Homework Assignment 0

Problem Size	Insertion Sort	Merge Sort	std::sort
10	1.67e-07	1.5e-06	4.16e-07
100	3.292e-06	1.0167e-05	1.75e-06
1000	0.000247958	9.6083e-05	1.2834e-05
10000	0.0304359	0.00106429	0.000148375
100000	2.10267	0.0119728	0.00147625
1000000	223.436	0.131316	0.0160793
10000000	-- Took too long --	1.45678	0.175029
100000000	-- Took too long --	16.2793	1.90853

Questions:

a. For each of the three sort methods, what pattern do you observe in the execution time as the problem size is increased (in other words, if you look at each column in the table above, do you see any patterns in execution time as related to the problem size)? Did you expect to see this pattern? If so, why? If not, then what did you expect to see?

Insertion sort: Time grows very fast as size increases (roughly quadratic). Small inputs are fine; large ones take up a lot of time (as can be seen). This is expected, since insertion sort is $O(n^2)$.

Merge sort: Time increases smoothly and much slower (near nlogn). Expected, matching its theoretical complexity.

std::sort: Grows similarly to merge sort but consistently faster. Expected, since it's a highly optimized O(nlogn) algorithm.

b. How does the execution time of insertion sort compare with execution time of merge sort for different problem sizes? Explain the difference in execution time.

For insertion sort, we observed that every time we multiply 10 to the problem size, we multiply 100 to the time taken. Which is a clear indication of $O(n^2)$ time complexity. However, for Merge sort, we see a similar 10x multiple for time, indication an O(nlogn) complexity. We can clearly see that as size grows, Merge sort is clearly faster.

c. How does the performance of your insertion sort and merge sort implementations compare with the performance of the std::sort? Explain any similarities or differences you observe.

std::sort outperforms both the implementations across all problem sizes. It's slightly slower than insertion sort for tiny inputs but significantly faster at larger sizes. Compared to merge sort, std::sort is about 10x faster, likely due to highly optimized implementation and an adaptive algorithm. It is clearly close to O(nlogn) but faster than Merge sort (by almost 10x) starting even from the base cases because of it's better and faster approach.