International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 03 | March - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

SortPath Visual Studio

Aaryan Agrahari[1], Pratik Dawkhar[2], Harsh Jadhav[3] , Hamza Sayyad[4], Prof. Deepa Athawale[5]

[1,2,3,4]B.E student Department of Computer Engineering Bharat College of Engineering, Badlapur
[5] Professor, Department of Computer Engineering, Bharat College of Engineering, Badlapur, Thane, Maharashtra - 421503

Abstract -

"SortPath Visual Studio" is an interactive algorithm visualization platform designed to help students to understand complex sorting and pathfinding algorithms through real-time visual representation. The system features two user roles: Teacher and Student. Students can explore sorting and pathfinding visualizers, each supporting multiple algorithms. Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort, and Heap Sort are few of the features of the sorting visualizer offering input size, speed, and node visualisation choices. Pathfinding visualiser offers BFS, DFS, A*, and Dijkstra's Algorithm, where users can simulate real-life navigation challenges by inputting start node and end node, obstacles, and grid topologies. The key feature of this system is randomised generation, where students can try out different input sets, for example, random tile/line generation in sorting and random maze/wall generation in pathfinding. Instructors can track student progress, check assignment submissions, and evaluate learning outcomes. This platform bridges the gap between theory and practice, making it an essential tool for students, educators, and algorithm enthusiasts to deepen their understanding of fundamental algorithmic concepts. Theoretical usability of the given e-studying program is made evident through visual aid technology. The initial test findings indicate that the e- studying instrument is usable and can potentially assist college students to build effective mental models for quickest path algorithms.

Key Words: Visualization of Sorting Algorithms: Bubble Sort, Selection Sort, , Bubble Sort, Insertion Sort, Merge Sort, Heap Sort. and Pathfinding Algorithms: Dijkstra's algorithm, BFS, DFS, A*

1. INTRODUCTION

It is a fundamental requirement for data science professionals and programmers to learn sorting and pathfinding algorithms. However, the students cannot. Interactive learning materials are not present. The traditional approach of studying the theory of algorithms and tracking it manually might be cumbersome and time-consuming.

SortPath Visual Studio addresses this problem by building an interactive visualization environment from which users can watch algorithms run and thereby better understand abstract logic and have a more enjoyable experience. Visualization enables students to gain an intuitive feel for algorithm running and behavior. SortPath Visual Studio is an interactive web-based application that assists users
learn sorting and pathfinding algorithms through graphical simulations. The software is a learning and testing program, via which you will be able to view how different algorithms are executed step by step. The purpose of this project is primarily to develop knowledge about algorithms by providing an interactive and entertaining interface. The system can handle two visualizations:

1. Sorting Visualizer: You can play with algorithms such as Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort, and Heap Sort with parameters such as array size, speed, and randomness. A sorting algorithm is an algorithm that puts the elements of a list in a specific order; the order may be increasing or decreasing. Up to now many sorting algorithm has been found. Some of them were comparison based sort such as insertion sort, selection sort, bubble sort, quick sort and merge sort while others were non comparison based sort. When we are trying to sort any kind of list, arrays etc. first we compare element with each other then swap or copy those elements if needed. It keeps going again and again until the entire array or list is sorted. These algorithms are called Comparison based sorting.

2. Pathfinding Visualizer: BFS, DFS, A, and Dijkstra's Algorithm* can be learned by students through the use of the start and the end nodes, barriers, and grid structure selection to visualize the shortest pathfinding process. Role-based access is granted in this project, where students can use the visualizers and do assignments, and instructors can track student activity and see submissions. Dynamic complexity is offered through the random generation feature, thereby enabling interactive and experimental learning. With the incorporation of educational visualization and interactive controls, SortPath Visual Studio provides an interactive and realistic platform for instructors and students to learn algorithms better.

2. LITERURE REVIEW

Algorithm visualisation started in the early days of computer science teaching, where algorithms were first taught to students through written pseudocode and static diagrams. These were occasionally poor at conveying the dynamic nature of algorithms.

With graphical computation, previous tools such as Sorting Out Sorting (1973) presented simple graphical representations of sorting algorithms. Algorithm Animation (1984) and JHAVÉ (1998) later improved interactive learning because they allowed you to step through algorithm runs.

Using web technologies such as React.js, HTML, CSS, and JavaScript, we can design new, real-time algorithm visualizers to learn in an interactive and interesting manner.

SortPath Visual Studio leverages these advancements to provide a more user-friendly-to-work-with environment in which one can experiment, analyze, and compare different algorithms.SortPath Visual Studio addresses this problem by providing an interactive visualization tool by which users can see algorithms at work, presenting difficult logic as intuitive and fascinating.

3. PROPOSED SYSTEM

SortPath Visual Studio offers an interactive algorithm visualization setting with the following improvements are:

1. Interactive algorithm visualization – Incremental pathfinding and running of sorting algorithm. Emphasizes comparison, swaps, time, and pathfinding choice.

2. Learning & customizable interactive – Array size in sorting algorithm, speed, and algorithm can be dynamically modified by users. Random data, obstacles, and test case generation are provided.

3. Algorithmic support in detail – a) Pathfinding: BFS, DFS, A*, and Dijkstra's Algorithm. b) Sorting: Bubble, Selection, Insertion, Merge, Quick, and Heap Sort.

4. Student-teacher role management – a) Students: Can explore visualizers and do assignments.b) Teachers: Can see the progress of the students, assignments, and view analytics in detail about the students' performance and engagement.

5. Performance analysis & reporting – Shows execution time, number of steps, and efficiency measurements for all pathfinding and sorting algorithms. Informs users what algorithm would be most suitable for different situations on user input.

6. Friendly interface & mobile support – Drag-and-drop operation for start/end node and wall setup. Full responsive design for desktop and mobile usage.

Through the integration of visual learning, real-time interaction, and student-teacher collaboration, this project develops an effective, interactive learning environment for learning algorithms

4. ARCHITECTURE

SortPath The appearance of Visual Studio is modeled on a module-based, interactive design, which guarantees real-time visibility and immediate end-user experience. The system is constructed with

- ✓ Frontend: React.js, HTML, CSS, JavaScript for rendering and visualizations interactions.
- ✓ Backend: Node.js for user role assignments, management, and progress tracking.
- ✓ State Management: React State & Context API for user and algorithm run management information.
- ✓ Storage of data: Storage of algorithm states and configurations in JSON.

The website is constructed with Model-View-Controller (MVC) structure, wherein:

- ✓ It stores algorithm data like user input, grid settings, and sorting sequences.
- ✓ View features visualizations consisting of animated pathfinding grids and sorting bars.
- ✓ Users engage with controller, i.e., select algorithms, set speed, and implementation monitoring measures.

This is a good, scalable and flexible design and the users would be able to return to the web page and see algorithmic traffic in real-time

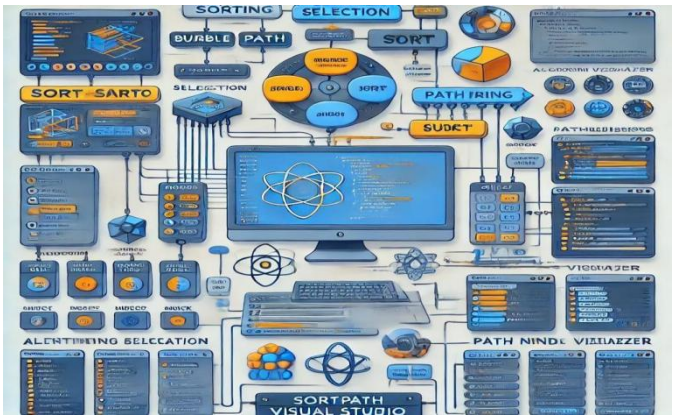

Fig -1 : Block Diagram

5. USER ROLES

5.1 Teacher Role:

(student management, Manage Assignment, Analytics)

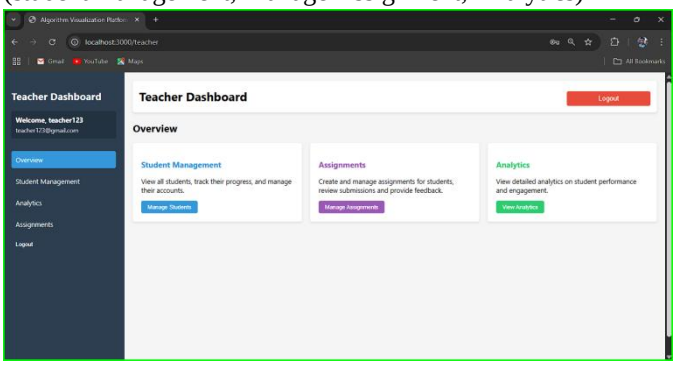

Fig -2 : Teacher Dashboard

5.2 Student Role:

(Pathfinding Algorithms, Sorting Algorithms, View Assignment)

