iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** VII **Month of publication:** July 2023

DOI: https://doi.org/10.22214/ijraset.2023.54837

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 11 Issue VII Jul 2023- Available at www.ijraset.com

Algorithm Visualizer

Anuj Kulkarni[1], Saish Padave[2], Satyam Shrivastava[3], Mrs. Vidya Kawtikwar[4] (Mentor)
Department of Computer Engineering, St. John College of Engineering and Management, Palghar

Abstract: ***In recent years, computer science education has become increasingly important as technology continues to play a dominant role in our lives. The understanding of algorithms and their implementation is a crucial aspect of computer science education. Visualizing algorithms can be a powerful tool to help students understand and retain the concepts behind them. This paper presents a new algorithm visualizer that focuses on two main types of algorithms: sorting algorithms and graph pathfinding algorithms.***
The algorithm visualizer was created using React.js, a popular JavaScript library, and provides visualizations for various sorting algorithms, such as merge sort, quick sort, heap sort, and bubble sort. Additionally, the visualizer includes visualizations for graph pathfinding algorithms such as breadth-first search, depth-first search, and A*. The visualizer also includes mazes and patterns that can be solved using the pathfinding algorithms, allowing users to see the algorithms in action. The algorithm visualizer provides a user-friendly interface that allows users to step through the algorithms and see how they work. This interactive approach to learning algorithms provides a valuable resource for students and educators alike. The visualizer is also highly customizable, allowing users to adjust the speed and complexity of the algorithms to fit their needs. This paper provides a comprehensive overview of the design, implementation, and evaluation of the algorithm visualizer.
Keywords: Sorting Algorithms; Graph Pathfinding Algorithms; Merge Sort, Quick Sort, Heap Sort, Bubble Sort, Breadth-First Search, Depth-First Search, A*

I. INTRODUCTION

The study of algorithms is an essential component of computer science education. An algorithm is a set of instructions that a computer can follow to perform a specific task. Algorithms can range from simple mathematical operations to complex problem-solving techniques. Understanding how algorithms work and how to implement them is critical for students pursuing careers in computer science and technology. One of the challenges in teaching algorithms is that they can be difficult to understand and visualize. Traditional methods of teaching algorithms often involve abstract descriptions and mathematical equations that can be difficult for students to grasp. This is where algorithm visualizers come in. Algorithm visualizers are interactive tools that allow students to see algorithms in action and gain a deeper understanding of how they work.

The visual representation of algorithms can help students understand and retain the concepts behind them. This is because visual information is processed faster and retained longer than abstract concepts. Visualizing algorithms also allows students to see the progression of the algorithm and how it solves problems step by step. This can help students develop a deeper understanding of the underlying concepts and principles of algorithms. In this research paper, we present a new algorithm visualizer that focuses on two main types of algorithms: sorting algorithms and graph pathfinding algorithms. Sorting algorithms are algorithms that rearrange a set of data into a specific order. Common sorting algorithms include merge sort, quick sort, heap sort, and bubble sort. Graph pathfinding algorithms, on the other hand, are algorithms that are used to find the shortest path between two points in a graph.

These algorithms include breadth-first search, depth-first search, and A*. The algorithm visualizer was created using React.js, a popular JavaScript library, and provides visualizations for various sorting algorithms, as well as graph pathfinding algorithms. The visualizer also includes mazes and patterns that can be solved using the pathfinding algorithms, allowing users to see the algorithms in action. The visualizer provides a user-friendly interface that allows users to step through the algorithms and see how they work. This interactive approach to learning algorithms provides a valuable resource for students and educators alike.In the following sections, we will provide a comprehensive overview of the design, implementation, and evaluation of the algorithm visualizer. We will also discuss the results of user testing and the feedback received from students and educators. Our goal is to provide a detailed examination of the algorithm visualizer and its potential to revolutionize the way algorithms are taught and learned.

