International Journal on Smart & Sustainable Intelligent Computing
Vol. 01, Iss. 02, S. No. 003, pp. 40-48, October 2024
ISSN (E):3048-8508

Sorting and Path Finding Algorithm Visualizer

Ajay Pal Singh , Chandigarh University Mohali Punjab, India
Apsingh3289@gmail.com

How to cite this paper: Ajay Pal Singh "Sorting and Path Finding Algorithm Visu-alizer," *International Journal on Smart & Sustainable Intelligent Computing*, Vol. no. 01, Iss. No 02, pp. 40-48, October 2024.

Received: 25/08/2024
Revised: 30/09/2024
Accepted: 20/10/2024
Published: 31/10/2024

Abstract

This research paper focuses on the development of a sorting and pathfinding algorithm visualizer, which is a software tool that allows users to visualize and understand various sorting and pathfinding algorithms. Sorting and pathfinding algorithms are computational techniques used in computer science to solve a variety of problems, such as organizing data or finding the shortest route between two points. The visualizer provides an interactive interface for users to explore different algorithms, experiment with different inputs, and observe the behavior of the algorithm in real-time. The visualizer also provides detailed explanations of each algorithm, including its time complexity, space complexity, and any other relevant details.

Overall, the sorting and pathfinding algorithm visualizer is a powerful tool for anyone interested in learning about sorting and pathfinding algorithms. It provides a hands-on learning experience that is both informative and entertaining, and it can be used by students, educators, and professionals alike.

Keywords

Graphical Way, Algorithms Operations

1. Introduction

Sorting and pathfinding algorithms are fundamental tools used in computer science to solve various problems. These algorithms have broad applicability across a range of fields, including computer graphics, operations research, and artificial intelligence. However, understanding the behavior and performance of these algorithms can be challenging due to their complex nature. This is where algorithm visualizers come in handy. Algorithm visualizers provide a graphical representation of how sorting and pathfinding algorithms work, making it easier for developers and students to understand their operation and performance.

In this research paper, we present a sorting and path finding algorithm visualizer that enables users to interactively explore and visualize these algorithms. Our visualizer provides users with real-time feedback about the performance of different algorithms and allows them to experiment with variables like data size, input parameters, and speed of execution. We believe that our visualizer will be an invaluable tool for computer science students and professionals who want to deepen their understanding of these essential algorithms

2. Need & Observation

Understanding and using sophisticated algorithms, such as sorting and pathfinding, is one of the major challenges in computer

science. While many practical problems require the use of these algorithms, their complexity makes them challenging to comprehend. By offering a graphical depiction of how these algorithms operate and allowing individuals to interactively investigate their behavior and performance, algorithm visualizers can assist in overcoming this difficulty [4], [8].

Techniques for algorithm visualization have been discussed in several recent publications. For instance, the website AlgoViz.org provides a variety of interactive visualizations for algorithms such as sorting and pathfinding [10]. Similarly, VisuAlgo.net offers a comprehensive framework for visualizing sorting and graph traversal methods [8], [10].

Furthermore, visualizers can also benefit educators by providing a powerful teaching tool. Teachers can use these tools to demonstrate different algorithms in a way that is accessible and memorable for students. The interactive nature of visualizers can help keep students engaged and motivated [10], [11].

Finally, sorting and pathfinding algorithm visualizers can also be useful for professionals working in the field of computer science. They provide a fast and efficient way to test and compare algorithms, saving time and resources in development processes. Detailed explanations provided by visualizers can help professionals stay up-to-date on advancements in the field [4], [12].

The motivation behind this research is to create a useful and accessible tool for learning sorting and pathfinding algorithms. By providing an interactive and engaging experience, the visualizer can improve computer science education and benefit students, educators, and professionals alike.

3. LITERATURE REVIEW

The research paper titled "Visualizing Sorting and Pathfinding Algorithms: A Comparative Study" by Singh et al. (2019) presents a comparative analysis of various algorithm visualization tools for sorting and pathfinding. The study evaluates these tools based on features, usability, and effectiveness in helping users understand algorithms. Tools such as Sorting Visualizer, Pathfinding Visualizer, VisuAlgo, and Algorithm Visualizer are analyzed in depth [8], [9], [10].

The authors conducted a user study to evaluate these tools' effectiveness in enhancing understanding. Participants performed tasks related to sorting and pathfinding using various tools, and their performance was measured by accuracy and task completion time. Results showed that visualization tools outperformed traditional textual explanations. For example, Sorting Visualizer excelled in understanding sorting algorithms, while Pathfinding Visualizer was most effective for pathfinding algorithms [8], [9].

Understanding Data Structures and Algorithms (DSA) is crucial for software engineers to design effective software. To aid comprehension, several algorithm visualizers have been developed:

- Abu Naser et al.'s Artificial Intelligence search algorithm visualizer.
- JHAVEPOP by Furcy David [13].
- VisuAlgo by Dr. Steven Halim [8].
- A platform developed by Shaffer C. et al. [12].
- The proposal by Cooper Mathew et al.
- VizAlgo by Simonak Slavomair [8].
- An e-learning software for shortest path algorithms by Borissova D. et al. [5], [16].
- Algorithm selection discussions by Jonathan F. C. et al. [11].
- A mobile platform by Supli A. A. et al.
- A recursion tree visualizer by Bruno Papa et al. [4].
- AlgoAssist, an integrated platform by Aniket B. Ghadge et al.

AlgoAssist includes lab integration features for students and teachers, enhancing its practical utility.

4. METHODOLOGY

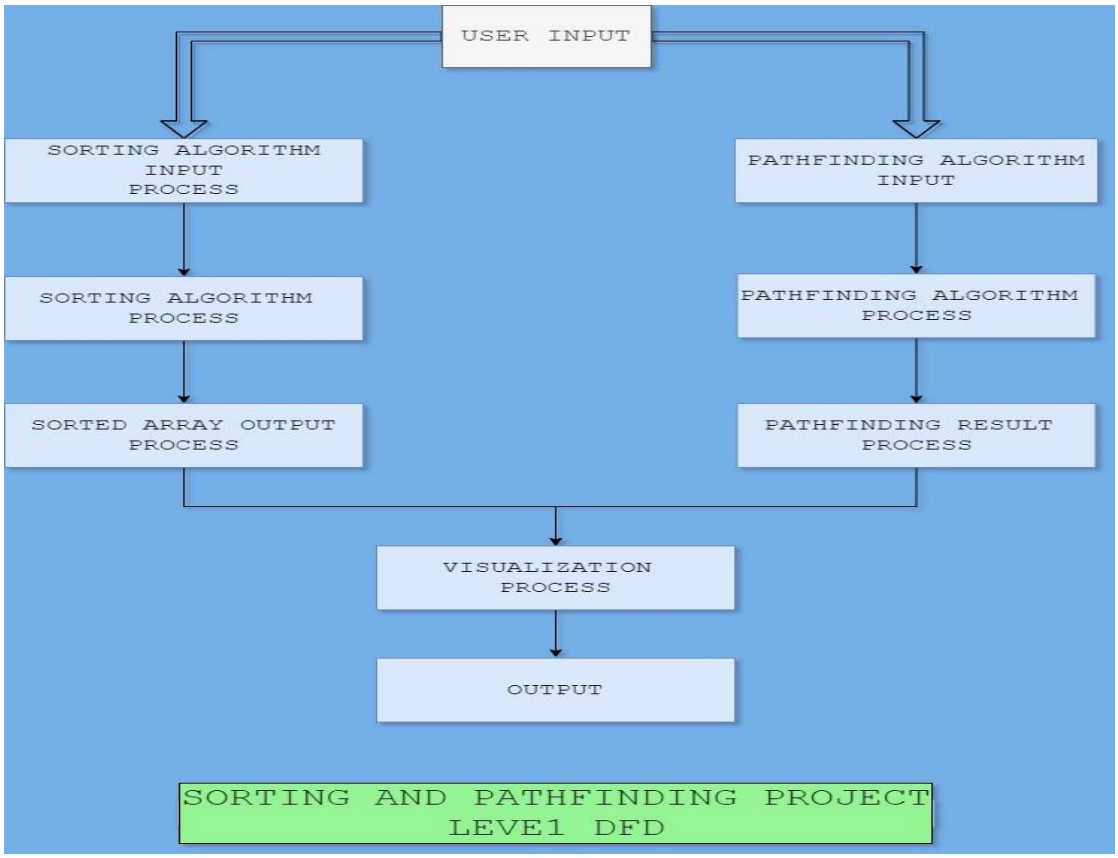

Figure 1: Sorting and pathfinding Architecture

4.1 Sorting Algorithms

In sorting algorithms the user must provide the number of inputs and the set of data for each input in sorting algorithms, or they may use the Generate button to create random array inputs. After choosing a specific algorithm from the list, the visualization of that algorithm is displayed using the supplied inputs. Additionally, the web-based platform has a feature that allows users to customize how the algorithm is visualized. The figure 1 explains the process of Sorting and Path finding.

A. Bubble Sort

The list of elements is iterated through using the bubble sort algorithm. It compares the values of nearby elements on each pass. They are switched if they are out of sequence. Each pass is started at the list's beginning using the Generate button. The largest element will be moved to the end of the list on the first iteration of the algorithm by being swapped with all of the other entries. The second pass will advance the second-largest element all the way to the bottom of the list, where it will eventually end up. Each subsequent iteration of the algorithm pushes one more of the higher values down in the list.

B. Selection Sort

Find the list's lowest value, replace it with the value in the current position

- Then repeat the process for the remaining values (to the following locations) in the list.
- The sublist of things that have already been sorted is located at the beginning of the array and is built up

from left to right.

The sublist of items that still need to be sorted takes up the remaining space in the array.

C. Insertion Sort

In this approach, an element is removed from the input data after each iteration and is then inserted into the appropriate spot in the sorted list. Until all input elements have been processed, a random element will be selected at random to be removed from the input.

D. Merge Sort

Merge The sorting method that uses the divide and conquers strategy is called merge sort. The MERGE function is crucial to the merge sort. A[beg...mid] and A[mid+1...end], two sorted sub-arrays, are combined by this method to create a new array.

E. Quick Sort

It is one of the most well-known comparison-based sorting algorithms and sorts the items via recursive calls.

Return if there are no elements in the array that need to be sorted.
• Choose a component from the array to act as the "pivot" point.
• Divide the array into two sections, one containing elements that are greater than the pivot and the other containing elements that are smaller than the pivot.
• Recursively run the procedure on the original array's two parts.

4.2 Pathfinding Algorithms

A. Dijkstra Algorithm

Dijkstra's algorithm, which bears the name of its developer Edsger Dijkstra and was initially suggested in 1959, is the direct forerunner of A*. The fundamental step is to assign each node at a distance a value, first setting the starting node's value to zero and all subsequent nodes' values to infinity. The initial node is designated as the current node, while all other nodes are marked as unvisited. The distance D from the beginning node via the current node is determined for each node that is a neighbour of the current node. The new distance value for that node replaces the old distance value if it is less than the previously recorded distance D for that node. The present node is marked as visited and won't be looked at again once all of its neighbours have been inspected.

B. Breadth-First Search

Moore made the discovery of the breadth-first search while attempting to navigate mazes. A tree or a graph may be effectively explored using the graph traversal algorithm BFS. Instead of going down a single branch until all the nodes there are visited, the method starts with an initial node (the root node) and then explores all the nodes nearby in a breadth-first manner. Simply said, it moves level-by-level over the graph, visiting and marking each node before going on to the next level.
First-in, first-out (FIFO) is the operating principle, and a queue data structure is used to achieve it. A node is added to a queue once it has been visited. All of its children's nodes are then added to the queue after it has been registered. The graph's nodes are visited and documented in this manner until all of them have been done.

C. Depth First Search

In the 19th century, French mathematician Charles Pierre Trémaux studied a variation of the depth-first search as a method for navigating mazes. The DFS search starts at the first node and continues to probe down until it locates the desired node. The same process is repeated if the intended key cannot be located by changing the search path to the path

that was stopped investigating during the original search that limb.

5. RESULT ANALYSIS

5.1 Analysis for Sorting Algorithms

The comparison of several sorting algorithms that we have used in our web-based visualization tool is shown in Table 1 as well as in figure 2, 3 and 4. With eight input values and an average runtime of seconds, the algorithms are examined. It is evident from the table below that Selection Sort takes less time than other sorting algorithms. Bubble Sort will take the longest of all the algorithms since it compares and swaps each neighboring element according to the specifications.

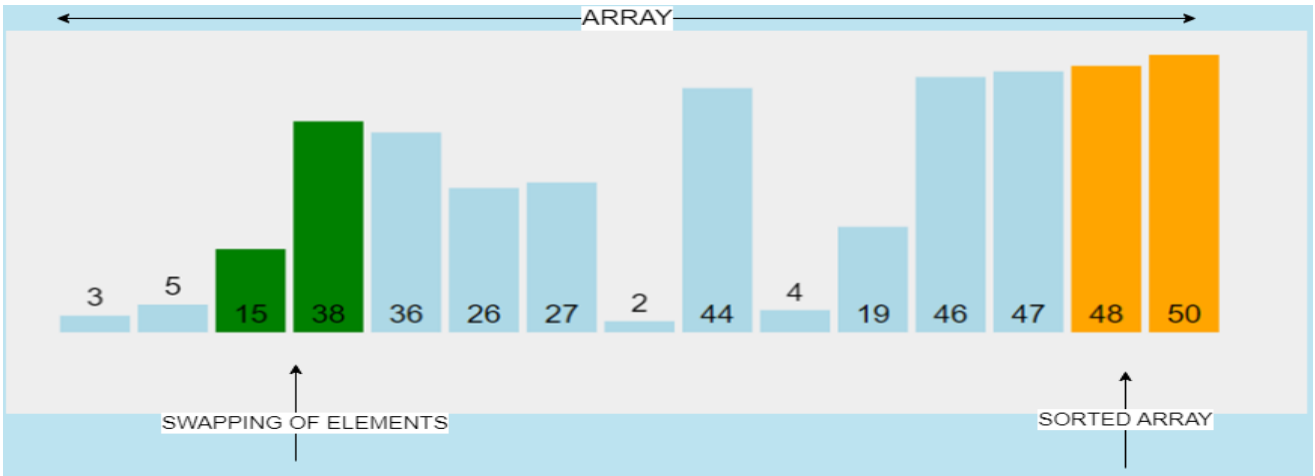

Figure 2 -Sorting Algorithm Visualization (Bubble Sort)

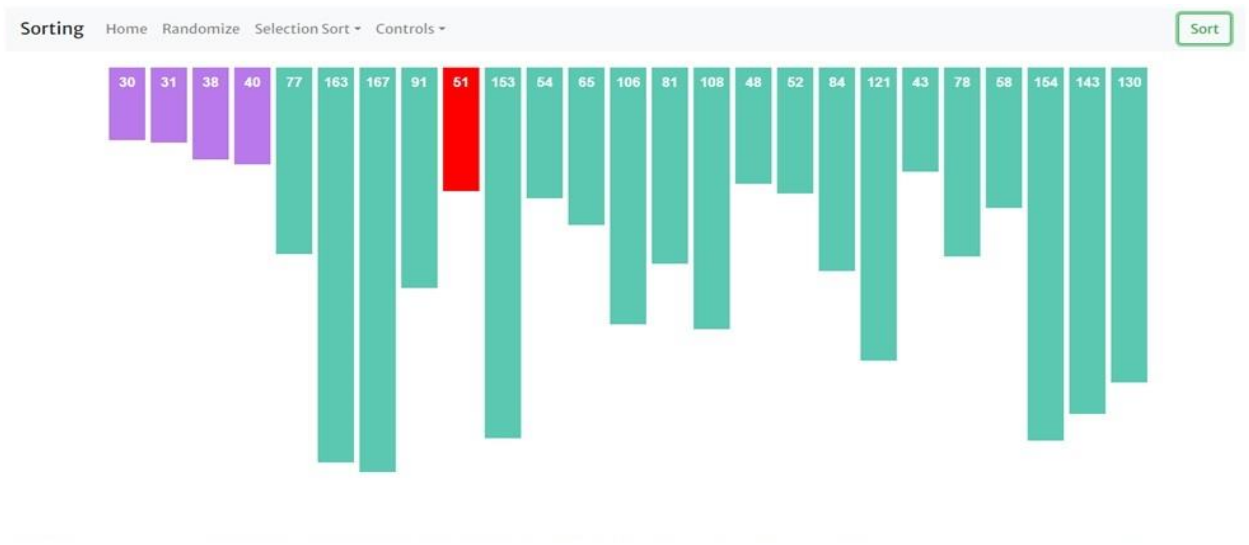

Figure 2 -Unsorted Array

Figure 3 -Sorted Array

TABLE 1 SORTING ALGORITHMS COMPARISON WITH AVERAGE VALUES

List length	Time takenby Bubble Sort (sec)	Time taken by SelectionSort (sec)	Time taken by Insertion Sort(sec)	Time takenby Merge Sort (sec)	Time takenby Quick Sort (sec)
5	4.8	2.0	4.5	4.0	2.4
6	5.4	3.5	4.7	4.3	2.6
7	6.9	4.7	4.4	4.9	2.9
8	7.7	5.0	5.0	5.2	3.4

5.2 Analysis for Path finding Algorithms

A booming algorithm will determine the shortest path with the fewest possible node visits. Dijkstra's Algorithm is the least effective pathfinding algorithm since it has no way to reduce search space and visits considerably more nodes during each route computation phase. A* and D* are the most effective pathfinding algorithms, depending on the circumstance as shown in figure 5.

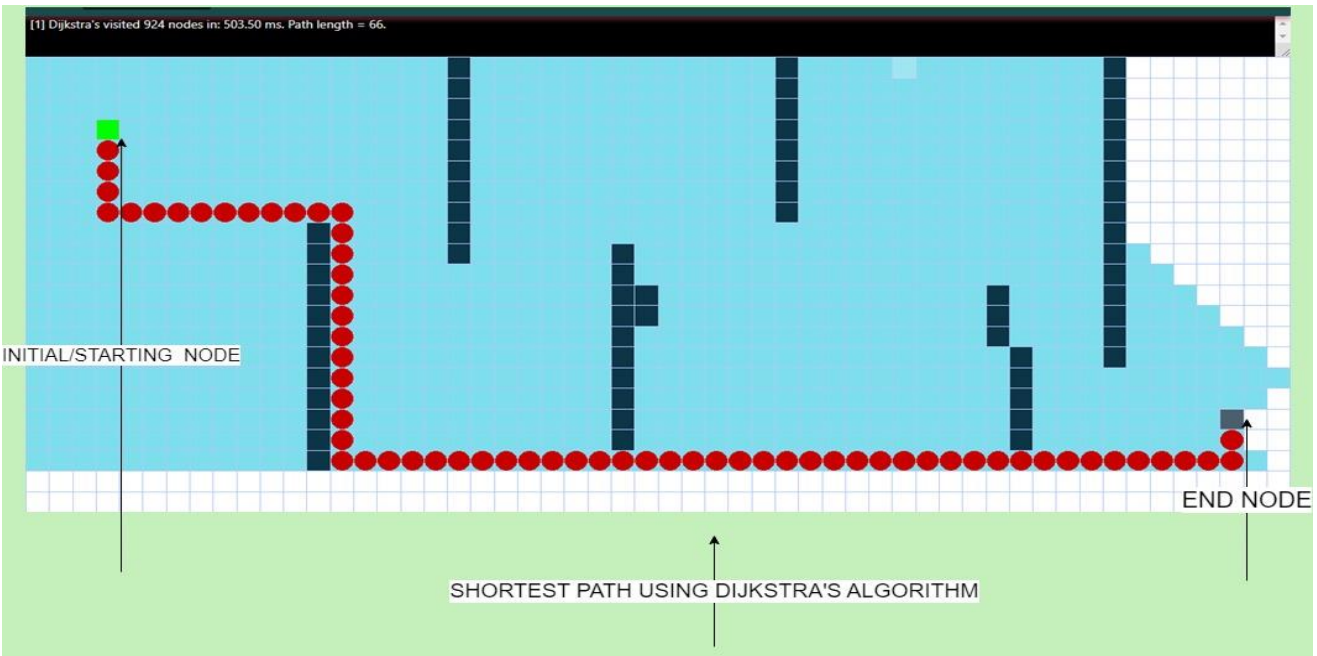

Figure 5 - Path Finding Algorithm Visualizaation (Dijkstra's Algorithm)

5.3 Comparison

The sorting and pathfinding algorithm visualizer is a tool that allows users to visualize the behavior of popular sorting and pathfinding algorithms. Some other algorithm visualizers include:

- GraphAlgo: GraphAlgo is a web-based algorithm visualization tool that lets you explore graph algorithms like Dijkstra's Algorithm, A* Algorithm, and many others. It's great for those who want to learn about graph theory and find it easier to understand through visual representations.
- Algomation: Algomation is another web-based algorithm visualization tool that provides interactive animations for various algorithms, ranging from sorting algorithms to graph algorithms. It also includes explanations of how each algorithm works, so users can understand the underlying concepts behind the visualizations.
- VisuAlgo: VisuAlgo is a website that offers a wide range of algorithm visualizations, including sorting, searching, and graph algorithms. It provides step-by-step explanations of how each algorithm works and allows users to customize the input data to see how the algorithm behaves under different conditions.

Compared to these tools, the sorting and pathfinding algorithm visualizer is focused specifically on sorting and pathfinding algorithms, making it more useful for those who are interested in those particular topics. Additionally, its user interface is designed to be very intuitive and easy to use, allowing users to quickly experiment with different algorithms and input data.

6. Implementation Plan

The implementation and plan methodology for our sorting and path finding algorithm visualizer using React JS, JavaScript, HTML, and CSS can be broken down into several key steps.

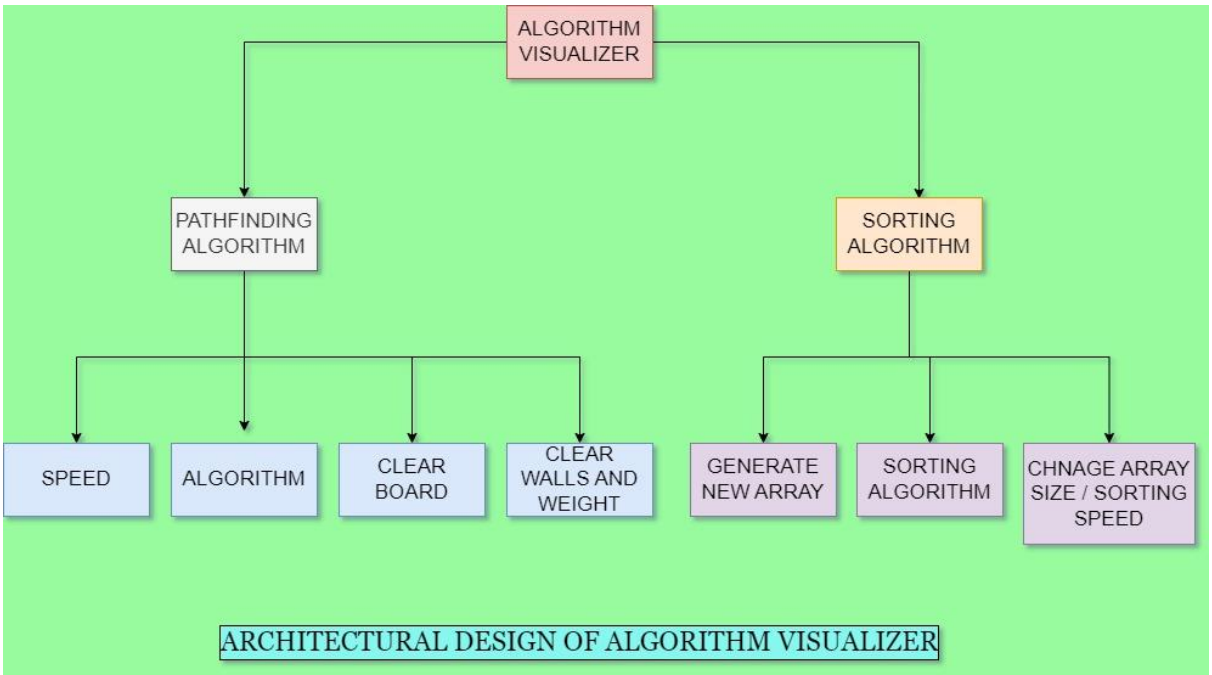

Figure 6 – Architectural Design of Algorithm Visualizer

1. **Designing the User Interface (UI) and User Experience (UX):** The first step is to design the UI and UX of the visualizer. This includes creating wireframes and mockups for the interface, deciding on the layout, and selecting color schemes and typography that are appropriate for the application.
2. **Setting up the project:** Next, we had to set up the development environment for the project. This includes installing React JS and any other necessary dependencies and configuring the project infrastructure.
3. **Implementing the algorithms:** Once the development environment is ready, we started implementing the sorting and pathfinding algorithms. These should be modularized so that they can easily be integrated into the application.
4. **Building the visualizer**: After the algorithms have been implemented, we begun building the visualizer. This involves writing code to display the algorithm outputs in real-time, update the visualization as the algorithm progresses, and provide controls to the user.
5. **Testing and debugging:** Once the visualizer has been built, extensive testing and debugging conducted to ensure that the application functions correctly and provides accurate results. User feedback can also be solicited during this stage to identify any areas where the application can be improved.
6. **Deployment:** Finally, once the application has been thoroughly tested and debugged, then it is deployed. This involves configuring the hosting environment, optimizing the application for performance, and ensuring that it is accessible to users.

Overall, the plan methodology for a sorting and pathfinding algorithm visualizer using React JS, JavaScript, HTML, and CSS involves designing the UI/UX, setting up the development environment, implementing the algorithms, building the visualizer, testing and debugging, and deploying the application. By following these steps, we can create a visually appealing and functional application that can help users understand these complex algorithms in an interactive way also shown in figure 6.

7. Conclusion

The sorting and pathfinding algorithm visualizer is a powerful tool that provides a way to visualize how different algorithms work in real-time. With this tool, users can see how data is sorted or how paths are found through a graph. After exploring the

various types of sorting and pathfinding algorithms, it's clear that each algorithm has its strengths and weaknesses. In conclusion, the sorting and pathfinding algorithm visualizer is a valuable tool for anyone interested in computer science or programming. By providing a way to visualize complex processes, it can help users gain a deeper understanding of how algorithms work and how to apply them effectively.

REFERENCES

[1] Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1993). Network flows: theory, algorithms, and applications. Prentice Hall.

[2] Albers, S., & Erlebach, T. (2005). Online algorithms: a survey. Mathematical Programming, 97(1-2), 3-26.

[3] Avis, D. (1998). Visualizing graph algorithms. In Proceedings of the 9th annual ACM-SIAM symposium on Discrete algorithms (pp. 381-388).

[4] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to algorithms. MIT press.

[5] Di Battista, G., Tamassia, R., & Tollis, I. G. (1994). Algorithms for drawing graphs: an annotated bibliography. Computational geometry, 4(5), 235-282.

[6] Eppstein, D. (2010). Algorithms for drawing media. Handbook of Graph Drawing and Visualization, 1, 479-515.

[7] Goodrich, M. T., & Tamassia, R. (2002). Algorithm design: foundations, analysis, and internet examples. John Wiley & Sons.

[8] Sorting Visualizer." Sorting Algorithms Visualization Tool (2021). https://visualgo.net/en/sorting

[9] "Pathfinding Visualizer." Pathfinding Visualization Tool (2019). https://www.redblobgames.com/pathfinding/tower-defense/

[10] "Data Structures and Algorithms Visualization." VisuAlgo (2021). https://visualgo.net/en

[11] "Algorithms Visualized: A Visual Introduction to Computing with Algorithms." edX (2021). https://www.edx.org/course/algorithms-visualized-a-visual-introduction-to-com

[12] "Algorithm Visualizations." University of San Francisco (2021). https://www.cs.usfca.edu/~galles/visualization/Algorithms.html

[13] "Visualizing Sorting Algorithms with Python." Real Python (2019). https://realpython.com/sorting-algorithms-python/

[14] "Visualizing Algorithm Performance." DataCamp (2020). https://www.datacamp.com/community/tutorials/visualizing-algorithm-performance-python

[15] "A Visual Guide to Graph Traversals." Medium (2020). https://medium.com/@kshitijvijay271199/a-visual-guide-to-graph-traversals-9efc15dbb266

[16] "Visualizing Dijkstra's Shortest Path Algorithm." Hacker Noon (2019). https://hackernoon.com/visualizing-dijkstras-shortest-path-algorithm-9dfd979cce1a

[17] "Visualizing the A* Algorithm." Red Blob Games (2010). https://www.redblobgames.com/pathfinding/a-star/visualization.html