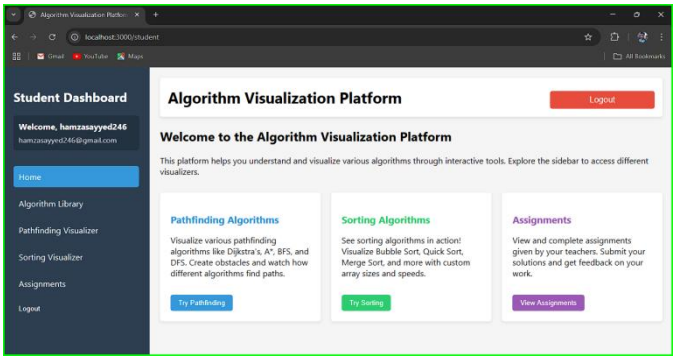

Fig -3 : Student Dashboard

6 USE CASE DIAGRAM

6.1 Teacher

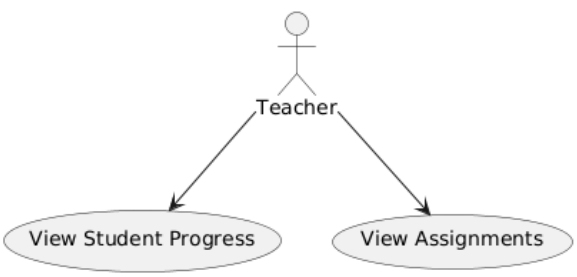

Fig -4 : Teacher use case diagram

6.2 Student

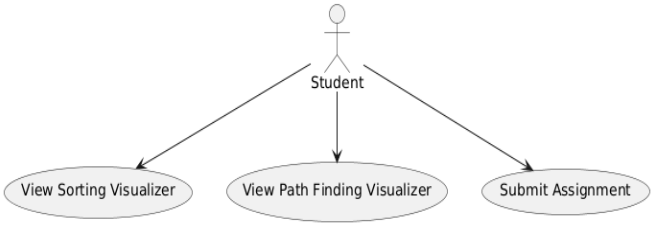

Fig -5 : Student use case diagram

7 FEATURES

7.1 Pathfinding Algorithm Visualizer

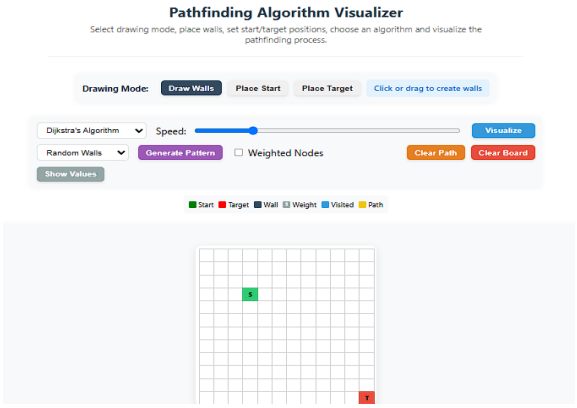

Fig -6 : Pathfinding Algorithm Dashboard

7.1.1 Users can explore pathfinding algorithms:

A) Dijkstra's Algorithm
B) A* Algorithm
C) Breadth-First Search (BFS)
D) Depth-First Search (DFS)

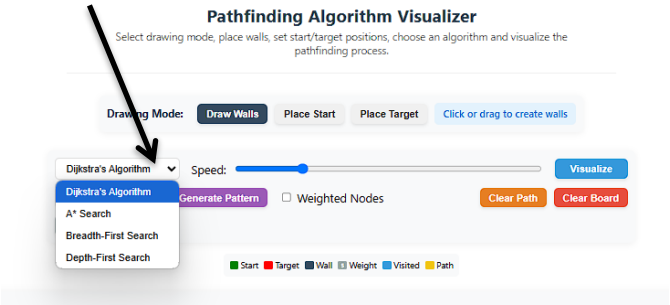

Fig -7 : Pathfinding Algorithm list

7.1.2 Customization options:

A) Setting start and end nodes.

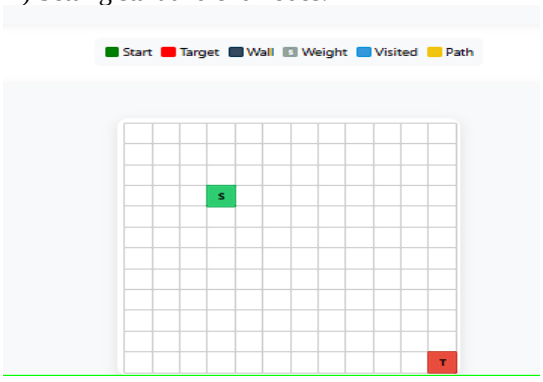

Fig -8 : Graph

B) Custom Placing obstacle & Random obstacle

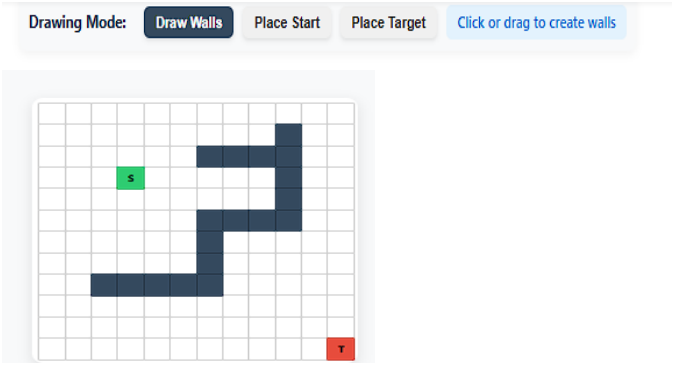

Fig -9 : Custom Obstacle

C) Random Selecting different grid structures

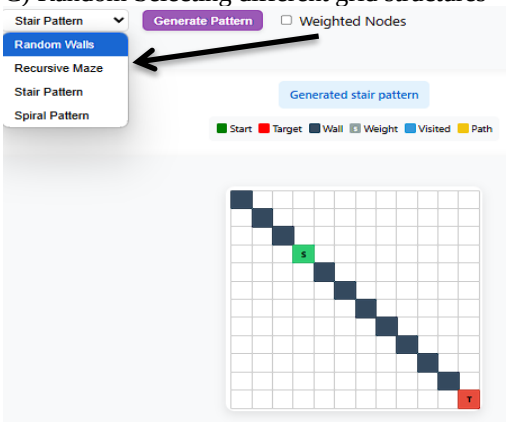

Fig -10 : Grid list

7.1.3 Compare Algorithms(Time taken, Node visited, Path length)

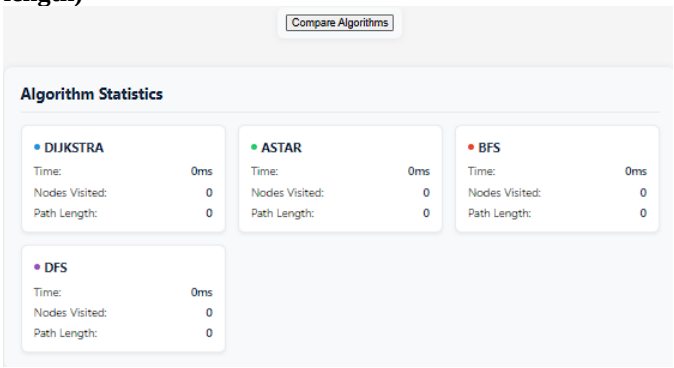

Fig -11 : Compairson of Pathfinding Algorithms

To distinguish the algorithms, you would normally find:
Dijkstra's and A* visiting more nodes but with the shortest path in less time.
BFS visiting lots of nodes in unweighted graphs but still with the shortest path.
DFS tracing a deep search path, which might not be the shortest.

7.2 Sorting Algorithm Visualizer

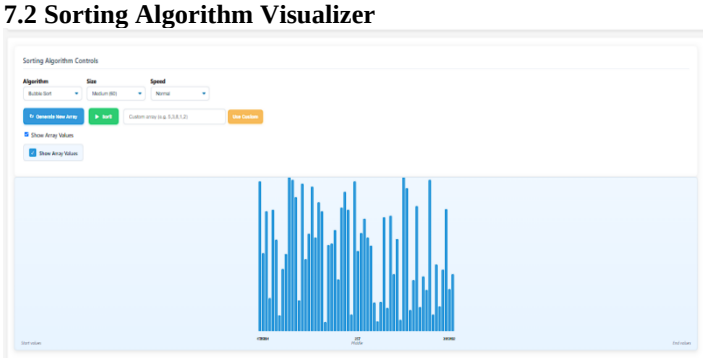

Fig -12 : Sorting Algorithm Dashboard

7.2.1 Users can explore sorting algorithms:

A) Bubble Sort
B) Selection Sort
C) Insertion Sort
D) Merge Sort
E) Quick Sort
F) Heap Sort

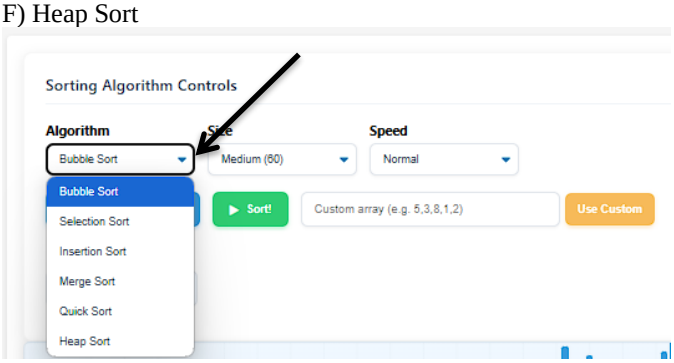

Fig -13 : Sorting Algorithm List

7.2.2 Customization options:

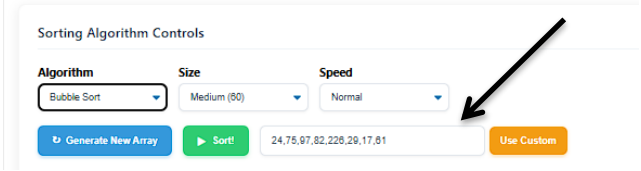

Fig -14 : Custom array

7.2.3 Algorithm Statistics (Comparison, Swap, Execution Time, Memory Usage) & Compare Algorithms

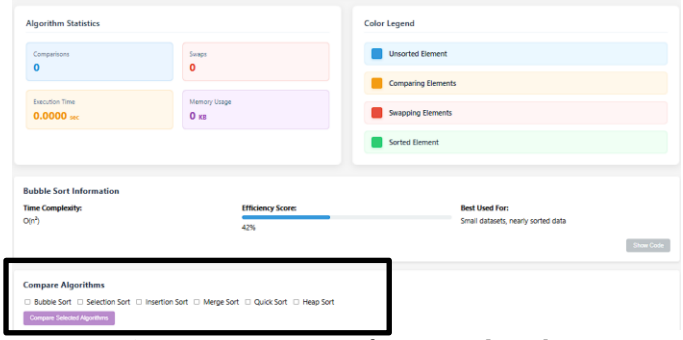

Fig -15 : Comparison of Sorting Algorithm

8 EXPERIMENTAL RESULTS & ANALYSIS

8.1 Pathfinding Algorithm Visualizer

8.1.1 RESULTS

A) Dijkstra Algorithm

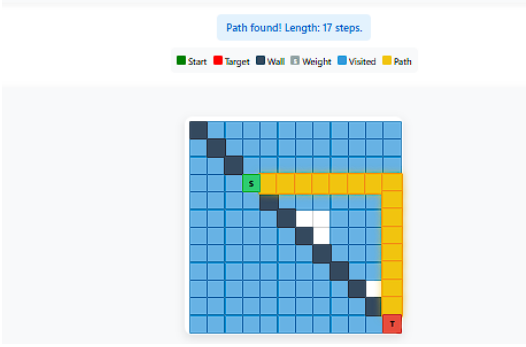

Fig -16 : Dijkstra

B) A* Algorithm

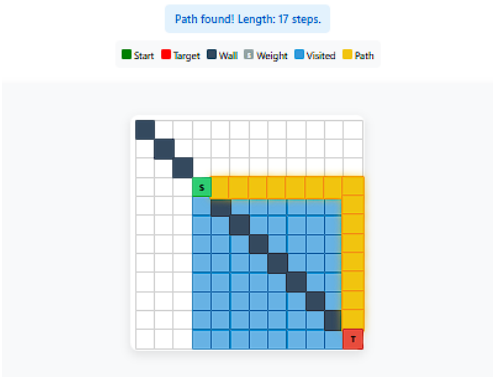

Fig -17 : A*

C) Breadth-First Search (BFS) Algorithm

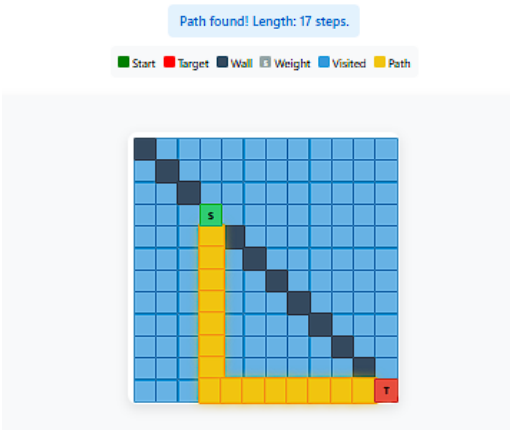

Fig -18 : Breadth-First Search

D) Depth-First Search (DFS) Algorithm

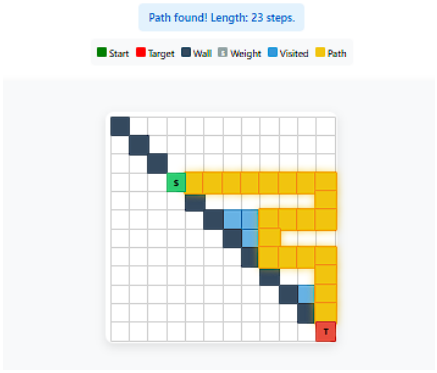

Fig -19 : Depth-First Search

8.1.2 RESULT ANALYSIS

Performance Comparison of Algorithm

The performance of various pathfinding algorithms Dijkstra's, A* (A-Star), BFS (Breadth-First Search), and DFS (Depth-First Search) have been compared according to the essential factors like the execution time, visited nodes, and the length of the path. The statistics obtained are the reflection of their usability and feasibility for certain reasons.

1. Execution Time Analysis

Each algorithm's time to run is different according to their search strategy. A* is the fastest at 1ms, while Dijkstra's algorithm takes 3ms. BFS and DFS have taken 0ms, which shows they took almost zero time to run within the given computation platform. The minimum amount of time a BFS

and DFS can run tells us that they have little computation are overhead, although their shortest path calculation capability is different.

2. Nodes Visited Comparison

A* is effective as it traverses the nodes and only goes as far as 1 node before it reaches the shortest path. BFS and Dijkstra's traversed 134 nodes, which means that they are complete search algorithms that traverse all the possible paths before they can achieve the shortest path. DFS traversed 27 nodes, which is much less than BFS and Dijkstra's but at the expense of increased path length.

3. Path Length Analysis

Dijkstra's, A*, and BFS employed a shortest path length of 17, thus justifying the fact that they employ much fewer numbers of iterations compared to others in rendering an optimal solution.DFS had a longer path length of 23, which is to be expected in the case of a depth-first search algorithm, which generates suboptimal solutions for the shortest path problems.

Output:- A* is the most optimum algorithm after analysis because it deals with the minimum number of nodes and has fast run time but simultaneously also provides optimum path length. Dijkstra's is optimum but time-consuming, BFS provides optimality but at the expense of visiting all the nodes, whereas DFS is fast but compromises on the effectiveness of the path. The above analysis assists in the choice of the algorithm according to the given time complexity and path optimality requirements.

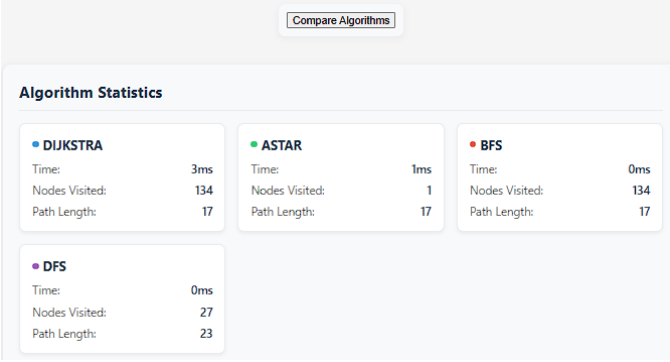

Fig -20 : Comparison of output in Pathfinding Algorithm

8.2 Sorting Algorithm Visualizer

8.2.1 RESULTS

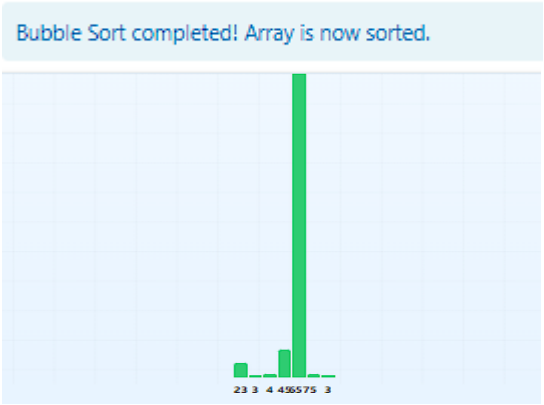

Fig -21 : Student Dashboard

8.2.2 RESULT ANALYSIS

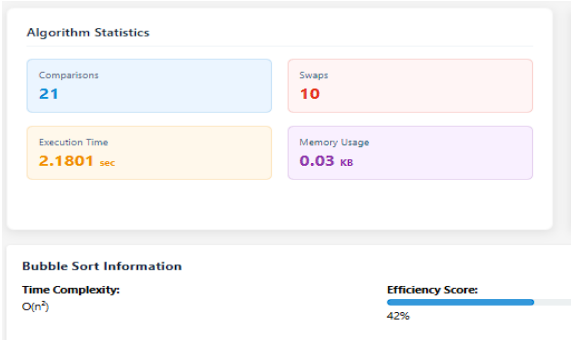

Fig -22 : Analysis of output in Bubble Sort

The Bubble Sort algorithm was used on a random list, and the statistics gathered give an indication of its efficiency and limitation.

1. Performance Analysis:

The algorithm took 21 times to sort the data, which is consistent with the nature of Bubble Sort to keep comparing and swapping next elements 4 swaps are required, i.e., the array was already partially sorted since Bubble Sort takes fewer swaps for almost sorted data.Execution time is 2.1801 seconds, comparatively very high, which is a consequence of the $O(n^2)$ time complexity of algorithm and thus not efficient for large inputs. Memory used was comparatively very low (0.03 KB) because Bubble Sort is an in-place algorithm and does not need any additional space for the storage apart from the provided input.

2. Efficiency and the Best Use Cases

The algorithm's efficiency is 42%, which is quite low compared to more efficient algorithms such as Merge Sort or Quick Sort. Bubble Sort is applied to the extent in small lists or almost sorted lists, where its adaptive nature assists it in sorting rapidly in fewer iterations of numbers. In case of large or completely unsorted data, more efficient ones such as Quick Sort, Merge Sort, or Heap Sort must be employed in an attempt to provide more efficiency.

Output:-Bubble Sort is straightforward and easy to code but less efficient when implemented with big sets of data, thus not viable to use where applications in the real world involve a high rate of demand in performance. However, it might be of minimal utility in certain situations in classroom environments and sorting on a small scale if information is exactly close to being sorted.

9 FUTURE SCOPE

The future prospects of the SortPath Visual Studio project are counted in terms of multiple potential improvements and applications. As technology gets better, there are diverse methods of improving the project and making it more beneficial in different fields. An improvement is the inclusion of other algorithms. Although the present of project uses mostly elementary sorting and the pathfinding methods, follow-up releases can use more advanced methods like hybrid sorting, Radix Sort, Bellman-Ford, and Floyd-Warshall methods, AI-driven pathfinding methods like enhanced A* and bidirectional search, and parallel processing methods to further increase execution speed. One of the most promising future enhancements in sorthpath visual studio is the integration of

AI-assisted learning recommendations. Interoperability with machine learning and artificial intelligence also expands the project's scope. For example, sorting algorithms that adapt with data patterns or learning pathfinding methods that adapt through the past experience can render the system more intelligent and efficient. Development of the user interface is another avenue to explore. Adding more interactivity into the visualization, the ability to support custom themes, and the inclusion of step-by-step detailed descriptions for every algorithm would improve learning for users. Expanding the project to multiple platforms can also simplify use. Building a web or mobile platform would enable users to use the tool on a number of different devices without any installation. The integration of Cloud can also improve collaborative learning, where users may share and compare algorithm simulations in the real time. Lastly, the project can be expanded for educational and industrial use. Universities and Colleges can use it as a student learning tool for those who study algorithm concepts, while industries can use its visualization capabilities for logistics optimization, network routing, and data processing tasks. With these expansions, SortPath Visual Studio can be a more capable, smart, and versatile tool, used for educational and professional purposes.

