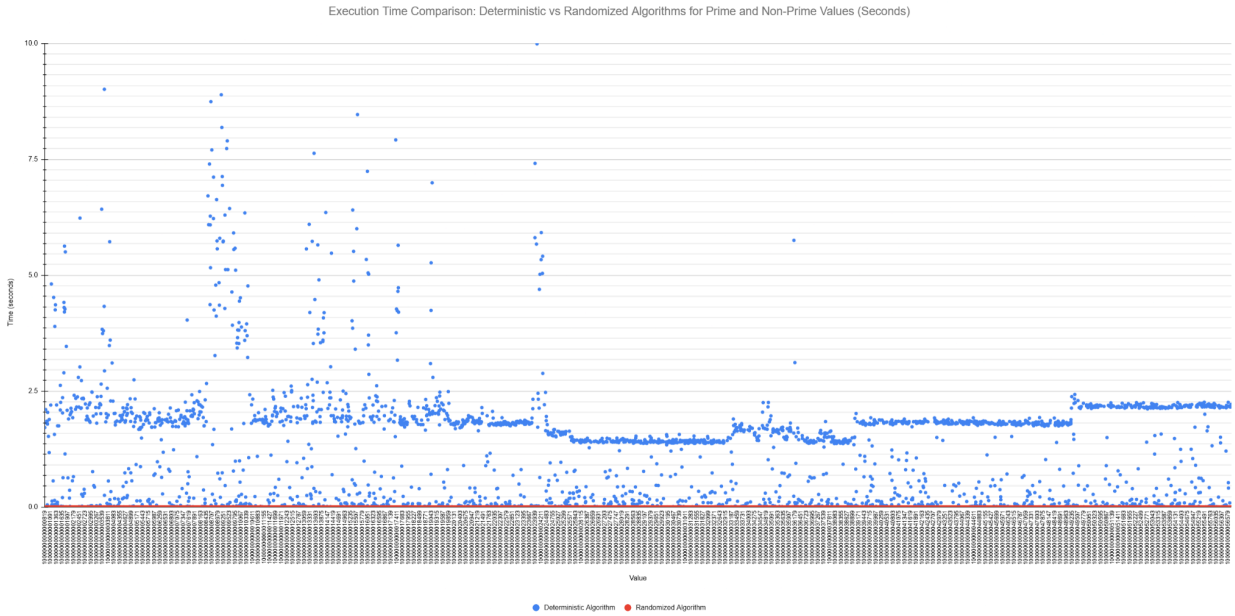
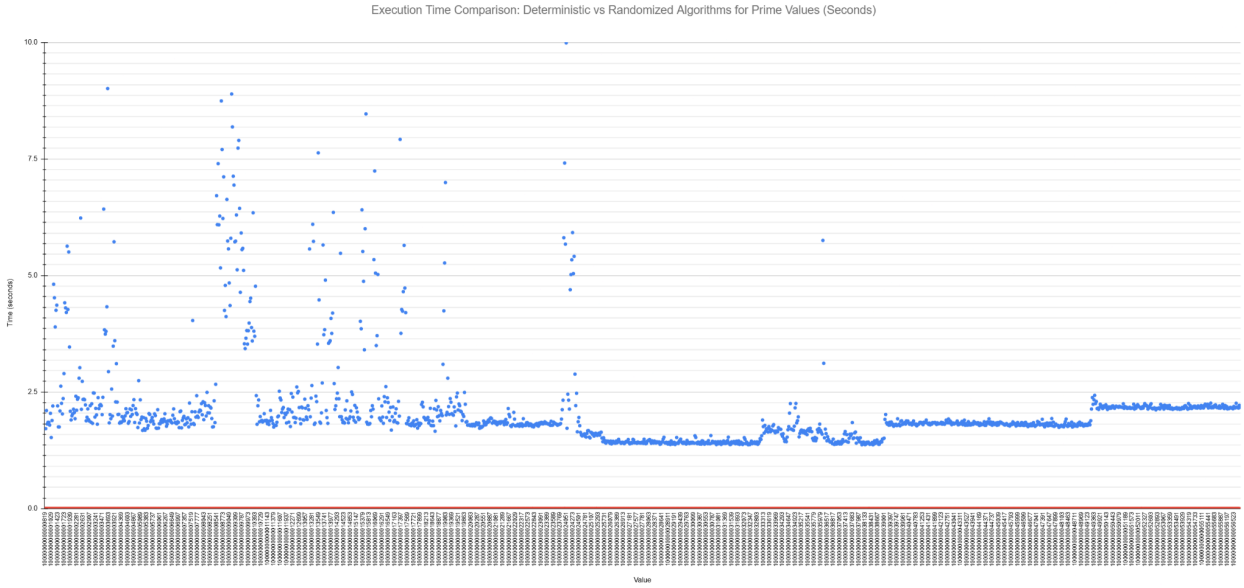
Name: Bryan A. Fernandez
Class: COP4534 U01 1251
Date: 2/16/2025

The assignment compares the performance of deterministic and randomized algorithms for primality testing for one hour.

