser and lag-free as needed for feature F5. AlgoViz also allows the users to control the amount of delay between two animation frames or algorithm steps. This helps users to visualize the animation at a favorable speed that suits to their personal preference.

3.2 Interface and functionality. As simplicity and ease of use are one of the most necessitated features for AlgoViz. We tried to develop an AV tool which is user-friendly by having modern looking design and easy to understand controls. On the home screen of AlgoViz, there are two buttons for sorting and searching which links to their respective pages. Both the pages have similar user interfaces for a unified experience.

3.2.1 User interface. AlgoViz's user interface consists of a visual representation of data stored in the array data set.

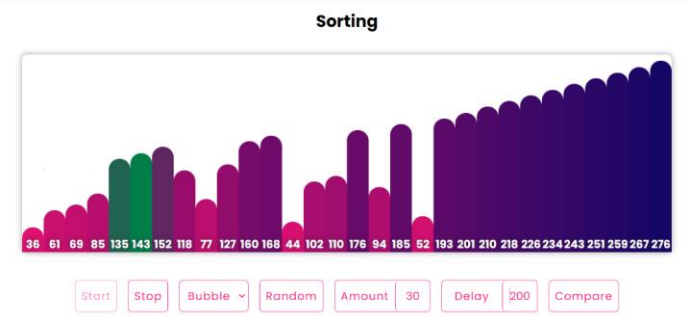

Figure 1. Sorting visualization UI.

This data is represented in form of bars with their height corresponding to the value of the element as shown in figure 1. Height is one of the most easily recognized factor for differentiating. This representation section is called Animation Arena and it is the main section of the AlgoViz to visualize the process of sorting or searching. Figure 1 also shows the bars have colors in increasing order of contrast between them to further easily differentiate between bars of different values. The darker color is associated with the larger height representing larger values while the lighter shade of color is given to the smaller height bars representing small values. As the animation starts, in each frame, some bars are highlighted in a different color to

represent the indexes at which the algorithm is currently processing in this step. Animation is updated frame by frame, in which each frame represents the current state of data in the array after the current step of processing in an algorithm.

Below the Animation Arena, AlgoViz has a control bar to control the animation attributes. Starting from the left, the first button is the start button to start an animation with the selected algorithm. The second button is the stop button to stop the currently ongoing animation. By using these two buttons a user can control the flow of the animation. The next control is a drop selection for all the available algorithms for the visualization of a category. The different algorithms available are Bubble sort, Insertion sort, Selection sort, Quick sort, Merge sort and Heap sort in sorting and Linear search, Binary search and Interpolation search in searching algorithms.

The next control button is a random data generation button which replaces all elements in the array with new data. To avoid ambiguity, it stops any current ongoing animation. The number of elements can be changed by the user by changing the value in amount. It has a limit of 900 elements for overflow conditions. Next control is an important feature for any algorithm visualize. It represents the delay between each animation frame. This helps the users to perceive the visualization at their own speed. The last button is a compare button to show the comparison between different similar task performing algorithms. It is not a control feature and we will explain it in further sections.

3.2.2 Sorting visualization. AlgoViz has six popular sorting algorithms, that are commonly taught in universities and colleges. These algorithms are Bubble sort, Insertion sort, Selection sort, Quick sort, Merge sort and Heap sort. The best sorting used in the industry is index tree sort used for big data sorting [14]. A user can start the visualization process and animation by selecting the algorithm to be visualized and giving the necessary inputs like number of elements and delay time. The type of algorithm user wants to visualize in the graphical animation arena can be selected from the dropdown list in control bar.

After completion of sorting, the animation will stop and animation area will contain sorted bars with increasing order of height. Each bar will represent an element in the array that is used internally for sorting. The

start button's color is faded out to represent that the visualization is in progress. The bars which are getting compared by the insertion sort are highlighted in green color. By this simulation, a learner can understand how each algorithm has different processing for an equal amount of random data and also understand about its time and space complexity.

3.2.3 Searching visualization. AlgoViz have three simple and most taught searching algorithms to simulate. The algorithms are Linear search, Binary search and Interpolation search. AlgoViz implement these algorithms on a one-dimensional linear array to make things simpler for the naive in computer technology. Here, alike sorting, the parameters like selected algorithm, the number of elements and the delay time can be changed as per the user preference.

