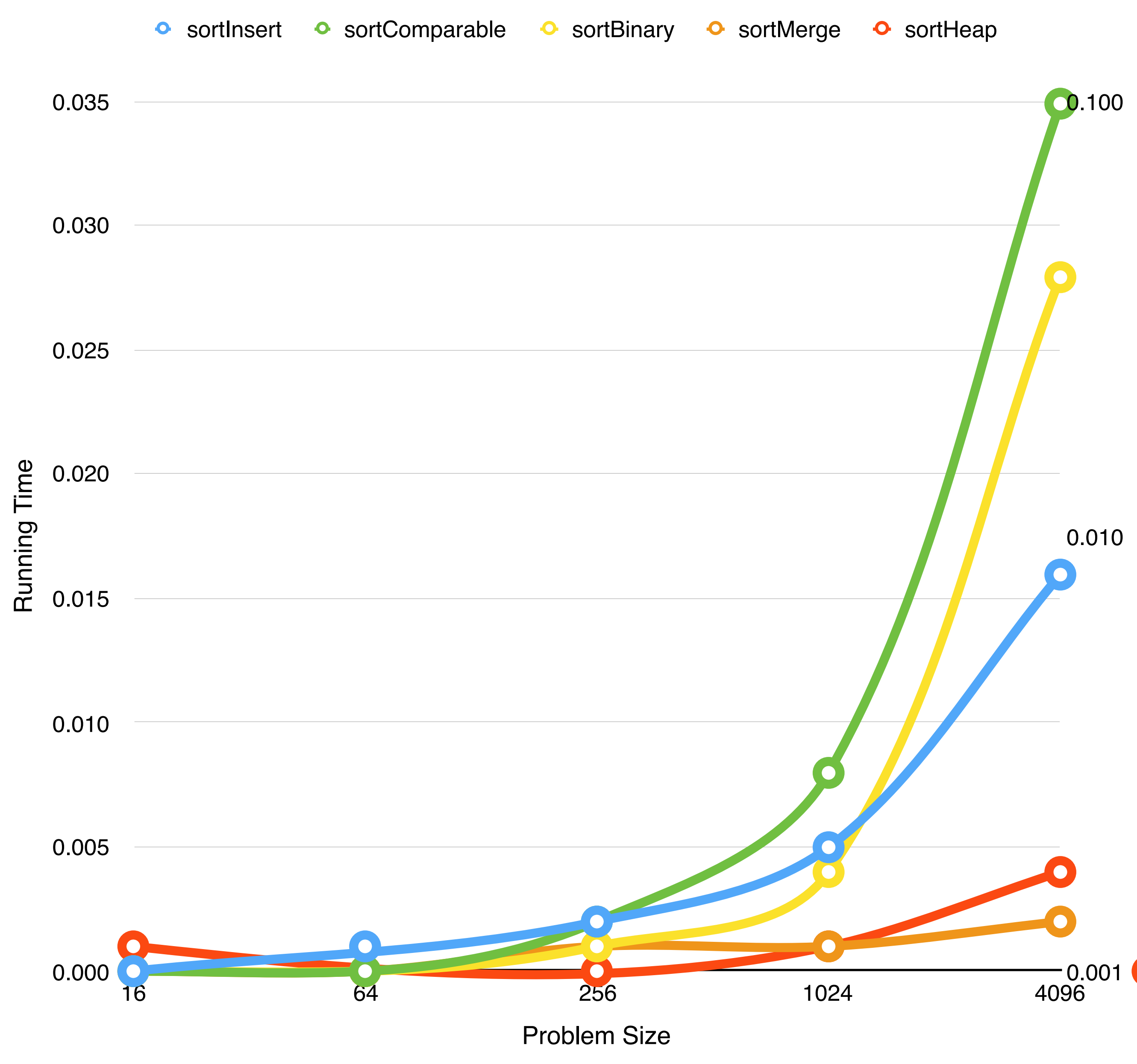
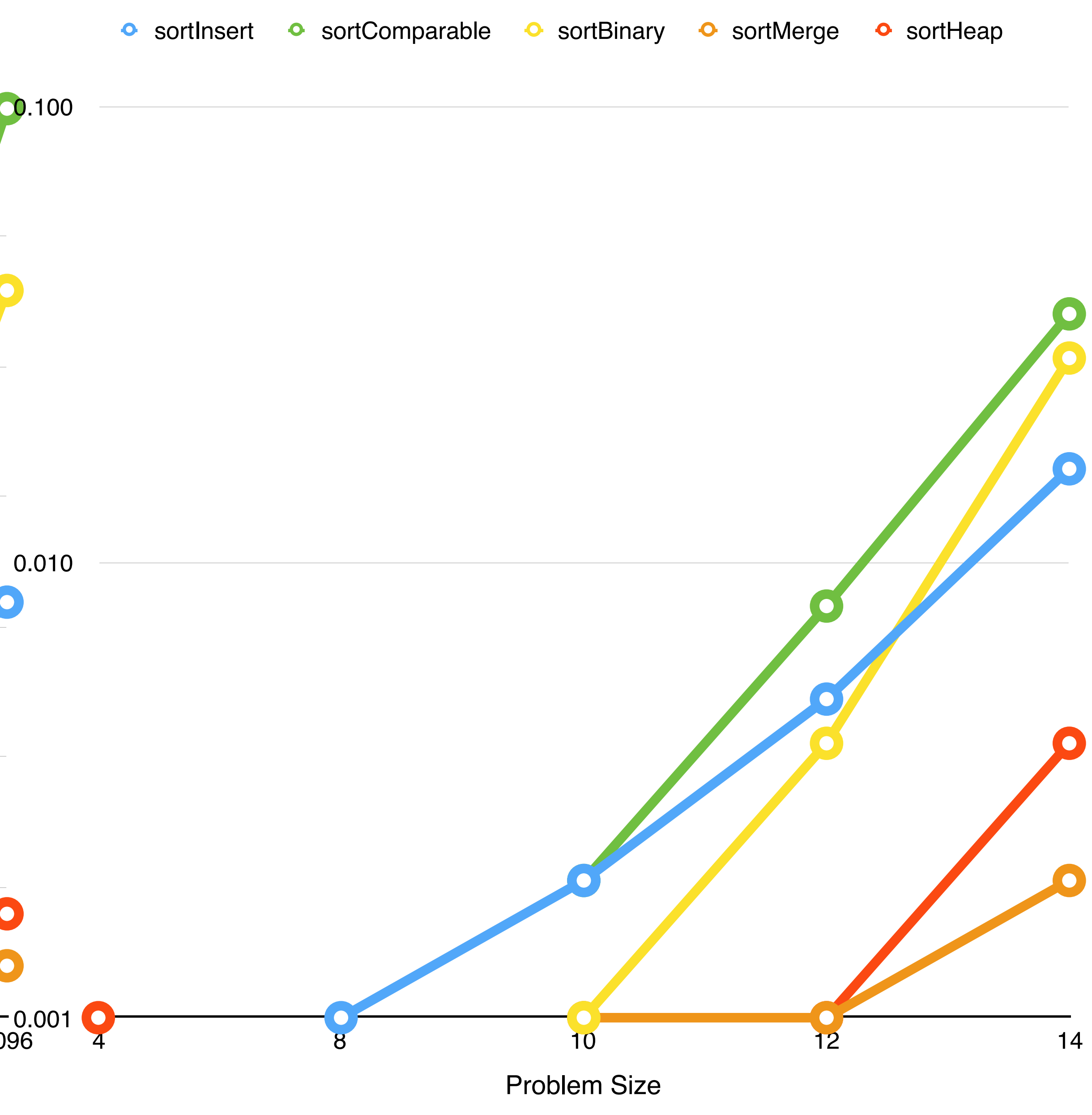
Sizes of the Datasets	Execution Times of Sorting Algorithms				
	sortInsert	sortComparable	sortBinary	sortMerge	sortHeap
16	0.000	0.000	0.000	0.000	0.001
64	0.001	0.000	0.000	0.000	0.000
256	0.002	0.002	0.001	0.001	0.000
1024	0.005	0.008	0.004	0.001	0.001
4096	0.016	0.035	0.028	0.002	0.004





`sortInsert:`

Hypothesis: The running time is O(N^2)
Predictions:
- 0.209 seconds for N = 2^{14}
- 8.001 seconds for N = 2^{16}

Actual Running Time:
- 0.207 seconds for N = 2^{14}
- 8.157 seconds for N = 2^{16}

Hypothesis validated.

`sortComparable:`

Hypothesis: The running time is O(N^2)
Predictions:
- 0.647 seconds for N = 2^{14}
- 20.34 seconds for N = 2^{16}

Actual Running Time:
- 0.611 seconds for N = 2^{14}
- 25.36 seconds for N = 2^{16}

Hypothesis validated.

`sortBinary:`

Hypothesis: The running time is O(N^2)
Predictions:
- 0.371 seconds for N = 2^{14}
- 11.891 seconds for N = 2^{16}

Actual Running Time:
- 0.362 seconds for N = 2^{14}
- 12.731 seconds for N = 2^{16}

Hypothesis validated.

`sortMerge:`

Hypothesis: The running time is O(NlogN) (about 9.965 x 10^{-8} x NlogN)
Predictions:
- 0.023 seconds for N = 2^{14}
- 0.104 seconds for N = 2^{16}

Actual Running Time:
- 0.024 seconds for N = 2^{14}
- 0.092 seconds for N = 2^{16}

Hypothesis validated.

`sortHeap:`

Hypothesis: The running time is O(NlogN) (about 3.92 x 10^{-8} x NlogN)
Predictions:
- 0.009 seconds for N = 2^{14}
- 0.041 seconds for N = 2^{16}

Actual Running Time:
- 0.009 seconds for N = 2^{14}
- 0.042 seconds for N = 2^{16}

Hypothesis validated.

According to the chart in the normal scale, we can easily observe that all the plots for Insertion Sorts are similar to the quadratic plots. And log-log chart confirms our hypotheses, that is the running time of Insertion Sort is quadratic, since its plot in log-log are linear. Moreover, we can take a closer look at the code for Insertion Sorts. Each time, the loops iterates 1,2,3,…, N times, in the worst case which shows that the running time is also O(N^2).
As for merge sort, it takes O(logN) times to split the array and O(N) times to merge the subarrays together. Hence, the time complexity is O(NlogN). In terms of Heap Sort, we need logN times for each element to sink after we take out the maximum key, in the worst case and there are N elements indicating that the running time would also be O(NlogN). And from the log-log plot we can tell the plot for both merge sort and heap sort are linearithmic which confirms our hypotheses as well.