"SIMULATION BASED LEARNING SYSTEM FOR DATA STRUCTURES AND ALGORITHMS"

Prof. A. D. GAWALI
Mr. DATTATRAYA JANKIRAM KATKHADE
Miss. RITUJA YOGESH PURANIK
Miss. SWAPNALI RAOSAHEB JAWALE
Mr. OWES NOORMOHMMAD SHAIKH

DEPARTMENT OF INFORMATION TECHNOLOGY

AMRUTVAHINI COLLEGE OF ENGINEERING, SANGAMNER

ABSTRACT

To understand the data structures and algorithms, we often employ some kind of visualization like drawing it on paper, use of images and videos for understanding purposes. But while going through only pseudo-code and explanation of algorithm and trying to imagine how it works it becomes difficult and frustrating sometimes.

Visualizations are always better as compared to textual data for understanding purposes. The same problem can be resolved using the web application AlgoViz which will visualize data structures and algorithms in the form of animations. This will help teachers and students to identify problematic areas/steps and work on improving them.

Keywords: Visualization, Data, AlgoViz, Pseudo-code

Introduction:

In the mathematical and empirical analyses of algorithms, there is yet a third way to study algorithms. It is called ***algorithm visualization*** and can be defined as the use of images to convey some useful information about algorithms. That information can be a visual illustration of an algorithm's operation, of its performance on different kinds of inputs, or of its execution speed versus that of other algorithms for the same problem. To accomplish this goal, algorithm visualization uses graphic elements—points, line segments, two- or three-dimensional bars, and so on—to represent some "interesting events" in the algorithm's operation.
There are two principal variations of algorithm visualization:
1] Static algorithm visualization
2] Dynamic algorithm visualization, also called algorithm animation

Static algorithm visualization shows an algorithm's progress through a series of still images. Algorithm animation, on the other hand, shows a continuous, movie-like presentation of an algorithm's operations. Animation is an arguably more sophisticated option, which, of course, is much more difficult to implement.
Early efforts in the area of algorithm visualization go back to the 1970s. The watershed event happened in 1981 with the appearance of a 30-minute colour sound film titled *Sorting Out Sorting*. This algorithm visualization classic was produced at the University of Toronto by Ronald Baecker with the assistance of D. Sherman [Bae81, Bae98]. It contained visualizations of nine well-known sorting algorithms (more than half of them are discussed later in the book) and provided quite a convincing demonstration of their relative speeds.

Motivation:

Many students find it difficult to understand how data structures work because it requires abstract thinking. Imagining algorithms only with the help of textual data is time-consuming and confusing many times. There are a wide variety of algorithm visualizers which deal with some specific algorithm.

It would be very helpful if there was a one-stop visualization tool of data structures such as arrays, queues, stacks, trees, and graphs for students to manipulate. This motivates us to bring the idea that overcomes this problem by providing them a simple solution.

Scope:

The rundown of presently available modules includes visualizations of searching algorithms like linear search, binary search, jump search, and some sorting algorithms like the bubble sort, insertion sort, selection sort, and Quick sort. Visualization for basic data structures like the stack, queue, and binary search tree is also available.

The extension can be made on different visual views on running algorithm or simultaneous comparison ofifferent algorithm visualizations.

We can add some more complex data structures like heap, priority queue, segment tree, etc. The future implementation will include visualization of complex graph algorithms and greedy algorithms like job sequencing, etc. Also, we can implement a system in which the system can check its knowledge opposite to system-generated implementation.

Expected Outcome:

- To visualize basic data structures stack, queue, and binary search tree
- To visualize sorting and searching algorithms
- To analyse understanding with the help of the test section
- To provide a one-stop solution for visualizing data structures and algorithms
- To enhance traditional learning of complex concepts

RELATED THEORY AND PROBLEM DEFINITION

Problem Definition:

The basic idea behind the project is to develop data structures and algorithms visualization platform called AlgoViz. This platform is intended to be used as a support tool for the subject data structures and algorithms taught in engineering studies. The application is built using algorithms concepts and object-oriented concepts for developing data structure. The target audience for our application will be students for learning purposes, teachers for teaching purposes, and people who are curious about how the algorithms works.

