

Name: Akshaj Prakash Maldikar

Student ID: 1001679565

Problem statement:

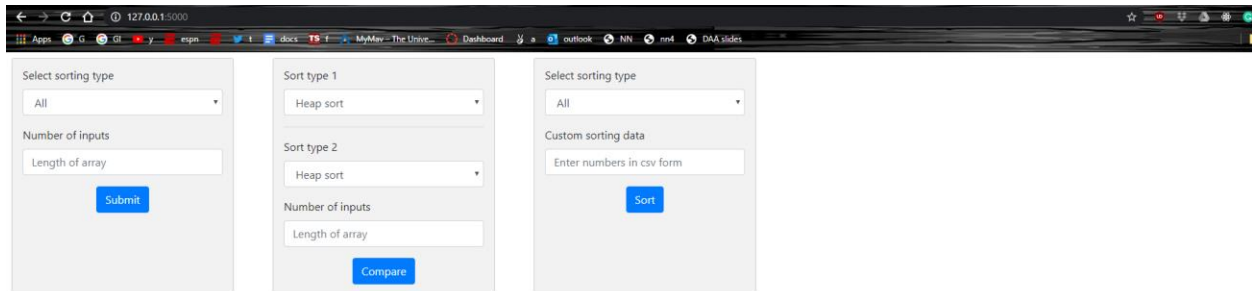
1. Implement and compare the following sorting algorithm :

- Mergesort
- Heapsort
- Quicksort (Regular quick sort* and quick sort using 3 medians)
- Insertion sort
- Selection sort
- Bubble sort

* For regular quick sort you can decide between choosing first, last or a random element as pivot. But you need to include both regular and 3 medians as separate algorithms.

About project:

I have created a basic website to show the time complexity of various graphs with respect to input size. For the front-end development, I have used HTML, CSS, JavaScript and Flask and for backend development, I have used Python. The initial view of the website is this:



First form: In the first form, the first input field is a dropdown which consists of all sorting types. The second input is length of array which is the size of array. After selecting appropriate inputs, the form values will be sent to backend and random numbers will be generated based on “Number of inputs” field.

Example: Selecting all sorting types and then 5000 number of inputs, the solution graph will be:

Select sorting type

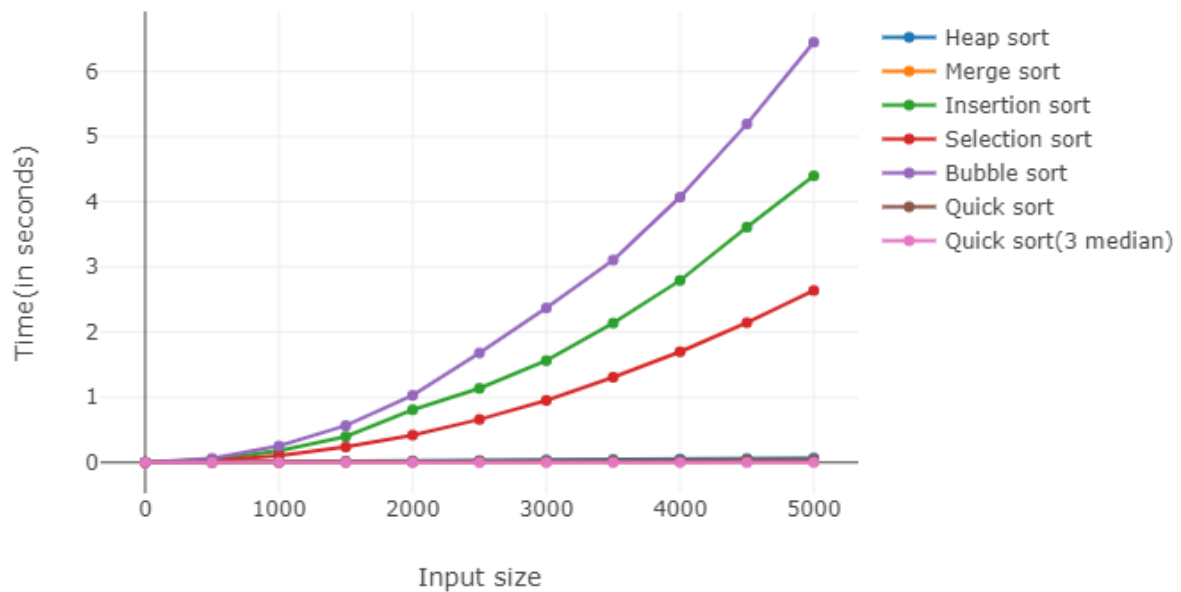
All ▼

Number of inputs

5000

Submit

All sorting techniques



The graph shows that as the various sorting techniques with respect to input size. The various worst case time complexities of the techniques are:

Sorting type	Worst case complexity
Heap sort	$O(n \log n)$
Merge sort	$O(n \log n)$
Insertion sort	$O(n^2)$
Selection sort	$O(n^2)$
Bubble sort	$O(n^2)$
Quick sort	$O(n^2)$
Quick sort (3 median)	$O(n \log n)$

Note: Although the worst case of quick sort is n^2 , it is considered as quicker based upon the positioning of the pivot element.

Second form: The second form consists of comparison between two sorting types. The form also provides an input for the length of array.

Example 1: Merge sort v/s Quick sort for 50k numbers.

Sort type 1

Merge sort

Sort type 2

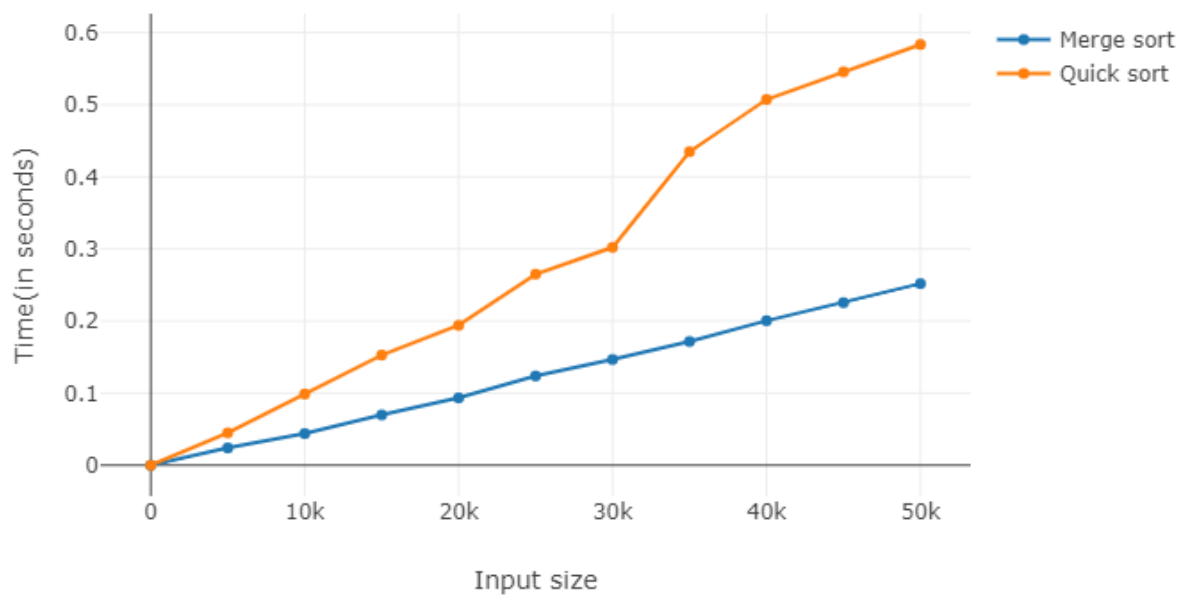
Quick sort

Number of inputs

50000

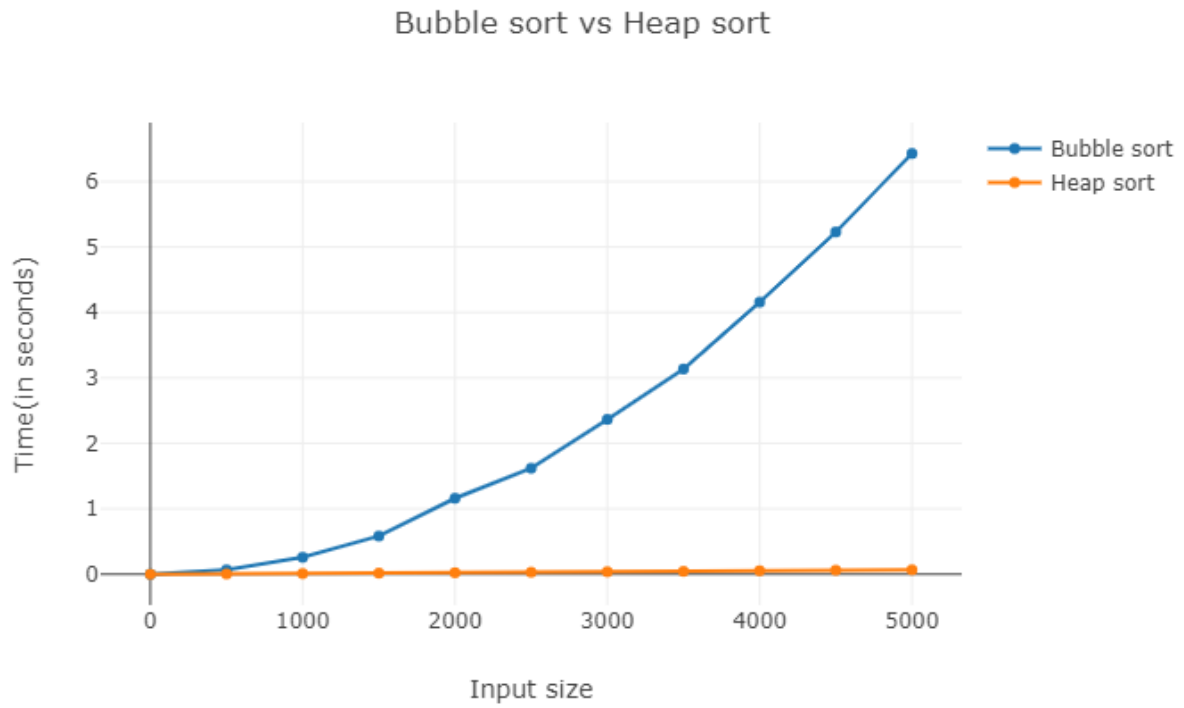
Compare

Merge sort vs Quick sort



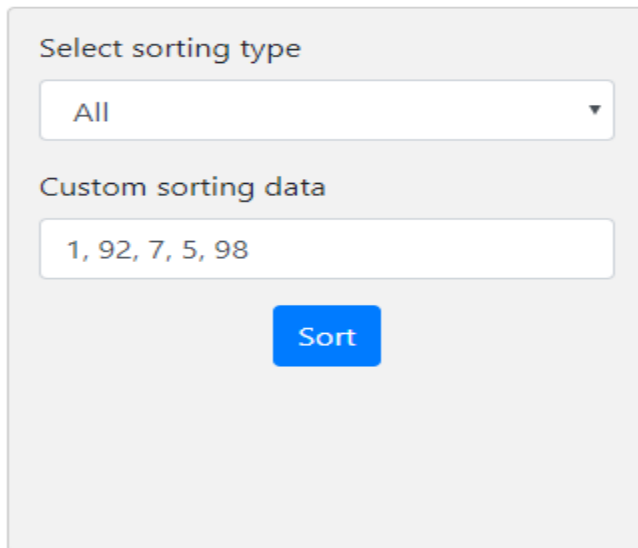
This shows that for larger input size, Merge sort is efficient with respect to time.

Example 2: Bubble sort v/s Heap sort for 5k numbers:

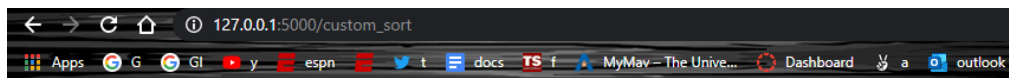


Third form: This form is for custom data input where you can add data in a csv format, select the sorting type and then we will get the sorted data and the time required for sorting.

Example 1: For all sorting types, input numbers = 1, 92, 7, 5, 98



The image shows a web form with a light gray background. At the top, it says "Select sorting type" in blue text. Below this is a white dropdown menu with "All" selected and a small downward arrow. Underneath the dropdown, it says "Custom sorting data" in blue text. Below that is a white text input field containing the numbers "1, 92, 7, 5, 98". At the bottom of the form is a blue button with the word "Sort" in white text.



Heap sort: [1, 5, 7, 92, 98], Time taken: 0.0

Merge sort: [1, 5, 7, 92, 98], Time taken: 0.0009975433349609375

Insertion sort: [1, 5, 7, 92, 98], Time taken: 0.0

Selection sort: [1, 5, 7, 92, 98], Time taken: 0.0

Bubble sort: [1, 5, 7, 92, 98], Time taken: 0.0

Quick sort: [1, 5, 7, 92, 98], Time taken: 0.0

Quick sort(3 median): [1, 5, 7, 92, 98], Time taken: 0.0

Note: As the size of input array is very small, the time taken is almost negligible