The user has to input is the element to be searched by the algorithm in the array. The selected algorithm will search for the entered number in the search input box and keep updating the user about the comparison of elements that are happening at any iteration below the animation arena.

In linear searching, the algorithm will compare the search element directly from all the elements in the array starting from left and display the result after the search is completed. It starts by comparing the elements in the array to the number in the search input box. For example, in figure 2, element 44 is found at index 19 of the array.

In the binary search, the array will be sorted for the search algorithm to perform, as the binary search only works on sorted data. If the element is found in the array it will display the index at which the element has been found in the array or will display that element is not found if the element does not exist in the array. For example, in figure 3, the array is sorted before starting the search. Also, the searched element 136 is not found in the array using the binary search in this example.

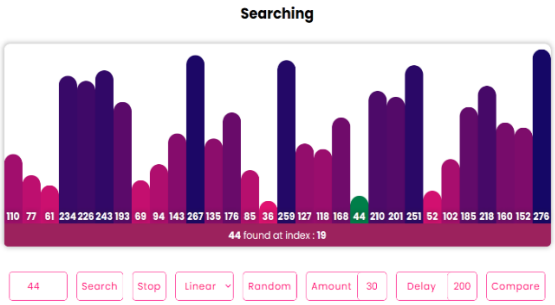

Figure 2. Searching algorithm UI.

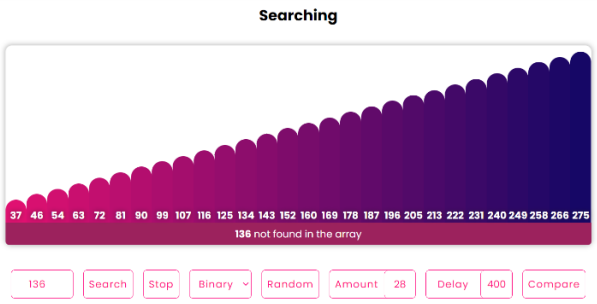

Figure 3. Binary search visualization process.

3.3 Working on web. For an algorithm visualizer to be effective, it has to reach a wide amount of audience. From the very ideation of the project, we are assured to demonstrate our tool on a web browser. In this modern technology age, browsers are being used almost in each device and each field of computer science. This makes things more accessible to a wider audience.

The increasing capabilities of browsers to render complex graphics and perform high-level calculations have enabled developers to build large applications that are solely web-based. The benefit of using a completely web-based application is that it can be run on practically any digital device that has web browser support on it.

Almost every device with internet connectivity has the ability to install a web browser on it. From smartphones to tablets to laptops and desktop PCs, each internet-enabled device is equipped with a web browser. AlgoViz utilizes advanced graphical capabilities of the web browsers to render its animation or perform the simulation of algorithms.

4. Comparison graph

AlgoViz has a compare button to display comparison chart for sorting and searching algorithm. The graph display consists of a double bar graph in which, the first teal colored bar shows the time taken by an algorithm to complete the operation on a given data. The second pink colored bar shows the computing space required to complete the algorithm process. These charts present how different algorithms work in comparison to each other on a similar dataset. The graph enables the user to understand the time and space taken and occupied by different algorithms when implemented on a similar dataset. It enables the thinking and processing amongst the novice computer science students about how effective an algorithm can be and encourages the user to explore the areas where the different algorithms can be useable and where it is not. There are basically three cases for evaluation of the efficiency of an algorithm, i.e., the average case, worst case and best case. We included these scenarios for the algorithms to work upon. A worst-case occurs for an algorithm when it takes maximum time to complete in a situation or on a set of data. For example, the performance of Bubble sort will be worst with time complexity $O(\log n)$ when the input array is reversely sorted as clearly visible in the figure 4.

A best-case scenario as shown in the figure 5 is when an algorithm performs its task under smallest amount of time in some conditions. Just like the performance of Bubble sort will be best when the input array is already sorted. The Average Case is the closest to the expected result in the real world. It is calculated by taking all the possible inputs and taking the average of the time and space taken. Like the unsorted array is given input to the bubble sort to implement.

