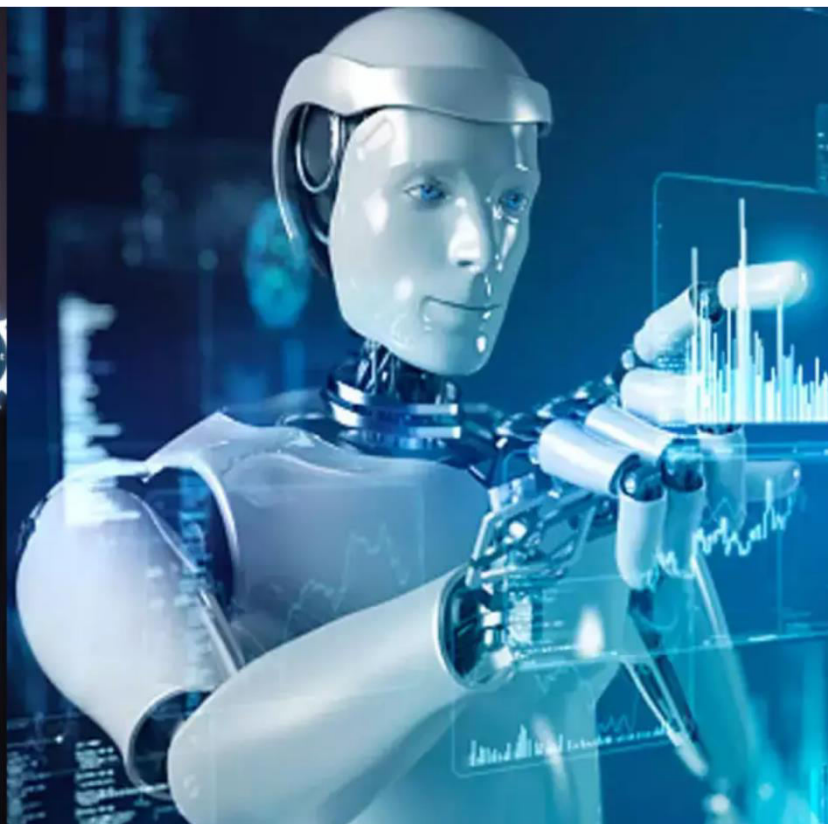




International Journal of Innovative Research in Science Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.699

Volume 14, Issue 4, April 2025

Algorithm Visualizer

Riya Gole ¹, Sanidhya Jambhulkar ², Labana Mohit Singh³

Engineering Student, Department of Computer Engineering, K.C. College of Engineering & Management Studies & Research, Thane, Maharashtra, India¹

Engineering Student, Department of Computer Engineering, K.C. College of Engineering & Management Studies & Research, Thane, Maharashtra, India²

Engineering Student, Department of Computer Engineering, K.C. College of Engineering & Management Studies & Research, Thane, Maharashtra, India³

ABSTRACT: “In the field of computer science and education, understanding algorithms is often challenging due to their abstract nature. This project introduces an Algorithm Visualizer, an interactive web-based tool designed to bridge the gap between theoretical concepts and practical understanding. The tool provides real-time graphical representations of various algorithms, including pathfinding algorithms like A* Search and Dijkstra's Algorithm, as well as backtracking problems such as the N-Queens problem. Through dynamic animations and user interaction, the visualizer allows users to observe the step-by-step execution of algorithms, aiding in deeper comprehension and analysis. It serves as both an educational resource for students and a debugging tool for developers. The application is built using modern web technologies such as JavaScript, HTML, and CSS to ensure responsiveness and accessibility across devices. By transforming complex code into visual narratives, this Algorithm Visualizer enhances algorithm literacy and promotes hands-on learning.

I. INTRODUCTION

Understanding the inner workings of algorithms is a crucial aspect of learning computer science, but it can often be challenging due to their abstract and logical complexity. This project presents an Algorithm Visualizer—a dynamic, interactive web-based tool designed to visually demonstrate how different algorithms operate step by step. The visualizer includes a diverse set of algorithms, such as pathfinding algorithms like A* Search, backtracking algorithms like the N-Queens problem, and sorting algorithms such as Bubble Sort, Insertion Sort, and more. By transforming lines of code into animated visual processes, this tool enables users to observe how data changes in real-time as each algorithm executes. This approach fosters a deeper understanding of concepts such as recursion, iteration, decision trees, and time complexity. This platform is fully interactive and accessible directly through the browser, requiring no additional software. It supports user interaction, allowing customization of input data, speed, and algorithm selection, making it a flexible and engaging learning tool.

II. MINI-PROJECT PLAN

1. Project Overview
2. Project Timeline: Duration: 10 weeks
3. Key Milestones:
 - a. Week 1 -2: Planning and Research
 - Conduct a detailed literature review to understand existing systems and identify unique features.
 - Define the specific requirements and functionalities.
 - Create a project plan outlining tasks.
 - b. Week 3-4: Design and Prototyping
 - Design the user interface (UI) for the website, emphasizing simplicity and ease of use.
 - Finalize the UI design based on user input.

c. Week 5-6: Frontend Development

- Begin frontend development, focusing on creating a responsive and intuitive interface.
- Conduct iterative testing to ensure a smooth user experience.

e. Week 7-8 : Testing and Quality Assurance

- Conduct comprehensive testing, including functionality, usability, and performance testing.
- Address any bugs or issues identified during testing.
- Optimize performance for a smooth user experience.

f. Week 9-10: Deployment and Launch**III. PROBLEM STATEMENT & OBJECTIVE**

- Understanding how algorithms work can be challenging for beginners due to their abstract nature and lack of visual representation.
- The objective of an Algorithm Visualizer is to simplify this learning process by providing interactive, step-by-step visual demonstrations of various algorithms.
- Making it easier for users to understand their logic, flow, and performance through real-time animations and user inputs.

IV. METHODOLOGY AND IMPLEMENTATION

Many students and beginners in programming face difficulties in understanding the working of different algorithms due to their abstract and complex nature.

They often struggle to visualize how data changes at each step and how different algorithms operate internally. Therefore, to assist learners, we are developing an Algorithm Visualizer that will provide clear, interactive, and step-by-step visual representations of various algorithms.

By using our platform, users will be able to better understand the logic, flow, and efficiency of algorithms, making their learning process easier and more effective.

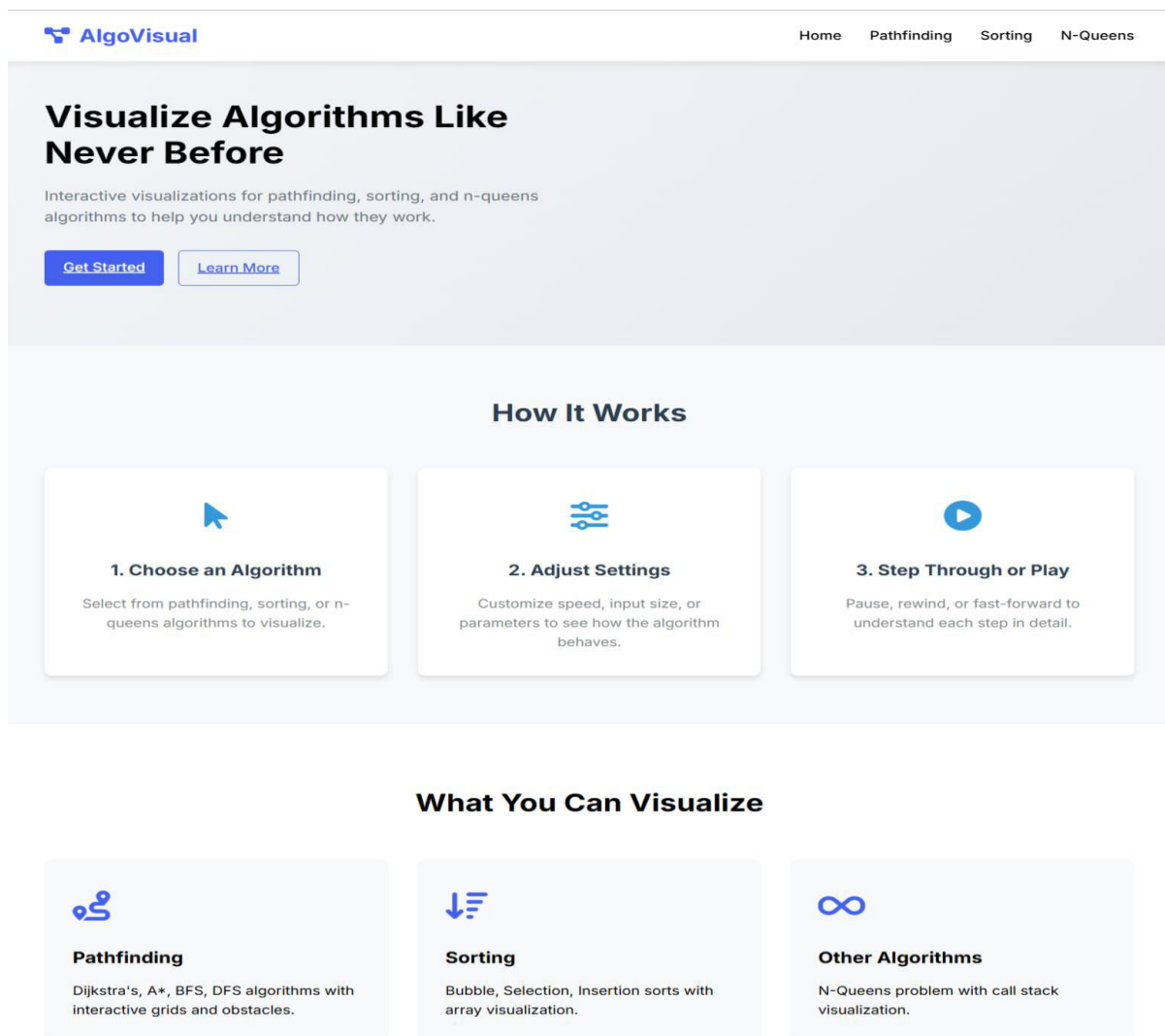
2. Hardware & Software requirements :**Software Requirements**

- For Users (Running the Visualization)
 - Web Browser (any modern browser):
 - Google Chrome (v90+)
 - Mozilla Firefox (v85+)
 - Microsoft Edge (v90+)
 - Safari (v14+)
 - Operating System:
 - Windows 10/11
 - macOS 10.15+
 - Linux (Ubuntu 20.04+, Fedora 34+, etc.)
 - ChromeOS (for Chromebooks)
 - Code Editor:
 - VS Code (Recommended)
 - Sublime Text / Atom / WebStorm
 - Local Web Server (Optional):
 - Live Server (VS Code Extension)
 - python -m http.server

- Accessibility Considerations:
 - Works on mobile/tablet (responsive design)
 - No heavy computations (runs smoothly on low-end devices)
 - No backend/server needed (static HTML/CSS/JS)

Hardware Requirements:

- System:- Intel Core i3 (Min)
- SSD:- 256GB
- Monitor:- Standard LED Monitor
- Input Devices:- Keyboard
- Ram:- 8 Gb and above
- Processor:- x32 bit, x64b

Experimental Output

The screenshot displays the AlgoVisual website interface. At the top, the logo 'AlgoVisual' is on the left, and navigation links 'Home', 'Pathfinding', 'Sorting', and 'N-Queens' are on the right. The main heading is 'Visualize Algorithms Like Never Before', followed by a subtext: 'Interactive visualizations for pathfinding, sorting, and n-queens algorithms to help you understand how they work.' Below this are two buttons: 'Get Started' and 'Learn More'. The 'How It Works' section contains three steps: 1. Choose an Algorithm (with a cursor icon), 2. Adjust Settings (with a settings icon), and 3. Step Through or Play (with a play button icon). Each step has a brief description. The 'What You Can Visualize' section lists three categories: Pathfinding (Dijkstra's, A*, BFS, DFS), Sorting (Bubble, Selection, Insertion), and Other Algorithms (N-Queens problem).

HOME PAGE

Pathfinding Algorithms

Controls

Algorithm:

Dijkstra's

Speed:



Start

Reset

Draw:

Wall

Start

End

Algorithm Details

Dijkstra's algorithm finds the shortest path between nodes in a graph with non-negative edge weights.

Time Complexity

Best Case

Average Case

Worst Case

$O(E + V \log V)$

$O(E + V \log V)$

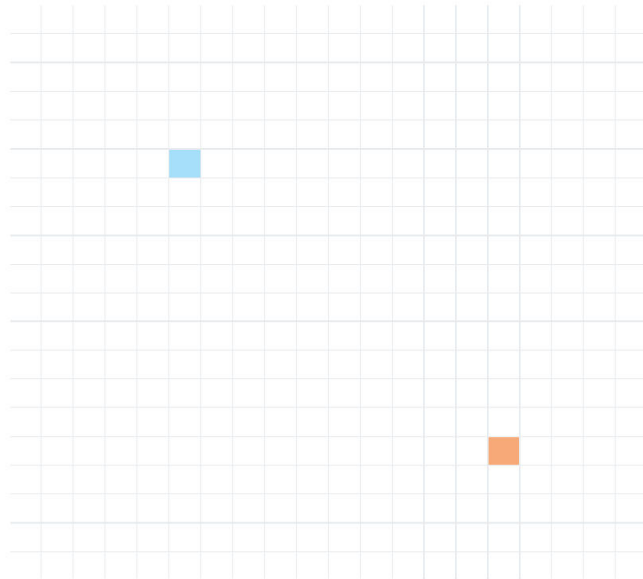
$O(E + V \log V)$

Space Complexity

$O(V)$

Algorithm Steps

1. Initialize distance values for all nodes (INF for all except start node = 0)
2. Create a priority queue with all nodes
3. While queue is not empty, extract node with minimum distance
4. For each neighbor, update distances if shorter path found
5. When end node is reached, reconstruct the path



PATHFINDING ALGORITHM

Sorting Algorithms

Controls

Algorithm:

Bubble Sort

Array Size:



Speed:



Start

Reset

Shuffle

Algorithm Details

Bubble sort repeatedly steps through the list, compares adjacent elements and swaps them if they are in the wrong order.

Time Complexity

Best Case

Average Case

Worst Case

$O(n)$

$O(n^2)$

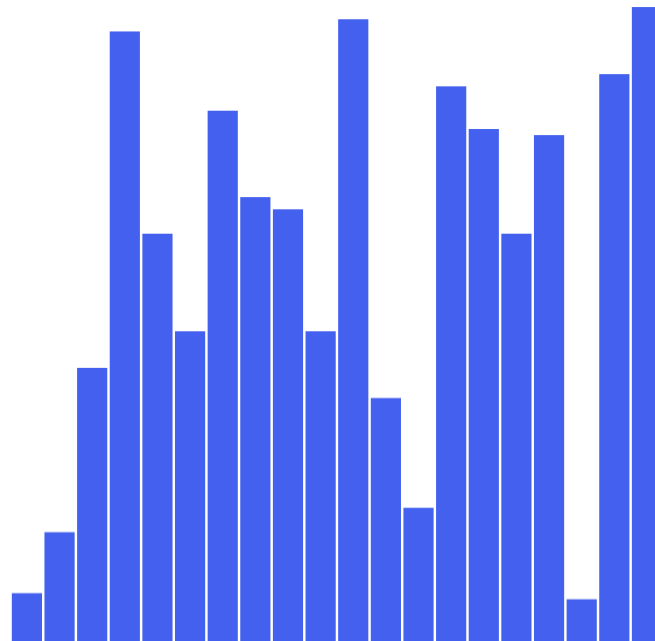
$O(n^2)$

Space Complexity

$O(1)$

Algorithm Steps

1. Start with first element
2. Compare with next element
3. Swap if in wrong order
4. Move to next element
5. Repeat until no swaps needed



SORTING ALGORITHM

N-Queens Problem

Controls

Board Size:



Speed:



Pause

Step

Solution found!

Start

Reset

Algorithm Details

The N-Queens problem places N chess queens on an N×N chessboard so that no two queens threaten each other.

Time Complexity

Best Case

Average Case

Worst Case

O(N!)

O(N!)

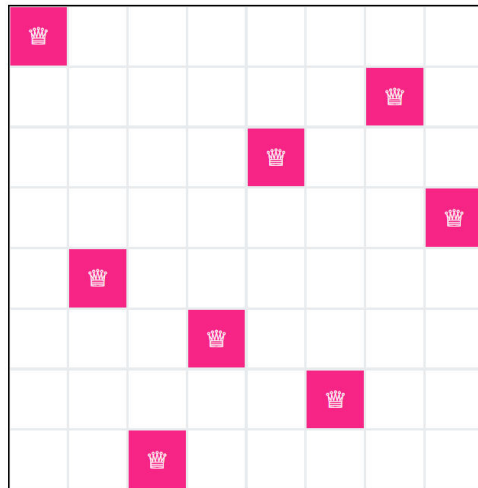
O(N!)

Space Complexity

O(N²)

Algorithm Steps

1. Place a queen in the first column
2. Move to next column and place queen safely
3. If no safe position, backtrack
4. Repeat until all queens placed



AlgoVisual

Interactive algorithm visualizer to help you understand how algorithms work through visualization.

Quick Links

[Home](#)

[Pathfinding](#)

[Sorting](#)

Contact

Email: info@algovisual.com



© 2025 AlgoVisual. All rights reserved.

N- QUEENS PROBLEM

V. CONCLUSION & FUTURE SCOPE**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.

Future Scope:

Implement algorithm complexity analysis, displaying time and space complexity in real-time as the algorithm runs.

Add code-view integration, showing the pseudocode or actual code running alongside the visualization.

Provide multi-language support for algorithm explanations (e.g. Hindi, Spanish).

Allow user-defined algorithms, where users can input their own logic and visualize its execution.

Integrate performance comparison tools to compare execution time and efficiency of different algorithms.

Develop a mobile-friendly version of the visualizer for better accessibility on smartphones and tablets.

Enable user accounts and progress tracking, useful for students learning through modules or tutorials.

Gamify learning with quizzes and challenges, where users solve algorithm-based puzzles interactively.

REFERENCES

1. 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.
2. E. Vrachnos and A. Jimoyiannis, "Design and evaluation of a web-based dynamic algorithm visualization environment for novices," *Procedia Computer Science*, vol. 27, pp. 229-239, 2014.
3. F. E. A. M and S. C, "The role of visualization in computer science education," *Computers in the Schools*, vol. 29, pp. 95-117, 2012.
4. D. Hundhausen, S. Douglas and J. Stasko, "A metastudy of algorithm visualization effectiveness," *Journal of Visual Languages and Computing*, vol. 3, no. 3, pp. 259-290, 2002.
5. G. Robling and T. L. Naps, "A testbed for pedagogical requirements in algorithm visualizations," in *Conference on Innovation and Technology in Computer Science Education*, New York, USA, 2002.
6. J. Urquiza-Fuentes and J. A. Velazquez-Iturbide, "A survey of successful evaluations of program visualization and algorithm animation systems," *ACM Transactions on Computing Education*, vol. 9, no. 2, pp. 1-24, 2009.
7. Bremananth R, Radhika V Thenmozhi S "Visualizing Sequence of Algorithms for Searching and Sorting" 2009 International Conference on Advances in Recent Technologies in Communication and Computing 2009
8. T. L. Naps, G. Robling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger and J. A. Velazquez-Iturbide, "Exploring the role of visualization and engagement in computer science education," in *Proceedings of ITiCSE on Innovation and Technology in Computer Science Education*, New York, USA, 2002.
9. A. Dix, J. Finlay, G. D. Abowd and R. Beale, *Human-Computer Interaction*, 3. Edition, Ed., Harlow: Pearson Education, 2004.



INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

 9940 572 462  6381 907 438  ijirset@gmail.com



www.ijirset.com

Scan to save the contact details