iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 12 **Issue:** VI **Month of publication:** June 2024

DOI: https://doi.org/10.22214/ijraset.2024.63330

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue VI June 2024- Available at www.ijraset.com

Algorithm Visualization: Bridging the Gap between Theory and Practice

Smriti Chawla[1], Sneha Jindal[2]
[1, 2]*Under Graduate Student, Computer Science Engineering, Dr. Akhilesh Das Gupta Institute of Professional Studies*

Abstract: ***Data structures are the components to store, organize and manage data efficiently. They provide a way to represent and store data in a computer program.***
Algorithms are the set of rules that are set to solve specific task or a problem. Together, data structures and algorithms make the fundamental block of computer science.
We often fail to understand the concepts just because we can't visualize how they work. Visualization can motivate students to learn and understand the concepts which they fail to do otherwise. Many researchers found that visual learning techniques are more promising than the conventional learning techniques. With this inspiration, we developed a tool to visualize different data structures and algorithms.
An e-learning device for different data structures and algorithms visualization is defined in this work. The powerful e-learning tool allows user to choose data structure with variable size and value and see how it works. The application is built using JavaScript as its primary language and uses React.JS framework.
Keywords: Visualization, Algorithms, Sorting, Learning, Data Structures

I. INTRODUCTION

Understanding data and algorithms is important in computer science and software engineering. These concepts form the foundation of software design and development and provide the tools needed to solve complex problems and improve productivity. However, the abstract nature of data structures and algorithms often pose serious challenges to students and professionals. To address these challenges, we created a comprehensive data model and visualization tool, a virtual laboratory designed to enhance learning and understanding through interactive visualization.

A hands-on, visual approach to learning. Data model and algorithm visualization tools provide users with an interactive platform to explore different data types such as arrays, linked lists, trees, graphs, and trees. It also includes analysis methods such as integration, fast analysis, bubble separation, and registration for simple algorithms such as depth-first search (DFS), breadth-first search (BFS), and Dijkstra algorithm.

Interpreting these patterns and processes to better understand how they work can be difficult. Using interactive visualizations, our tools allow users to analyse the step-by-step behaviour of algorithms and dynamic changes in data structures to promote deeper, longer-term understanding. It provides a user-friendly interface that allows users to select the data structure or process they are looking for, access their data, and view detailed views. This interactive approach not only makes learning more effective, but also deepens understanding through learning by allowing users to try different situations and evaluate the results. The visual tools are designed to be platform independent, allowing users to access them from any device using a web browser. We've also used a variety of optimization techniques to ensure that visualizations work well even when working with large files. Interactive visualizations help you create abstract concepts to enhance learning. It serves as a bridge between theory and practice, helping users build a strong foundation in data structures and algorithms. By providing an intuitive and engaging platform, we aim to foster deeper understanding and understanding of these important concepts, ultimately helping to produce more knowledgeable and confident software engineers.

Our visualizers serve as a bridge between theory and practice, helping users build a solid foundation of data structures and algorithms. The motivation for this project stems from the need to create a fun and intuitive learning tool that can help students and professionals understand the complex operation of data structures and algorithms. Traditional methods of teaching these concepts often rely heavily on static diagrams and pseudocode, making it difficult to gain a deep understanding of how these structures and algorithms work in real time.

Data structures and algorithms are important components of computer science curricula around the world. It forms the basis of most programming and problem-solving tasks.

Despite the importance of these concepts, many students find them difficult to understand due to their abstract nature. Visuals can greatly improve understanding, but static images and descriptions are often not enough. Our visualizer solves this problem by providing dynamic, interactive views that can be manipulated and observed in real time. This document is a template.

II. LITERATURE OVERVIEW

"A tool for teaching advanced data structure to computer science students: an overview of the BDP system" by K. Becker and M. Beacham describes a system that gives overview of complex data structures through interactive visualizations. The BDP system enhances the student understanding of concepts in data structures. The paper outlines how BDP system facilitates effective learning and comprehension of advanced data structures.

Dix, Finlay, Abdow and Beale's "Human-Computer Interaction" is a guide to understand interaction between human and computer. The book covers essential topics in HCI including usability, design principles, and user-centred approaches. The book serves as a valuable resource for students and professionals in the field of human-computer interaction.

Ugile, Barure, and Sawant's review paper describes the importance of visualization in data structure and algorithms. How these tools are used to increase understanding using interactive platforms is discussed in this paper. This paper also tells potential research applications of various algorithm visualizers. This paper offers insights of strengths, limitations and future scopes of algorithm visualizers.

