CSE 5311-004
Design and Analysis of Algorithms
Project Report

Project Topic -Sorting Algorithms

Sorting Algorithm is the algorithm in which elements are arrange in certain order, in order of numeric digits, lexicographical order.

Sorting Algorithms included in this project are as follows:

- Selection Sort
- Insertion Sort
- Bubble Sort
- Merge Sort
- Heap Sort
- Quick Sort
- Quick Sort - Using 3 medians

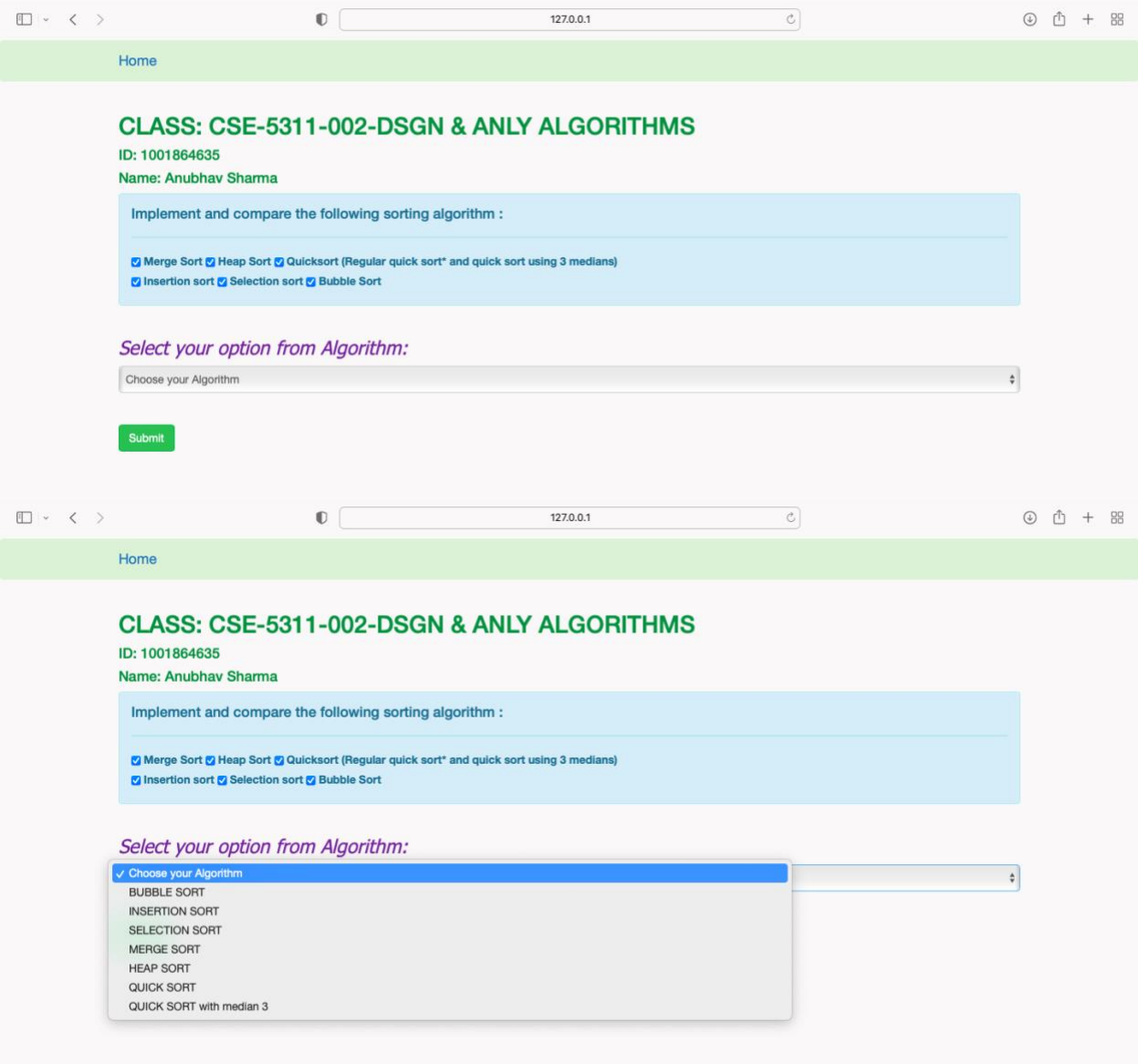
Tools and Language Details:

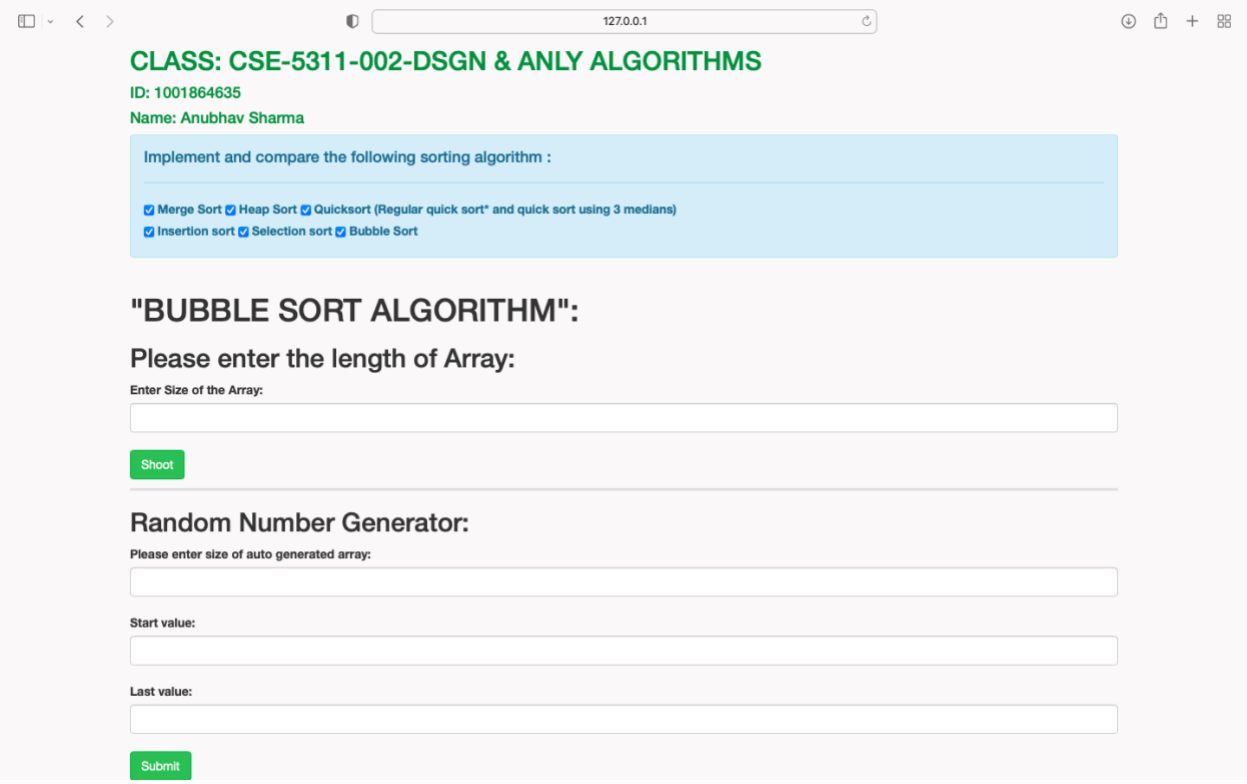
- Python
- IDE: PY Charm
- Framework: Flask

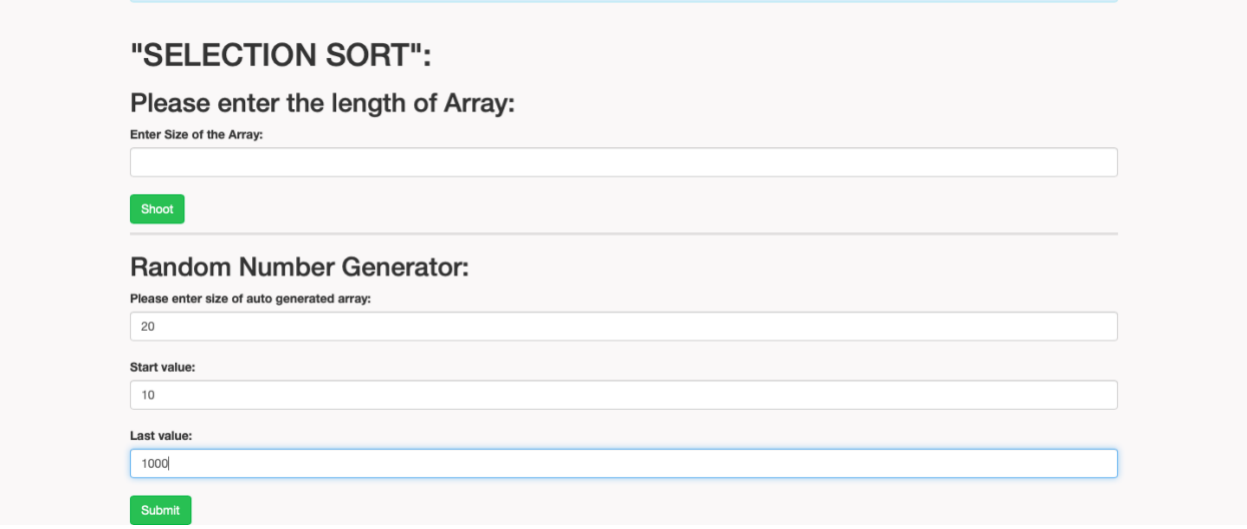
By: Anubhav Sharma

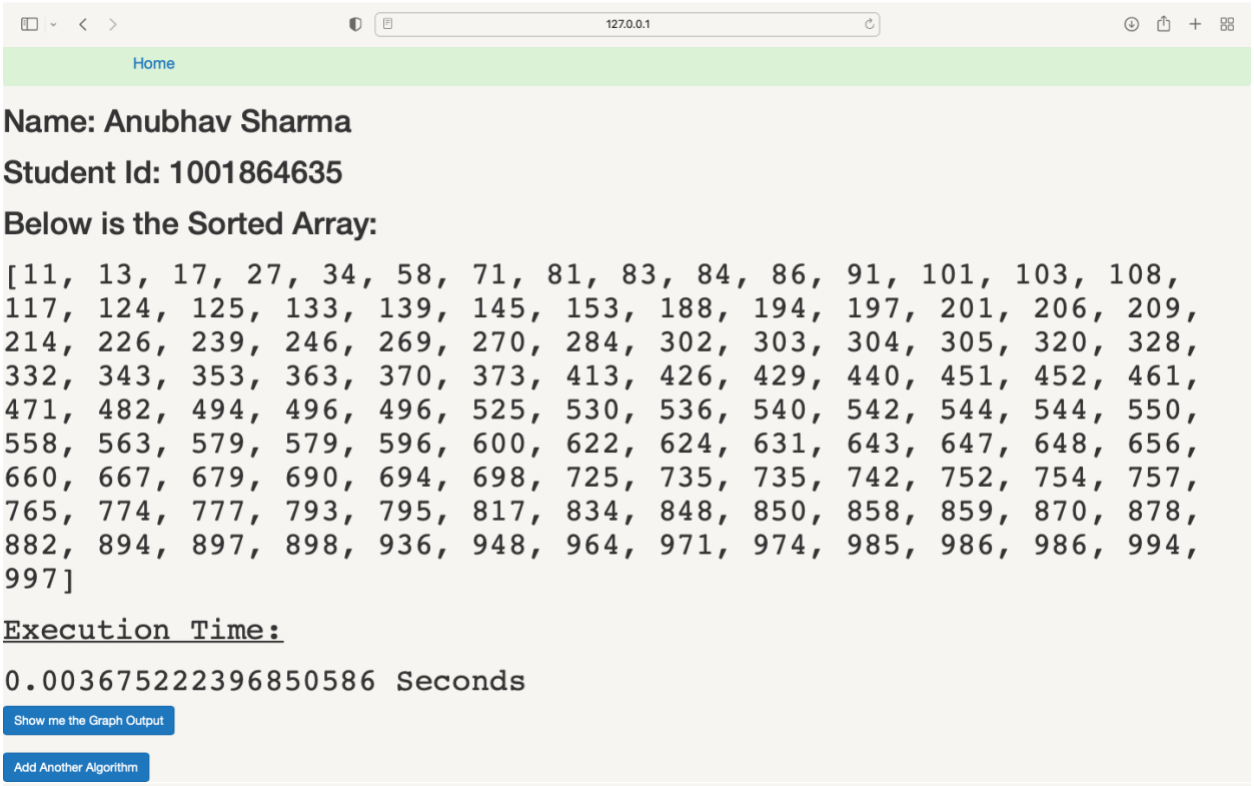
ID: 1001864635

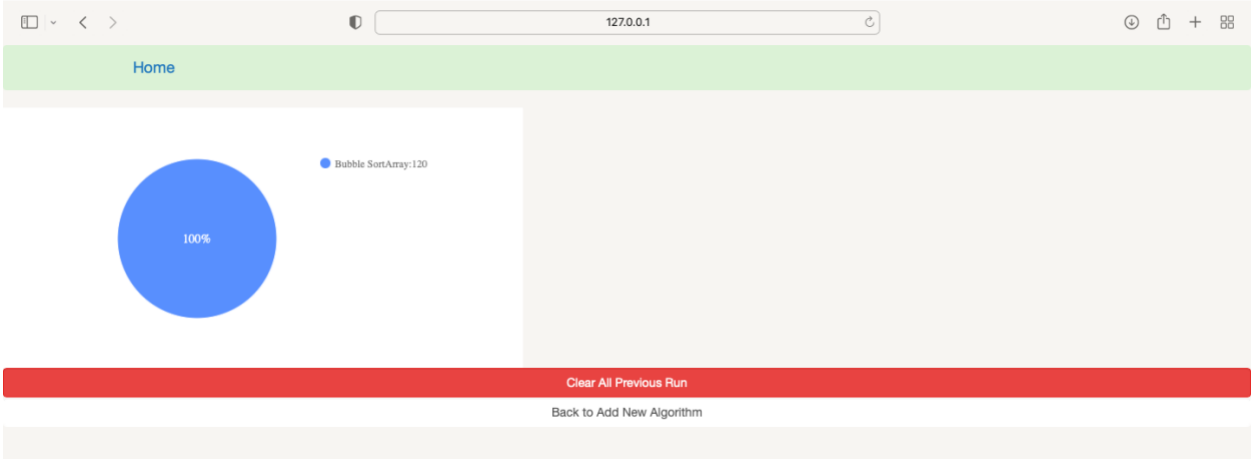
GRAPHICAL USER INERFACE




127.0.0.1
CLASS: CSE-5311-002-DSGN & ANLY ALGORITHMS
ID: 1001864635
Name: Anubhav Sharma
Implement and compare the following sorting algorithm :
Merge Sort Heap Sort Quicksort (Regular quick sort* and quick sort using 3 medians)
Insertion sort Selection sort Bubble Sort
"BUBBLE SORT ALGORITHM":
Please enter the length of Array:
Enter Size of the Array:
Shoot
Random Number Generator:
Please enter size of auto generated array:
Start value:
Last value:
Submit


"SELECTION SORT":
Please enter the length of Array:
Enter Size of the Array:
Shoot
Random Number Generator:
Please enter size of auto generated array:
20
Start value:
10
Last value:
1000
Submit


127.0.0.1
Home
Name: Anubhav Sharma
Student Id: 1001864635
Below is the Sorted Array:
[11, 13, 17, 27, 34, 58, 71, 81, 83, 84, 86, 91, 101, 103, 108,
117, 124, 125, 133, 139, 145, 153, 188, 194, 197, 201, 206, 209,
214, 226, 239, 246, 269, 270, 284, 302, 303, 304, 305, 320, 328,
332, 343, 353, 363, 370, 373, 413, 426, 429, 440, 451, 452, 461,
471, 482, 494, 496, 496, 525, 530, 536, 540, 542, 544, 544, 550,
558, 563, 579, 579, 596, 600, 622, 624, 631, 643, 647, 648, 656,
660, 667, 679, 690, 694, 698, 725, 735, 735, 742, 752, 754, 757,
765, 774, 777, 793, 795, 817, 834, 848, 850, 858, 859, 870, 878,
882, 894, 897, 898, 936, 948, 964, 971, 974, 985, 986, 986, 994,
997]
Execution Time:
0.003675222396850586 Seconds
Show me the Graph Output
Add Another Algorithm


127.0.0.1
Home
Bubble SortArray:120
100%
Clear All Previous Run
Back to Add New Algorithm

SELECTION SORT

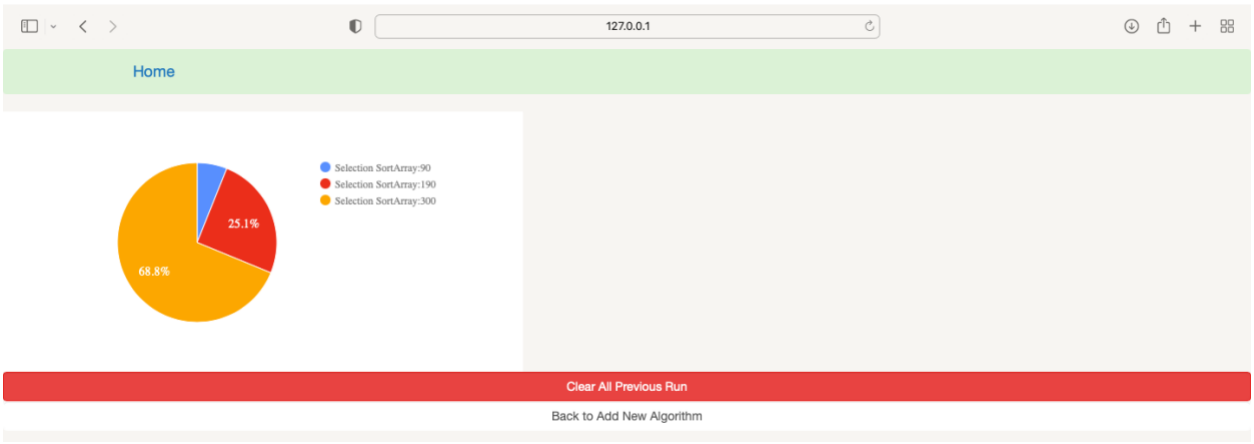
- This algorithm selects the smallest element in the array and places it in the first position.
- It will search the minimum element and if that element is smaller than the current then it will swap with the minimum element.
- If it is greater than the minimum element then, increase the minimum.
- If array is already sorted, then time complexity can be improved.
- No more than O(n) swaps are performed.

Selection Sort is an in-place algorithm and is useful for small lists

Best Time Complexity: $O(n^2)$ Worst Time Complexity: $O(n^2)$
Space Complexity: O(1)

Observation:

We can observe that as length of Array is increasing time also increases



INSERTION SORT

- Insertion Sort is an in-place algorithm wherein the input array is destroyed while sorting process.
- This algorithm first initialises an element and then moves that element to the correct position by shifting other elements.
- Time complexity can be improved by providing input as sorted array or reverse sorted array

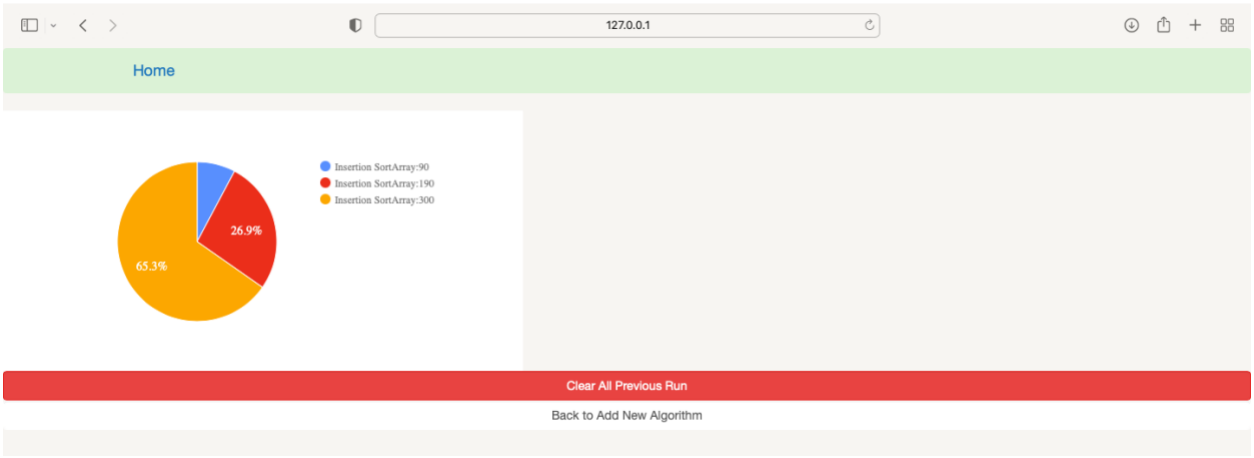
Worst case Time Complexity: $O(n^2)$

Best Case Time Complexity: O(n)

Space Complexity: O(1)

Observation:

We can Observe that as length of Array is increasing time also increases.



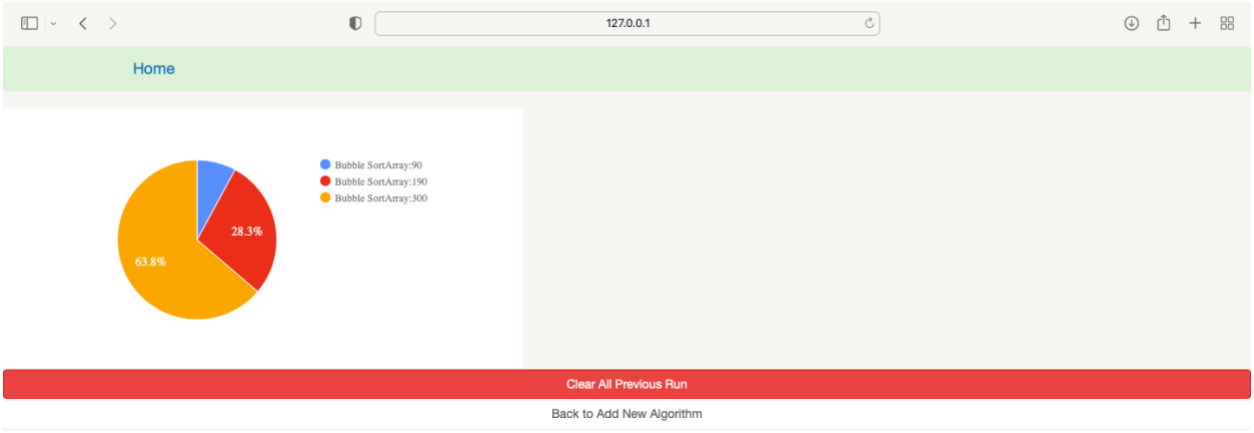
BUBBLE SORT

- Bubble Sort is a comparison-based algorithm
- This algorithm is based on comparison of i^{th} position element with the $(i+1)^{th}$ position element.
- If i^{th} element is greater than the $(i+1)^{th}$ position element then we swap those two elements.

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

Observation:

We can Observe that as length of Array is increasing time also increases.



MERGE SORT

- Merge Sort uses divide and conquer approach to sort the array.
- It is done by breaking down an array into several sub arrays until each sub array consists of a single element and then merging those sub arrays in a way that results into a sorted array at the end.
- It is an Out of Place algorithm.

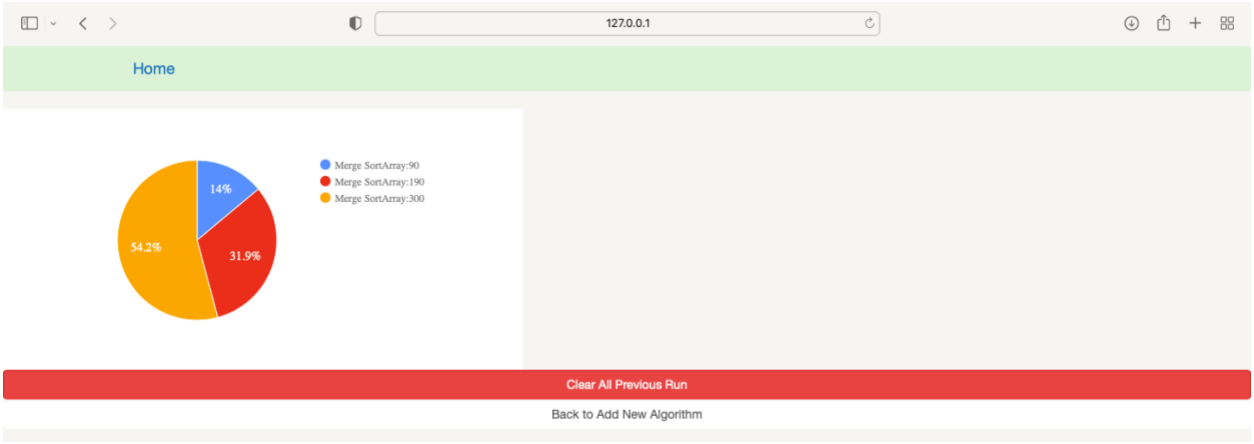
Time complexity of Merge Sort is O(nlogn) in all cases (i.e worst, average and best case).

Runtime can be improved by passing a small, sorted array.

Space Complexity: n

Observation:

We can Observe that as length of Array is increasing time also increases.



HEAPSORT

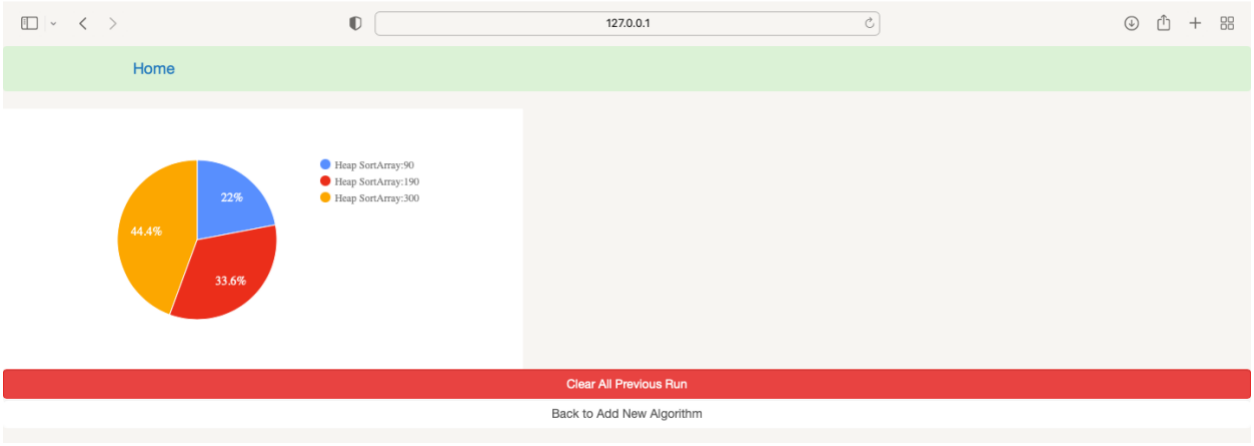
- Heap Sort is a binary search based sorting algorithm.
- Here, the maximum element will always be at the root in case of max-heaps.
- We can use optimised heap sort technique to improve time complexity.

Time complexity of Merge Sort is O(nlogn) in all cases (i.e worst, average and best case).

Space Complexity: O(1)

Observation:

We can Observe that as length of Array is increasing time also increases.



QUICKSORT

- Quicksort is also based on divide and conquer algorithm and is used for large data
- Here, pivot can be first element, last element, median or any random element.
- Pivot is used to create the partition in the array like:- the left side of pivot contains all the elements that are less than the pivot element and right side contains all elements greater than the pivot.
- We can improve the Runtime of Quick Sort by choosing either a random index for the pivot or we can choose median of first, middle and last element as pivot

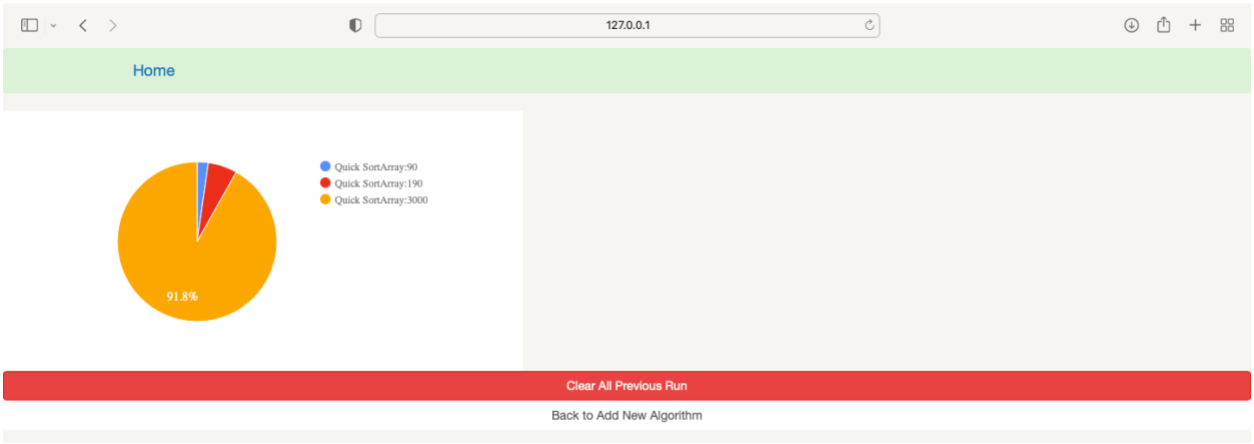
Worst Time complexity: O(n^2)

Average Time Complexity: O(nlogn).

Space Complexity: O(log n)

Observation:

We can Observe that as length of Array is increasing time also increases.



QUICKSORT-Using 3 Medians

Considering 3 medians i.e. first element last element and the median element in the array.

Swapping of numbers: median with first and first with last element.

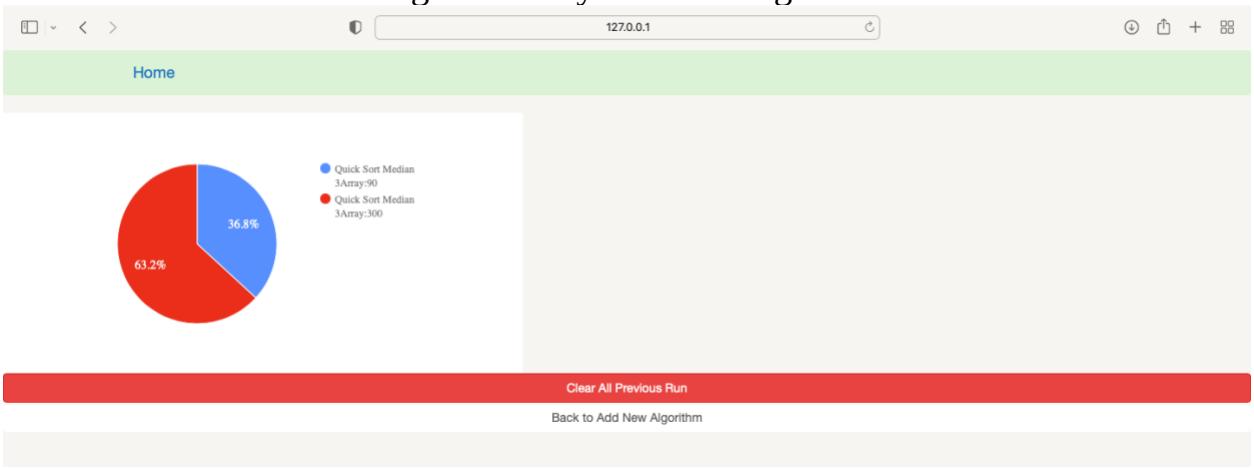
Then we need to find the element which is greater than the pivot element and then swap that element with the element back which is greater than pivot

Best and Average Time Complexity: Ω(nlogn) Worst Time complexity: O(n^2)

Space Complexity: O(nlogn)

Observation:

We can Observe that as length of Array is increasing time also increases.



COMPARISON OF ALGORITHMS

Now I have compared all algorithms giving same number of lengths as 2000 to each algorithm.

Observation:

From the above graph we can see that bubble sort takes maximum time to execute and quick sort is the quickest.