# EsitmateSecretAlgorithms2

Samuel Sicklick, Oct. 2 2021

## Experimental Apparatus

To generate data from which a conclusion could be drawn regarding the order of growth of each given "SecretAlgorithm", I composed a client code which contained four methods in correspondence to the four algorithms. Each method runs a for loop, where some integer *n* is doubled until a certain higher quantity.[1] Iterating through this loop, the value *n* is used in a call to the BigOMeasurable "setup" method, which sets up the data structure behind each algorithm as a function of *n*. Thus, the algorithm will be positioned to run though data whose quantity is *n*. At this point the client code saves some double value as the starting time, provided by calls to "System.currentTimeMillis()" or "System.nanoTime()",[2] and then executes the program by calling the BigOMeasurable "execute" method. Upon completion of each "SecretAlgorithm"'s execution another double value of time is saved, by calling the aforementioned System methods a second time, and the time elapsed is calculated by subtracting the starting time from this ending time. Once the data has been recorded for how much time the "SecretAlgorithm"s took to run on each data size, the logarithms (base 2) of both the data size *n* and corresponding times are calculated via a private method in my client code.[3] These logarithms allow for the data to be plotted on a log-log scale, where a power law holds if they fall out on a straight line.[4] The slope of this line is equivalent to *b* in the equation $aN^b$ which is the curve that fits the original data.[5] That value *b* is then the power of 2 for which the time increases as the data sizes double, and can be used to determine the "SecretAlgorithm"'s order of growth.[6]

---

[1] These higher quantities of *n* were determined based on step 4 on slide 22 from the "Analysis of Algorithms: Motivation, Techniques & Order-of-Growth" lecture. Each algorithm was run with doubled data sizes until enough data had been collected to estimate the order of growth.

[2] Whether time was measured in units of milliseconds or nanoseconds was dependent on the performance of the algorithms. The data from SecretAlgorithm1 and SecretAlgorithm3 was discernible from measurements of milliseconds, while SecretAlgorithm2 and SecretAlgorithm4 were faster and therefore required a more refined measurement of time.

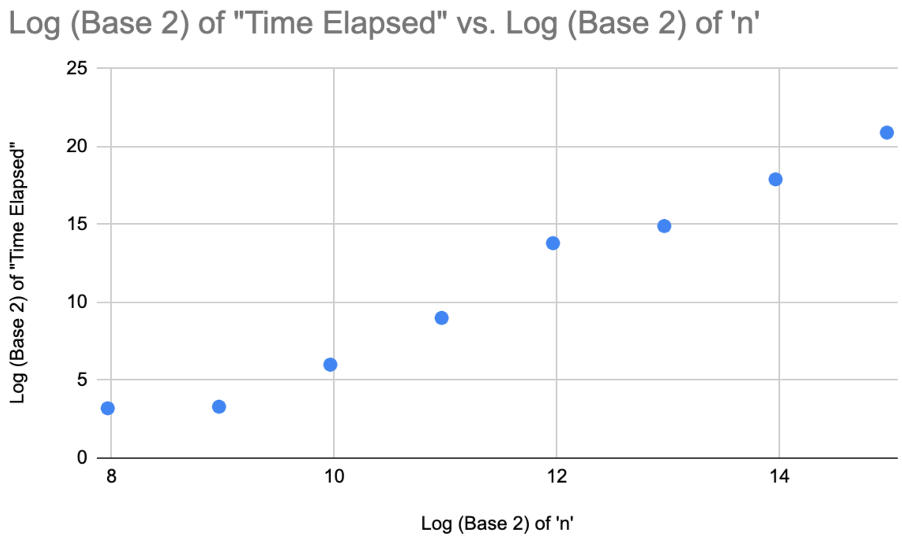
[3] The method generates a logarithm of base 2 of a given integer *i* by dividing the logarithm of base 10 of *i* by the logarithm of base 10 of 2. Both logarithms of base 10 are calculated by the "Math.log()" method.

[4] As per slide 23 from the "Analysis of Algorithms: Motivation, Techniques & Order-of-Growth" lecture.

[5] Meaning the data sizes and elapsed times prior to the taking of their base 2 logarithms.

[6] Hence, if *b* is 1 then as the data sizes double the time elapsed increases by 2, meaning that the algorithm is linear. If *b* is 2 then as the data sizes double the time elapsed increases by 4, meaning that the algorithm is quadratic. Similarly if *b* is 3 then as the data sizes double the time elapsed increases by 8, meaning that the algorithm is cubic.

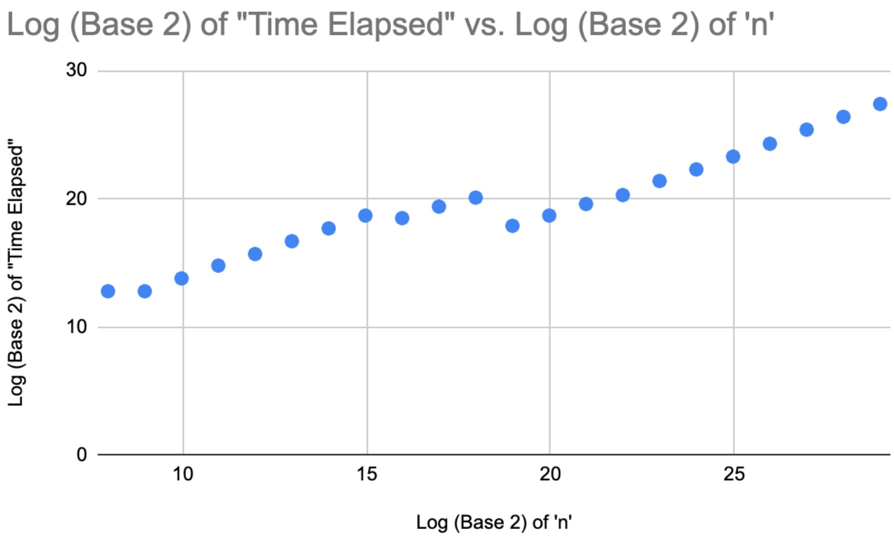
# SecretAlgorithm1




| Data Size 'n' | Log (Base 2) of 'n' | $f$(n) / Time Elapsed (milliseconds) | Log (Base 2) of $f$(n) / "Time Elapsed" |
|---|---|---|---|
| 250 | 8.0 | 9 | 3.2 |
| 500 | 9.0 | 10 | 3.3 |
| 1000 | 10.0 | 66 | 6.0 |
| 2000 | 11.0 | 526 | 9.0 |
| 4000 | 12.0 | 14500 | 13.8 |
| 8000 | 13.0 | 30555 | 14.9 |
| 16000 | 14.0 | 240792 | 17.9 |
| 32000 | 15.0 | 1921785 | 20.9 |

The line of the log-log scale has a rough slope of 3, due to the fact that there are some imperfect points as the data was produced from a real experiment. Thus the value of $b$ is 3, meaning that as the data size doubles the time required for the algorithm to process it increases by a ratio of 8. Accordingly it can be concluded that the order of growth of “SecretAlgorithm1” is cubic, $O(n^3)$.

# SecretAlgorithm2

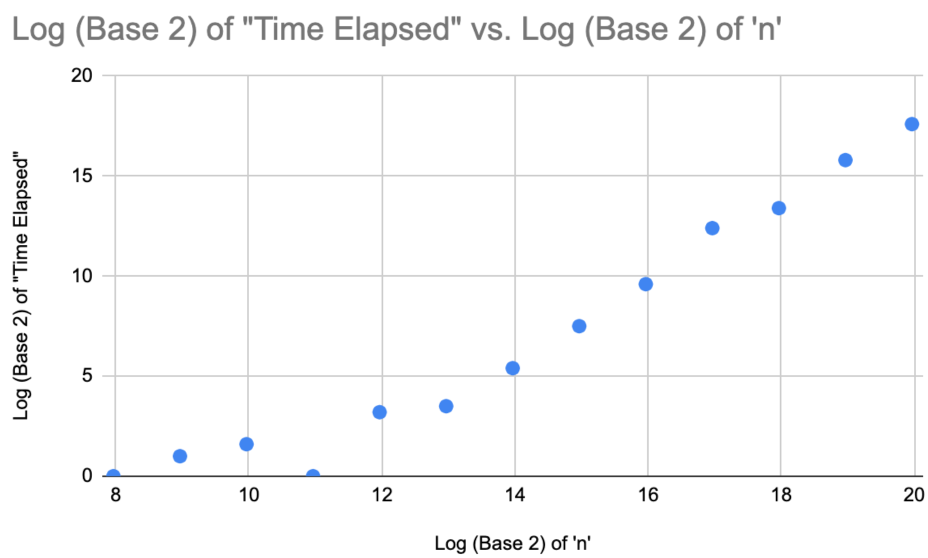



| Data Size 'n' | Log (Base 2) of 'n' | $f$(n) / Time Elapsed (nanoseconds) | Log (Base 2) of $f$(n) / "Time Elapsed" |
|---|---|---|---|
| 250 | 8.0 | 7334 | 12.8 |
| 500 | 9.0 | 7072 | 12.8 |
| 1000 | 10.0 | 14698 | 13.8 |
| 2000 | 11.0 | 27673 | 14.8 |
| 4000 | 12.0 | 54083 | 15.7 |
| 8000 | 13.0 | 106973 | 16.7 |
| 16000 | 14.0 | 213192 | 17.7 |
| 32000 | 15.0 | 428249 | 18.7 |
| 64000 | 16.0 | 358814 | 18.5 |
| 128000 | 17.0 | 676526 | 19.4 |
| 256000 | 18.0 | 1113756 | 20.1 |
| 512000 | 19.0 | 245115 | 17.9 |
| 1024000 | 20.0 | 430445 | 18.7 |
| 2048000 | 21.0 | 791531 | 19.6 |
| 4096000 | 22.0 | 1327809 | 20.3 |
| 8192000 | 23.0 | 2682214 | 21.4 |
| 16384000 | 24.0 | 5208455 | 22.3 |
| 32768000 | 25.0 | 10252449 | 23.3 |
| 65536000 | 26.0 | 20658606 | 24.3 |
| 131072000 | 27.0 | 42814520 | 25.4 |

| | | | |
|---|---|---|---|
| 262144000 | 28.0 | 86774249 | 26.4 |
| 524288000 | 29.0 | 181548323 | 27.4 |

The line of the log-log scale has a rough slope of 1, due to the fact that there are some imperfect points as the data was produced from a real experiment. Thus the value of $b$ is 1, meaning that as the data size doubles the time required for the algorithm to process it increases by a ratio of 2. Accordingly it can be concluded that the order of growth of "SecretAlgorithm2" is linear, $O(n)$.

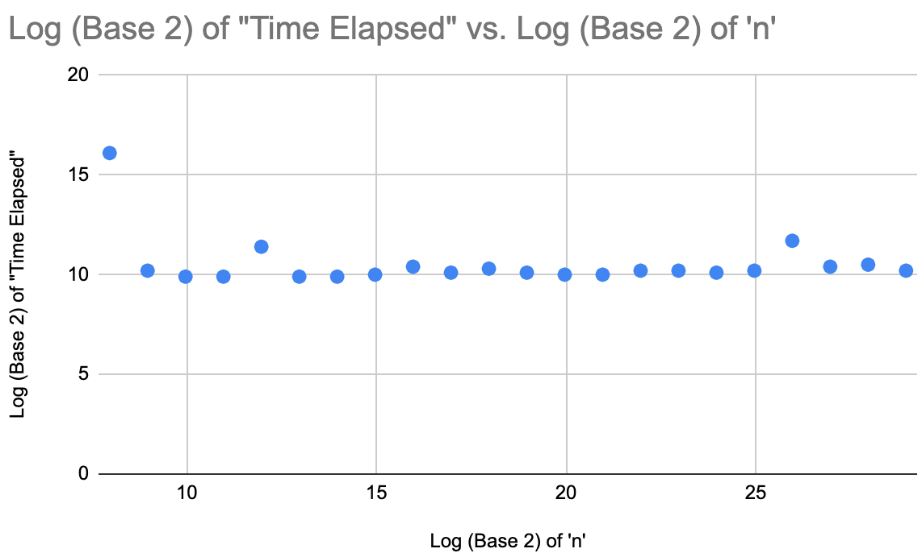
## SecretAlgorithm3




| Data Size 'n' | Log (Base 2) of 'n' | $f$(n) / Time Elapsed (milliseconds) | Log (Base 2) of $f$(n) / "Time Elapsed" |
|---|---|---|---|
| 250 | 8.0 | 1 | 0.0 |
| 500 | 9.0 | 2 | 1.0 |
| 1000 | 10.0 | 3 | 1.6 |
| 2000 | 11.0 | 1 | 0.0 |
| 4000 | 12.0 | 9 | 3.2 |
| 8000 | 13.0 | 11 | 3.5 |
| 16000 | 14.0 | 42 | 5.4 |
| 32000 | 15.0 | 175 | 7.5 |
| 64000 | 16.0 | 784 | 9.6 |
| 128000 | 17.0 | 5299 | 12.4 |
| 256000 | 18.0 | 10555 | 13.4 |
| 512000 | 19.0 | 58004 | 15.8 |
| 1024000 | 20.0 | 198218 | 17.6 |

The line of the log-log scale has a rough slope of 2, due to the fact that there are some imperfect points as the data was produced from a real experiment. Thus the value of $b$ is 2, meaning that as the data size doubles the time required for the algorithm to process it increases by a ratio of 4. Accordingly it can be concluded that the order of growth of "SecretAlgorithm3" is quadratic, $O(n^2)$.

# SecretAlgorithm4




| Data Size 'n' | Log (Base 2) of 'n' | $f$(n) / Time Elapsed (nanoseconds) | Log (Base 2) of $f$(n) / "Time Elapsed" |
|---|---|---|---|
| 250 | 8.0 | 71630 | 16.1 |
| 500 | 9.0 | 1179 | 10.2 |
| 1000 | 10.0 | 973 | 9.9 |
| 2000 | 11.0 | 951 | 9.9 |
| 4000 | 12.0 | 2649 | 11.4 |
| 8000 | 13.0 | 931 | 9.9 |
| 16000 | 14.0 | 978 | 9.9 |
| 32000 | 15.0 | 1021 | 10 |
| 64000 | 16.0 | 1378 | 10.4 |
| 128000 | 17.0 | 1125 | 10.1 |
| 256000 | 18.0 | 1241 | 10.3 |
| 512000 | 19.0 | 1073 | 10.1 |
| 1024000 | 20.0 | 1048 | 10 |
| 2048000 | 21.0 | 1057 | 10 |

| | | | |
|---|---|---|---|
| 4096000 | 22.0 | 1216 | 10.2 |
| 8192000 | 23.0 | 1166 | 10.2 |
| 16384000 | 24.0 | 1084 | 10.1 |
| 32768000 | 25.0 | 1180 | 10.2 |
| 65536000 | 26.0 | 3220 | 11.7 |
| 131072000 | 27.0 | 1323 | 10.4 |
| 262144000 | 28.0 | 1410 | 10.5 |
| 524288000 | 29.0 | 1176 | 10.2 |

The line of the log-log scale has a rough slope of 0, due to the fact that there are some imperfect points as the data was produced from a real experiment. Thus the value of $b$ is 0, meaning that as the data size doubles the time required for the algorithm to process it does not increase. Accordingly it can be concluded that the order of growth of “SecretAlgorithm4” is constant, $O(1)$.