International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

From Pixels to Paths: A Visual Approach to Understanding Pathfinding and Sorting Algorithms

Isha Thakur[1], Sabah Shaikh[2], Toshi Jain[3], Aparna Tiwari[4]

[1,2,3,4]Dept. of Computer Engineering Usha Mittal Institute of Technology Mumbai, India

Abstract

In computer science and algorithmic understanding studies, visual tools play an important role in improving un- derstanding and collaboration. This research paper introduces a visualization algorithm that aims to bridge the gap between the- oretical ideas and practical applications. Visualization tools use intuitive graphical representations to explain the inner workings of various algorithms and encourage deeper understanding.

We aim at developing a easily accessible user interface to provide our users to visualize the real-time algorithms in a interesting and fun way.This visualization tool supports a variety of algorithms, from identification and search to image traversal, providing various platforms for learning and algorithm analysis. The un- derlying architecture integrates modern networking technologies to ensure accessibility across multiple devices and platforms.

Keywords: Algorithms, visualization, sorting, path-finding, e- learning

INTRODUCTION

In computer science, algorithms play an important role in solving complex problems and improving the performance of various tasks. However, understanding and analyzing algorithms can be difficult, es- pecially for beginners. This is where software algorithm visualization tools come into play. It is a powerful tool that helps visualize various algorithms, making it easier for users to understand and analyze them. A software algorithm visualization tool is a computer program that shows the step-by-step execution of an algorithm with consistent data and different models. It provides a graphical representation of the algorithm's behavior, making it easier to understand and analyze the logic behind it. This tool is useful for students and professionals in computer science because it can be used for learning, teaching, and research. One of the main benefits of using algorithm visualization tools is the ability to simplify algorithms. It breaks the algorithm into small steps and visually displays each step, making it easy to follow. This helps identify any errors or issues in the algorithm that can be fixed and improved. Additionally, the graphical representation makes it easy to compare different algorithms and understand their differences in terms of time complexity, space complexity, and overall performance. Another advantage of using software algorithmic visualization tools is their interaction. Users can interact with the visual representation by pausing, repeating, or iterating the execution of the algorithms. This allows users to focus on a specific step or data structure and understand its role in the overall algorithm. Additionally, some visu- alization tools enable optimization and debugging, making it

easier to test different ideas and understand the behavior of the algorithm. Software algorithm visualization tools can simplify algorithms and make them more interactive, as well as making learning easier, more informative, and fun. It is a useful tool for students to understand and visualize complex algorithms, leading to better understanding and retention of concepts. Similarly, teachers can use it to explain more complex algorithms, thus making the learning process more interactive and engaging for students.

The Algorithm Visualization website is an essential tool for re- searchers, engineers, and game developers. For researchers, these platforms provide a good environment for interactive research and analysis of algorithms, providing a deeper understanding of their complexity and allowing comparison of algorithm performance. Ad- ditionally, algorithm visualization tools aid teaching and communi- cation because visualization programs improve the explanation of complex algorithms. Researchers can also use these tools to create prototypes and experiments to test hypotheses and explore new algorithmic methods. Additionally, visualization can facilitate data analysis, helping researchers see patterns, trends, and insights during research.

Algorithm visualization tools for experts can be used as a good debugging tool that can identify errors in program code or algorithms by visualizing the representation of success. This tool helps with algorithm optimization, allowing experts to test different methods to improve code performance and efficiency. Algorithm visualization tools also aid learning and skill development, providing employees with a practical and visual understanding of algorithms and data struc- tures. In addition, it enables collaboration and information sharing between teams by processing the visualization data generated during the development of the algorithm.

Algorithm visualization tools play an important role in game develop- ment. Business developers can use these platforms to implement and optimize search algorithms to ensure best performance in the game. In addition, algorithm visualization tools provide a way to solve many of the algorithmic challenges inherent in game development, allowing developers to test different solutions and monitor their effects on game play and performance. This tool also serves as an educational tool for game developers, helping them understand the fundamentals of algorithms used in game development and facilitating better com- munication between developers and developers. Real-time simulation capabilities provided by visualization tools allow game developers to see how different algorithms behave during game play, making rapid prototyping and testing easier. In summary, the Algorithm Visualiza- tion website provides researchers, practitioners, and game developers with a versatile and useful platform that supports understanding, problem solving, and collaboration in many fields.

LITERATURE SURVEY

The study of various algorithm visualizers presents a comprehensive analysis of the current state of algorithm visualization (AV) based on the examination of over 500 AVs. The study addresses the distribution of AVs among topics, quality assessment, dissemination methods, and the need for improvement in AV development and access. The authors found that while there are good AVs and active developers, many AVs are of low quality and coverage is skewed towards easier topics, making it challenging for instructors to locate what they need.The study indicates that AVs play a crucial role in computer science education, particularly in visualizing algorithms and data structures. It addresses the effectiveness of AVs in educating computer science students, citing mixed findings from previous studies, indicating the difficulty in creating and deploying effective AVs. The document also provides insights into the historical

development of AV systems, their pedagogical effectiveness, and the challenges in disseminating and accessing AVs[1].

In [2] the authors discuss the development and implementation of an e-learning tool for visualizing shortest paths algorithms. Specifically, the tool focuses on the implementation of the Dijkstra algorithm, intending to integrate different algorithms for shortest path deter- mination. The preliminary test results indicate the usability of the e-learning tool and its potential to support students in developing efficient mental models regarding shortest paths algorithms.

Importance of algorithms and the driving force behind the research. Additionally, the article presents VizAlgo, an online platform. The model was prepared using Java and then made available online. The intriguing thing about VizAlgo is that it allows one to view exactly what happens behind the scenes, for example, when an algorithm for backtracking is run. The visualizer has been built by the author through the design of multiple Java plugin modules. The study presents the findings of a survey regarding the effectiveness of VizAlgo and focuses on well-known, if not the most significant, sorting algorithms[3].

This paper analyzes developments in pathfinding algorithms such as Dijkstra's, A*, BFS, and DFS, with a focus on their efficacy in multi- agent situations. It explores the issues that A* faces as a result of agent conflicts and recommends the use of bidirectional techniques to increase pathfinding accuracy. Furthermore, it emphasizes the usage of Pygame technology for interactive visualization and user interface development, which provides a potential platform for creating and testing pathfinding methods.[4]

Work about the JavaScript Visualization Library also brought a new look on visualization.Key features of JSAV include automated layout for a number of traditional data structures, support for presentation slideshows, and support for TRAKLA2-style proficiency exercises that require the student to demonstrate proficiency with an algorithm by simulating its key steps.At the University of Turku, JSAV-based content was found to be helpful for students' learning, easy to use, and provided valuable feedback. The library has been used in various courses, covering topics such as logical gates, machine code, assembly code, code parsing and compiling, and artificial intelligence, and has been integrated into learning management systems to report scores and student interactions.The integration of JSAV into different learning environments, along with the support for interactive exercises and visualizations, highlights its potential to revolutionize computer science education.[5]

Camil Demetrescu et.al proposed a web application named WAVE. In this, they have utilized the distribution driven methodology for calculation representation over the Web. By utilizing this methodol- ogy, calculations execute on a designer's server and their information structures are distributed on black boards held by the customers. Liveliness is counted by joining perception overseers to the informa- tion structures distributed on the customer's blackboard: adjustments to these constructions, because of the far-off calculation execution, trigger the running of the comparing controllers on the customer's side.[6]

John Hamer, worked on the lightweight tool (LJV) that provides high- quality visualizations of Java data structures [7]. The vital distinctive characteristic of the tool is its simplicity. For teaching purpose LJV can be easily adopted by the instructors without requiring any change in the methodology i.e., we can keep the same editor, Java environment, code examples, etc. For users uses we need to call a single (static) method that takes a single (Object) argument. LJV works by using Java reflection to traverse the fields of an object (and the fields of the fields, etc.), generating a textual description of the connectivity as it goes. The description is then passed to the graph drawing program Graph Viz (North and

Koutsofios, 1994), Graph Viz automatically produces an image of the graph, in a choice of bitmap or vector formats [8].

METHODOLOGY

The landing page serves as the approach to our interactive pathfinding and sorting algorithm visualization tool. It comprises of four primary elements: home, pathfinding algorithms, sorting algorithms and about section.

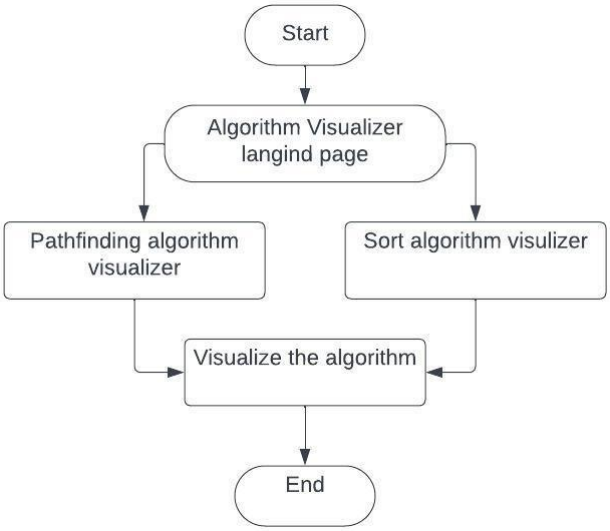

Fig. 1. Landing Page Block Diagram

A. PathFinding Algorithm

Our incorporated pathfinding algorithms provide a comprehensive range of options for users to explore and appreciate various tech- niques for determining optimal routes. The available algorithms are as follows:

1. Dijkstra's Algorithmii.A* Algorithm
2. Bi-directional Search iv.Breadth-first Search (BFS) v.Depth-first Search (DFS) vi.Convergent Swarm Algorithm

Example of Pathfinding Algorithm Components of Pathfinding Section

1. ***Pathfinding Tutorial:*** Users are guided by a tutorial to un- derstand the fundamentals of pathfinding algorithms and ensure a seamless learning experience.
2. ***List of Algorithms:*** Presents users with a selection of pathfind- ing algorithms to choose from, enabling them to visualize and compare different strategies for route optimization.
3. ***Mazes and Patterns:*** Allows users to investigate predefined mazes and patterns, including recursive division, random maze, and stair patterns. These predefined layouts acted as the test environments for executing the chosen algorithms.
4. ***Bomb Node Addition:*** Permits users to introduce a "bomb" node into the pathfinding process, necessitating the resulting path to navigate around this obstacle, adding complexity to the route-finding task.
5. ***Clearing Options:*** Provides users the ability to reset the board, eliminate obstacles, and clear paths, facilitating iterative experimen- tation with different algorithms configurations.
6. ***Speed Adjustment:*** Users can control the execution speed of algorithms by selecting fast, average, and slow settings to tailor the visualization speed to their preferences.

B. Sorting Algorithm

This application offers a user-friendly interface for investigating various algorithms and their efficiency, regardless of whether you're studying about sorting algorithms or just want to see them in action.

1. ***Algorithm Selection:*** Choose from a variety of available sorting algorithms, including popular ones like Bubble Sort, Insertion Sort, Selection Sort, Merge Sort, Quick Sort, and more.
2. ***Array Size Selection:*** Modify the array's dimensions before sorting it. With this option, you may see how various algorithms function with various data sets.
3. ***Mode of Darkness and Lightness Toggle:*** To improve read- ability and fit your mood, alternate between bright and dark settings.
4. ***Randomize Array:*** To create various sorting scenarios, simply randomise the array's elements.
5. ***Pause Sorting:*** You can stop the sorting process whenever you want to take a closer look at the array's condition.
6. ***Speed Control:*** To see the algorithm's behaviour in real time, change the sorting process's speed. You can adjust the animation's speed using the speed sliders as desired.
7. ***Color-Coded Visualization:*** The graph's bars each represent an array element, and the colour of each bar indicates where it is in the sorting process right now. As an illustration

- Green: This shows that the element has been sorted to itsultimate location.
- Yellow: Indicates elements that are currently being compared.
- Pink: Indicates elements that are being swapped.

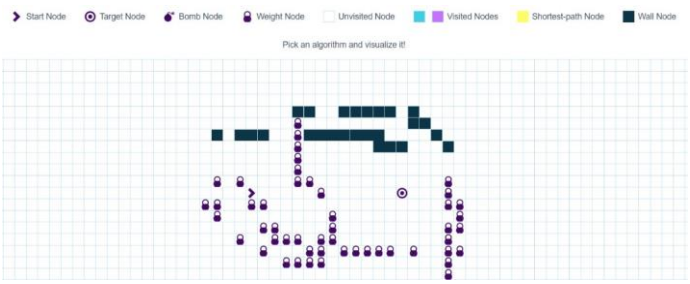

Fig. 2. Initial setup of start and target nodes

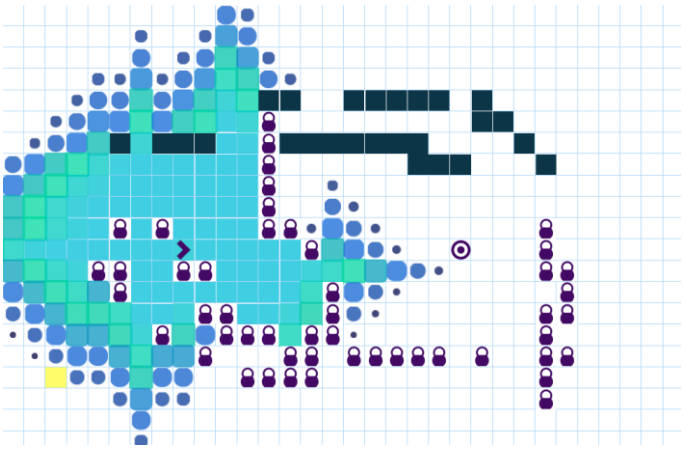

Fig. 3. Finding Shortest Path

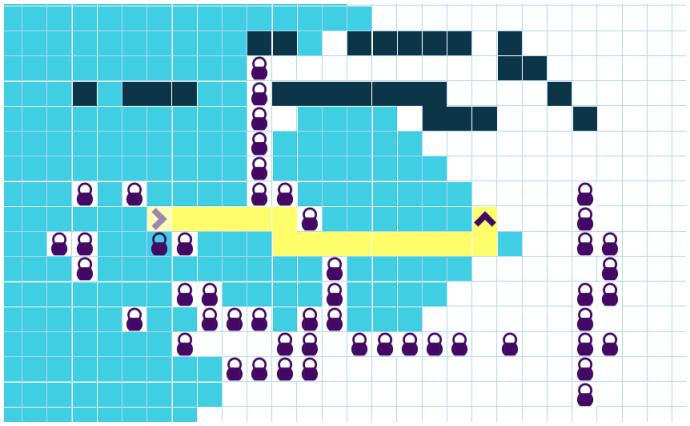

Fig. 4. Shortest path found

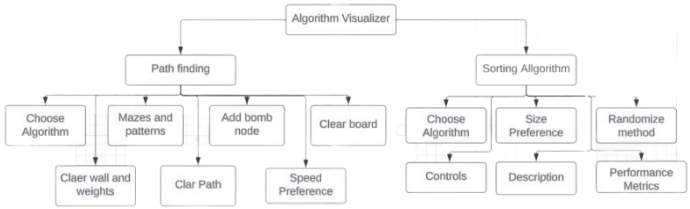

Fig. 5. Block diagram

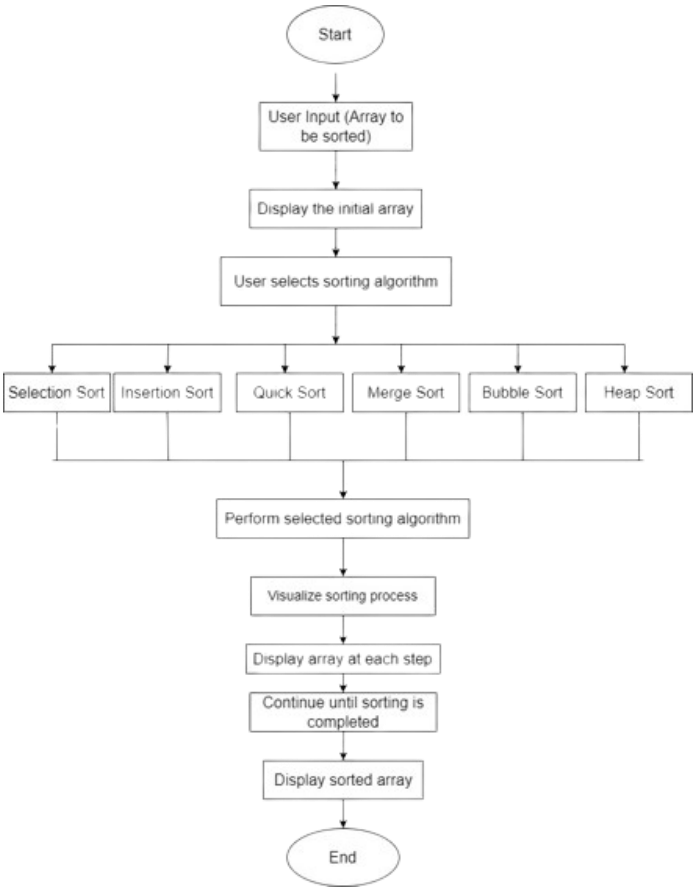

Fig. 6. Sorting Flow Chart

RESULTS AND DISCUSSIONS

Previous studies have established the significance of algorithm vi- sualizers in aiding comprehension and analysis. However, many existing solutions may lack certain features crucial for comprehensive understanding and user engagement. Our visualizer addresses these limitations by introducing innovative functionalities that significantly enhance its utility and effectiveness.Some of the features which are additionally added are

Weighted Nodes in Pathfinding Visualizer: Our implementation introduces weighted nodes, albeit with a fixed weight of 15 units, to the pathfinding visualizer. While traditional pathfinding algorithms typically treat all nodes uniformly, our inclusion of weighted nodes allows users to designate certain nodes as more or less traversable than others. In our implementation, the fixed weight of 15 units for these special nodes offers a simplified yet effective means of incorporating varying node weights into the visualization.This feature adds a layer of complexity to pathfinding simulations, enabling users to model scenarios where certain paths are significantly more or less favorable than others.

Bomb Nodes in Pathfinding Visualizer: In our algorithm visualizer, we introduce the concept of bomb nodes, which serve as strategically placed waypoints that the pathfinding algorithm must navigate before reaching the target node. Unlike regular nodes, bomb nodes impose a mandatory route that the algorithm must follow, adding an element of complexity and strategy to pathfinding simulations.The rationale behind bomb nodes lies in their ability to simulate scenarios where certain areas must be explored or obstacles must be overcome before reaching the final destination.For instance incase of cab pooling feature in many applications some passengers are to be picked on route to others.Bomb nodes allow users to model such scenarios realistically, offering insights into the optimal route planning and resource allocation.The combination of weighted nodes and bomb nodes enables users to model realistic environments with diverse terrain features and navigation constraints, fostering a deeper understanding of pathfinding algorithms' behavior and performance.

Predefined Maze Patterns in Pathfinding Visualizer: In our algorithm visualizer, we introduce a unique feature that allows users to select from a variety of predefined maze patterns in the pathfinding visualizer. Instead of manually placing wall nodes or weighted nodes to create a maze, users can simply choose from a selection of preset maze options, each offering a distinct layout and level of complexity.By providing a collection of preconfigured maze templates, the visualizer offers users a convenient and efficient way to explore different maze configurations and visualize pathfinding algorithms in action.Our visualizer includes a diverse range of maze options, ranging from simple grids to weighted mazes. Each predefined maze pattern is carefully crafted to offer a unique challenge and test the capabilities of pathfinding algorithms in various scenarios. Users can choose from different mazeshapes, and complexities, allowing for endless exploration and experimentation.

Pause and Play Feature in Sorting Visualizer: Our algorithm visualizer incorporates a Pause and Play feature within the sorting module, offering users granular control over the sorting process. This feature allows users to pause the sorting animation at any point during execution, enabling them to visualize and analyze the algorithm's operation step by step.The Pause and Play feature serves as a powerful educational tool, allowing users to observe the sorting algorithm's behavior in real time and gain insights into its underlying mechanics. By pausing the animation at strategic intervals, users can examine the state of the data array at each iteration, track the movement of elements, and understand how the algorithm rearranges the data to achieve the desired order.

Visualizing Speed Control: Our algorithm visualizer includes a Visualizing Speed Control feature,

allowing users to adjust the animation speed of both the pathfinding and sorting visualizers. The Visualizing Speed Control feature offers users the ability to customize the animation speed according to their preferences. Users can choose from a range of predefined speed settings or adjust the animation speed dynamically using a slider or toggle control. This granularity in speed adjustment allows users to tai- lor the visualization experience to their individual learning pace, ensuring optimal comprehension and engagement.By offering control over the animation speed, the visualizer promotes deeper understanding and comprehension of the underlying algorithms. Users can adjust the speed to match their cognitive processing speed, enabling them to follow the algorithm's execution more closely and absorb key concepts and insights effectively. Slower speeds facilitate detailed observation and analysis, while faster speeds allow users to grasp the algorithm's overall behavior and dynamics more efficiently.

CONCLUSION AND FUTURE SCOPE

In conclusion, our algorithm visualizer is a testament to the con- tinuous evolution of computational tools, which is driven by the combination of innovative ideas and empirical findings. Delving into the wealth of knowledge offered by previous articles, we discovered a treasure trove of algorithms, each of which laid the foundation for our efforts. However, armed with a vision of improvement, we set out on a journey to redefine the boundaries of algorithm visualization.

Enriched with the introduction of weighted nodes and bomb nodes, our pathfinding visualizer transcends conventional boundaries by offering a nuanced view of traversal strategies. Weighted nodes bring realism to our simulations and mirror real-world scenarios where certain paths have varying degrees of importance. Meanwhile, the inclusion of bomb nodes pioneers a new approach that forces users to strategize and prioritize their traversal paths and promotes a deeper understanding of optimization techniques.

In parallel, our sorting visualizer rises to new heights with the integration of pause and replay functions, giving users more control over the sorting process. This feature not only facilitates granular analysis, but also supports an interactive learning environment where users can dissect and understand the complex dynamics of sorting algorithms at their own pace.

Our commitment to user-centered design is illustrated by incorpo- rating visualization speed controls into both visualizers, allowing users to adjust the pace of simulations to their preferences. Whether you're looking for a quick overview or a thorough examination, our visualizer adapts to different learning styles, ensuring an enriching and personalized experience for everyone.

Essentially, our algorithm visualizer goes beyond the realm of mere simulation and emerges as a dynamic platform for exploration, discovery and innovation. It embodies the collective wisdom of the past, the ingenuity of the present, and the promise of the future. As we continue to push the boundaries of what is possible, we invite users to join us on this journey as we unravel the mysteries of algorithms together and pave the way for a brighter tomorrow. In envisioning the future trajectory of our algorithm visualizer, we embark on a journey of innovation and refinement aimed at providing users with an unparalleled learning experience. Introducing real-time visualization capabilities will revolutionize the way users interact with algorithms, allowing them to witness the intricate processes unfold before their eyes with immediacy and clarity. This immersive feature will not only deepen comprehension but also foster a sense of engagement and wonder. Moreover, the integration of a robust Integrated Development Environment (IDE) will empower users to seamlessly transition from visual exploration to practical implementation, enabling them to test and refine their own algorithmic solutions within the

same platform.

Expanding the horizons of our visualizer, we aspire to incorporate a diverse array of cutting-edge machine learning algorithms, reflecting the forefront of artificial intelligence research and application. From classic supervised learning techniques to state-of-the-art deep learning models, users will have the opportunity to delve into the inner workings of these algorithms, gaining insight into their strengths, limitations, and real-world applications. Furthermore, we envision a suite of complementary features designed to enhance the learn- ing journey, including interactive tutorials tailored to different skill levels, community forums for knowledge sharing and collaboration, and performance analytics to track progress and identify areas for improvement.

With these enhancements, our algorithm visualizer will transcend its current role as a mere educational tool, emerging as a dynamic platform for exploration, experimentation, and innovation in the field of computer science and beyond. Whether users are students seeking to deepen their understanding of algorithms, educators looking for engaging teaching resources, or professionals aiming to stay abreast of the latest developments in machine learning, our visualizer will serve as an indispensable companion on their learning journey.

REFERENCES

1. Clifford A. Shaffer, Matthew L. Cooper, ‘Algorithm Visualization: The State of the Field‘, Virginia Tech, 2017.
2. Daniela Borissova, Ivan Mustakerov, ‘E-learning Tool for Visualization of Shortest Paths Algorithms‘, Trends of journal of Sciences Research Vol. 2, No. 3, 2015, pp 84-89
3. S. Šimoňák, ”Using algorithm visualizations in computer science edu- cation.” Central European Journal pf Computer Science,.vol 4 no.3, pp 183-190, 2014
4. Brian Faria(2017) ‘Visualizing sorting algorithms‘, Rhode Island Col- lege algorithm teaching methodologies IEE(2023).
5. V. Karavirta and C. A. Shaffer, ”Creating engaging online learning material with the JSAV JavaScript algorithm visualization library,” IEEE Transactions on Learning Technologies, vol. 9, pp. 171-183, 2016.
6. Demetrescu, C., Finocchi, I., Italiano, G.F., Naher, S., Visualization in algorithm engineering: Tools and techniques, Experimental algorithmics, 2002.
7. Hamer, John., A Lightweight Visualizer for Java, Third Program Visu- alization Workshop, 2014
8. Stephen C. North and Eleftherios Koutsofios. Application of graph visu- alization. In GI’94 Graphics Interface, pages 235–245, Banff, Alberta, Canada, 1994. URL citeseer.nj.nec. com/221206.html.