B. Goswami, A. Dhar, A. Gupta and A. Gupta published "Algorithm Visualizer: its features and working" where they observed that algorithm design and analysis is one of the building block subjects for a computer science student and can be difficult yet interesting for beginners. Through their work they aim to create a fun-filled learning platform so that sorting, path-finding, CPU scheduling algorithms can be taught in an interactive and perceivable way. Not only students but to teachers also, this idea gives an innovative way to convey their ideas clearly.

A. B. Ghandage, B. P. Udhane, H. R. Yadav, P. S. Thakre, V. G. Kottawar and P. B. Deshmukh in their paper "AlgoAssist: Algorithm Visualizer and Coding Platform for Remote Classroom Learning" proposed that with a shift of remote and digital learning, a platform serving effective learning needs of students is required. AlgoAssist is an integrated platform that fulfils the need of teachers and students for effective online learning. It mainly focuses on the algorithm visualization for effective and better learning.

Brian J. Faria published "Visualizing Sorting Algorithms" where they successfully created a web-based animation tool for visualizing algorithms such as selection sort, bubble sort, merge sort, insertion sort. It was found that learning became much easier with animation which we also found in our research. This received on overall positive feedback from students. This gives a strong mindset to create animations to improve learning.

III. PROBLEM DEFINITION

Being a computer science engineering student, this problem usually comes to our notice that students often face difficulty in developing and implementing the algorithms that are in our programming curriculum. During our placements, data structures and algorithms plays and a very vital role.

It often becomes difficult to remember the techniques due to lack of in-depth exploration of how the algorithms work. Students are often made to learn these algorithms with pen and paper which makes it even harder to understand the core concept. Visuals and animations are considered to be the strongest memory of human.

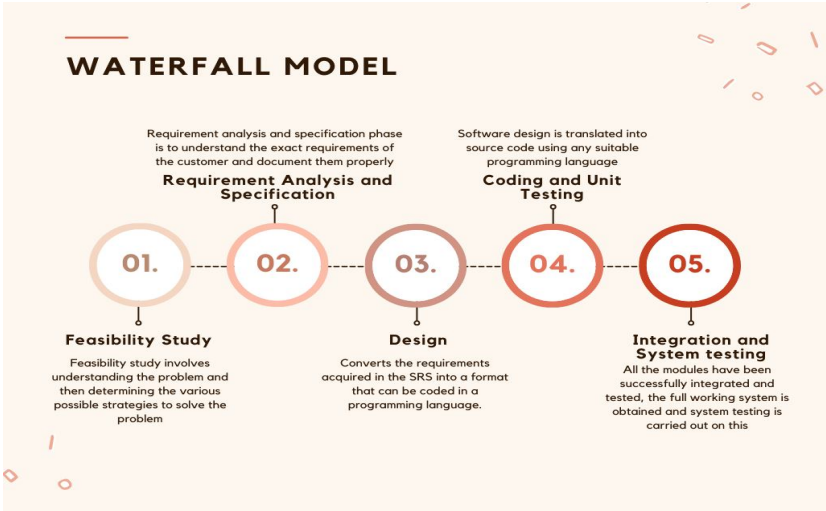
They are hard to forget and helps them to clarify their concepts. Addressing the problems stated above, the main aim of this research is to provide clear and visually appealing way of understanding the various algorithms find inside the data structures, so as to make it easier for students to understand the core concepts. Through this project students could not only grasp the concept easily, but also would be able to assess the type of algorithm to use in a particular environment, enabling them to use this knowledge to develop more optimized algorithms.

IV. RESEARCH METHODOLOGIES

This section describes the research methodologies which are followed to build this project.

A. Planning Phase

This phase mainly focuses on how the project should be developed. This project has followed a waterfall model approach for building the project.



Technology Stack Used

1) HTML

HTML or Hypertext Markup Language is a markup language used to design and create web pages. It creates content which includes titles, sentences, links, images, etc. and defines the contents for the website.

HTML 4.01: It is the fourth major version released in 1999 and has been around for a long time.

XHTML: XML-based HTML reconstruction that combines HTML with XML syntax.

HTML5: The latest and most widely used standards released in 2014, focusing on multimedia support, APIs, and providing consistent syntax.

HTML documents follow a hierarchical structure called the Document Object Model (DOM).

<DOCTYPE HTML>: It is the declaration that specifies that the document is of HTML type.

<HTML>: It signifies the beginning of the HTML file.

<HEAD>: It holds the Metadata of the File and holds essential links used in the file.

<BODY>: It contain the actual code of the HTML file. It is the content that is displayed on the webpage.

2) CSS

CSS or Cascading Style Sheets is a language used to describe documents written in HTML or XML. It controls the structure, format and appearance of web page content to improve users' visual perception and experience.

CSS Type:

Inline CSS: Apply directly to HTML elements using the "style" property.

Internal or Embedded CSS: Defined within the HTML document using the <style> tag in the document's head.

External CSS: Defined in a separate CSS file and linked to the HTML document using the <link> tag.

CSS plays an important role in web development by separating content from presentation, promoting consistent design across multiple pages and devices. Its continued evolution reflects the nature of web design and the need for advanced styling capabilities.

3) JavaScript

JavaScript is a scriptable, object-oriented, cross-platform programming language. The hosting site can connect to and manipulate JavaScript. JavaScript includes a standard library of objects such as arrays, dates, and numbers, as well as basic programming language concepts such as operators, controls, and expressions.

By adding objects, JavaScript can be extended to various principles, such as:

Client-side JavaScript: JavaScript is created using objects that control the browser and the DOM. For example, client-side extensions allow applications to interact with the content of an HTML page and respond to user actions such as mouse-overs, form inputs, and page changes.

Server-side JavaScript: JavaScript is a plugin that requires the use of JavaScript on the server side. For example, this extension of the server allows an application to connect to data, actively transfer data from one application to another part of the application, or complete the application using other files running on the server.
In 1996 JavaScript was called ECMAScript. ECMAScript 2 was released in 1998 and ECMAScript 3 in 1999. It continued to evolve into the JavaScript it is today, and is now available on all browsers and devices, from mobile to desktop. Organizations can develop their own JavaScript applications using the open-source language. The ECMAScript standard is part of the ECMA-262 specification.

4) ReactJS

Developed and maintained by Facebook, React has become a well-known and widely used JavaScript library for building user interfaces. Released in 2013, React is known for its expressive and object-oriented architecture that allows developers to create powerful and efficient web applications.
The essence of React revolves around the concept of modular, reusable building blocks that encapsulate specific functions and UI elements. This modular approach improves policy management and facilitates collaborative development. React's virtual DOM (Document Object Model) effectively manages changes to the real DOM, resulting in improved performance and performance.
One of React's greatest strengths is its reporting. The developer declares the desired UI state and React takes care of updating the DOM to match that state. This method simplifies the development process, making it easier and less error-prone.
React also popularized the concept of one-way data flow, making state management easier. React applications that maintain a one-way data flow are easy to understand and debug. State management in React is mostly implemented using “useState” hooks, allowing components to manage their internal state without relying on external libraries.
With tools like React Router to manage navigation, Redux to manage state in large scripts, and integration of the React ecosystem with various design tools like Webpack and Babel.
In addition, React's compatibility with Server-Side Rendering (SSR) and the ability to create mobile applications with React Native demonstrate its versatility. React Native is a React extension that allows developers to create different mobile applications with a beautiful interface using React design.
In short, React has greatly influenced the web development landscape, providing developers with powerful and useful tools to create powerful, responsive and controlled user interfaces. Its popularity and widespread use point to its importance as a foundational technology in modern web development.

B. Design /discussion phase

This section consists of all the features that are implemented in the visualizer.

1) For Sorting

a) User Interface

It is defined as the point of intersection between the user and a computer system. It is a view of how the website is seen by the user.

1.A. Input section

1.A.A Array input size

This feature enables the user to freely enter the size of the array which they wish to visualize. Elements of their choice can also be entered. This allows them to customize the array according to their choice which helps them to observe and understand the algorithms more clearly. There also exists a generate button which generates the elements of the array randomly.

2. Algorithm selection

The visualizer includes various sorting algorithms such as bubble sort, selection Sort, Insertion sort, Heap sort, Merge sort, Quick sort and shell sort. It allows user to choose from the above algorithms according to their preference or the algorithms they wish to understand which enhances their learning experience.

3. Speed Adjustments

In this feature user can adjust the speed of the visualizer this allows user to control the pace at which the sorting algorithms work that is two elements are compared and swapped. There is a feature of Pause and replay through which one can pause the algorithm visualization at a particular point to understand what is happening. This enables them to enhance the learning experience and understand the core concept more deeply. These features enable all the users from a beginner to an advance programmer to understand the algorithms more easily in a very interactive way.

