

Program Structures and Algorithms
Spring 2024

NAME: Ashish Nevan Gade

NUID: 002889005

GITHUB LINK: <https://github.com/AshishNevan/INFO6205>

Task: Determine the best predictor of runtime for sorting algorithms

Relationship Conclusion:

Hits is the best predictor for a sorting algorithm's runtime

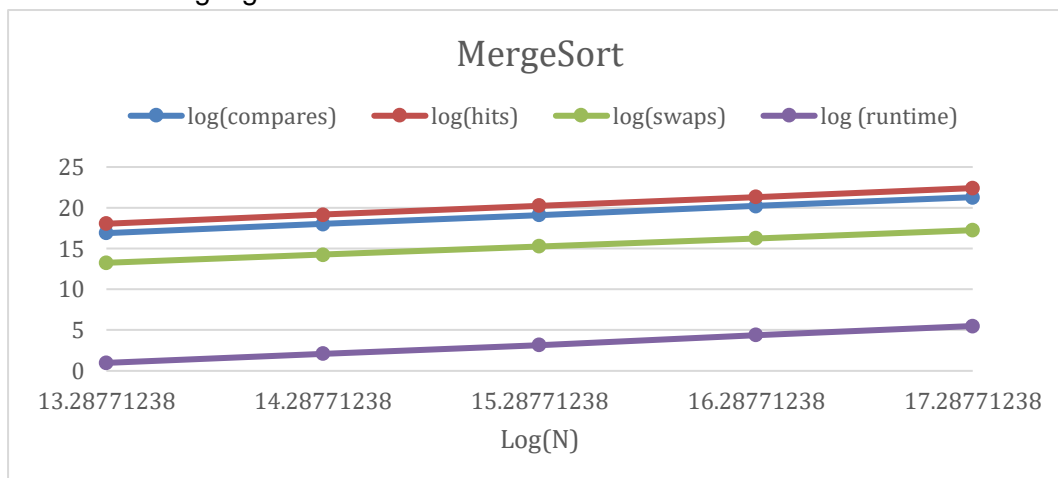
Evidence to support that conclusion:

N	Sorter	raw runtime (ms)	mean hits	copies	mean swaps	mean compares	log(N)	log(compares)	log(hits)	log(swaps)	log(runtime)
10K	MergeSort	1.97	269,783	110,000	9,761	121,503	13.28771238	16.89063241	18.04143992	13.25281324	0.97819563
20K	MergeSort	4.3	579,561	240,000	19,520	263,003	14.28771238	18.00471973	19.14460099	14.25266543	2.10433666
40K	MergeSort	9.01	1,239,131	520,000	39,043	566,008	15.28771238	19.11046292	20.24089729	15.25277629	3.17152711
80K	MergeSort	21.18	2,638,185	1,120,000	78,061	1,211,991	16.28771238	20.20894756	21.3311143	16.25231432	4.40463068
160K	MergeSort	45.19	5,596,409	2,400,000	156,130	2,583,994	17.28771238	21.30117129	22.41606997	17.25238825	5.49793165

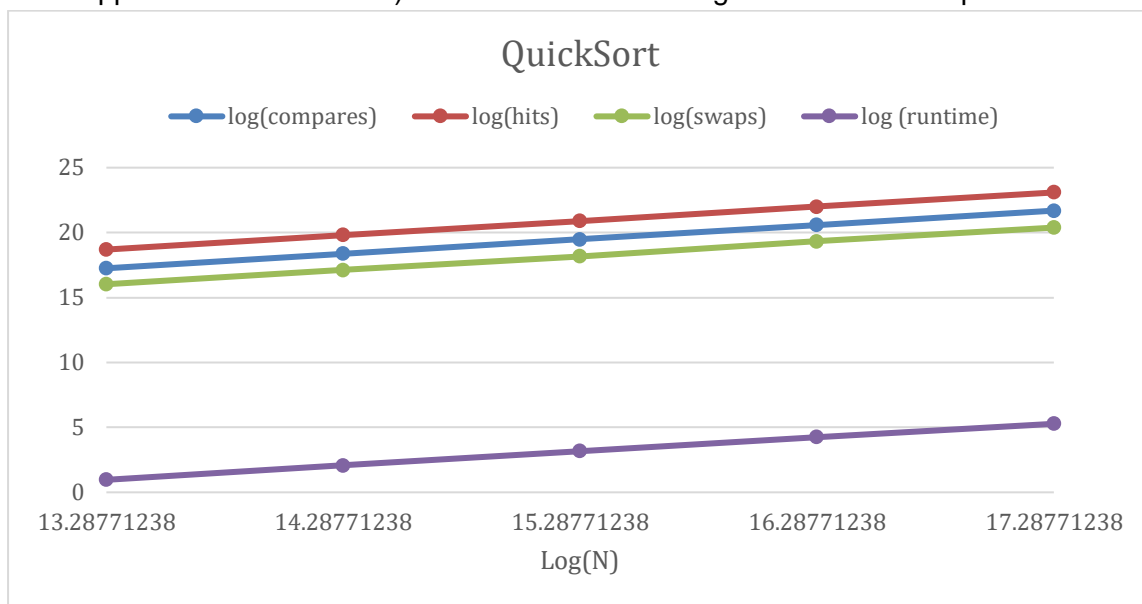
N	Sorter	raw runtime (ms)	mean hits	copies	mean swaps	mean compares	log(N)	log(compares)	log(hits)	log(swaps)	log(runtime)
10K	QuickSort(DP)	1.95	423,853	0	66,540	155,918	13.28771238	17.25042796	18.69320447	16.02193425	0.96347412
20K	QuickSort(DP)	4.2	915,593	0	142,489	340,550	14.28771238	18.37750711	19.80434691	17.12049102	2.07038933
40K	QuickSort(DP)	9.01	1,946,601	0	298,033	738,603	15.28771238	19.4944396	20.89252577	18.18511256	3.17152711
80K	QuickSort(DP)	19.1	4,212,998	0	654,778	1,572,033	16.28771238	20.58420007	22.0064158	19.32064632	4.25550073
160K	QuickSort(DP)	38.76	8,933,612	0	1,368,975	3,389,239	17.28771238	21.69252994	23.09081217	20.38466467	5.27649667

N	Sorter	raw runtime (ms)	mean hits	copies	mean swaps	mean compares	log(N)	log(compares)	log(hits)	log(swaps)	log(runtime)
10K	HeapSort	2.68	967,556	0	124,203	235,371	13.28771238	17.84457705	19.88398564	16.9223405	1.422233
20K	HeapSort	5.75	2,095,056	0	268,396	510,736	14.28771238	18.96221823	20.99855738	18.03400365	2.52356196
40K	HeapSort	12.53	4,510,173	0	576,795	1,101,497	15.28771238	20.07103414	22.10475134	19.13769913	3.64731451
80K	HeapSort	30.16	9,660,290	0	1,233,593	2,362,959	16.28771238	21.17216317	23.20363507	20.23443505	4.91456452
160K	HeapSort	60.66	20,600,649	0	2,627,179	5,045,966	17.28771238	22.26669905	24.29618645	21.32508307	5.92267359

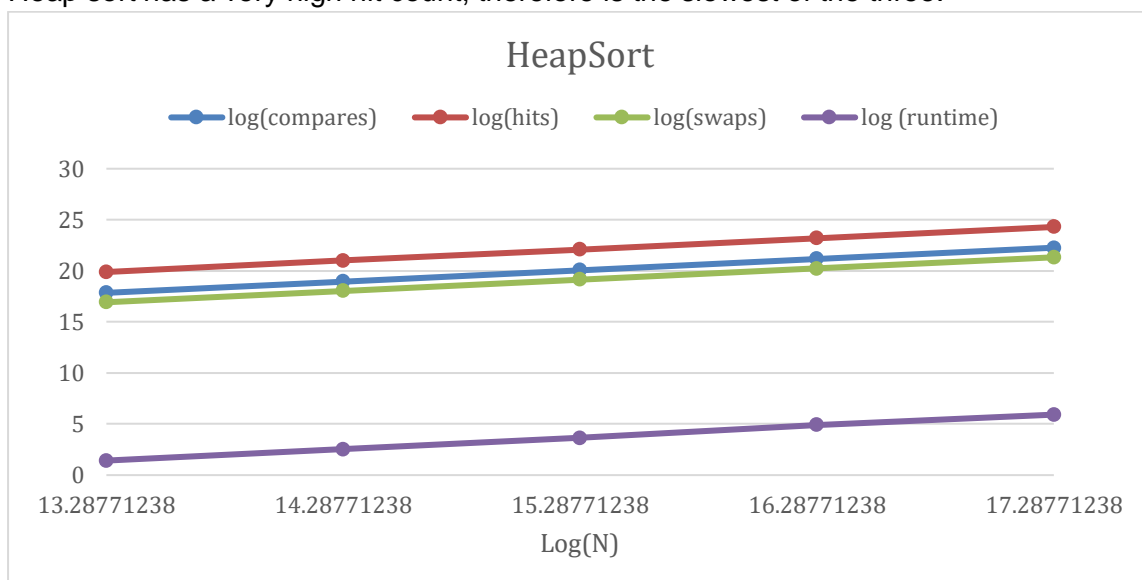
The following graphs below are plotted using various data observed during execution of 3 different sorting algorithms.



Merge sort has a high number of copies involving slower memory access times, (writing to RAM opposed to faster cache) which contribute to a higher runtime than quick sort.



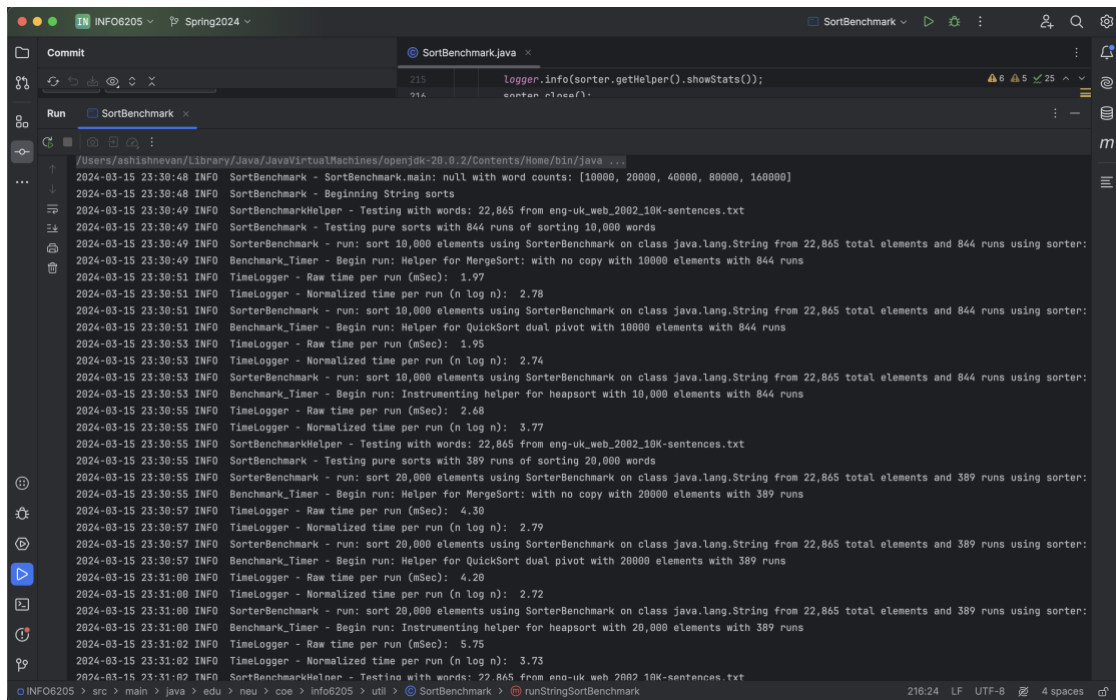
Heap sort has a very high hit count, therefore is the slowest of the three.



From the graphs, we can observe high correlation between hits, compares and swaps with runtime. However, array accesses or hits are involved for swaps as well as compares. Hits are directly proportional to the time spent by the algorithm performing slower I/O tasks (reading memory). Therefore, it is safe to conclude that array accesses or hits are the best predictor of a sorting algorithm's runtime.

Console output:

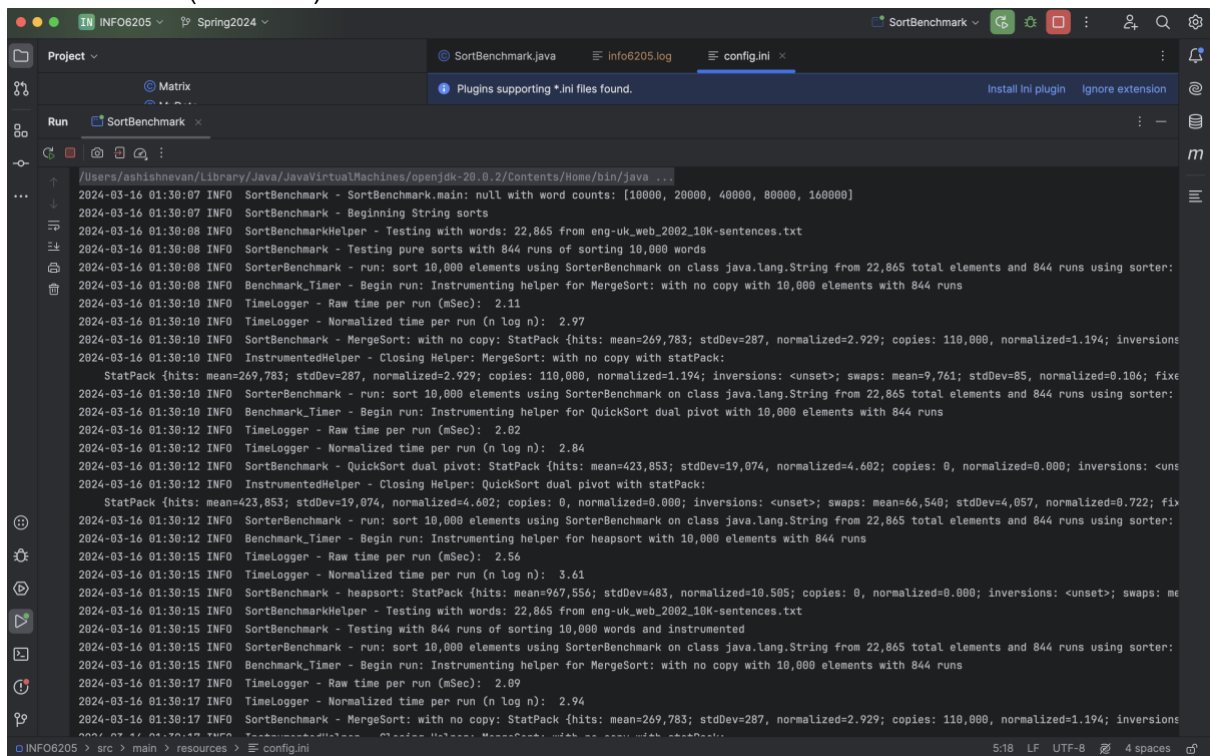
Non-instrumented (runtimes):



```
215 logger.info(sorter.getHelper().showStats());
216 sorter.run();

2024-03-15 23:30:48 INFO SortBenchmark - SortBenchmark.main: null with word counts: [10000, 20000, 40000, 80000, 160000]
2024-03-15 23:30:48 INFO SortBenchmark - Beginning String sorts
2024-03-15 23:30:49 INFO SortBenchmarkHelper - Testing with words: 22,865 from eng-uk_web_2002_10K-sentences.txt
2024-03-15 23:30:49 INFO SortBenchmark - Testing pure sorts with 844 runs of sorting 10,000 words
2024-03-15 23:30:49 INFO SorterBenchmark - run: sort 10,000 elements using SorterBenchmark on class java.lang.String from 22,865 total elements and 844 runs using sorter:
2024-03-15 23:30:49 INFO Benchmark_Timer - Begin run: Helper for MergeSort: with no copy with 10000 elements with 844 runs
2024-03-15 23:30:51 INFO TimeLogger - Raw time per run (mSec): 1.97
2024-03-15 23:30:51 INFO TimeLogger - Normalized time per run (n log n): 2.78
2024-03-15 23:30:51 INFO SorterBenchmark - run: sort 10,000 elements using SorterBenchmark on class java.lang.String from 22,865 total elements and 844 runs using sorter:
2024-03-15 23:30:51 INFO Benchmark_Timer - Begin run: Helper for QuickSort dual pivot with 10000 elements with 844 runs
2024-03-15 23:30:53 INFO TimeLogger - Raw time per run (mSec): 1.95
2024-03-15 23:30:53 INFO TimeLogger - Normalized time per run (n log n): 2.74
2024-03-15 23:30:53 INFO SorterBenchmark - run: sort 10,000 elements using SorterBenchmark on class java.lang.String from 22,865 total elements and 844 runs using sorter:
