Project 1: Program Performance Measurement

Problem Statement:

In the algorithmic study, we often use the Big-O notation to describe the growth of each computer algorithm. In this project, we would like to take a totally different approach. We would like to measure the exact execution time by measurement.

Therefore, you will be required to measure the starting time and the ending time of a function call. Each function will be an example sorting algorithm implementation.

For example,

```
random_list_10_elements = generate_random(10)

start_time = measure_of_system_time()

sorted_list_10_elements = selection_sort(random_list_10_elements)

end_time = measure_of_system_time()

time_selection_sort_10_elements = end_time – start_time
```

This is the pseudo code for the performance measurement of selection sort over 10 elements.

To generate the 10-elements, please use random number generator which can generate number in the range of ($0 <= x < 10^5$), for each x element generated.

Please measure each sorting algorithm for **10, 100, 1000, 10000, 100000**. At least these 5 cases.

All sorting results should be in ascending order. Duplicated elements are allowed.

Each sorting test case must have newly generated random elements.

Please measure for the following 5 sorting algorithms:

Selection Sort, Insertion Sort, Bubble Sort, Merge Sort, Quick Sort.

So, totally there are 25 test cases to be measured.

All sorting algorithms must be coded by the student. **DO NOT COPY** sorting program from other places or do not use sorting function from the library. Violation of this rule will get 0 point on each sorting algorithm grading component. If you copy or use library code, you will get 0 points for this project. No matter how much points you get on comments or documentation. This is considered to be an incident of plagiarism.

After getting all the results, please briefly write down your results and creating a line chart which can show the trending of the performance for each sorting algorithm.

Have fun!!!

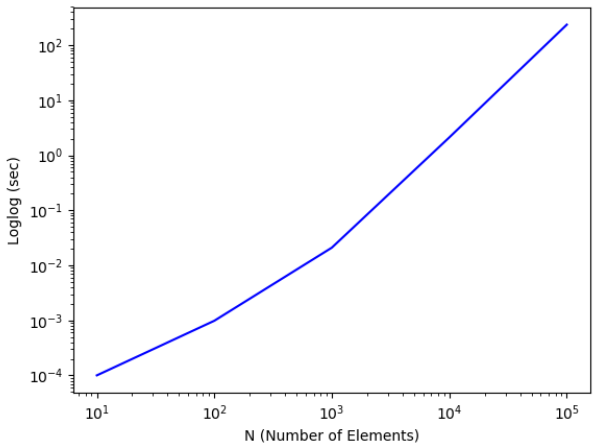
Example Program:

Measurement of Program Execution time:

```
import time
start = time.time()
for i in range(10000): print("hello")
end = time.time()
print(end - start)

Results:
… (too many hello omitted)
hello
hello
hello
hello
hello
hello
hello
hello
0.03093862533569336
```

Expected Results:



Use log-log chart can show the grow better than linear scale in this case. This example chart only show the results for the selection sort. You must complete all the 5 sorts.

Grading Rubric:

Components	Points Possible	Points Earned
Comments include name, date, purpose of the program	20	
Correctly implemented the time measurement feature	20	
Correctly coded selection sort	20	
Correctly coded insertion sort	20	
Correctly coded bubble sort	20	
Correctly coded merge sort	20	
Correctly coded quick sort	20	
Correctly collected the performance data	20	
Fully documented report	20	
Performance Chart and Growth rate analysis	20	
Total (Extra-points may be given to students who coded each sorting algorithm by using recursion)	200	