VISUAL ALGO

A visualization of Data Structure Algorithms

A Project Report

By

Pooja Harshad Parmar

Under the Guidance of

Prof. Zerksis D Umrigar

State University of New York at Binghamton

BINGHAMTON UNIVERSITY | THE GRADUATE SCHOOL

Introduction

‘Visual Algo’ is a web application that provides visualization of Computer Science algorithms like Selection sort and Binary Search Tree. It can be used by any Computer Science student trying to understand the concepts of various Data structure algorithms visually. The application provides a user-friendly interface that allows the user to navigate through different parts of the application easily. It uses Bootstrap 4 as its front-end which is an open source CSS framework for responsive, front-end web development.

Each algorithm provides a simple pseudo code that helps understand the working of the algorithm and manipulate the data items by providing their own custom inputs. Custom input helps students to understand the Best case, Average case and the Worst case of the algorithms. It also provides features like Play/Pause and Reset to help understand the process of the algorithms.

JavaScript

JavaScript is the most commonly used programming language used by Web browsers to create a dynamic and interactive experience for the user. Hence, I used JavaScript and its various libraries to visualize various Computer Science Algorithms.

I started out by using only JavaScript for different data manipulation, however, I explored a lot of JavaScript libraries that provides speed and accuracy and saves time, so 'why reinvent the wheel?', when the code is already written?

This web application taught me various features of JavaScript like asynchronous programming. I majorly faced a problem while writing an iterative approach of the Selection sort algorithm. I used a 'for-loop' for comparing the input array to sort and then swap the current element with the minimum-found element. While using this approach, I wanted to implement the Play/Pause feature on the application, that allows the user to Pause the algorithm and start from where it had stopped. To implement this, I used setTimeout() and clearTimeout() which executes a given function or a piece of code once the timer expires. So, when the user clicks on the pause button, I used clearTimeout() to stop the implementation of the function. However, I noticed that, since I had used a for-loop it will always run-to-completion even if there is a callback that is initiated in the middle like setTimeout.

https://www.pineboat.in/post/javascript-run-to-completion-event-loop-asynchronous-foundations/

CanvasJS

Before exploring CanvasJS, I started out by simply using SVG and CSS to create rectangular bars to represent the elements in the array. It was working fine, until I reached a point trying to swap bars (swap elements) for sorting and it got complicated.

I started exploring various JavaScript libraries used to represent charts such as Google charts, ChartJS, D3.js but it didn't work well with my purpose. Then I came across Canvas.js which is a responsive HTML5 charting library with a very simple API. I used JSON data API to render Vertical Bar charts accordingly and to separate the UI from the data. CanvasJS allows you to easily convert the JSON data in various chart types by just changing the 'type' property to Bar Chart, Doughnut Chart, Pie Chart, Area Chart, etc and CanvasJS works its magic by converting the data to the respective chart. It also provides various themes, and add chart title, axis labels, grid lines, etc.

The application allowed users to create their own inputs for Selection Sort, Play/Pause the algorithm, Reset whenever necessary along with a Pseudo Code that explains the sorting steps. These features provided a great way to understand the algorithm deeply.

However, I came across various limitations of CanvasJS at a significant point and wanted to explore all the limitations and research a way to work with it.

It does not allow you to swap the bars visually by just changing the x-co-ordinate, so every time you want to swap elements, the chart had to be re-rendered.

D3.js

Before starting out with D3.js, I explored a lot of JavaScript libraries that allows you to create a detailed tree diagram like Treant.js and Vis.js. Both Treant.js and Vis.js required a lot of configuration was pretty hectic for me. However, I also came across D3.js exploring on Quora and read good reviews by the users.

D3.js (Data Driven Documents) is a JavaScript library for manipulating documents based on data by using HTML, CSS and SVG. It is extremely fast and supports large datasets and provides dynamic behavior for interaction and animation. It helps you to attach your data to DOM and then use CSS3, HTML and SVG to showcase this data.

I used JSON to create parent and children nodes for Binary Search Tree. D3.js allowed me to convert this JSON data to create nodes by appending a circle with each node with its x and y co-ordinate on the DOM. Then, using tree.links(), linked each parent node with its child node in the form of an array of 'source' (parent node) and 'target' (child node). Also, using the D3.js feature of creating a line (path) from the source to the target using the x and y co-ordinates and appended the entire tree to the DOM.

The application allowed a user to Search an element and to Insert a new element in the tree. GSAP, an animation library was used to visualize the path from the parent node to the target node by using CSS selectors and manipulating the CSS properties to provide an animation. It also provided information for each step in the process of searching for an element or inserting a new element for the user to understand how the animation works.

Future Scope

1. Selection Sort using CanvasJS had a lot of limitations and was not very scalable. Instead, D3.js could have been a significantly better option.
2. Providing additional features like Rewind, control the speed of the animation.
3. Implementing other Data Structure Algorithms

References

1. MDN JavaScript
2. https://www.pineboat.in/post/javascript-run-to-completion-event-loop-asynchronous-foundations/
3. https://canvasjs.com/
4. https://www.dashingd3js.com/table-of-contents
5. https://d3js.org/
6. https://greensock.com/