4.Visualization Area

Real time visualization

It is a dynamic visualization area. This area enables the user to visualize the sorting algorithm in real time by using colors and animations. It provides an interactive and intuitive environment by allowing the user to witness the step-by-step execution of the sorting algorithm. Elements in an array are represented in the form of tiles which have numbers written on it entered by the user. The visualizer colors the elements which are compared at present to differentiate them from the rest. It then, based on the logic of the algorithm swaps the elements which the user can see through animations. By this the user can dive deep into the mechanics of sorting.

2) For Path Finding

This section enables the user to find the shortest path from the source to the destination.

a) User interface

Grid creation

The user has the choice to select the number of the cells in the grid that is the number of rows and the number of columns in the grid through the slider provided.

The user can now create the grid to represent the environment with the start and the end points. By selecting the entry and exit points, one can determine the source and the destination user can create obstacles in the path by selecting "wall" button. This allows the user to place obstacles on the grade for realistic scenarios.

One can also generate a random maze so as to have a deeper understanding of the concepts. It also includes an option to reset the entire maze.

Visualization Area

It is a dynamic visualization area. This area enables the user to visualize the sorting algorithm in real time by using COLORS and animations. It provides an interactive and intuitive environment by allowing the user to witness the STEP-BY-STEP execution of the sorting algorithm. Elements in an array are represented in the form of tiles which have numbers written on it entered by the user. The visualizer colors the elements which are compared at present to differentiate them from the rest. It then, based on the LOGIC OF the algorithm swaps the elements which the user can see through animations. By this the user can dive deep into the mechanics of sorting.

Visualiser performs a variety of task which makes the understanding of the algorithm much easier. Visualiser has a starting and an ending point, both of these are defined by the user. It then travels the whole field and find the shortest pathway between the start and the end this process is done through animation. Visualiser while traversing the field, changes the colour as it moves forward in search of the shortest path as so as to differentiate the path travelled with the path which is remaining.

3) For Linked List

A linked list is a data structure that stores a sequence of elements. Each element in the list is called a node, and each node has a reference to the next node in the list.

This section enables the user to understand the concept of linked thoroughly.

a) User Interface

It is defined as a section which is visible to the user on the screen on the computer system.

1.A.A User defined input

There is a dedicated section through which the input can be taken from the user for the elements to be inserted into the linked list.

1.B. Choice selection

At first, the user enters an element and thereafter given a choice to either insert the element/delete or clear the linked list.

1.C. Speed adjustment

This feature enables the user to adjust the speed of the visualiser. This allows the user to control the face at which the animation can be seen. Here the user has the freedom to pause play or go forward and backwards. This feature allows the user to understand the concept of linked list thoroughly.

b) Visualization Area

Once the element is retrieved from the user to be pushed / inserted into the link list, it is displayed as pushing value into the visualisation area. There is a node pointer named top, which does not point towards any value. The received value from the user gets inserted into the node consisting of the data and the pointer, pointing towards null. The top pointer then points towards the newly made node. This denotes the initialisation of the link list.

If the list already has some elements present in it, then the pointer of the newly made node points towards the node last inserted and the top pointer starts pointing towards the newly made node. This process can be repeated to insert new elements in the linked list in order to extend it.

In case of pop function:

The data of the node which is pointed by the top gets stored in the pop value. Stop starts pointing towards the second node of the linked list i.e. (top. next. next.). In this way the first element from the linked list is removed.

4) For Stacks

A stack can be defined as a container in which insertion and deletion can be done from the one end known as the top of the stack. It follows LIFO (last in first out) principle.

This section enables the user to understand the concept of stacks thoroughly.

a) User Interface

It is defined as a section which is visible to the user on the screen on the computer system.

A block of array with size 30 along with the variable top, is present in the visualisation area.

1.A. User defined input

There is a dedicated section through which the input can be taken from the user for the elements to be inserted into the stack.

1.B. Choice selection

At first, the user enters an element and thereafter given a choice to either push the element or pop/ clear the stack.

1.C. Speed adjustment

This feature enables the user to adjust the speed of the visualiser. This allows the user to control the face at which the animation can be seen. Here the user has the freedom to pause play or go forward and backwards. This feature allows the user to understand the concept of stacks thoroughly.

b) Visual Representation

For push ( ):

Push means to insert Elements into the stack.