Related Theory

To understand the data structures and algorithms, we often employ some kind of visualization like drawing it on paper, use of images and videos for understanding purposes. But while going through only pseudo-code and explanation of algorithm and trying to imagine how it works it becomes difficult and frustrating sometimes.

Visualizations are always better as compared to textual data for understanding purposes. The same problem can be resolved using the web application AlgoViz which will visualize data structures and algorithms in the form of animations. This will help teachers and students to identify problematic areas/steps and work on improving them.

DESIGN METHODOLOGY

Proposed system Architecture:

The proposed application will serve as a teaching and learning tool with 3 major components.

1] Testing Mode: Data structures and algorithm operations will be demonstrated step by step on random data set generated by the system in which each step is narrated as a status in detail.

2] Learning Mode: Users can give their data as input and users can perform necessary operations to get the correct final output. They can test their knowledge by replicating the algorithm's steps.

3] Analysis Mode: Users can give random tests provided on the platform to test their knowledge about data structures and algorithms.

Including these three modules, the application is using a database server for storing user's session information. This whole setup is deployed on the Kubernetes - an open-source container orchestration system for automating web application deployment, scaling, and management.

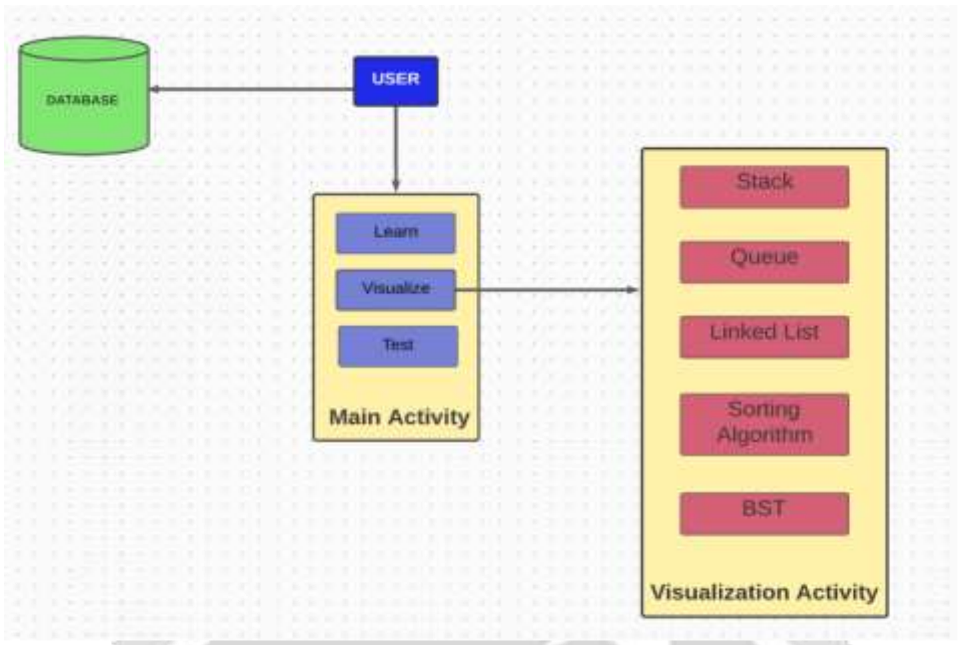


Fig. 1 Proposed architecture of AlgoViz

Data Flow and UML Diagram
Data Flow Diagram:

Data flow diagram maps the flow of information for any process or entity. The DFD diagram has predefined symbols such as rectangle, circle, double arrows with text to show entities, processes, and flow of data within the system. The DFD flow can vary from a simple hand-drawn path to a multilevel in-depth data flow.

Level 0 DFD:
Level 0 DFD shows the simple path of data within the system. The data path shows the flow from server to client. The algorithm visualizer captures the inputs from hardware and server and it shows output/visualizations to the client.

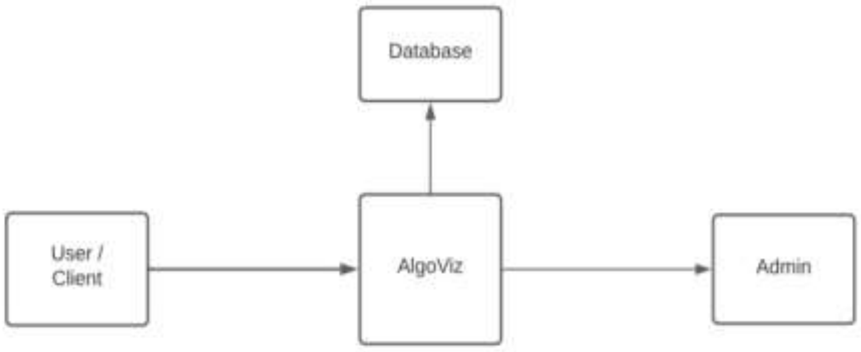


Fig. 2 Level 0 DFD

Level 1 DFD:

Level 1 DFD elaborates the level 0. It shows the data flow within the system with more details. In level 1, the entity's database or server sends data to the system. Learning mode and Testing Mode are available at level 1. Desktop systems and web applications display visualizations to the users.

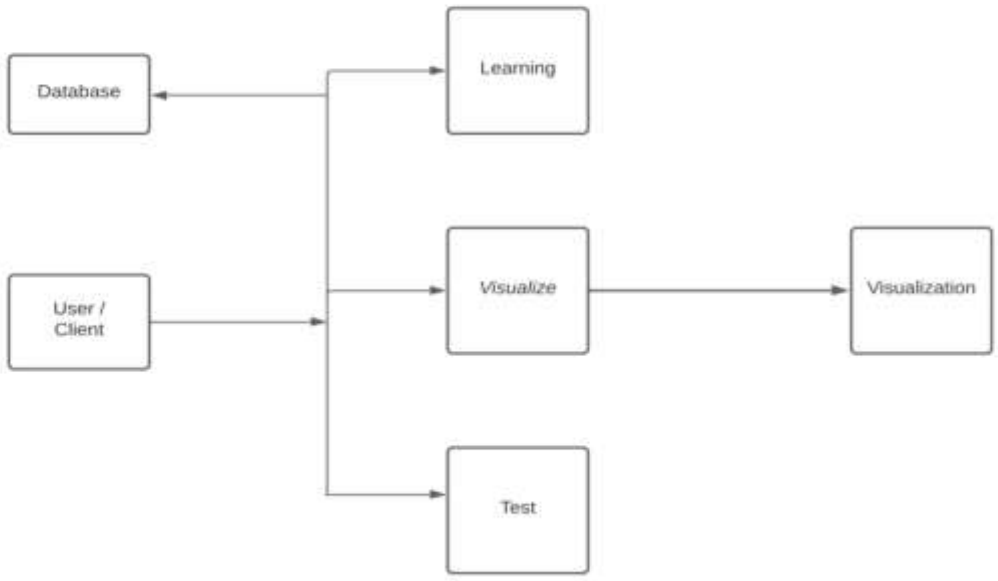


Fig. 3 Level 1 DFD

Level 2 DFD:

Level 2 DFD shows the most detailed path of data within the system. It elaborates the level 1 of DFD. The entities involved in level 2 are client, On-screen data, Visualization of data structures, and Algorithms. Firstly, the system does the authentication of the server with help of auth keys.
User login into the system with their credentials. They will access mainly three modes learning mode, test mode, and explore mode. The explore mode is containing the different data structures and algorithms to visualize.

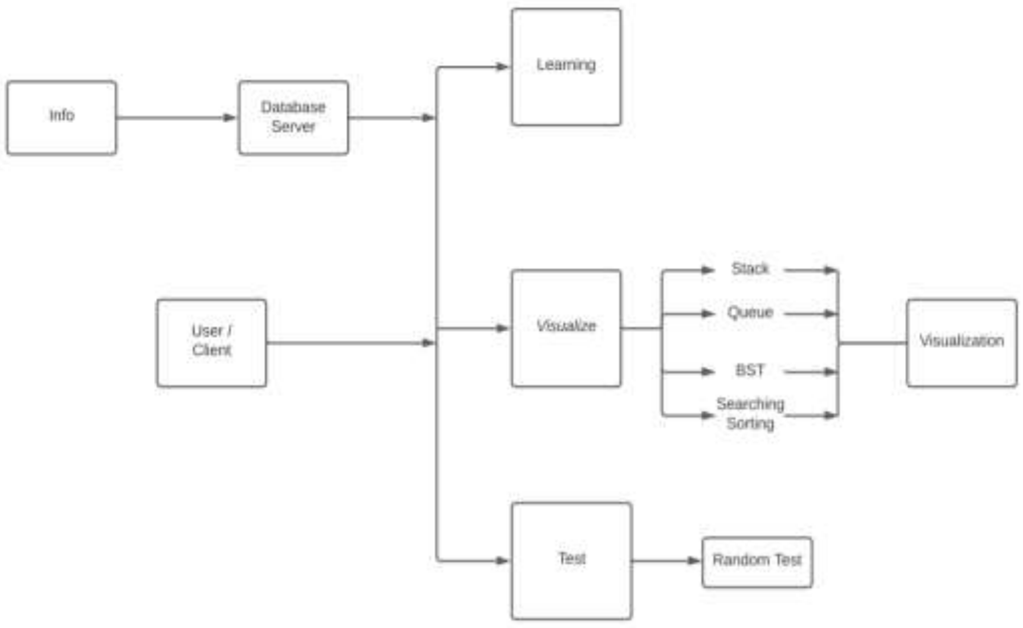


Fig. 4 Level 2 DFD

Technical Specifications

Operating Systems:
Windows, RedHat Linux

Software Requirements:
Visual Studio Code, MongoDB Compass

Languages used for Front-end:
HTML, CSS, JavaScript, p5.js library

Back-end:
Node JS, Express Framework, MongoDB Server, Kubernetes

IMPLEMENTATION

Implementation of AlgoViz:

Software:

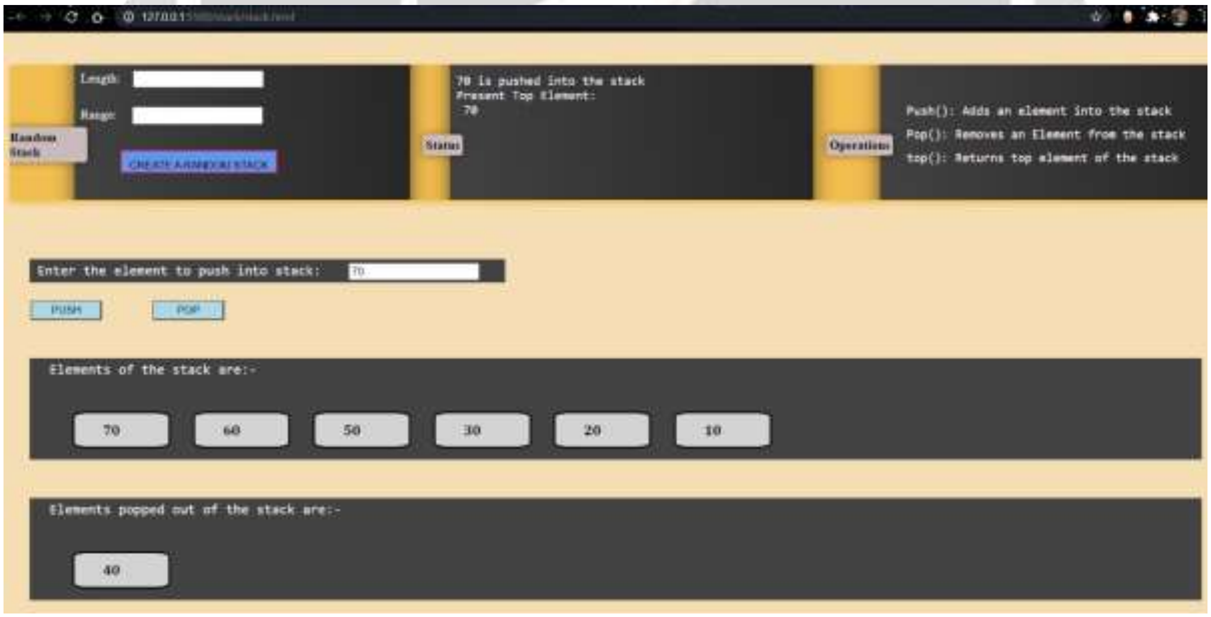
Here, we used Visual studio code software for writing programs/codes for webpages in HTML, CSS, and JavaScript for the frontend as well as for manipulations on our website.

Used p5.js to create a canvas on our website to visualize binary search trees. The whole application uses MongoDB server for storing user's session management. This application is deployed on the top of the Kubernetes which is an open-source management tool for management, scaling and load management, etc.

RESULT AND DISCUSSION

Results:

Stack Visualizer:



```
70 is pushed into the stack
Present Top Element:
 70
```

```
Push(): Adds an element into the stack
Pop(): Removes an Element from the stack
top(): Returns top element of the stack
```

Enter the element to push into stack: 70

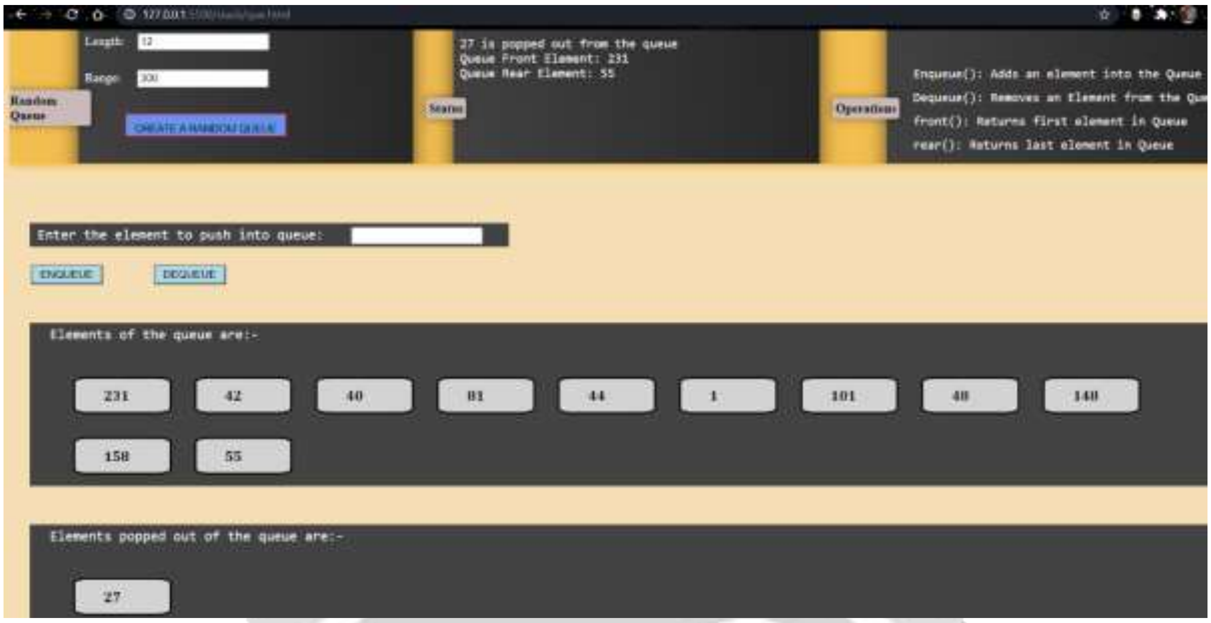
Elements of the stack are:-

70 60 50 30 20 10

Elements popped out of the stack are:-

40

Queue Visualizer:



```
27 is popped out from the queue
Queue Front Element: 231
Queue Rear Element: 55
```

```
Enqueue(): Adds an element into the Queue
Dequeue(): Removes an Element from the Qu
front(): Returns first element in Queue
rear(): Returns last element in Queue
```