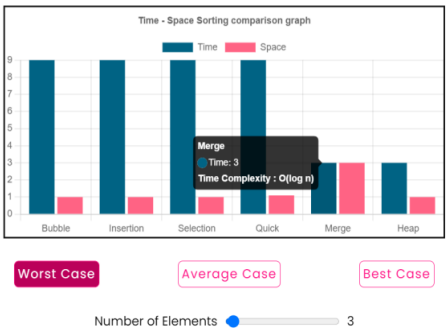

Figure 4. Sorting comparison for worst case.

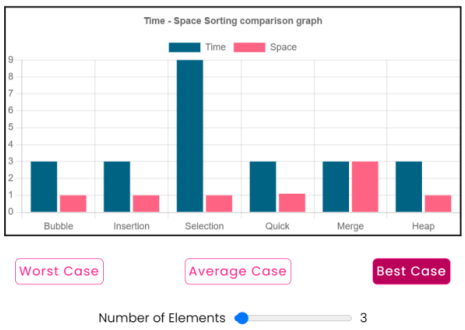

Figure 5. Sorting comparison for best case.

The comparison will update based on the number of elements in the array. Each algorithm performs differently according to the size of the data. Some algorithms might be a good option for a small data set but can be the worst for a large dataset. For example, the Merge sort might not be preferred for small data set but can be handy for large datasets because of its algorithm which divides the array into smaller units. These smaller units can be processed at a much faster speed when the data cannot be entirely loaded to the main memory. AlgoViz comparison graph works on the exact time and space complexity of all the algorithms.

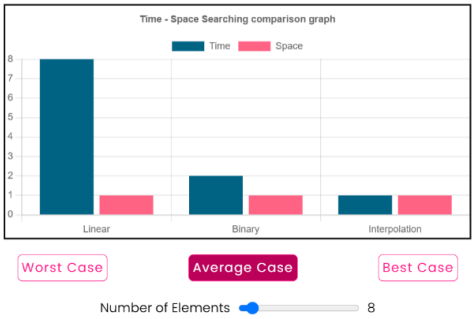

Figure 6. Searching comparison for Average case.

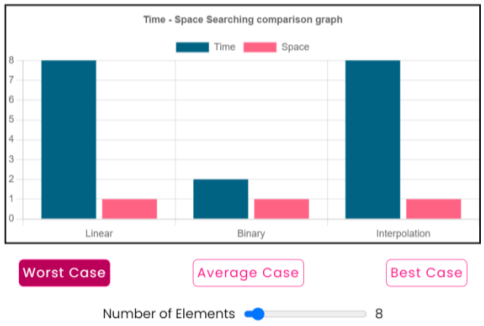

Figure 7. Searching comparison for worst case.

We can verify that the quicksort algorithm does not perform well in the worst-case scenario as its time complexity is $O(n^*n)$ [15]. The heapsort algorithm does not change its performance in any scenario and gives a time complexity. Bubble and Insertion sorting algorithms behave alike with their worst-case scenarios. In best case situations, they tend to process the array faster. This verifies the behavior of insertion sort for sorting sorted elements.

Just like this, we can conclude that the Selection sort performance is similar in each of the three cases as shown in the comparison graphs. This sorting algorithm has a time complexity of $O(n^*n)$ irrespective of the case scenario [15].

Just like the sorting algorithm comparison graph, AlgoViz has a comparison graph for searching algorithms too. These graphs show the behavior of algorithms under each case. These graphs conclude that the linear search is not a viable option for average and worst case because of the

such a large time complexity, that is $O(n)$, which increases linearly with the size of the array [16]. This is clearly visible in the figure 6 and figure 7 in the comparison graph.

There is an interesting result we can conclude from the figure 7, in which, worst case comparisons are shown of searching algorithms that although interpolation search is better in average case as shown in figure 6 but under worse conditions, when element is not present in the array, it goes to time complexity of linear search, which is $O(n)$.

Although each algorithm takes similar time in searching the element in best case, the benefit of using linear search for very small data set is that the array or dataset is not required to be sorted unlike the other two searching algorithms whose prerequisite is a sorted array.

These graphs also present the effectiveness of binary search and Interpolation search over linear search. The tradeoff of sorting the array got negligible on a very large dataset.

5. Conclusion and Future Scope

In this paper, a new modern Algorithm Visualizer for various sorting and searching algorithms is presented with its implementation and features in detail. It describes the need for an algorithm visualizer and compares AlgoViz with various popular and efficient Algorithm Visualizers. It states the important features that every AV should implement in order to create an effective Algorithm Visualizer. This paper details why a complete web-based simulation software is built instead of classic desktop software based upon Java language or a web-based with Java applets.

AlgoViz can have more features which can enhance the learning of algorithms even better. We can also customize the data as well as add more sorting and searching algorithms to enhance the software reliability. More features i.e., greedy and dynamic programming algorithms, can also add there. AlgoViz can have more responsive for devices with smaller screen sizes.

References

[1] P. Agrawal, A. Zabrovskiy, A. Ilangovan, C. Timmerer and R. Prodan, FastTTPS: fast

approach for video transcending time prediction and scheduling for HTTP adaptive streaming videos, Cluster Computing 24(3) (2021), 1605-1621.

[2] H. Danielsiek, W. Paul and J. Vahrenhold, Detecting and understanding students' misconceptions related to algorithms and data structures, 43rd ACM technical symposium on Computer Science Education (2012), 21-26

[3] G. Tuparov, D. Tuparova and V. Jordanov, Teaching sorting and searching algorithms through simulation-based learning objects in an introductory programming course, Procedia-Social and Behavioral Sciences 116 (2014), 2962-2966.

[4] K. Booten, Harvesting ReRites, Johnston, DJ ReRites: Human+ AI Poetry+ Raw Output+ Responses. Montréal, QC: Anteism., (2019)

[5] G. Rößling, M. Schüler and B. Freisleben, The ANIMAL algorithm animation tool, 5th annual SIGCSE/SIGCUE ITiCSE conference on Innovation and technology in computer science education (2000), 37-40.

[6] W. C. Pierson and S. H. Rodger, Web-based animation of data structures using JAWAA, ACM SIGCSE Bulletin 30(1) (1998), 267-271.

[7] A. Moreno and M. S. Joy, Jeliot in a demanding educational setting, Electronic Notes in Theoretical Computer Science 178 (2007), 51-59.

[8] T. L. Naps, Jhavé: Supporting algorithm visualization, IEEE Computer Graphics and Applications 25(5) (2005), 49-55.

[9] A. Korhonen, L. Malmi, P. Silvasti, J. Nikander, P. Tenhunen, P. Mård, H. Salonen and V. Karavirta, Trakla2, 9th Koli Calling International Conference on Computing Education Research (2003), 43-46.

[10] V. Karavirta and C. A. Shaffer, Creating engaging online learning material with the JSAV javascript algorithm visualization library, IEEE Transactions on Learning Technologies 9(2) (2015), 171-183.

[11] E. Vrachnos and A. Jimoyiannis, Dave: A dynamic algorithm visualization environment for novice learners, 2008 Eighth IEEE International Conference on Advanced Learning Technologies (2008), 319-323.

[12] Dalibor Topic, White paper Migrating from Java Applets to plugin-free Java technologies, Oracle Corporation, (2016).

[13] P. J. Guo, Online python tutor: embeddable web-based program visualization for cs education, 44th ACM Technical Symposium on Computer Science Education (2013), 579-584.

[14] P. Agrawal, H. Kaur, G. Singh, Indexed Tree Sort: An Approach to Sort Huge Data with Improved Time Complexity, International Journal of Computer Applications 57(18) (2012).

[15] A. D. Mishra and D. Garg, Selection of best sorting algorithm, International Journal of Intelligent Information Processing 2(2) (2008), 363-368.

[16] R. Rahim, S. Nurarif, M. Ramadhan, S. Aisyah and W. Purba, December, Comparison searching process of linear, binary and interpolation algorithm, Journal of Physics: Conference Series, IOP Publishing (930)(1) (2017), 012007.