User can enter the elements of their choice in the space provided. Top, in the beginning is set to zero (0). Value in the top is the index value which represents the next empty space. As soon as the user enters a value, the visualisation area stores this element in the pushing value. Then the values in the top is treated as the index on which the values should be inserted. The value is inserted at the desired index and the top is updated.

For pop ( ):

Pop means to delete the top most element from the stack. Firstly, the value of top is decremented. This decremented value is treated as the index from which the element has to be removed. This element is removed and stored into the pop value.

Clear stack enables you to delete all the values from the stack. Value of the top is updated to zero (0).

5) For Queue

A queue can be defined as a container in which insertion and deletion is done from opposite ends. It follows FIFO (first in first out) principle. This section enables the user to understand the queue of linked thoroughly.

a) User Interface

It is defined as a section which is visible to the user on the screen on the computer system. A block of array with size 15 along with the variables, Head (initial value 0) and tail (initial value 0) are present in the visualization area. Head indicates the first index of the queue and tail represents the last index of the queue.

1.A. User defined input
There is a dedicated section through which the input can be taken from the user for the elements to be inserted into the queue.

1.B. Choice selection
At first, the user enters an element and thereafter given a choice to either insert(enqueue)/ delete(dequeue)/ clear the queue.

1.C. Speed adjustment
This feature enables the user to adjust the speed of the visualiser. This allows the user to control the face at which the animation can be seen. Here the user has the freedom to pause play or go forward and backwards. This feature allows the user to understand the concept of queue thoroughly.

b) Visual representation
Enqueue ( )
Enqueue means to insert elements into the queue. User can enter the element of their choice in the space provided. For enqueue operations, the value on the tail denotes the index where the next element is to be inserted. The given value is inserted on the index denoted by the tail and the tail variable is incremented. The process is repeated for inserting other elements as well.

Dequeue ( )
It means deleting the elements from the queue.
As queues follow First in First Out, the head variable helps us to remove the elements from the beginning. The head denotes the index from which the element is to be removed as it indicated the starting point of the queue. So, the index denotes by the head is accessed, then the value at that index is stored in the dequeued value and head is incremented to indicate the new beginning of the queue. Clear queue enables to clear the whole queue and sets both head and tail to zero (0).

6) For Trees
Tree data structure is a hierarchical structure that is used to represent and organize data in a way that is easy to navigate and search. This section enables the user to understand the concept of trees thoroughly.
a) User Interface
It is defined as a section which is visible to the user on the screen on the computer system.
User defined input
There is a dedicated section through which the input can be taken from the user for the elements to be inserted/ deleted or found into the tree.

b) Visual representation
- *Insert*

User enters the value to be Inserted. If the entered element is the first node, it becomes the root of the tree.
For a tree having a single node, the element inserted entered is compared to and whether it is smaller or greater than the root element. These comparisons are made recursively within the subtree and the nodes are placed in the correct positions.
There is also a guide text which tells how the comparisons are made and elements are given their correct position.

- *Delete*

The entered element is first searched into the tree through comparisons. When the element is found, the node is deleted and tree is adjusted accordingly.

- *Find*

The entered element is searched based on the comparisons. If the element is found, the text ' element found ' is displayed. If the element is not present, the text is displayed so.

- *Print*

It prints the element of the tree.

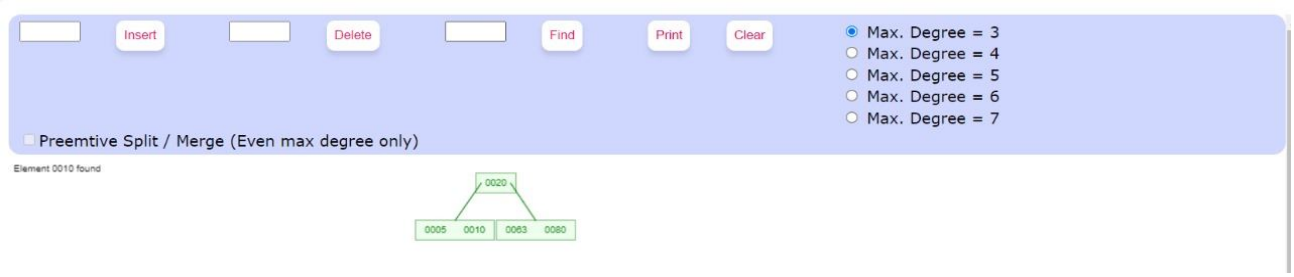
V. OBJECTIVES

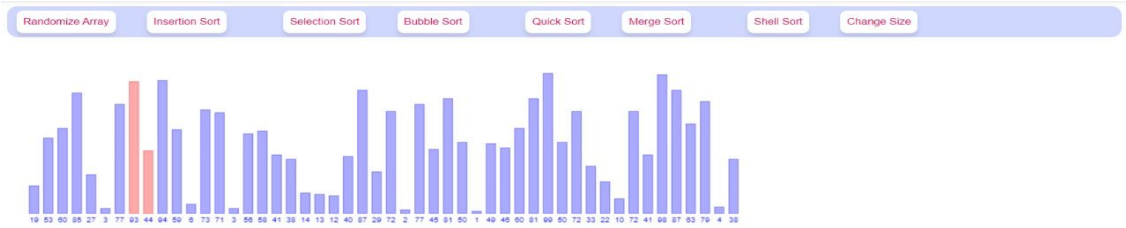
1) *E learning Tool:* To understand Algorithms better.
2) *Educational Enhancement:* To assist in teaching and learning algorithms by providing visual representation.

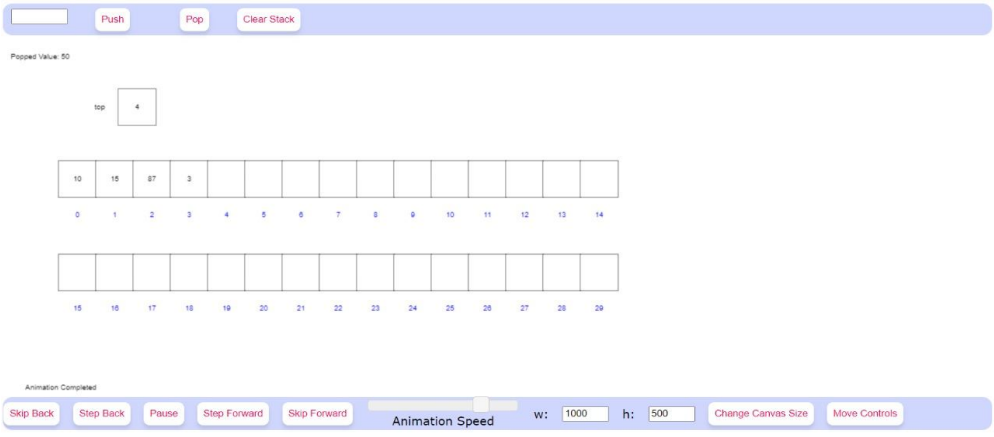
Algorithm Comparison and Understanding

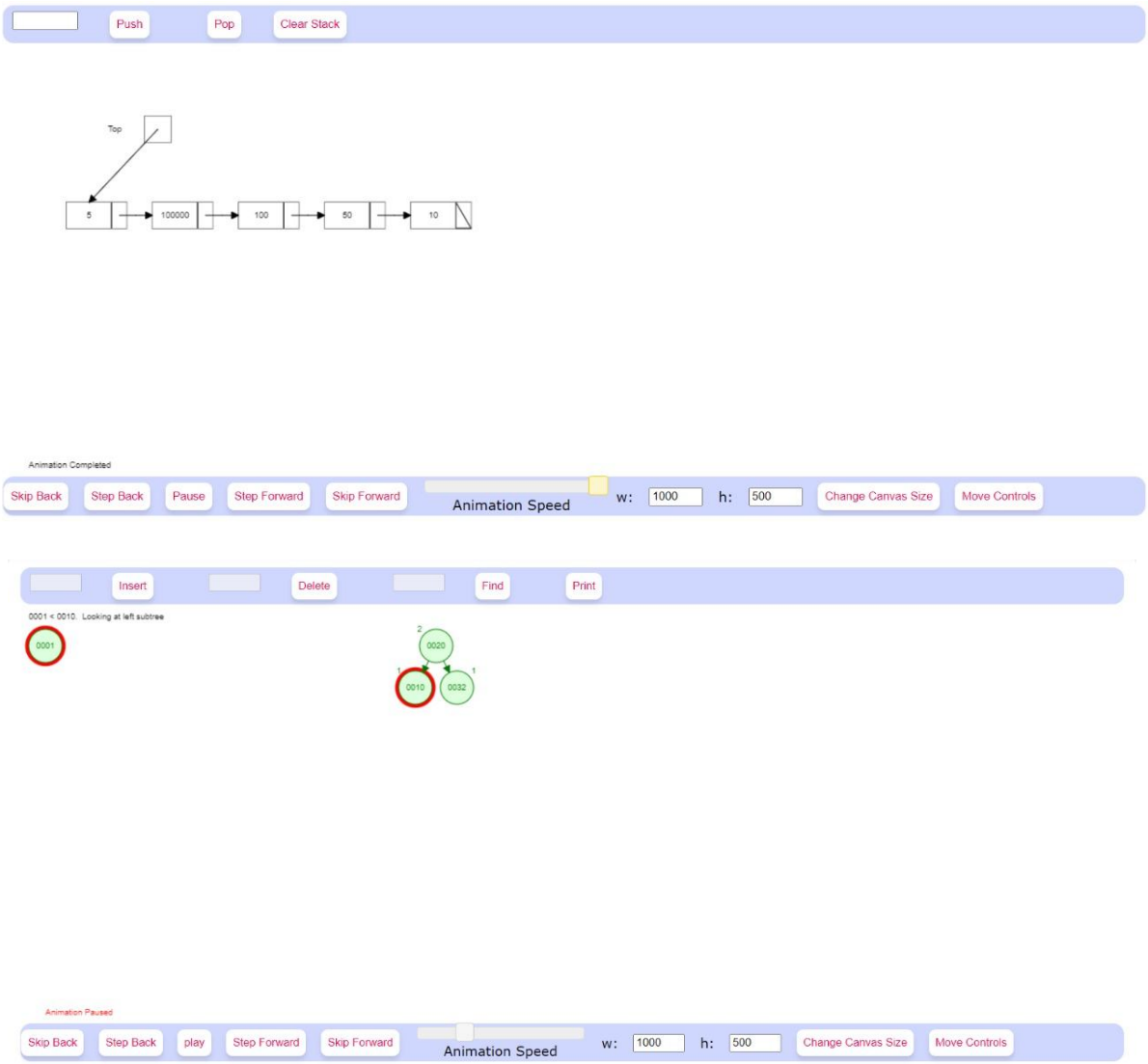
a) *Debugging Tool:* To help developers debug and optimize algorithms by visualizing each step of execution.
b) *User Engagement:* To make learning algorithms more engaging and interactive.
c) *Interactive Learning:* Many algorithm visualizers offer interactive features, allowing users to modify inputs and see how these changes affect the algorithm's performance and outcome
d) *IP Routing:* To find Open shortest Path First. We can make a GPS system which will guide you to the locales.
e) *Engagement and Motivation:* Visual tools can make learning more engaging and less intimidating, potentially increasing the motivation of learners to delve deeper into algorithmic concepts.

VI. RESULT









VII. FUTURE SCOPE

1) *Education and Training:* The project can be utilized in educational area to help students understand and analyses sorting and path-finding algorithms better.
2) *Software Development and Optimization:* The project can be used in development phase of a software. Developers can choose most appropriate sorting and path-finding algorithm.
3) *Game Development:* Developers can understand and implement path-finding algorithms for non-player character in games.
4) *Logistics and Routing:* Path-finding visualizations can be used in transportation system to find shortest route for deliveries.

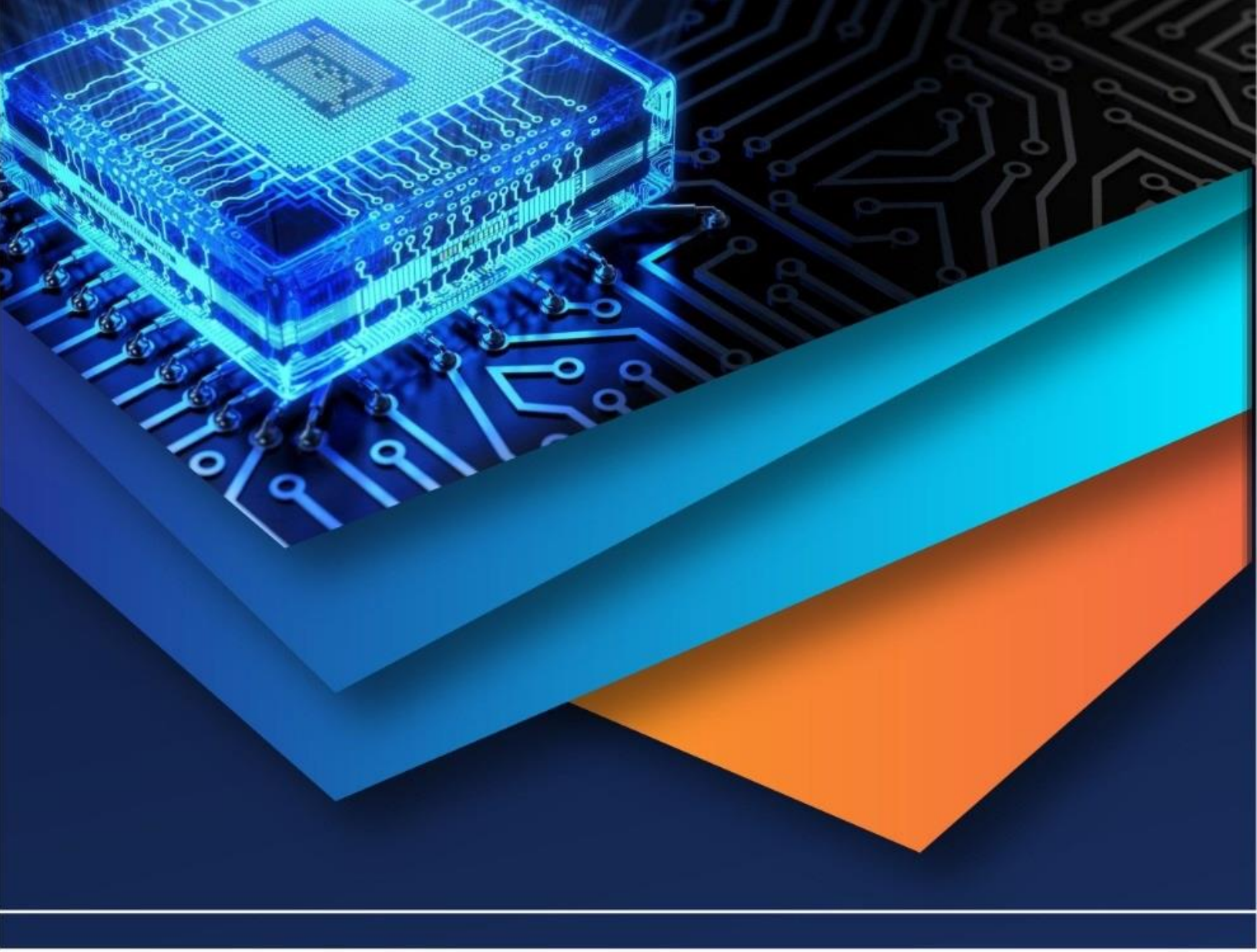
VIII. CONCLUSION

In the previous sections, it is seen that data structures and algorithms play an important role in computer science. Students face difficulties in learning and understanding algorithms with traditional methods.

We aim to create an interactive and easy-to-use platform for students in order to make learning a fun and enjoyable experience using visuals. Sorting Visualizer and Pathfinder is broadly divided into two parts: Sorting Algorithms Visualizer and Shortest Pathfinder. In Sorting Algorithm Visualizer, one can generate a random array with a random or they can enter a new array. Additional information such as time used, number of comparisons and number of swaps is also generated with the sorted array. In Shortest Pathfinder, a random can be generated as well as a desired maze can also be generated. Length of shortest path along with the path itself is displayed. Using this platform, we want to make learning a much simpler task for students as well as for teachers.

REFERENCES

[1] "Visualizing sorting algorithms" by Brian J. Faria
[2] "Algorithm visualizer: its features and working " by B. Goswami, A Dhar, A. Gupta and A. Gupta
[3] "Algorithm visualizer application " by Aditya, Shipra Srivastav, Gulshan Gupta, Bilal Ibrahim and Jatin Kumar
[4] "AlgoRhythm: A sorting and pathfinding visualizer tool to improve existing algorithms teaching methodologies" by A. Trivedi, K. Pandey, V. Gupta and M. K. Jha
[5] "AlgoAssist: algorithm visualizer and coding platform for classroom learning" by A. B. Ghandage, B. P. Udhane, H. R. Yadav, P. S. Thakre, V. G. Kottawar and P. B. Deshmukh.
[6] " REVIEW PAPER ON ALGORITHM VISUALIZER" by Rohit L. Ugile, Ritesh B. Barure, Gajanan S. Sawant
[7] K. Becker and M. Beacham, "A tool for teaching advanced data structures to computer science students: an overview of the BDP system", Journal of Computing Sciences, vol. 16, no. 2, pp. 65-71, 2001
[8] A. Dix, J. Finlay, G. D. Abowd and R. Beale, Human-Computer Interaction, 3. Edition, Ed., Harlow: Pearson Education, 2004



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY