Report

Park Jun Hyung

**Lab 3: Comparison of Sorting Algorithm**

**Goal:** Compare the start time and end time of different sorting algorithms and prove the given sorting algorithm’s time complexity.

**Given Sorting algorithms**

* Bubble Sort
* Selection Sort
* Insertion Sort

**Pre-Condition**

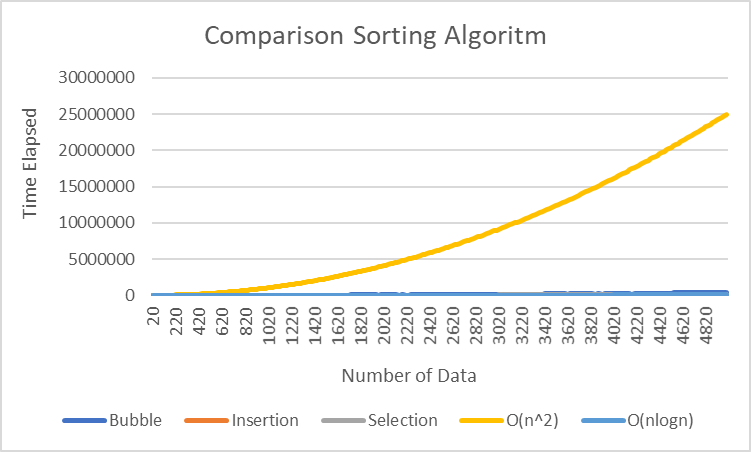
* Random numbers
* Start from 20 to 5000 (include)

**Data**

Provided in Compare.txt I submitted.

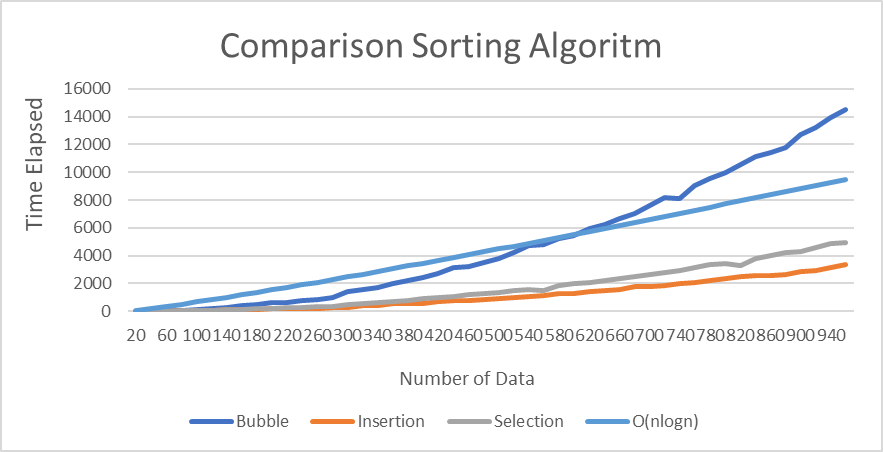
1. **Get Upper bound and Lower bound**

**Graph 1: Comparison among sorting algorithm with O(n^2) and O(nlogn)**



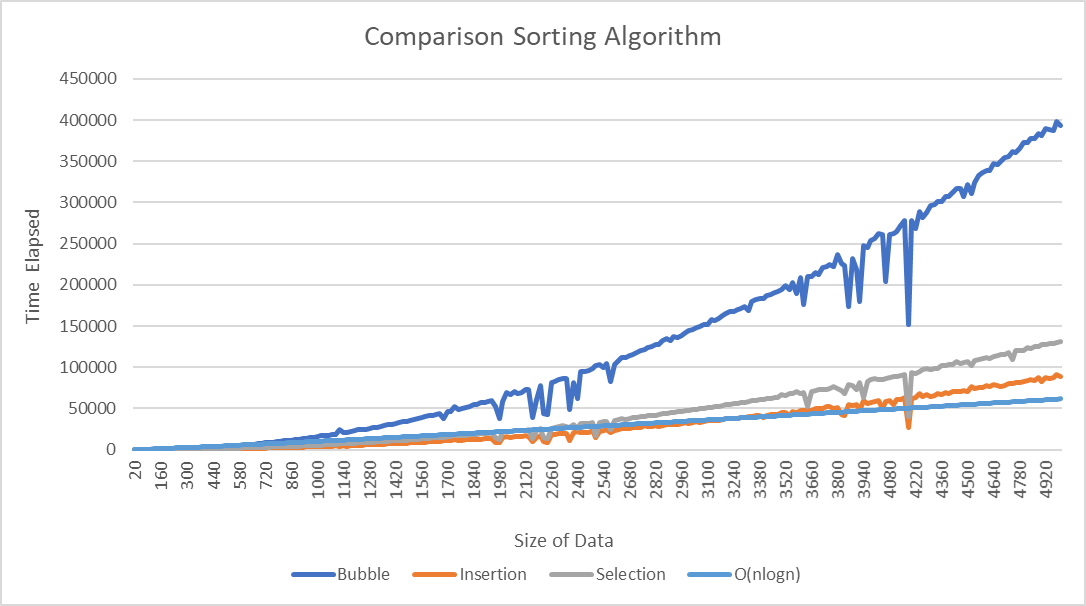
It is clear that the orange line of graph, which represent O(n^2) is upper bounded. Because of Dominant factor O(n^2), it is hard to see other lines of graphs.

**Graph 2: Comparing Sorting Algorithm with O(nlog(n)) data number 20 to 940  
(N^2 had been removed to see clear other lines)**



This graph represents the number of data from 20 to 940. Interestingly, O(nlogn) takes more time than bubble sort till data between [20, 620) and after 620 it is slower. The Insertion and Selection sort is faster than O(nlogn) and bubble sort.

**Graph 3: Comparing Sorting Algorithm with O(nlog(n))  
(N^2 had been removed to see clear other line)**



Unlike Graph2, as number of data is increasing, O(nlog(n)) is the fastest comparing with Bubble, Selection, and Insertion sorting algorithm.

As proved by Graph1, the upper bound is O(n^2) and by proving in Graph3, the lower bound is O(nlog(n)).

There is no reason to make a additional graph that contains n since, n is smaller than nlog(n).

1. **Prove Time complexity of Sorting Algorithm**

To prove the sorting algorithm, I have to divide my sorting algorithm to both n^2(upper bound) and nlog(n) (lower bound).

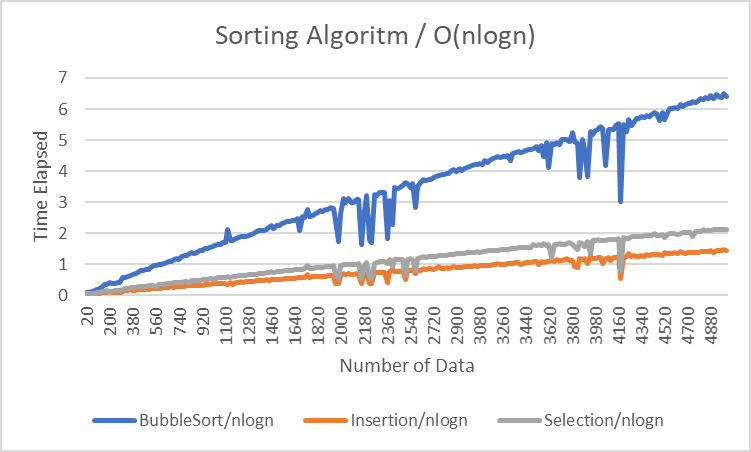
**Conditions 1:**  
If my sorting algorithm has time complexity of O(n^2) or similar behavior to O(n^2), the graph of dividing with n^2 will give me constant.

Since (n^2 / n^2) is constant.

**Condition 2:**If my sorting algorithm has time complexity of O(nlog(n)) or similar behavior to O(nlog(n)), the graph of diving with nlog(n) will give me constant.

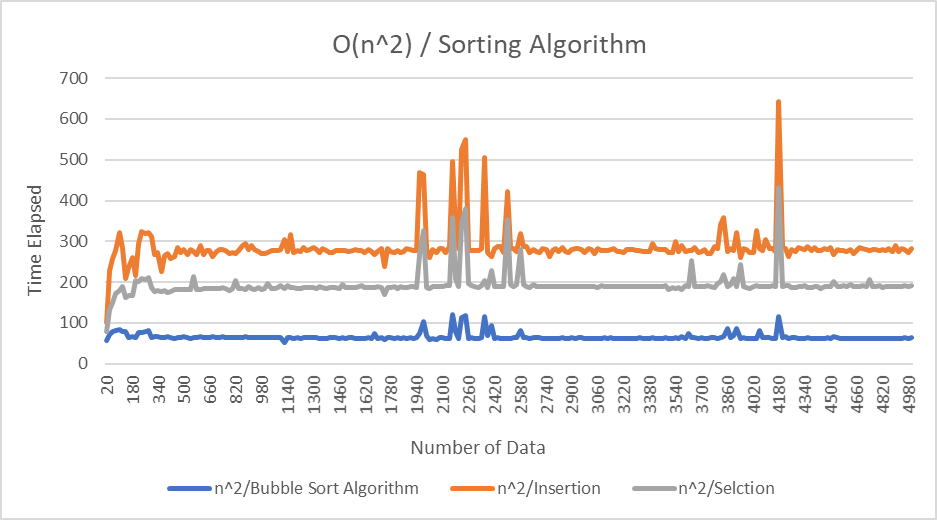
Since (nlog(n))/(nlog(n)) is constant.

**Graph 4: Dividing Sorting Algorithm with O(nlog(n))**



The result of graph proves that the result is not constant, since they are the time elapsed is increasing as the numbers of data is increasing.

**Graph 5: Dividing Sorting Algorithm with O(n^2)**



The result of the graph5 proves that the result of division is constant, which means those three-sorting algorithms are O(n^2).

**Conclusion**:

By getting upper bound and lower bound, which are O(n^2) and O(nlog(n)) since the time complexity of the three algorithms are between those two. Dividing the sorting algorithm’s time elapsed with those bounds and if the result is constant (n^2/an^2 is constant), it means that the algorithm’s time complexity is same as the bound (either low or high). In this given sorting algorithm, the time complexity is O(n^2).