**Algorithms and Data Structure Narrative**

Karina Sanchez

Southern New Hampshire University

CS – 499 Computer Science Capstone

June 9th, 2021

Algorithms and Data Structure Narrative

During my computer science journey, I have had the opportunity to work on several different projects that have enhanced my technological skills. A particular project that I wanted to better enhance and further perfect is ‘My Visualizer Application’. This artifact, in which I created a year ago, was one of the first tasks given to me during my internship with HCL Academy. This application visually displays the difference between a merging algorithm and a quicksort algorithm, which is meant to enhance my skills and knowledge with algorithms and data structures.

The HTML file displays the title, footer, three buttons, two slide bars, and a bar graph. The CSS file organizes everything into place and gives it additional random colors. The three JavaScript files contain the functionality code for the two algorithms, three buttons, and two scrollers with each button containing its own purpose. The first button allows the user to select a merging algorithm; the second button, allows the user to select the quicksort algorithm, and the third button, allows the user to change the array. The two slide bars change the speed of the algorithm and size of the array, meanwhile, as the algorithm was running, the bar graph displays the array and a color scheme.

My visualizer application had no major errors at first glimpse. The application ran properly, however, the algorithms and color schemes did not properly portray the specific algorithm I had selected. I decided to focus on re-learning each algorithm and enhancing my artifact to portray an accurate display. I learned that a merging algorithm is “a family of algorithms that take multiple sorted lists as input and produce a single list as output, containing all the elements of the lists in sorted order” (Wikimedia Foundation, 2020). It is known as a divide and conquer algorithm. “It divides the input into two halves, calls itself for the two halves, and then merges the two sorted Chart

Description automatically generatedhalves” (Wikimedia Foundation, 2020). The JavaScript contains two different arrays which I arranged the color schemes to reflect one in orange and one in yellow. Visually, the application displays when the two arrays are divided and when they start being sorted and organized. Once they are merged and in chronological order, each bar turns light green. The function will loop until every random number within the array is in order and the visualizer should display a completely light green bar graph, as it can be seen in *figure 1*.

Figure

Chart

Description automatically generated I learned that a quicksort algorithm is comparable to merge sort, it is also known as a divide and conquer algorithm. Quicksort “works by selecting a ‘pivot’ element from the array and partitioning the other elements into two sub-arrays, according to whether they are less than or greater than the pivot” (GeeksforGeeks, 2021). I created two functions within the JavaScript file for the quicksort algorithm, one function oversaw partitioning the data and the other function oversaw merging the data together based on the ‘pivot’ element. I arranged the color scheme to reflect a yellow bar when an item is being reviewed. The random pivot number will be red when an item being compared to it is less than the pivot number. The pivot number will turn orange and be swapped when the item is more than the pivot and will turn pink and will remain in place. When function loops around each bar and it is still out of place, it will be grey, while each bar that is in place will turn green. Once all numbers within the array are in chronological order, each bar will be green, all examples seen in *figure 2*.

Figure

Within this week, I focused on learning more about the algorithms and focusing on how I could properly display the knowledge gained on my visualizer application. After correcting my previous algorithms, I added two additional algorithms: selection and insertion. I first adjusted the HTML code to reflect two additional buttons and imported the JavaScript files. I updated the CSS file to reflect a similar style as the original application, however, this new updated version has more of a lighter theme for the application so that the color schemes of the algorithms being ran could be appreciated. I updated the main sorting JavaScript file, which brings all the algorithms together and is in charge of disabling and enabling buttons throughout the application. Lastly, I created the two new additional JavaScript files, one for each new algorithm.

Chart

Description automatically generatedAfter some research, I learned that selection sorting is considered an in-place algorithm. It sorts an array “by finding the minimum value of the unsorted part and then swapping it with the first unsorted element” (Heydari, 2020). This means that it will review each number in the array, once found, the lowest number is swapped towards the front. Afterwards, it will loop around reviewing each number again until the whole array is chronologically ordered. I arranged the color scheme to reflect all reviewed numbers in yellow, the place holder will be blue, the smallest value found will turn red, and the remaining red until it has been swapped towards the front of the array, as seen in *figure 3.* The current value being compared will also be red and will substitute the previous smallest value found only if it has a smaller value. Once the entire array is in order, all the bars in the bar graph should be light green as previously stated.

Figure

Chart

Description automatically generatedThe last algorithm that I learned about was the insertion sorting. This sorts an array by transferring one element at a time to the correct position. It “compares values in turn, starting with the second value in the list. If this value is greater than the value to the left of it, no changes are made. Otherwise, this value is repeatedly moved left until it meets a value that is less than it” (BBC, 2021). I arranged the color scheme to reflect the number being reviewed in blue and the numbers being organized chronologically in light green, as seen in *figure 4*. Out of the four algorithms I was able to work with and learn from, this was by far the simplest and most straight forward. It was also visually understandable and easily to follow.

Figure

My original goals have been met. I have corrected my previous code and color schemes to reflect accurate information regarding each algorithm being ran, furthermore, I have successfully added two additional algorithms that portray my knowledge and understanding on algorithms, visually. I learned about each of these 4-sorting algorithms, their functionalities, and how to apply them to my own project. I learned specifically about partitioning data and merging arrays, something I previously did not fully understand when I first created this project. I learned about additional algorithms, like linear searches, which are considered “the simplest method of searching data” (BBC, 2021), binary searches, which are “an efficient method for searching ordered lists” (BBC, 2021), and bubble sort, which is “the simplest of the sorting algorithms” (BBC, 2021).

My biggest challenge, throughout this week, was understanding to distinguish each algorithm, its purpose, and the code that would help me display it visually. It was helpful to reference back to the original artifact’s JavaScript files and review how the other two algorithms were coded, which gave me a reminder, in addition, better insight on how to code the rest of the files. Being able to enhance the artifact allowed me to see how my proficiencies have grown along with the expansion of my knowledge on skills like algorithms and data structures. I’ve become more familiar with conditional statements and nested loops ingratitude to this project. Expanding on this project has allowed me to continue practicing and evolving my algorithm and data structure skills. I have selected this artifact for my ePortfolio to showcase and portray an accurate picture of myself and my skills, thus far, attained.

# Works Cited

BBC. (2021). *Common algorithms - Common algorithms - OCR - GCSE Computer Science Revision - OCR - BBC Bitesize*. BBC News. https://www.bbc.co.uk/bitesize/guides/zjdkw6f/revision/.

Heydari, S. (2020). *Selection Sort in Python*. Stack Abuse. https://stackabuse.com/selection-sort-in-python.

*Merge Sort*. GeeksforGeeks. (2021, May 18). https://www.geeksforgeeks.org/merge-sort/.

Wikimedia Foundation. (2020, December 19). *Merge algorithm*. Wikipedia. https://en.wikipedia.org/wiki/Merge\_algorithm.