Enter the element to push into queue:

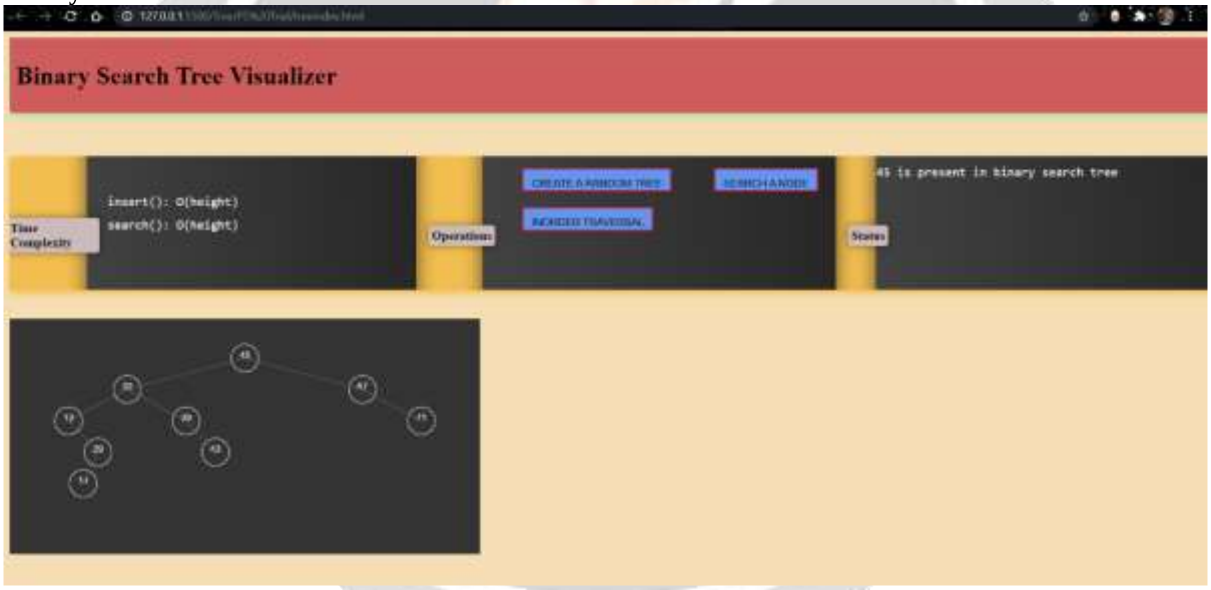
Elements of the queue are:-

231 42 40 81 44 1 101 48 140
158 55

Elements popped out of the queue are:-

27

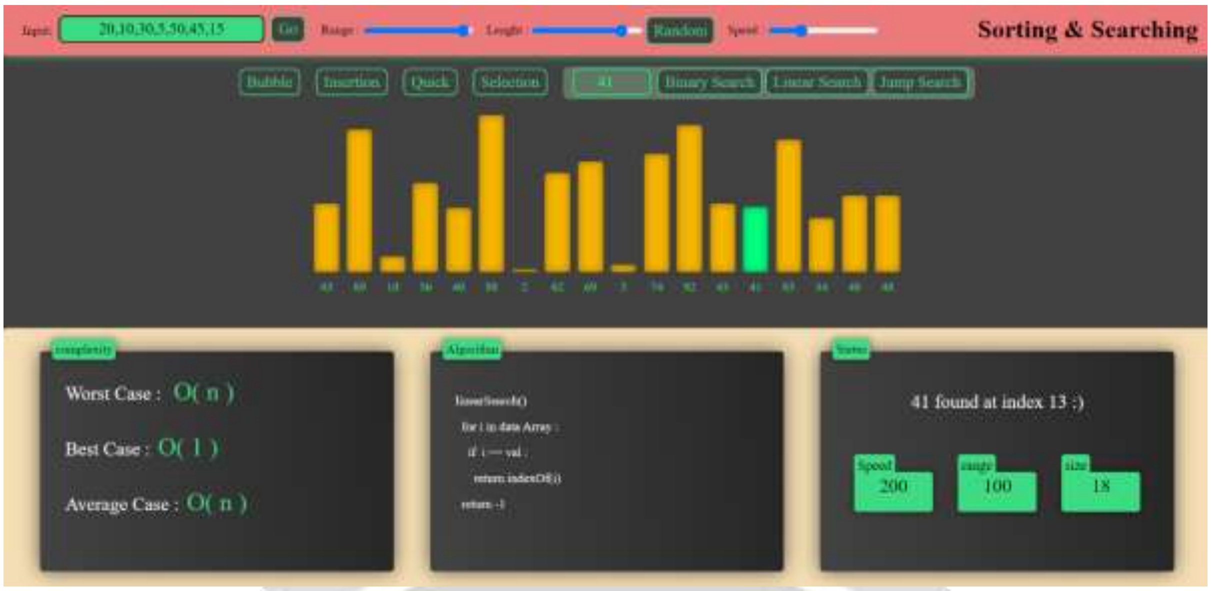
Binary Search Tree Visualizer:



```
insert(): O(height)
search(): O(height)
```

```
45 is present in binary search tree
```

Searching & Sorting Visualizer:



Worst Case : O(n)

Best Case : O(1)

Average Case : O(n)

```
linearSearch()
 for i in data Array :
  if i == val :
   return indexOf(i)
 return -1
```

41 found at index 13 :)

Speed 200 | range 100 | size 18

.

Discussion:
Data structures and algorithms operations will be demonstrated step by step on random data set generated by the system in which each step is narrated as a status in detail. Users can give their data as input and users can perform necessary operations to get the correct final output. They can test their knowledge by replicating the algorithm's steps. Users can give random tests provided on the platform to test their knowledge about data structures and algorithms. Including these three modules, the application is using a database server for storing user's session information.

CONCLUSIONAND FUTURE SCOPE

Conclusion

Algorithm visualization can be seen as a valuable supporting tool, used in addition to standard ways of education in the field of computer science. We believe that it helps to improve the quality of education in the field of computer science. Algorithm visualizations can help to understand the principles, but do not replace the need to implement algorithms by students in a chosen programming language.

This tool can be used as an effective supplement to the traditional classroom education and textbook for Data Structures and Algorithms course.

Future Scope

The rundown of presently available modules includes visualizations of searching algorithms like linear search, binary search, jump search, and some sorting algorithms like the bubble sort, insertion sort, selection sort, and Quicksort. Visualization for basic data structures like the stack, queue, and binary search tree is also available.

The extension can be made on different visual views on running algorithm or simultaneous comparison of different algorithm visualizations.

We can add some more complex data structures like heap, priority queue, segment tree, etc. The future implementation will include visualization of complex graph algorithms and greedy algorithms like job sequencing, etc.

REFERENCES

[1] Ville Karavirta, Clifford A. Shaffer (2016), *“Creating Engaging Online Learning Material with the JSAV JavaScript Algorithm Visualization Library”,* http://jsav.io/ , 2016 IEEE Transactions on Learning Technologies

[2] Ahmad Supli, Norshuhada Shiratuddin, Syamsul Bahrin Zaibon (2016), *“Critical Analysis on Algorithm Visualization Study”,* International Journal of Computer Applications

[3] E. Fouh, M. Akbar, and C. Shaffer, “*The role of visualization in computer science education*”, Comput. Schools, vol. 29, pp. 95–117, 2012.

[4] Tao Chen', Tarek Sobh' (2001), *"A Tool for Data Structure Visualization and User-Defined Algorithm Animation",* ASEE/IEEE Frontiers in Education Conference

[5] Jauhar Ali (2009). *"A Visualization Tool for Data Structures Course",* IEEE International Conference on Computer Science and Information Technology

[6] S. Hall, E. Fouh, D. Breakiron, M. Elshehaly, and C. Shaffer, "*Evaluating online tutorials for data structures and algorithms courses*", presented at the ASEE Annu. Conf., Atlanta, GA, USA, Jun. 2013, Paper 5951.