CONSTRUCTIVE PRIME GENERATION VIA DYNAMIC SIEVE METHODS

PU JUSTIN SCARFY YANG

Abstract. We present a constructive, dynamic sieve-based algorithm that incrementally generates the sequence of prime numbers starting from 2. In contrast to classical sieve methods such as the Sieve of Eratosthenes, our method does not require a pre-defined upper bound and produces each prime in order, using only previously discovered primes. We prove the algorithm's correctness and completeness, compare it to known approaches, and discuss its philosophical implications and potential extensions to non-standard number systems such as the Yang_n framework.

Contents

1. Introduction

The prime numbers are the fundamental building blocks of the integers. Classical sieve methods such as the Sieve of Eratosthenes have played a foundational role in identifying primes up to a fixed bound. However, such methods are not inherently incremental or infinite in their generation. This paper proposes a dynamic, constructive sieve-based method to generate primes one-by-one, without requiring any upper limit.

Our goal is to develop a systematic algorithm that starts from the initial primes and uses them to determine the next prime, mimicking a sieve while maintaining complete generality and minimal storage.

2. Related Work

Historically, several prime generation techniques have been studied:

- The classical *Sieve of Eratosthenes*, attributed to the ancient Greeks, efficiently eliminates non-primes up to a fixed upper bound.

Date: May 15, 2025.

- Trial division methods test the primality of a candidate number using known smaller primes.
- More recent work includes *wheel sieves* and *priority queue sieves*, which aim to optimize space and time usage in incremental generation [2, 1].
- In functional programming, O'Neill's lazy sieve demonstrates how infinite streams of primes can be generated using Haskell's laziness.

Unlike these, our method emphasizes mathematical clarity, minimalism, and extensibility.

3. The Constructive Sieve Algorithm

Definition 1 (Dynamic Constructive Sieve)**.** *Let $\mathcal{P}_n = \{p_1, p_2, \dots, p_n\}$ be the first n prime numbers. Define the candidate set:*

$$S_n := \left\{ m \in \mathbb{Z}_{>p_n} \,\middle|\, \forall\, p \in \mathcal{P}_n,\, p^2 \le m \Rightarrow p \nmid m \right\}.$$

Then the $(n+1)$-th prime number is:

$$p_{n+1} := \min S_n.$$

Theorem 1 (Correctness)**.** *The value p_{n+1} obtained by this method is prime.*

Proof. Suppose $m = p_{n+1}$ is not prime. Then m has a factor a with $1 < a \le \sqrt{m}$. Since all primes less than or equal to $\sqrt{m}$ are in $\mathcal{P}_n$, a divides m contradicting the condition. Hence m is prime. □

Theorem 2 (Completeness)**.** *Every prime number appears eventually in the sequence $\{p_1, p_2, \dots\}$ generated by the algorithm.*

Proof. Suppose not. Let q be the smallest prime not appearing in the generated sequence. All primes less than q have been generated. When q is reached, it passes the sieve test, and so must be appended—contradiction. Thus, no prime is missed. □

4. Algorithm Implementation

A high-level pseudocode version of the algorithm is:

```
P := [2]
candidate := 3
while true:
    is_prime := true
    for p in P:
        if p * p > candidate:
            break
        if candidate % p == 0:
            is_prime := false
            break
    if is_prime:
        P.append(candidate)
        yield candidate
    candidate := candidate + 2
```

5. Philosophical and Mathematical Extensions

This algorithm provides a constructive foundation for defining the prime sequence. Beyond classical arithmetic, it opens potential for:

- Generalizations to non-standard number systems, e.g., Yang_n
- Studying analogues of primality in categorical, non-commutative, or algebraic geometry frameworks
- Exploring the role of minimal generation principles in model-theoretic settings

6. Conclusion and Future Work

We have defined and proven the correctness of a simple yet powerful dynamic sieve-based algorithm that constructs the sequence of primes. We aim to further extend this model toward generalized algebraic systems, and explore formalization in Lean, Coq, and homotopy type theory frameworks.

References

[1] Melissa O'Neill, *The Genuine Sieve of Eratosthenes*, Journal of Functional Programming, 15(4): 415–433, 2004.

[2] Richard Bird, *Pearls of Functional Algorithm Design*, Cambridge University Press, 2010.