iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** V **Month of publication:** May 2023

DOI: https://doi.org/10.22214/ijraset.2023.52237

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue V May 2023- Available at www.ijraset.com

Visualizer for Algorithms and Path Finding with User Interface Design

Leena I. Sakri[1], Deepa Bendigeri[2], Sachin Joshi[3], Arpita Hiremagadi[4], Pooja Gurumurti[5], Keerti Angadi[6], Shraddha Kittur[7]
[1, 2, 3, 4, 5, 6, 7]*S D M College of Engineering and Technology, Dharwad*

Abstract: ***A graphical description of how algorithms like sorting and path finding work is provided by the Algorithm and Path Finding visualizer. The system thoroughly explains each process using both text and animation. Dots, lines, dimensional bars, and other graphic elements are combined to create an algorithm visualisation, which shows some of the algorithm's more "interesting events."***
The benefits of the algorithm visualizer include a user-friendly UI. It's easy to use the UI. Users move easily and without exertion. input of data entry by users is possible in the system. User command Users can adjust the speed of the algorithm animation to make it simpler or harder for them to grasp the backend functionality depending on how quickly they learn and process new information, and it will show the space and time complexity of each algorithm.

I. INTRODUCTION

Students need to acquire a solid grasp of complex topics like algorithms since they form the foundation for their computational thinking and programming skills. We had seen that using traditional teaching techniques made it more challenging for teachers and students to articulate their ideas. Since, as the saying goes, "a picture is worth a thousand words," many Academics and educators believe that implementing algorithm visualization techniques can help students learn an algorithm more quickly and thoroughly. Therefore, we created a teaching technique that helps students as well as teachers learn about various searching and sorting algorithms through visualization and hands-on experience.

A well-done visualization brings algorithms to life by graphically displaying their multiple states and animating the transitions between them. More specifically, dynamic algorithm visualization plays an algorithm's actions like a continuous movie. The human visual system provides a better way of understanding these important conceptual processes through visualisation. Researchers have come up with several techniques to get around these obstacles, including graphical representations, traditional animations, explanatory films, etc.

These techniques help the student better understand the concepts. Algorithmic visualization is a technique that has advanced in the past and is gaining popularity in computer science. With simple animations and transition effects, the Algorithm Viewer is a useful tool. a tool that allows students to see how the algorithm works at each step.

II. METHODOLOGY

This paper includes different algorithms: sorting, searching, pathfinding, trees, travelling salesman, minimum spanning tree, binary heap, and graph traversal. The following section includes the methodology used to visualize different types of algorithms.

Essential functions of the algorithm visualizer are as follows:

1) The system shall take inputs in array format.
2) The system will take user inputs otherwise it will take default inputs.
3) When a user enters new data, the dataset should be updated with the new data.
4) It will sort array using different algorithms which is selected by users.
5) It shows the step-by-step process of a algorithms with bar graphical animation.
6) Through this paper every student can learn at their own pace with our three speeds of learning: slow, medium, and fast.

A. Architectural design

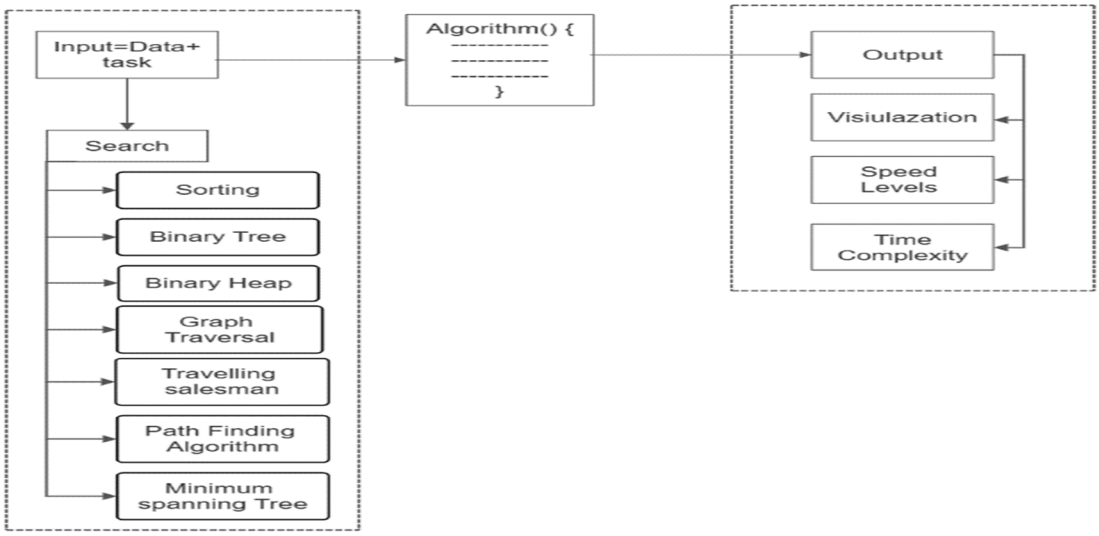


Figure:1.1 Architectural design

Algorithm and Path Finding Visualizer's architectural design is divided into several stages. Algo Magic has a Welcome Page that includes sorting algorithms, a binary heap, a tree, a graph traversal algorithm, a minimum spanning tree, path finding algorithms, and a travelling salesman. The subheads in that are contained in each of the algorithms. The algorithms are seen when a user clicks the subheading. A new array is first generated by the user, or else the default array size and items are used. The array size and sorting speed are user-adjustable for convenience. The user may quickly comprehend each of the algorithms by selecting either the Visualize or the Tutorial View.

B. Sorting Algorithm

Sorting section includes different sorting The sorting section includes different sorting algorithms like bubble sort, insertion sort, merge sort, quick sort, radix sort, selection sort and counting sort. When the user gives an input, the input will be taken as an array and displayed using a graphical representation with bar graphs. Each bar's height is equal to the numerical value it represents.

Different colours are used for bars between the currently sorted and unsorted arrays. Both the array size and sorting speed can be adjusted as per user demand, which also affects speed. The user also has the option of choosing any algorithm from the list. After sorting, the bars colours will change to a single colour, and the elements of the array will be rearranged in ascending order.

1) *Path-finding Algorithm (SSSP):* Different algorithms are used, including Dijkstra's algorithm and the A* algorithm. The user's algorithms were chosen based on their complexity and level of popularity. After the visualization is complete, the viewer will be able to differentiate between the functionality of several algorithms based on time complexity. Mazes and patterns are used to provide a better and clearer understanding of algorithms. Users can relate the visualization to situations in the real world since there will be walls or other impediments between the starting node and destination node. The user will also be able to select the best method based on algorithmic time complexity. It features a speed option that enables the user to customise the speed of the visualization based on personal preferences. Because everyone learns at a different rate, this functionality was added.

C. Binary Heap

A full binary tree that upholds the Max Heap property is known as a "binary (max) heap". One potential data structure for simulating an effective priority queue (PQ) using an abstract data type is the binary heap. This visualization displays a binary maximum heap of integers with allowable duplications to narrow the discussion's focus. For a simple conversion to binary min heap, see this. The binary (max) heap is implemented internally as a compact array, but the user can choose between this and a more visually intuitive whole binary tree representation. In this diagram, you can carry out a variety of binary (max) heap operations, such as

1) Create (A) – O (N log N) version,
2) Create (A) – O (N) version

3) Insert (v) in O (log N),
4) Versions of Extract Max (),
5) Update Key, and so on.

D. Graph Traversal (Searching)

A linear search starts at one end of a list and moves sequentially across it, checking each item until it is located and going to the next if necessary.

The search interval in a binary search is repeatedly cut in half. Utilizing the knowledge of a sorted array, binary search attempts to decrease the time complexity to O (log n). There are two methods of graph traversal: the depth-first search algorithm (DFS) and breadth-first search.

DFS only accepts one input variable: the origin vertex is s. The breadth first search (BFS). This visualization has many DFS and BFS versions, all of which are executed, including a binary tree that meets the next BST, the bipartite graph checker algorithm, and the topological sort algorithm (both DFS and BFS versions are employed).

E. Trees (BST, AVL)

A binary tree that satisfies the following BST properties is known as a "binary search tree" (BST): Because it was assumed that all the numbers in this visualization are distinct integers, a minor adjustment is required to account for duplicates and non-integer values. Every vertex must have vertices in its left subtree containing values that area lower than its own and every vertex must have vertices in its right subtree having values that are greater than its own.

A balanced BST, such as the AVL tree, is an efficient data structure for creating a certain form of table-based Abstract Data Type (ADT).

F. Minimum Spanning Tree

The spanning tree is an undirected, weighted, connected graph like a tree. Every vertex in subgraph G is connected to every other vertex. The overall weight of each spanning tree in graph G can vary.

Kruskal's approach is a greedy minimum spanning tree algorithm that first creates a forest of MSTs and then combines them into a single MST with the shortest overall weight. For Kruskal's to work, the input graph's edges, which are often kept in an edge list data structure, must be sorted by non-decreasing weight. Union-find disjoint sets (UFDS) is a data structure useful for detecting and preventing cycles.

Prime's strategy: Another O (E log V) greedy MST method is Prim's, which involves gradually expanding a minimum-spanning tree outwards from a single source vertex until it covers the entire graph. Prims uses a direct-addressing table (a Boolean array), a priority queue data structure, and an effective vertex neighbour enumeration technique to dynamically organise the edges under consideration based on non-diminishing weight.

G. Travelling Salesman

The Travelling Salesperson Problem (TSP) seeks to identify a tour with the lowest possible cost that stops in each city just once. This suggests that the distance function satisfies the triangle inequality, and for the sake of this visualization, the underlying graph is a complete graph with (near-) metric distance. It is possible to provide input using two sources.

Draw Graph: To maintain the near-metric property of the graph, you can provide a number of points on the drawing box, but you must avoid drawing any edges when you are done placing the points, the edges will be automatically drawn for you. Example Graphs: To get you started, pick one from list of examples.

Brute force: Since it doesn't matter where it starts, it only tests the (V-1) permutation of vertices using brute force and enumerates all possibilities. O(V!) = O(V*(V-1)!) is the time complexity.

Dynamic programming: This method utilises the well-known Held-Karp algorithm. Like the brute-force method, this visualization uses RAM to cache the DFS search results. As a result, the run-time complexity is reduced to O (2 V * V2). The time complexity is O (2V * V2).

III. RESULT AND DISCUSSION

Sorting algorithm analysis: The comparison of the various sorting algorithms is used in this web-based visualization tool, as shown in Table 1. Which algorithms have different input values and an average runtime in seconds 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 because it compares and swaps each neighbouring element according to the specifications. For each traversal, the entire process will be repeated.as a result Bubble Sort has a higher temporal complexity than the others.

Table 1: Sorting algorithm comparison

Quantity of inputs	Duration of Bubble Sort in secs	Duration of Selection Sort in secs	Duration of Insertion Sort in secs	Duration of Merge Sort in secs	Duration of Quick Sort in secs
06	5.2	3.6	4.7	4.2	2.4
07	6.0	4.5	4.4	4.4	2.1
08	7.5	6.3	5.0	5.1	3.2
09	8.2	7.4	5.3	5.2	3.5

Pathfinding Algorithm Analysis: A successful 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. The A* algorithm is one of the most effective path finding algorithms, depending on the circumstances. Here, users are welcome to contribute and can choose an algorithm from a list while the algorithm is visually explained. A time complexity parameter is also provided, which will be displayed once the respective pathfinding algorithm has completed its task and can be used for comparison purposes.

The User interface of Algorithm and Path finding Visualize

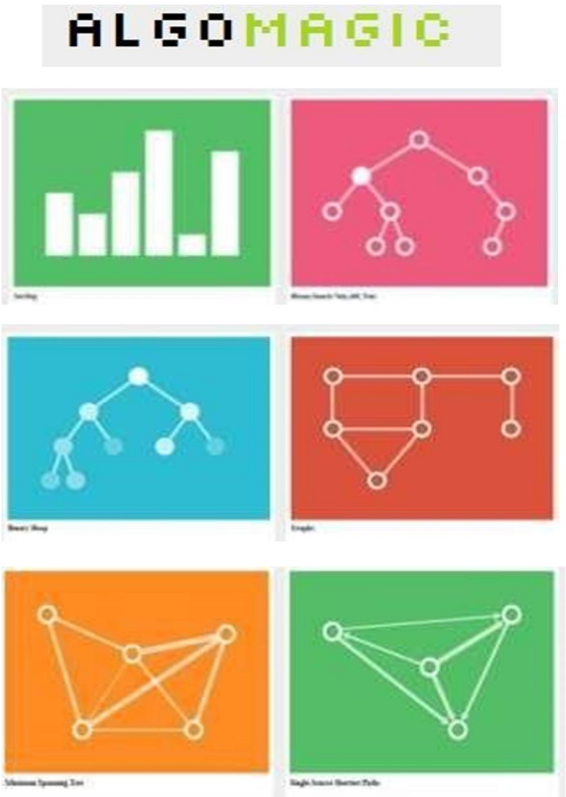


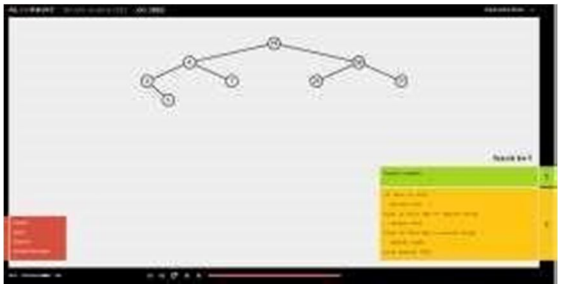
Figure:1.2 User Interface

First, the Welcome Page will appear, which includes sorting algorithms, binary heaps, trees, graph traversal algorithms, minimum scanning trees, and path finding algorithms. Each of the algorithms contains subheadings. The algorithms are visualized when a user selects the subheading. The user can provide new input values for visualization or else use the default input values. The array size and sorting speed are user-adjustable for ease. The users can adjust the speed of the algorithms as per their convenience. The user can understand each of the algorithms by selecting either the Visualize or the Tutorial View. when the user click n one of the algorithms. The inputs are controlled and managed by users, or by default it will take inputs in array format. Users can adjust the speed levels, such as slow, medium, and fast. User can make use of tutorial mode option when they are not aware of a particular algorithm. The pseudo code is visible while visualizing the algorithm.

A. Sorting

A sorting algorithm will help rearrange a given array of elements according to a comparison operator on the elements

B. Binary Search Tree



The node-based binary tree data structure known as the "Binary Search Tree" includes the following characteristics: A node's left sub tree only has nodes with keys lower than the node itself. A node's right sub tree only has node with keys higher than the node itself. A binary search tree must also be present in both left and right sub trees. The AVL tree is a self-balancing binary search tree in which there can never be more than one difference between the heights of the left and right sub trees for any given node.

C. Binary Heap

Either the Min Heap or the Max Heap is a binary heap. The root key of a min binary heap must rank lowest among all other keys in the heap. All nodes in a binary tree must have the same property, recursively true. Min Heap and Max Binary Heap are comparable.

D. Minimum Spanning Tree

A minimum spanning tree (MST), or spanning tree with a weight less than or equal to the weight of every other feasible spanning tree, is the minimal weight spanning tree for a weighted, linked, undirected graph. A spanning tree's weight is the total of the weights assigned to each of its edges. In a nutshell, MST is the spanning tree with the lowest weight among all the spanning trees in a given graph.

E. Single source shortest path

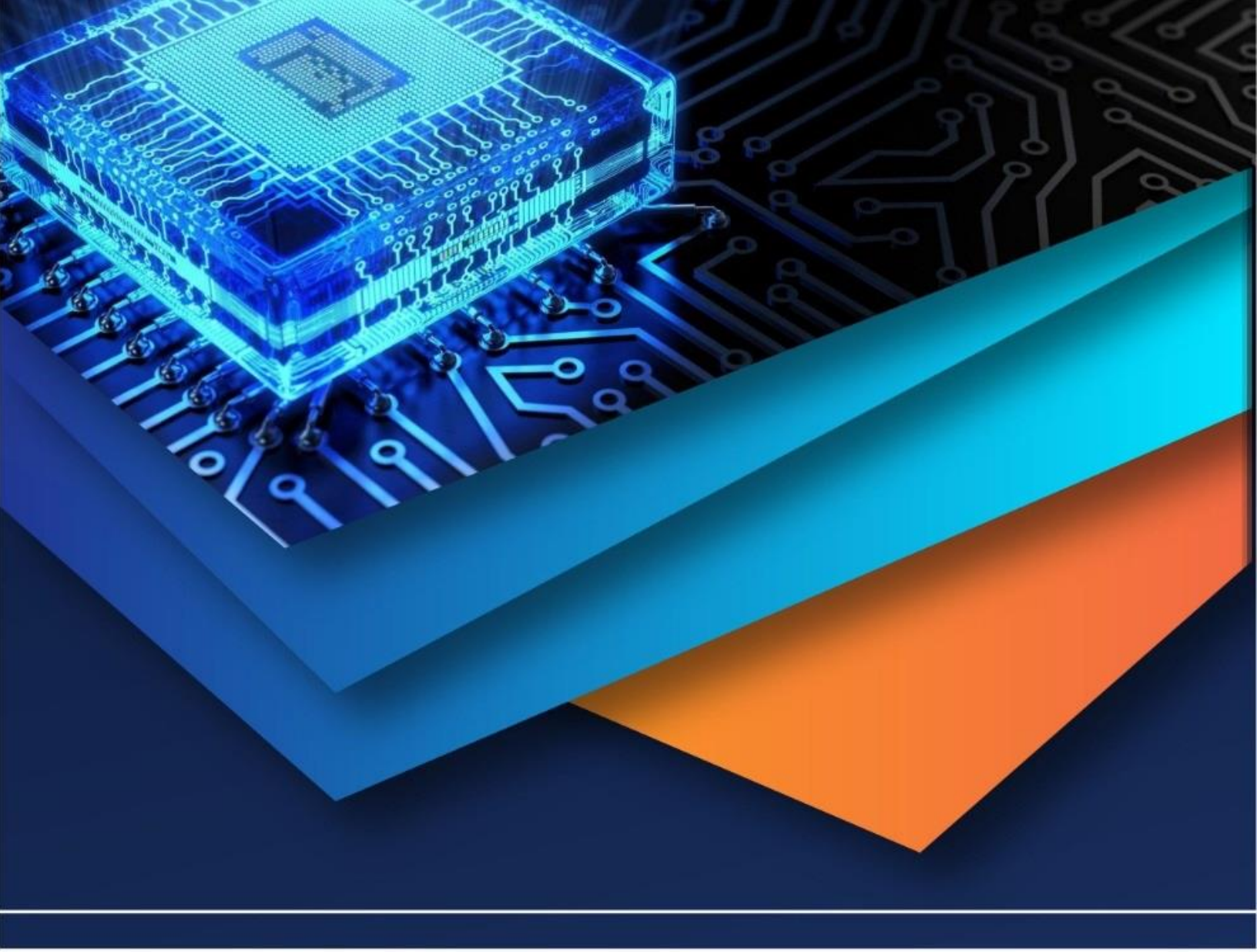
The single-source shortest path (SSSP) problem consists of finding the shortest paths between a given vertex v and all other vertices in the graph. Algorithms such as Breadth-First-Search (BFS) for unweighted graphs or Dijkstra [1] solve this problem.

