IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 7.39 ⁜ Vol. 11, Issue 1, January 2022

DOI: 10.17148/IJARCCE.2022.11113

SORTING VISUALIZER USING HTML, CSS, AND JAVASCRIPT

Datta Sai Akash Patchipulusu[1], Dr. A. Vijay Kumar[2]

Department of Computer Science and Engineering FET- Jain University Bangalore, Karnataka, India[1-2]

Abstract: - In the present work we tried to develop a Sorting Visualizer using the technologies like HTML, CSS and JScript. Sorting Visualizer will be displaying the working mechanism of various sorting algorithms like, Bubble Sort, Selection Sort, Insertion Sort, Quick Sort, Heap Sort and Merge Sort. The main objective of developing this Visualizer is to make a learner comfortable in learning these techniques quickly and easily. We know the sorting algorithms are the most widely used algorithms in many applications including Discrete event simulation, Operating Systems, real time systems and many other as well. In the general case the efficiency of an application depends on the efficiency of sorting algorithm used. The only limitation of the Sorting Visualizer is that we should have graphic cards in the general purpose computer.

Key-words: - HTML, CSS, JS, Selection Sort, Bubble Sort, Insertion Sort, Quick Sort, Merge Sort, Heap Sort

1. INTRODUCTION

Learning is a continuous process which will begin just after taking a birth and ends with one's death. Every human should be a consisting learner to get his independence and to adapt to various environmental changes. Here we made a gentle attempt to help the scholars to learn the Sorting algorithms in a friendly, comfortable and easy way to learn the sorting algorithms. The major use of sorting algorithm is to help in arranging the list of elements in some order; it can be ascending or descending order. These sorting algorithms will play a vital role in optimizing the usage of other algorithms, such as search or merge algorithms. Because these algorithm accept only ordered data. More formally, the output must satisfy two conditions:

1. The output is in non-decreasing order (each element is no smaller than the previous element according to the desired total order);
2. The output is a permutation (reordering) of the input. Further, the data is often taken to be in an array, which allows random access, rather than a list, which only allows sequential access, though often algorithms can be applied with suitable modification to either type of data. Since the dawn of computing, the sorting problem has attracted a great deal of research, perhaps due to the complexity of solving it efficiently despite its simple, familiar statement. For example, bubble sort was analyzed as early as 1956.

Visualizing techniques are firstly gaining popularity in the sectors like education, it can be in regular mode or distance mode. The use of visualization techniques in education sector made the education a transit from teacher learner to student learner.

Visualizer allows the users to Generate New Array, select the sorting algorithm and its size and speed of the visualization. Here, each element of the array is displayed as a-bar-graph. To create this visualizer, I had followed the Separation of Concerns principle. Here I want to make the source code into different blocks which are logically related and put them into different files so that if you have a problem with a specific part of the code then we can go directly to that file and solve it. This model is very scalable and it is trending in the industry right now. Because if you want to debug a program this model will help you a lot in the way that there is no need to go through all the code, directly we can go to the particular part of the code we are looking for. Because of this advantage, I had chosen this principle called separation of concerns.

Here, I had created separate JavaScript files for **Bubble Sort, Selection Sort, Insertion Sort, Quick Sort, Heap Sort, Merge Sort** and a Separate JavaScript file for Visualization, all of these are linked by the main JavaScript file.

2. RELATED WORK

Robert Sedgewick, also made an attempt to visualize the Sorting algorithms, he developed some simulator kind of application using Pascal programming language.

Philippe Flajolet is also tried to promote analytics combinatorics, which was majorly based on the complex analysis to enumerate combinatorial structures and to study their asymptotic properties.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 7.39 ⁜ Vol. 11, Issue 1, January 2022

DOI: 10.17148/IJARCCE.2022.11113

3. PROPOSED SYSTEM

In order to better visualize the sorting techniques we have used HTML, CSS, JavaScript software technologies. The DOM manipulations, timer functions are revised using Java Script. To simulate the detailed working mechanisms of Sorting algorithms, event listeners, separation of concern principle the Java Script, CSS and HTML are employed nicely. Those are actually, can be considered as building blocks of the sorting visualizer system.

HTML stands for Hyper Text Markup Language technology, which is generally used for developing general structure of the web sites. This technology is modified and enriched by some technologies in the past few years like CSS and Java Script.

CSS stands for cascading style sheet, generally used for controlling, presentation, formatting and layout. **Using the JavaScript technology we can** control the behavior of different elements.

JavaScript Event Listeners:

For any event which is just occurred we can do two things:

1: Listen to the events.

2: React for the events.

1. Listening for Events:

To listen to the enormous events which will be occurring constantly we have used the addEventListener() function. It is the sole responsibility of the addEventListener() function to listen to the functions and sending alert or informative information to the other parts of the application with in milliseconds.

Usage:

source.addEventListener(eventName, eventHandler, false);

Description: The source is an object for which the addEventListener has to listen for the events. It would be a DOM element, which can be our document, window, or any object specially designed to shoot events.

The Event Handler: It is the function which will be executed to the events upon listening.

Ex: document.addEventListener("click", changeColor, false);

In the above example the document is an object for which we have added addEventListener event listener, first parameter to the addEventListener is click which is a mouse click event over document object, changeColor is an event handler which will be changing the color of the document upon the mouse click event occurs successfully.

2. Reacting to Events

Upon listening to the event which is just happened on the document the addEventListener will in turn calls the event handler. The event handler is a function which will be performing the desired action for the predefined event. The following is a sample code where we can find the event listener and event handler.

```
document.addEventListener("click", changeColor, false);
function changeColor(event) {
  // I am important!!!
}
```

In the above small chunk of code addEventListener is event listener and changecolor is event handler. In the event handler function we will write the block of statements for performing the desired action.

Basic DOM manipulations

The member function querySelector() of the document, will gets the first element of the document that is matched with specific selector or selectors. In the case of no match a null value will be returned.

Syntax: element = document.querySelector(selectors);

Similar to querySelector(), querySelectorAll() will return a constant NodeList, the NodeList will have a list of document's elements which are matching with the specified selectors.

Syntax: elementList = parentNode.querySelectorAll(selectors);

When the above statement is gets executed it will store the static list of document's elements in the elements list. Similarly, when we are only interested in getting the specific element with the known id, can use the getElementById() member function of the document.

Example: const elementRef = document.getElementById('myId').

The **getElementsByTagName()** function will returns an array kind of object that contains elements on the page of a given type. For example <p>s, <a>s etc. The element type is supplied as a parameter to the function. For better understanding consider the below statement,

i.e. const elementRefArray = document.getElementsByTagName('p').

The **createElement()** can be used to create a HTML element mentioned by tag name or HTML unknown element if the tag name is not recognized.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 7.39 ⁜ Vol. 11, Issue 1, January 2022

DOI: 10.17148/IJARCCE.2022.11113

Syntax:
element = document.createElement(tagName[, options]);
In the above statement tagName is a string that specifies the type of element to be created. The nodeName of the created element is initialized with the value of tagName. We should not use the qualified names with this method, for example "html:a". When the above statement is created createElement() converts the tagName into lower case before creating the element. In the web browser like Firefox, Opera, and Chrome, createElement(null) works like createElement("null").
The second parameter to the createElement() is options, which is optional, may have a single property name, whose value is the tag name of a custom element which is defined using customElements.define().
The method createTextNode() will create a new Text node, can be used to flee HTML characters.
Syntax: var text = document.createTextNode(data);

A **window object** represents browser's window that is supported by all the browsers. The functions, variables and Global objects like JavaScript are the members of the window object. The general properties of the window object are the global variables. Whereas global functions are, the members of the window object. The HTML DOM is a document object of a window.
The below statement 1 is similar to statement 2.
Ex: window.document.getElementById("header"); -----1
document.getElementById("header"); -----2

The innerHTML property sets or returns the HTML content of an element,
Example: HTMLElementObject.innerHTML= text

The class list property returns the class name of an element as a DOMTokenListObject, Which can be used to add, remove and toggle CSS classes on an element. The class list property is read only still we can modify that using the add() and remove() methods.
Syntax:element.classList

The disabled property can be used to set or reset the dropdown list. Once an element is disabled it can't be usable and clickable. The disabled elements are displayed in gray color by default in the browsers.
To get the disabled property we need to write the statement as follows
selectObject.disabled
To set or to unset the selected object we need to write the statement as follows
selectObject.disabled = true|false
By setting the true value to the disabled property we can disable the selected object. And similarly by setting the false value to the disabled property we can make the property enable.

Working with Timers:
To specifically execute a function at a given time we can make use of timer function. For example if we wish to delay the execution of the code, means here we wish to execute the code not at the moment of the event is triggered or the page has been loaded. We can make use of these timers to change the advertisements at the regular intervals or disable a real time clock, etc.
In JavaScript we have two types of timer functions,
1: setTimeout()
The setTimeout() function can be used to perform a task or operation just after completion of the specified time interval. The general syntax of the function is as follows:
setTimeout(function, milliseconds)
The function accepts two parameters function and time in milliseconds. The function is nothing but the piece of code we wants to execute and time in milliseconds represents the duration after which we wants to execute the code.
2: setInterval().
To execute the code in the regular intervals we can use setInterval(). To this method we need to supply two parameters, function and milliseconds similar to setInterval.
setInterval(function, milliseconds)
To stop code execution or Cancel a Timer we can make use of two functions, setTimeout() or setInterval(). Both of the above functions will return an unique ID, i.e. Timer id, which identifies the timer created by these methods. Timer can be cleared using two functions,
1: clearTimeout()
2: clearInterval().

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 7.39 ⁜ Vol. 11, Issue 1, January 2022

DOI: 10.17148/IJARCCE.2022.11113

The setTimeout() function takes a single parameter, an ID, and clear a setTimeout() timer associated with that ID. Similarly, the clearInterval() method is used to clear or disable a setInterval() timer.

4. SORTING ALGORITHMS

A. Selection Sort

Selection sort is a simple iterative and in place sorting technique. Selection sort logic is very interesting in this technique every time we pick a smallest element and try to place it in the appropriate place. If there are n elements to be sorted it takes n-1 iterations, in each iteration it takes n-I comparisons where I represents number of the iteration. Though selection sort is simple and interesting people won't prefer this because it takes O (n^2) computational steps in the average and worst cases.

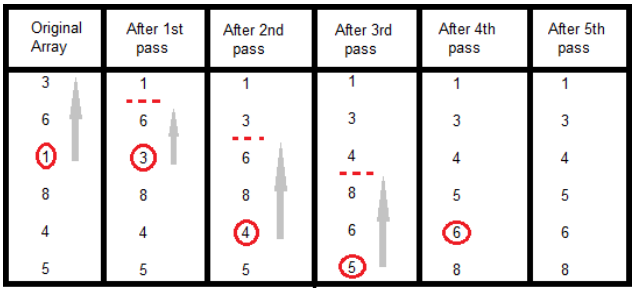

Figure-1. Selection sort.

B. Insertion Sort

Insertion sort technique is one of the simple in place sorting technique. It is also called as internal sorting technique because it won't require additional memory for sorting the elements. To sort given n elements it takes n-1 iterations. In this sorting technique every time pick some element and place it in the appropriate location. The logic of the insertion sort is similar to the playing cards game, in which also every time pick some new card and try to place it the appropriate place. Generally, people won't prefer insertion sort because of its performance. It's average and worst case performance is O (n^2).

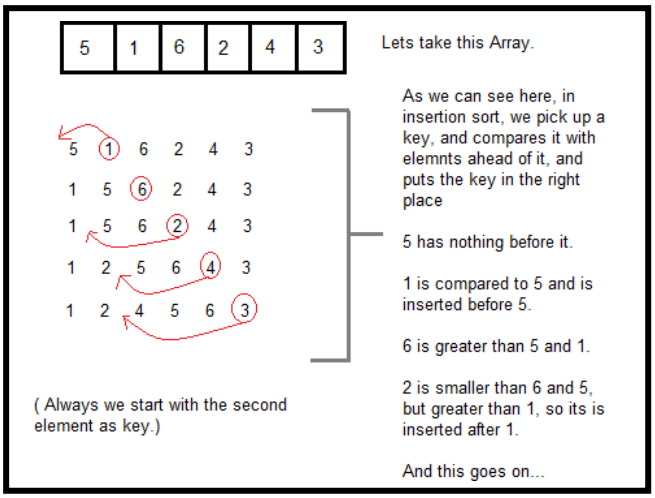

Figure-2. Insertion sort.

C. Bubble Sort

Bubble sort is one of the simple, easy to understand and implement sorting technique. Though it is simple, easy to understand and implement sorting technique we won't prefer it, because it is not an efficient sorting technique compared with quick sort and merge sort. In the best case it requires O (n) computation steps, in the average and worst cases it requires O (n^2) computation steps.

Logic: In bubble sort every time we compare two adjacent elements, if they are in the expected order we leave them as they are otherwise we swap them. If there are n elements to be sorted then we require n-1 iterations. For each and every

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 7.39 ⁜ Vol. 11, Issue 1, January 2022

DOI: 10.17148/IJARCCE.2022.11113

round we perform n-i comparisons where i, represents iteration number. The specialty of bubble sort is in this the smallest element will bubble up and largest element will sink.

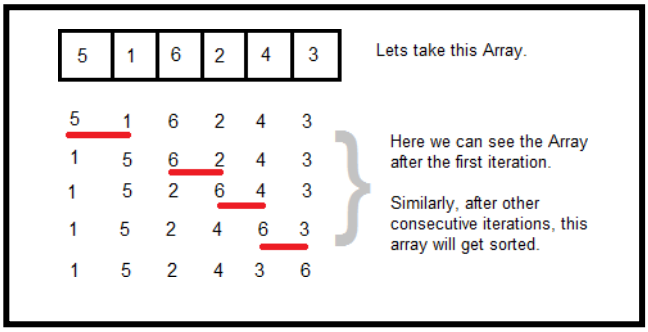

Figure-3: Bubble sort.

D. Quick Sort

Quick sort is an efficient sorting technique compared with merge sort and other sorting techniques. Quick sort follows divide and conquer technique for problem solving. This recursive technique is proposed by Tony Hoare. In quick sort to sort the given n numbers, one is picked as the pivot, the elements which are less than the pivot are placed on the left side to the pivot and greater elements are placed on the right side of the pivot. We can select any element as the pivot. Generally we pick the first, last or the middle.

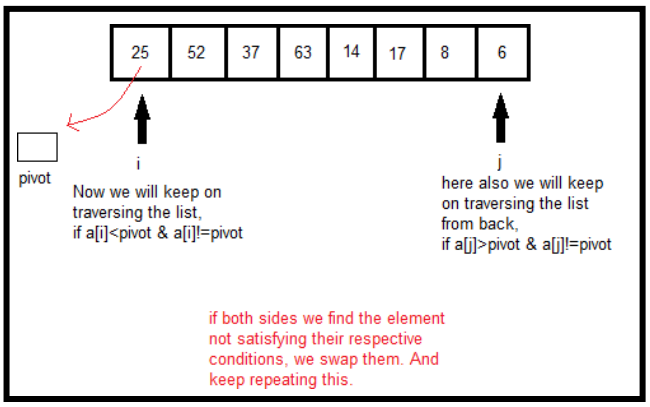

Figure4: Quick sort.

E. Merge Sort

Like Quick Sort, Merge Sort is also a divide and conquer technique based algorithm. It is proposed by John Von Neuman in 1945. Merge Sort algorithm divides the given array of elements into two sub arrays of equal size. Again the two sub arrays are also divided into two of the equal size until we get a sub array with a single element. In the next step these sorted sub arrays are merged. Merge sort time complexity is O(nlogn) in all the 3 (best, average and worst) cases.

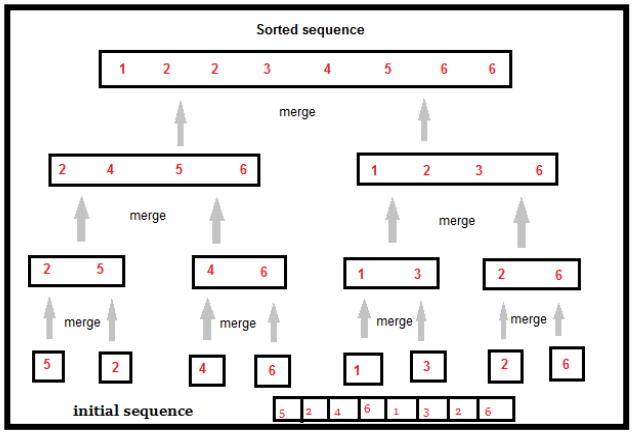

Figure-5. Merge sort.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 7.39 ⁜ Vol. 11, Issue 1, January 2022

DOI: 10.17148/IJARCCE.2022.11113

F. Heap Sort

It is one of the comparison based sorting techniques which will be commonly using Binary Heap data structure. Heap sort will works very similar to selection sort, in which first the minimum element is found and it will be placed in the beginning. In the same manner rest of the all elements will be processed. It will be less used compared with other sorting techniques like QuickSort and MergeSort. There are some best things with Heap sort also, they are as follows:

1: Efficiency – The time taken by HeapSort will increases logarithmically while other techniques time will increase exponentially. The algorithm will be slow as the no of items to sort increases.

2: Memory Usage – Memory usage is minimal compared to other techniques.

3: Simplicity – It is very simple to understand and use.

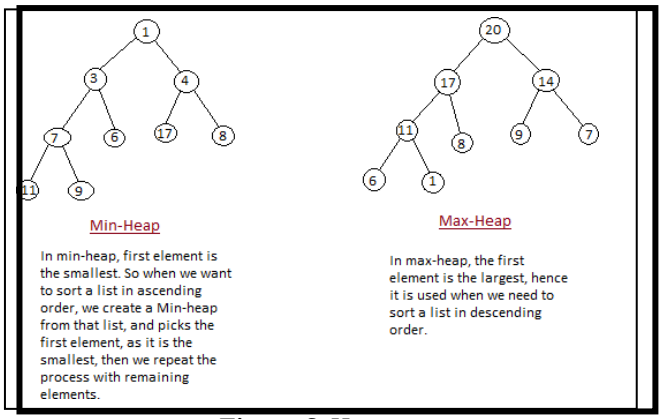

Figure-6. Heap sort

5. VISUALIZATION

Visualization techniques are generally used for the graphical representation or presentation of data or information. This visualizations or simulations produce lot of files that contains data values. Data are converted into visual form so that it can be analyzed to solve the problems. Science, engineering and medicine fields uses graphical form for information representation. Whereas business, management, social, industry and other non scientific areas uses visualization techniques known as Business Visualization[7].

6. METHOD OF SORTING

We have a very long list of sorting techniques in the market, but among them very few are very useful and dominating the remaining algorithms. Among these algorithms each is having its specialty, for example Insertion sort is best suited for small datasets, for the larger data sets heap sort, merge sort or quick sort are used. Each algorithm is special with its advantages, it is because of the different mechanism is used.

Pseudo code:

Bubble Sort (A) [8] for i ← 1 to length[A] do for j ← length[A] down to i + 1 do if A[j] < A[j - 1] then exchange A[j] & A[j - 1]	Insertion Sort (A) [8] for j ← 2 to length[A] do key ← A[j] Insert A[j] into the sorted sequence A[1 : j - 1]. i ← j - 1 while i > 0 and A[i] > key do A[i + 1] ← A[i] i ← i - 1 A[i + 1] ← key
Selection Sort (A) [8] for i ← 1 to length[A] – 1 do min ← i for j ← i + 1 to length[A] do if A[j] < A[min] then min ← j exchange A[i] & A[min]	Merge Sort (A, P, R) [8] if p < r then q ← (p + r)/2 MERGE-SORT(A, p, q) MERGE-SORT(A, q + 1, r) MERGE(A, p, q, r)

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 7.39 ⁜ Vol. 11, Issue 1, January 2022

DOI: 10.17148/IJARCCE.2022.11113

Heap Sort (A) [8] BUILD-MAX-HEAP(A) for i ← length[A] downto 2 do exchange A[1] ↔ A[i] heap-size[A] ← heap- size[A] - 1 MAX-HEAPIFY(A, 1)	Quick Sort (A, P, R) [8] if p < r then q ← PARTITION(A, p, r) QUICKSORT(A, p, q - 1) QUICKSORT(A, q + 1, r)

7. VISUALIZATION OF ALGORITHMS

By using Visualizer we can create new array, choose the size of the array, speed. The speed and size of the array are divided into 5 ranges. The speed of the visualization is directly depends upon the size of the array. The size of the array directly determines width and block size. The sorting process majorly depends upon the sorting algorithm used, array size, so the time taken to visualize the sorting process may also change. In the present work, to keep the visualization process time same for all the algorithms we have normalized the speed of the sorting algorithms with respect to the speed selected by the user. When we choose generate new array, a new array with some random size is created using Math.random() and array is visualized in the form of a bar graphs. We have added a very interesting feature to the visualizer that is buttons and controls, which can be unclickable or locked during the running to avoid problems like missing the flow of algorithms.

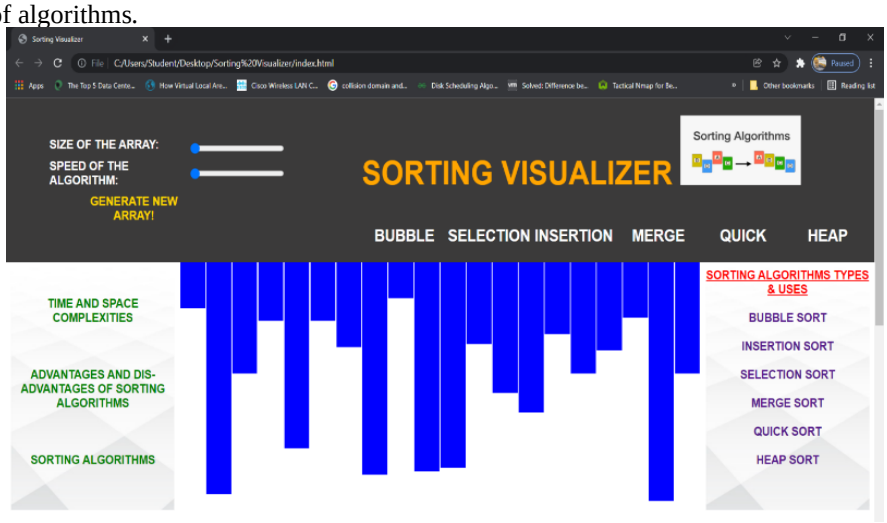

Figure-7. Main window of Sorting Visualizer.

For the effective visual effects we have used different colors such that: Red – Swap; Blue – Comparison; Green - Element is in sorted position. These color updates are visualized during the process of sorting algorithms.

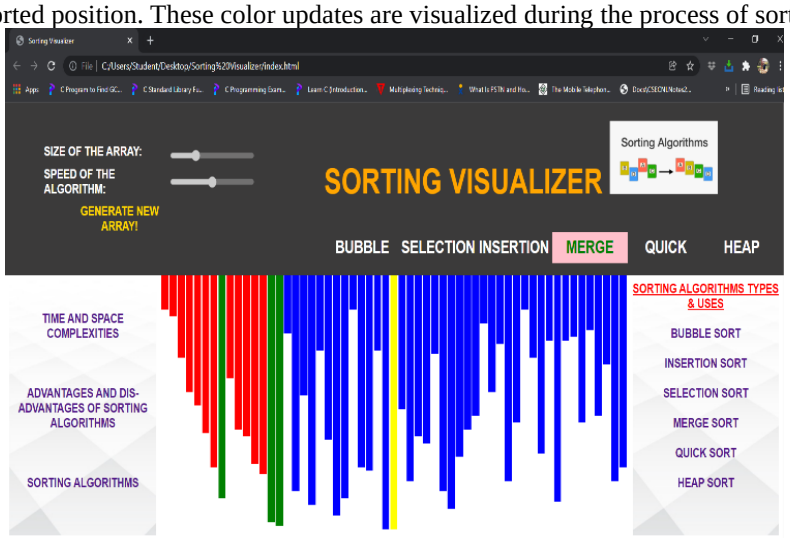

Figure-8. View of Merge Sorting

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 7.39 ⁜ Vol. 11, Issue 1, January 2022

