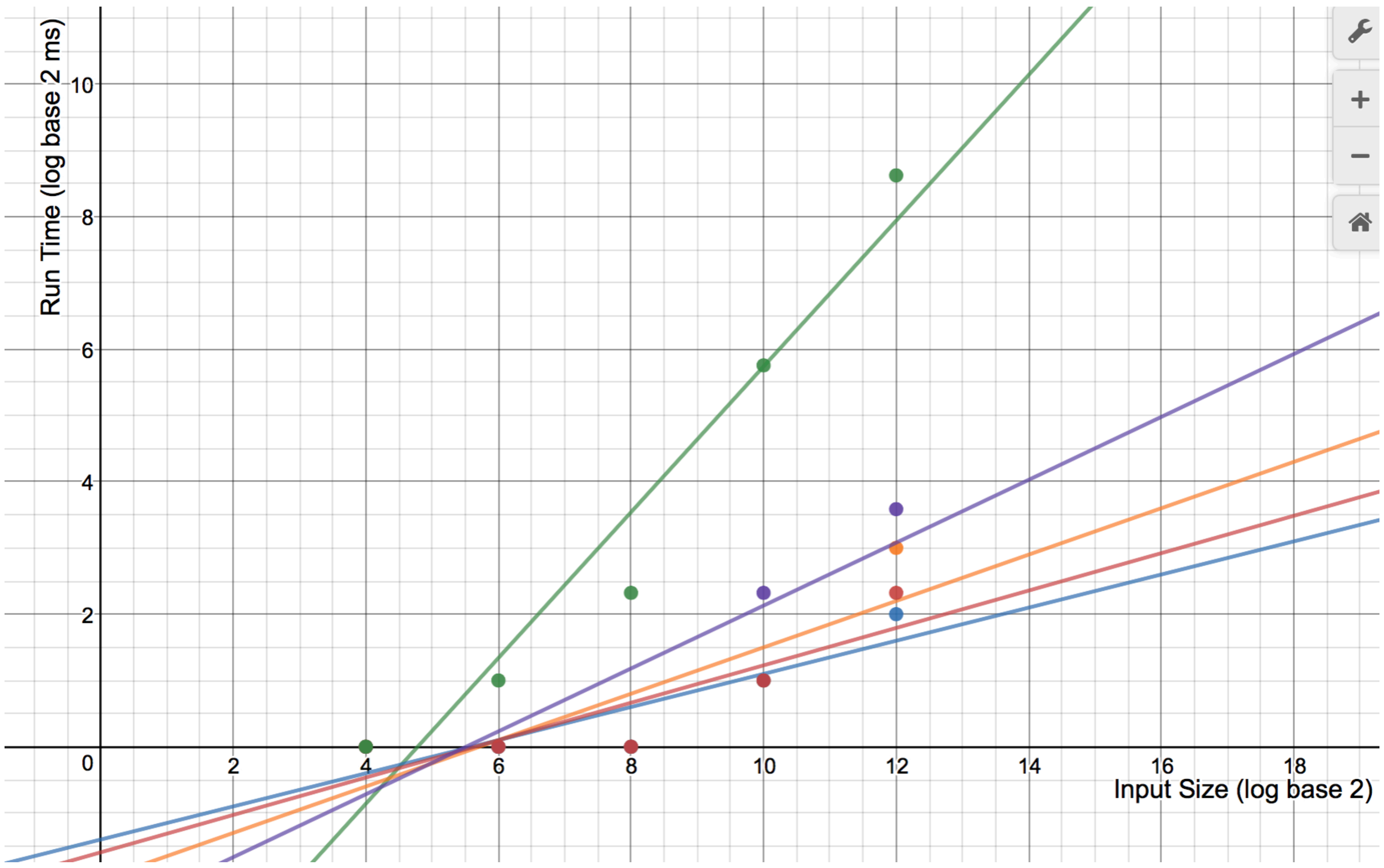
Assignment 1 Analysis

The run times of the different sorting algorithms is as followed (times are in seconds)

	Insert Comparable	Heap	Top Down Merge	Bottom Up Merge	Quicksort	3 - Quicksort
2^6	0.002	0.0	0.0	0.0	0.0	0.0
2^8	0.005	0.001	0.0	0.001	0.001	0.001
2^10	0.054	0.005	0.002	0.002	0.002	0.002
2^12	0.394	0.012	0.008	0.004	0.005	0.005

The results of graphing the log-log plot of run time (ms) vs input size was:



Legend:
Green - Comparable Insertion
Purple - Heapsort
Orange - Top Down Mergesort
Blue - Bottom Up Mergesort
Red - Both Quicksorts

From this I was able to get the slope intercept equation from the graph using a line of best fit, which are as follows:

Comparable Insertion	y = 1.09995x - 5.25982
Heap	y = 0.474593x - 2.61536
TD Merge	y = 0.35x - 2
BU Mergey	y = 0.25x -1.4
Both Quicksorts	y = 0.282193x - 1.59316

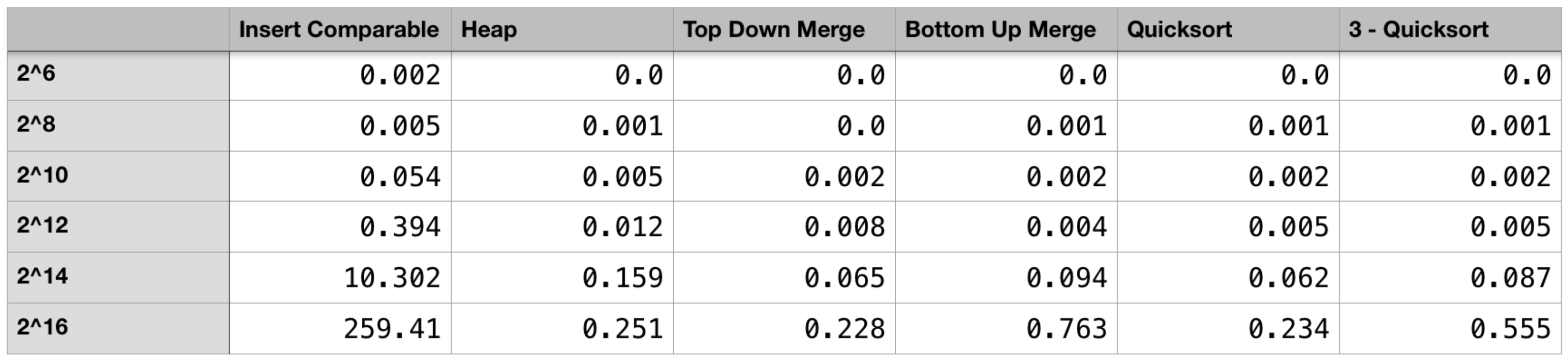
With this information I was able to derive the equations for the run times (in ms) for any input size:

Comparable Insertion	y = 0.02609975365 x^(1.09995)
Heap	y = 0.16319174658 x^(0.474593)
TD Merge	y = 0.25 x^(0.35)
BU Merge	y = 0.37892914162 x^(0.25)
Both Quicksorts	y = 0.33144467964 x^(0.282193)

With these equations I could estimate the run times in ms for input sizes of 2^{14} and 2^{16}:

	2^{14}	2^{16}
Comparable Insertion	1127.94421292	5182.31142472
Heap	16.3242182597	31.5185224611
TD Merge	7.46442338301	12.1257325321
BU Merge	4.28709385006	6.06286626592
Both Quicksorts	5.12496445985	7.57858233399

After running the different sorts with input sizes of 2^14 and 2^16 I got the run times of (in seconds):



	Insert Comparable	Heap	Top Down Merge	Bottom Up Merge	Quicksort	3 - Quicksort
2^6	0.002	0.0	0.0	0.0	0.0	0.0
2^8	0.005	0.001	0.0	0.001	0.001	0.001
2^10	0.054	0.005	0.002	0.002	0.002	0.002
2^12	0.394	0.012	0.008	0.004	0.005	0.005
2^14	10.302	0.159	0.065	0.094	0.062	0.087
2^16	259.41	0.251	0.228	0.763	0.234	0.555

As can be easily seen the hypothesis for the run times with input sizes for 2^14 and 2^16 were all very off. One reason why this might be is that the stopwatch in eclipse does not take accurate enough data. Many of the run times for smaller values are showing up as 0 and the ones with higher inputs have drastically different run times every time the program is executed.

NOTE: As my insertion without comparable would never finish running in the junit tests, and the binary insertion did not work I did the analysis with the 2 types of quicksort instead.