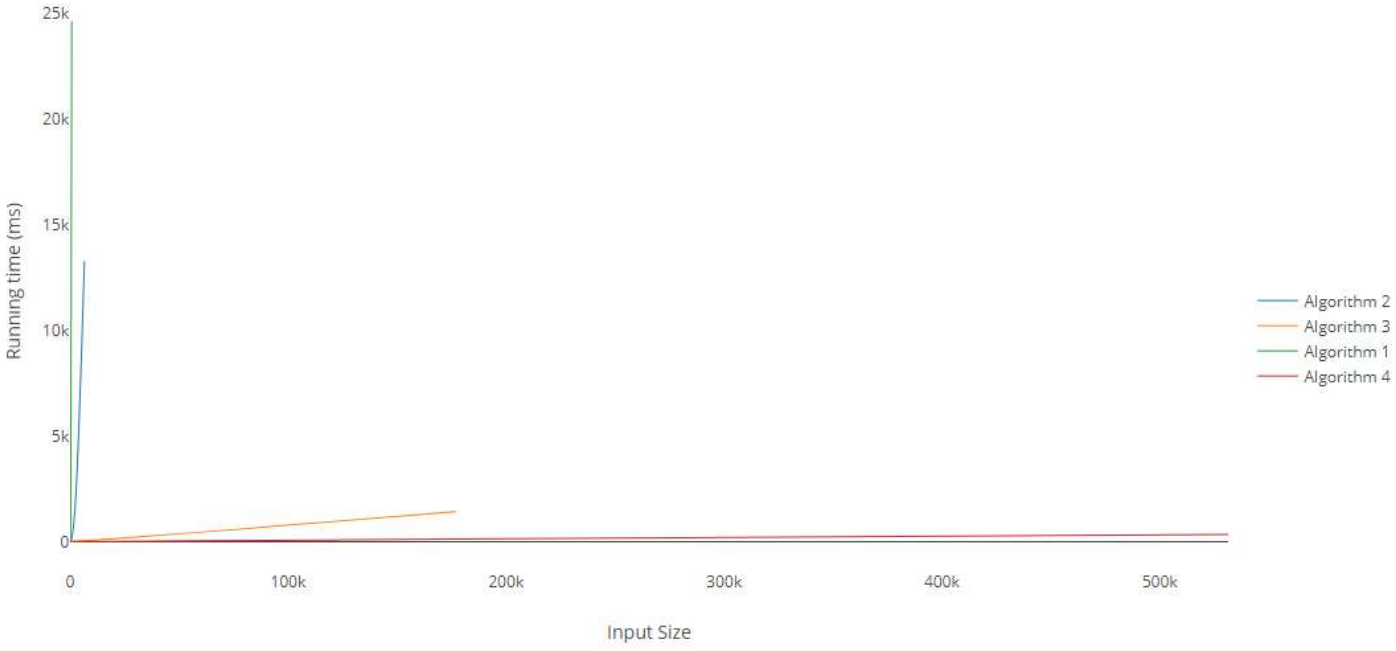
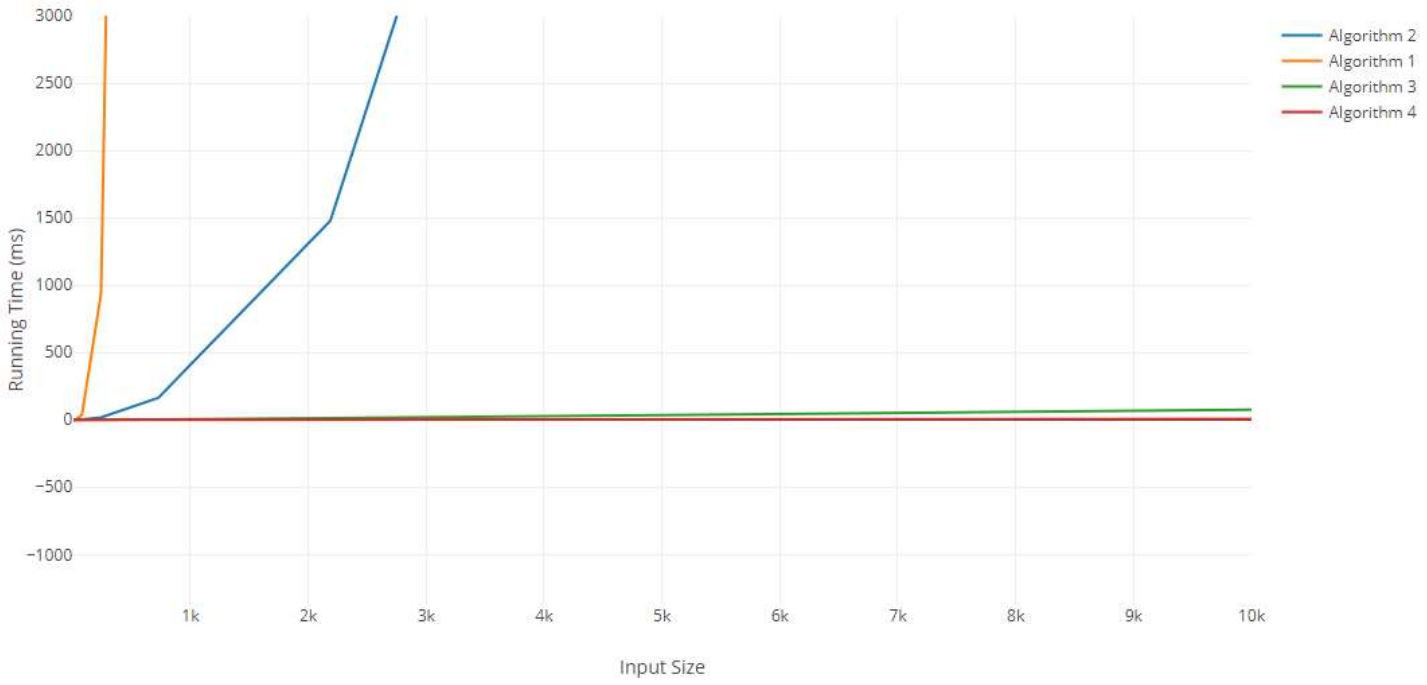
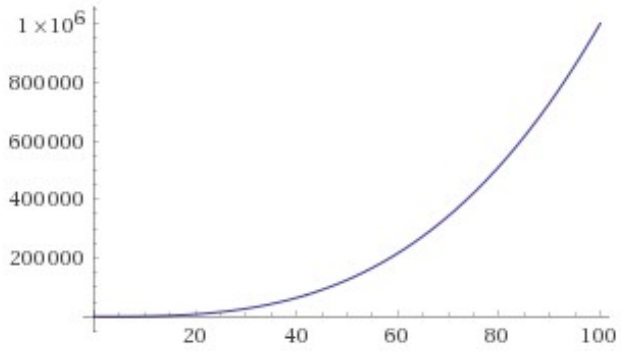
Input Size	Algorithm Time (ms)			
	1	2	3	4
	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4
N = 9	0	0	0	0
N = 27	1	0	0	0
N = 81	36	2	1	0
N = 243	939	18	1	0
N = 729	24640	164	3	1
N = 2187	NA	1476	12	1
N = 6561	NA	13283	42	4
N = 19683	NA	NA	133	13
N = 59049	NA	NA	437	37
N = 177147	NA	NA	1420	109
N = 531441	NA	NA	NA	329
N = 1594323	NA	NA	NA	983

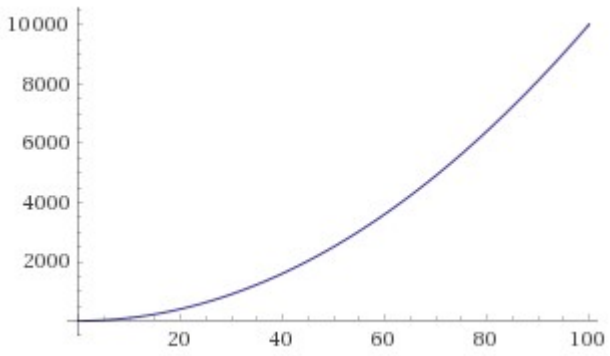




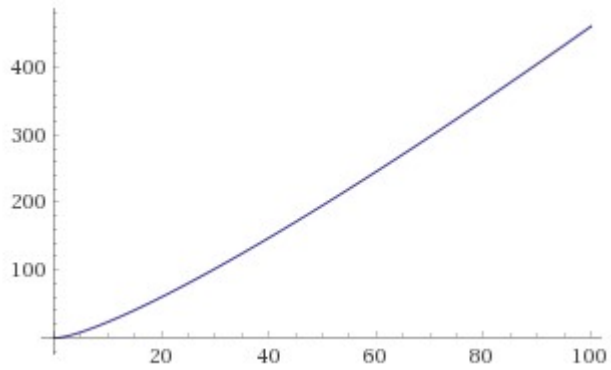
x^3 Plot



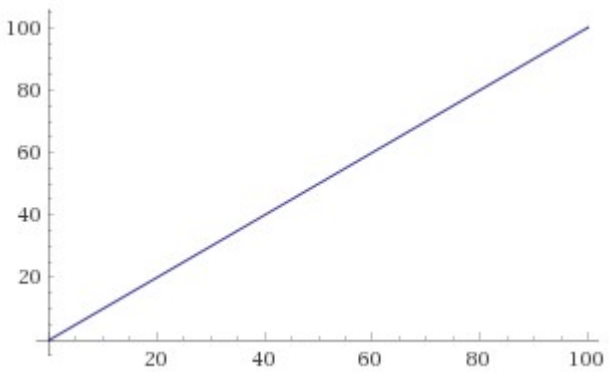
x^2 Plot



x*log(x) Plot



x Plot



Computer specifications:

Processor:	Intel® Core™ i7-4702MQ CPU @ 2.20GHz 2.19 GHz
Installed memory (RAM):	8.00 GB (7.76 GB usable)

Discussion:

The test result plots are similar to their corresponding mathematical plots. However, they slightly differ because y axis is shown in milliseconds which can be changed to other time measurement units for different values, and the Big O notation of algorithms only regard their most run time-consuming function, which can also have a different multiplier than 1. For example, first algorithm (as calculated in the handout) uses "(N^3+3N^2+2N)/6" run time for N size arrays. This algorithms plot is similar to x^3's plot in higher N values, as N^3/6 increases much faster than (3N^2+2N)/6. Yet, there is a multiplier difference (1/6 and 1) and y axis is milliseconds which can be changed to any other time measurement unit. Also at lower N values algorithm's run time can be more dependent on (3N^2+2N)/6. Nevertheless, the test result plots are very similar to their theoretical plots in a big scale.

The different growth rates between algorithms are a result of difference in their efficiency. Algorithm 4 is the most inefficient algorithm as it has most unnecessary calculations for example.