Efficient construction of infinitely many canonical primes

Bogdan Grechuk • 5 May 2026

The recently published April 2026 issue of the *Journal of the ACM* contains a paper by Chen, Lu, Oliveira, Ren, and Santhanam (Chen et al. 2026), in which the authors present a “pseudodeterministic” construction of prime numbers in the infinitely-often regime.

An improtant algorithmic task in number theory is the task of constructing primes. Given a positive integer n as input, we would like an algorithm that runs in time polynomial in n and returns an n-digit prime. There is an obvious randomized algorithm for this task: generate random n-digit integers, test them for primality (this can be done in polynomial time (Agrawal et al. 2004)), and output the first prime found. Since the proportion of primes among n-digit integers is about c/n for some constant c, it is easy to check that, with high probability, only polynomially many integers need to be tested. The drawback is that different runs of this algorithm on the same input n are likely to return different primes.

A major open problem is to develop a deterministic polynomial-time algorithm for this task. Such an algorithm would, for each n, return a specific prime p_n. A well-believed conjecture of Cramér states that there is a constant C such that every sequence of Cn^2 consecutive n-digit positive integers contains a prime. If this conjecture is true, then one can simply test the first n-digit integers in order until the first prime is found, and the running time will be polynomial in n. However, Cramér’s conjecture is very difficult, and the current best upper bound on the gap between consecutive primes (Baker et al. 2001) is exponential in the number of digits.

Motivated by this difficulty, Gat and Goldwasser (Gat and Goldwasser 2011) introduced the concept of *pseudodeterministic algorithms*. These are randomized algorithms that, with high probability, return the same correct answer on each fixed input. The main open question raised in (Gat and Goldwasser 2011) asks whether such an algorithm exists for prime construction: for each given n, it should run in time polynomial in n and, with high probability, output a specific n-digit prime p_n. In 2026, Chen, Lu, Oliveira, Ren, and Santhanam (Chen et al. 2026) proved that this can be done for infinitely many input lengths.

Theorem 1 *There exists a sequence $p_1, p_2, \ldots$, where p_n is an n-digit prime for every n, an infinite set A of positive integers, and a randomized algorithm that, given any $n \in A$ as input, runs in time polynomial in n and outputs p_n with probability at least $1 - 2^{-n}$.*

The proof of Theorem 1 applies not only to prime generation, but more generally to any problem whose set of valid outputs is both dense and easy to recognize. Here *dense* means that the proportion of correct outputs among strings of length n is at least $n^{-\rho}$ for some constant $\rho > 1$, while *easy to recognize* means that there is a deterministic polynomial-time algorithm for checking whether a proposed output is correct. Prime numbers satisfy these two conditions: they are dense by the prime number theorem and easy to recognize by (Agrawal et al. 2004). Of course, the set of all primes is not the only set with these properties. For example, an analogue of Theorem 1 holds for generating primes congruent to a modulo b, for any fixed coprime integers a and b.

References

Agrawal, Manindra, Neeraj Kayal, and Nitin Saxena. 2004. “PRIMES Is in P.” *Ann. of Math.* 160 (2): 781–93.

Baker, Roger C., Glyn Harman, and János Pintz. 2001. “The Difference Between Consecutive Primes, II.” *Proc. Lond. Math. Soc.* 83 (3): 532–62.

Chen, Lijie, Zhenjian Lu, Igor Oliveira, Hanlin Ren, and Rahul Santhanam. 2026. “Polynomial-Time Pseudodeterministic Construction of Primes.” *Journal of the ACM* (New York, NY, USA) 73 (2). https://doi.org/10.1145/3803408.

Gat, Eran, and Shafi Goldwasser. 2011. “Probabilistic Search Algorithms with Unique Answers and Their Cryptographic Applications.” *Electron. Colloquium Comput. Complex.* 18: 136.