ViSA: Visualization of Sorting Algorithms

Nithishkumar S R
Dept. of Computer Science
R N S Institute of Technology
Bengaluru, India
1rn22cs411.nithishkumarsr@rnsit.ac.in

Prajwal S R
Dept. of Computer Science
R N S Institute of Technology
Bengaluru, India
1rn22cs413.prajwalsr@rnsit.ac.in

Rashmi M
Asst.Professor
Dept. of Computer Science
R N S Institute of Technology
Bengaluru, India
rashmi.m@rnsit.ac.in

Kiran P
Professor
Dept. of Computer Science
R N S Institute of Technology
Bengaluru, India
Hod.cse@rnsit.ac.in

Abstract - Algorithm analysis and design may be challenging for students majoring in computer science and information science. A number of variables, such as a lack of motivation, a fear of programming, and the abstract nature of programming principles, all contribute to the high dropout and failure rates in basic programming courses. Numerous strategies have been put forth by researchers in an effort to motivate and support students.Despite claims that some of these tools help with the development of programming skills, the problem is still largely unresolved. This study describes ViSA, a tool for visualizing sorting algorithms. ViSA is an easy-to-set-up, fully automatic visualization system. It compares sorting methods and offers detailed directions. The visualization system's design ideas, technical framework, as well as its applications in the real world and educational advantages are shown and discussed.

I. INTRODUCTION

The core of both the computer and information science curricula is programming. They aim to assist students develop programming abilities by introducing them to fundamental programming ideas (such as control structures, aggregation techniques, sorting algorithms, etc.)Lack of prior knowledge, problem-solving techniques, a programming phobia, and the impression that programming language syntax is difficult are the main obstacles to learning programming. TThey also discovered that problem-solving approaches are much more difficult for programmers to master than programming language syntax. According to research on compilation behavior,[6] novice programmers most frequently use undeclared variables, define variables during iterations, use the incorrect operator in logical expressions, and other issues with the syntax of the C++ programming language.

The goal of beginning programmers is to translate the abstract process of algorithm execution into a form or language that the computer can comprehend. However, in order to "explain" something to the computer, a programmer must first fully comprehend it. In other words, learning and comprehending complex algorithms like iterations, recursions, or sorting algorithms is exceedingly challenging when doing so solely by observing code or a flowchart.The ability to control their own data sets and observe the entire algorithm execution process, on the other hand, allows students to analyze the output data and draw conclusions from it.By allowing students to create their own input data sets and engage with animation, you can also involve them in the algorithm visualization process. This will eventually result in their own predictions and logical deductions.[5].

The understanding and mastery of algorithmic ideas, particularly sorting algorithms, represent a fundamental cornerstone in the study of computer science and programming.Sorting algorithms serve as a launching pad for students to develop in their programming knowledge and problem-solving skills because they are crucial for structuring and successfully organizing data[7].However, the intricacy of sorting techniques and the abstract nature of algorithmic concepts sometimes cause students to struggle, which increases the dropout rate and creates learning challenges.The incorporation of visualization tools has emerged as a viable strategy to ease sorting algorithm comprehension and learning in response to these pedagogical difficulties. By converting complex abstract algorithms into understandable visual representations, visualization has the ability to close the gap between academic understanding and real-world application. By providing students with a visual means to monitor and interact with algorithmic processes, teachers may improve student engagement, decrease programming fear, and foster a deeper knowledge of the underlying concepts.

This project investigates the topic of sorting algorithm visualization with a focus on developing the state-of-the-art teaching tool known as ViSA (Visualization of Sorting Algorithms). With the help of ViSA, students will be able to generate their own input data sets and make precise predictions using visualization. ViSA aspires to be a user-friendly and interactive system that not only displays how various sorting algorithms operate but also provides step-by-step guidance[8].

II. LITERATURE SURVEY

The CS education community is already aware of a few AV development programs. Since the 1980s, collaborative visualization has been utilized in computer science teaching.

Teachers have used AV software to demonstrate the efficiency of an algorithm during a lecture, support students while they learn fundamental computer science methods, and more.

A mismatch between the way algorithms are taught and the learning preferences of the students is one of the main causes of the notion that basic programming courses are difficult. Even though the majority of people are visual learners, most instruction is verbal in nature.

People that prefer using visuals, photographs, colors, and maps to organize information in order to understand complicated topics are said to learn visually. Algorithm visualization (AV) technologies are used in the teaching and learning process primarily for this reason. Numerous

AV tools, including Alice, Software Visualization System, JCAT, Guido von Robot, Samba Algorithm Animation System, JHAVÉ, etc., are used in literature. The majority of them have a user-friendly control panel and support step-by-step operation. However, only a select few of them (like Samba) let the user change the animation's speed. Additionally, it is not very typical to include a feature that allows the user to go back a specific number of steps, like in DDD.

Tools for simulation and animation make up the two categories of AV. With the help of the animation tool Algorithma 99, users can enter algorithms in a preset pseudo-code language and view them afterwards. A knowledge of putting up the pseudocode and connecting it to the control elements that carry out the animation is necessary for this type of animation. Systems such as JAWAA and JSamba replicate the specified algorithm in a unique syntax that must be learnt before using the tools[8].

Simulator tools are easier to use and effective for teaching programming. A general-purpose simulation tool designed to teach any kind of programming ideas is one of them. Its name is Alice. However, the user must master its unique syntax before using it to teach particular algorithms. Furthermore, it prohibits any interaction with animation while programming. According to Hansen et al., using Alice results in a significant amount of time spent learning a new syntax rather than mastering the current techniques. For instance, a user would need to understand how to utilize the syntax, set up 3D models, and design the triggers that interact with the models in order to make a straightforward iteration. A tool like Alice would also require extensive setup on the host computer, such as the installation of 3D drivers, etc.

Sorting algorithms are a common component of modern computer software. On your PC, for instance, you might see files arranged in different ways if you launch file explorer. In sorted data, searching is more effective than in unsorted ones. Sorting algorithms are one of the many algorithms that computer science students begin learning in their first year of study.

III. VISUALIZATION OF SORTING ALGORITHMS (VISA)

The objective of ViSA is to provide a user-friendly learning interface that presents the user with as much feedback information as possible through explanation. One-click installation is sufficient to make the application usable, enable users to construct their own input data sets, and engage with the visualization, as was already said. This is a must for a powerful antivirus tool.

For ViSA installation, Microsoft's click-one-install method is employed, allowing the program to instantly download all necessary drivers and then run entirely on its own following installation.

Visualization is important since sorting algorithms can be challenging for those new to programming and computer science. The step-by-step execution of these algorithms can be demonstrated in a concrete fashion using visualization, which also enables students to see how data pieces are changed and reorganized in real time.

Improved Learning Environment: Visualization provides a fun and dynamic learning environment. Insights into how data items are compared, switched out, and moved during the sorting process can be gained by learners by seeing the algorithm in operation.

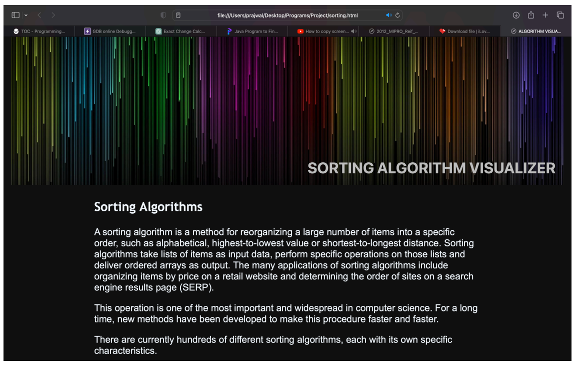
Strong conceptual comprehension of sorting algorithms can be developed with the help of visualization. The effects of algorithmic choices, such as pivot selection in Quick Sort or element comparisons in Bubble Sort, can be seen by learners.

Algorithmic Complexity: The time and space complexity of sorting algorithms varies. By displaying the amount of comparisons, swaps, and iterations made throughout the sorting process, visualization can aid learners in understanding these complexity.

Comparative Analysis: Students can frequently compare several sorting algorithms side by side using visualization tools. Understanding each algorithm's advantages and disadvantages as well as how it functions with various sorts of data is aided by this.

Interactivity: A variety of visualization tools let users interact with how an algorithm works. They have the ability to regulate the visualization's speed, pause at key moments, and proceed through the process one iteration at a time[9].

Accessible Learning: Different learning modalities are supported by visualization tools. Visual learners gain from watching the algorithm in action, and kinesthetic learners who value hands-on experiences can benefit from interactive components.



Sorting Algorithms

A sorting algorithm is a method for reorganizing a large number of items into a specific order, such as alphabetical, highest-to-lowest value or shortest-to-longest distance. Sorting algorithms take lists of items as input data, perform specific operations on those lists and deliver ordered arrays as output. The many applications of sorting algorithms include organizing items by price on a retail website and determining the order of sites on a search engine results page (SERP).

This operation is one of the most important and widespread in computer science. For a long time, new methods have been developed to make this procedure faster and faster.

There are currently hundreds of different sorting algorithms, each with its own specific characteristics.

Figure 1. ViSA Home Page

ViSA supports well-known sorting techniques as Bubble sort, Quick sort, Selection sort, Merge sort, Heap sort, and Insertion sort.

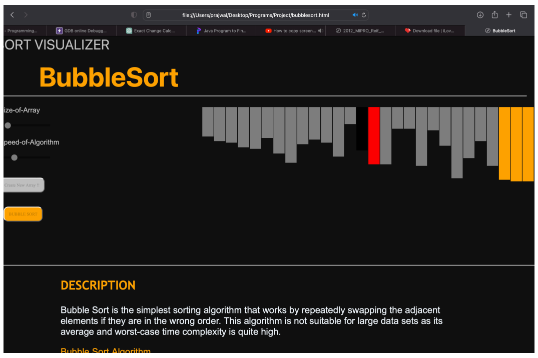
```
function Bubble()
{
c_delay=0;
for(var I=0;i<array_size-1;i++)
{
  for(var j=0;j<array_size-i-1;j++)
  {
 div_update(ele[j],ele_sizes[j],"indigo");
 ] if(ele_sizes[j]>ele_sizes[j+1])
   {
     div_update(ele[j],ele_sizes[j], "black");
      div_update(ele[j+1],ele_sizes[j+1], "black");
      var temp=ele_sizes[j];
      ele_sizes[j]=ele_sizes[j+1];
```

```
        ele_sizes[j+1]=temp;
  div_update(ele[j],ele_sizes[j], “black");
 div_update(ele[j+1],ele_sizes[j+1], “red");
  }
    div_update(ele[j],ele_sizes[j], “grey");
    }
     div_update(ele[j],ele_sizes[j], “orange");
  }
   div_update(ele[0],ele_sizes[0], “orange");
 enable_buttons();
 }
```

Code block 1. Bubble Sort[5]

Once the process is complete, the operations for the visualization are executed one at a time from the Listbox. This makes it simple to add more algorithm examples to the ViSA repository. If an algorithm is added by introducing the appropriate syntax, as seen in code block, any of the control components may be activated.

ViSA's usefulness is increased by its efficient addition of additional algorithm instances. ViSA can readily adapt to new algorithms as they are released in the industry, keeping the educational material current. This adaptability is especially crucial in the dynamic field of computer science and programming instruction, where new methodologies and algorithms are constantly reshaping the curriculum[10].



Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order. This algorithm is not suitable for large data sets as its average and worst-case time complexity is quite high.

Figure 2. ViSA Sort window

Several common controls can be found in the ViSA main animation window (see Figure 2):

(i) block types that represent an array's elements,

(ii) a slider for adjusting the animation speed,

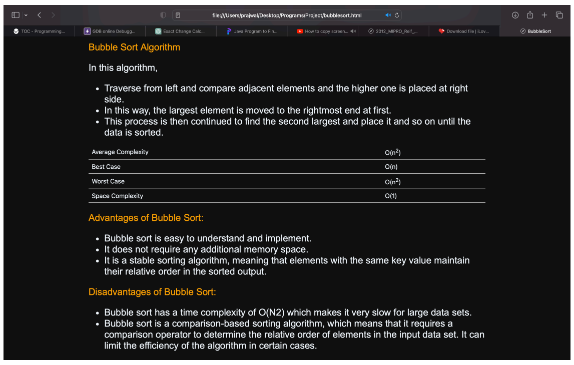
(iii) start and stop buttons,

(iv) information about the complexity of the algorithm, including the number of swaps and comparisons that are executed during the animation, and

(v) controls that have been added to make it simpler to comprehend the operations occurring inside the array[4].

Time Complexity:

The phrase "time complexity" refers to how long an algorithm takes to execute in comparison to the size of its input. It helps us analyze how the algorithm's runtime increases as the number of inputs increases. The standard way of representing it, Big O notation, provides an upper constraint on how quickly the algorithm's execution time will increase.



Bubble Sort Algorithm

In this algorithm,

- Traverse from left and compare adjacent elements and the higher one is placed at right side.
- In this way, the largest element is moved to the rightmost end at first.
- This process is then continued to find the second largest and place it and so on until the data is sorted.

Average Complexity	$O(n^2)$
Best Case	O(n)
Worst Case	$O(n^2)$
Space Complexity	O(1)

Advantages of Bubble Sort:

- Bubble sort is easy to understand and implement.
- It does not require any additional memory space.
- It is a stable sorting algorithm, meaning that elements with the same key value maintain their relative order in the sorted output.

Disadvantages of Bubble Sort:

- Bubble sort has a time complexity of O(N2) which makes it very slow for large data sets.
- Bubble sort is a comparison-based sorting algorithm, which means that it requires a comparison operator to determine the relative order of elements in the input data set. It can limit the efficiency of the algorithm in certain cases.

Figure 3. Time and Space Complexity

Common time complexity classes:

- **Constant-O(1)**:The size of the input has no bearing on how quickly the algorithm runs. Regardless of the input size, it runs in a set length of time.

- **Logarithmic Time-O(log n)**: Algorithms with logarithmic complexity divide the input in each step. Binary search is a classic example.

- **Linear Time-O(n)**:With increasing input size, the execution time increases linearly. This category includes operations that are straightforward on each input element.

- **Linearithmic Time-O(n log n)**:Frequently encountered in effective sorting algorithms like Merge Sort and Quick Sort.

- **Polynomial Time-O(n^2),O(n^3),…**: Algorithms with nested loops or recursive calls that lead to multiple iterations over the input.

- **Exponential Time-O(2^n),O(3^n),…**: Algorithms with a high degree of branching or recursion. They become very slow for even moderately sized inputs[11].

Space Complexity:

How much memory an algorithm uses in relation to the amount of its input is known as the "space complexity" of the algorithm. It helps us analyze how the algorithm's memory usage increases as the number of inputs increases. Similar to time complexity, it is often expressed using big O notation.

Both auxiliary space (memory used by the algorithm) and input space (memory used by the input itself) are considered to be components of space complexity[11].

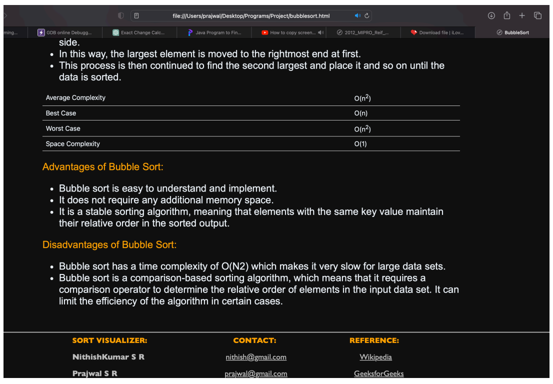
Common space complexity classes:

• **Constant Space-O(1)**: Regardless of the size of the input, the method consumes a set amount of RAM.

• **Linear Space-O(n)**:Memory use increases linearly as input size increases.

• **Polynomial Space-O(n^2),O(n^3),...**:: Algorithms that create data structures like matrices or 2D arrays that have a size based on a polynomial function of the input size.

• **Exponential Space-O(2^n),O(3^n),...**: Algorithms that create a large number of recursive calls, resulting in a huge memory usage



side.
- In this way, the largest element is moved to the rightmost end at first.
- This process is then continued to find the second largest and place it and so on until the data is sorted.

Average Complexity	$O(n^2)$
Best Case	O(n)
Worst Case	$O(n^2)$
Space Complexity	O(1)

Advantages of Bubble Sort:

- Bubble sort is easy to understand and implement.
- It does not require any additional memory space.
- It is a stable sorting algorithm, meaning that elements with the same key value maintain their relative order in the sorted output.

Disadvantages of Bubble Sort:

- Bubble sort has a time complexity of O(N2) which makes it very slow for large data sets.
- Bubble sort is a comparison-based sorting algorithm, which means that it requires a comparison operator to determine the relative order of elements in the input data set. It can limit the efficiency of the algorithm in certain cases.

SORT VISUALIZER:	CONTACT:	REFERENCE:
NithishKumar S R	nithish@gmail.com	Wikipedia
Prajwal S R	prajwal@gmail.com	GeeksforGeeks

Figure 4. Simultaneous animation

Black and red are used as flag colors to indicate an array element's status. While Orange indicates the user that the element has been sorted and is placed at its final position within the array, Black and Red alert the user that two elements are being compared or exchanged.

One of ViSA's unique features, simultaneous visualization of particular sorting methods, is seen in Figure 4. This tool allows the user to contrast the speed and complexity of the selected sorting algorithms. For a particular algorithm, the user can choose a fresh set of data, or they can choose any algorithm from the list to sort the same array.

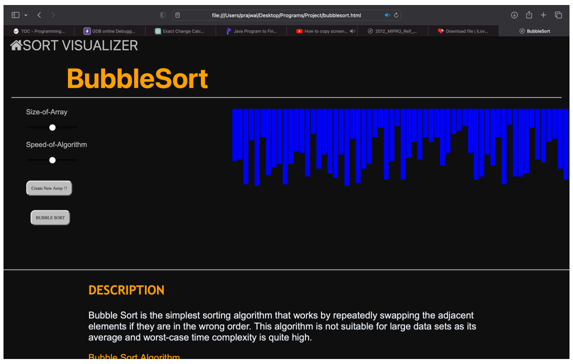
ViSA, your sorting algorithm visualization tool, enhances the learning process with depth and adaptability through the use of color coding and the simultaneous viewing function. Better user comprehension is facilitated by the thoughtful selection and application of colors, and the opportunity to compare sorting algorithms side by side provides novel insights. A more extensive discussion of the issues you brought up is provided below:

Color Coding for Element Status: ViSA successfully communicates the status of array elements during sorting by using the color combinations of black, red, and orange. The user is visually informed that two things are being compared or exchanged by the colors red and black. This real-time feedback helps with comprehension of the algorithm's development by emphasizing key steps.

Orange for Final location: Using orange to indicate that an element has been sorted and put in its final location within the array makes it easier to understand the results of the algorithm. Sorting is reinforced because learners can easily tell apart between elements that are still in motion and those that have already arrived at their destinations[4].

Simultaneous visualizing: One distinguishing characteristic of ViSA is its simultaneous visualizing capability. Users' ability to choose an algorithm with confidence is improved when they can compare the efficiency, complexity, and speed of several sorting algorithms in real time. This functionality promotes a greater understanding of algorithm efficiency in addition to helping with algorithmic intricacies.

Algorithm Complexity and Speed Comparison:
By using the simultaneous display option, users can observe firsthand how several sorting algorithms handle the same data set. This feature provides a direct way to observe and contrast the various characteristics of computationally difficult algorithms such as bubble Sort, merge Sort, quick Sort, and others.



DESCRIPTION

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order. This algorithm is not suitable for large data sets as its average and worst-case time complexity is quite high.

Bubble Sort Algorithm

Figure 5. Advantages and Disadvantages

Data Set and Algorithm Flexibility: ViSA's capacity to accept new data sets and algorithm examples enhances its adaptability and utility. Users can choose from a variety of techniques to sort the same data or choose different data sets. This function promotes a flexible learning environment and promotes investigation of various algorithms.

Data organization and analysis in a more effective and relevant way is one of the key benefits of sorting algorithms. For instance, categorizing a sizable dataset can make it simpler to look for particular components or spot patterns and trends. Other algorithms, like search algorithms and graph algorithms, can benefit from the employment of sorting algorithms to improve their performance.

Advantages of Sorting:

• **Efficient Data Retrieval:**Sorted data enables more efficient and quick search operations. One kind of method that can be used to reduce search time to logarithmic complexity is binary search.

• **Improved Data Presentation:**Sorted data is arranged visually, which facilitates user comprehension, analysis, and interpretation. This is especially helpful for giving users information.

• **Enhanced Data Analysis:**Sorting the data makes it easier to spot patterns and trends. It makes it easier to discover the minimum, maximum, median, and quartiles, among other data analysis tasks.

- **Optimized Algorithms:** On sorted data, several algorithms operate more effectively. For instance, sorted input helps merge sort and binary search, which reduces total time complexity.

- **Facilitates Database Operations:**Databases commonly employ sorted data for indexing. As a result, database operations are more productive and querying is hastened.

- **Efficient Output Generation:** Sorted data is essential for producing structured and organized outputs like reports, invoices, and summaries.

- **Optimized Memory Usage:** In-place sorting is one sorting method that minimizes the need for additional memory, making it effective for contexts with limited resources[3].

IV. SHORTING ALGORITHMS

- **Bubble Sort:**

Bubble Sort compares and swaps two adjacent components that are out of order. The largest unsorted element always "bubbles up" to the end of the array, where it belongs[1].

