Program Structures and Algorithms

Spring 2023 (SEC – 8)

NAME: Rishi Desai

NUID: 002751030

Assignment 6

Task: Finding out to the best predictor for each given sorting algorithm based on deciding factors such as swaps, compares, hits, copies, etc.

Conclusion:

Quick sort (dual pivot) is the best performing algorithm of all of them. Although its swaps, compares and hits are higher than merge sort, its copies are 0. For merge sort, copies are a significant number due to usage of auxillary array. Due to this merge sort falls behind quick sort overall. Heap sort is the worst performing algorithm of all of them. This is because even though the copies are 0, the swaps, compares, and hits are significantly higher. This makes it fall behind merge sort.

For merge sort, the best R^2 value obtained was 0.9965. This was for log(time taken) vs log(swaps). Hence, we can say that swaps are the most accurate predictor for this algorithm.

For quick sort (dual pivot), the best R^2 value obtained was 0.9926. This was for log(time taken) vs log(swaps). Hence, we can say that swaps are the most accurate predictor for this algorithm.

For heap sort, the best R^2 value obtained was 0.9966. This was for log(time taken) vs the logarithm for each of the deciding factors (swaps, compares and hits). Hence, we can say that swaps, compares, and hits are all accurate predictors for this algorithm.

Evidence to support that conclusion and graphical representation:

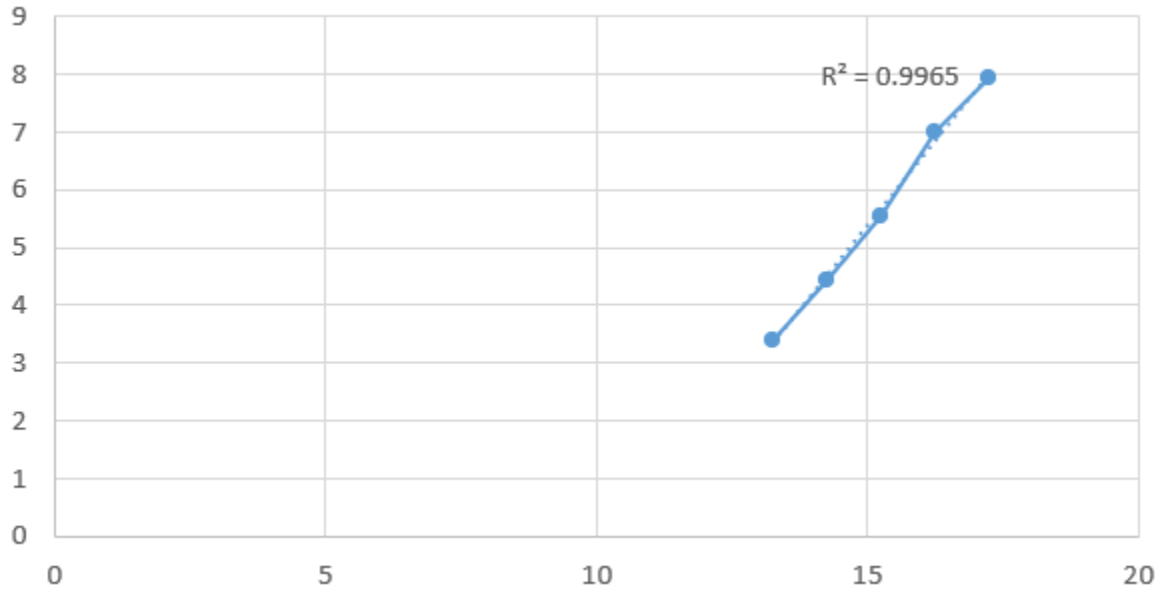
The following tables show the time taken for execution for various array sizes. They also show the time taken when instrumentation was enabled and the corresponding number of swaps, compares, hits and copies.

