Data Structure and Algorithm Visualizer

R.Shrenik
Dept. of Computer Science
PES University
Bengaluru, India
shrenikr235@gmail.com

Niranjan Bhaskar K
Dept. of Computer Science
PES University
Bengaluru, India
niranjanbk532@gmail.com

Vidhisha Shankar
Dept. of Computer Science
PES University
Bengaluru, India
vidhisha20s@gmail.com

***Abstract*—Algorithms and data structures form the foundation of Computer Science. An algorithm is a way to implement a solution to a computational problem while a data structure is just a way to store data. Despite numerous advances in hardware and software technologies over the decades, it is imperative that the problem solver possess the tools and mental models necessary to implement an efficient solution (in terms of time and space) . The right algorithm and data structure combo can make the difference between a solution that runs in milliseconds versus one that takes hours / days to run. Working with large amounts of data to solve complex problems fast is the need of the hour. Gaining a good understanding of how they fundamentally work at a low level and using them to model computational problems has proven to be a challenging task for both students and tutors alike. Thus, the project aims to bridge this gap by creating a visually-pleasing, intuitive online learning experience to help problem-solvers understand how they work by allowing the users of the web application to input their data, create and see the data structures and algorithms in an animated action**

Keywords—Algorithm, Data Structure, Visualizer

I. Introduction

The project allows users to create and visualize data structures and algorithms online through intuitive web animations. This project will allow users to: Enter their own data to create, modify, visualize and therefore understand the core concepts of data structures, time complexity and space complexity in a step-by-step manner and visualize almost all pathfinding algorithms and sorting algorithms. Further, the user will have the ability to control the animation sequences of each of the visualizations concerned. General animation controls and transition speed controls have also been provided. The primary goal of our system is to simplify the teaching and learning process of complex topics through an intuitive, animated online learning platform used by anyone wishing to learn these topics.

II. Literature survey

Tao Chen and T.Sobh [1] describe a visualization tool designed to aid beginner CS students grasp concepts of basic data structures has been implemented as a desktop application.The tool allows students to write algorithms in Java and observe their execution, although in a limited respect to the range of data structures available to visualize. A special parser and grammar is defined to parse user-defined algorithms and visualize them on a canvas. This extends the scope of the project in the future.

Ryan Baker, Boilen and Michael T. Goodrich's paper [2] provides a unique implementation of visualizers and testers as a desktop application. The application uses API-specific methods to visualize data structures on a canvas. It keeps track of the history of methods invoked and provides real-time configuration of the structure.

The tester as a grading tool had been introduced to allow evaluators to test a student's code and structure. It compares string outputs and delivers the appropriate result report to the evaluator. Their future work entails creating generic button implementations for methods (create(), insert(), delete()) across the application with a better UI. Since the course CS2 was tested for quality of the implementation as well, it would serve the evaluators well to include a time and space complexity calculator as well

Baranik, Robert and Steingartner described [3] how thinking recursively might be confusing in most cases although it may have fewer lines of code, having an in-depth understanding about the working of recursive algorithms through visualizations makes it all way easier and faster for both lecturers and students.

Osman, Waleed and Elmusharaf [4] explain how their implemented system combines both Algorithm Animation and Program Animation. The visualization engine is embedded with the program script.The visualization and use of the system doesn't involve rich interfaces, like a window-based modal form, and using the mouse, so it is clearly simple in order to avoid student distractions but this can also be viewed as a downside as far as the features of the system are concerned.

III. Product perspective

The product is intended to provide the following features to its users:

A. Animation Controls

Understanding the visualization of a particular data structure or an algorithm is a step-by-step process. Therefore, it is imperative that the user have control over how the structure behaves on their screen. Media controls like play, pause, speed up, speed down, previous and next are required for a better user experience,

B. Operation Visualizations

Data structures should ideally support creation i.e insertion of data into the structure and enable the ability to provide user-defined input that the algorithm can work on. The user should also ideally have the ability to delete or remove nodes during visualization..

C. Sorting algorithms

Sorting is an important problem in computer science and is one of the most fundamental concepts when trying to arrange data for efficient access and utilization. The project must ideally include implementations of both comparison-based sorting algorithms as well as other important ones.

Comparison-based sorting algorithms and other implementations included are:

- Bubble sort
- Insertion sort
- Selection sort
- Merge sort
- Quick sort
- Random Quick sort
- Counting sort
- Radix sort

D. *Linear and non-linear data structures*

In a linear data structure, the elements are arranged in a sequential manner. Based upon the implementation by the programming language, these elements can either be objects of the same type or of different types. The project includes implementations of:

- Linked lists (Singly)
- Stacks (With predefined size)
- Queues (Single-ended)

With nonlinear data structures, the elements are not constrained under sequential order and can be connected via pointers or references. The project includes implementations of:

- Binary Search Trees
- Heap Trees
- AVL Trees
- Conversion of Tree to Binary Tree and vice versa
- Graph algorithms (DFS, BFS, Prim, Kruskal, Dijkstra)

Thus, the project covers the most fundamental building blocks of organizing data to efficiently manipulate it. This is especially useful when designing large-scale systems where efficient organization of data can make the difference between an algorithm that runs in 1 second vs an algorithm that takes minutes or even hours to run.

E. *Search and simple pattern matching*

The project also includes simple but useful algorithms like binary search, linear search and string matching algorithms like the simple string search and the KMP algorithm.

IV. System design

Keeping long term development in mind, the application uses an MVC (Model View Controller) structure. The system consists of 3 parts including the visualization library, visualization widgets and state objects. These 3 modules interact by passing objects between one another. The system makes use of the jquery library and uses plain HTML, CSS and JavaScript to create and render the web-based UI functional. The project uses a visualization library, widgets and state objects.

A. Design considerations

- The proposed model aims to create an interactive, user-friendly modular online platform with easy extensibility, high-performing animations with preferably reusable components for an easier and faster development/maintenance experience. The UI would involve providing the user with controllable animation controls while creating and manipulating data structures, algorithms and their associated operations.
- Existing systems are not extensible i.e addition of new components to the website would require manually editing HTML files, writing specific DS/A specific functions that interact with the UI without an abstraction layer and therefore the proposed model makes use of widgets that abstract away the UI rendering details from the logic.
- Speed: The animations are smoothly rendered on the UI with no lag. A good internet connection is required.
- Security: The website doesn't inherently deal with any sensitive content and there is no data collection. The hosted website will take care of vulnerabilities like directory listing. The project code is written keeping security in mind.
- Availability: Subject to the hosting plan we end up choosing.
- Privacy: No risks. No data collection taking place.

B. Architecture choices

- The visualization library consists of existing JS files (some existing, some added, some edited) that define basic drawing elements used for complex animations involving different kinds of structures (nodes and edges). For each DS/A we make a widget relating to an HTML file. This controls the animation of each of the operations as well as the displaying of the code by manipulating the DOM.
- Visualization Widgets: These widgets contain the logic of the data structures and algorithms visualized. It passes the objects to the graph library which will be reflected on the UI. We make use of the Canvas API and the library to do so.
- State Object: It is a JS object containing information on what to draw on the UI. This object is passed into the library to be drawn on the UI. The state object will contain coordinates of vertices and edges, their CSS attributes etc.
- Benefits include a reusable visualization library. Here, the library can function independently from the visualization widgets. As long as an array of state objects is supplied, it can display the DS/A on the UI => Other future tools can use this library to achieve the same look and feel.
- Modularization: The library only contains the rendering information, separating the logic and the UI. Also, the library we modify will directly manipulate the DOM objects hiding the rendering details from the viz. logic => easier to manage code.
- The drawbacks include development time and the fact that direct DOM manipulation can slow down rendering of components on the page

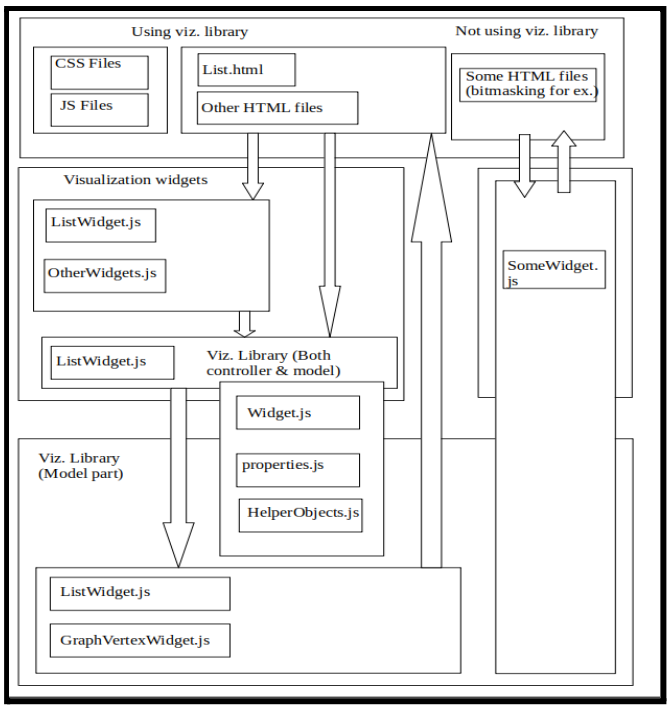

Fig: Overall architecture diagram denoting the visualization library and the widgets

C. *Constraints, assumptions and dependencies*

- Extensibility and effort: Functions (start,end,algo) do the rendering, not the widgets. This means that new tools can't use existing code and we'd have to reimplement the UI every time we want to add something. Abstraction using widgets solves this problem.
- Animation efficiency: In our initial implementation, when a user decided to return to the previous frame, the library replayed the entire animation from the first frame to the requested frame => Noticeably slow. Now, the visualizations can jump b/w frames in O(1) time since all that needs to be done is to load the state object for that frame.
- Availability and security are dependent on the hosting platform and the payment-tier we choose.
- No mobile access: Screens with a minimum width of 768px is required for smooth animations and interactivity on the platform.

V. Class and Interface Diagrams

The UML Class diagram is a graphical notation used to construct and visualize object oriented systems. A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods) and the relationships among objects.

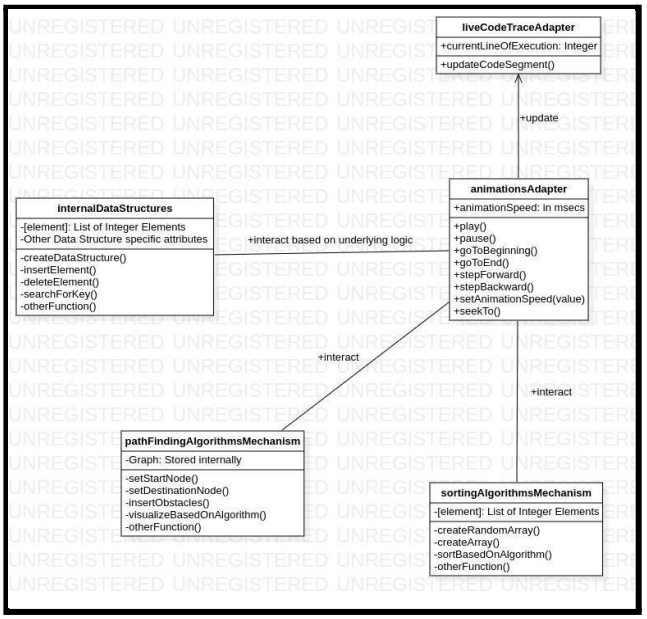

Fig: Master class diagram

The following images describe the interface used to interact with the visualization system:

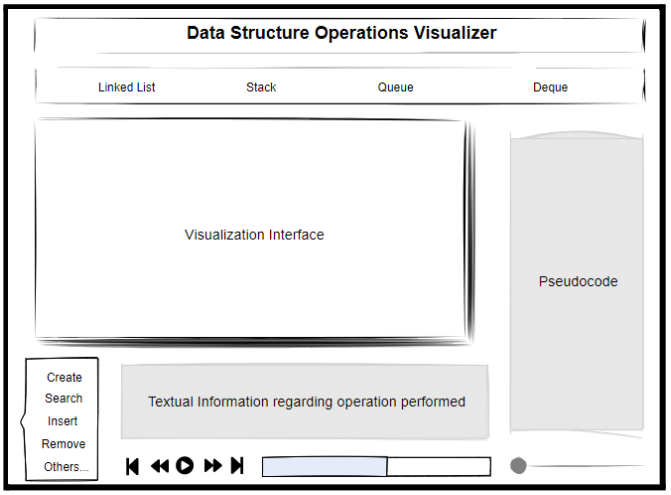

Fig: The linear data structures landing page

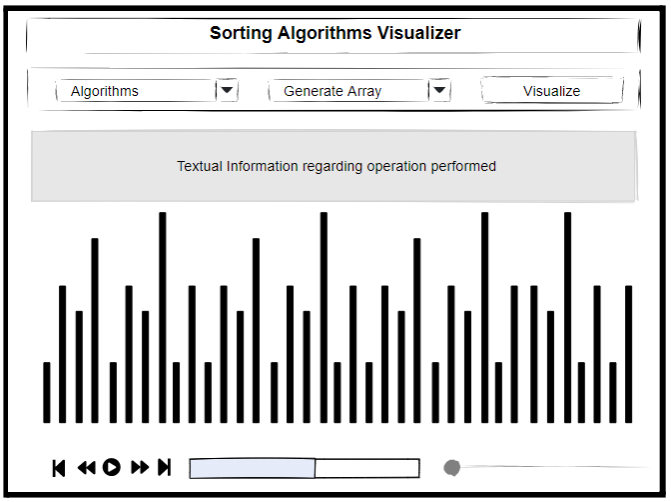

Fig: The sorting visualizer landing page

VI. Conclusion

The project implements linear and nonlinear data structures, graph algorithms, sorting algorithms and search algorithms in an easy-to-use, performant, bug-free web interface. The idea of a platform to visualize data structures and algorithms, see them in action and the ability for the user to perform operations on them is vital to learning how they work. It is important to note however that the platform only acts as a supplement to traditional methods of teaching and not as a replacement. Offline doubt-clarification and additional problem solving are important factors when conveying highly technical, detail-oriented concepts like algorithms. Working on building well-tested, error-free implementations is more important than covering a large set of data structures since a buggy implementation will confuse users and drive them away. This project involved a lot of reusable code for the application and has followed a modular approach. The platform is fast and visually pleasing and has low downtime to appeal to teachers and students. As an essential feature of the visualization system, the several animation control features benefit the user groups largely allowing them to seek through any portion of the animation sequence. Moreover, such an application that helps its users to visualize the core concepts in Data Structures and Algorithms, be it basic data structure operations or sorting algorithms or even the diverse pathfinding algorithms will largely help the user group concerned whether they are teachers, or students.

References

[1] Tao Chen and T. Sobh, "A tool for data structure visualization and user-defined algorithm animation," 31st Annual Frontiers in Education Conference. Impact on Engineering and Science Education. Conference Proceedings (Cat. No.01CH37193), Reno, NV, USA, 2001, pp. TID-2, doi: 10.1109/FIE.2001.963845

[2] Osman, Waleed & Elmusharaf, Mudawi. (2014). Effectiveness of Combining Algorithm and Program Animation: A Case Study with Data Structure Course. Issues in Informing Science and Information Technology. 11. 155-168. 10.28945/1986.

[3] Teaching Support for the Visualization of Selected Recursive Algorithms Baraník, Róbert and Steingartner, William. ipsitransactions.org/journals/papers/tar/2021jan/p3.pdf

[4] Ryan S. Baker, Michael Boilen, Michael T. Goodrich, Roberto Tamassia, and B. Aaron Stibel. 1999. Testers and visualizers for teaching data structures. SIGCSE Bull. 31, 1 (March 1999), 261–265. DOI:https://doi.org/10.1145/384266.299779

[5] Cyber Security Risks For Modern Web Applications: Case Study Paper For Developers And Security Testers by Devanshu Bhatt.

[6] https://google.github.io/styleguide/jsguide.html

[7] https://material.io/design/guidelines-overview