Name: Jennifer Phuc Tran
Date: 10/22/2021
Exercise 1 Report

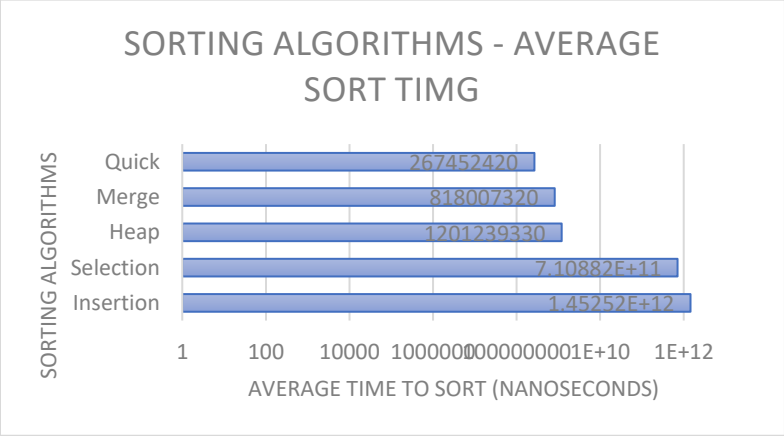
EXERCISE 2 REPORT

Write a one-page report with your findings, including the theoretical complexity analysis.

Findings:

This table below illustrates how much time it takes on average to sort the words in *pride-and-prejudice.txt* file using Insertion, Selection, Heap, Merge and Quick sorting algorithms.

	Insertion	Selection	Heap	Merge	Quick
Average time (ns)	1.5E+12	7.11E+11	1.2E+09	8E+08	2.7E+08
Standard Deviation	4.7E+10	2.16E+10	3.6E+07	6E+07	1.5E+07



Graph of sorting algorithms vs. sorting time in nanoseconds (Log scale)

Complexity Analysis:

Insertion Sort: Time complexity O(n^2). For this algorithm, we need to go through the array, compare the current value to the ones before it, and swap them if the current value is smaller.

Selection Sort: Time complexity O(n^2). This algorithm sorts an array by repeatedly finding the minimum element from unsorted subarray and putting it at the beginning, this takes O(n). Then compare that element to the sorted subarray to see where to place it, this takes another O(n).

Heap Sort: Time complexity O(logn). This is like Binary Tree data structure. We need to visit the nodes, place the larger value than the parent to the left child and smaller value to the parent to the right child.

Merge Sort: Time complexity O(nlogn). This algorithm divides the array into 2 subarrays, then recursively sort arrays until done.

Quick Sort: Time complexity O(nlogn). This algorithm chooses a pivot and split the array into 2 subarrays with smaller values than the pivot in 1 subarrays and bigger values than pivot in the other subarray.

Summary:

The data collected matched with the theoretical analysis in most parts.

Insertion sort = Selection sort *O(n^2)* > Merge = Quick *O(nlogn)* > Heap *O(logn)*

However, for the Quick sort, since the text file is large and has many repeated words, I modified my algorithm to store all the same repeated words in one array called “middle”. That is why the Quick sort seemed to run in shorter time than the Heap, when we expected the Heap sort to run the fastest.