The graphs are used to determine the value of R2 which will help us determine which factor closely predicts the time taken. They are graphs which have the logarithm of the deciding factors (swaps/compares/hits/copies) on the X-axis and the logarithm of the time taken on the Y-axis. Note that I have considered the time taken when instrumentation was enabled since the deciding factors were calculated at that time.

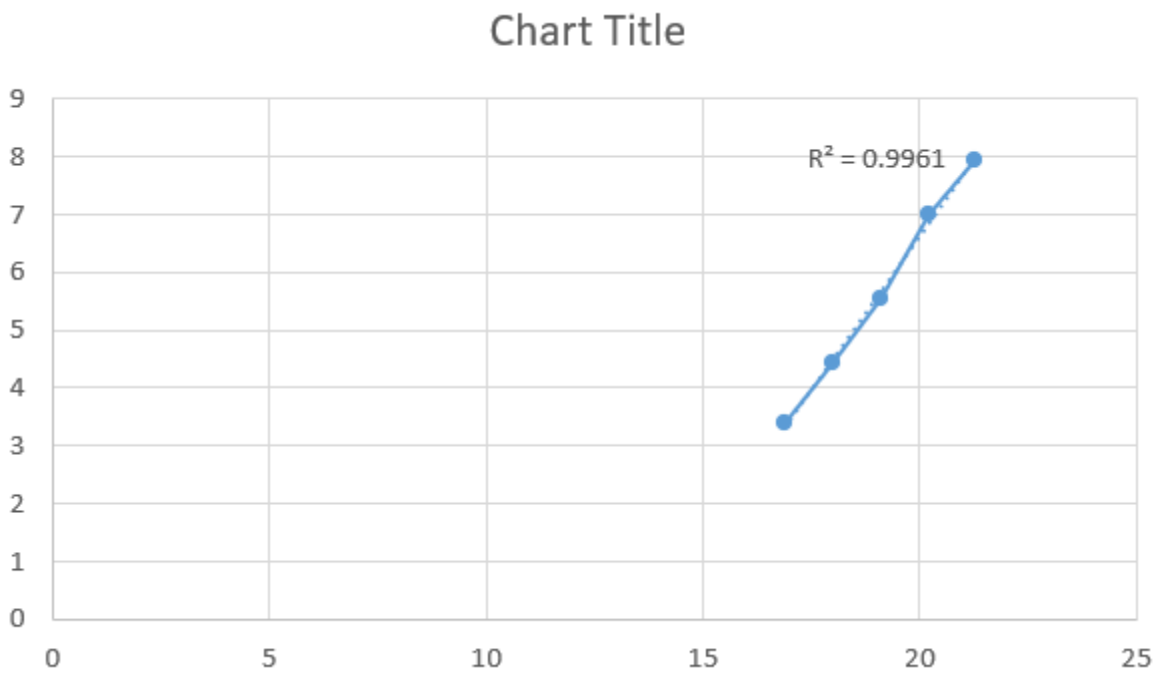
Merge Sort

Array size	Time taken in milliseconds	Time taken (Instrumented) in milliseconds	Swaps	Compares	Hits	Copies
10000	9.51	10.58	9896	121595	259584	110000
20000	22.05	21.72	19445	262982	557780	240000
40000	47.91	46.59	38755	565772	1195020	520000
80000	116.18	126.79	77784	1211936	2551136	1120000
160000	231.99	243.68	156244	2584064	5424976	2400000

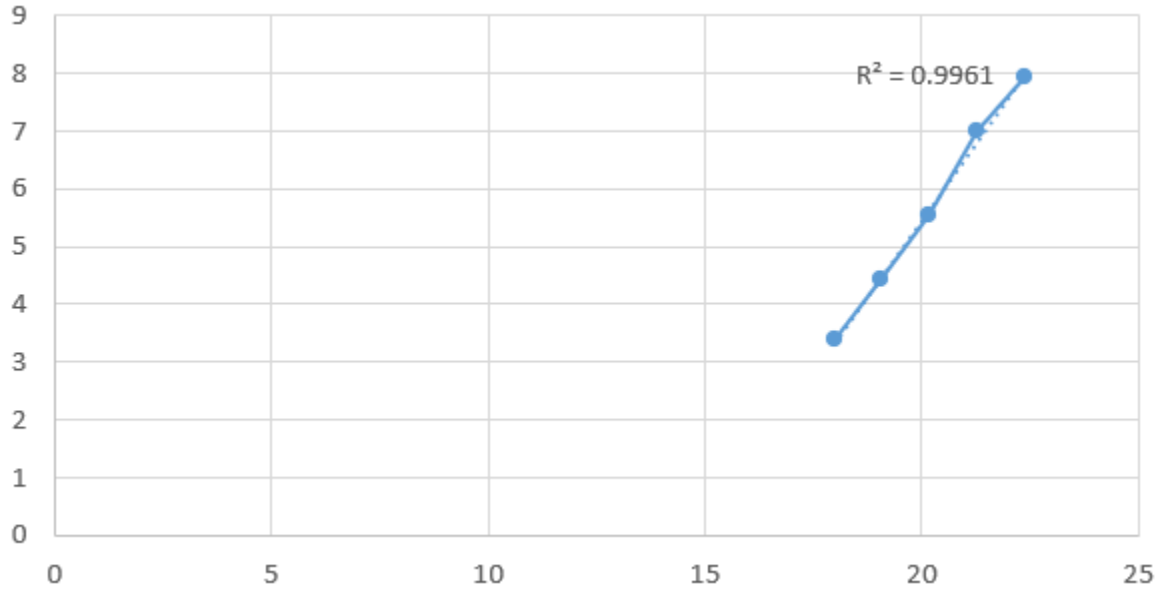
log (time taken) vs log (swaps)



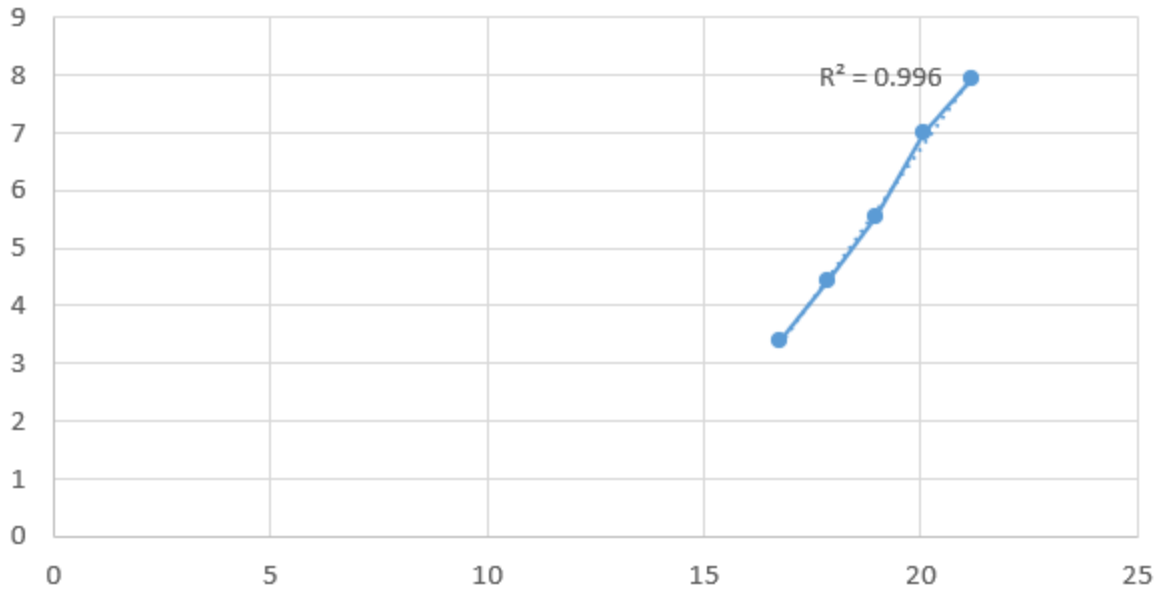
log (time taken) vs log (compares)



log (time taken) vs log (hits)



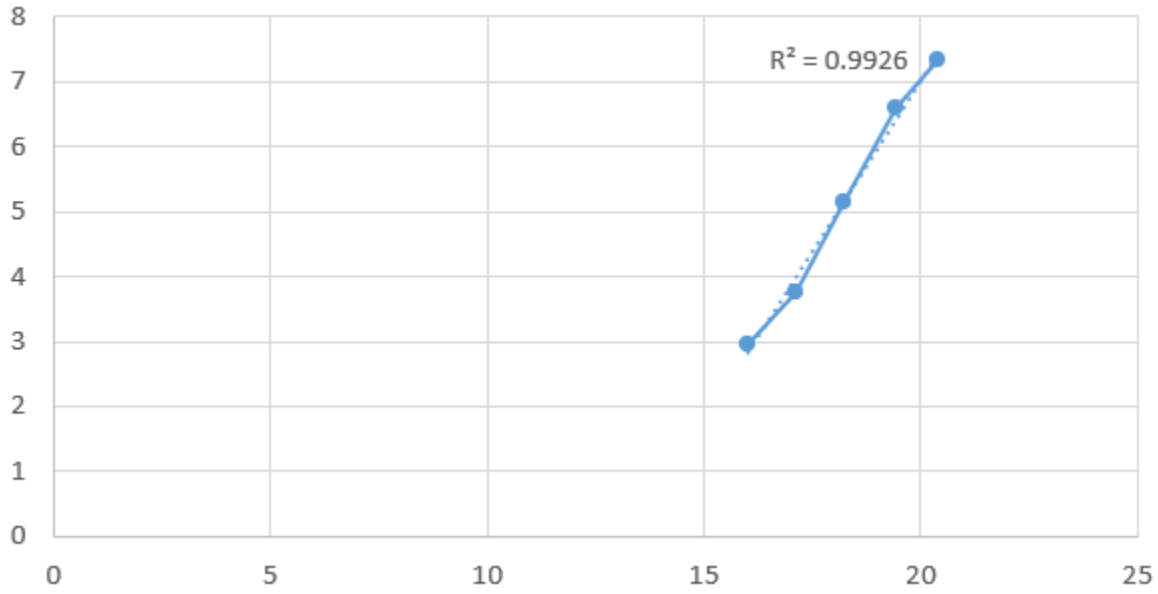
log (time taken) vs log (copies)



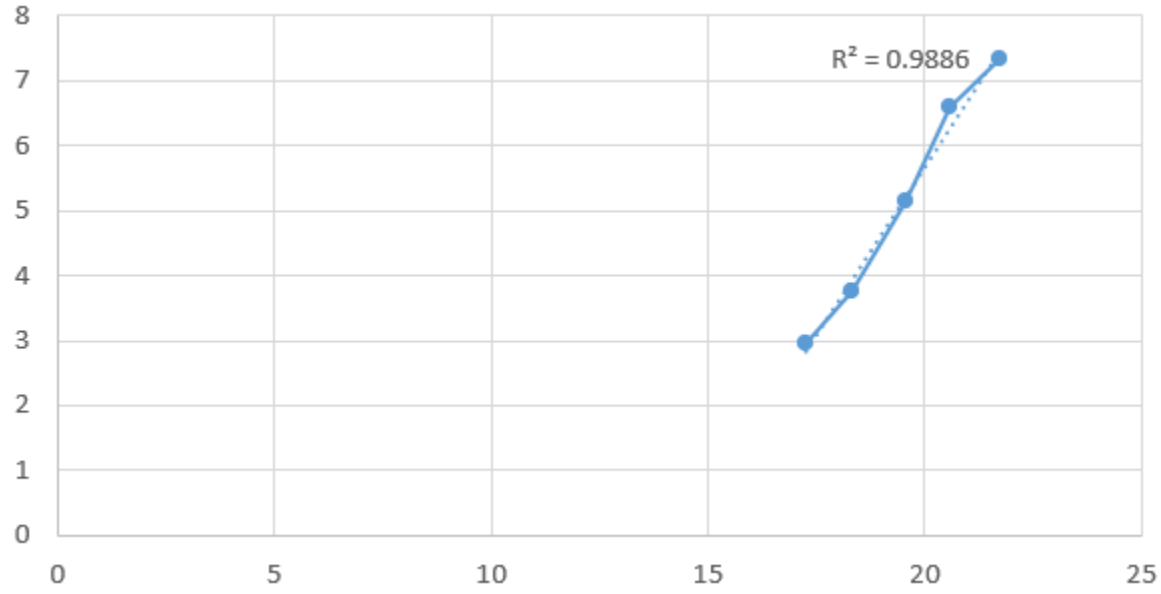
Quick Sort (Dual Pivot)

Array size	Time taken in milliseconds	Time taken (Instrumented) in milliseconds	Swaps	Compares	Hits	Copies
10000	7.45	7.79	67467	159342	424599	0
20000	15.84	13.5	142898	333240	896033	0
40000	25.01	35.16	306644	778537	1989777	0
80000	84.93	95.93	710274	1599654	4405804	0
160000	162.87	162.79	1389645	3493449	8992426	0

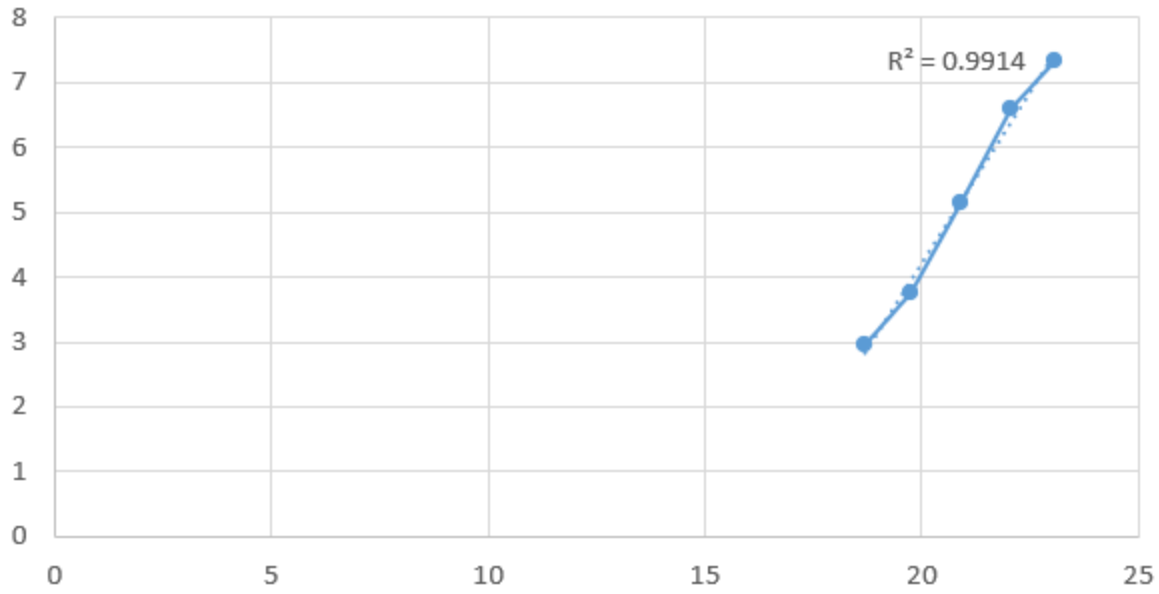
log (time taken) vs log (swaps)



log (time taken) vs log (compares)



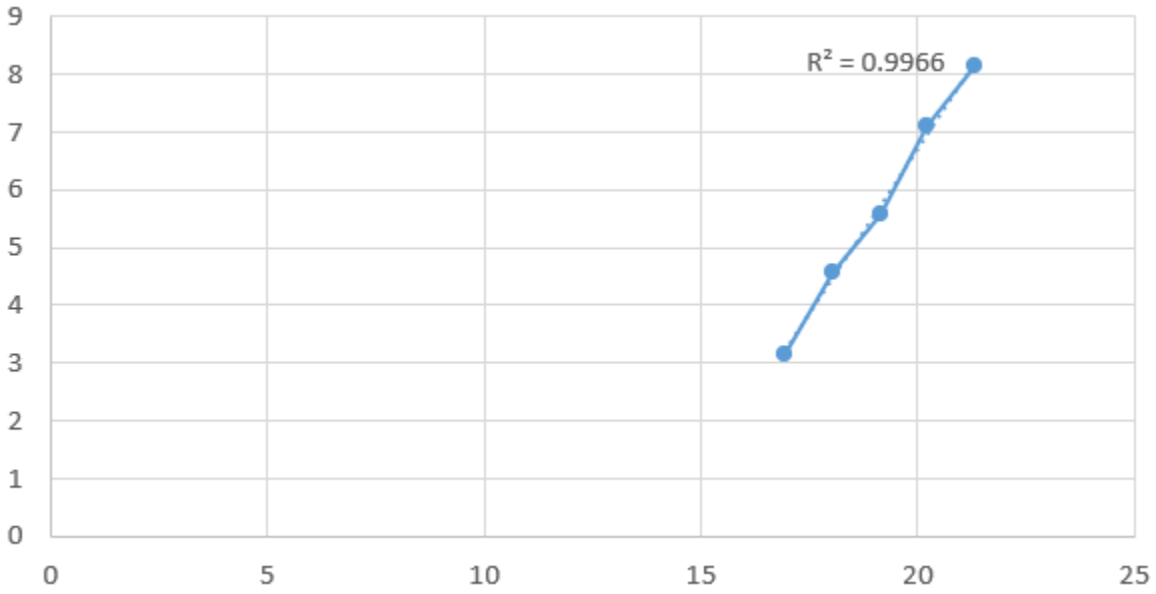
log (time taken) vs log (hits)



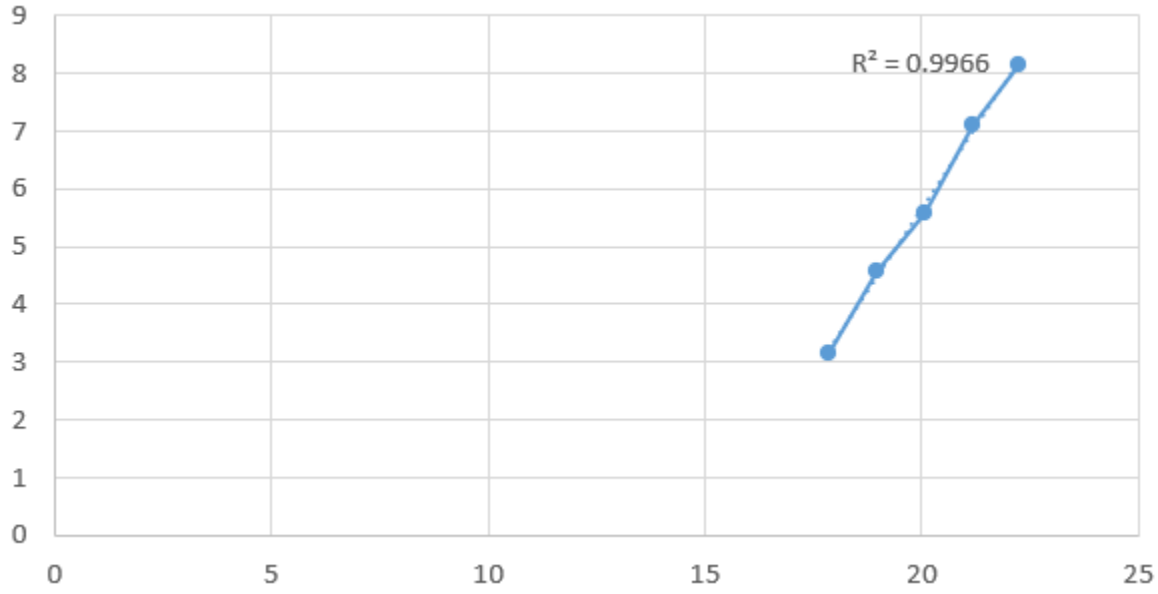
Heap Sort

Array size	Time taken in milliseconds	Time taken (Instrumented) in milliseconds	Swaps	Compares	Hits	Copies
10000	11.02	8.92	124334	235429	968194	0
20000	18.58	23.83	268432	510715	2095158	0
40000	48.79	48.18	576654	1101303	4509222	0
80000	154.98	137.44	1233749	2363060	9661116	0
160000	247.84	283.29	2627435	5046578	20602896	0

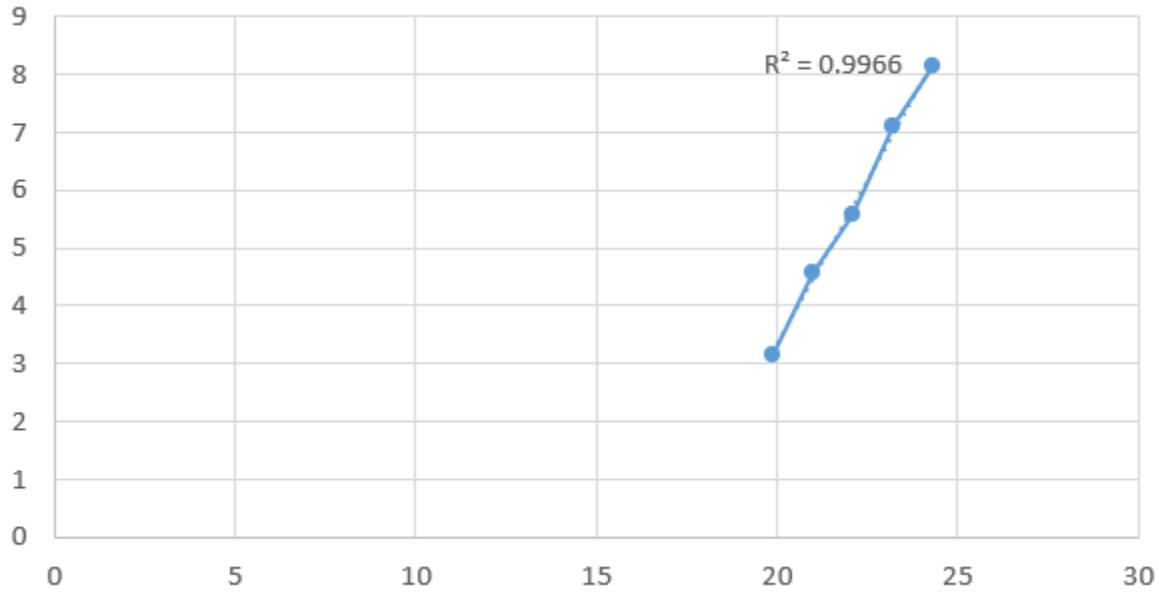
log (time taken) vs log (swaps)



log (time taken) vs log (compares)



log (time taken) vs log (hits)



Unit test screenshots: