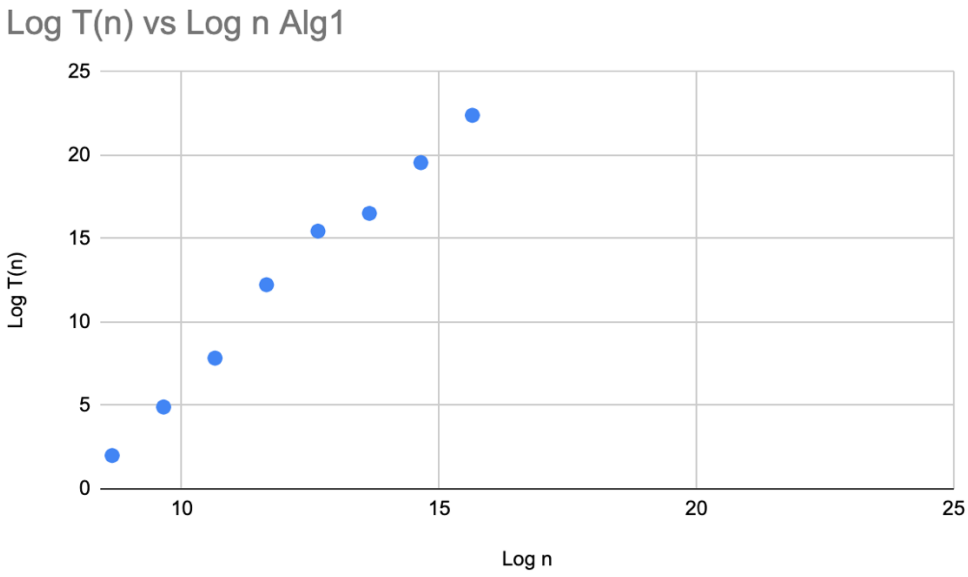
Estimate Secret Algorithms #2: Benjamin Katz

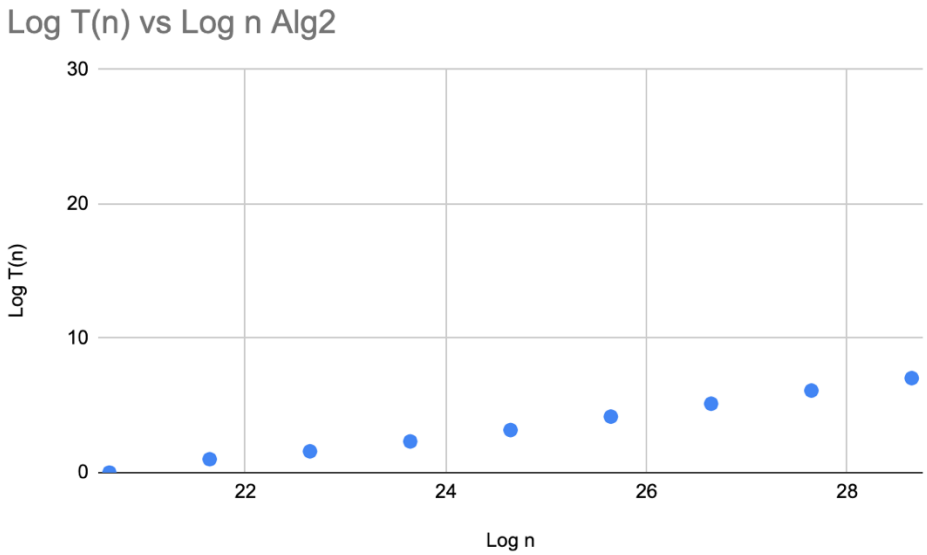
My client code that interacts with the SecretAlgorithms class sets up a file for the data to output, then it runs through a while loop that starts the testing apparatus with a value of 200. The code calls a method to set up every secret algorithm and then starts a timer using System.currentTimeMillis and executes a specific algorithm. After the algorithm finishes executing the timer is stopped and the data is recorded in a file. Once every algorithm has been tested with a given input and the data is recorded, the data is doubled the process begins again.

Algorithm1			
n input	T(n) (reported in milliseconds)	Log n	Log T(n)
400	4	8.6	2
800	30	9.6	4.9
1600	229	10.6	7.8
3200	4825	11.6	12.2
6400	44805	12.6	15.5
12800	93858	13.6	16.5
25600	771815	14.6	19.6
51200	5525644	15.6	22.4



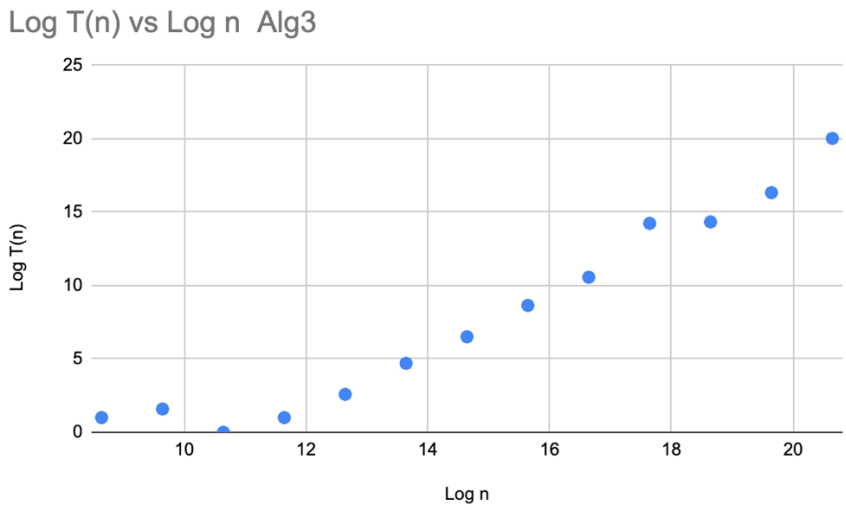
Looking at a best fit line of the log-log graph of this data it is apparent that it has a slope of roughly 3. This can be calculated by doing a simple (y1-y2)/(x1-x2) : (4.9-19.6)/(9.6-14.6) = 2.94 the equation of the line is therefore $\lg(T(N)) = 3 \lg N + \lg a$ Which is equivalent to $T(N) = aN^3$ therefore the order of growth is n cubed.

Algorithm2			
n input	T(n) (reported in milliseconds)	Log n	Log T(n)
1638400	1	20.6	0
3276800	2	21.6	1
6553600	3	22.6	1.6
13107200	5	23.6	2.3
26214400	9	24.6	3.2
52428800	18	25.6	4.2
104857600	35	26.6	5.1
209715200	69	27.6	6.1
419430400	131	28.6	7.0



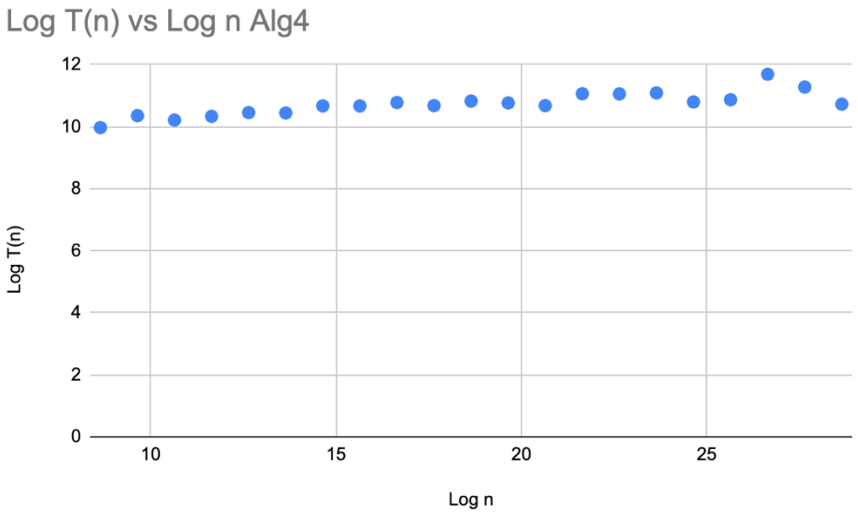
Looking at a best fit line of the log-log graph of this data it is apparent that it has a slope of roughly 1. This can be calculated by doing a simple (y1-y2)/(x1-x2) : (2.3-6.1)/(23.6-27.6) = .95 the equation of the line is therefore $\lg(T(N)) = 1 \lg N + \lg a$ Which is equivalent to $T(N) = aN$ therefore the order of growth is linear

Algorithm3			
n input	T(n) (reported in milliseconds)	Log n	Log T(n)
400	2	8.6	1
800	3	9.6	1.6
1600	1	10.6	0
3200	2	11.6	1
6400	6	12.6	2.6
12800	26	13.6	4.7
25600	91	14.6	6.5
51200	400	15.6	8.6
102400	1515	16.6	10.6
204800	19316	17.6	14.2
409600	20644	18.6	14.3
819200	82350	19.6	16.3
1638400	1067202	20.6	20.0



Looking at a best fit line of the log-log graph of this data it is apparent that it has a slope of roughly 2. This can be calculated by doing a simple (y1-y2)/(x1-x2) : (1-20)/(11.6-20.6) = 2.1 the equation of the line is therefore $\lg(T(N)) = 2 \lg N + \lg a$ Which is equivalent to $T(N) = aN^2$ therefore the order of growth is N squared

Algorithm4			
n input	t output(nanos)	Log n	Log T(n)
400	1003	8.6	9.9
800	1310	9.6	10.6
1600	1187	10.6	10.2
3200	1290	11.6	10.3
6400	1403	12.6	10.5
12800	1391	13.6	10.4
25600	1629	14.6	10.7
51200	1628	15.6	10.7
102400	1758	16.6	10.8
204800	1639	17.6	10.7
409600	1816	18.6	10.8
819200	1741	19.6	10.8
1638400	1640	20.6	10.7
3276800	2137	21.6	11.1
6553600	2130	22.6	11.1
13107200	2176	23.6	11.1
26214400	1780	24.6	10.8
52428800	1870	25.6	10.7
104857600	3304	26.6	11.7
209715200	2479	27.6	11.3
419430400	1693	28.6	10.7



Looking at a best fit line of the log-log graph of this data it is apparent that it has a slope of roughly undefined. Since with the increase of n there is virtually no change to T(n). This could be an indication of either constant runtime or logarithmic performance. Based on the fact that there seems to be a slight increase I believe it is indicative of logarithmic performance.