Visualization:

Consider a set of values. When comparing consecutive numbers from left to right, the algorithm swaps them if they are out of order. Every time this process is repeated, the greatest value "bubbles up" to the end of array.

- **Selection Sort:**

Selection Sort repeatedly selects the smallest element from the array's unsorted segment and replaces it with the array's initial unsorted element.

Visualization:

Searching through the list for the lowest value, the algorithm replaces it with the top value. Once the list is sorted, it moves on to the remaining unsorted section.

- **Insertion Sort:**

By continuously removing elements from the unsorted portion and putting them in the appropriate position in the sorted portion, insertion sort creates the sorted array one item at a time.

Visualization:

Visualize the portion that was sorted growing to the right and left. The procedure inserts the inserted components into the sorted section of the data after removing other items to make room for them from the unsorted component of the data[2].

- **Merge Sort:**

Merge Sort divides an array in half, evenly divides each half, sorts each half independently, and then rejoins the sorted halves.

Visualization:

Repeatedly divide the list in half to create short sublists. Then, merge these sublists in the desired order, merging them to create progressively larger sublists that are sorted, and so on, until the entire list is sorted.

- **Quick Sort:**

Quick Sort selects a "pivot" element and splits the array into two subarrays: those below the pivot and those above the pivot. The subarrays are then sorted recursively after that.

Visualization:

Make a decision on a pivot and divide the list into two sublists: values below the pivot and values above the pivot. This procedure is repeated iteratively for the sublists[1].

V. CONLUSION

Our research's goal was to create a teaching tool that would actively include students in their education while assisting them in learning about well-known sorting algorithms. A full range of functions, including data set entry, animation control, explanation, and in-depth algorithm examination, are available with the visualization of sorting algorithms (ViSA).

ViSA should dramatically shorten the learning curve and learning and comprehending time. By removing the necessity to write the programming code, ViSA builds a bridge between students and algorithms, lowering their phobia of programming.If students were familiar with the algorithm, they would approach programming with a lot more confidence. According to the results of our pilot study, ViSA is regarded as a valuable educational tool whose simplicity and value result in a positive attitude and behavioral intention toward its use.

REFERENCES:

[1] GeeksforGeeks - Sorting Algorithms

- The GeeksforGeeks website offers detailed explanations and visualizations of various sorting algorithms, making it a helpful resource for learning and understanding sorting techniques.

[2] visualgo.net

- This website provides visualizations of various sorting algorithms, making it easier to understand how they work step by step.

[3] Sorting algorithm -Wikipedia

• Wikipedia website offers detailed explanations of various sorting algorithms, making it a helpful resource for learning and understanding sorting techniques.

[4] "Algorithm Visualizations: A Survey”

• Authors: Iman Keivanloo, Joseph Bergin

• This paper offers an overview of algorithm visualizations and all of their elements, including user interface design.

[5] YouTube - "Sorting Out Sorting" by Ronald Baecker

• This video series provides explanations and visualizations of different sorting algorithms.

• Video Series: https://www.youtube.com/watch?v=8Kp-8OGwphY

[6] “Learning to Program—Challenges and Solutions”

• Authors: Raymond Lister, Elizabeth S. Adams, Sue Fitzgerald, etc.

• This paper discusses the challenges faced by beginner programmers, including the abstract nature of programming principles and difficulties with syntax.

[7] “Teaching Algorithms in an Undergraduate Course Using Algorithm Visualization”

• Authors: Prabir Bhattacharya, Yatish Agarwal

• This paper might discuss the use of algorithm visualization tools in teaching algorithms

[8] “Algorithm Animation Systems: A Survey”

• Author: Peter B. Henderson

• This survey paper discusses various algorithm animation systems, including their features and functionalities.

[9]“Teaching Sorting Algorithms with the Aid of Interactive Visualization”

• Authors: Yue Zhao, Joseph Maguire

• This paper discuss the use of interactive visualization for teaching sorting algorithms.

[10] “Visual Learning of Algorithms: A Study of Students’ Visualizations of Sorting Algorithms”

• Authors: Päivi Kinnunen, Ari Korhonen

• This paper discuss the visual learning of algorithms and how visualization tools can be adapted to different sorting algorithms for educational purposes.

[11]“Big-O Notation and Algorithm Analysis" (Tutorial)

• Author: Vaidehi Joshi

• This online tutorial provides a clear explanation of Big-O notation and algorithm analysis, focusing on time complexity.