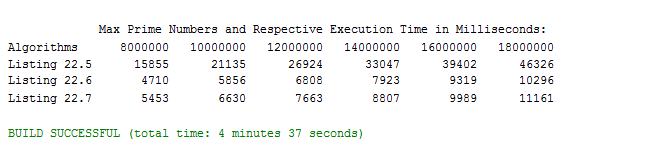
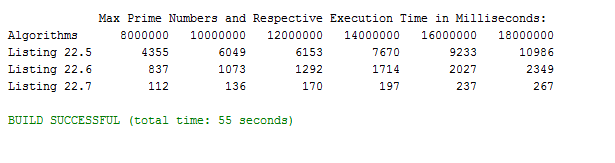
See the Java program for code used to generate these charts. Results may vary depending on your system.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Execution Time (in milliseconds) With Output | | | | | | |
| Max Value: | **8,000,000** | **10,000,000** | **12,000,000** | **14,000,000** | **16,000,000** | **18,000,000** |
| Algorithms |  |  |  |  |  |  |
| Listing 22.5 | 15855 | 21135 | 26924 | 33047 | 39402 | 46326 |
| Listing 22.6 | 4710 | 5856 | 6808 | 7923 | 9319 | 10296 |
| Listing 22.7 | 5453 | 6630 | 7663 | 8807 | 9989 | 11161 |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Execution Time (in milliseconds) Without Output | | | | | | |
| Max Value: | **8,000,000** | **10,000,000** | **12,000,000** | **14,000,000** | **16,000,000** | **18,000,000** |
| Algorithms |  |  |  |  |  |  |
| Listing 22.5 | 4355 | 6049 | 6153 | 7670 | 9233 | 10986 |
| Listing 22.6 | 837 | 1073 | 1292 | 1714 | 2027 | 2349 |
| Listing 22.7 | 112 | 136 | 170 | 197 | 237 | 267 |



# Analysis

When comparing algorithms with respect to execution time and growth rates, the time complexity can be best compared using Big-O notation. When examining time complexity, for simplicity, we can ignore the instructions executed with constant time.

With this in mind, when looking at the code for finding primes using Listing 22.5, the most important lines are the while-loop which executes with a time complexity of O(n) and a nested for-loop which executes with a time complexity of O. Because this loop is nested, the resulting time complexity for the entire routine is O. This explains why Listing 22.5 has such a large growth rate and, if regression methods are used, would be a perfect fit.

Listing 22.6 mostly contains instructions that are all executed with constant time complexity with the exception of a while-loop which executes with a time complexity of O(n). This explains why its growth rate is linear. Listing 22.7 is also linear. When looking at Listing 22.7, there are three for-loops that execute with a time complexity of O(n). However, since none of these for-loops are nested, the time complexity for these instructions are summed, O(3n), which can be simplified to O(n).

Print instructions to the console take a tremendous amount of time, as illustrated in the graph. With the print instructions removed, the time required to execute even the worst algorithm, Listing 22.5, was reduced to be roughly the same as Listing 22.6 and Listing 22.7 with their print statements. Listing 22.6, without its print statements, took less than 3000 ms in every case, and Listing 22.7 dropped to less than 300 ms to perform the same tasks.