NATIONAL UNIVERSITY OF COMPUTING AND EMERGING SCIENCES

DESIGN AND ANALYSIS OF ALGORITHM

SORTING VISUALIZATIONS

Submitted by:
Alishba Subhani (20K-0351)
Mannahil Miftah (20K-0234)

SECTION: BAI-5A

Submitted to:
Aqsa Zahid

ABSTRACT

The project is aimed towards visualization of sorting algorithms to incorporate better understanding of algorithms by showing the intermediate sorting steps.

INTRODUCTION

Since the passage of time, there has been a constant development in the sorting strategies used. These developments devised some new algorithms, and some modified the previous ones. Since sorting plays a pivotal role in the Computer Science domain, students are recommended to have sorting strategies at their fingertips. Therefore, time complexity of every algorithm is also shown to deduce why computer scientists felt the need for better algorithms, and how some algorithms produce the same desired results efficiently in much less time than rigorous solutions also known as brute-force algorithms.

The algorithms implemented are

- Bubble sort
- Insertion sort
- Count sort
- Radix sort
- Heap sort
- Merge sort
- Bucket sort
- Quick sort
- Modified quick sort
- Modified count sort

PROGRAMMING DESIGN

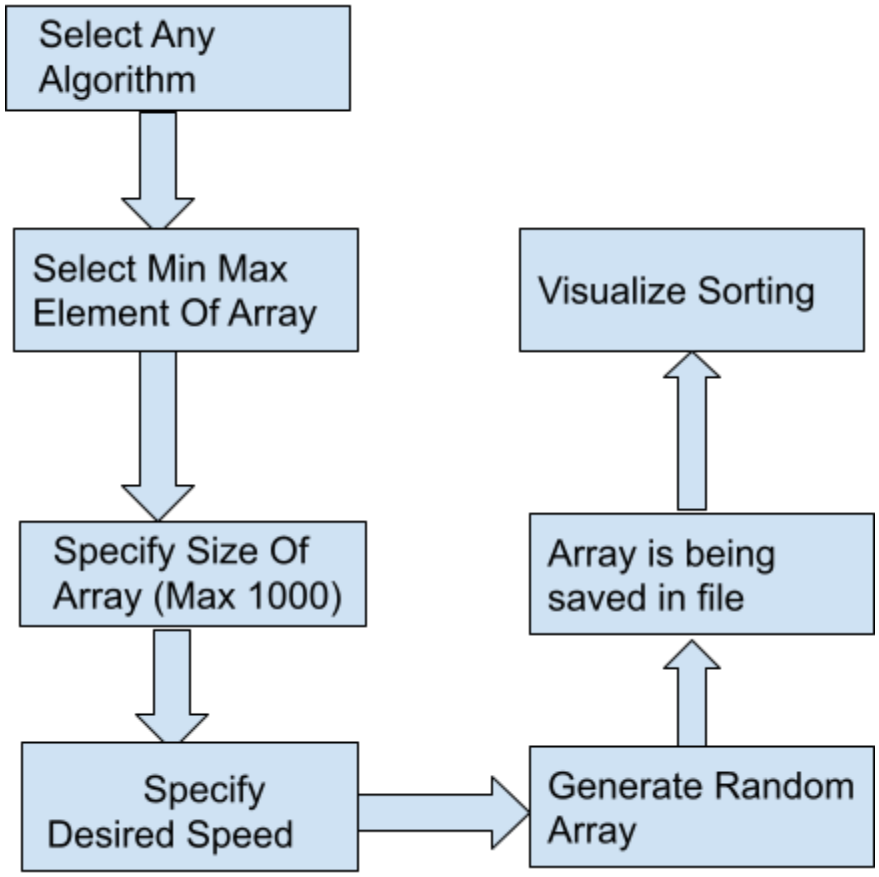
Language: Python

Library: Tkinter

EXPERIMENTAL SETUP

The design and structure of the user interface components has remained unchanged even if the underlying back-end code was refactored midway through the construction. Following are the buttons used to navigate through the sorting visualizer window

- Algorithm selection button to select the desired algorithm.
- Speed button to control the running speed of an algorithm. It materializes sorting at slow speed to depict better insights about algorithms.
- Generate array button to randomly generate the array of numbers.
- Minimum Size button to specify the smallest number being generated by the generate array button.
- Maximum Size button to specify the largest number being generated by the generate array button.

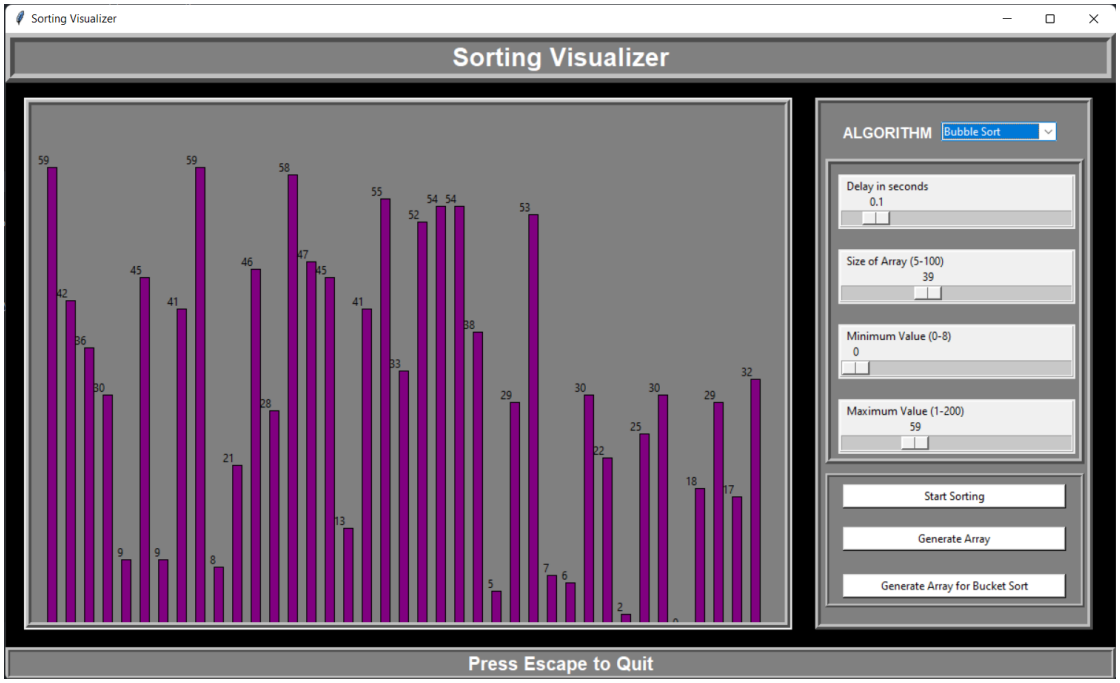


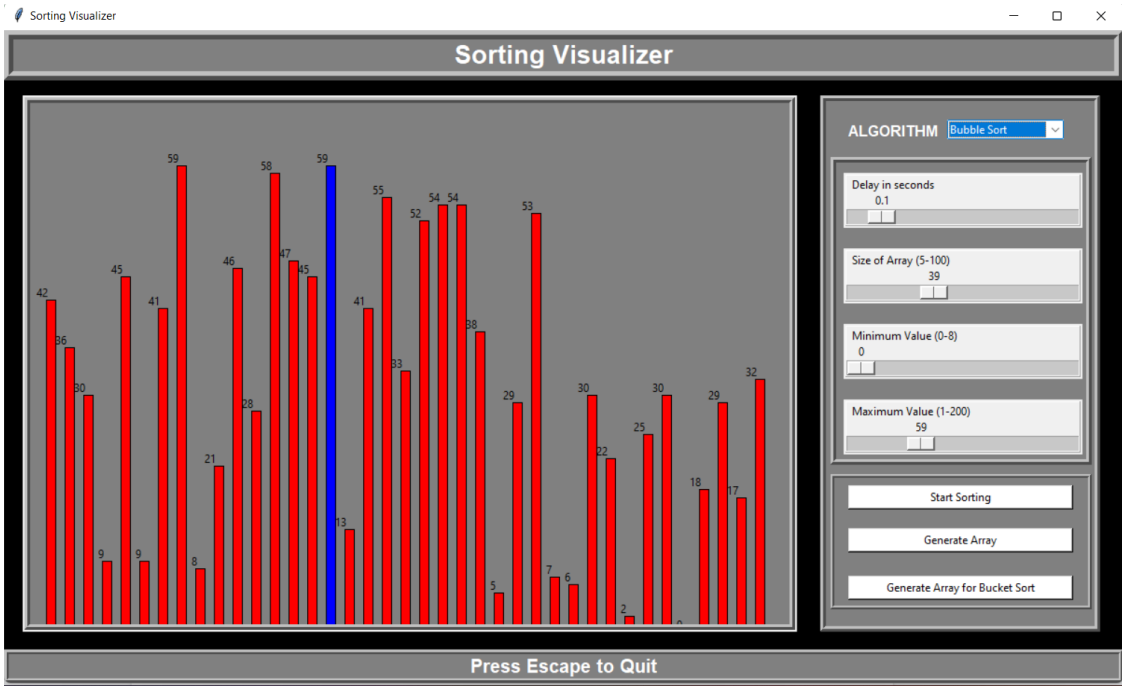
EXPLANATION OF SETUP

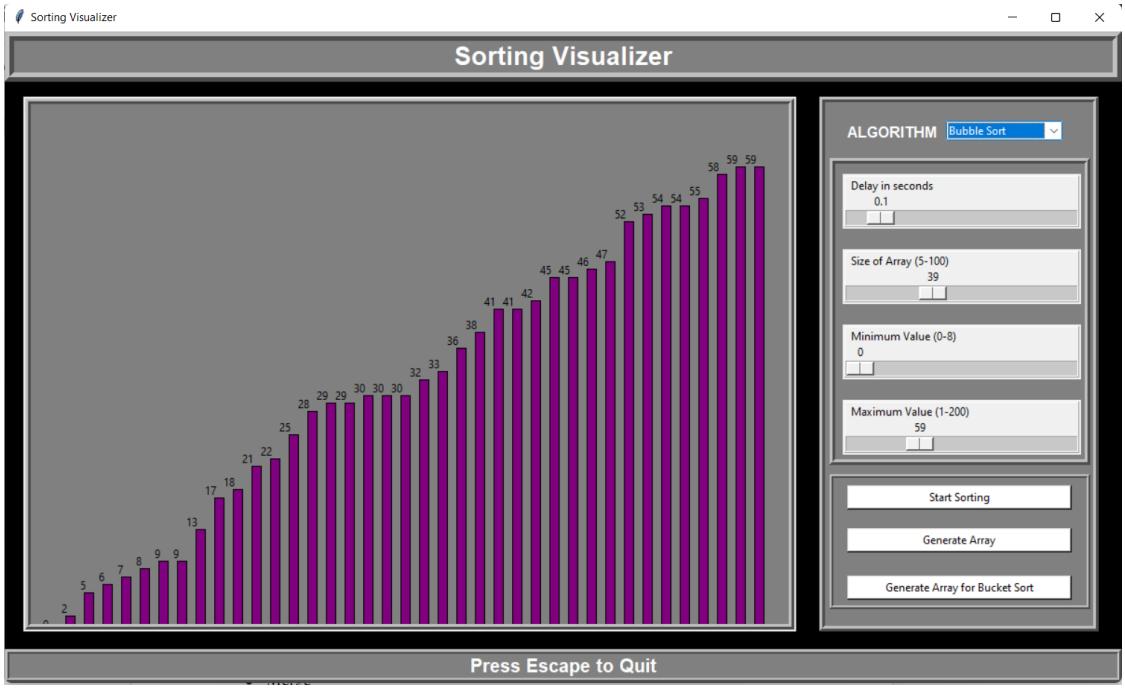
1. Our GUI is implemented via python library TKINTER which is rendering the visualizer window.
2. The window is composed of several frames mainly consisting of button frame, sliding frame and main frame within which these two frames are incorporataed.
3. When generate array button is hit, random array of specified size is generated through Generate_array() function, and the array is written into the file.
4. Elements are rendered on visualizer window with the help of draw_bars() function.
5. When the array is sorted, color of bars are changed.
6. During sorted, the elements being sorted are visualized through different colors.

RESULTS

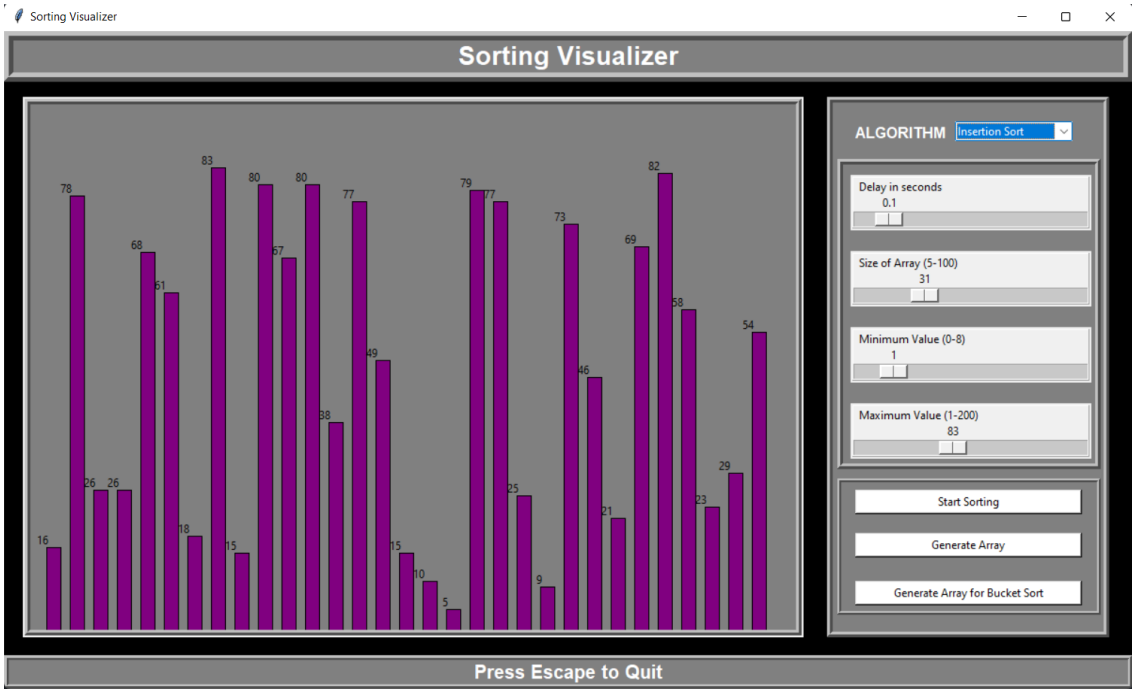
- Bubble Sort



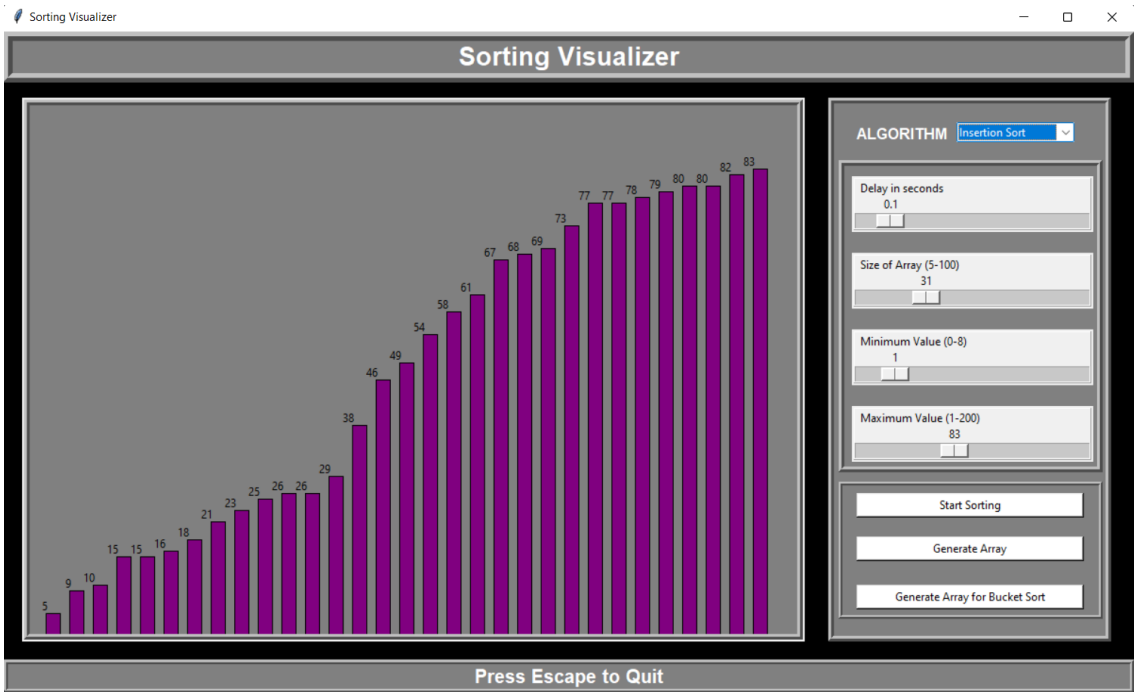

Sorting Visualizer
Sorting Visualizer
ALGORITHM
Bubble Sort
Delay in seconds
0.1
Size of Array (5-100)
39
Minimum Value (0-8)
0
Maximum Value (1-200)
59
Start Sorting
Generate Array
Generate Array for Bucket Sort
Press Escape to Quit


Sorting Visualizer
Sorting Visualizer
ALGORITHM
Bubble Sort
Delay in seconds
0.1
Size of Array (5-100)
39
Minimum Value (0-8)
0
Maximum Value (1-200)
59
Start Sorting
Generate Array
Generate Array for Bucket Sort
Press Escape to Quit

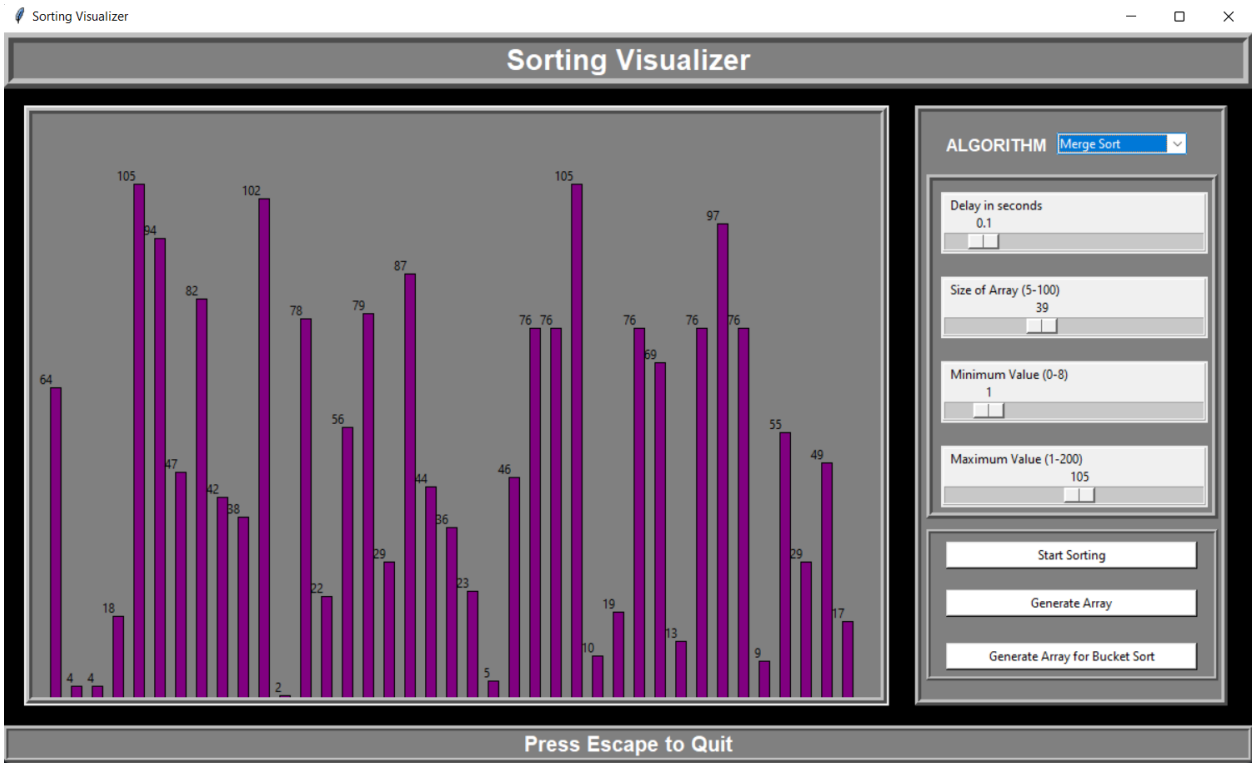
- Insertion

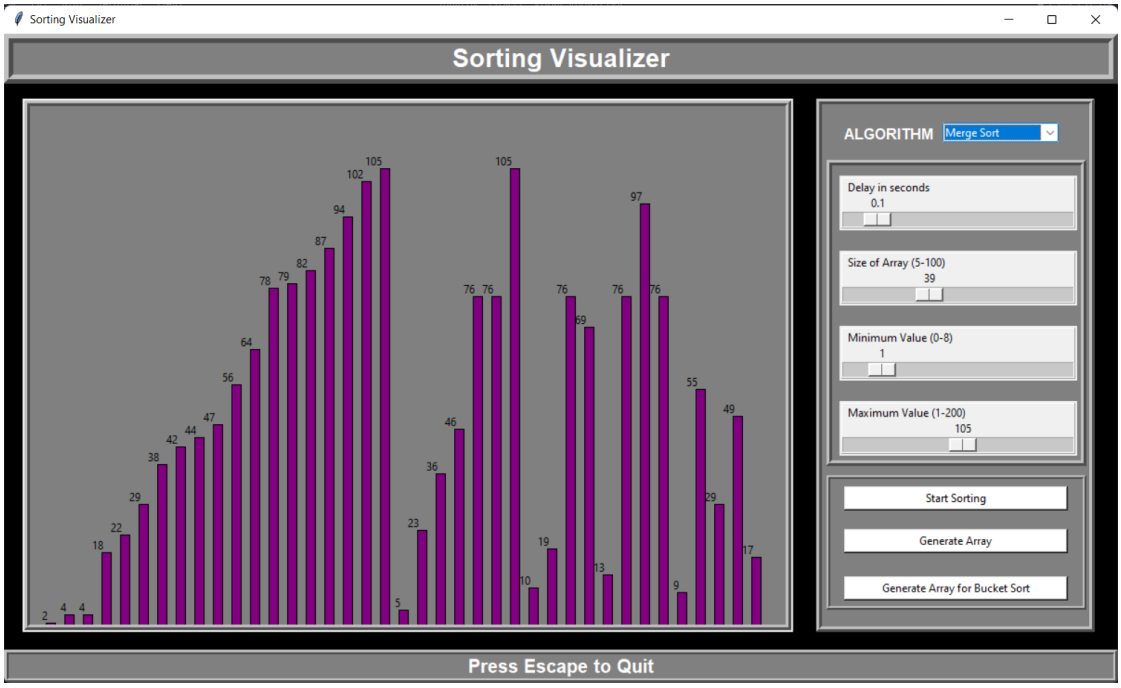

Sorting Visualizer
Sorting Visualizer
ALGORITHM
Insertion Sort
Delay in seconds
0.1
Size of Array (5-100)
31
Minimum Value (0-8)
1
Maximum Value (1-200)
83
Start Sorting
Generate Array
Generate Array for Bucket Sort
Press Escape to Quit

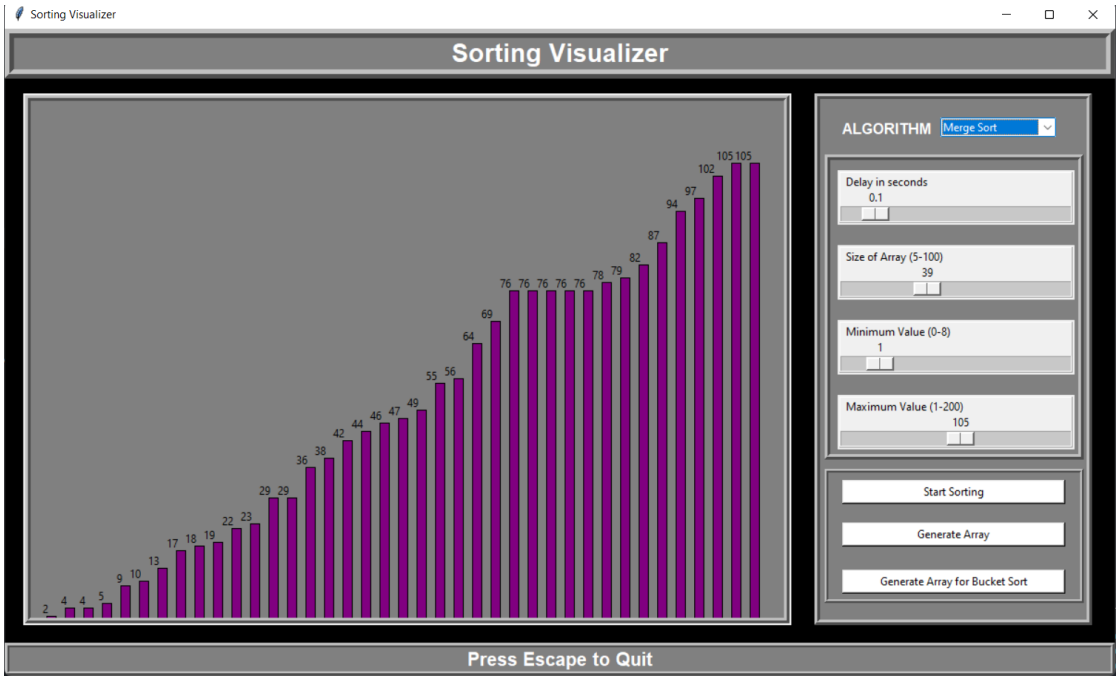

Sorting Visualizer
Sorting Visualizer
ALGORITHM
Insertion Sort
Delay in seconds
0.1
Size of Array (5-100)
31
Minimum Value (0-8)
1
Maximum Value (1-200)
83
Start Sorting
Generate Array
Generate Array for Bucket Sort
Press Escape to Quit


Sorting Visualizer
Sorting Visualizer
ALGORITHM
Insertion Sort
Delay in seconds
0.1
Size of Array (5-100)
31
Minimum Value (0-8)
1
Maximum Value (1-200)
83
Start Sorting
Generate Array
Generate Array for Bucket Sort
Press Escape to Quit

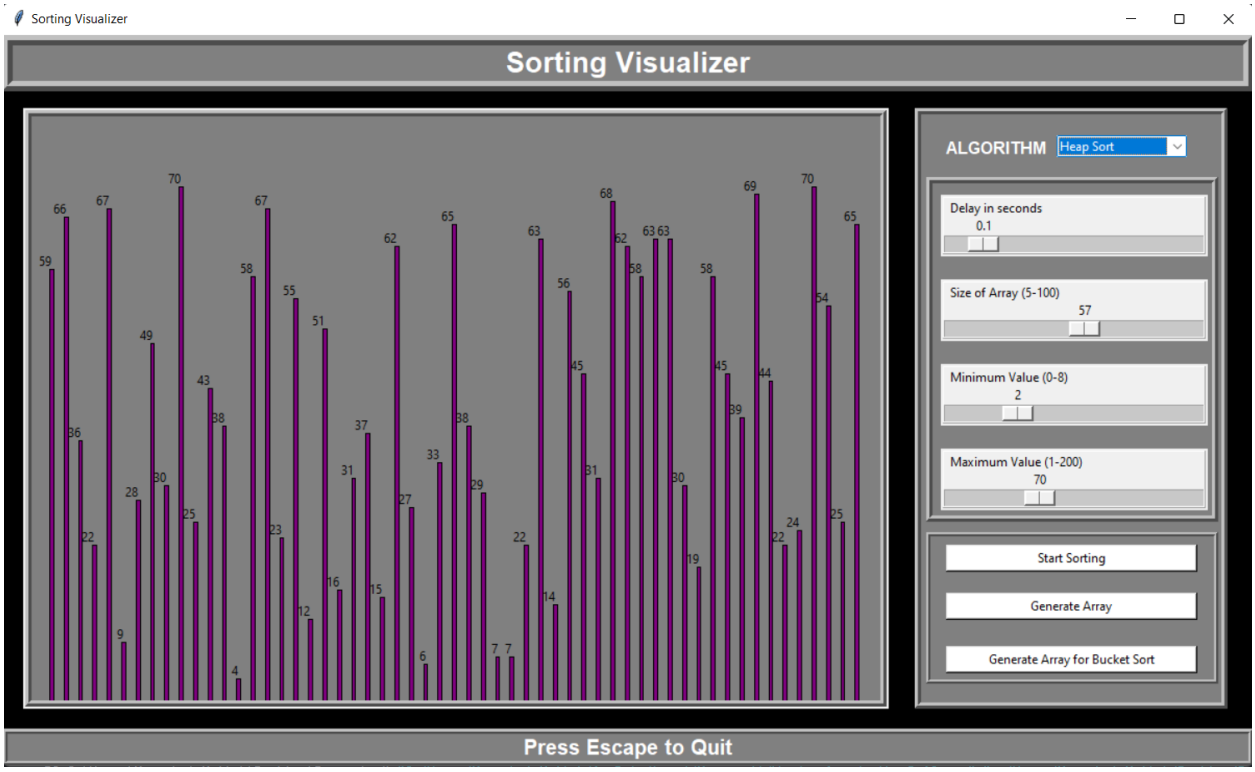
- Merge

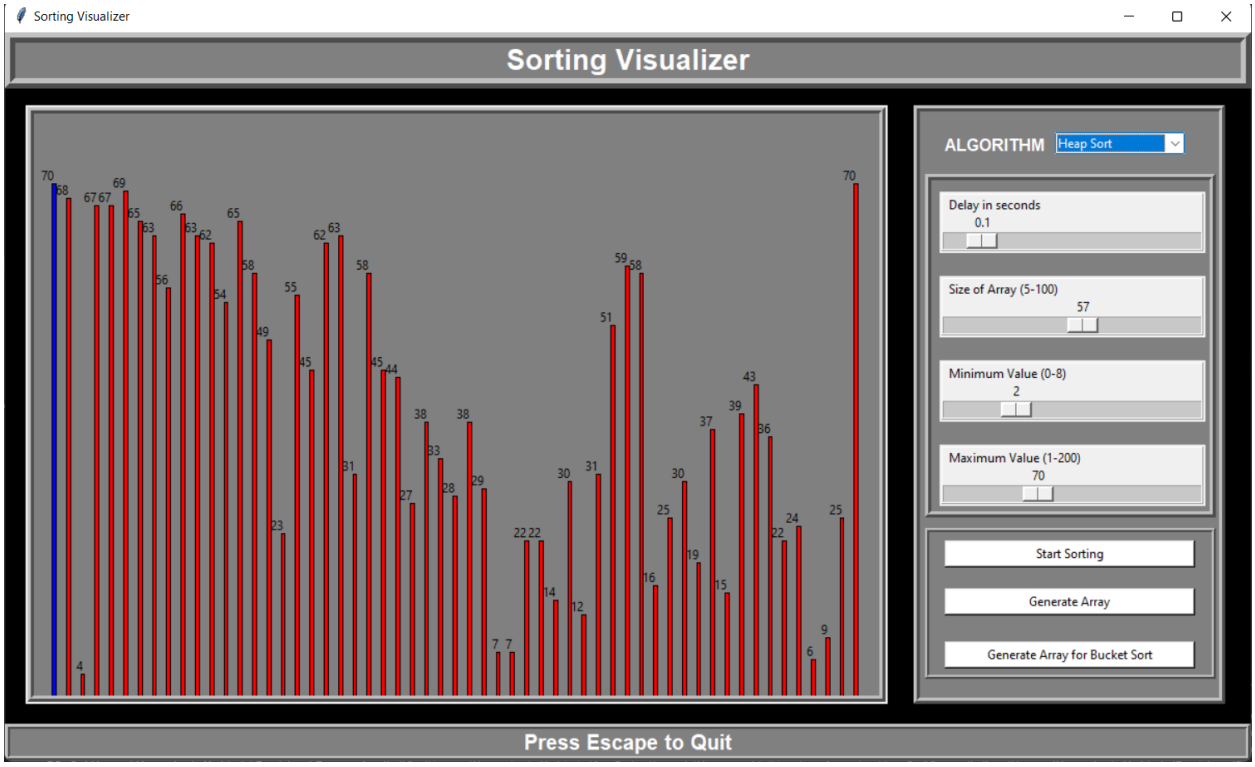

Sorting Visualizer
Sorting Visualizer
ALGORITHM
Merge Sort
Delay in seconds
0.1
Size of Array (5-100)
39
Minimum Value (0-8)
1
Maximum Value (1-200)
105
Start Sorting
Generate Array
Generate Array for Bucket Sort
Press Escape to Quit

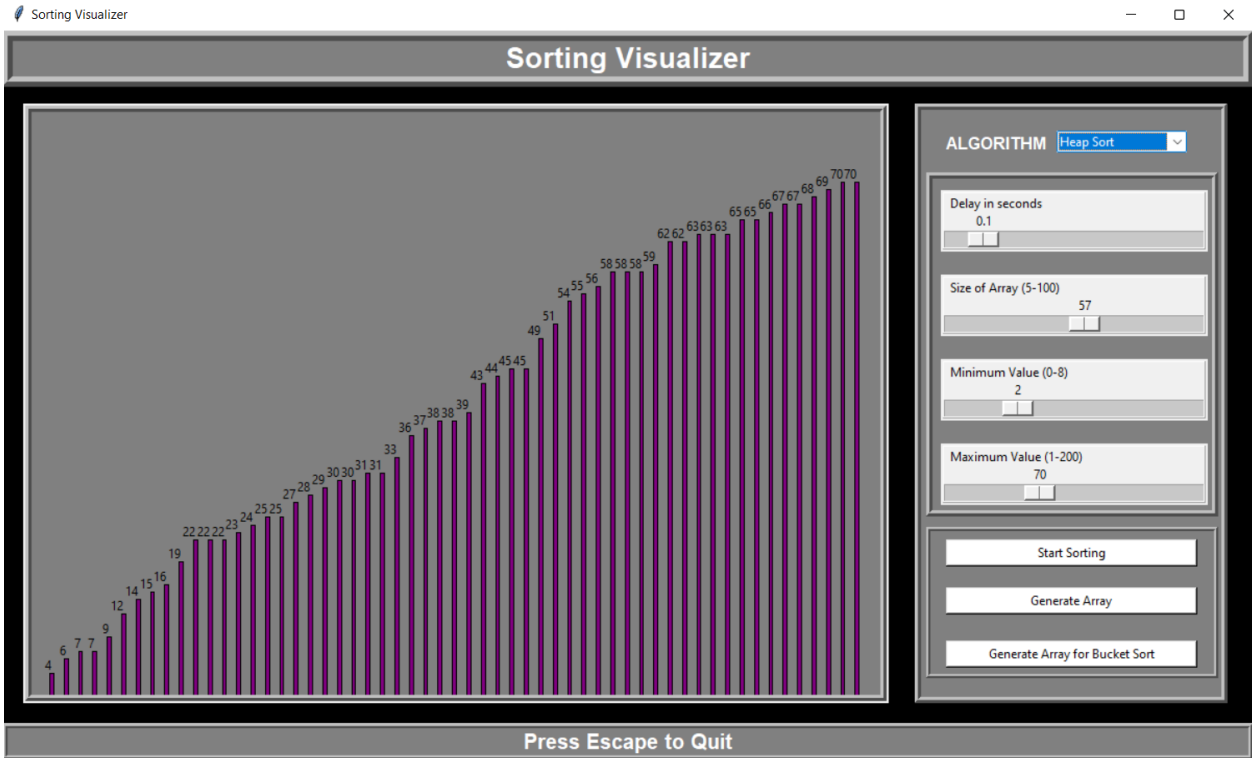

Sorting Visualizer
Sorting Visualizer
ALGORITHM
Merge Sort
Delay in seconds
0.1
Size of Array (5-100)
39
Minimum Value (0-8)
1
Maximum Value (1-200)
105
Start Sorting
Generate Array
Generate Array for Bucket Sort
Press Escape to Quit


Sorting Visualizer
Sorting Visualizer
ALGORITHM
Merge Sort
Delay in seconds
0.1
Size of Array (5-100)
39
Minimum Value (0-8)
1
Maximum Value (1-200)
105
Start Sorting
Generate Array
Generate Array for Bucket Sort
Press Escape to Quit

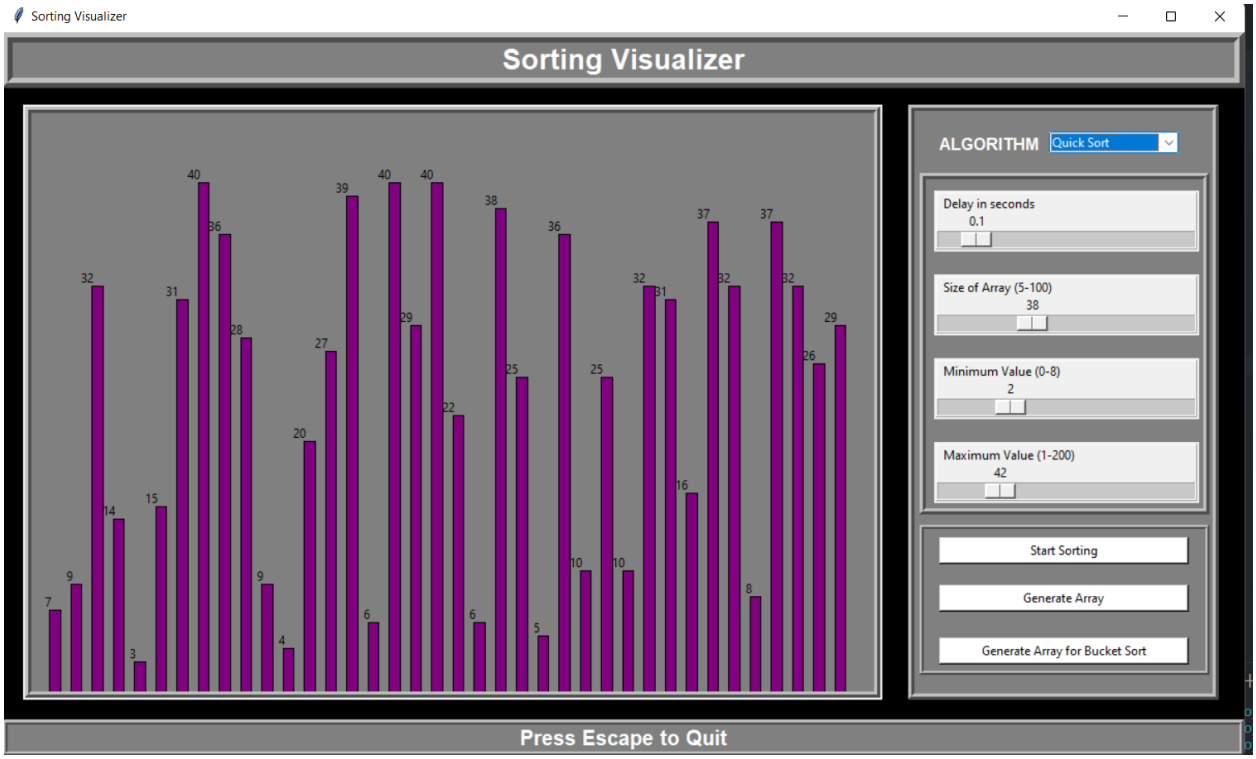
- Heap

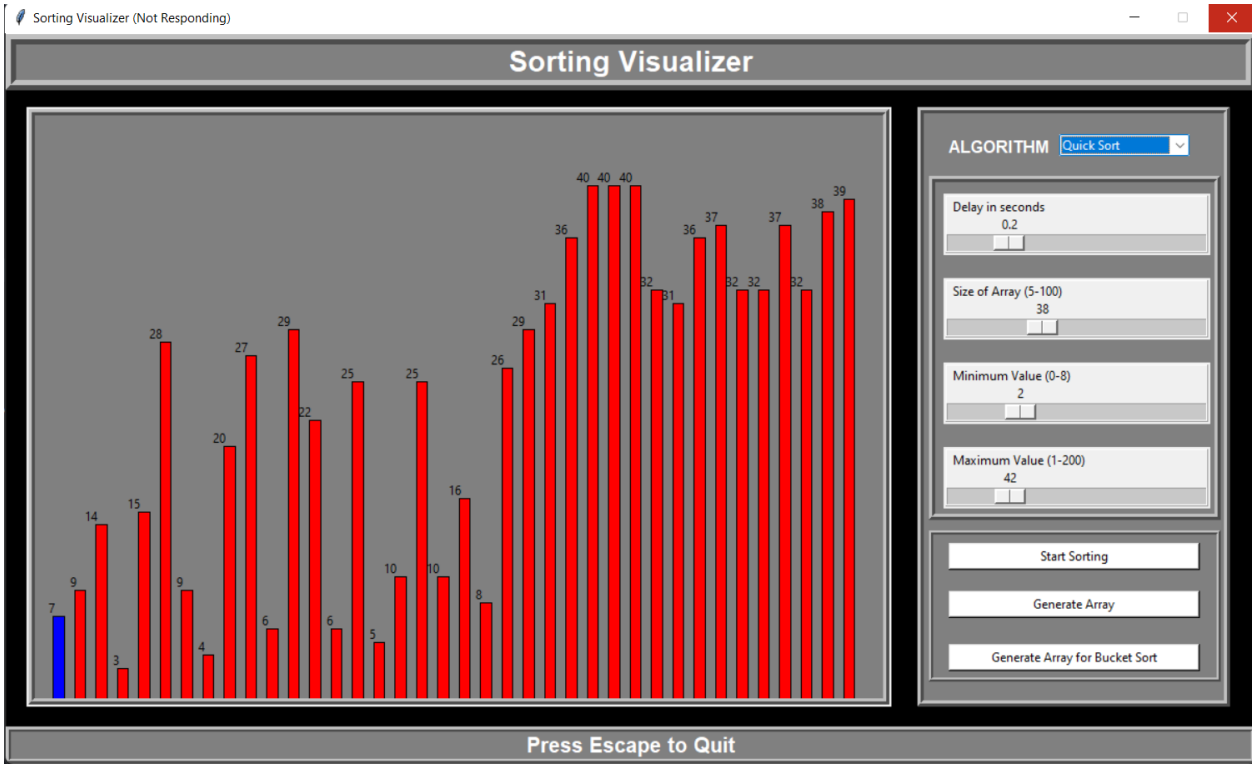

Sorting Visualizer
Sorting Visualizer
ALGORITHM
Heap Sort
Delay in seconds
0.1
Size of Array (5-100)
57
Minimum Value (0-8)
2
Maximum Value (1-200)
70
Start Sorting
Generate Array
Generate Array for Bucket Sort
Press Escape to Quit

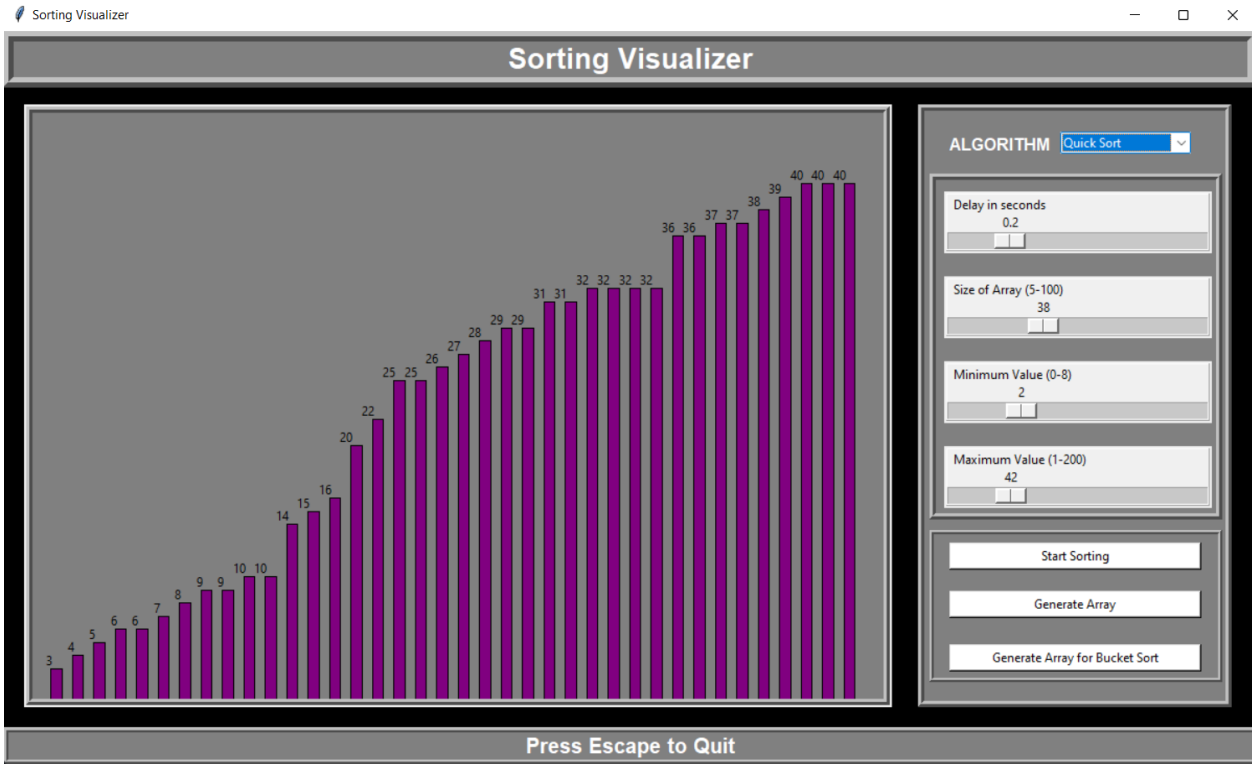

Sorting Visualizer
Sorting Visualizer
ALGORITHM
Heap Sort
Delay in seconds
0.1
Size of Array (5-100)
57
Minimum Value (0-8)
2
Maximum Value (1-200)
70
Start Sorting
Generate Array
Generate Array for Bucket Sort
Press Escape to Quit


Sorting Visualizer
Sorting Visualizer
ALGORITHM
Heap Sort
Delay in seconds
0.1
Size of Array (5-100)
57
Minimum Value (0-8)
2
Maximum Value (1-200)
70
Start Sorting
Generate Array
Generate Array for Bucket Sort
Press Escape to Quit

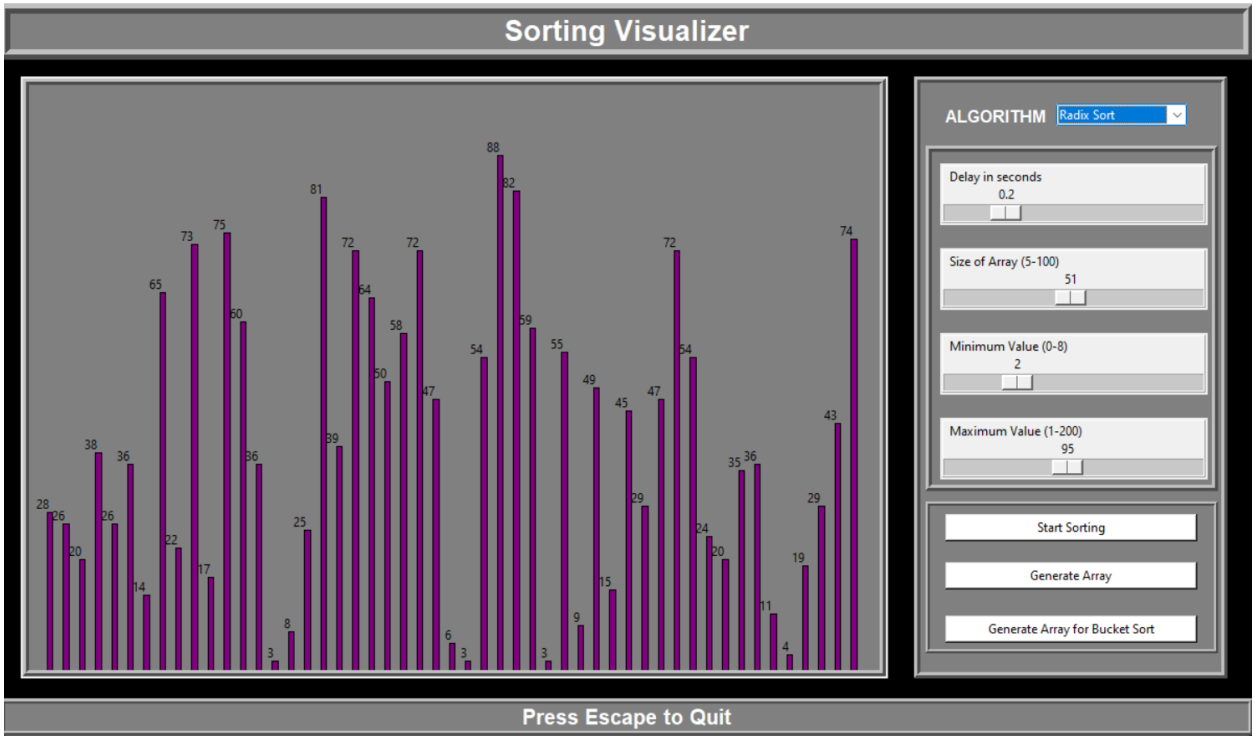
- Quick

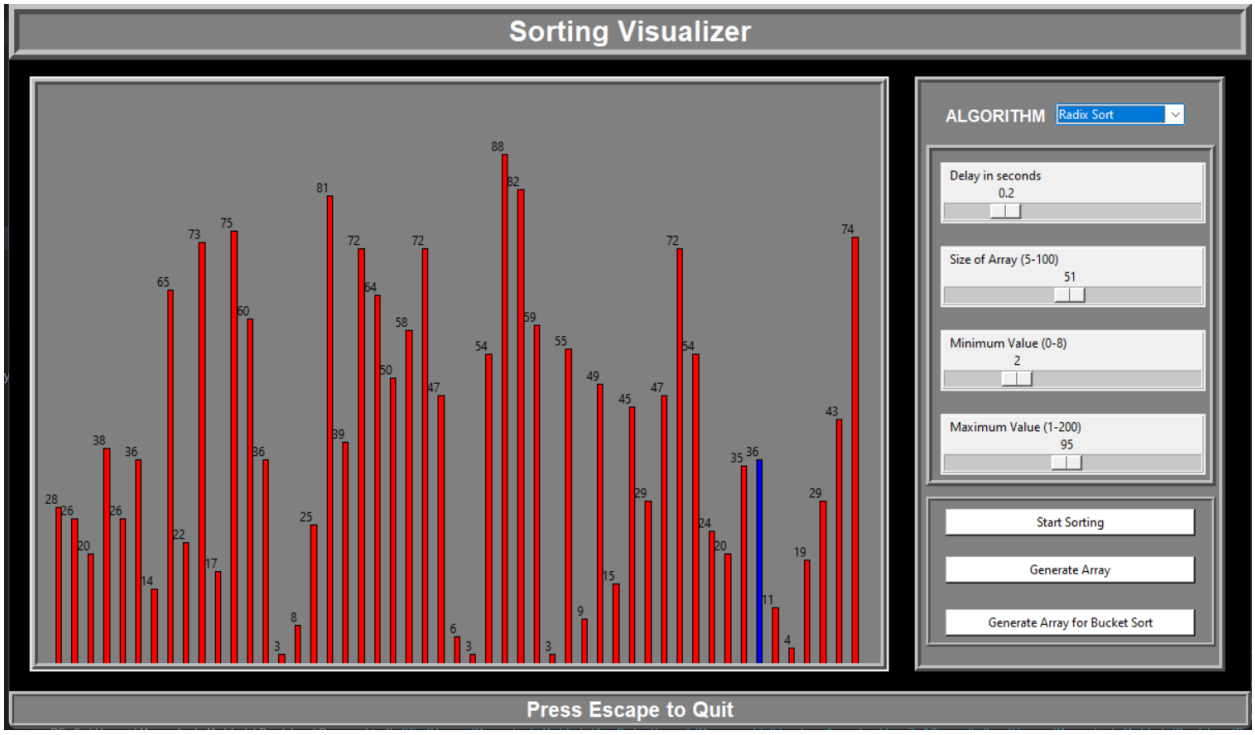

Sorting Visualizer
Sorting Visualizer
ALGORITHM
Quick Sort
Delay in seconds
0.1
Size of Array (5-100)
38
Minimum Value (0-8)
2
Maximum Value (1-200)
42
Start Sorting
Generate Array
Generate Array for Bucket Sort
Press Escape to Quit

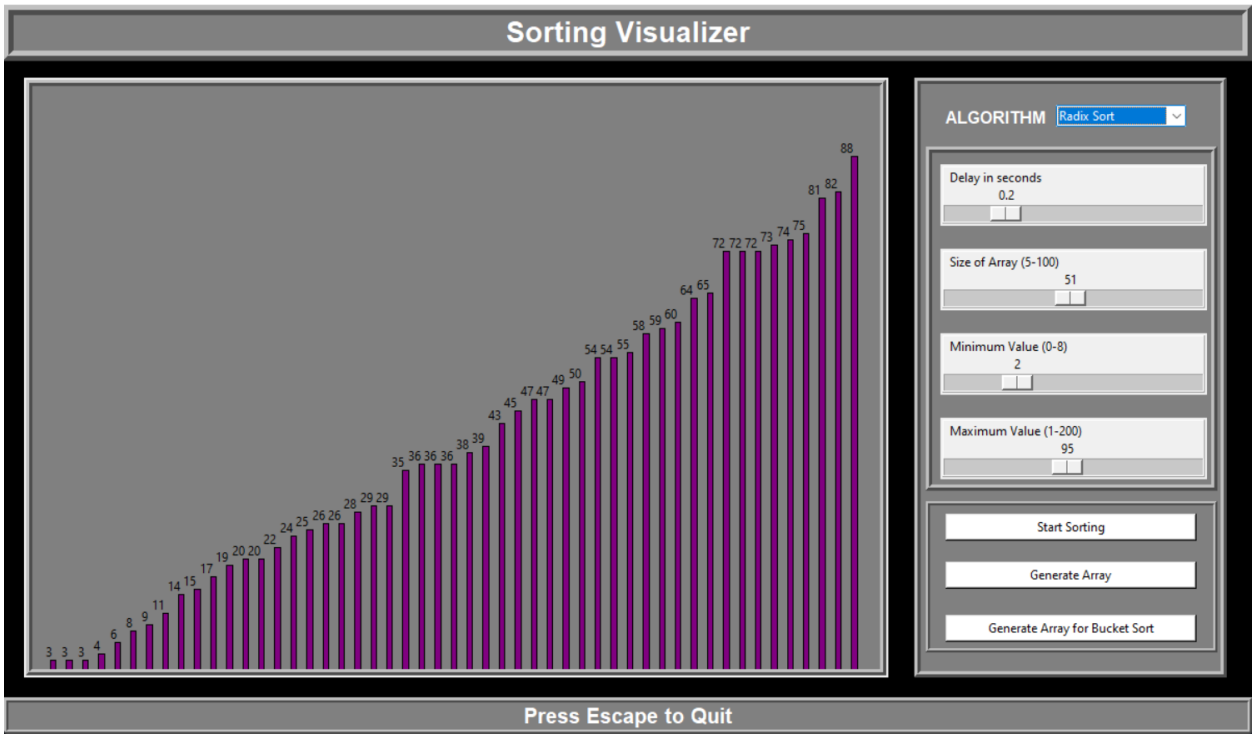

Sorting Visualizer (Not Responding)
Sorting Visualizer
ALGORITHM
Quick Sort
Delay in seconds
0.2
Size of Array (5-100)
38
Minimum Value (0-8)
2
Maximum Value (1-200)
42
Start Sorting
Generate Array
Generate Array for Bucket Sort
Press Escape to Quit


Sorting Visualizer
Sorting Visualizer
ALGORITHM
Quick Sort
Delay in seconds
0.2
Size of Array (5-100)
38
Minimum Value (0-8)
2
Maximum Value (1-200)
42
Start Sorting
Generate Array
Generate Array for Bucket Sort
Press Escape to Quit

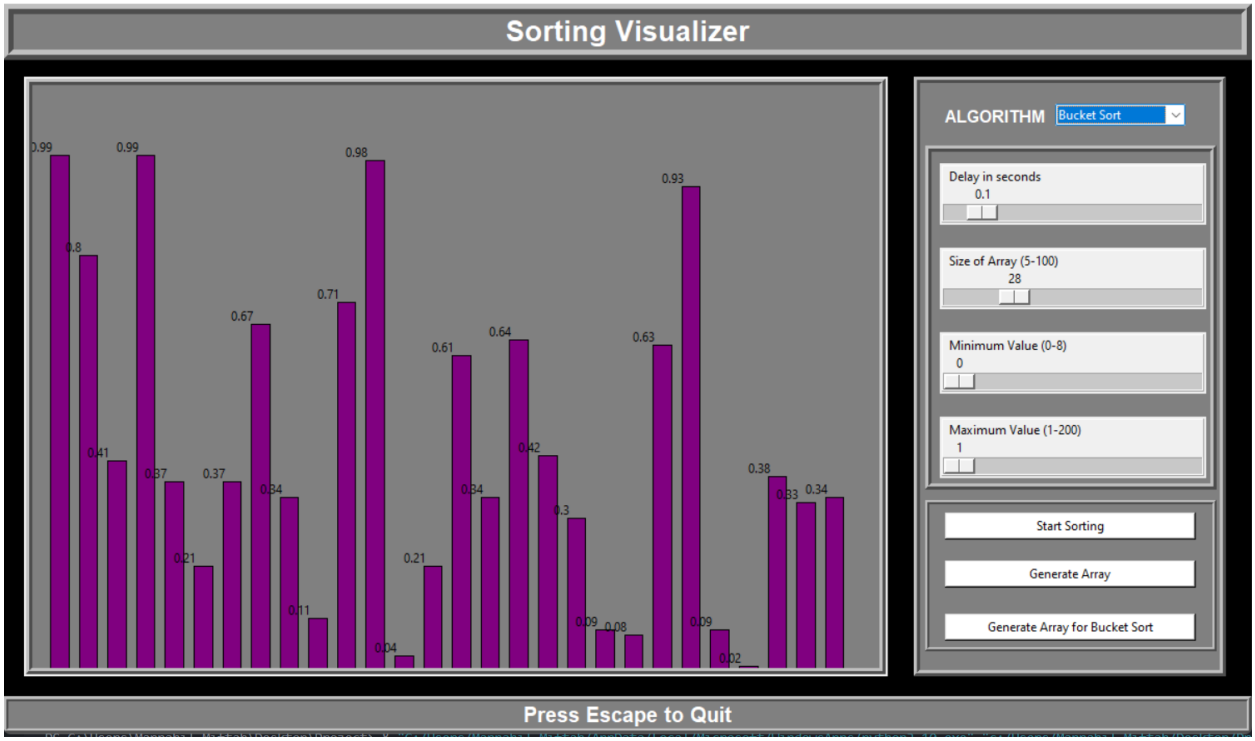
- Radix

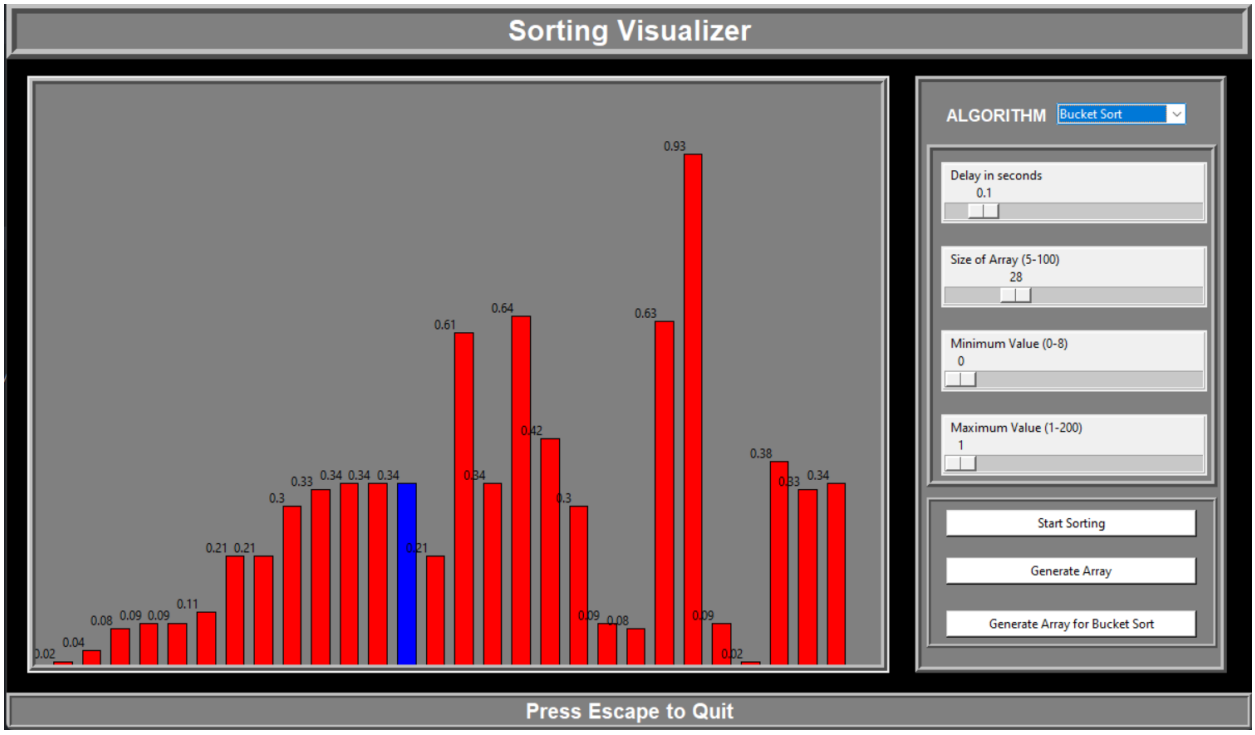

Sorting Visualizer
ALGORITHM
Radix Sort
Delay in seconds
0.2
Size of Array (5-100)
51
Minimum Value (0-8)
2
Maximum Value (1-200)
95
Start Sorting
Generate Array
Generate Array for Bucket Sort
Press Escape to Quit

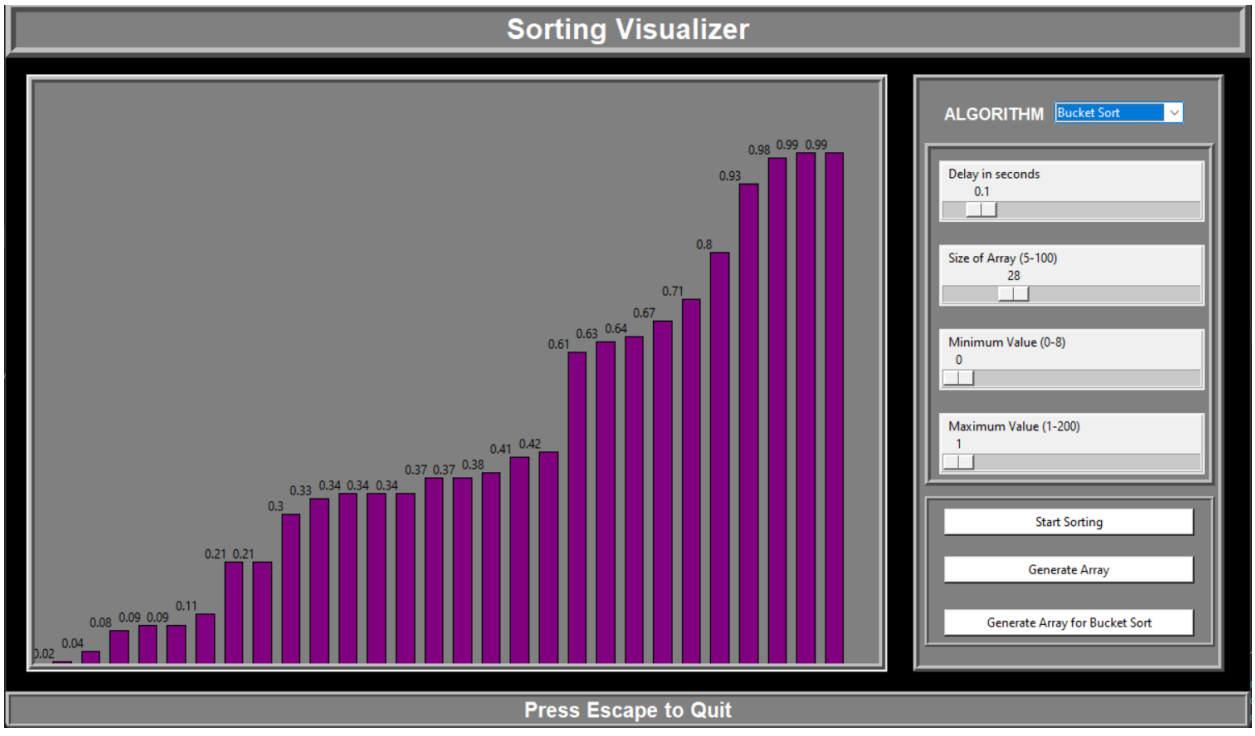

Sorting Visualizer
ALGORITHM
Radix Sort
Delay in seconds
0.2
Size of Array (5-100)
51
Minimum Value (0-8)
2
Maximum Value (1-200)
95
Start Sorting
Generate Array
Generate Array for Bucket Sort
Press Escape to Quit


Sorting Visualizer
ALGORITHM
Radix Sort
Delay in seconds
0.2
Size of Array (5-100)
51
Minimum Value (0-8)
2
Maximum Value (1-200)
95
Start Sorting
Generate Array
Generate Array for Bucket Sort
Press Escape to Quit

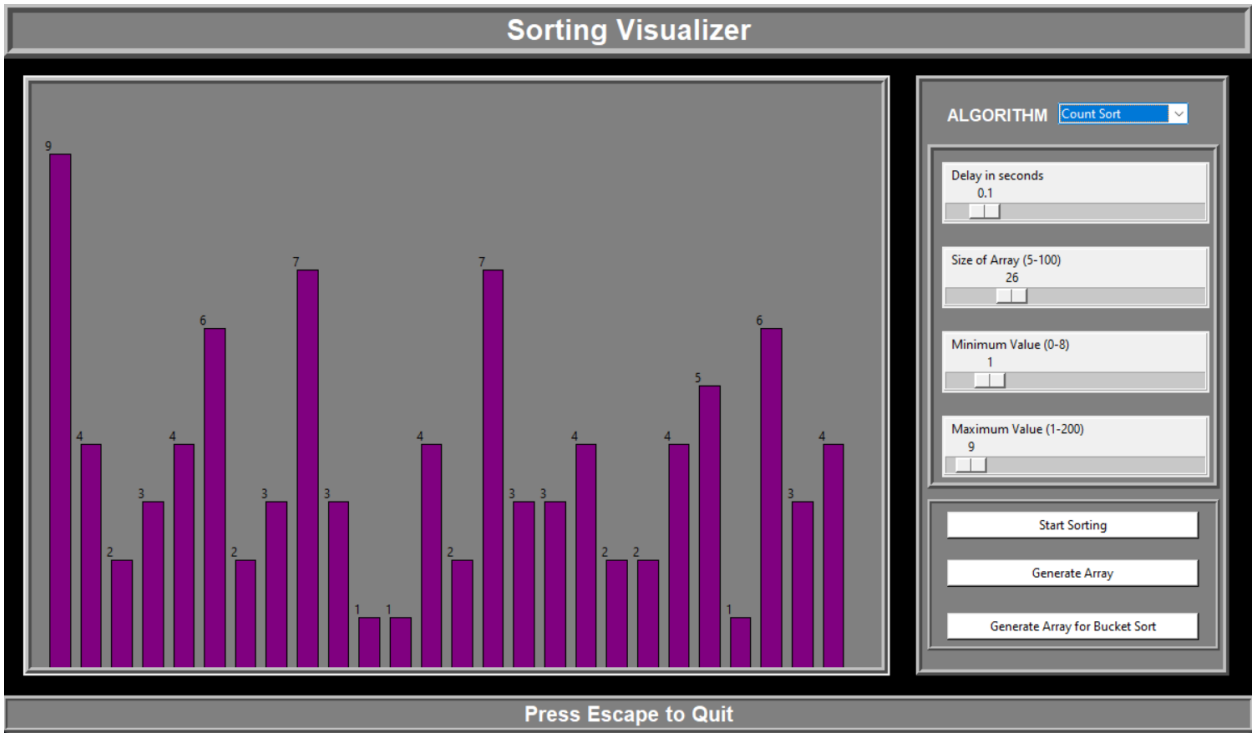
- Bucket

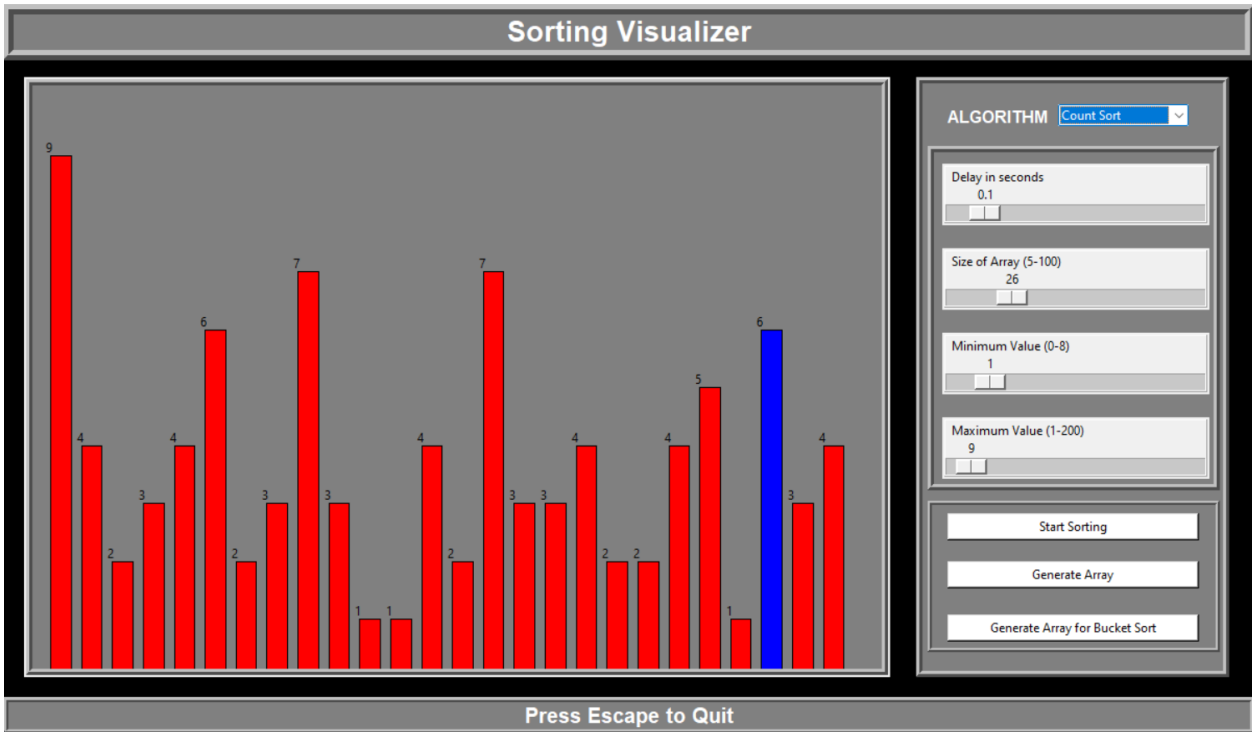

Sorting Visualizer
0.99
0.8
0.41
0.99
0.37
0.21
0.37
0.67
0.34
0.11
0.71
0.98
0.04
0.21
0.61
0.34
0.64
0.42
0.3
0.09
0.08
0.63
0.93
0.09
0.02
0.38
0.33
0.34
ALGORITHM
Bucket Sort
Delay in seconds
0.1
Size of Array (5-100)
28
Minimum Value (0-8)
0
Maximum Value (1-200)
1
Start Sorting
Generate Array
Generate Array for Bucket Sort
Press Escape to Quit

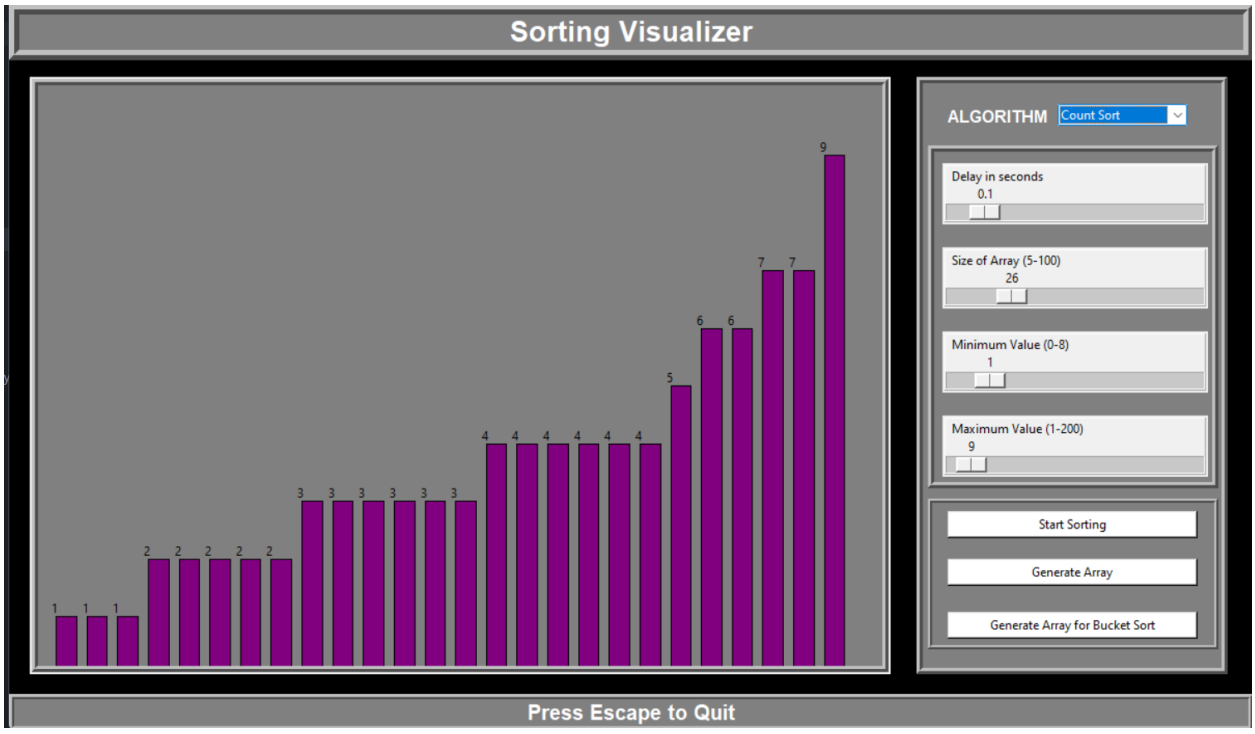

Sorting Visualizer
0.02 0.04 0.08 0.09 0.09 0.11 0.21 0.21 0.3 0.33 0.34 0.34 0.34 0.21 0.61 0.34 0.64 0.42 0.3 0.09 0.08 0.63 0.93 0.09 0.02 0.38 0.33 0.34
ALGORITHM
Bucket Sort
Delay in seconds
0.1
Size of Array (5-100)
28
Minimum Value (0-8)
0
Maximum Value (1-200)
1
Start Sorting
Generate Array
Generate Array for Bucket Sort
Press Escape to Quit


Sorting Visualizer
0.02 0.04 0.08 0.09 0.09 0.11 0.21 0.21 0.3 0.33 0.34 0.34 0.34 0.37 0.37 0.38 0.41 0.42 0.61 0.63 0.64 0.67 0.71 0.8 0.93 0.98 0.99 0.99
ALGORITHM
Bucket Sort
Delay in seconds
0.1
Size of Array (5-100)
28
Minimum Value (0-8)
0
Maximum Value (1-200)
1
Start Sorting
Generate Array
Generate Array for Bucket Sort
Press Escape to Quit

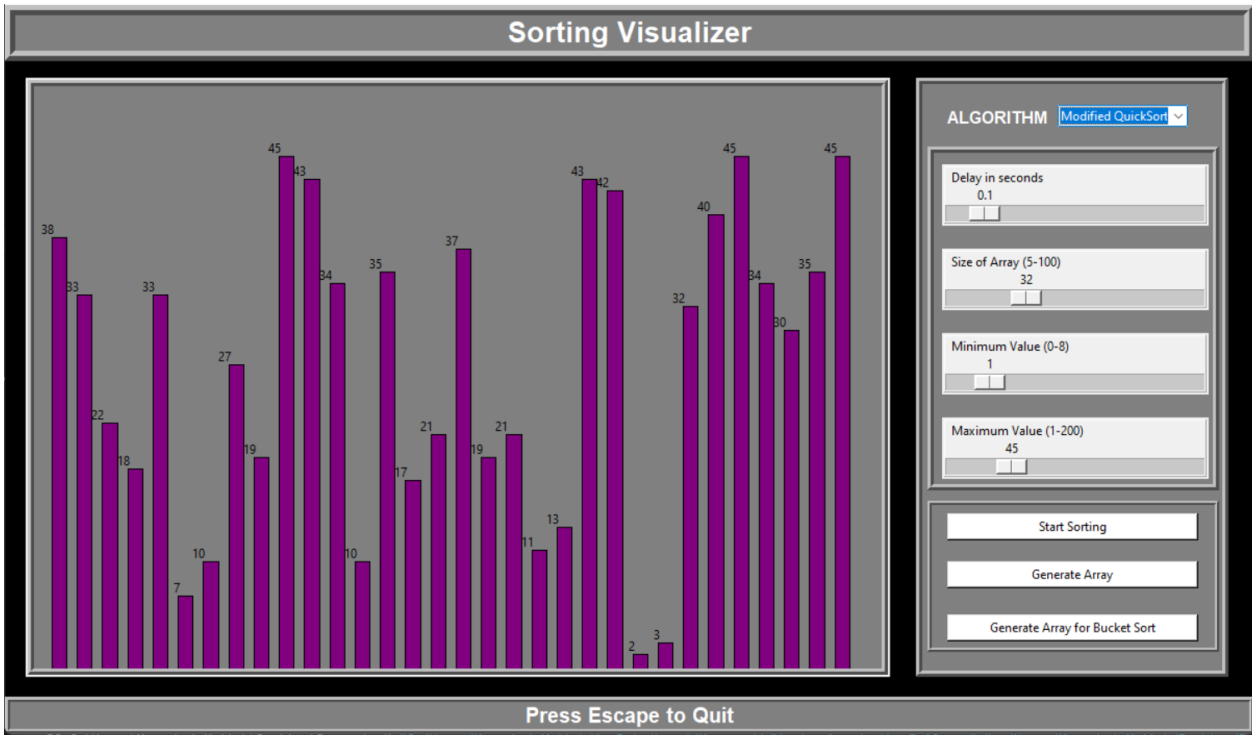
- Count

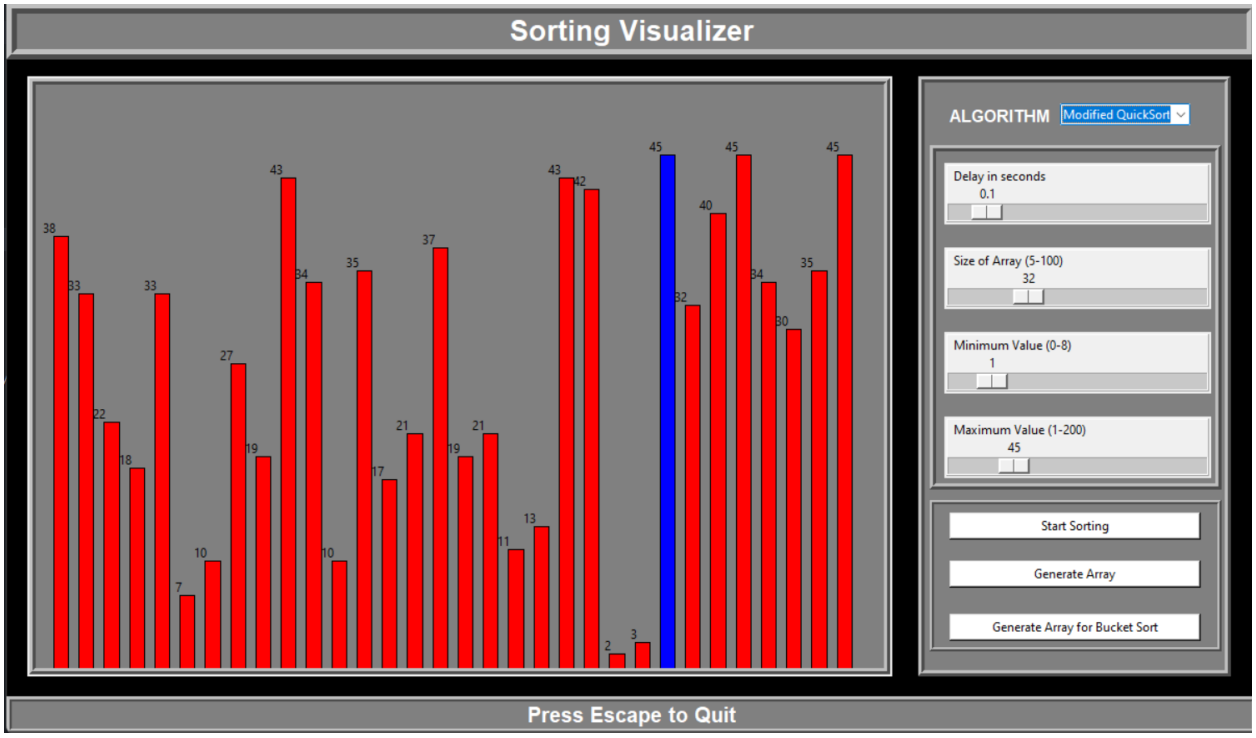

Sorting Visualizer
ALGORITHM
Count Sort
Delay in seconds
0.1
Size of Array (5-100)
26
Minimum Value (0-8)
1
Maximum Value (1-200)
9
Start Sorting
Generate Array
Generate Array for Bucket Sort
Press Escape to Quit

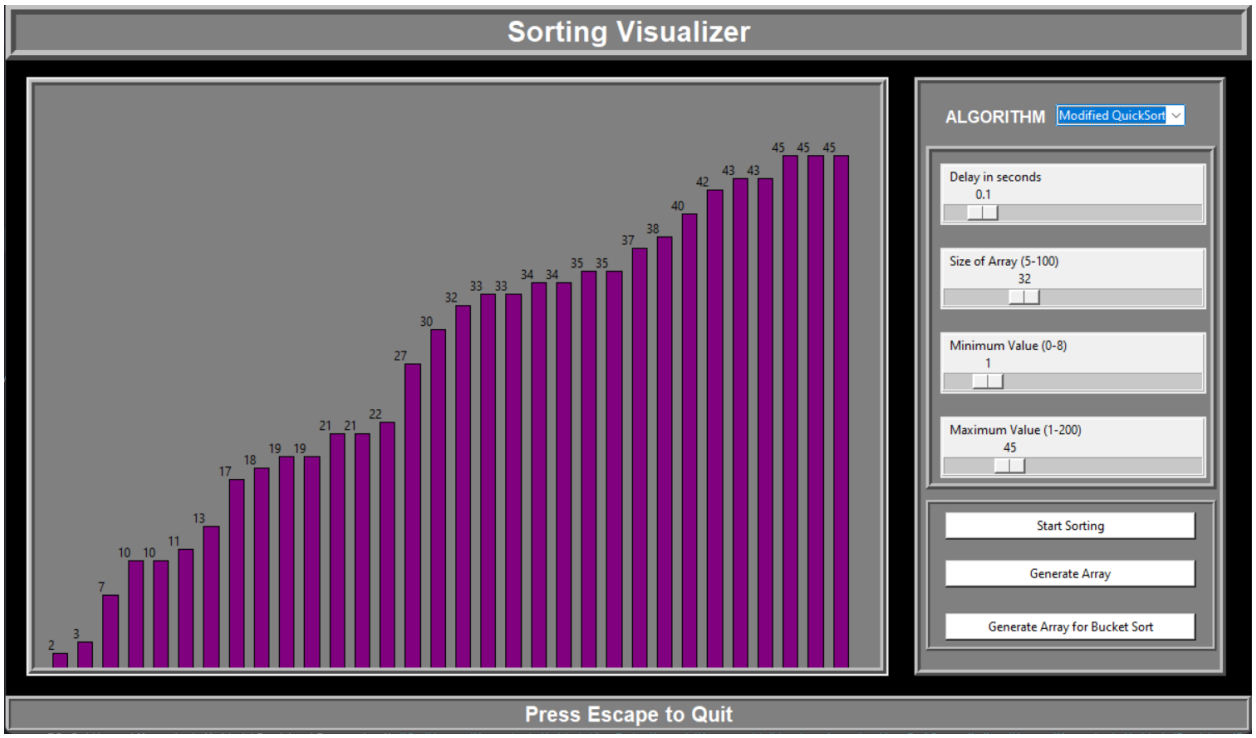

Sorting Visualizer
ALGORITHM
Count Sort
Delay in seconds
0.1
Size of Array (5-100)
26
Minimum Value (0-8)
1
Maximum Value (1-200)
9
Start Sorting
Generate Array
Generate Array for Bucket Sort
Press Escape to Quit


Sorting Visualizer
ALGORITHM
Count Sort
Delay in seconds
0.1
Size of Array (5-100)
26
Minimum Value (0-8)
1
Maximum Value (1-200)
9
Start Sorting
Generate Array
Generate Array for Bucket Sort
Press Escape to Quit

- Modified Quick




Sorting Visualizer
ALGORITHM
Modified QuickSort
Delay in seconds
0.1
Size of Array (5-100)
32
Minimum Value (0-8)
1
Maximum Value (1-200)
45
Start Sorting
Generate Array
Generate Array for Bucket Sort
Press Escape to Quit


Sorting Visualizer
ALGORITHM
Modified QuickSort
Delay in seconds
0.1
Size of Array (5-100)
32
Minimum Value (0-8)
1
Maximum Value (1-200)
45
Start Sorting
Generate Array
Generate Array for Bucket Sort
Press Escape to Quit

- Modified Count

modifiedcount.py > ...

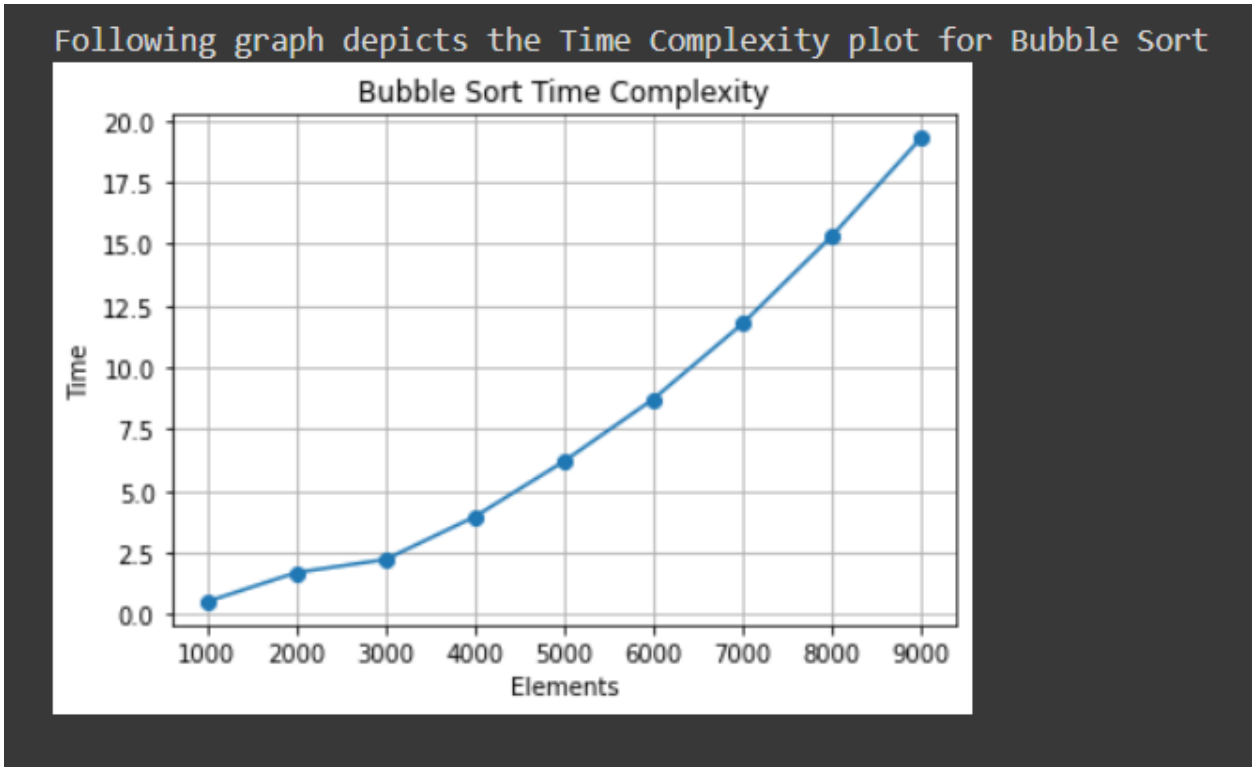
```python
def m_count(data,a,b):
    #a=int(input("Enter Starting Range"))
    #b=int(input("Enter Ending Range"))
    k = max(data)+1
    #output = [0] * len(data)
    count = [0] * k
    for i in range(0, len(data)):
        count[data[i]] += 1
    for i in range(1,k):
        count[i] += count[i-1]
    if a == 0:
        print(count[b])
    else:
        print(count[b]-count[a-1])
    '''for i in range(0,len(data)):
        drawData(data, ['Green' if x == data[i] else 'Red' for x in range(len(data))])
        time.sleep(timer)'''

data=[9,4,2,7,1,2,4,3,10,12,22,23]
a=int(input("Enter starting range:"))
b=int(input("Enter ending range:"))
m_count(data,a,b)
```

```
Enter starting range:5
Enter ending range:13
4
PS C:\Users\Mannahil Miftah\Desktop\Project>
```

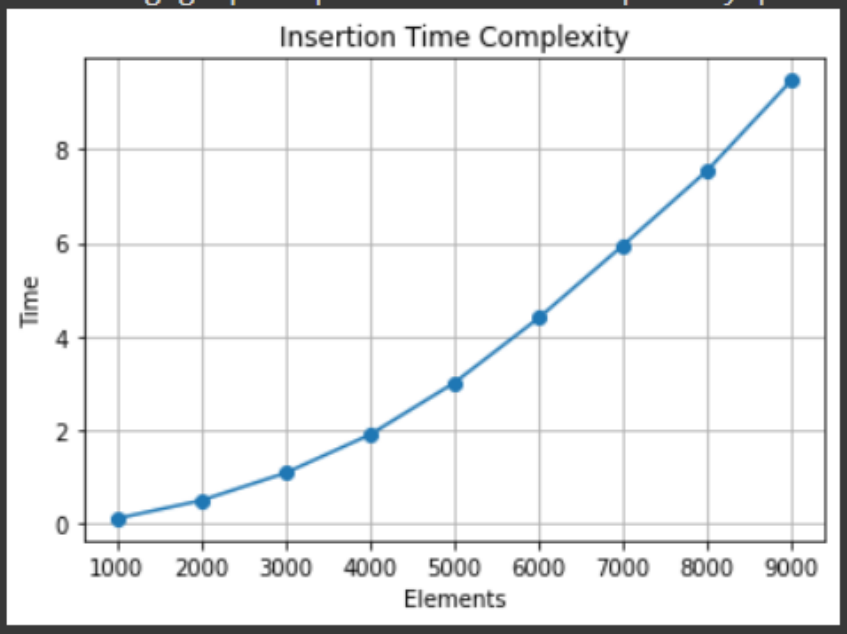
TIME COMPLEXITY (1000-9000 numbers)

- Bubble Sort



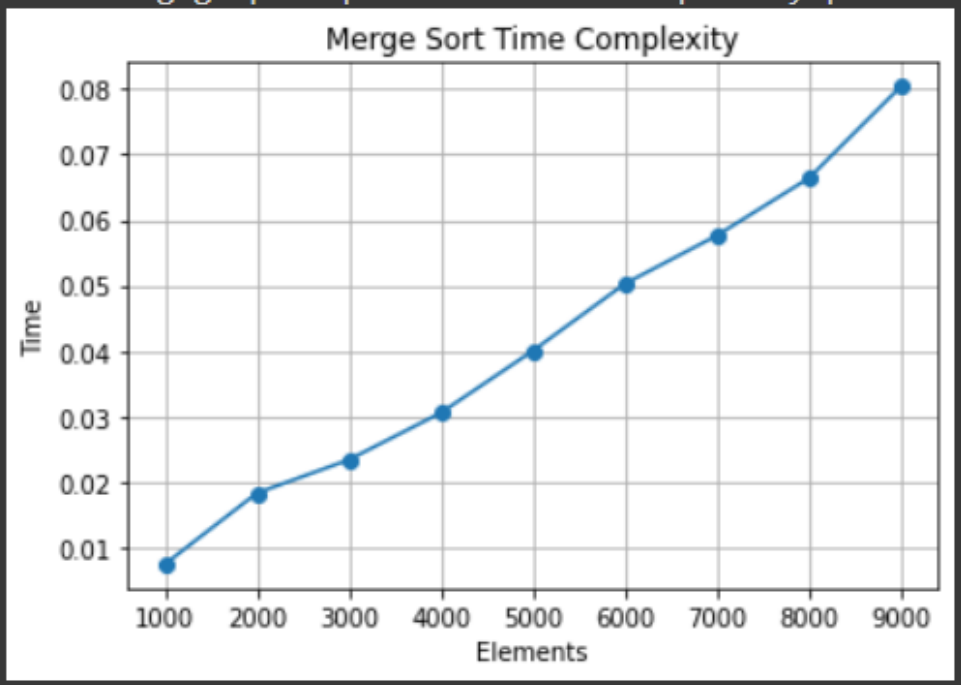
- Insertion

Following graph depicts the Time Complexity plot for Insertion Sort



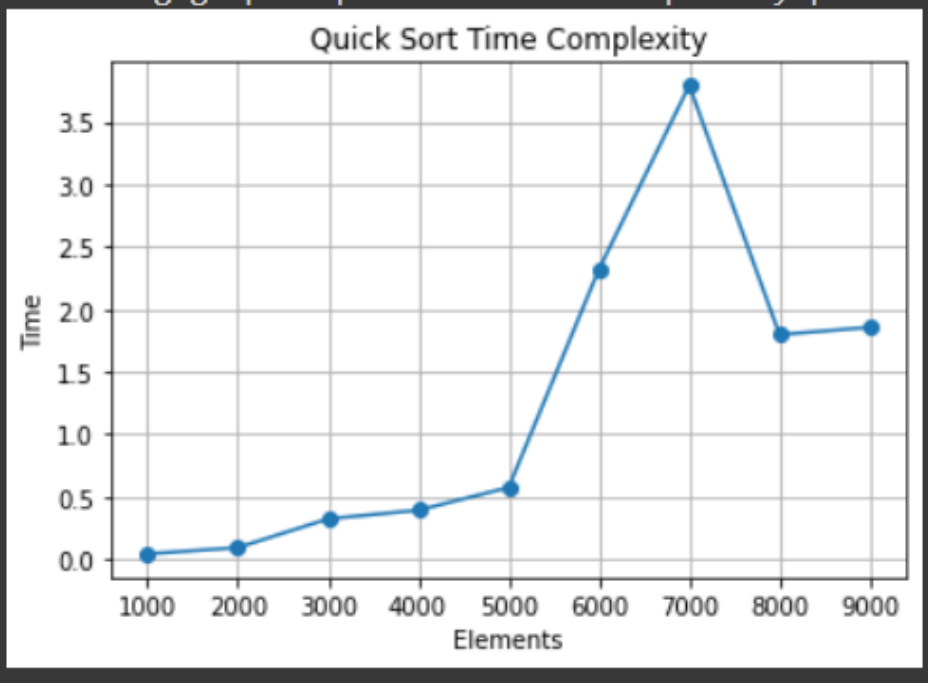
- Merge

Following graph depicts the Time Complexity plot for Merge Sort



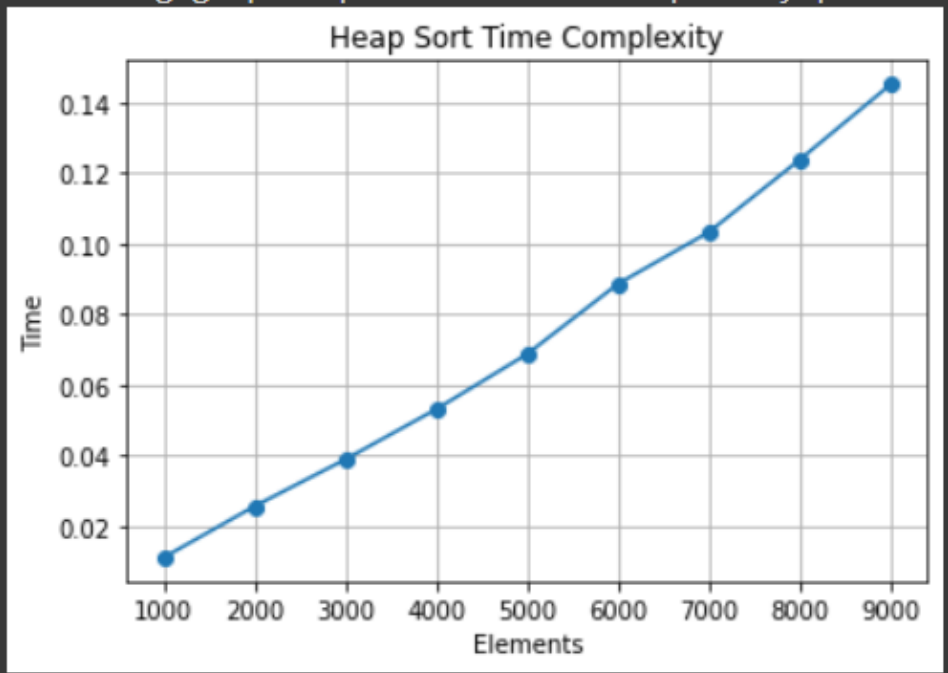
- Quick

Following graph depicts the Time Complexity plot for Quick Sort:



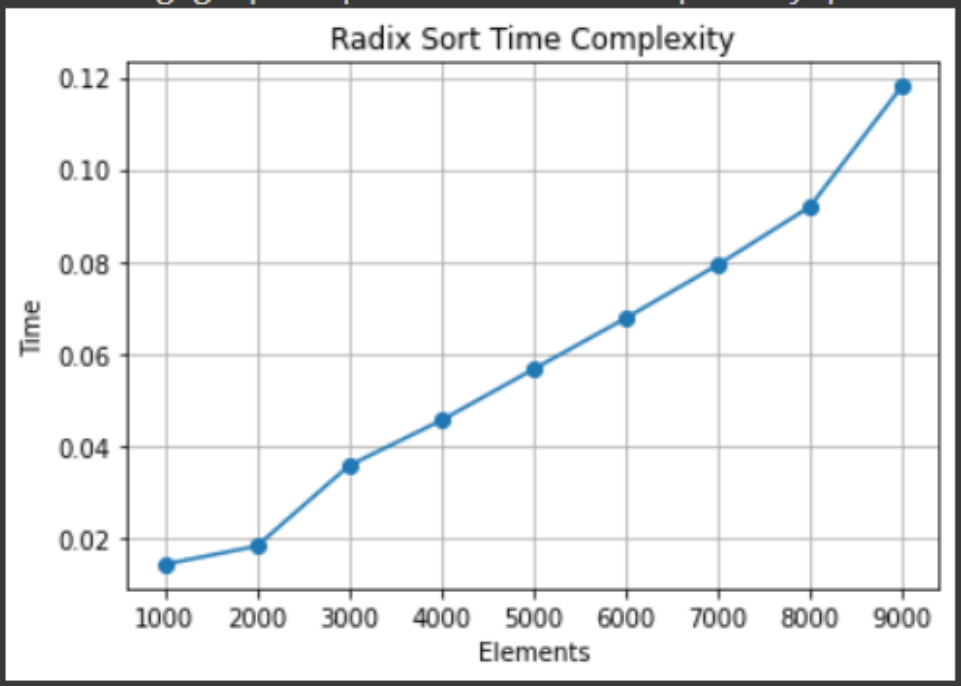
- Heap

Following graph depicts the Time Complexity plot for Heap Sort



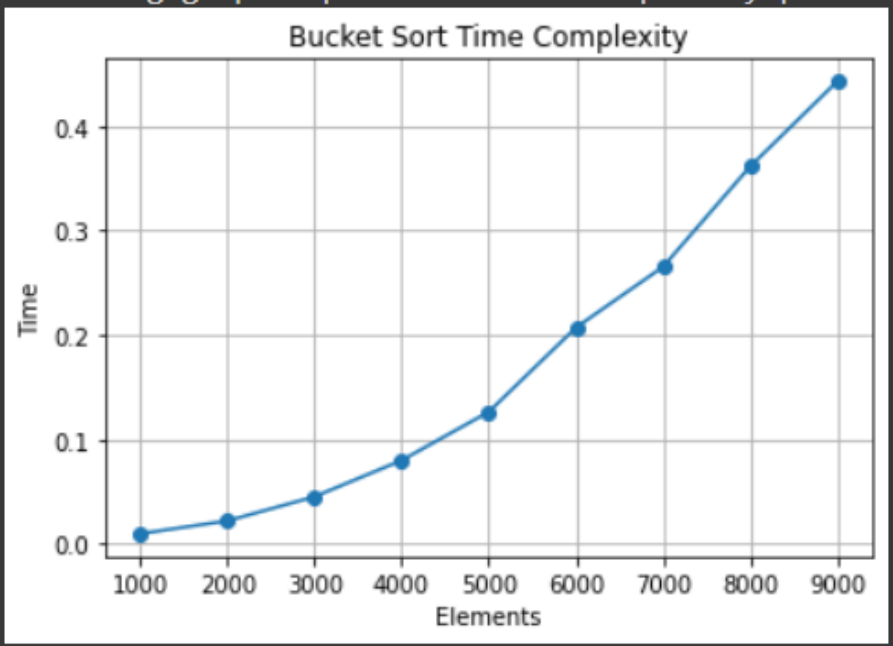
- Radix

Following graph depicts the Time Complexity plot for Radix Sort



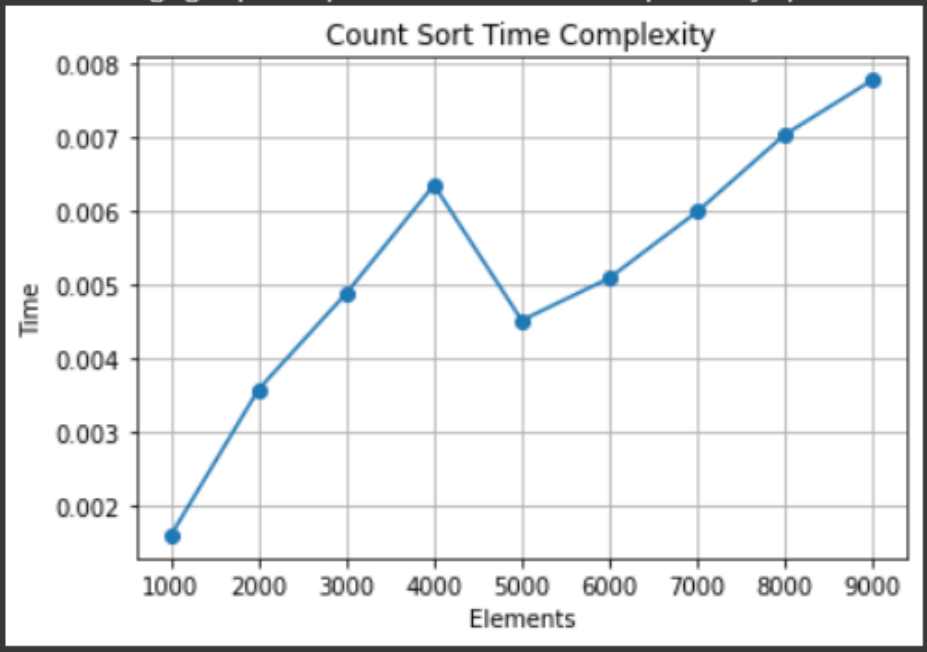
- Bucket

Following graph depicts the Time Complexity plot for Bucket Sort



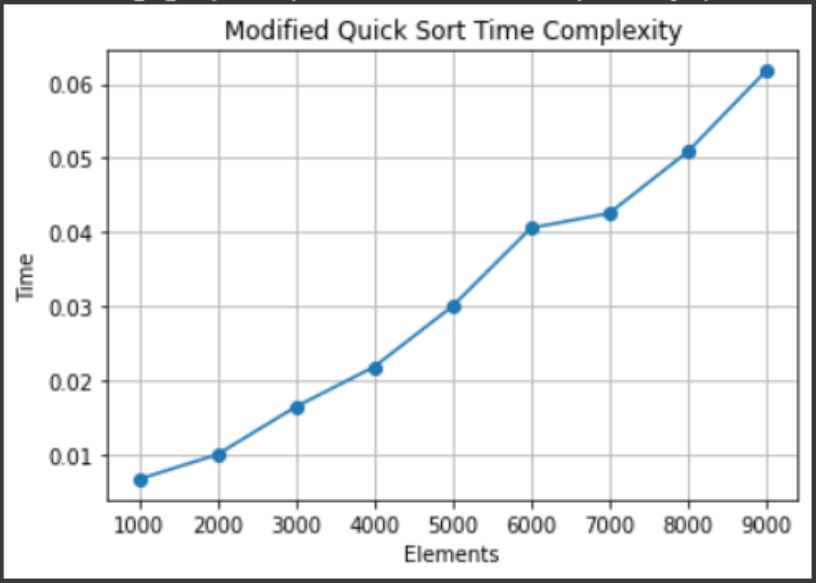
- Count

Following graph depicts the Time Complexity plot for Count Sort

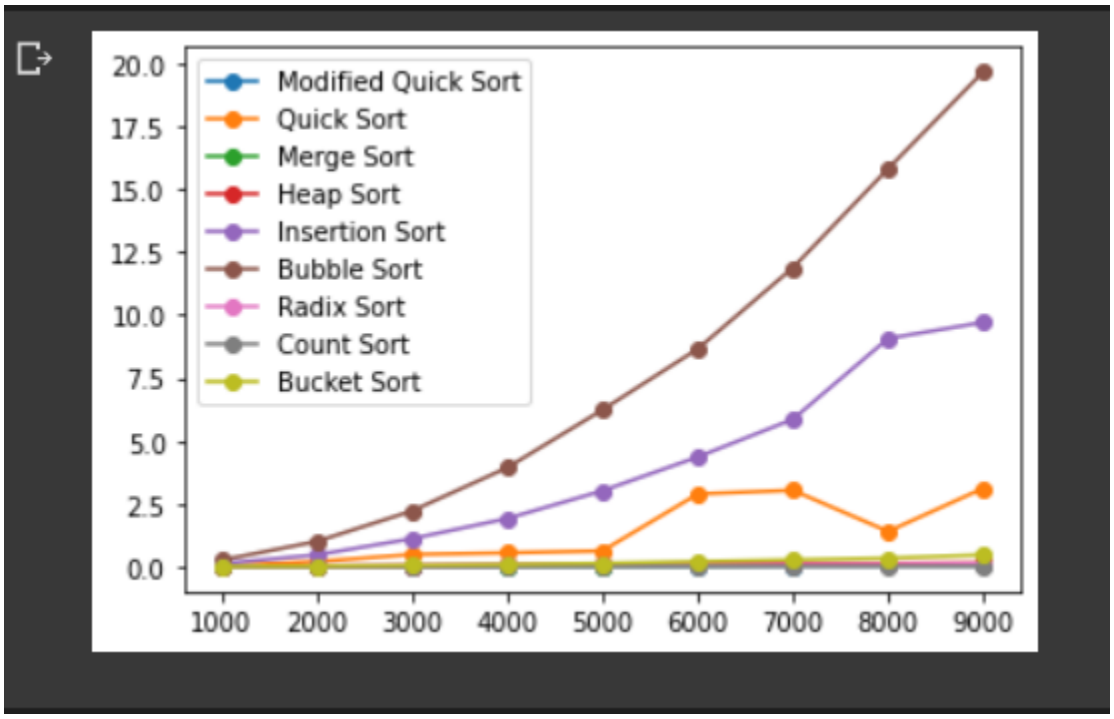


- Modified Quick

Following graph depicts the Time Complexity plot for Modified Quick Sort



COMPARISON OF TIME COMPLEXITIES



CONCLUSION

Implementing the algorithms while trying hard to materialize every incorporated step within, hounded us to learn exponentially about every aspect of every sorting strategy. The visualized effect of sorting algorithms ingrained its picture deep into the memory that the learned steps are hard to efface.

REFERENCES

https://docs.python.org/3/library/tk.html

https://anzeljg.github.io/rin2/book2/2405/docs/tkinter/index.htm