2024-03-15 23:30:53 INFO Benchmark_Timer - Begin run: Instrumenting helper for heapsort with 10,000 elements with 844 runs
2024-03-15 23:30:55 INFO TimeLogger - Raw time per run (mSec): 2.68
2024-03-15 23:30:55 INFO TimeLogger - Normalized time per run (n log n): 3.77
2024-03-15 23:30:55 INFO SortBenchmarkHelper - Testing with words: 22,865 from eng-uk_web_2002_10K-sentences.txt
2024-03-15 23:30:55 INFO SortBenchmark - Testing pure sorts with 389 runs of sorting 20,000 words
2024-03-15 23:30:55 INFO SorterBenchmark - run: sort 20,000 elements using SorterBenchmark on class java.lang.String from 22,865 total elements and 389 runs using sorter:
2024-03-15 23:30:55 INFO Benchmark_Timer - Begin run: Helper for MergeSort: with no copy with 20000 elements with 389 runs
2024-03-15 23:30:57 INFO TimeLogger - Raw time per run (mSec): 4.30
2024-03-15 23:30:57 INFO TimeLogger - Normalized time per run (n log n): 2.79
2024-03-15 23:30:57 INFO SorterBenchmark - run: sort 20,000 elements using SorterBenchmark on class java.lang.String from 22,865 total elements and 389 runs using sorter:
2024-03-15 23:30:57 INFO Benchmark_Timer - Begin run: Helper for QuickSort dual pivot with 20000 elements with 389 runs
2024-03-15 23:31:00 INFO TimeLogger - Raw time per run (mSec): 4.28
2024-03-15 23:31:00 INFO TimeLogger - Normalized time per run (n log n): 2.72
2024-03-15 23:31:00 INFO SorterBenchmark - run: sort 20,000 elements using SorterBenchmark on class java.lang.String from 22,865 total elements and 389 runs using sorter:
2024-03-15 23:31:00 INFO Benchmark_Timer - Begin run: Instrumenting helper for heapsort with 20,000 elements with 389 runs
2024-03-15 23:31:02 INFO TimeLogger - Raw time per run (mSec): 5.75
2024-03-15 23:31:02 INFO TimeLogger - Normalized time per run (n log n): 3.73
2024-03-15 23:31:02 INFO SortBenchmarkHelper - Testing with words: 22,865 from eng-uk_web_2002_10K-sentences.txt
```

Instrumented (Statistics):



```
2024-03-16 01:30:07 INFO SortBenchmark - SortBenchmark.main: null with word counts: [10000, 20000, 40000, 80000, 160000]
2024-03-16 01:30:07 INFO SortBenchmark - Beginning String sorts
2024-03-16 01:30:08 INFO SortBenchmarkHelper - Testing with words: 22,865 from eng-uk_web_2002_10K-sentences.txt
2024-03-16 01:30:08 INFO SortBenchmark - Testing pure sorts with 844 runs of sorting 10,000 words
2024-03-16 01:30:08 INFO SorterBenchmark - run: sort 10,000 elements using SorterBenchmark on class java.lang.String from 22,865 total elements and 844 runs using sorter:
2024-03-16 01:30:08 INFO Benchmark_Timer - Begin run: Instrumenting helper for MergeSort: with no copy with 10,000 elements with 844 runs
2024-03-16 01:30:10 INFO TimeLogger - Raw time per run (mSec): 2.11
2024-03-16 01:30:10 INFO TimeLogger - Normalized time per run (n log n): 2.97
2024-03-16 01:30:10 INFO SortBenchmark - MergeSort: with no copy: StatPack {hits: mean=269,783; stdDev=287, normalized=2.929; copies: 110,000, normalized=1.194; inversions:
StatPack {hits: mean=269,783; stdDev=287, normalized=2.929; copies: 110,000, normalized=1.194; inversions: <unset>; swaps: mean=9,761; stdDev=85, normalized=0.106; fixe
2024-03-16 01:30:10 INFO SorterBenchmark - run: sort 10,000 elements using SorterBenchmark on class java.lang.String from 22,865 total elements and 844 runs using sorter:
2024-03-16 01:30:10 INFO Benchmark_Timer - Begin run: Instrumenting helper for QuickSort dual pivot with 10,000 elements with 844 runs
2024-03-16 01:30:12 INFO TimeLogger - Raw time per run (mSec): 2.02
2024-03-16 01:30:12 INFO TimeLogger - Normalized time per run (n log n): 2.84
2024-03-16 01:30:12 INFO SortBenchmark - QuickSort dual pivot: StatPack {hits: mean=423,853; stdDev=19,074, normalized=4.602; copies: 0, normalized=0.000; inversions: <uns
InstrumentedHelper - Closing Helper: QuickSort dual pivot with statPack:
StatPack {hits: mean=423,853; stdDev=19,074, normalized=4.602; copies: 0, normalized=0.000; inversions: <unset>; swaps: mean=66,540; stdDev=4,057, normalized=0.722; fi
2024-03-16 01:30:12 INFO SorterBenchmark - run: sort 10,000 elements using SorterBenchmark on class java.lang.String from 22,865 total elements and 844 runs using sorter:
2024-03-16 01:30:12 INFO Benchmark_Timer - Begin run: Instrumenting helper for heapsort with 10,000 elements with 844 runs
2024-03-16 01:30:15 INFO TimeLogger - Raw time per run (mSec): 2.56
2024-03-16 01:30:15 INFO TimeLogger - Normalized time per run (n log n): 3.61
2024-03-16 01:30:15 INFO SortBenchmark - heapsort: StatPack {hits: mean=967,556; stdDev=483, normalized=10.505; copies: 0, normalized=0.000; inversions: <unset>; swaps: me
SortBenchmarkHelper - Testing with words: 22,865 from eng-uk_web_2002_10K-sentences.txt
2024-03-16 01:30:15 INFO SorterBenchmark - run: sort 10,000 elements using SorterBenchmark on class java.lang.String from 22,865 total elements and 844 runs using sorter:
2024-03-16 01:30:15 INFO Benchmark_Timer - Begin run: Instrumenting helper for MergeSort: with no copy with 10,000 elements with 844 runs
2024-03-16 01:30:17 INFO TimeLogger - Raw time per run (mSec): 2.09
2024-03-16 01:30:17 INFO TimeLogger - Normalized time per run (n log n): 2.94
2024-03-16 01:30:17 INFO SortBenchmark - MergeSort: with no copy: StatPack {hits: mean=269,783; stdDev=287, normalized=2.929; copies: 110,000, normalized=1.194; inversions:
SorterBenchmarkHelper - Closing Helper: MergeSort: with no copy with statPack:
StatPack {hits: mean=269,783; stdDev=287, normalized=2.929; copies: 110,000, normalized=1.194; inversions: <unset>; swaps: mean=9,761; stdDev=85, normalized=0.106; fixe
```