Assignment 1
Sorting Techniques

Sorting is one of the most fundamental algorithmic problems within computer science. It has been claimed that as many as 25% of all CPU cycles are spent sorting, which provides a lot of incentive for further study and optimization of sorting. In addition, there are many other tasks (searching, calculating the median, finding the mode, removing duplicates, etc.) that can be implemented much more efficiently once the data is sorted. The wide variety of algorithms gives us a lot of richness to explore, especially when considering the tradeoffs offered in terms of efficiency, operation mix, code complexity, best/worst case inputs, and so on.

1. Lab Goal

- The goal of this lab is to understand different running times for each algorithm, best and worst-case running times.
- Become familiar with the binary heap data structure as well as different sorting techniques.

2. Sorting Techniques

You are required to implement:

- three O (n^2) sorting algorithms: Selection Sort, Bubble Sort, and Insertion Sort.
- three O (n log n) sorting algorithms: Merge Sort, Heap Sort and Quick sort algorithm in the average case.

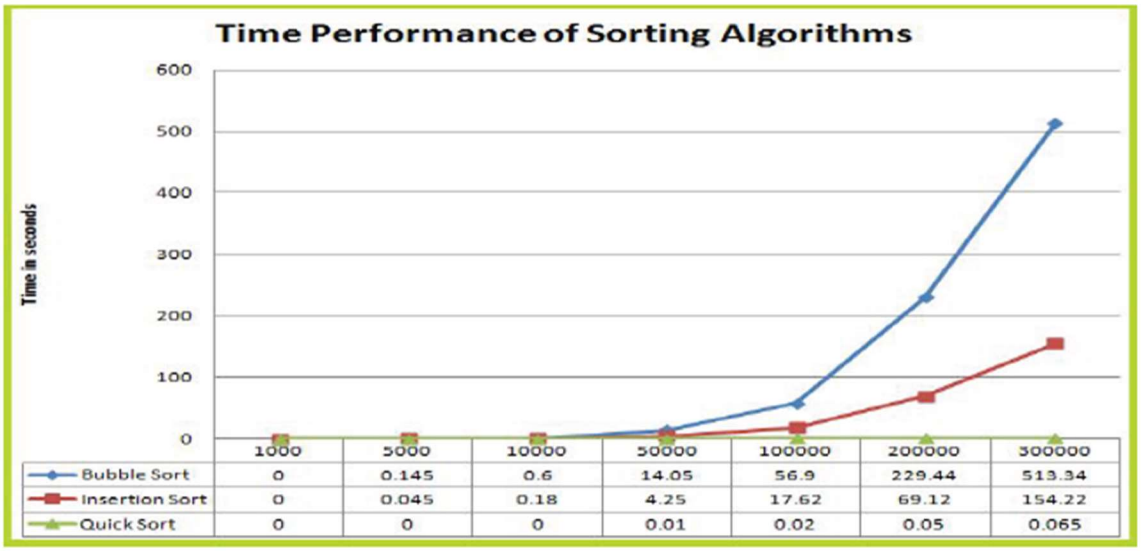
Now you have implemented six sorting techniques: three of them are O (n^2) and the other three are O (n log n).

You are required to compare the running time performance of your algorithms against each other.

To test your implementation and analyze the running time performance, generate a dataset of random numbers, and plot the relationship between the execution time of the sorting algorithm versus the input size.

3. Report

A graph is required between **Time (milliseconds)** vs **Array Size** for different sorting algorithms, generate random arrays of different sizes and calculate the time required to sort it using the algorithms you implemented above and plot the results to a graph as what is shown in the image, you can use Excel sheet to achieve the job.



	1000	5000	10000	50000	100000	200000	300000
Bubble Sort	0	0.145	0.6	14.05	56.9	229.44	513.34
Insertion Sort	0	0.045	0.18	4.25	17.62	69.12	154.22
Quick Sort	0	0	0	0.01	0.02	0.05	0.065

4. Deliverables

- Part 1:
  - Three O (n^2) and three O (n log n) sorting algorithms.
  - Generate random integer arrays of sizes (10, 100, 1000, 10000, 100000) then sort each of them using the 6 different sorting techniques and compute the running time for each algorithm.
  - **Hints**:
    - Do not include random arrays generation time when measuring each algorithm running time, and make sure each algorithm runs on the randomly generated arrays.
    - When you call other sorting, technique make sure that you call it with original array not the sorted one.
- Part 2:
  - Report that contains the following
    - Description of the program
    - Pseudo code for each algorithm
    - Sample Runs
    - Graph described in part 3

5. References

Some references that can help you understand sorting and its application.

- Link: Algorithms and Running times.
- Link: Visualization for different sorting algorithms.
- Link: Measure time taken by function in C.

6. Notes

- Implement your algorithms using (Java, C/C++, or Python).
- You should work in groups **of 3 members.**

Good Luck