International Journal of Aquatic Science
ISSN: 2008-8019
Vol 13, Issue 01, 2022

Sorting Visualization of Algorithm Using Different Types of Sorts

Rushikesh Tangade[1], Bhagyashri Dadhe[2], Gitanjali Chavhan[3], Tejal Chaudhari[4], Yash Tarachandi[5], Ritesh Deshmukh[6]

[1,2,3,4,5]B.E Student of Computer Engineering, Jagadambha College of Engineering and Technology, Yavatmal, Maharashtra, India
[6]Assistant Professor at Jagadambha College of Engineering and Technology Yavatmal, Maharashtra, India
Sant Gadge Baba Amravati University, Amravati, Maharashtra, India

Abstract: This paper outlines a study that dependable the benefits of animated sorting algorithms for teaching. To visualize four sorting algorithms, a web-based animation application was established. Visualization of data is implemented as a bar graph, after which a data sorting and algorithm may be adapted. The resulting animation is then performed either automatically or by the user, who then sets their own pace. This is research on the computer science subject's approach to learning algorithms. The experiment featured a presentation and a survey, both of which asked students questions that may illustrate improvements in algorithm conception. These findings and reactions are cataloged in this document and compared to earlier investigations Selection Sort, Merge Sort, Bubble Sort, Insertion Sort, Heap Sort

Keywords: Sorting, visualization, algorithm, Implementation

1. INTRODUCTION

How do you get something done? You don't have to be highly complex in solving the problem, for example, if your car's headlight is broken (although nowadays, manufacturers are trying the patience of the community with their increasingly abstract, space-age designs). The main argument is figuring out the best way to set about it. To locate step-by-step directions in your car's handbook, do you conduct research, or do you use instinct to find someone who knows how to do it? In short, my instinct tells me that I am a visual learner and hence more suited to acquire topics by watching them than by reading about them. In this case, I found that seeing the data move to its rightful spot as the result of an algorithm is MUCH easier to follow than looking at the source code and trying to figure out where the data was supposed to drive. My project was born out of my curiosity about sorting algorithms, which inspired the idea for this paper, which details an online tool I built that explains how sorting algorithms transform and organize sets of data. It is possible to organize a list of people, for example, by their age in ascending order using different methods. To aid my visualization, I created a histogram of numerical data to represent four well-known examples. Each number is depicted as a bar, and the height of each bar represents the value of that number. It is being shifted by the algorithm from its original, unordered location to its final ordered place, making it distinct from the rest of the data. Selection Sort, Bubble Sort, Insertion Sort, and Merge Sort is the four sorting algorithms. 2 Let's imagine that you have printed each person's age on a separate index card. carry the youngest card to the anterior and then sort the cards by age. To discover the next

smallest item, identify the age that has already been ordered and position it behind the already ordered age. Index cards full of ages will be at the boundary of the hill. Selection Sort works in the same way as this. In this case, to sort a set of data, you select the smallest first, and then the next smallest, and so on until you've sorted all of the data. This technique is quite simple to explain to someone in conversation, but more advanced sorting algorithms, such as Quick Sort, which requires the data to be moved around a pivot point, are not easy to grasp using text alone. I wanted the animation to appeal to a wide spectrum of individuals utilizing various technology media, so I had it made in a web-based format. Instead of requiring the user to install extra software or attempt to organize setups to use the tool, this helps to remove this source of anxiety. It uses HTML5 (Hypertext Markup Text Language) JavaScript and CSS for the website's layout.

AIM

The main aim of this Project is nowadays sorting algorithms are widely used in computer software. For example, if you open file explorer on your PC, you may see files sorted in different ways. Searching in sorted data is more capable than in not sorted ones. Students of computer science start learning different algorithms in the first year of studies and sorting algorithms are among them. Since We faced the problems of sorting during algorithm design in the first year of my studies, there is an understanding that visual representation is a vital part of the studying process. During working on the thesis, it was very exciting to learn different techniques of sorting algorithms in the depth. The main goal of the thesis was to create a program that would serve as a tool for understanding how most known sorting algorithms work. There was an attempt to make the best possible user experience. The demonstration software is made in a user-friendly and easy-to-use style. To gain maximal benefit from learning you can try each sorting algorithm on your data. The text of the thesis describes the principles of the most known sorting algorithms which are demonstrated in the computer program. It might be used as a source for learning algorithms by students. Also, the program might be easily used as a demonstration by lecturers and tutors during classes. Besides, there is programmer documentation and a user guide to the provided software. Readers of this text are expected to have some programming experience to know basic data structures such as arrays, lists, and trees and understand recursive procedures Also, knowledge of some simple algorithms and their implementations could be helpful. To understand the topic better, knowledge of linear algebra and calculus is involved.

DESIGN

a) The User Interface

The design and structure of the user interface components have remained unchanged even if the underlying back-end code was refactored midway through the construction. Each component has its feature: The canvas has twelve features; 10 control buttons, and a volume toggle button. The canvas area is where the four sorting algorithms are visualized, and that area will be the position where the sorting algorithms' output is adapted in. The first row of four blue-bordered buttons at the bottom of the canvas is the selectable algorithms: Selection Sort, Bubble Sort, Insertion Sort, and Merge/Insertion Sort. This type of visualization is offered to users to select an algorithm of their choice and to observe how that algorithm functions. Before launching the animation, the user will need to select an algorithm. The sorting algorithm must be selected before the input data type is specified. To choose between sorting input data that is already in order (as shown in Figure 1 on the previous page) or reversing and randomizing the order, the three gray-bordered buttons on the left of the bottom row are available (shown in

Figures 2 and 3 in the following paragraphs). Sorted order is the default. The sorting algorithm is called once the input and sorting methods have been chosen. Following, the "Start" button in the next row of buttons is clicked to perform the sort from the beginning to the end. The user can click the yellow-orange-bordered "Step" button to watch the algorithm step-by-step. Once the process is already underway, you can simply stop it by pressing the "Stop" button. To reduce misunderstanding in user-friendliness, I've tried to make the interface basic and placed buttons related to one another for easy access. The algorithm buttons are on abstracted tiers and have a blue coloration approximate to them. Additional buttons are arranged in a grouped fashion, with a gray background. The "oddballs" are the animation controls. Despite being grouped, each controller has a distinct color to denote the kind of animation it does. The colors are modeled on a traffic light with green being the go signal, red being the stop signal, and yellow being the signal to slow down (or in this case, yellow-orange means to pace yourself). Additionally, each button also adds visual feedback to the user by altering the color as the cursor hovers over it. The volume toggle button is the final function that is available on the Web page. The button appears to the left of a speaker picture on the lower left-hand side of the web page. While conducting some study on sound effects for animations, I realized the tool could be more participatory by both hearings and seeing the animation, rather than merely observing. To determine the sound each bar makes, use the following rule: each bar has a sound allocated to it based on its height. To hear four octaves on a piano, from small to big, the bars need to be in order. With the sound approved, each bar in the sound animation plays a divergent tone from left to right when it appears. You'll only be able to hear the complete four octaves in order if the bars are out of order. The bar's presently being played color will change to green, and then, when the sound animation is finished, it will change back to blue. However, because the animation is hungry for memory, the animation may stop momentarily and then resume. This means that the sound animation is turned off by default, but the user has the option to toggle it. You'll get the best results if you use the sound animation with Selection Sort. Finally, there is one feature that you will discover only after making a selection and attempting to sort: you may modify the sorting algorithm on the fly. In other words, you can pause an animation while using a sorting algorithm like Selection Sort, then choose another algorithm, like the Bubble sort, while the animation is paused. continue sorting the bars, and start the animation by clicking "Start" or "Stop." How the movement of data changes based on what algorithms are already affecting it is remarkable to behold. Also, sorting semi-sorted data using the provided options provides a unique view of how the algorithms function.

The main window of the Application

b) System Architecture:
HTML5, CSS, and JavaScript make up the backend code. There are three varieties of code in one .html file and all three can be executed from this file alone. Including different types of web languages on a single page is one of the shortcomings of HTML 5. Since, therefore, there were three different types, each had been segregated, producing three different files (plus the miscellaneous sound and image files). Readability and keeping relevant code together are benefits of excellent programming practices. However, in the end, I opted not to break the code into two separate sections because of these two reasons. By just having to worry about one project file instead of three, the project may be more easily transported and sent. And because the changes to the coding languages are identified unambiguously in the project file, they do not reduce readability. An RIA can have more than one programming language in a single file (Rich Internet Application). As you can see, the three coding languages are the only extensive components. However, since JavaScript runs immediately in the browser, it is unnecessary to employ a server on the back end (like PHP). HTML5 and CSS are employed in web development. As illustrated with a single, bidirectional arrow, the HTML5 and JavaScript communicate to run the relevant algorithms and update the interface. The code for HTML5 and CSS did not change significantly throughout the project. The parts of HTML5 that were updated were the function calls for each button, since they were altered from a functional programming mindset to an object-oriented one. We've abstracted away all of the back-end code behind all of the different algorithms and animation selectors.

IMPLEMENTATION
The use of HTML5 (Hypertext Markup Language 5), JavaScript, and CSS combine to form this project's implementation (Cascading Style Sheets). There is only one project file which is an HTML file that contains the code. The only additional piece of code added to the main HTML file is the .m4a sound files to support the sound animation functionality (which are saved as .m4a files). As of now, I only did extensive testing using the Mozilla Firefox browser, and it's the browser of choice in this context. However, tests done quickly revealed the possibility of Google Chrome and Safari integration. This software operates both object-oriented and functional programming paradigms in how it constructs the code. Before the final phase of development, the design was almost completely functional, where only three objects were used: one to control the canvas that displayed the animation, another to represent a piece of data, or a "bar" object (blue rectangle with dynamically changing height and position), and a final one to represent the positions that each bar moved to, or "pos" objects. Although this incorporated several functions calls, some instance variables and Boolean values were utilized to keep track of the algorithm picked and when to animate, but this led to a greatly interspersed mass of code that was crucial to maintain. Several big refactoring later, the code has now taken on the form of a Model-View-Controller Architecture. Although, because of its functional character, it possesses a multitude of individualized functions that alter the instance variables and Boolean values, which means it has a multitude of functions that directly alter the View and Controller. The major module in the HTML code between the tags is known as the global scope. Everything within the framework can access the aforementioned variables and methods.

The View:
there are three items on the view: the sort area, the bar, and the position. These objects operate within the container defined by the tags in the Html file. This function's space is sometimes referred to as the "main" function, the first function invoked in a program It is the sort area that keeps the bars up to date working a timer, while at equal time generating the bar graph. As a

result, whenever "Step" is invoked, the bar values are updated depending on the steps array (discussed later). In the sort area, after every second iteration of the timer, the rectangles will be redrawn with varied heights that represent the new values. The bars change sixty times per second, so when the “Step” button is selected, the change is instantaneous. In the sort area, the bar object represents each piece of data. The statement encompasses all of the aspects of color, value, location, height, and sound. While having a defined array named bars for the current bars in the bar graph helps conserve attributes such as the total number of bars (total value) absolute of other characteristics, it is simple to update any or all of the attributes by iterating over the array as necessary. To update the bars to a natural data configuration, the In Order, Reverse, and Random buttons iterate fast beyond the array. The bar object is related to the post object (which is short for position). The region on the canvas that is updated when the Sort Area event fires is an XY-pixel coordinate grid. This item was invented to make arranging the bars a little bit simpler (1- 32). Thus, if I wanted to move a bar, I would supply merely the number of the bar's location. For the bar to move, it will first determine the exact coordinates and then go to that location. Another way to say this is to say that, position one is defined by the two-dimensional coordinate pair (9, 135), which is the bottom left-hand corner of the bar. As long as each bar has a rectangular object that is associated with it, as well as a top left-hand corner point that defines the rectangle's height, the bar must be relocated to its right location.

Unsorted array

The Model:

The model is composed of one item, known as the sorter. This object houses the algorithm's code divided into methods. The start method centralizes on an integer constant and uses it to order the algorithm's possible algorithms. This object is directly composed of the four sorting algorithms shown on the user interface as "Selection Sort, Bubble Sort, Insertion Sort, and Merge/Insertion Sort." The sort algorithm mechanism is invoked as the user selects a sorting technique and clicks on one of the sort algorithms buttons. Then, when the algorithm sorts the data, a trace is created. The steps array, which contains all the movements in the animation, is a two-dimensional integer array that is available to methods on the web user interface. A typical back-end code interface is implemented using the steps array. The "Start," "Stop," and "Step" buttons function as controllers for which sub-array will be displayed on the canvas after the computational back-end has completed the tracing. If this loads before the user have selected "Start" or "Step", then this represents a user action. When you click the "Step" button, the next step on the canvas loads and animates. An animated sequence of a single bar moving across

the others is produced because the timer continually redraws the bars. There's no "Start" button, only a "Step" button that is set to go off on a timer. Using a two-dimensional array gives you the option to view the sorting algorithm's stages within the View. The process of adding an algorithm is similar to writing down the trace of the new algorithm, which is then saved in the same location. To complete the algorithm's walkthrough, the View will cycle through the data and update the bars in the bar graph to show how the algorithm calculated the steps it took. It's important to note that if the algorithm generated a change in the position of a piece of data, the steps are merely recorded.

Let's give an example: When sorting the pieces of data using Selection Sort, each piece of data is moved to its final and accurate location after one step, whilst the others require numerous steps to get to their final positions. While this sorting method appears to do the most effort compared to other sorting methods, it finishes sorting the most slowly. As a result, the visualization doesn't provide the correct visual impression of the data comparisons, which is one of the most important aspects of sorting algorithms. Two-dimensional arrays do demand more memory than a one-dimensional array. The size of the array is based on the number of steps that are required to sort the data. We may assess the algorithm's space needs by examining how long it takes. In Computer Science, using Big-Oh analysis is the standard way of determining how long something will take. The notation consists of a capital letter O, which represents the worst-case performance of the algorithm in question, followed by a constraint in parentheses that describes the worst-case performance of the algorithm.

View of insertion

The Controller:

all web page on the web has at most one Controller. The buttons have been programmed in HTML and lead the browser to execute a JavaScript procedure upon clicking. CSS is used for making the buttons look attractive, arranging them, choosing their colors, and applying visual effects. The methods alter the Model's state, prepare it for the update, and then apply the modifications to the View. Four groups of buttons are distinguished. The blue-bordered buttons at the top of the interface are the sorting algorithm buttons, and they interact directly with the sorter object. The Model calls functions as needed to construct an array of steps for animation. In this animation, the bottom-right buttons allow you to activate the animation, pause it, and advance a single frame. This pulls crisp values from the step array to keep the bar graph crisp. These bar changes are seen in the View, which causes the bars to shift position dependent on the methodology used. The lone button in this category is the Step button. When using the Start

and Stop buttons, you may either set a timer to call the Step button every quarter of a second, or you may halt the timer. To the left of the gray-bordered buttons, the bars are controlled directly by the bottom left buttons. To arrange the bars in the correct sequence, in reverse, or random order, they move the beginning location of the bars. The bars do not physically move. Each button alternative loads an array of data that represents the sequence the user w. This alteration appears on the canvas instantly as soon as it is made. This last group is a single button, and it's an image. A speaker icon adjuster between "mute" and "audio" in the bottom-left corner of
sound action just selects the "Sound Animations On/Off" option. When the web page loads, the sound will be muted by default (for reasons of memory and performance issues).

UNIT TESTING:
To experiment to evaluate my animation tool, I recruited coworkers to act as test subjects and survey respondents. To that end, the plan was to expose her Computer Science Data Structures class to the tool and get them to take part in a survey comprising questions that gave the students the ability to write down what they observed and learned, even if it was little. The questions contained in Appendix C are the questions that were utilized for the survey. Despite the class size being twenty-one students, only thirteen were present; fortunately, all of the students who had completed the survey were there. The setup here was ideal since it addressed the audience for whom I was designing this course: a group of college students who were taking their first court in computer science and who needed assistance with algorithms.

2. CONCLUSION AND FUTURE WORK

This web-based animation tool for viewing the following sorting algorithms functions in great part because of all the time and effort that I invested into it. Despite its memory overhead, the feedback given to it was mostly good from the students that worked with it. This is consistent with my prior research, which revealed that there was no substantial difference in learning the content. What I do agree with totally is the attitude that holds there is a great need to investigate and produce animated presentations to enhance education in the classroom. Overall, I am not concerned that a large rework to a different language will be required soon because JavaScript is still one of the most popular web languages. We all know about my laundry list of upcoming projects, but there is one elephant in the room that still has to be addressed: resolving memory difficulties. Following this, we would implement Merge/Insertion Sort, which takes into account the Merge Sort. Then, I would start up Quick Sort to finish the job because the code is ready to be integrated. Finally, I would make the online tool available to the public, with the feature I want most, which is to make it available to the public. This might be tough as well. The application that created the animation tool knows that it's available locally, but because of concurrency, it can serve numerous requests to the website by separate users. As I try to figure out how to make the code as efficient as possible, I'd need to spend some time thinking about how to make it work with numerous people using it. This would be excellent, as it would enable a form of a comparison study.

3. REFERENCES

[1] CORMEN, T. H.; LEISERSON, C. E.; RIVEST, D. L.; STEIN, C. Introduction to algorithms. Second Edition. 2001. ISBN 0-262- 03293-7.
[2] KNUTH, D. The Art of Computer Programming: Fundamental Algorithms. Third Edition. 2004. ISBN 0-201-89683-4.
[3] sipser, M. Introduction to the Theory of Computation. Boston, MA: PWS Publishing Company, 1997. ISBN 0-534-94728-X.
[4] BELOHLÁVEK, R. Algoritmická matematika 1: ˇ část 2. Available also from: hhttp://belohlavek.inf.upol.cz/vyuka/algoritmickamatematika-1-2.pdfi.
[5] KNUTH, D. The Art of Computer Programming: Sorting and Searching. Second Edition. 2004. ISBN 0- 201-89685-0.
[6] SEDGWICK, R. Algorithms in C: Fundamentals, data structures, sorting, searching. Third Edition. 2007. ISBN 0-201-31452-5.
[7] Geeks for Geeks. Available from: https://www.geeksforgeeks.org/i.
[8] BELOHLÁVEK, R. Algoritmická matematika 1: ˇ část 1. Available also from: ˇ hhttp://belohlavek.inf.upol.cz/vyuka/algoritmickamatematika-1-1.pdfi.
[9] Stackoverflow. Available from: hhttps://stackoverflow.com/i. [10] Java documentation. Available from: hhttps://docs.oracle.com/javase/8/i.