# Execution time for different problem sizes

|  |  |  |  |
| --- | --- | --- | --- |
| Problem Size | Merge Sort | Insertion Sort | std::sort |
| 10 | 2.417e-06 | 2.5e-07 | 4.17e-07 |
| 100 | 3e-05 | 7.959e-06 | 2.75e-06 |
| 1,000 | 0.000381291 | 0.000633833 | 3.15e-05 |
| 10,000 | 0.00480604 | 0.0640655 | 0.000420833 |
| 100,000 | 0.0584811 | 6.34567 | 0.00646434 |
| 1,000,000 | 0.676747 | 558.613 | 0.0563432 |
| 10,000,000 | 7.76971 | 6600 (est) | 0.754653 |
| 100,000,000 | 113.456 | - | 8.64566 |

# Questions

1. I observed that the execution time increases overall as the problem size increases. I did expect to see this pattern since the problem size is increasing. But the way it was increasing for different things was interesting because it was much slower for insertion sort compared to merge sort or std::sort.
2. Insertion Sort was faster than merge sort for small datasets but as soon as the data got bigger there was significant difference in their run times. It is possibly because of the different time complexities they have O(N^2) vs O(NlogN). Insertion sort is much slower and thus I was not able to find the last execution time for it.
3. Honestly, I did expect to see much difference between merge sort and std:sort since I assumed both of them to be O(NlogN) but I guess it is an optimization problem. For small data sets it was not a big difference but as the time increases, the difference increasingly gets bigger and bigger.

/\*

Name: Jay K Roy

Email: jroy4@crimson.ua.edu

Course Section: Fall 2024 CS 201

Homework #: 0

\*/