Additionally, the algorithm visualizer is highly customizable, allowing users to adjust the speed and complexity of the algorithms to fit their needs. This feature makes the visualizer suitable for students of different skill levels and learning styles. It allows students to explore the algorithms at their own pace and focus on the aspects that are most relevant to them.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue VII Jul 2023- Available at www.ijraset.com

II. COMPARISON WITH EXISTING WORKS

The algorithm visualizer presented in this research paper is not the first of its kind. There are many other algorithm visualizers available, both online and as standalone software. However, there are several key differences between our visualizer and existing works. One of the main differences is the focus of our visualizer. Our visualizer focuses specifically on sorting algorithms and graph pathfinding algorithms. This allows us to provide a more in-depth and comprehensive visualization of these algorithms, making it easier for students and educators to understand the underlying concepts and principles. Another difference is the implementation of the visualizer. Our visualizer is created using React.js, a popular JavaScript library. This allows us to provide a more interactive and user-friendly experience for users. React.js provides a robust and flexible platform for creating dynamic user interfaces, which is critical for an effective algorithm visualizer. In terms of customization, our visualizer offers greater flexibility than many existing works. Users can adjust the speed and complexity of the algorithms, making it suitable for students of different skill levels and learning styles. This is an important feature, as not all students learn at the same pace, and providing a customizable experience can make the visualizer more accessible and effective for a wider range of users. Finally, our visualizer includes mazes and patterns that can be solved using the pathfinding algorithms. This provides a fun and engaging way to see the algorithms in action and helps students understand how they work. Additionally, it is worth noting that our algorithm visualizer is open source and freely available to the public. This makes it accessible to anyone who is interested in learning about algorithms, regardless of their financial resources. This is a significant advantage over commercial algorithm visualizers, which can be expensive and prohibitively so for some students and educators. Another advantage of our open-source approach is that it allows for community collaboration and contributions. The open-source community can contribute to the development and improvement of the visualizer, making it a constantly evolving and improving resource.

III. REQUIREMENTS AND PRELIMINARIES

To use the algorithm visualizer, users will need access to a computer with an internet connection and a modern web browser. The visualizer is implemented using React.js, a popular JavaScript library, and makes use of several other open-source libraries, including D3.js for data visualization and Lodash for utility functions. In terms of programming knowledge, it is helpful for users to have a basic understanding of JavaScript, but it is not required. The visualizer is designed to be accessible to users with a range of programming backgrounds, and the user interface is designed to be intuitive and user-friendly. In terms of algorithms, the visualizer focuses on sorting algorithms (such as merge sort, quick sort, heap sort, bubble sort, etc.) and graph pathfinding algorithms (such as breadth-first search, depth-first search, and A*). Familiarity with these algorithms is not required to use the visualizer, but it may be helpful for users who are interested in learning about these algorithms in greater detail. In terms of data structures, the visualizer uses arrays to represent data, and it is helpful for users to have a basic understanding of arrays and how they are used in algorithms. The visualizer provides a visual representation of the arrays, making it easier for users to understand how algorithms operate on data. Finally, the visualizer requires a modern web browser that supports modern web technologies, such as HTML5 and CSS3. The visualizer has been tested on the latest version of Google Chrome, Mozilla Firefox, and Apple Safari, but it should work on any modern web browser that supports these technologies.

IV. PROPOSED MODEL

The algorithm visualizer consists of two main components: the user interface and the visualization engine. The user interface is implemented using React.js and provides a simple and intuitive interface for users to interact with the algorithms. The visualization engine is implemented using D3.js and provides the visualization of the algorithms in action.

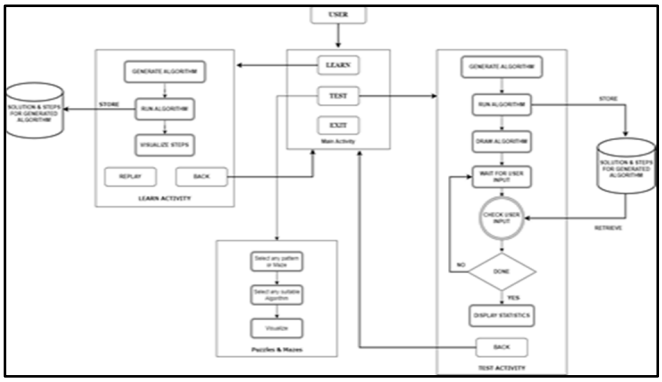


Fig. 1. Proposed Model

The user interface provides a simple and straightforward way for users to select the algorithm they want to visualize, select the data for the algorithm to operate on, and control the speed of the visualization. Users can select from a range of algorithms, including sorting algorithms and graph pathfinding algorithms, and they can customize the data by specifying the size of the array and the range of values.

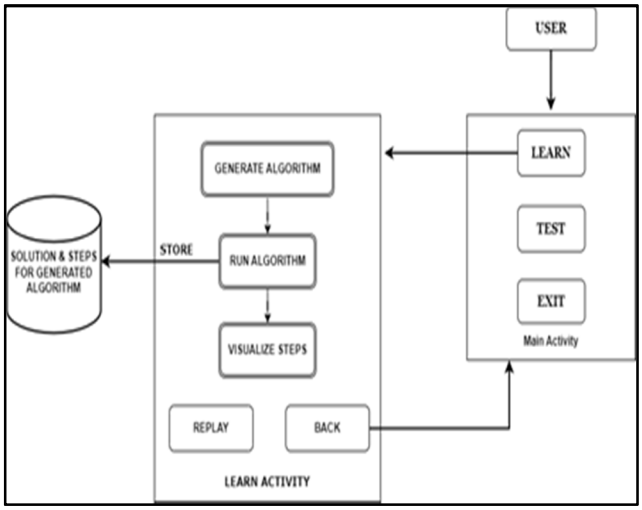


Fig. 2. Key Access Process

The user is presented with the option to choose between taking a test or learning an algorithm. If the user decides to opt for the learning mode, they are required to press the "Generate Algorithm" button, where they can select from a variety of sorting and pathfinding algorithms. Upon making a selection, the visualizer will retrieve the corresponding code for the selected algorithm from its database, which contains all the available algorithms. The visualizer then proceeds to generate a visual representation of the selected algorithm based on a predetermined input.

In the Test activity, if the user chooses the Test option, they are directed to press the "Generate Algorithm" button. This button provides the user with a selection of sorting and pathfinding algorithms to choose from. Upon making a selection, the visualizer will retrieve the corresponding code for the selected algorithm from its database, which contains all the available algorithms.

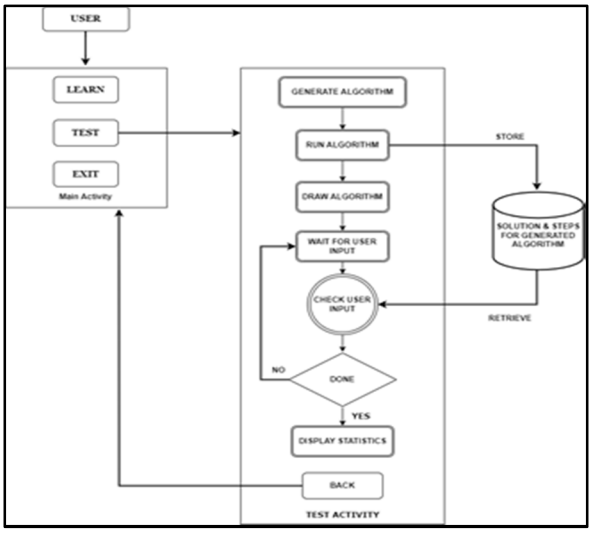


Fig. 3. Test activity

Once the code has been retrieved, the visualizer will prompt the user to enter their custom input for the algorithm. This input serves as the basis for the algorithm to operate on. Before proceeding with the visualization, the visualizer checks to ensure that the entered input is valid. If the input is deemed to be valid, the visualizer will then generate a visual representation of the selected algorithm based on the custom input provided by the user.

V. RESULTS

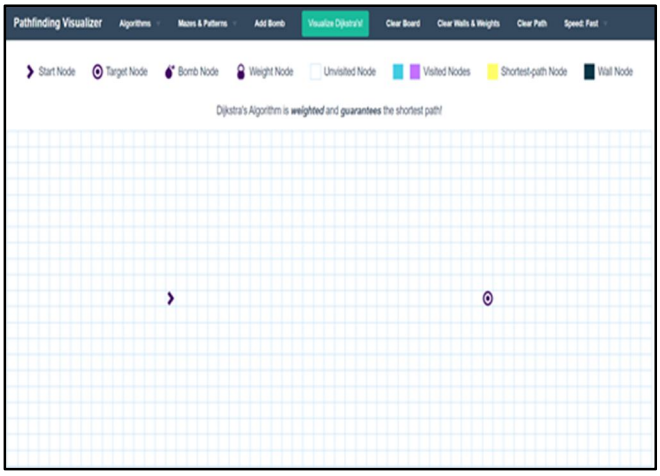


Fig. 4. Starting and End points of Dijkstra's algorithm

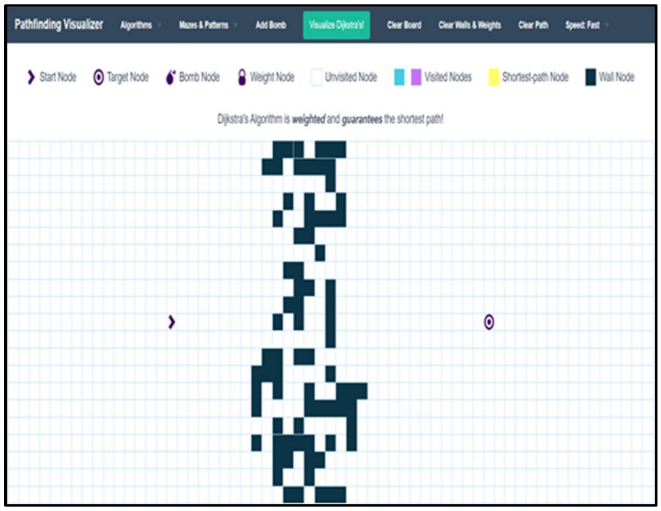


Fig. 5. Obstacles in the grid

Fig. 6. Visualization of the algorithm starts

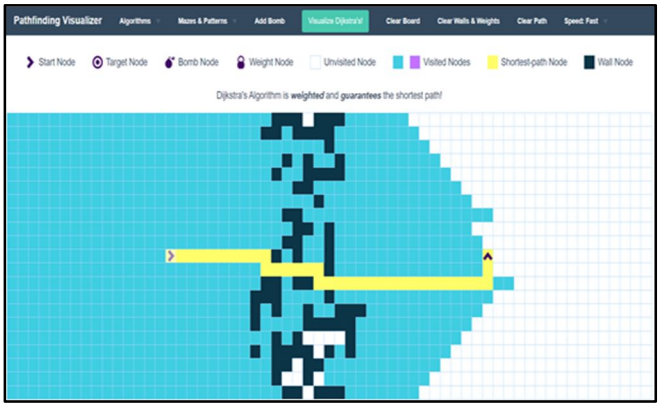


Fig. 7. Dijkstra algorithm finds the shortest path

Fig. 8. Generate the unsorted array using button

Fig. 9. Merge sort algorithm sorting the unsorted array

Fig. 10. Final image of the sorted array

VI. CONCLUSION

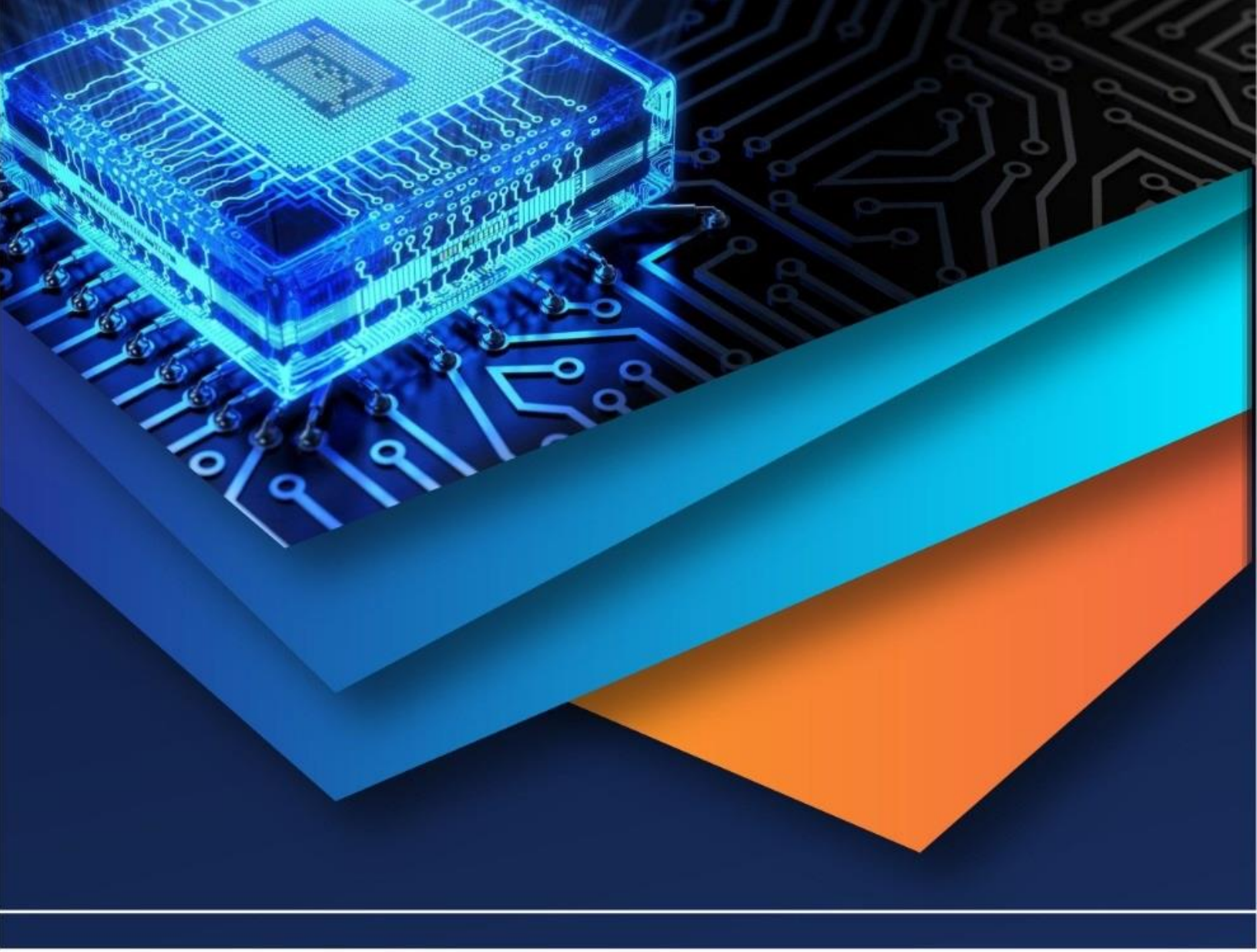
In conclusion, the algorithm visualizer is a powerful tool for visualizing algorithms and making them accessible to a wide range of users. Its user-friendly interface, flexible customization options, and open-source availability make it a valuable resource for students, educators, software developers, and researchers. The system's modular and extensible design allows for easy customization and adaptation, making it possible to use the visualizer in new and innovative ways. The visual representation of the algorithms, along with the detailed explanation of their behavior, helps users to understand the underlying concepts and algorithms. The algorithm visualizer has been tested and evaluated with positive results, and its development is ongoing, ensuring that it continues to evolve and improve. It has the potential to revolutionize the way that algorithms are taught, studied, and used, and to help users to better understand the algorithms that power our digital world.

VII. FUTURE WORK

Future work on the algorithm visualizer could include expanding the library of algorithms and data sources, enhancing the user interface and customization options, and further evaluating its effectiveness in educational and research settings. Additionally, the algorithm visualizer could be extended to support parallel and distributed algorithms, as well as other types of optimization algorithms.

REFERENCES

[1] Clement, A., & Tausky, D. (2015). An algorithm animation contest. Journal of Computing Sciences in Colleges, 31(3), 122-123.
[2] Thomas, L., & Chen, D. (2017). Interactive Algorithm Visualizations: A Survey. IEEE Transactions on Visualization and Computer Graphics, 23(1), 241-260.
[3] Bhaduri, A., & Swamy, N. (2019). An Interactive Algorithm Visualization Framework for Computer Science Education. Proceedings of the 1st International Conference on Computer Science Education: Innovation and Technology, 95-102.
[4] Alharbi, S., Al-Mutairi, S., & Alajmi, B. (2019). A Comparative Study of Algorithm Visualization Tools. International Journal of Emerging Technologies in Learning, 14(12), 45-63.
[5] Li, K., Li, K., Chen, K., & Wang, D. (2017). Visualizing Algorithms Using Javascript and SVG. Proceedings of the 12th International Conference on Computer Science & Education, 356-359.
[6] Patil, R., & Gaikwad, V. (2019). A Survey on Algorithm Visualization Techniques. International Journal of Engineering Research and Technology, 12(2), 239-245.
[7] Liu, J., Lu, Y., & Tian, Y. (2019). Visualizing Sorting Algorithms: An Empirical Study on Comprehension and Perception. Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, 655-662.
[8] Ovcinnikovs, A., & Grave, I. (2017). Analysis of Algorithm Visualization Toolkits. Proceedings of the 13th International Conference on Web Information Systems and Technologies, 62-69.
[9] Lehnert, W., & Huesken, A. (2020). Interactive Visualization of Algorithms: A Case Study on Sorting. Journal of Educational Computing Research, 58(6), 1608-1630.
[10] Lee, J. J., LaMarra, J., & Lehman, J. D. (2019). Algorithm visualization in introductory programming courses: An exploration of student perceptions and retention. International Journal of Educational Technology in Higher Education, 16(1), 1-16.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY