Program Structures and Algorithms

Spring 2023(SEC – 1)

NAME: Aishwarya Venkatesan

NUID: 001569213

**Task:** To determine--for sorting algorithms--what is the best predictor of total execution time: comparisons, swaps/copies, hits (array accesses), or something else.

You will run the benchmarks for merge sort, (dual-pivot) quick sort, and heap sort. You will sort randomly generated arrays of between 10,000 and 256,000 elements (doubling the size each time). If you use the *SortBenchmark*, as I expect, the number of runs is chosen for you. So, you can ignore the instructions about setting the number of runs.

For each experiment (a sort method of a given size), you will run it twice: once for the instrumentation, once (without instrumentation) for the timing.

**Relationship Conclusion:**

Based on the graphs, we can see that for heap sort, the execution time can be best predicted using swaps and comparisons. As the size of the array increases, the number of comparisons increases depending on the difference in array size.

For merge sort, as size increases, the number of copies increases, which is the best predictor for merge sort. This can be seen in the graphs from the next section.

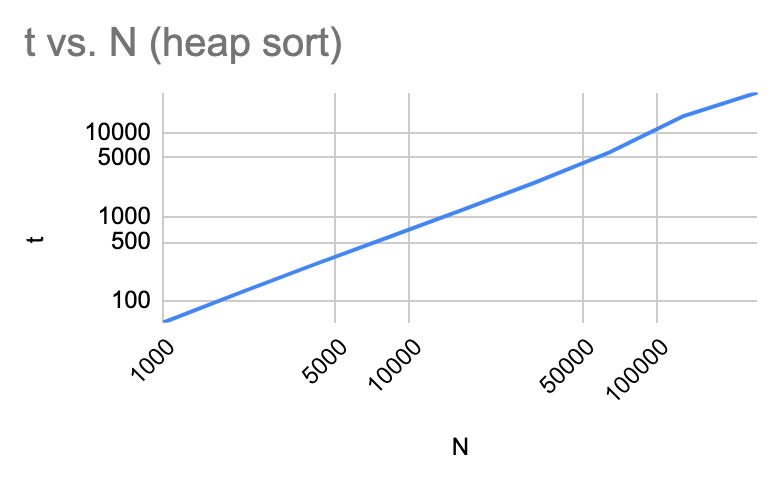
For quick sort dual pivot, execution time can be best predicted using the number of compares for a given array size.

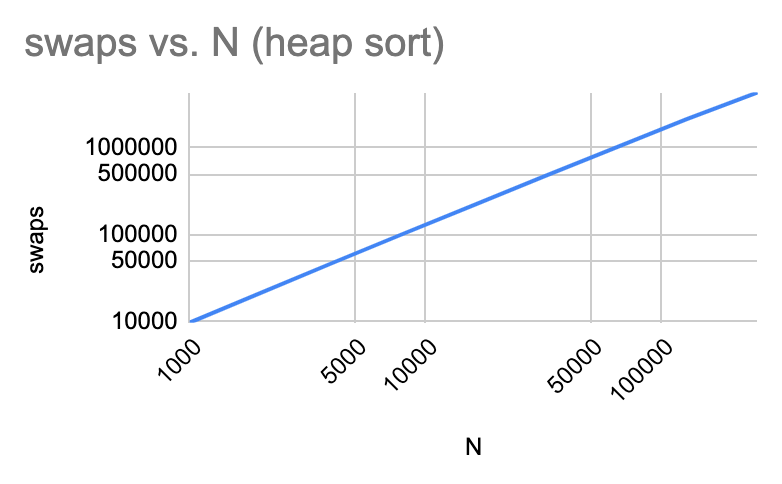
**Evidence to support that conclusion:**

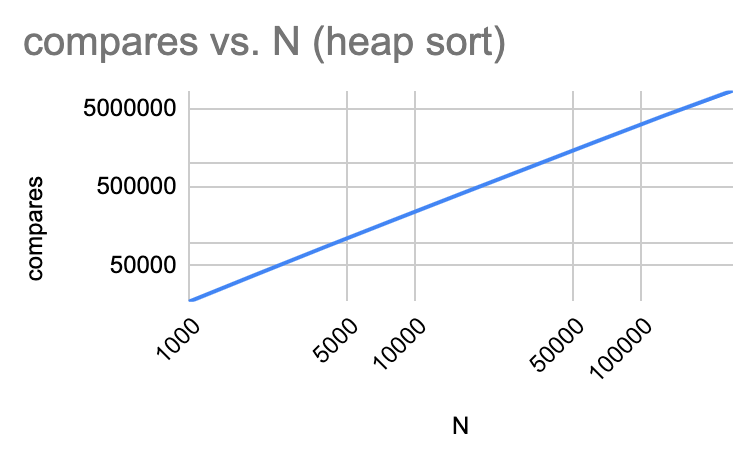
Tabular column for heap sort timings,swaps and compares, quick sort dual pivot timing and compares, merge sort timings and copies is given in the spreadsheet file

**Graphical Representation:**

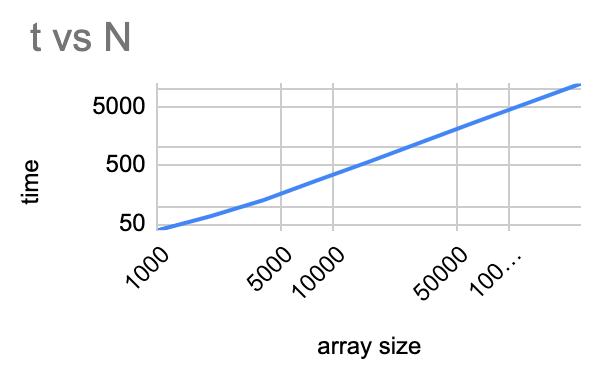
For each sort, one (log/log) graph represents the relationship between execution time and size of the array. Another (log/log) graph represents the relationship between execution time and predictor.

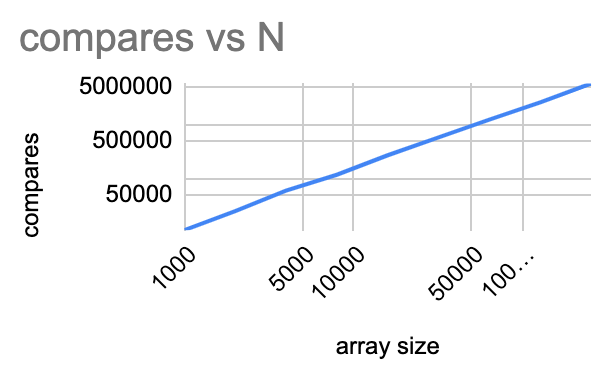
Heap sort:

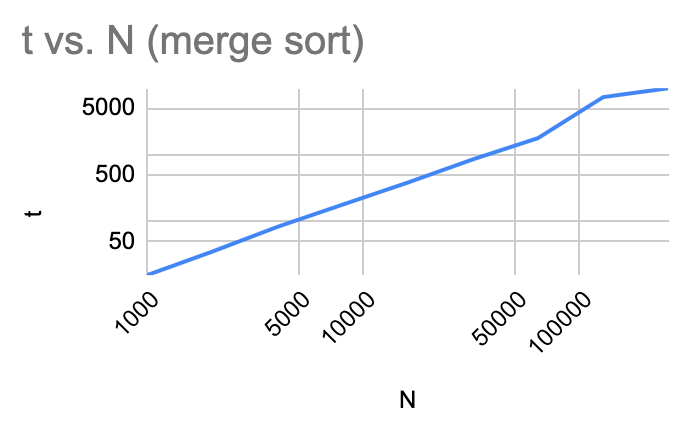


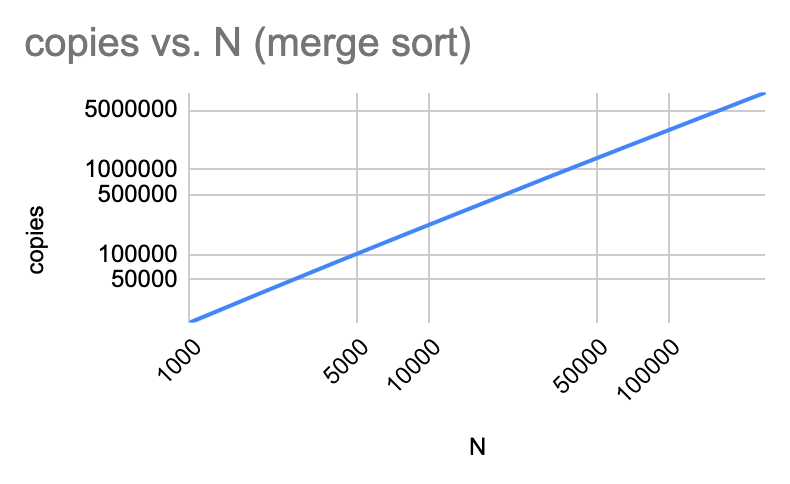


Quick Sort:





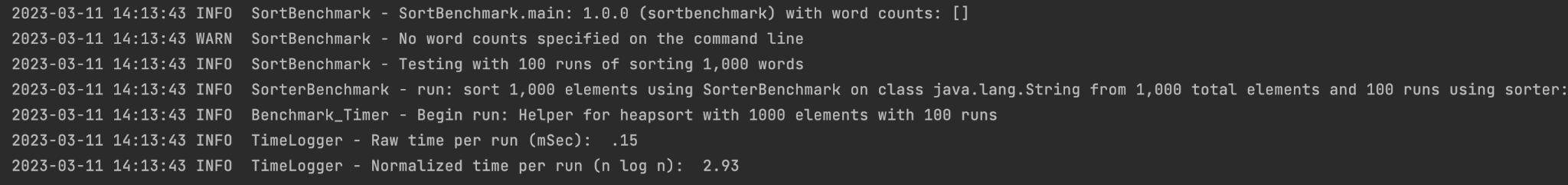
Merge Sort:

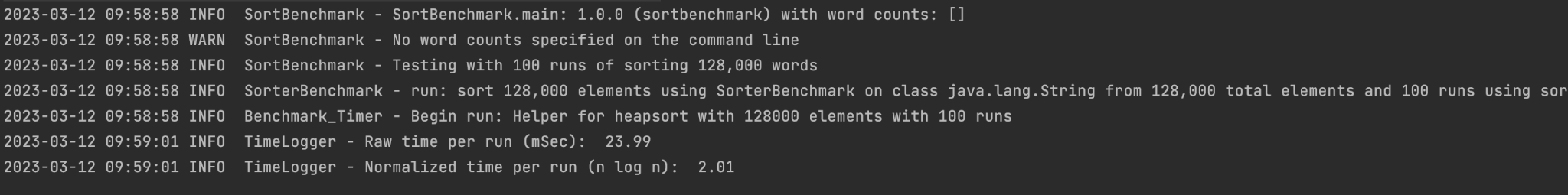


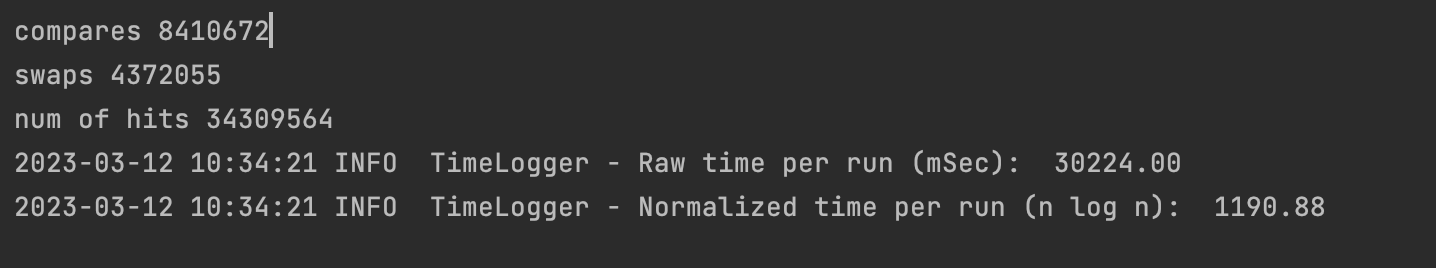
**Console output:**

**Heap Sort:**

Value for array size 1000

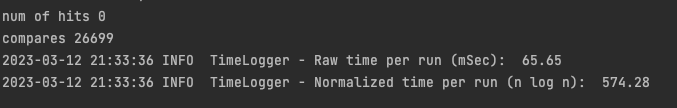
****

Value for array size 128000

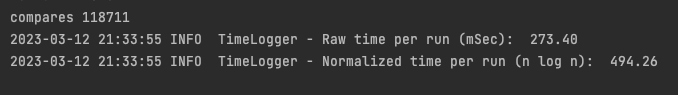
Value for array size 256000

Quick sort:

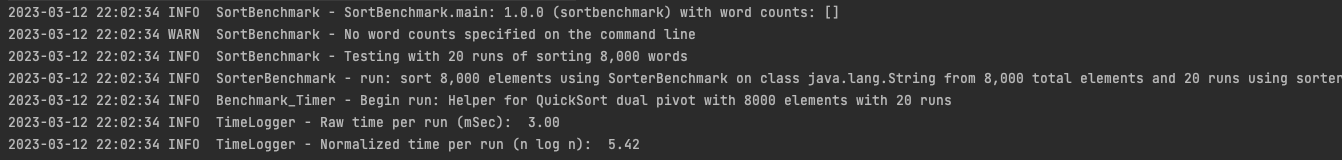
Value of array size 2000



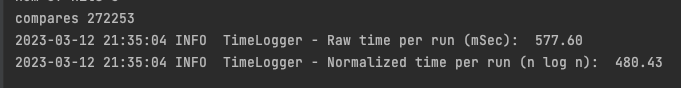
Value of array size 8000



Value of array size 8000 for instrumentation = false

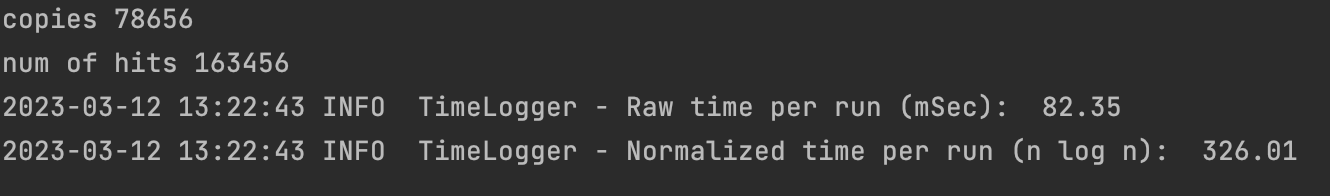


Value of array size 32000

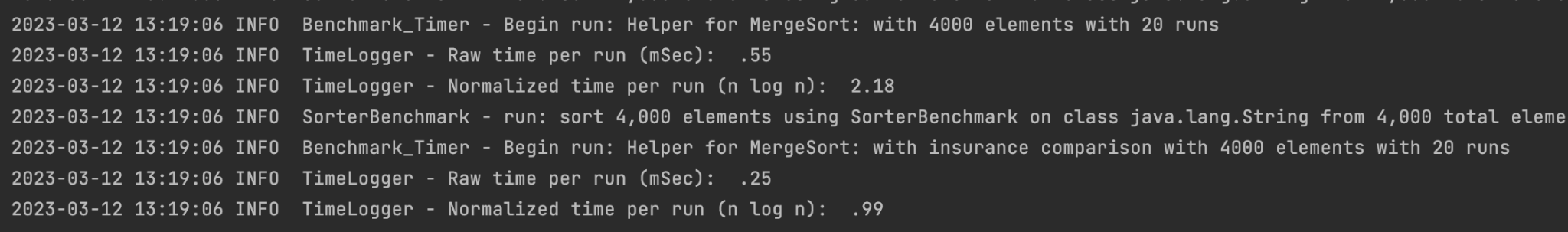


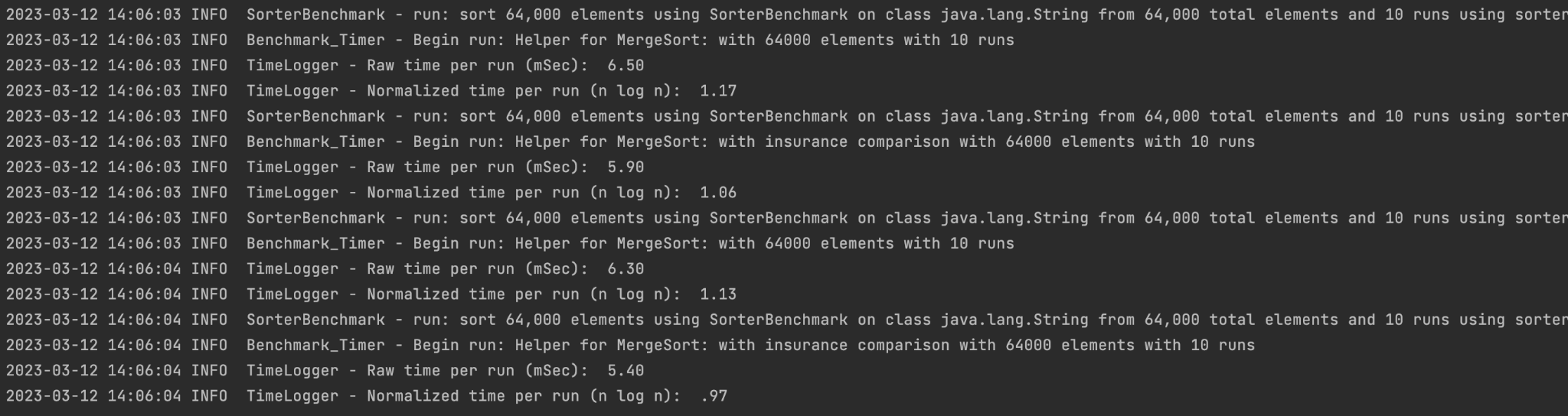
Merge Sort

Value of array size 4000



Value of array size 4000 and instrumentation is false



Value of array size 64000 and instrumentation is false

Value of array size 64000