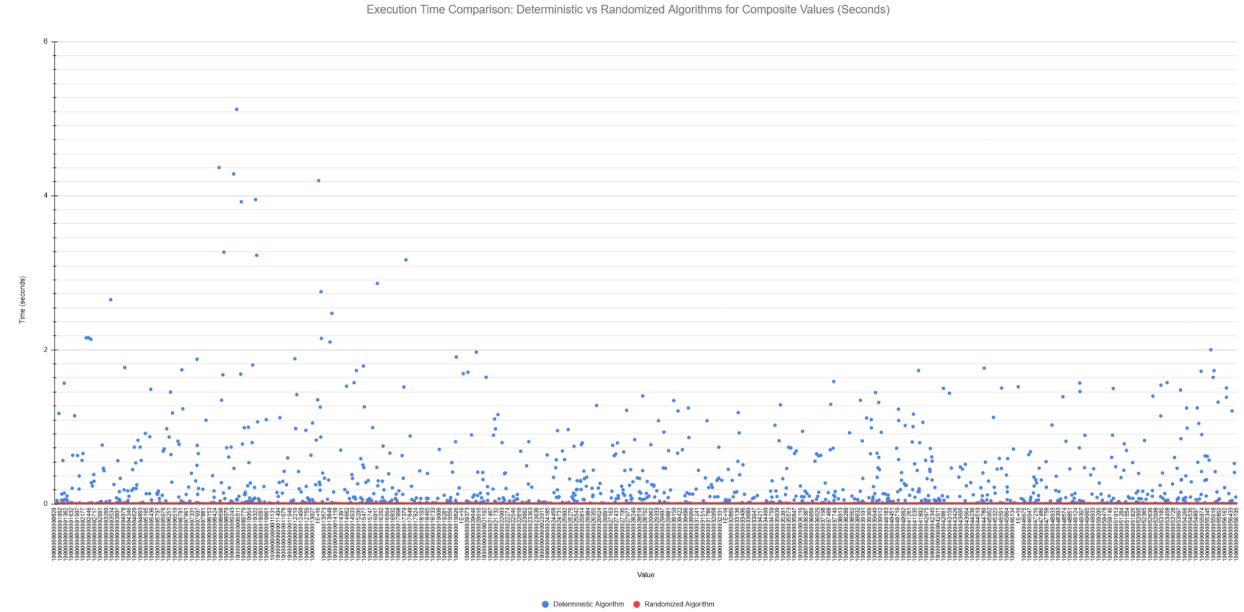
What did you observe in the experiment?
Based on the output, the Randomized Algorithm performed better than the Deterministic Algorithm for prime and composite numbers.



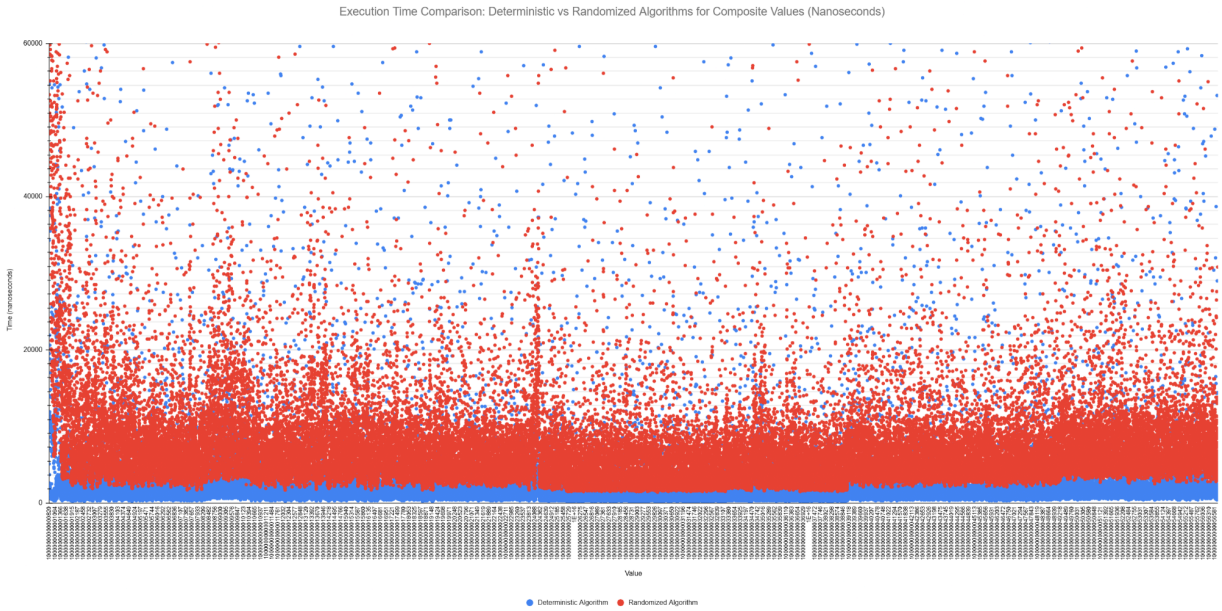
This graph compares how long the deterministic and randomized algorithms take to test both prime and composite numbers. The results make it pretty clear that the randomized algorithm is faster and more consistent. It consistently finished its work in less than a second for every value tested, showing it's really stable and predictable. On the other hand, the deterministic algorithm was much slower and all over the place—some tests took around 2.5 seconds, while others went up to 5.0 seconds or more. This inconsistency seems to come from the fact that the deterministic algorithm's performance depends a lot on the specific number it's testing, especially as the numbers get bigger. So, while the randomized algorithm is quick and reliable, the deterministic one, though accurate, struggles with speed and consistency, especially for larger inputs.



This graph, which compares the performance specifically for prime numbers, highlights the inconsistency between the two algorithms and underscores the impact of primality on their behavior. The Randomized algorithm maintains its consistency, whereas the Deterministic algorithm shows a significant increase in processing time when the input is a prime number. While most of the values cluster around the 2.5-second mark, there are notable outliers, with one extreme value reaching as high as 10 seconds.



This graph, focusing on composite values, further illustrates the inconsistency observed in the deterministic algorithm, while the randomized algorithm remains consistent. However, intriguing patterns emerge when the maximum time on the y-axis is reduced, revealing finer details in the performance trends.



When we zoom in by setting the y-axis max to 60,000 nanoseconds, the differences between the two algorithms become clearer. The randomized algorithm is all over the place, with execution times ranging from almost 0 to 60,000 nanoseconds. Meanwhile, the deterministic algorithm stays pretty tight, mostly hanging out below 10,000 nanoseconds. Both algorithms have a lot of points bunched up at the lower end, meaning most operations finish quickly. But the randomized algorithm has way more outliers that take a lot longer to run. This shows that the deterministic algorithm is usually faster and more reliable, while the randomized one can be all over the map because of its random decisions.

Avg of DA	Avg OF RA
64,099,065.24	9,806.75

Avg of DA (Prime)	Avg of RA (Prime)
2,136,468,507.25	30,442.35

Avg of DA (Composite)	Avg of RA (Composite)
6,822,044.44	9,236.42

On average, the randomized algorithm outperformed the deterministic algorithm. This trend holds true whether the input values are exclusively prime or exclusively composite.

When filtered for prime numbers, the deterministic algorithm's execution time increased by a factor of 33, while the randomized algorithm's time only increased by a factor of 3. This indicates that prime numbers had a significantly greater impact on the performance of the deterministic algorithm compared to the randomized one. For composite values, the deterministic algorithm's

execution time decreased by a factor of 9, whereas the randomized algorithm's time decreased by just 600 nanoseconds. These results further highlight the differing sensitivities of the two algorithms to the nature of the input values.

Were there false positives? Which ones?

	# of Prime Numbers	# of Composite Numbers
DA	1504	54417
RA	1504	54417

Unexpectedly, the data shows that the randomized algorithm did not produce any false positives, as it identified the same 1504 prime numbers and 54417 composite numbers as the deterministic algorithm. This outcome is likely due to the test (based on Fermat's Primality Test) running multiple iterations (10), significantly reducing the chances of misclassifying a composite number as prime. Additionally, the results suggest that no Carmichael numbers were present in the tested range, which further minimized the risk of false positives.

Conclusion

In conclusion, the experiment showed that the randomized algorithm was not only faster but also way more consistent than the deterministic one, especially when dealing with large numbers. When zoomed in at 60,000 nanoseconds and only factoring in composite values, the deterministic algorithm actually performed better in that small range because it followed a structured approach, avoiding the randomness that sometimes slows things down. Still, the randomized algorithm dominated overall, handling every test quickly and without false positives. While the deterministic method was reliable, it struggled with bigger numbers, making it less practical for large-scale testing.