JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

VIZALGO

Km Abha dohre[1], Prof. Kaveri kari[2], Rucha Barbole[3], Rutvik sutar[4], Manvi Saini[5]
Dept of Information Technology Engineering
[1]GenbaSopanraoMoze College of Engineering,,Balewadi.

Abstract- *In today's data-driven world, aalgorithms play a fundamental role in solving complex problems across various domains. However, the This paper presents an overview of visual algorithms and their significance in facilitating algorithmic understanding. We explore different types of visualizations employed in visual algorithms, such as flowcharts, graphs, and animations, which help in visualizing the steps, data flow, and decision-making processes of algorithms. abstract nature of algorithms often poses challenges for both experts and novices in comprehending and analyzing their inner workings. To address this issue, researchers and developers have turned to visual representations as a means to enhance algorithmic understanding. Visual algorithms combine the power of visualizations with the logical structure of algorithms, providing an intuitive and interactive framework for problem-solving and analysis.*

II. SYSTEM ARCHITECTURE

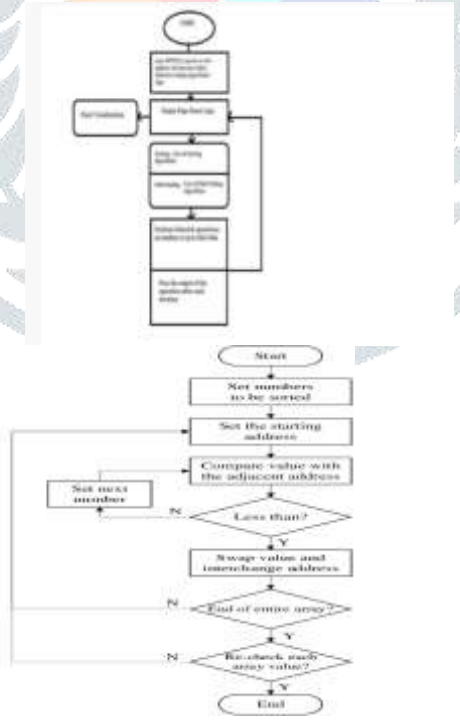


Fig: Block diagram of sorting visual algorithm

Keywords- *graphical way, algorithm operation*

I. INTRODUCTION

Visual algorithms, also known as algorithm visualization or algorithm animation, are techniques used to represent and communicate the behavior and execution flow of algorithms visually. They provide a graphical representation of complex algorithms, making it easier for individuals to understand and analyze their inner workings. The use of visual algorithms is particularly beneficial in the field of computer science and programming, where algorithms play a crucial role in solving various computational problems. Traditional textual representations of algorithms, such as pseudocode or code snippets, can sometimes be challenging to comprehend, especially for beginners or individuals who are not familiar with a specific programming language.

III. MPLIMENTATION

Sorting:

Sorting is a fundamental operation in computer science and refers to the process of arranging a collection of items or refers to the process of arranging a collection of items or data elements in a particular order. The order could be ascending or descending, based on a defined criterion such as numerical value, alphabetical order, or any other comparison metric.

Bubble Sort: This algorithm repeatedly compares adjacent elements and swaps them if they are in the wrong order until the entire list is sorted.

Selection Sort: In this algorithm, the list is divided into two portions: the sorted and the unsorted portion. The algorithm repeatedly selects the smallest element from the unsorted portion and swaps it with the leftmost unsorted element.

Insertion Sort: This algorithm builds the final sorted list one item at a time. It iterates over the list, comparing each element with those before it and inserting it into the correct position within the sorted portion of the list..

Merge Sort: It is a divide-and-conquer algorithm that divides the list into smaller sublists, recursively sorts them, and then merges them to obtain a sorted list. It typically has a better average and worst-case time complexity compared to the previous algorithms.

Quick Sort: Another divide-and-conquer algorithm, Quick Sort selects a "pivot" element from the list and partitions the other elements into two sublists, according to whether they are less than or greater than the pivot. The algorithm then recursively sorts the sublists. Quick Sort is known for its efficiency and is widely used.

Heap Sort: This algorithm uses a binary heap data structure to sort elements. It first creates a max-heap (for ascending order) or min-heap (for descending order) from the elements and then repeatedly extracts the top element from the heap and restores heap properties until the list is sorted.

Dijkstra Algorithm

The basic process of dijkastra algorithm to assign each node at a distance value, at first set to zero for the initial node and infinity for all other nodes.

Algorithm Pseudocode:

1. Mark the ending vertex with a distance of zero. Designate this vertex as current.
2. Find all vertices leading to the current vertex. Calculate their distances to the end Since we already know the distance the current vertex is from the end, this will just require adding the most recent edge. Don't record this distance if it is longer than a previously.
3. Mark the current vertex as visited. We will never look at this vertex again.
4. Mark the vertex with the smallest distance as current, and repeat from step 2.

A* Algorithm:

A* Search algorithm is one of the best and popular technique used in path-findinhg and graph traversals.

A* so powerful is the use of weighted graphs in its implementation. A weighted graph uses numbers to represent the cost of taking each path or course of action.

Algorithms Pseudocode:

1. Place the starting node in the OPEN list.
2. Check if the OPEN list is empty or not, if the list is empty then return failure and stops.
3. Select the node from the OPEN list which has thesmallest value of evaluation function (g+h), ifnode n is goal node then return success and stop, otherwise.
4. Expand node n and generate all of its successors, and put n into the closed list. For each successor n', check whether n' is already in the OPEN or CLOSEDcompute evaluation function for n' and place into Open list
5. Else if node n' is already in OPEN and CLOSED, then it should be attached to the back, pointer which reflects the lowest g(n') value.
6. Return to Step 2.

Breadth-First Search:

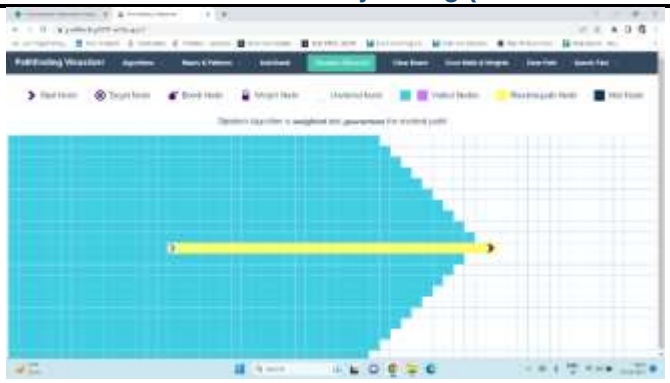
BFS is a traversing algorithm where you should start traversing from a selected

node (source or starting node) and traverse the graph layerwise thus exploring the neighbour nodes (nodes which are directly connected to source node.

The purpose of the algorithm is to mark each vertex as visited while avoidingcycles.

Algorithms Pseudocode:

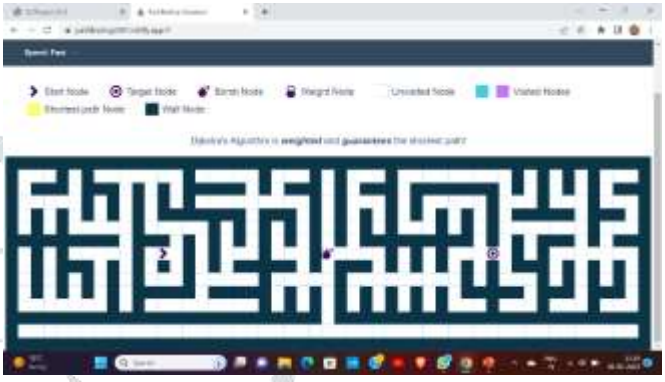
1.Declare a queue and insert the starting vertex.
2.Initialize a visited array and mark the starting vertex as visited.
3.Follow the below process till the queue becomes empty. Remove the first vertex of the queue.
4.Mark that vertex as visited.
5.Insert all the unvisited neighbours of the vertex into the Queue.



Depth First Search:

Depth-first search is an algorithm for traversing or searching tree or graph data structures.

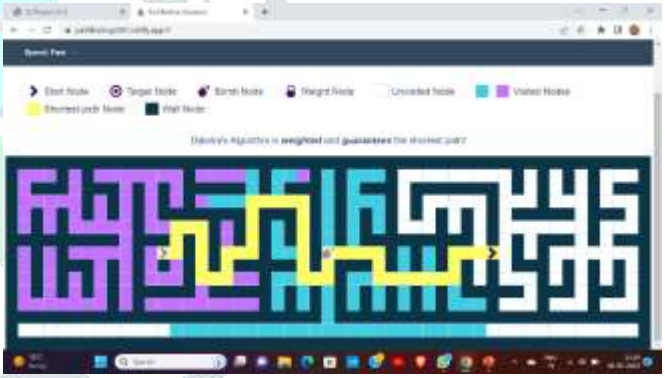
The algorithm starts at the root node and explores as far as possible along each branch before backtracking.

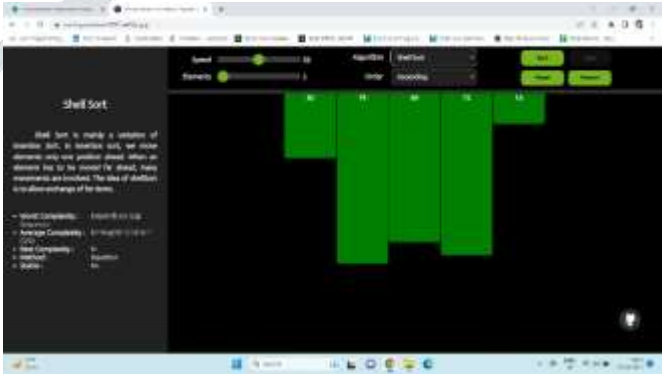


Algorithms Pseudocode:

1.Create a recursive function that takes the index of the node and visited array.
2.Mark the current node as visited and print the node.
3.Traverse all the adjacent and unmarked nodes and call the recursive function with the index of the adjacent.

SCREENSHOTS TS OF SOME OUTPUTS





CONCLUSION

The Proposed System offers a complete perspective of visualisation for sorting and

path finding algorithm.

future plan is to add more data structures visualizations. Along with that we aim to improve the UI/UX of the website.

As a future scope, more Algorithms for Trees, Graphs, Linked Lists and many more can be added. To empower the system's objective is to reduce the fear level in the minds of learners especially students regarding Algorithms and Data Structures.

One key conclusion about visual algorithms is their significant impact on numerous industries.

REFERENCES

[1]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.

[2] Furcy, David. (2009). JHAVEPOP: visualizing linked-list operations in C++ and Java. Journal of Computing Sciences in Colleges. 25. 32-41

[3] 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

[4] Supli, A. A., & Shiratuddin, N. (2017). Designing algorithm visualization on mobile platforms: The proposed guidelines.
https://doi.org/10.1063/1.5005943

[5] 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