# Interactive Explorable explanation for Quicksort algorithm

Hemant Joshi, Bhalchandra Shirole, *INFX 598*

**Abstract**—Quicksort is an efficient sorting algorithm that finds applications in several areas. On analysing the algorithm, one finds that the implementation using code is a little bit different from the high-level understanding of the algorithm. Through this website, we intend to create an easy, interactive way for the user to understand the algorithm. We implement the algorithm in three steps, *Explanation, Simulation and Exploration*. In this paper, we discuss the implementation of this tutorial.

# 1. INTRODUCTION

Quicksort is an efficient method of sorting arrays that has found widespread applications in Computer Science and related fields. When implemented well, it can be two-three times faster than its competitors, namely Merge Sort and Heap sort.

The algorithm was developed by Tony Hoare in 1959, and has grown in popularity since. It is based on comparisons, and can be applied for all items for which a less-than relationship can be established. To establish an ordering, elements are compared with one another to see which element is greater[1]. The algorithm works on placing elements in their right order. That is, an element should be placed in an array such that all the elements smaller than that element are towards its left and those that are greater than it are towards its right.

The problem with the algorithm is that while it can be easily explained at a higher level, the implementation details of the algorithm with respect to the code written is somewhat tricky. It needs us to process the array of elements iteratively, at the end of which, we get an array of sorted elements.

That being said, we found it intriguing if an interactive explanation could help solve the problem. If a person needs to implement an algorithm using code, it would be better if he can visually see how the algorithm works. Considering this, we decided to create a bar-chart based visualization that deals with the problem.

To help explain the algorithm completely, and yet allow room for exploring the algorithm, we used 3 steps: *Explanation, Simulation and Exploration.* The first step gives a high level understanding of the algorithm, which gives us a conceptual idea of the algorithm. The second step gives the user a stepwise implementation of one iteration of the algorithm. The third step of the algorithm is a mini-test where the user has to apply the principles he learned and provide him with the freedom to use his own techniques. We believe the user should use his own judgement and realize by trying the algorithm why it is designed the way it is. The best way to do it is by allowing him/her freedom to explore.

# 2. RELATED WORK

While we defined the scope for our work, we planned to build an a visualization which will not only demo the Quick sort in an interactive, step by step way with great cues but will also provide a mechanism through which the users can practice quick sort and master the algorithm through our visualization & supplementary explanation.

Our research couldn’t find a solid work done on building such an explorable explanation for the algorithm. However, we found out some visualizations created to explain the sorting algorithm. The main factor that these visualizations lack is they don’t allow users to explore the explanations through interactions.

One author provides an interactive explanation for the quick sort but it’s not very intuitive. Moreover, none of these provide an opportunity to the users to explore the technique in visual representation and master it. Also, none of the visualization explains the time complexity of the algorithm in a visual manner. We will take a look at all of these one by one.

The Wikipedia page for Quicksort [1] gives a visual explanation in the form of a gif image. The animation very quickly proceeds through the steps of selecting the pivot element, comparing & swapping the elements on the left & right. The transitions are very fast & there are no interactions. Also, there are no annotations or guided tips. Hence, it’s quite difficult to understand the process of sorting. Lastly, there is no way that the user could practice the sorting an array on his/her own using quick sort.

The visual explanation on the Algomation[2] has very nice cues for the visualization of the Quick sort. Also, it gives a way so that the user can control the speed of the transition and also run the animation in a step by step by manner. This way it demos the Quick sort in a much better way. On the other hand, there is no way the users can interact with the visualization and practice the Quick sort algorithm. It is just a pure demo. Also, the explanation for the various steps is not very exhaustive.

The third visualization that we found for quick sort is a blogpost by the author Thai Pangsakulyanont[3]. The author provide a very detailed explanation for the various aspects of the Quick sort algorithm like selection of the pivot element, partitioning the dataset, comparison & swapping of the elements using his visual depictions of an array. The users can iterate through each step if the algorithm and see what happens in each steps. The main problem with the visualization is it does not uses squares of equal size for all the elements of the array. Also, as in the above visualization, even this does not provide the mechanism for the users to practice the sorting algorithm in their own and master it.

# 3. METHODOLOGY

The best way to explain a computer algorithm would be to show how the computer would use it. There are several common explanations that can be found in books that provide some diagrams at the most to explain an algorithm. But with the advent of technology providing us the tools to explain complex concepts in an easier way, it is necessary for us to use these tools.

One such tool is explorable explanation[4]. These explanations allow us to use exploratory tools and allow the user himself to learn the concepts through a process of discovering new tricks and concepts at every step. With this in mind, we started thinking of how to implement the quicksort algorithm.

3.1 High-level details

After looking at several implementations of the quicksort algorithm as described in the previous section, we thought of providing a step-by-step validation mechanism to the user, such that at each step his actions will be validated. This was close to what has been shown in the explanations we came across. But soon after, deliberating at the purpose of allowing the user to explore, we decided to provide the user with small hints at steps and visual cues to tell him if he is proceeding in the right manner or not. Coming to this decision was easy, but to implement this, we had to decide on steps at which we should provide cues and those steps where we shouldn’t.

After deliberating further on the issue, we had come up with a plan to create a three-step visualization as explained further ahead in this paper.

3.2 Low-level details

The quicksort algorithm works with a special element called the pivot element, which is selected for every iteration. The algorithm says that for each array, we need to place this pivot element in such a way that all the elements that are smaller than it are to the left of the pivot and all the elements that are greater than it are to the right of the pivot. This is considering we want to create an ascending order. If we want a descending order of elements, then the opposite has to happen. After one iteration is done, the pivot element is placed in its position in the final array.

With this clearly understood, we detailed the process of explaining the algorithm as follows:

## 3.2.1 Step 1: Explanation

In this step, we give an explanation of the algorithm such that the user can visually see what is happening when a pivot element is selected. This is done using simple transitions and animations that allow us for an interactive visualization for the algorithm.

## 3.2.2 Step 2: Simulation

Having understood the high-level working of the quicksort, we move on to a detailed description of how the process that was shown in the first step actually happens. Here, using a 10-step slider, we show what happens in each step of the algorithm when we code it. While doing this, we decided to show only a single iteration of the process, assuming that the user will be able to decipher what would happen in the coming steps.

## 3.2.3 Step 3: Exploration

This is the most critical step of our tutorial. Once we have given all the necessary details of the algorithm to the user, we move on to testing the user. The test does not provide validation at each step, but consists enough hints and cues at the end of each step to allow the user to follow the procedure we have stated earlier. Through trial and error, we expect the user to eventually be able to find out how the complexity of the algorithm can be brought down. The way this happens is through selection of the right pivot element.

3.3. Working And Results

The first thing that a user sees on the website is a description of the algorithm, giving cues about the complexity of the algorithm. Since we expect the users of the website to have some idea about complexity and notations associated with it, we decided to just state the complexity of the algorithm. With the algorithm’s gist clear to the user, the array shown on the webpage allows the user to select a pivot. When clicked on an element, the system selects it as a pivot element and realigns the elements of the array accordingly. The explanations given tell the user what happens.

The transitions added on click might allow the user to focus attention on elements which are sorted and the pivot element only. An intermediate step to sort the array shown is as below:

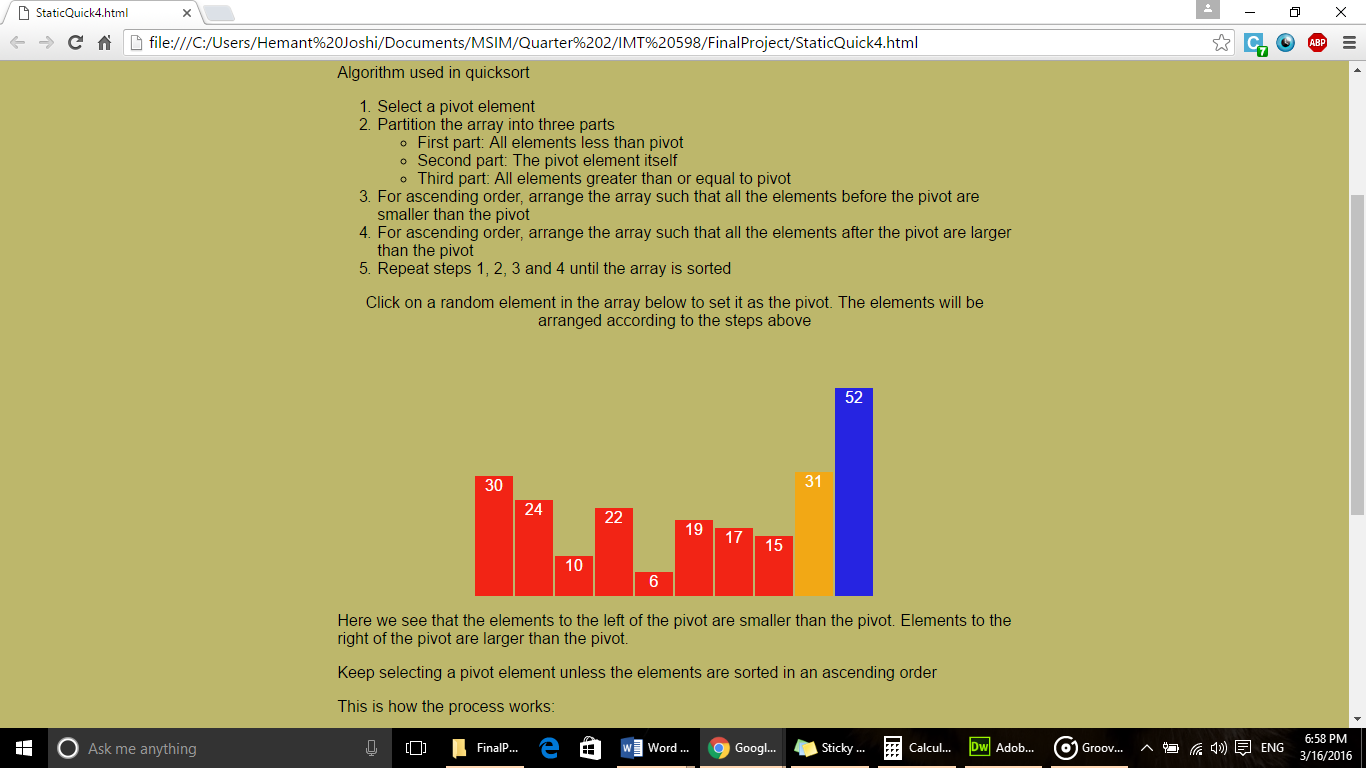


Figure 1

Here, the element 31 is selected as the pivot element. All the elements less than 31 are towards the left of it, and 52 being the only element greater than it, is towards the right of 31. In a similar manner, subsequent pivot elements can be selected.

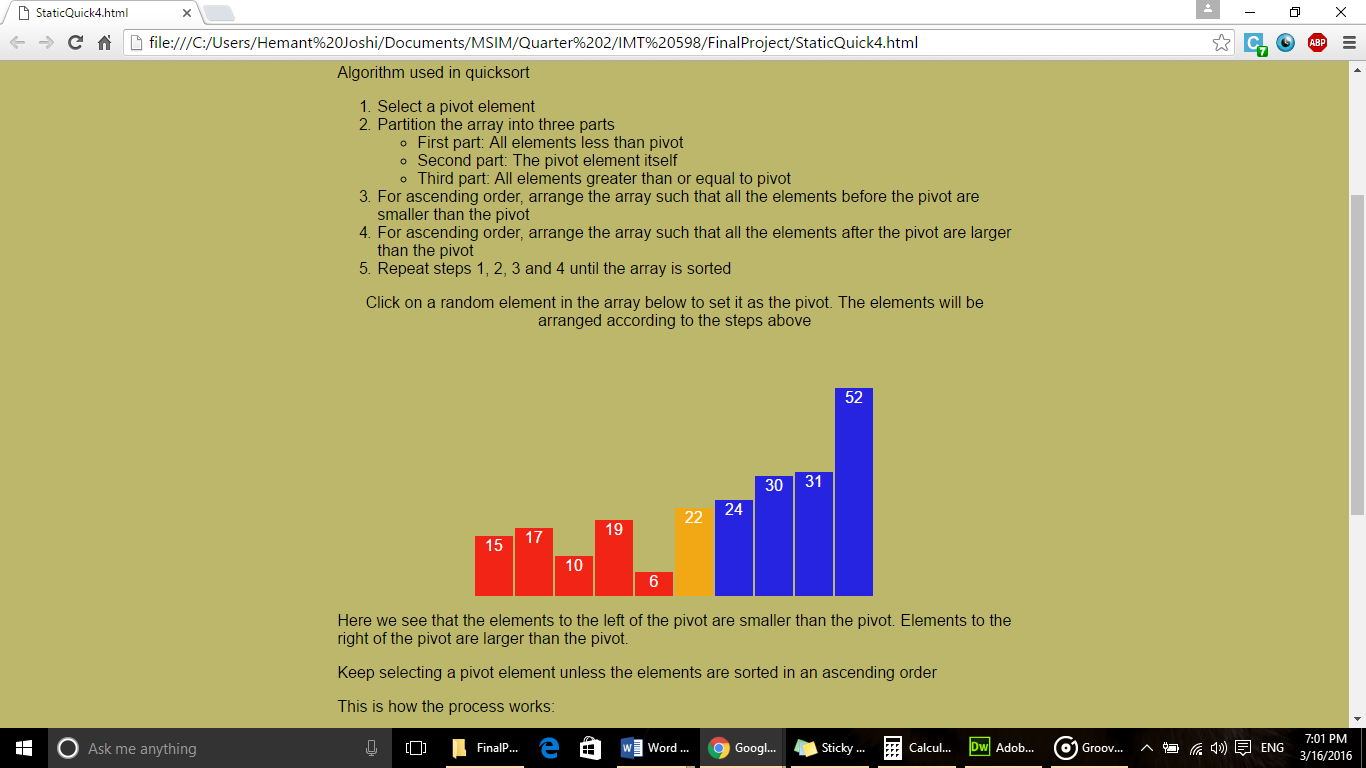


Figure 2

Here, the element 22 is selected as the pivot element. The user is able to verify all the steps that are stated above the visualization using the interactivity features of this visualization.

In the visualization below this, we see a slider that acts as a step-by-step guideline of how the process works. Using a slider works here because the user can know how many steps are required before he can iterate to the next step. Here is the first step of this visualization.

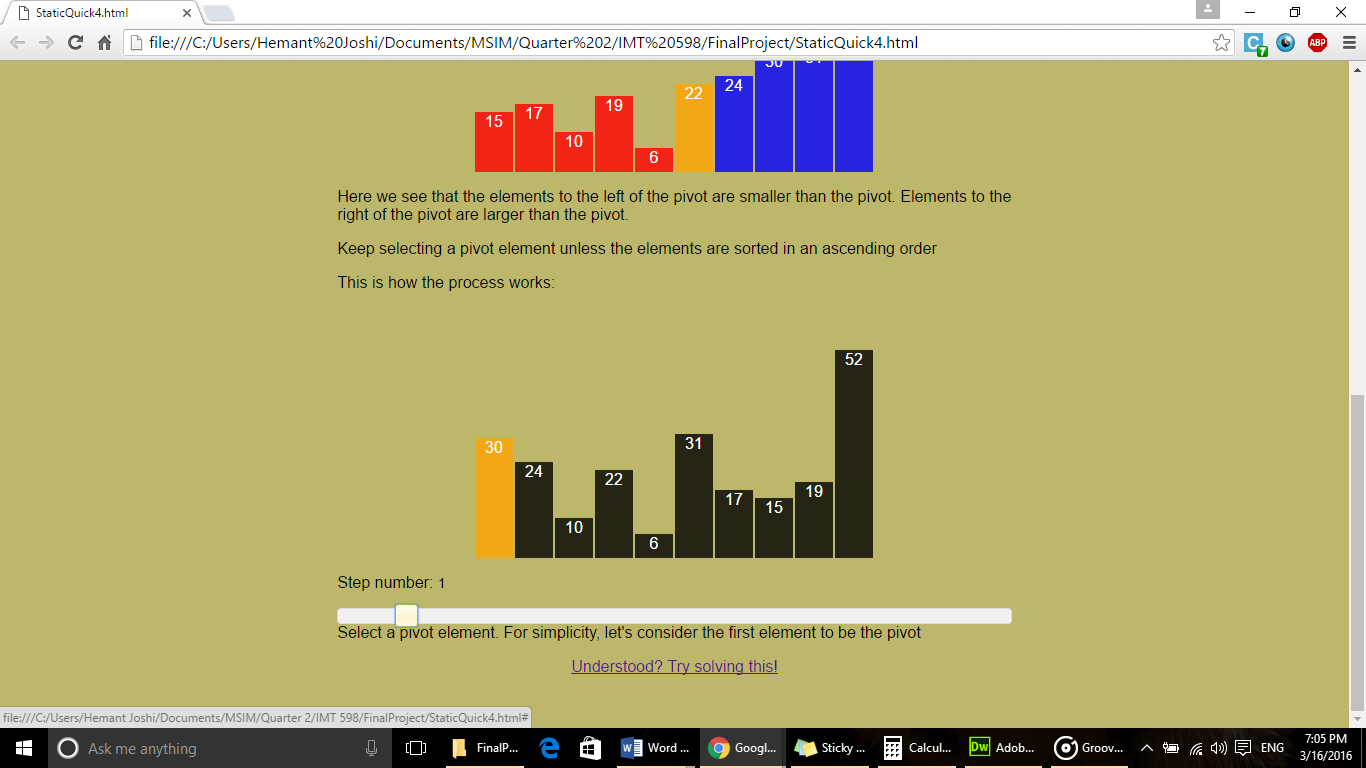


Figure 3

Because we are selecting the first element as the pivot element here, we are going to take 10 steps to be able to put 30 in its appropriate position. While interacting through this visualization, we expect the user to see the effect of selecting the first element on the complexity of the algorithm. While this is just a hint of complexity being affected by the selection of pivot elements, we do not expect the user to do anything right now.

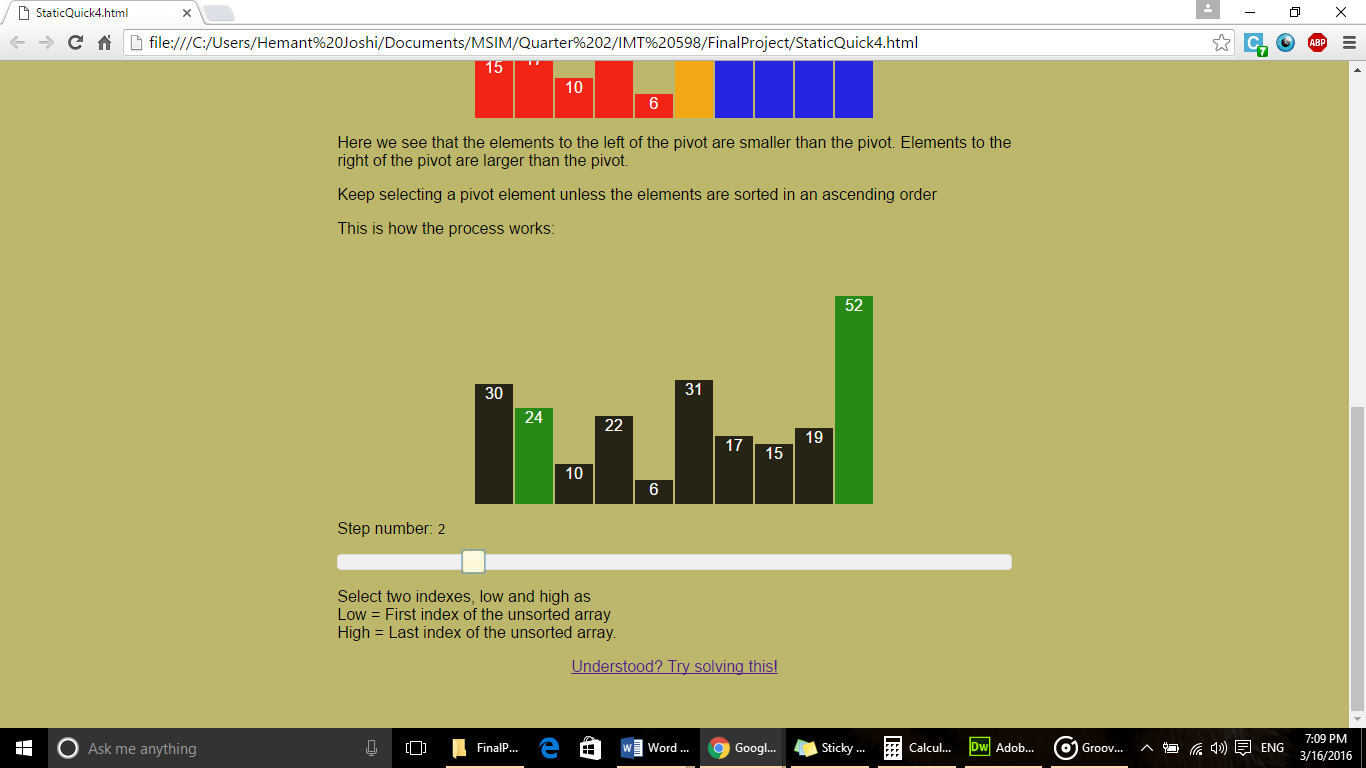
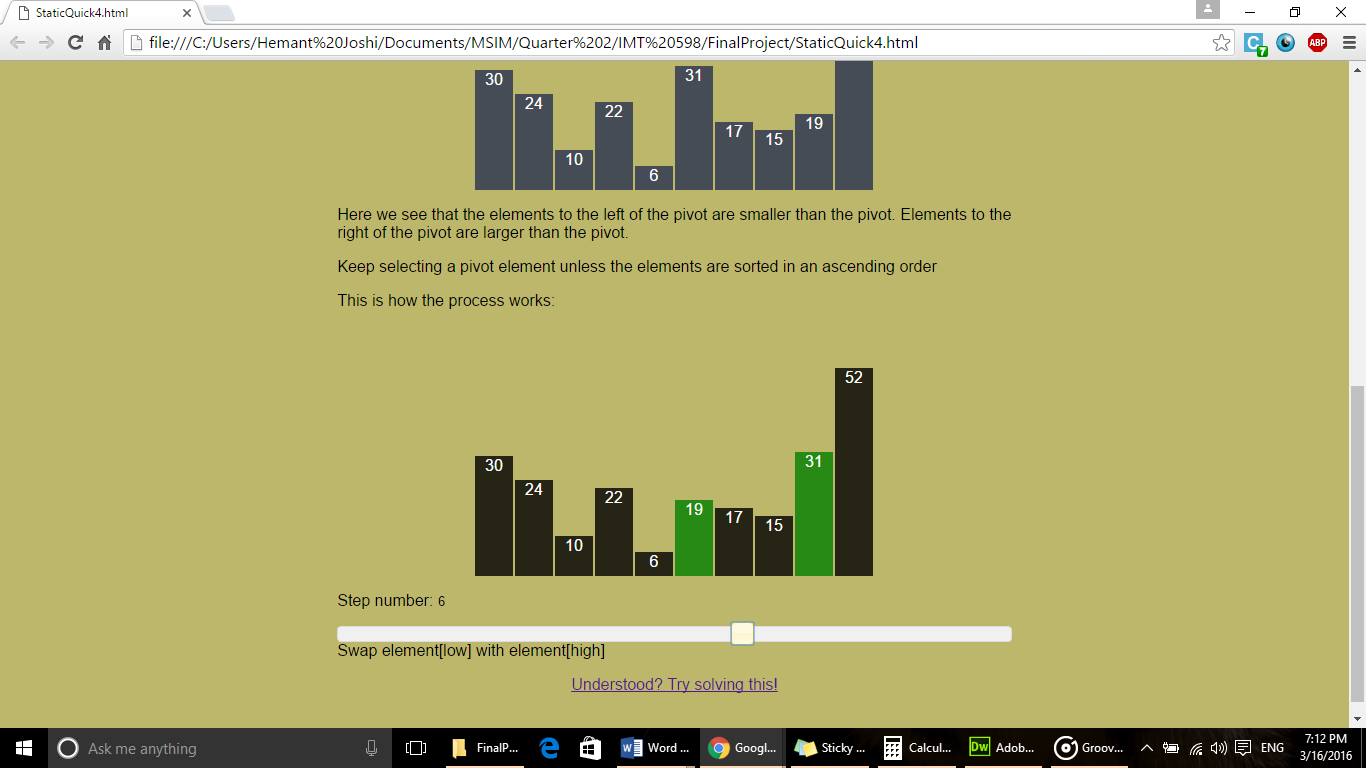


Figure 4

Moving to the next step on the slider, we see that the two elements are selected with indexes as shown in the explanation. In the subsequent steps, further iterations of the process are shown. This makes it easy for the user to understand how the process works. Here is another step in the process:



## Figure 5

## As we see, elements 19 and 31 are swapped. This forms one of the steps of the first iteration of the process.

## Through this tutorial, we have established how one iteration of the sorting algorithm works. After this, we consider the pivot element to be in its appropriate position. Now it is left to the user to implement the further iterations for the smaller arrays to the left and right of the pivot.

## 

Figure 6

As we can see, element 30 is placed in its appropriate position.

With the algorithm understood, we move on to the exploration part of the tutorial.

On clicking the link at the bottom of the page, we see a randomly generated array which needs to be sorted. The steps are listed once again here. So, we expect the user to see how the algorithm would work. While the complexity of the algorithm is stated, we do not know how many steps the user takes, as the user might choose to solve it without using our tutorial. Thus, we decided to provide hints rather than validation checks at each step.