IV. CONCLUSION

Several searching and sorting methods are visualized using graphical and animation techniques. It is a useful tool for students and researchers to quickly comprehend the implicit algorithmic sequences. Here, users are welcome to choose an algorithm from a list while the algorithm is visually explained. It has been discovered the learning through animations rather than merely text or oral explanations makes it easier for humans to remember concepts. The method is designed to be easily accessible. Students can easily understand the time and space complexity as well.

REFERENCES

[1] Algorithm Visualization - Modern Web-Based Visualization of Sorting & Searching Algorithms Ashish Kumar, Manav Mittal, Vipul Jha, Abhishek Sahu, Manish Kumar, Neeti Sangwan and Navdeep BOHRA, Advances & Applications in Mathematical Sciences Volume 21, Issue 5, March 2022.
[2] Algorithm Visualizer Barnini Goswami, Anushka Dhar, Akash Gupta, Antriksh Gupta International Research Journal of Modernization in Engineering Technology & Science eISSN:2582-5208Volume:03/Issue:03/ March-2021
[3] Algorithm Visualizer ICONIC RESEARCH & ENGINEERING JOURNALS Prof. Asha P, Gurpreet Kaur, Anmolpreet S, Shashank K, Akshay Kumar5 JUL 2022 | IRE Journals | Volume 6 Issue 1 | ISSN: 2456-8880.
[4] Sorting Algorithm Visualizer Shubham Nath, Jatin Gupta, Abhinav Gupta, International Research Journal of Modernization in Engineering Technology and Science. eISSN:2582-5208 Volume:03/Issue:07/July-2021
[5] A Path Finding Visualization Using Algorithm and Dijkstra's Algorithm International Journal of Trend in Scientific Research and Development (IJTSRD) Volume 5 Issue 1, November-December 2020.
[6] Node Path Visualizer Using Shortest Path Algorithms Deep Singh1, Brahmbind Singh2, Gagandeep Singh3, Harleen Kaur4, Kanwar Jeet Singh5 International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395- 0056 Volume: 09 Issue: 06 Jun 2022.
[7] Research on path planning of three neighbour search A* algorithm combined with artificial potential field Jiqing Chen1,2, Chenzhi Tan1, Rongxian Mo1, Hongdu Zhang1 International Journal.
[8] Simulation of Sorting Algorithms Shubhad Vishnu Parabi, Sharvari R Pednekar, Tejaswini Rajesh IRE Journals Volume 5 Issue 10 | ISSN: 2456-8880.
[9] The development of system for algorithms visualization using simjava Journal of Engineering and Applied Sciences Jamil Abedal rahim Jamil Alsayaydeh, Maslan Zainon, A. Oliinyk6, Azwan Aziz,.Rahman, Zikri Abadi Baharudin1VOL. 15, NO. 24, December 2020.
[10] Towards Developing an Effective Algorithm Visualization Tool for Online Learning 2018 IEEE Smart World, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY