IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 12, December 2021

DOI: 10.17148/IJARCCE.2021.101256

ALGOVIZ: VISUALIZE SORTING AND PATHFINDING ALGORITHMS

Prince Gupta [1], Sahaj Dhawan [2], Divya Soni[3]

Student, Department of Computer Science Engineering, HMR Institute of Technology and Management, Delhi 110036, India[1,2]

Assistant Professor, Department of Computer Science Engineering, HMR Institute of Technology and Management, Delhi 110036, India[3]

Abstract: Algorithm visualization illustrates how algorithms work in a graphical way. It mainly aims to simplify and deepen the understanding of algorithms operations. Within the paper, we discuss the possibility of enriching the standard methods of teaching algorithms, with algorithm visualizations. As a step in this direction, we introduce the Algorithm visualizer platform, present our practical experiences and describe possible future directions, based on our experiences and exploration performed by means of a simple questionnaire

Keywords: Graphical Way, Algorithms Operations

I. INTRODUCTION

Understanding Data Structures and Algorithms (DSA), which includes the arrangement of algorithm theory, is a very challenging task in the computer science field [1-2]. DSA is one of the most important subjects, but due to its abstract character, it is also one of the most difficult to master [3]. The difficulty is generally due to the algorithm, which is derived from dynamic step-by-step procedures.

Programming classes are included in both computer science and information science curriculum. Their objective is to educate students on fundamental programming principles such as control, methods of aggregation, sorting algorithms, and structures, etc. pupils, to support them in their learning [4]. The objective of novice programmers is to implement the abstract process of algorithm execution in a way or language that will be understandable to the computer. However, if a programmer wants to 'explain' something to the computer, he or she first has to understand it perfectly [5]. Namely, it is very difficult to understand and learn complex algorithms such as iterations, recursions or sorting algorithms only by watching code lines or a flow chart. On the contrary, if students can control their own data sets and see the whole process of algorithm execution, they are able to draw conclusions from the resulting data or in course of the algorithm visualization [6].

This paper is presented as follows. Section 2 throws light on the Relevance of the work. This section points out the Need and Observations related to the proposed system followed by the Motivation behind it and a Literature Survey of the work. Section 3 of the paper describes AlgoViz. Section 4 discusses the merits of the proposed system. Section 5 Analyses the Result and Section 6 discusses the Conclusion and Future Work.

II. RELATED WORK

II.I Need & Observation

Algorithms and Data Structures are critical components of computing and should be included in any computer professional's education. A thorough comprehension of the topic in the last few decades, due to an increase in the number of students enrolled in technical schools introducing approaches to help and enrich their schooling [7]. The system that has been proposed focuses on learning and comprehending algorithms with the assistance of visualizations. Visualizations assist students in understanding concepts.

To better understand and grasp the power of algorithms, students need to understand the algorithms at a low level, which is not possible in today's learning methods. With the rise in the online education system during the pandemic [8], there is a need for a relevant platform where teachers can teach algorithms easily and assess students remotely.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 12, December 2021

DOI: 10.17148/IJARCCE.2021.101256

II.II MOTIVATION

In unprecedented times such as Covid-19 where online education is of utmost importance, there is a need for a platform for teachers to teach students effectively. DSA is the crux of Computer Engineering and its understanding is of utmost importance. Visualization of any topic gives a clear understanding of a particular topic and helps the learner grasp the concept quickly. VizAlgo is designed with this motivation to help learners easily grasp and understand any data structure or algorithm concept and to bring teachers and students in one place to provide a collaborative platform to ease remote learning.
The project is developed with the motivation to help students, scholars, professionals to learn and understand the concepts of Data Structures and Algorithms. It also provides a synergetic platform to ease remote learning

II.III LITERATURE REVIEW

To design effective software, every software engineer needs to have a thorough understanding of DSA. Visualizers have a proven track record of giving useful information to the user's comprehension. The various Algorithm Visualizers developed so far over the past few years are:

An Artificial intelligence search Algorithm visualizer has been developed in the past by Abu Naser et al., in 2008 [9]. The tool was deployed to solve the water jug problem with strategic three operational modes.

A platform named JHAVEPOP was made by Furcy David in 2009 [10]. Languages like C++ or Java having Data Structures as Linked List can be visualized using this platform. The platform creates step by step visualization of the code written by the scholar in C++ or Java.

A platform VisuAlgo was designed by Dr Steven Halim [11] in 2011 to productively explain the foundation of distinct algorithms to their students. Many Advanced algorithms were present in this platform as introduced in the book 'Competitive Programming' [12] which was co-authored with his brother Dr Felix Halim. The platform was limited to running only on small devices.

A platform for Algorithm visualization was designed by Shaffer C. et Al., [13] in 2011 which was deployed on the belief of group learning, unlike other platforms that provided barely a limited set of algorithms to be visualized. Group deployed learning through forums, discussions were promoted by the platform.

The idea of the operation for expanding algorithm visualization tools by the programmers perhaps upgrade the future of algorithm visualizers was proposed by Cooper Mathew et al., [14] in his publications in 2014.

A platform named VizAlgo was initiated by Simonak Slavomair [15] in 2015. Visualization of sorting algorithms was the principal focus of the platform. It is comprised of mainly two interlinked components: the main component and a pair of individualistic parts. There were many classes for helping in the execution of code in the main segment while the code for visualization was present in the independent component.

For the Visualization of shortest path algorithms, an e-learning software was developed by Borissova D. et al., [16] in 2015. The platform replenishes step by step visualization of the shortest path algorithm execution by conceiving, visualizing, and improving different graph structures.

Algorithm visualizations were discussed and also an aspect was introduced to choose the particular algorithm deployed on the performance and virtue of a certain problem by Jonathan F. C. et al., [17] in 2016

Algorithm visualization on the mobile platform was initiated by Supli A. A. et al., [18] in 2017. It throws light on two main characteristics: the layout of the User interface (UI) and its interactivity with the user.
The designing of an algorithm visualizer was executed by Romanowska K. et al ., [19] in 2018. Moreover, discussions were made regarding visualizer traits deployed on training and reliability goals.
A recursion tree visualizer was generated by Bruno Papa et al., [20] in 2020. The platform was meant to generate a visualized recursion tree from a certain recursion code. Reingold -Tilford's algorithm was set up to place the nodes of trees in a productive way
An integrated platform named AlgoAssist has been developed by Aniket B. Ghadge et al., in 2021 [21]. The tool contained a lab integration feature which makes it more realistic for both students and teachers.
.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 12, December 2021

DOI: 10.17148/IJARCCE.2021.101256

III. IMPLEMENTATION

III.I Sorting Algorithms

In sorting algorithms, the user has to give how many inputs and the set of data for inputs or he/she can generate random array inputs using the Generate Button. Then select the particular algorithm from the list and then the visualization of the selected algorithm is shown with the given inputs. Also, the web-based platform has the feature to visualize the algorithm as per the user's need.

A. Bubble Sort

The bubble sort algorithm makes a number of passes through the list of elements. On each pass, it compares adjacent element values. If they are out of order, they are swapped. We start each of the passes at the beginning of the list Generate button. On the first pass, once the algorithm reaches the largest element, it will be swapped with all of the remaining elements, moving it to the end of the list. The second pass will move the second largest element down the list until it is in the second to last location. The process continues with each additional pass moving one more of the larger values down in the list. If on any pass there are no swaps, all of the elements are now in order and the algorithm can stop

B. Selection Sort

The algorithm works as follows:

- Find the minimum value in the list
- Swap it with the value in the current position
- Repeat the steps above for the remaining of the list (to the consecutive positions)

Effectively, we divide the list into two parts: the sublist of items already sorted, which we build up from left to right and is found at the beginning, and the sublist of items remaining to be sorted, occupying the remainder of the array.

C. Insertion Sort

Insertion sort is a simple and efficient comparison sort. In this algorithm, each iteration removes an element from the input data and inserts it into the correct position in the list is sorted. The choice of the element being removed from the input is random and this process is repeated until all input elements have gone through.

D. Merge Sort

Merge sort is the sorting technique that follows the divide and conquers approach. It divides the given list into two equal halves, calls itself for the two halves and then merges the two sorted halves. We have to define the merge() function to perform the merging. The sub-lists are divided again and again into halves until the list cannot be divided further. Then we combine the pair of one element lists into two-element lists, sorting them in the process. The sorted two-element pairs are merged into the four-element lists, and so on until we get the sorted list.
The important part of the merge sort is the MERGE function. This function performs the merging of two sorted sub-arrays that are A[beg…mid] and A[mid+1…end], to build one sorted array A[beg…end]. So, the inputs of the MERGE function are A[], beg, mid, and end.

E. Quick Sort

It is an example of the divide-and-conquer algorithmic technique. It uses recursive calls for sorting the elements, and it is one of the most famous algorithms among comparison-based sorting algorithms.
The recursive algorithms are as follows:

- If there are one no elements in the array to be sorted, return.
- Pick an element in the array to serve as the "pivot" point.
- Split the array into two parts - one with elements larger than the pivot and the other with elements smaller than the pivot
- Recursively repeat the algorithm for both halves of the original array

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 12, December 2021

DOI: 10.17148/IJARCCE.2021.101256

III.II Pathfinding Algorithms

In Pathfinding Algorithms, the user can choose from a variety of mazes available under the Generate Maze button or the user can build its own maze through the interactive platform. Then, the user can select from various Pathfinding Algorithms and then visualization of the selected pathfinding algorithm is shown.

A. Dijkstra Algorithm

Named for its creator, Edsger Dijkstra, Dijkstra's algorithm was first proposed in 1959 and is the immediate precursor of A*. The basic process is to assign each node at a distance value, at first set to zero for the initial node and infinity for all other nodes. All nodes are marked as unvisited and the initial, node is marked as the current node. All nodes that are neighbours to the current node are examined and their distance D from the initial node is calculated through the current node is calculated. If this new distance D is less than the previously recorded distance D for that node, the new distance value replaces the old distance value for that node. When all neighbours of the current node are examined, the current node is marked as visited and will not be looked at again. The neighbour node with the lowest distance value is marked as the new current node and the process repeats until the target is marked as visited or all nodes are marked as visited without the target being found [22].

B. A* Algorithm

The A* search algorithm is generally regarded as the de facto standard in-game pathfinding. It was first described in 1968 by Peter Hart, Nils Nilsson, and Bertram Raphael.
For every node in the graph, A* maintains three values: f(x), g(x), and h(x). g(x) is the distance, or cost, from the initial node to the node currently being examined. h(x) is an estimate or heuristic distance from the node being examined to the target.
The value of g(x) is the distance from the initial node to the current node through all previous nodes traversed to get to that point. Therefore, if A* is examining node C as a possible next step in the path after traversing node B, then the g(x) value of node C is equal to the distance from the origin A to node B, plus the distance from node B to node C. This makes the g(x) value for a given node equal to the actual distance required to travel from the origin to that node, through all preceding nodes. h(x) is an estimate of the distance from the current node to the node located at the target. f(x) is the sum of g(x) and h(x).
A* also maintains an "open list," which is a list of all unvisited nodes, and a "closed list," or a list of visited nodes. At the beginning of the search, all nodes are on the open list, and the initial node is marked as current. The values of g(x), h(x), and f(x) are calculated for each of its neighbours. If the new f(x) value of a node being examined is less than the previous f(x) value for that node, the new f value replaces the old f value. The current node is moved from the open list to the closed list, the neighbour node with the lowest f(x) value is marked as the new current node and the process repeats until the target is added to the closed list, or there are no more nodes on the open list.[23].

C. Breadth-First Search

The breadth-first search was discovered by Moore in the context of finding paths through mazes. BFS is a graph traversal algorithm to explore a tree or a graph efficiently. The algorithm starts with an initial node (root node) and then proceeds to explore all the nodes adjacent to it, in a breadth-first fashion, as opposed to depth-first, which goes down a particular branch till all the nodes in that branch are visited. Put simply, it traverses the graph level-wise, not moving down a level till all the nodes in that level are visited and marked.
It operates on the first-in-first-out (FIFO) principle and is implemented using a queue data structure. Once a node is visited, it is inserted into a queue. Then it is recorded and all its children's nodes are inserted into the queue. This process goes on till all the nodes in the graph are visited and recorded.

D. Depth First Search

A version of the depth-first search was investigated in the 19th century by French mathematician Charles Pierre Trémaux as a strategy for solving mazes. The DFS search begins starting from the first node and goes deeper and deeper, exploring down until the targeted node is found. If the targeted key is not found, the search path is changed to the path that was stopped exploring during the initial search, and the same procedure is repeated for

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 12, December 2021

DOI: 10.17148/IJARCCE.2021.101256

that branch. The spanning tree is produced from the result of this search. This tree method is without the loops. The total number of nodes in the stack data structure is used to implement DFS traversal.

IV. RESULT ANALYSIS

IV.I Analysis for Sorting Algorithms

Table I depicts the comparison between various sorting algorithms that we have implemented in our web-based visualization tool. The algorithms are analysed with 8 input values with average runtime in seconds. From the table below it is clear that Selection Sort consumes less time as compared to other Sorting Algorithms. Among all the algorithms Bubble Sort will be the most time-consuming algorithm as each adjacent element are compared and are swapped as per the requirements. The entire process will be repeated for each traversal. Hence, the time complexity of Bubble Sort is worse than the others.

No. of Inputs	Time taken by Bubble Sort in secs	Time taken by Selection Sort in secs	Time taken by Insertion Sort in secs	Time taken by Merge Sort in secs	Time taken by Quick Sort in secs
5	4.6	2.9	4.5	4.0	2.4
6	5.3	3.8	4.7	4.3	2.6
7	6.3	4.9	4.9	4.9	2.9
8	7.4	6.1	5.0	5.2	3.4

TABLE I SORTING ALGORITHMS COMPARISON WITH AVERAGE VALUES

IV.II Analysis for Pathfinding Algorithms

An efficient algorithm is one that calculates the shortest path with the fewest number of node visitations. Among all the Pathfinding Algorithms, Dijkstra's Algorithms is least efficient as it has no method of cutting down on search space and it made far more node visitations during each path calculation phase. Depending, on the situation A* and D* are the most efficient pathfinding Algorithms

CONCLUSION AND FUTURE WORK

The developed platform offers a complete perspective of visualizations for sorting and pathfinding algorithms so far. Also, the web-based platform can run on small devices while providing the feature to visualize at one's own pace.

As a future scope, more Algorithms for Trees, Graphs, Linked List and many more can be added. To empower learning of the learner concept of community learning through forums, discussion and user-based feedback can be added. The objective of the platform is to reduce the fear level in the minds of learners especially students regarding Algorithms and Data Structures.

REFERENCES

[1] Crescenzi, Pilu, Malizia, Alessio, Verri, M Cecilia, Díaz, Paloma, & Aedo, Ignacio. (2012). Integrating Algorithm Visualization Video into a First-Year Algorithm and Data Structure Course. Educational Technology & Society, 15(2), 115-124.

[2] Osman, Waleed Ibrahim, & Elmusharaf, Mudawi M. (2014). Effectiveness of Combining Algorithm and Program Animation: A Case Study with Data Structure Course. Issues in Informing Science and Information Technology, 11.

[3] Sadikan, Siti Fairuz Nurr, & Yassin, Siti Fatimah Md. (2012). Role of Interactive Computer Programming Courseware in Facilitating Teaching and Learning Process Based on Perception of Students in Bangi, Selangor, Malaysia. Learning, 3, 0.51.

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 12, December 2021

DOI: 10.17148/IJARCCE.2021.101256

[4] S. Hansen, N. H. Narayanan, and M. Hegarty, "Designing Educationally Effective Algorithm Visualizations", Journal of Visual Languages and Computing, vol. 13, no. 3, 2002, pp. 291-317.
[5] J. Stasko, "Samba algorithm Animation System", http://www.cc.gatech.edu/gvu/softviz/algoanim/samba.html
[6] W. C. Pierson, and S. H. Rodger, "Web-based animation od data structures using JAWAA", ACM SIGCSE Bulletin, vol. 30, no. 1, 1998, pp. 267-271.
[7] Urquiza-Fuentes, J., & Velázquez-Iturbide, J. N. (2009). A Survey of Successful Evaluations of Program Visualization and Algorithm.Animation Systems. ACM Transactions on Computing Education, 9(2), 1–21. https://doi.org/10.1145/1538234.1538236
[8] Mishra, L., Gupta, T., & Shree, A. (2020). Online teaching-learning in higher education during the lockdown period of COVID-19 pandemic. International Journal of Educational Research Open, 1,100012. https://doi.org/10.1016/j.ijedro.2020.100012.
[9] Abu Naser, S. S. (2008). Developing Visualization Tool for Teaching AI Searching Algorithms. Information Technology Journal, 7(2), 350–355. https://doi.org/10.3923/itj.2008.350.355
[10] Furcy, David. (2009). JHAVEPOP: visualizing linked-list operations in C++ and Java. Journal of Computing Sciences in Colleges. 25. 32-41.
[11] Halim, S. (2011). VisuAlgo - visualising data structures and algorithms through animation. VisuAlgo. https://visualgo.net/en
[12] Halim, S., Halim, F. (2011). Competitive Programming 2: This increases the lower bound of programming contests. Available: http://www.lulu.com
[13] Shaffer, C. A., Akbar, M., Alon, A. J. D., Stewart, M., & Edwards, S. H. (2011). Getting algorithm visualizations into the classroom. Proceedings of the 42nd ACM Technical Symposium on Computer Science Education - SIGCSE '11, 121. https://doi.org/10.1145/1953163.1953204
[14] Cooper, M. L., Shaffer, C. A., Edwards, S. H., & Ponce, S. P. (2014). Open-source software and the algorithm visualization community. Science of Computer Programming, 88, 82–91. https://doi.org/10.1016/j.scico.2013.12.008.
[15] Šimonak, S. (2014). Using algorithm visualizations in computer science education. Open Computer Science, 4(3). https://doi.org/10.2478/s13537-014-0215-4
[16] Borissova, D., & Mustakerov, I. (2015). E-learning Tool for Visualization of Shortest Paths Algorithms. Trends Journal of Sciences Research, 2(3), 84–89. https://doi.org/10.31586/informationprocesses.0203.01
[17] Jonathan, F. C., Karnalim, O., & Ayub, M. (2016). Extending The Effectiveness of Algorithm Visualization with Performance Comparison through Evaluation-integrated Development. Seminar Nasional Aplikasi Teknologi Informasi (SNATi), 16–21. https://journal.uii.ac.id/Snati/article/view/6263
[18] Supli, A. A., & Shiratuddin, N. (2017). Designing algorithm visualization on mobile platforms: The proposed guidelines. https://doi.org/10.1063/1.5005943
[19] Romanowska, K., Singh, G., Dewan, M. A. A., & Lin, F. (2018). Towards Developing an Effective Algorithm Visualization Tool for Online Learning. 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI). https://doi.org/10.1109/smartworld.2018.00336
[20] Bruno, P. (2021). Recursion Tree Visualizer. Recursion Tree Visualizer. https://recursion.vercel.app/.
[21] Aniket B. Ghandge, Bhagyashree P., Hrithik R., Prateek S, Parmod B.(2021).AlgoAssist: Algorithm Visualizer and Coding Platform for Remote Classroom Learning.2021 5th International Conference on Computer, Communication and Signal Processing (ICCCSP - 2021).http://dx.doi.org/10.1109/ICCCSP52374.2021.9465503
[22] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs", June 11, 1959.http://www-m3.ma.tum.de/foswiki/pub/MN0506/WebHome/dijkstra.pdf
[23] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", July 1968.http://www.cs.auckland.ac.nz/compsci709s2c/resources/Mike.d/astarNilsson.pdf