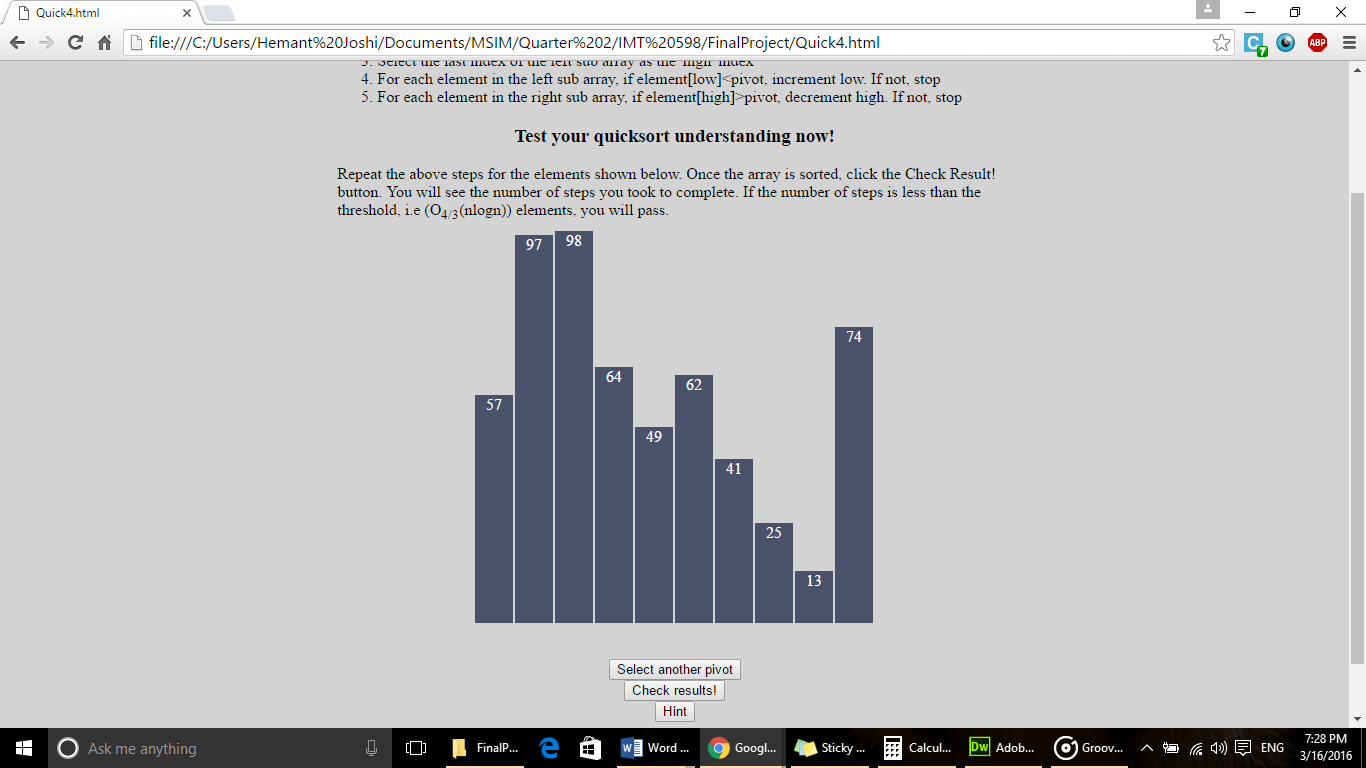


Figure 7

Here, we have a randomly generated array which has to be sorted. The first step in this process needs to be selecting a pivot, and then selecting two elements which need to be sorted. After the pivot is selected once again, an alert is generated, which indicates the user to select another pivot element.

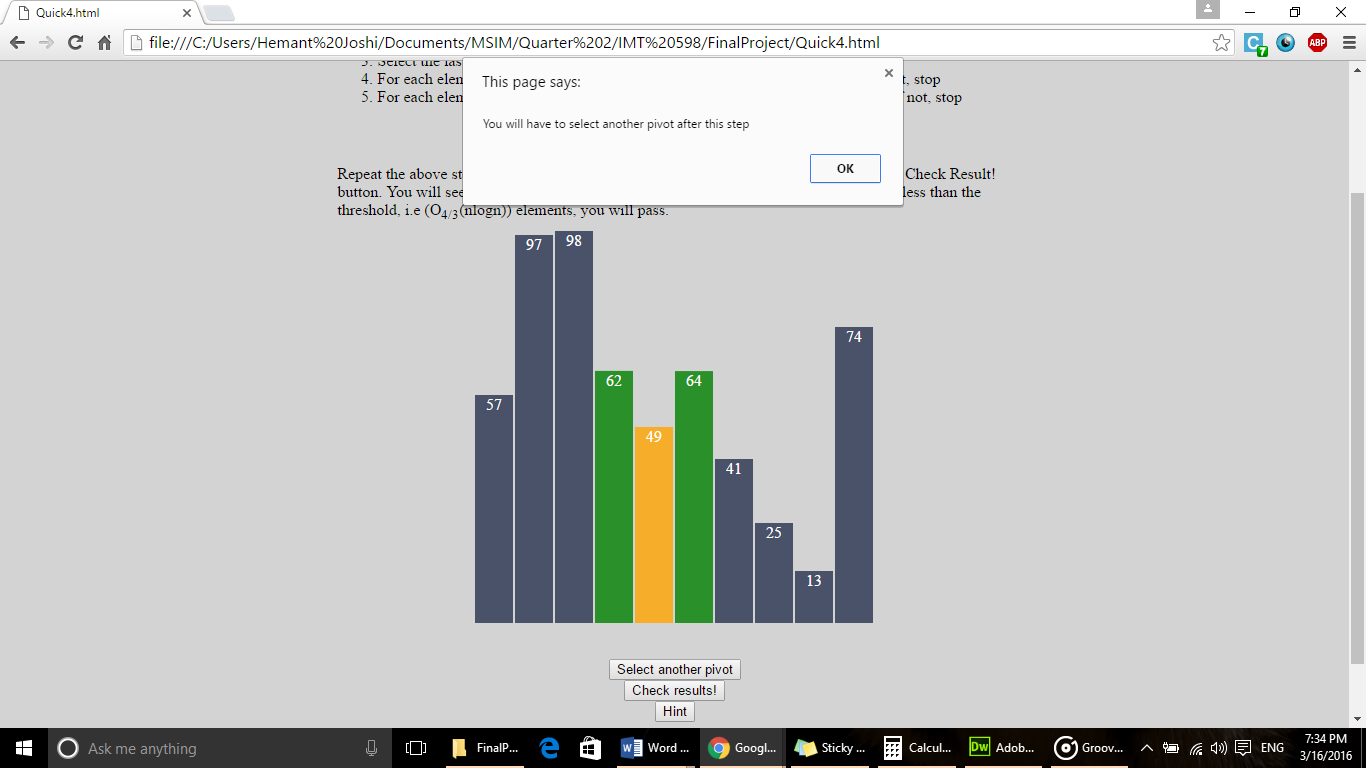


Figure 8

Once a user is done with swapping a pivot element, the last step in the process, he has to click the select another pivot button. After this step, a hint showing the condition that is required is shown. The user may choose to proceed with the implementation from here, but he might realize his mistake and select the same pivot to rectify the mistake.

But at this stage, to rectify his mistake, he would have to swap more number of times, thus increasing his count of swaps. As a result, once a user has selected a wrong element, the user has failed.

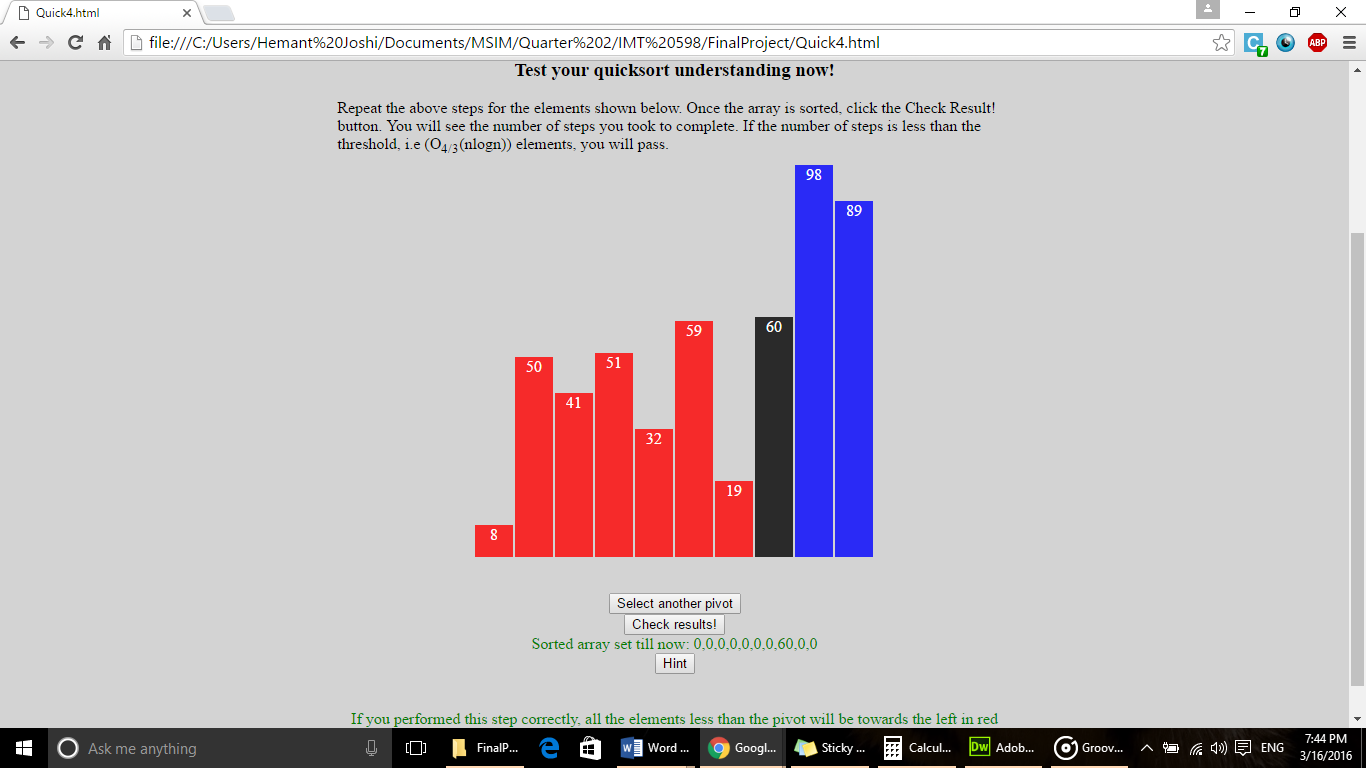


Figure 9

In the figure above, the sorted array contains only the element 60. As other pivots get sorted, they will be added to it. User can use this to not repeat the already selected pivot again.

The buttons ‘Hint’ and ‘Check Results’ provide a hint and the number of the result of the mini-test. We judge the result of the test as pass or fail based on the number of swaps made by the user. If the number of swaps are less than the size of the array (10) in this case, we say that the test is passed successfully. Otherwise, we tell the user to try again. The hint gives an idea of how to implement the algorithm by taking minimum steps. If the middle element is chosen as the pivot, we can be assured that minimum number of steps will be required to sort the array.

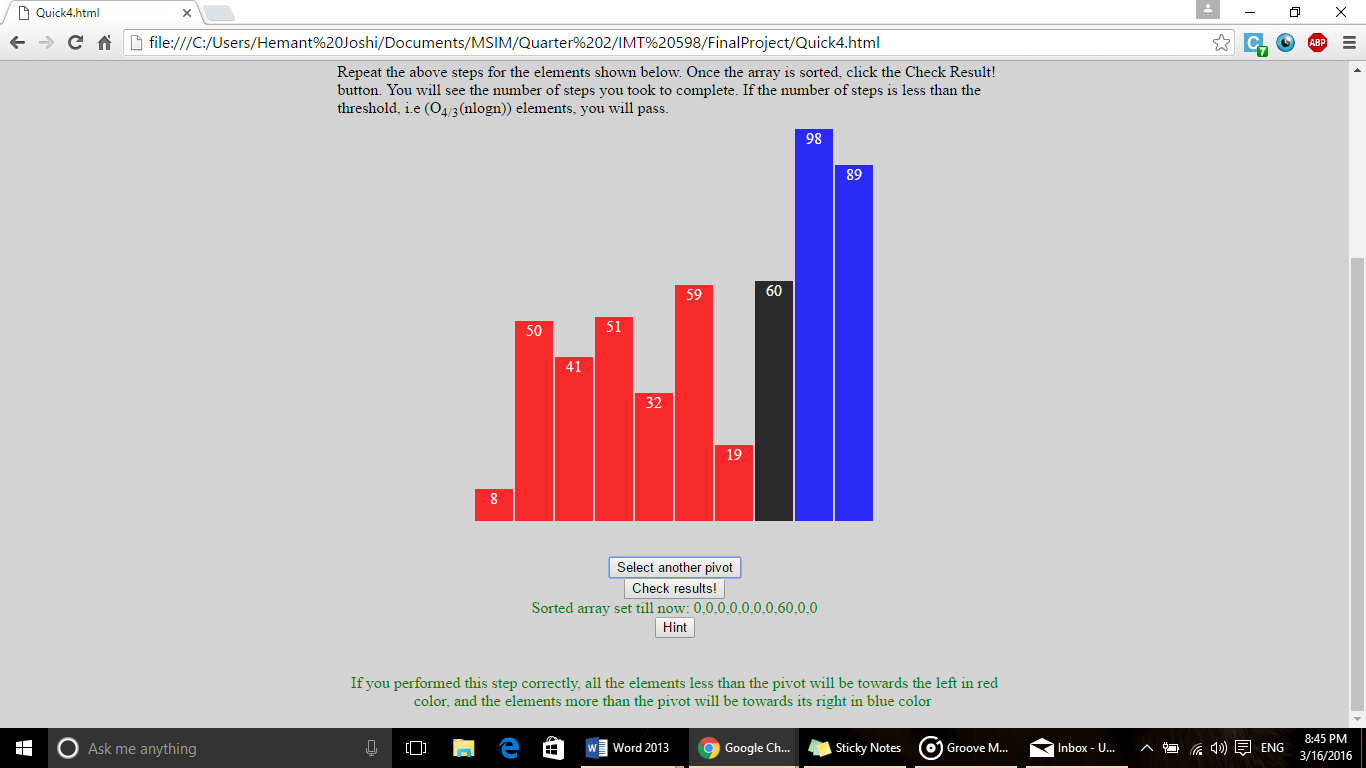


Figure 10

The figure above shows the hint. It clearly states that if the step was performed correctly, all the elements to the left of the pivot element will be red and elements to the right will be greater than the element.

# Future Work:

The future scope of this explorable explanation can be organized into the following parts.

Firstly, we can include stepwise evaluation of the users’ sorting actions. As the target audience for this explanation is Computer Science students, they would not just like to learn the working of the Quick sort but they would love to ace it. Currently, we do not provide feedback on every step because we want users to learn their mistakes on their own. To help them, we have given a hint so that they will select the appropriate pivot element. We can also evaluate the correctness of their sorting steps and then provide them some kind of hints so that they will come to know their mistakes. We are against comparing their steps with the correct steps and providing them the right answer because we want them to explore the various ways they can perform sorting and learn the impact on the complexity of the sorting algorithm.

Secondly, we can use this model for explaining other algorithms that are fundamental to computer science and information technology. This framework can be easily used for explanations for all the sorting algorithms, namely Selection, Insertion, Bubble, Bucket, Merge, Heap, Quick & Radix sort. Further, comparison between the complexities of these techniques can be shown visually.

Thirdly, we can build on this visualization technique to simplify explanations of complex concepts. For example, we can explain graphs and networks in computer science and algorithms such as depth-first-search and breadth-first-search using trees. Of course, these are just examples to an array of examples.

# REFERENCES

[1] Quicksort, Wikipedia. Retrieved March 14,2016 from <https://en.wikipedia.org/wiki/Quicksort>

[2] Algomation, retrieved March 14, 2016 from <http://www.algomation.com/algorithm/quick-sort-visualization>

[3] Understanding Quicksort (with interactive demo), retrieved March 14, 2016 from <http://me.dt.in.th/page/Quicksort/>

[4] Explorable explanations, retrieved March 14, 2016 from <http://explorableexplanations.com/>