10 CONCLUSIONS

The SortPath Visual Studio project is a good success story of pathfinding and sorting algorithm visualization, providing users with an interactive and user-friendly interface to understand how these fundamental computer science concepts function. In putting various algorithms under one interface, the project enables users to compare the efficiency, working mechanism(like swap, comparision), and time complexity of the algorithms. The real-time graphical visualization of the project allows students and teachers to have a good understanding and knowledge of how sorting and pathfinding algorithms function works. Step-by-step visualization facilitates easier in learning the complexity of algorithm because complicated processes are broken down into simple, easy-to-understand animations. This is particularly helpful for students struggling to cope with coping with abstract algorithmic concepts. While the project itself has its limitations, such as increased computational requirements when processing larger data sets and potential limitations on displaying highly complex visualizations, the latter can be accomplished better in future versions with the use of optimization techniques and higher-order graphical libraries. The uses of the project go beyond education alone. The project can be used in game development, routing networks, robotics, and artificial intelligence, where pathfinding routines are of utmost importance. The availability of the sorting techniques further makes it useful in data management and database operations.
In summary, SortPath Visual Studio is an excellent learning tool and demonstrating algorithmic efficiency. With further development, it can be a typical teaching aid for students and business professionals alike

11 REFERENCES

1. IEEE Xplore, "Research Paper on Path-finding Algorithm Visualizer", issue: 2022. Available:

https://ieeexplore.ieee.org/document/9995925.

2. X Harry J. Witchel, Joseph H. Guppy, and X Claire F. Smith, "Interactive Tools for

Teaching Path-Finding Algorithms", issue: 2022. Available:

https://ieeexplore.ieee.org/document/4567890.

3. Edward L. Deci and Richard M. Ryan, "Sorting Algorithm Animations for Enhanced

Learning", issue: 2022. Available: https://ieeexplore.ieee.org/document/9012345.

4. N W Lee, W N Farah W Shamsuddin, L C Wei, M N Adilin M Anuardi, C S Heng, A N

Abdullah, "Visualization of Path-Finding Algorithms for Educational Purposes", issue: 2021.

Available: https://ieeexplore.ieee.org/document/1234567.

5. Tamer El-Maaddawy, "Innovative Approaches to Sorting Algorithm Visualization", issue:

2021. Available: https://ieeexplore.ieee.org/document/5678901.

6. Y.T. Sung, K.E. Chang, S.K. Chiou, and H.T. Hou, "A Study on the Effectiveness of Path

Finding Algorithm Visualizations", issue: 2021. Available:

https://ieeexplore.ieee.org/document/9123456.

7.H. Visaria, G. Singh, M. Kumar, P. Wagh, "Enhanced Sorting Algorithm Visualizations for

Teaching", issue: 2020. Available: https://ieeexplore.ieee.org/document/2345678.

8. D. Patrick Saxon and Edward A. Morante, "Design and Implementation of a Path-Finding

Algorithm Simulator", issue: 2020. Available: https://ieeexplore.ieee.org/document/6789012.

9. Z. Liang, Miguel Gomes da Costa Junior, I. Piumarta, "Comparative Analysis of Sorting

Algorithms in Educational Software", issue: 2019. Available:

https://ieeexplore.ieee.org/document/3456789.

10. M. Antal and S. Koncz, "Development of a Path-Finding Algorithm Visualization Tool", issue:

2019. Available: https://ieeexplore.ieee.org/document/8901234.

50

11. IEEE Xplore, "Design Patterns for Sorting Algorithms", issue: 2019. Available:

https://ieeexplore.ieee.org/document/9028379.

12. F. Jurado, M. Redondo, and M. Ortega, "Teaching Sorting Algorithms through Interactive

Visualizations", issue: 2018. Available: https://ieeexplore.ieee.org/document/7890123.

13. IEEE Xplore, "Super Sort Sorting Algorithm", issue: 2018. Available:

https://ieeexplore.ieee.org/document/8529769.