Using the Fermat test (requiring n to pass 10 tests with randomly generated values of $a < n$, which yields an error rate of less than $2^{-10} = \frac{1}{256}$ for non-Carmichael numbers), primes close to 10^{10} can be calculated, and rapidly so, in roughly 4 microseconds?, which is about 10 times faster than the most optimized version of the `smallest-divisor` version. To make logarithmic behavior easier to detect, the Fermat test is applied 100 times to each n:

Prime	**Average Time to Pass 100 Fermat Tests**	**Difference**
10^3	12	N/A
10^4	16.444	4.444
10^5	19.333	2.889
10^6	27.778	8.444
10^7	28.889	1.111
10^8	33.222	4.333
10^9	38.889	5.667

The average time seems to grow at a somewhat constant rate, despite the input size (n) increasing tenfold, which would support the claim that the Fermat test has $\Theta(\log n)$ growth.

Let's also use this data to answer the specific question that this exercise posed. Since the Fermat test has $\Theta(\log n)$ growth, we would expect the time to test primes near 1,000,000 to be twice as much as the time to test primes near 1,000.

$$\frac{\log(10^6)}{\log(10^3)} = \frac{6}{3} = 2$$

Using the values from above, the average time at 10^3 is exactly 12 and the average time at 10^6 is approximately 27.778. The time to test primes at 10^6 is $\frac{27.778}{12} \approx 2.315$ times slower than at 10^3, which is close to the 2 times difference we expected.

Note: `random` does not work for numbers bigger than 2^{32}, so so numbers higher than 10^9 cannot be tested.