DOI: 10.17148/IJARCCE.2022.11113

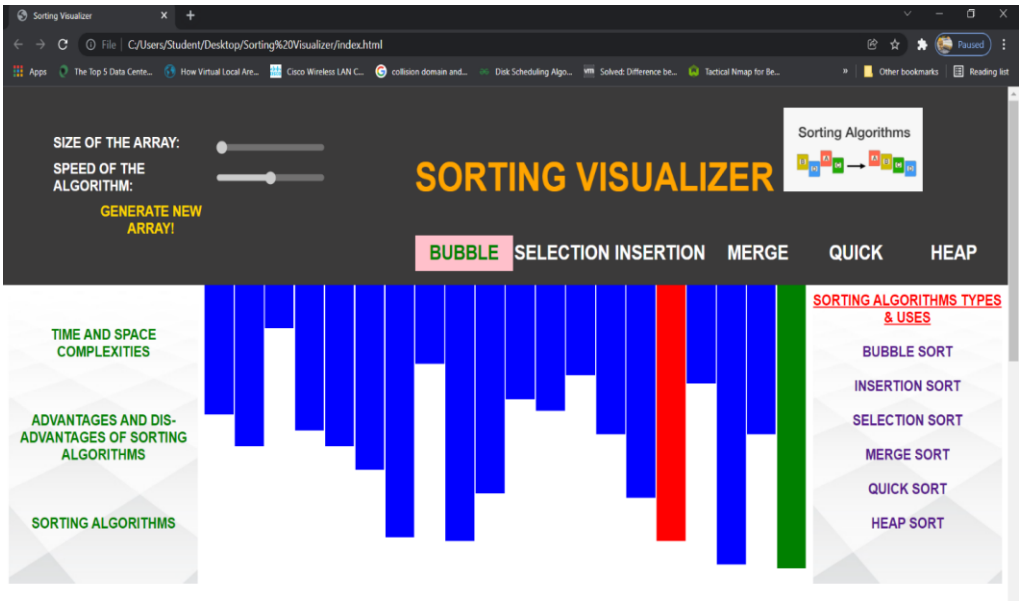

Figure-9. View of Bubble Sorting

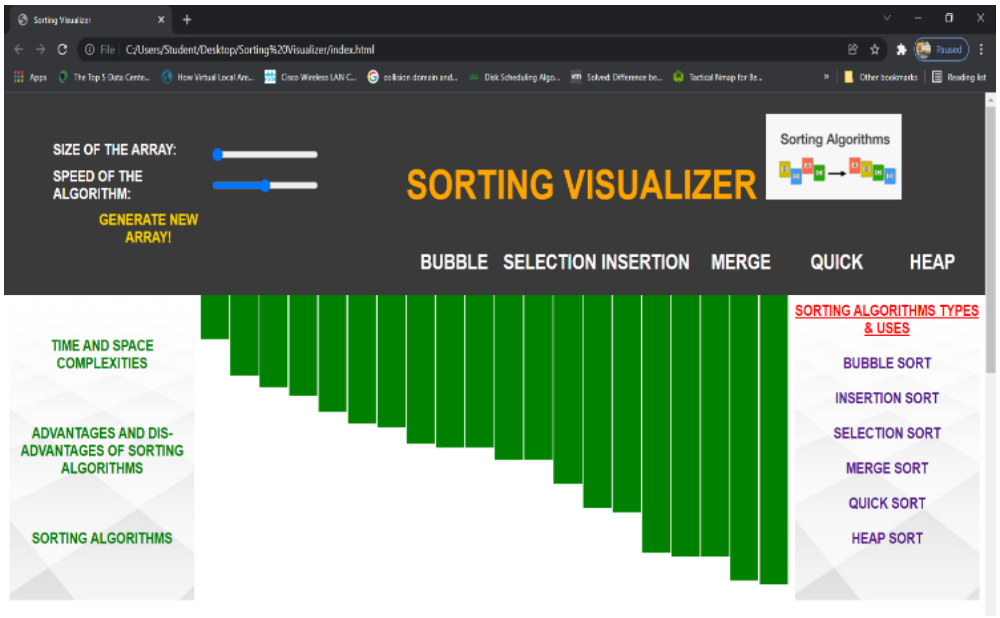

Figure-10. View after the array is sorted

7. CONCLUSION

One and only aim of the present project is to demonstrate the sorting algorithms using visualization through a website. This project aims to demonstrate the power of sorting algorithms through this website. Our website provides the clear and detail idea about the sorting algorithms, and more over how the comparisons and swaps are performed during the sorting. The visualizer will help the students to better understand about the different sorting algorithms considered in a very less time.

8. REFERENCES

[1] Semiawan, Conny R, (2009) Landasan Pembelajaran dalam Perkembangan Manusia, Jakarta: Center for Human Capacity Development.

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 7.39 ⁜ Vol. 11, Issue 1, January 2022

DOI: 10.17148/IJARCCE.2022.11113

[2] Sfenrianto, (2009) "A Model of Adaptive E-Learning System Based on Student's Motivation", Proceedings from ICCIT-09: International Conference on Creative Communication and Innovative Technology, 2009. Tangerang: CCIT Journal.

[3] Algorithms in Java, Parts 1-4, 3rd edition by Robert Sedgewick. Addison Wesley, 2003.

[4] Programming Pearls by Jon Bentley. Addison Wesley, 1986.

[5] Quicksort is Optimal by Robert Sedgewick and Jon Bentley, Knuthfest, Stanford University, January, 2002.

[6] Sorting Out Sorting, Ronald M. Baecker with the assistance of David Sherman, 30 minute color sound film, Dynamic Graphics Project, University of Toronto, 1981. Excerpted and reprinted in SIGGRAPH Video Review 7, 1983. Distributed by Morgan Kaufmann, Publishers.

[7] Hearn, Donald, and Pauline Baker, (1996) Computer Graphics C Version, 2nd edition. Upper Saddle River, NJ: Prentice Hall International, Inc.

[8] Reyha Verma and Jasbir Singh. 2015. A Comparative Analysis of Deterministic Sorting Algorithms based on Runtime and Count of Various Operations. International Journal of Advanced Computer Research (IJACR). 5(21): 380-385.

[9] B. Miller, D. Ranum. 2013. Problem Solving with Algorithms and Data Structures. [https://www.cs.auckland.ac.nz/compsci105s1c/resources/ProblemSolvingwithAlgorithmsandDataStructures.pdf]

