**VisuAlgo – Visualization of Sorting Algorithms**

**Delhi Technological University**

**Department of Information Technology**



**Semester:** August 2020 – November 2020

**Project Report for Discrete Structures**

**Submitted to:** Mrs. Swati Sharda

**Submitted by:** Mustafa Noman Rashid (2K19/IT/083)

**Class:** IT – B

**DELHI TECHNOLOGICAL UNIVERSITY**

**Declaration**

I, Mustafa Noman Rashid declare that is my original work. The work submitted by me in partial fulfilment of the requirement for the award of degree Bachelor of Technology in Information Technology is our own; it has not previously been presented for another assessment. Wherever work form other source has been used, all debts (for words data, arguments and ideas) have been appropriately acknowledged and referenced. I have not used work previously produced by another student or any other person to submit it as my own. I have not permitted, and will not permit, anybody to copy our work with the purpose of passing it off as his or her own work.

Date: ------------- Mustafa Noman Rashid

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**DELHI TECHNOLOGICAL UNIVERSITY**

**Certificate**

Certified that this report titled “VisuAlgo - Visualization of Sorting Algorithms” is prepared based on the Project undertaken by us in Delhi Technological University for the Semester of August 2020 - November 2020, under the able guidance of Mrs. Swati Sharda in partial fulfilment of the requirement for award of degree of Bachelors of Technology from Delhi Technological University, Delhi.

Date: ------------- Swati Sharda

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**DELHI TECHNOLOGICAL UNIVERSITY**

**Acknowledgment**

To make a project, proper guidance and references are an essential. We are thankful to our teacher, Mrs. Swati Sharda who has provided us an opportunity to gain such knowledge and whose valuable guidance has been the one that helped us patch this project and make it full proof success. Her suggestions and instructions has served as the major contributor towards the completion of this project. The experience from this project will help us in our future.

**Abstract**

The internet is a very useful place to learn various different subjects. The developed website for this project will provide a visual learning experience for students looking to learn about sorting algorithms in a simple and easy to use web-based format. There has been a wide shift towards online mode of education due to the pandemic. For such reasons, this website can be used for easy and effective learning purposes. The expectation for this idea is for the visual learners to have a slightly easier understanding of the sorting algorithms which play an important role in the journey of a student from a computer related field.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Serial Number** | **Topic** | **Page Number** |
| 1. | Abstract | 5 |
| 2. | Index | 6 |
| 3. | Objective | 7 |
| 4. | Introduction | 8 |
| 5. | Algorithms implemented | 9 |
| 6. | Languages used | 14 |
| 7. | Github Link | 15 |
| 8. | Website Link | 15 |

**Objective**

To create a platform for the visualization of sorting algorithms which can help in better understanding and learning of various sorting algorithms in a simple and easy way.

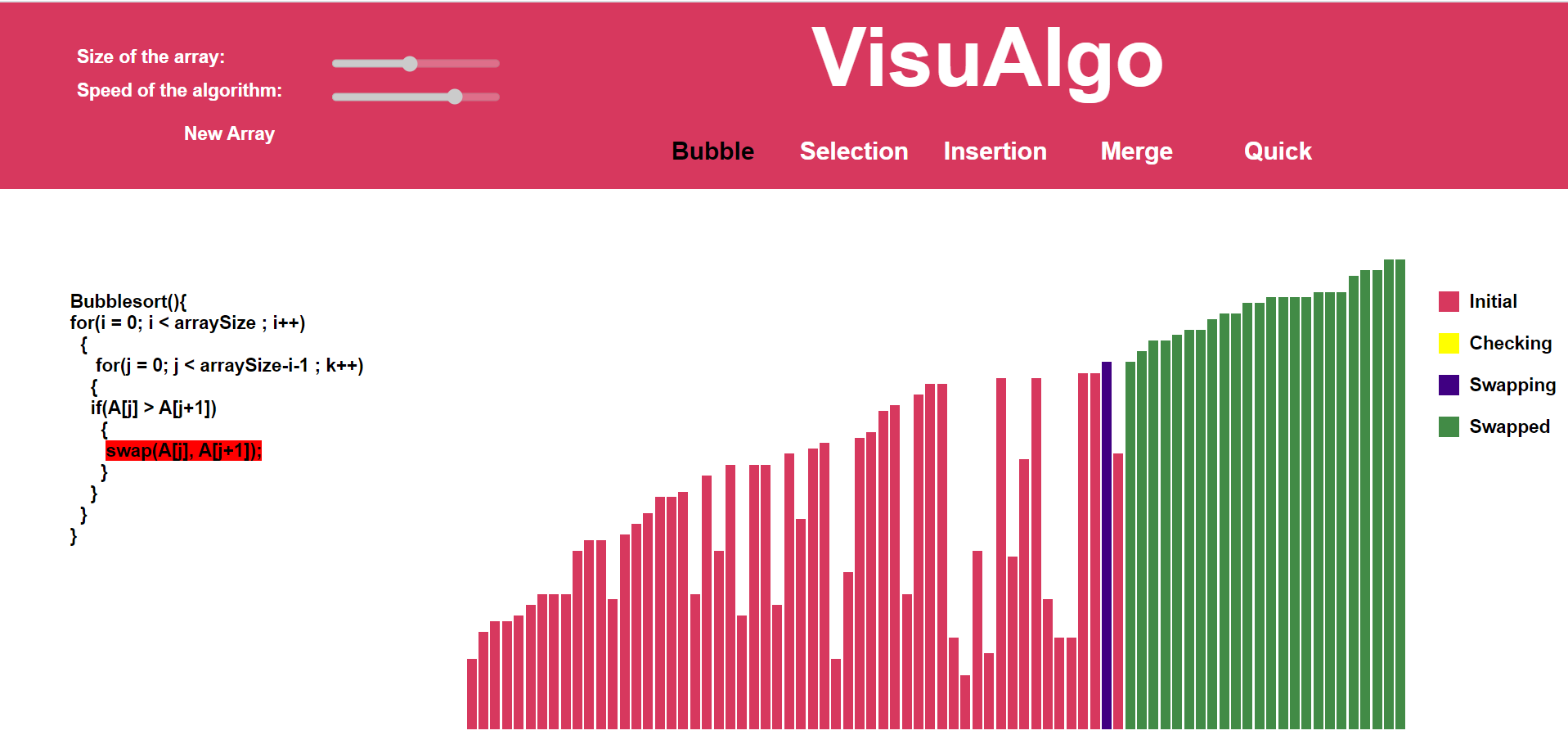
**Introduction**

There are many times that a student from a computer field will need to use a sorting algorithm that they may not understand completely or it can be someone completely new to programming. Many of these students will turn to the internet to understand the information they need. This website can also be used by the teachers to provide the students with proper material for sorting algorithms. The expectations for this idea is for the visual learners to have a slightly easier understanding of the sorting algorithms. Majority of the sorting algorithms tutorials available on the internet are text based. This project will aim on providing a complete visual platform to understand the various sorting algorithms.

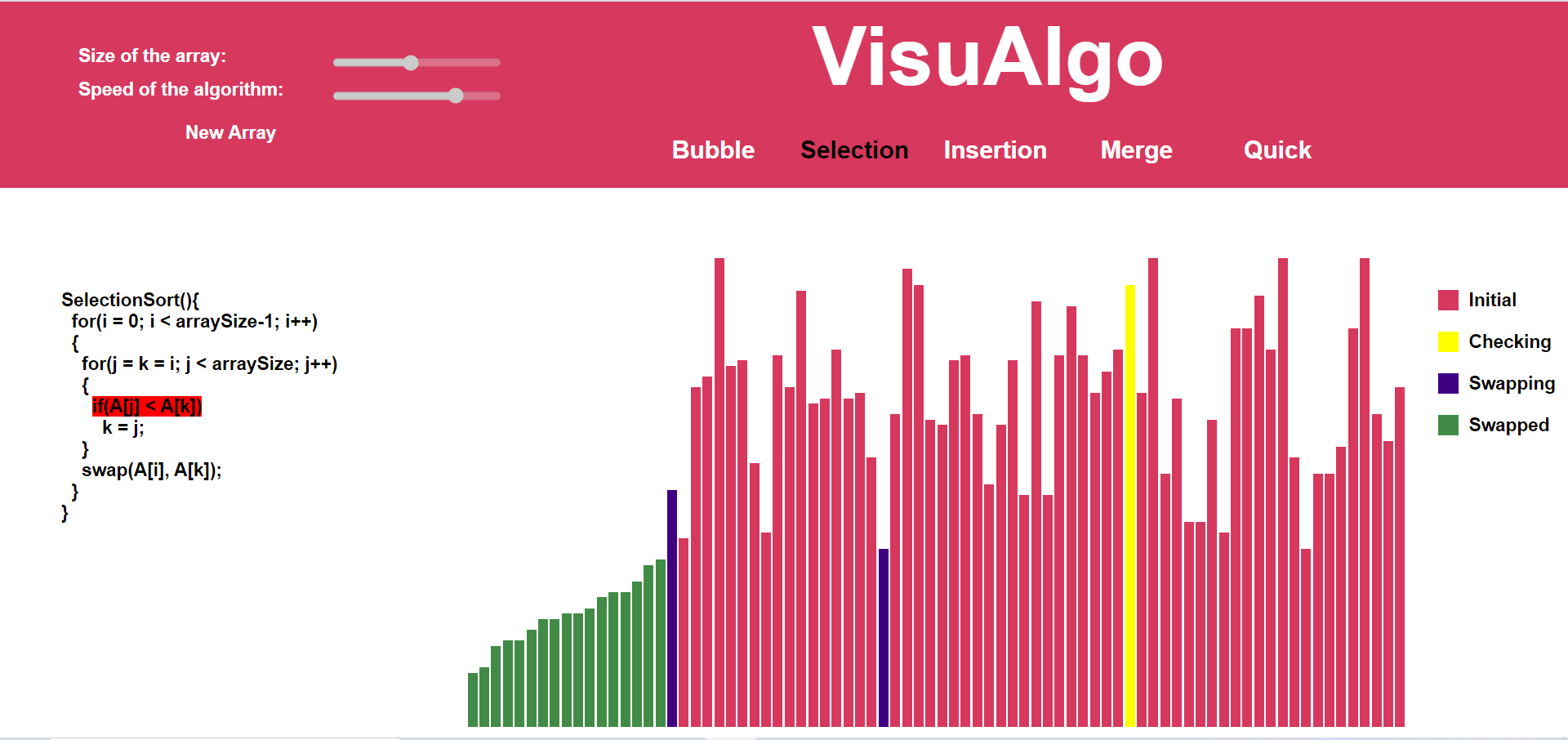
**Algortihms Implemented**

Initially, there will be a various number of bars with different random variable length. Once the particular algorithm is selected, the bars start sorting themselves along with the code for that particular algorithm shown on the side

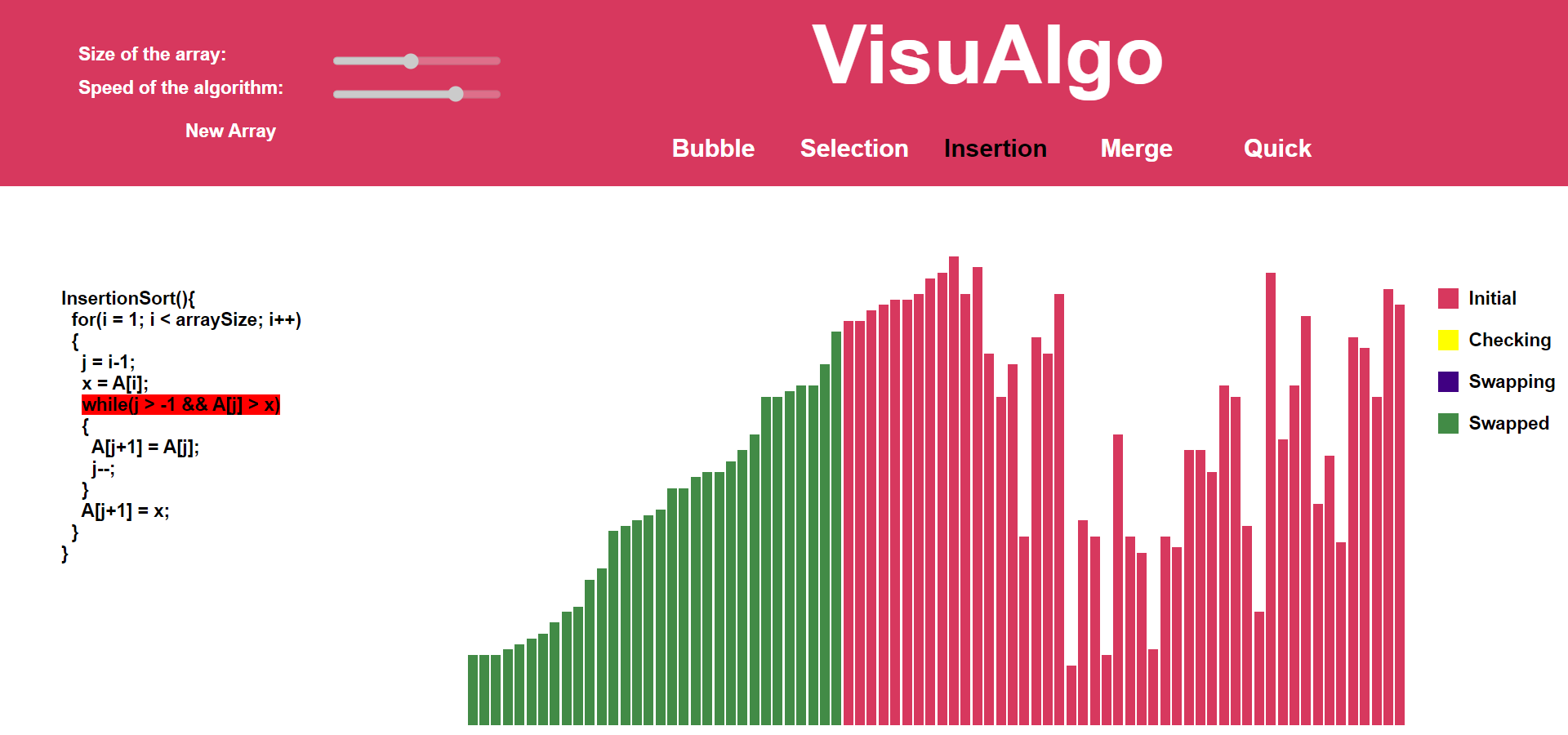
**Bubble Sort**-Bubble sort, sometimes referred to as sinking sort, is a simple [sorting algorithm](https://en.wikipedia.org/wiki/Sorting_algorithm) that repeatedly steps through the list, compares adjacent elements and [swaps](https://en.wikipedia.org/wiki/Swap_(computer_science)) them if they are in the wrong order. The pass through the list is repeated until the list is sorted. The algorithm, which is a [comparison sort](https://en.wikipedia.org/wiki/Comparison_sort), is named for the way smaller or larger elements "bubble" to the top of the list.



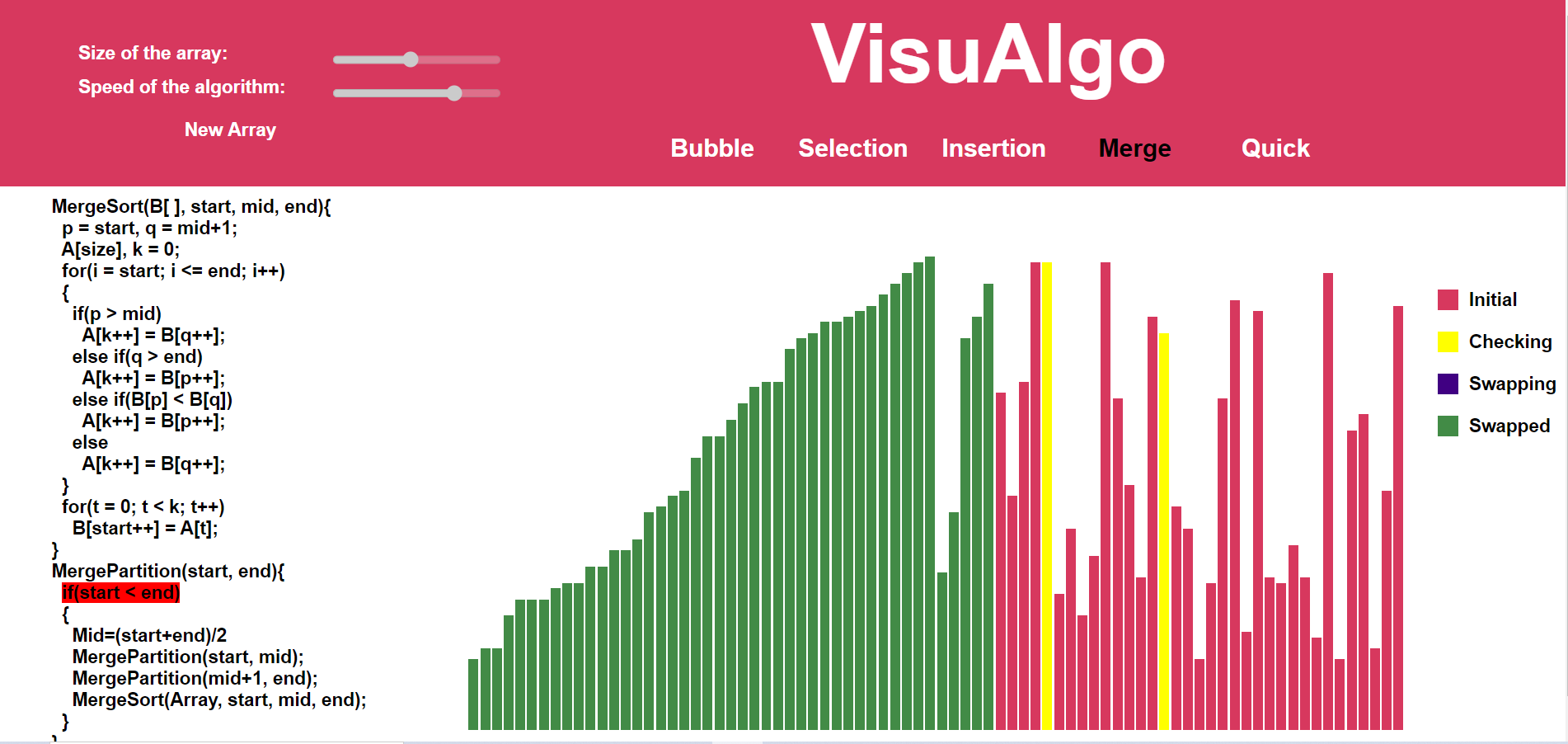
**Selection Sort-** The algorithm divides the input list into two parts: a sorted sublist of items which is built up from left to right at the front (left) of the list and a sublist of the remaining unsorted items that occupy the rest of the list. Initially, the sorted sublist is empty and the unsorted sublist is the entire input list. The algorithm proceeds by finding the smallest (or largest, depending on sorting order) element in the unsorted sublist, exchanging (swapping) it with the leftmost unsorted element (putting it in sorted order), and moving the sublist boundaries one element to the right.

****

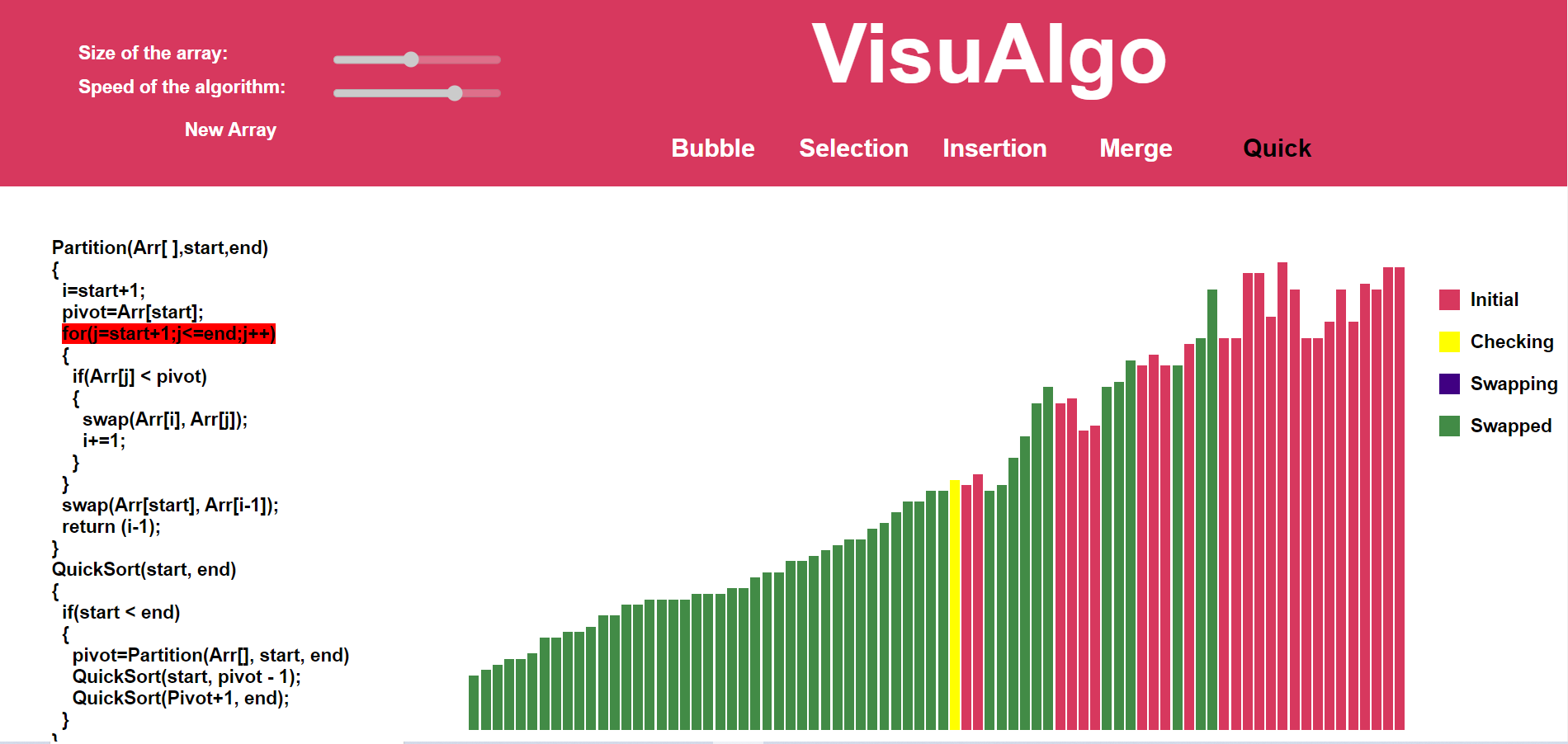
**Insertion Sort-** Insertion sort is based on the idea that one element from the input elements is consumed in each iteration to find its correct position i.e, the position to which it belongs in a sorted array. It iterates the input elements by growing the sorted array at each iteration. It compares the current element with the largest value in the sorted array. If the current element is greater, then it leaves the element in its place and moves on to the next element else it finds its correct position in the sorted array and moves it to that position. This is done by shifting all the elements, which are larger than the current element, in the sorted array to one position ahead.



**Merge Sort-** Merge sort is one of the most efficient sorting algorithms. It works on the principle of Divide and Conquer. Merge sort repeatedly breaks down a list into several sublists until each sublist consists of a single element and merging those sublists in a manner that results into a sorted list.



**Quick Sort-** Quicksort is a [divide-and-conquer algorithm](https://en.wikipedia.org/wiki/Divide-and-conquer_algorithm). It 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. The sub-arrays are then sorted [recursively](https://en.wikipedia.org/wiki/Recursion_(computer_science)). This can be done [in-place](https://en.wikipedia.org/wiki/In-place_algorithm), requiring small additional amounts of [memory](https://en.wikipedia.org/wiki/Main_memory) to perform the sorting. Quicksort is a [comparison sort](https://en.wikipedia.org/wiki/Comparison_sort), meaning that it can sort items of any type for which a "less-than" relation (formally, a [total order](https://en.wikipedia.org/wiki/Total_order)) is defined. Efficient implementations of Quicksort are not a [stable sort](https://en.wikipedia.org/wiki/Stable_sort), meaning that the relative order of equal sort items is not preserved.



**Languages Used**

**Frontend used-**

HTML and CSS

**Backend used-**

Javascript and Jquery

**Github Link**

<https://github.com/mnrmustafa/VisuAlgo>

**Website Link**

<https://mnrmustafa.github.io/VisuAlgo/>