An Efficient Deterministic Primality Test

[Working Draft]

Joseph M. Shunia

December 2023
Revised: January 2024

Abstract

We present a deterministic primality test with polynomial time complexity $\tilde{O}(\log^4(n))$. The basis for our test is a variant of Agrawal's conjecture, a prominent yet unresolved conjecture in primality testing. Agrawal's conjecture states that an integer n is prime if and only if the polynomial congruence $(x-1)^n \equiv x^n - 1 \pmod{n}$ holds in the quotient ring $\mathbb{Z}[x]/(x^r - 1)$, for some r satisfying $n^2 \not\equiv 1 \pmod{r}$. We modify the conjecture, and exploit the number-theoretic properties of Fermat pseudoprimes and $\mathcal{D}$-th power residues to establish an efficient deterministic primality test.

1 Introduction

Primality testing has seen remarkable advancements over the past few decades. A significant breakthrough in this field was the AKS primality test, introduced by Agrawal, Kayal, and Saxena (2002) [1]. The AKS test was the first to offer determinism and polynomial-time complexity, a monumental achievement that resolved a longstanding open question in computational number theory [2]. However, despite its theoretical importance, the AKS test has practical limitations due to its relatively high polynomial time complexity, rendering it inefficient for most applications. Agrawal, Kayal, and Saxena gave a time complexity of $\tilde{O}(\log^{12}(n))$ for the AKS test [1]. This bound was lowered significantly by Lenstra and Pomerance (2011) to $\tilde{O}(\log^6(n))$ [3]. Despite this reduction, AKS remains impractical and is mostly unused.

In the field of cryptography, the unique properties of prime numbers are widely exploited to create cryptographic primitives. It is often the case that many large primes must be generated in rapid succession [4]. To make these cryptographic operations practical, fast probabilistic primality tests such as the Baille-PSW primality test (BPSW) [5] or Miller-Rabin (MR) [6] [7] are used instead of AKS when searching for large primes. Probabilistic primality tests are by definition non-deterministic and may erroneously report a composite integer as being prime. Composite integers which pass a probabilistic primality test are relatively rare and are known as pseudoprimes (PSPs) for the respective test [8]. When generating primes for cryptographic purposes, probabilistic primality tests are often combined or repeated with different parameters in order to achieve an acceptable error-bound that makes it almost certain that no composite integer will pass. However, reducing the error-bound requires additional compute and increases running-time, creating a trade-off.

We present a deterministic primality test with polynomial time complexity $\tilde{O}(\log^4(n))$. This efficiency gain opens new avenues for practical applications, particularly in cryptography, where fast and reliable primality testing is desirable [9].

Our test is based on a famous conjecture issued by Manindra Argawal while he was an undergraduate student [1]. Let n and r be two coprime positive integers. If the following polynomial congruence holds, then n is prime or $n^2 = 1 \pmod{r}$:

$$(x-1)^n \equiv x^n - 1 \pmod{n, x^r - 1} \tag{1}$$

We alter the conditions slightly to leverage specific number theoretic properties unique to Fermat pseudoprimes to base 2 (See Definition 1) to construct a deterministic primality test.

Through the proof of our main theorem, we demonstrate that our modified test is equivalent to checking the polynomial congruence $(1+x)^n \equiv 1 + x^n \pmod{n} \in \mathbb{Z}[x]$, which is known to hold for only prime integers n [10].

1.1 Structure of the Paper

This paper is structured as follows: We begin by presenting the main theorem which defines our primality test. We follow up with supporting lemmas and theorems. Then, we present the proof of our main theorem, which demonstrates the test's validity for odd prime numbers and its failure for odd composite numbers. Through this, we establish the deterministic nature of our test. We then describe the algorithm used to compute our test and analyze its computational complexity. We conclude with a link to an open source implementation of our test.

2 Preliminaries and Definitions

Definition 1 (Fermat pseudoprime to base b)**.** Let $n, b \in \mathbb{Z}$ be coprime with n composite. We say that n is a Fermat pseudoprime to base b iff $b^{n-1} \equiv 1 \pmod{n}$.

Remark. When n is a prime integer, by Fermat's Little Theorem, the congruence holds trivially. Conversely, when n is composite, the congruence holds only rarely. For this reason, checking if n is a Fermat pseudoprime to a handful of small prime bases up to a limit that is logarithmic in n (e.g. $b = \{2, 3, 5, \ldots \log_2(n)\}$) makes for a decent primality test. The vast majority of composite n are quickly (and efficiently) ruled out. However, a problem presents: somewhat surprisingly, there exist composite n which pass the test to all bases b coprime to n. We call such composite n "Carmichael numbers" [11] and unfortunately, they are infinitely many [12]. A rigorous treatment of Carmichael numbers is outside the scope of this paper, but it is important to be aware that such composite integers exist.

Definition 2 (Wieferich primes)**.** The Wieferich primes, sequence A001220 [13] in the Online Encyclopedia of Integer Sequences (OEIS), is the sequence of prime integers p such that:

$$2^{p-1} \equiv 1 \pmod{p^2} \tag{2}$$

Remark. Denote by W the sequence of Wieferich primes. As of the year 2024, only two Wieferich primes are known:

$$W = \{1093, 3511\} \tag{3}$$

Computational searches up to 2^{64} have not found any additional Wieferich primes [14], and it is an open question whether the sequence is finite or infinite. However, Carella (2018) [15] established that the subset of non-Wieferich primes has asymptotic density 1 within the set of all primes, implying that Wieferich primes must be exceedingly rare.

Carella (2018) [15] also gave an upper bound on the number of Wieferich primes less than n. Let $W_{\#}(n)$ denote the count of Wieferich primes up to n, then we have:

$$W_{\#}(n) \leq 2 \cdot \log\log(n), \tag{4}$$

This upper bound suggests that if it happens to be true that there are infinitely many Wieferich primes, then they constitute only a vanishingly small fraction of all primes.

3 Statement of Main Theorem

Theorem 1 (Main theorem)**.** *Let $n > 3$ be an odd integer such that $2^{n-1} \equiv 1 \pmod{n}$. Let $r > 2$ be the least prime integer such that $n \not\equiv 1 \pmod{r}$. If the following polynomial congruence holds, then either n is prime or n has a prime divisor $p \leq r$:*

$$(1+x)^n \equiv 1 + x^n \pmod{n, x^r - 1} \tag{5}$$

4 Supporting Lemmas and Theorems

4.1 Supporting Lemmas

Lemma 1 (Upper bound on floor function and indivisibility)**.** *Let $a, b \in \mathbb{Z}^+$. Suppose $1 < b < \left\lfloor \frac{a}{b} \right\rfloor$. Then $b \leq \lfloor \sqrt{a} \rfloor$.*

Proof. We are given $a, b \in \mathbb{Z}^+$ such that $1 < b < \left\lfloor \frac{a}{b} \right\rfloor$. By definition, $\left\lfloor \frac{a}{b} \right\rfloor$ is the greatest integer less than or equal to $\frac{a}{b}$. Hence, $\left\lfloor \frac{a}{b} \right\rfloor \leq \frac{a}{b}$. Since $b < \left\lfloor \frac{a}{b} \right\rfloor$, we have $b^2 < b \cdot \left\lfloor \frac{a}{b} \right\rfloor \leq a$. Taking square roots on both sides of the inequality $b^2 < a$ and considering that b and $\sqrt{a}$ are both positive, we get $b < \sqrt{a}$. Since b and $\sqrt{a}$ are positive integers, and $b < \sqrt{a}$, it follows that $b \leq \lfloor \sqrt{a} \rfloor$. □

Lemma 2. *Let $a, b \in \mathbb{Z}^+$. Suppose $b \nmid a$ and $1 < b < \left\lfloor \frac{a}{b} \right\rfloor$. Then $\left\lfloor \frac{a}{b} \right\rfloor \nmid a$.*

Proof. Let $q = \left\lfloor \frac{a}{b} \right\rfloor$. By definition, q is the greatest integer which is less than $\frac{a}{b}$. Thus, $q \cdot b < a < b \cdot (q+1)$.

Suppose, for contradiction, that $q \mid a$. Then there exists an integer k such that $a = k \cdot q$. Substituting $a = k \cdot q$ into the inequality $q \cdot b < a < b \cdot (q+1)$, we get $q \cdot b < k \cdot q < b \cdot (q+1)$. Dividing this inequality by q, we obtain $b < k < (b + \frac{b}{q})$.

Since k is an integer, and $b \nmid a$ implies $k \neq b$, the next possible integer value for k is $b + 1$. Therefore, $k = b + 1$, which gives $a = k \cdot q = q \cdot (b+1)$. However, this leads to a contradiction: $a = q \cdot (b+1)$ implies $a \geq b \cdot (q+1)$, contradicting the established fact that $a < b \cdot (q+1)$. Hence, our assumption that q divides a is false. Therefore, $\left\lfloor \frac{a}{b} \right\rfloor \nmid a$. □

Lemma 3 (Least prime p coprime to n)**.** *Let $n \in \mathbb{Z}^+$ such that $n > 6$. There exists a prime p coprime to n, such that:*

$$p < \log(n) \left(\log\log(n) + \log\log\log(n)\right)$$

Proof. We are given $n > 6$. By the Prime Number Theorem, n must have fewer than $\log(n)$ distinct prime factors. Hence, we need to establish an upper bound on the $\log(n)$-th prime number.

Let p_n denote the n-th prime number. Then for $n \geq 6$, we have the inequality [16]:

$$\frac{p_n}{n} < \log(n) + \log\log(n) \tag{6}$$

Re-arranging terms:

$$p_n < n\log(n) + n\log\log(n) \tag{7}$$

We want to find an upper bound for $p_{\log(n)}$, the $\log(n)$-th prime number. Substituting $\log(n)$ for n in the inequality, we get:

$$p_{\log(n)} < \log(n)\,(\log\log(n) + \log\log\log(n)) \tag{8}$$

This expression gives an upper bound for the smallest prime p that is coprime to n. □

Corollary 1. Let $n > 6$ be an integer. Then there exists a prime integer $r \leq \tilde{O}(log(n))$ such that $r \nmid (n-1)$.

Proof. By Lemma 3, there exists a prime integer r coprime to $n-1$, satisfying:

$$r < \log(n-1)\,(\log\log(n-1) + \log\log\log(n-1))$$

Considering the soft-O notation, which simplifies the time complexity by ignoring sublogarithmic factors, we have:

$$\tilde{O}(\log(n-1)) \geq \log(n-1)\,(\log\log(n-1) + \log\log\log(n-1))$$

It follows that this bound can be expressed as $\tilde{O}(\log(n))$, indicating the existence of an algorithm capable of finding r within this time complexity. □

Corollary 2. *Let $n > 3$ be an odd composite integer. There exists a prime integer r, such that for some prime divisor p of n, $(p-1) \not\equiv 0 \pmod{r}$. Furthermore, an algorithm can determine such a prime r in $\tilde{O}(\log(n))$ time.*

Proof. Let p be a prime divisor of n. By Lemma 3, there exists a prime integer r coprime to $p-1$, satisfying $r < \log(p-1)\,(\log\log(p-1) + \log\log\log(p-1))$. Considering the soft-O notation, which simplifies the time complexity by ignoring sublogarithmic factors, the upper bound for finding such a r is $\tilde{O}(\log(p-1))$. Since $p \leq \lfloor \frac{n}{3} \rfloor$, it follows that this bound can be expressed as $\tilde{O}(\log(n))$, indicating the existence of an algorithm capable of finding r within this time complexity. □

Corollary 3. *Let $n > 3$ be an odd composite integer. There exists a prime integer r, such that for some prime divisor p of n, 1 is not a non-trivial r-th power residue modulo p. That is, there are no solutions $a^r \equiv 1 \pmod{p}$ for $a \in \mathbb{Z}_p$ such that $a \neq 1$. Furthermore, an algorithm can determine such a prime r in $\tilde{O}(\log(n))$ time.*

Proof. By Corollary 2, we can find a r such that $(p-1) \not\equiv 0 \pmod{r}$ for some prime p dividing n in $\tilde{O}(\log(n))$ time. By Fermat's Little Theorem, $a^r \equiv 1 \pmod{p}$ has a solution in the integers iff $r \mid (p-1)$. Hence, the proof follows trivially. □

Lemma 4. *Let n be an odd composite integer such that $n > 3$ and $2^{n-1} \equiv 1 \pmod{n}$. Then, there exists a prime integer r such that $2^{\left\lfloor \frac{p^e-1}{r} \right\rfloor} \not\equiv 1 \pmod{p^e}$ for some prime power divisor p^e of n. Furthermore, an algorithm can determine such a prime r in $\tilde{O}(\log(n))$ time.*

Proof. Consider an odd composite integer $n > 3$ such that $2^{n-1} \equiv 1 \pmod{n}$. According to the properties of the multiplicative order modulo n, the least $k \in \mathbb{Z}^+$ such that $2^k \equiv 1 \pmod{n}$ defines $\mathrm{ord}_n(2)$. That is, $\mathrm{ord}_n(2) = k$. In this case, $2^{n-1} \equiv 1 \pmod{n} \Rightarrow \mathrm{ord}_n(2) \mid (n-1)$.

Let r be the least prime integer such that $2 < r < n$ and $r \nmid (n-1)$. By Corollary 1, we can find such r in $\tilde{O}(log(n))$ time.

We have $r \nmid (n-1)$. Conversely, we also have $\mathrm{ord}_n(2) \mid (n-1)$. Hence, we deduce $r \neq \mathrm{ord}_n(2)$.

Since $r < \lfloor \sqrt{n} \rfloor$, we have $r < \left\lfloor \frac{n-1}{r} \right\rfloor$ (By Lemma 1). We also have $r \nmid (n-1)$ and $1 < r < \left\lfloor \frac{n-1}{r} \right\rfloor$, which collectively imply $\left\lfloor \frac{n-1}{r} \right\rfloor \nmid (n-1)$ (By Lemma 2).

Finally, since $\left\lfloor \frac{n-1}{r} \right\rfloor < (n-1)$ and $r \nmid (n-1)$, we deduce $2^{\left\lfloor \frac{n-1}{r} \right\rfloor} \not\equiv 1 \pmod{n}$. By the Chinese Remainder Theorem (CRT) [17], there must exist a prime power $p^e \mid n$ such that $2^{\left\lfloor \frac{p^e-1}{r} \right\rfloor} \not\equiv 1 \pmod{p^e}$. □

Lemma 5. *Let n be an odd composite integer such that $n > 3$ and $2^{n-1} \equiv 1 \pmod{n}$. Then, there exists a prime integer r such that $2^{\left\lfloor \frac{p-1}{r} \right\rfloor} \not\equiv 1 \pmod{p}$ for some prime divisor p of n. Furthermore, an algorithm can enumerate such a prime r in $\tilde{O}(\log(n))$ time.*

Proof. Consider an odd composite integer $n > 3$ such that $2^{n-1} \equiv 1 \pmod{n}$. By Lemma 4, there must exist a prime integer r, can be determined in $\tilde{O}(\log(n))$ time, such that $2^{\left\lfloor \frac{p^e-1}{r} \right\rfloor} \not\equiv 1 \pmod{p^e}$, where p^e is a prime power dividing n.

For each prime divisor p of n, we have $\phi(p) = p - 1$. By the Prime Number Theorem (PNT), $p - 1$ itself can have at most $\log(p)$ distinct prime divisors. Hence, for a given p, it is impossible to have $2^{\left\lfloor \frac{p-1}{r_i} \right\rfloor} \equiv 1 \pmod{p}$ for more than $\log(n)$ distinct primes r_i.

While we cannot find the specific r_i explicitly, it is clear that there must exist such a D_i within the first $\log(n) + 1$ primes (since otherwise $p - 1$ would violate the PNT). Thus, it follows that there must exist a prime $p \mid n$ such that $2^{\left\lfloor \frac{p-1}{r} \right\rfloor} \not\equiv 1 \pmod{p}$ which can be enumerated in $\tilde{O}(\log(n))$ time. □

4.2 Primes Case

Theorem 2 (Primes pass)**.** *Let n be an odd prime integer such that $n > 3$. Let $r > 2$ be the least prime integer such that $n \not\equiv 1 \pmod{r}$. Then $(1 + x)^n \equiv 1 + x^n \pmod{n, x^r - 1}$.*

Proof. We aim to show that the following polynomial congruence holds for all odd prime integers $n > 3$:

$$(1 + x)^n \equiv 1 + x^n \pmod{n, x^r - 1}, \tag{9}$$

where r is the least prime integer greater than 2 such that $n \not\equiv 1 \pmod{r}$.

We begin by taking the binomial expansion of the left-hand side:

$$\sum_{k=0}^{n} \binom{n}{k} x^k \equiv 1 + x^n \pmod{n, x^r - 1} \tag{10}$$

When n is prime, $\binom{n}{k} \equiv 0 \pmod{n}$ for all integers k in the range $1 \leq k \leq n-1$. Thus, we have $\sum_{k=1}^{n-1} \binom{n}{k} x^k \equiv 0 \pmod{n, x^r - 1}$. With this in mind, we isolate the inner terms in our original congruence and simplify:

$$\binom{n}{0} x^0 + \binom{n}{n} x^n + \sum_{k=1}^{n-1} \binom{n}{k} x^k \equiv 1 + x^n \pmod{n, x^r - 1} \tag{11}$$

$$\binom{n}{0} x^0 + \binom{n}{n} x^n \equiv 1 + x^n \pmod{n, x^r - 1} \tag{12}$$

By the binomial theorem, we have $\binom{n}{0} = \binom{n}{n} = 1$ for all $n \in \mathbb{Z}$. Substituting values and simplifying further reveals:

$$1 \cdot x^0 + 1 \cdot x^n \equiv 1 + x^n \pmod{n, x^r - 1} \tag{13}$$

$$1 + x^n \equiv 1 + x^n \pmod{n, x^r - 1} \tag{14}$$

Hence, we have shown that the left-hand side of the congruence is equal to the right-hand side when n is prime. Therefore, we conclude that the polynomial congruence holds under the given conditions. □

4.3 Composites Case

A fundamental theorem in polynomial ring theory states that an integer n is prime if and only if $(1+x)^n \equiv 1 + x^n \pmod{n} \in \mathbb{Z}[x]$ [10]. This congruence was used as the basis for the AKS test [1]. A short proof of the theorem, given by Granville (2004) [10], is that since $(x+1)^n - (x^n+1) = \sum_{k=1}^{n-1} \binom{n}{k} x^k$, we may have $(1+x)^n \equiv 1 + x^n \pmod{n}$ if and only if n divides $\binom{n}{k}$ for all k in the range $1 \leq k \leq n-1$. It is important to note that for the polynomial congruence to be valid, x^n and $(1+x)^n$ must be irreducible in $\mathbb{Z}_n[x]$.

Kopparty and Wang (2014) [18] proved several interesting theorems related to limits on the counts of consecutive zero coefficients in polynomials over finite fields. We take inspiration from their approach to show that composite integers will always fail our test.

Theorem 3 (Primality test)**.** *Let $n = p \cdot q$ be a composite integer greater than 3, with p a prime divisor. Let $r \in \mathbb{Z}$ be prime. Suppose 1 is not a non-trivial r-th power modulo p and n does not have a prime factor less than or equal to r. Then, $(1+x)^n \not\equiv 1 + x^n \pmod{n, x^r - 1}$.*

Proof. **Preliminaries:**

- Define the polynomial $f(x) := (1+x)^n - (1+x^n)$.
- Define the polynomial $g(x) := f(x) \pmod{x^r - 1}$.

Step 1. Establishing irreducibility and field structure:

Given p is prime, $\mathbb{Z}_p$ is a finite field and $\mathbb{Z}_p[x]$ is a polynomial ring over this field. We aim to establish that $\mathbb{Z}_p[x]/(x^r - 1)$ forms a field. To do so, we must show that $x^r - 1$ is irreducible in $\mathbb{Z}_p[x]$.

We reference a classical theorem from field theory (Irreducibility Theorem) [19]:

Suppose $c \in F$ where F is a field, and $0 < k \in \mathbb{Z}$. The polynomial $x^k - c$ is irreducible over F if and only if c is not a jth power in F for any prime j dividing k, and c is not in $-4F^4$ when 4 divides k.

In our case, $F = \mathbb{Z}_p$, $c = 1$, and $k = r$, where r is prime.

Regarding the first criterion, given r is prime, r is divisible by only itself. We have chosen p such that 1 is not a non-trivial r-th power residue modulo p, which informs that there is no element $e(x) \in F$ such that $e(x)^r \equiv 1 \pmod{p}$ and $e(x) \neq 1$. The first criterion is satisfied. Since r is prime, it is not divisible by 4, and thus second criterion is also satisfied.

By the Irreducibility Theorem, $x^r - 1$ is irreducible in $\mathbb{Z}_p[x]$ and hence, the quotient ring $\mathbb{Z}_p[x]/(x^r - 1)$ forms a finite field.

Step 2. Analyzing the reduction of $f(x)$ modulo $x^r - 1$:

We examine the reduction of $g(x) = f(x) \pmod{x^r - 1} \in \mathbb{Z}[x]$. After reduction modulo $x^r - 1$, the polynomial $g(x)$ has $\deg(g(x)) = r - 1$, and can be written as:

$$g(x) = \sum_{i=0}^{r-1} c_i x^i \tag{15}$$

To justify this: We first look to the expansion of $f(x) = (1+x)^n - (1+x^n) \in \mathbb{Z}[x]$:

$$f(x) = \sum_{k=1}^{n-1} \binom{n}{k} x^k \tag{16}$$

Notice that subtracting $1 + x^n$ from $(1+x)^n$ cancels out the terms $\binom{n}{0}x^0 = 1$ and $\binom{n}{n}x^n = x^n$ that would typically be present in the binomial expansion of $(1+x)^n$. Thus, we have $\deg(f(x)) = n - 1$.

Reducing $f(x)$ modulo $x^r - 1$ means replacing every term of the form $\binom{n}{k}x^k$ for $k \geq r$ with a lower-degree term, using the relation $x^r = 1$. During this reduction, terms in $f(x)$ with degree $1 \leq k < r$ will retain their degrees, as they are unaffected by the modulo operation. Since $r < n$, these terms are always present in the binomial expansion. This suggests $f(x)$ has $r - 1$ fixed terms when taken $x^r - 1$, with the highest-degree term among them being $\binom{n}{r-1}x^{r-1}$. Since the highest possible degree of any reduced terms is also $r - 1$, the degree of $g(x)$ remains $r - 1$ after the reduction of any additional terms.

The coefficients of these terms will be "wrapped" around $x^r - 1$ and added to the fixed term which corresponds to the value of their degree k, which is the term with the variable $x^{k \pmod r}$. Therefore, after the reduction of $f(x)$ modulo $x^r - 1$, the resultant polynomial $g(x)$ will have r polynomial terms. And since $f(x) \not\equiv 0 \pmod{x^r - 1}$, $g(x)$ is not the zero polynomial.

In summary, since x^r does not divide $f(x)$ in $\mathbb{Z}[x]$, $g(x)$ is nonzero and has a degree of $r - 1$.

Step 3. Confirming nonzero polynomial in quotient ring:

We look to the quotient ring $\mathbb{Z}_p[x]/(x^r - 1)$, which forms a finite field (See Step 1).

In Step 2, we showed that $g(x) = f(x) \pmod{x^r - 1}$ is nonzero in $\mathbb{Z}[x]$ with $\deg(g(x)) = r - 1$.

We aim to show that $g(x)$ is nonzero in $\mathbb{Z}_p[x]/(x^r - 1)$.

Now, assume for contradiction that $g(x)$ is the zero polynomial in $\mathbb{Z}_p[x]/(x^r - 1)$. This would necessarily imply that all coefficients c_i of $g(x) = \sum_{i=0}^{r-1} c_i x^i$ are zero when taken modulo p, where the c_i are aerated sums of binomial coefficients.

In a finite field, a polynomial of degree r can have at most r roots [20]. This is because a polynomial of degree r in a finite field can be factored into at most r linear factors (each corresponding to a root), within an algebraic closure of that field.

In our case, since $\mathbb{Z}_p[x]/(x^r - 1)$ is a finite field and $\deg(g(x)) = r - 1$, then $g(x)$ can have at most $r - 1$ roots in this field. The assumption that $g(x)$ is the zero polynomial is a direct contradiction unless $p \leq r - 1$, as it would necessarily imply that $g(x)$ has p roots. Clearly, this is not the case, as we are given n which does not have a prime divisor $\leq r$.

Therefore, $f(x)$ cannot be identically zero in $\mathbb{Z}_p[x]/(x^r - 1)$ and we conclude $(1 + x)^n \not\equiv 1 + x^n \pmod{p, x^r - 1}$. □

5 Proof of the Main Theorem

Proof of Theorem 1. Let n be an odd integer > 3 such that $2^{n-1} \equiv 1 \pmod{n}$. Let $r > 2$ be the least prime integer such that $n \not\equiv 1 \pmod{r}$.

Case 1: n is prime

If n is prime, then by Theorem 2, the polynomial congruence holds and n passes the test as expected.

Case 2: n is composite

If n composite, suppose n has a prime divisor $\leq r$. Clearly, it fails the test as expected.

By Corollary 3 we infer 1 is not a r-th power residue modulo n for at least 1 prime p dividing n. Hence, all of the necessary preconditions for Theorem 3 are satisfied and it applies. By Theorem 3, the polynomial congruence cannot hold, and therefore n fails the test as expected.

Conclusion:

Under the given conditions, when n is prime, the polynomial congruence holds and n passes the test as expected. When n is composite, the polynomial congruence does not hold and n fails the test as expected. The theorem is proven. □

6 Algorithm

INPUT: An integer $n > 1$.

1. If $n \equiv 0 \mod 2$:
   1.1. If n equals 2, output PRIME.
   1.2. Otherwise, output COMPOSITE.
2. If n equals 3, output PRIME.
3. If $2^{n-1} \bmod n$ does not equal 1, output COMPOSITE.
4. Set $r = 1$.
5. Set $\mathcal{V} = 1$.
6. While $\mathcal{V} \leq n$:
   6.1. Starting from r, find the next prime integer $p > r$ and set $r = p$.
   6.2. If $n \bmod r$ equals 0, output COMPOSITE.
   6.3. Compute the polynomial expansion of $x^n \bmod n$ in the ring $\mathbb{Z}_n[x]/(x^r - 1)$ with degree r, and store the result.
   6.4. Compute the polynomial expansion of $(1 + x)^n \bmod n$ in the ring $\mathbb{Z}_n[x]/(x^r - 1)$ with degree r, and store the result.
   6.5. If $(1 + x)^n \neq 1 + x^n$, output COMPOSITE.
   6.6. Set $\mathcal{V} = \mathcal{V} * r$.
7. Output PRIME;

6.1 Time Complexity Analysis

The given algorithm is a primality test that involves several computational steps, including modular arithmetic and polynomial exponentiation in the ring $\mathbb{Z}_n[x]/(x^r - 1)$. In this subsection, we use $M(n)$ to denote the worst-case time complexity of integer multiplication in terms of n.

6.1.1 Analysis of Individual Operations

1. **Check preconditions for** n:

   This step involves calculating $n \bmod 2$, $n \bmod 3$, and some miscellaneous conditional checks such as $n > 3$. The time complexity for each of these operations is $O(1)$.

$$T_2(n) = O(1) \tag{17}$$

2. **Finding the first** $\log(n)$ **primes (**r **values)**:

   Finding the first $\log(n)$ prime integers, which will be used as the r values in our test, can be done in $O(\log^2(n))$ time using the Sieve of Eratosthenes [17]. Ideally, the list of primes should be precomputed once in advance up to a reasonably large number of primes and cached for reuse in future computations. However, for the sake of completeness, we do not assume any precalculations.

$$T_2(n) = O(\log^2(n)) \tag{18}$$

3. **Main loop**:

The purpose of the main loop is to guarantee that our test is performed with at least one prime integer r that does not divide $p-1$, where p is some prime divisor of n. This guarantees that our test is performed with at least one r value such that 1 is not a nontrivial r-th power residue modulo said p, which is critical for ruling out all composites in our test.

Since p is a prime divisor of n, p itself may have at most $\log(n)$ distinct prime divisors. Therefore, to ensure the validity of our result, we must perform our test with $\log(n)$ distinct prime integers r_i from $i = 0 \dots \log(n)$. Along the way, we must also test the divisibility of n by each r_i, as it is a necessary for the validity of our test that n not have prime factor $\leq r_i$ for any r_i tested.

If n is squarefree, we need only test a single $r_i \not\equiv 1 \pmod{n}$. However, due to additional complexities in the nonsquarefree case, we must try up to $\log(n)$ distinct primes r_i.

We denote the time complexity for each step in the loop by $T_{3.j}$, where j is the position of the substep in the loop.

Loop subroutine:

For $i = 0$, $i \leq \log(n)$, $i++$:

3.1. **Initializing r_i**:
Initializing r_i involves setting r_i to the next term in the list of primes we prepoluated earlier via sieving, which is trivially $O(1)$.

$$T_{3.1} = O(1) \tag{19}$$

3.2. **Checking the divisibility of n by r_i**:
This step involves calculating $n \bmod r_i$ and a conditional check. The time complexity of the modulo operation is $O(\log(r_i))$. Since r_i is roughly of size $O(\log(n))$, the time complexity simplifies to:

$$T_{3.2} = O(\log\log(n)) \tag{20}$$

3.3. **Computing $2^{\lfloor \frac{n-1}{r} \rfloor}$**:
Calculating $2^{\lfloor \frac{n-1}{r} \rfloor} \pmod{n}$ requires modular exponentiation with a $\log(n)$-digit base and a $\log(n)$-digit exponent. The time complexity of modular exponentiation is $O(\log(n)M(n))$.

$$T_{3.3} = O(\log(n)M(n)) \tag{21}$$

3.4. **Computing polynomial consequences**:
Computing the polynomial expansions of $x^n \pmod{n}$ and $(1+x)^n \pmod{n}$ in the ring $\mathbb{Z}_n[x]/(x^r - 1)$ with degree r each involve exponentiating a polynomial in $\mathbb{Z}_n[x]/(x^{r_i} - 1)$ with r_i terms. Exponentiation using repeated squaring takes $O(\log(n))$ steps, and each step requires $O(M(n)r_i)$ time due to the multiplication of polynomials of size r_i. Since r_i is bounded above by $\log(n)$, we say that $r_i = O(\log(n))$.

$$T_{3.4} = O(\log^2(n)M(n) \tag{22}$$

3.5. **Checking the equality $(1+x)^n = 1 + x^n \pmod{n} \in \mathbb{Z}_n[x]/(x^r - 1)$**: The final steps involve comparing the equality of coefficients in the polynomials $(1+x)^n$ and $1 + x^n$. This requires $O(\log(n))$ comparisons, which are themselves $O(1)$ operations.

$$T_{3.5} = O(\log(n)) \tag{23}$$

In the loop subroutine, the $T_{3.4}$ step dominates. Since the subroutine's steps are repeated maximally $\log(n)$ times, the overall time complexity of this step is:

$$T_3 = O(\log(n)T_{3.4}) = O(\log(n)(\log^2(n)M(n))) = O(\log^3(n)M(n)) \tag{24}$$

6.1.2 Overall Time Complexity

The time complexity of the main loop ($T_3(n)$) dominates. Therefore, the overall time complexity of the algorithm is:

$$T(n) = O(T_3(n)) = O(\log^3(n)M(n))) \tag{25}$$

6.1.3 Overall Time Complexity Analysis

Harvey and van Der Heoven (2021) [21] have given an algorithm for integer multiplication which has a time complexity $M(n) = O(\log(n)\log\log(n))$. This would give our algorithm an overall time complexity of:

$$\begin{aligned} T(n) &= O(\log^3(n)M(n)) && (26)\\ &= O(\log^3(n)(\log(n)\log\log(n))) && (27)\\ &= O(\log^4(n)\log\log(n)) && (28) \end{aligned}$$

In soft-O notation [22], typically denoted as $\tilde{O}$, simplicity is achieved by omitting slower-growing logarithmic and lower-order factors that do not significantly contribute to the overall growth rate of the function.

In the context of our overall time complexity $T(n) = O(\log^4(n)\log\log(n))$, where the dominant term is $O(\log^4(n))$, the factor $\log\log(n)$ is omitted. Hence, our overall time complexity simplifies to:

$$\tilde{T}(n) = \tilde{O}(\log^4(n)) \tag{29}$$

6.1.4 Conclusion

The overall complexity is polynomial in the size of n when expressed in terms of bit operations, making the algorithm theoretically efficient in the worst case for large values of n. In practice, it is incredibly unlikely for the running time to exceed $\tilde{O}(log^3(n))$.

7 Implementation Details

Sample open source .NET and Python implementations, along with test data, are available on the author's Github page [23].

References

[1] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in p. *Annals of Mathematics*, pages 781–793, 2002.

[2] Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.

[3] Hendrik W Lenstra and Carl Pomerance. Primality testing with gaussian periods. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 369(1951): 3376–3390, 2011.

[4] Hendrik W Lenstra. Factoring integers with elliptic curves. *Annals of mathematics*, pages 649–673, 1987.

[5] Robert Baillie and Samuel S Jr Wagstaff. Lucas pseudoprimes. *Mathematics of Computation*, 35(152): 1391–1417, 1980.

[6] Michael O Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12(1): 128–138, 1980.

[7] Gary L Miller. Riemann's hypothesis and tests for primality. *Journal of Computer and System Sciences*, 13(3):300–317, 1976.

[8] Samuel S Jr Wagstaff. Pseudoprimes and a generalization of artin's conjecture. *Acta Arithmetica*, 41 (2):141–150, 1983.

[9] Carl Pomerance. The use of elliptic curves in cryptography. *Advances in Cryptology*, pages 203–208, 1984.

[10] Andrew Granville. It is easy to determine whether a given integer is prime. *Bulletin of the American Mathematical Society*, 42:3–38, 2004. URL `https://www.ams.org/journals/bull/2005-42-01/S0273-0979-04-01037-7`.

[11] Carl Pomerance Richard Crandall. *Prime Numbers: A Computational Perspective*. New York: Springer, 2 edition, 2005. ISBN 978-0387-25282-7.

[12] Carl Pomerance W. R. Alford, Andrew Granville. There are infinitely many carmichael numbers. *Annals of Mathematics*, 139(3):703–722, 1994. ISSN 0003486X. URL `http://www.jstor.org/stable/2118576`.

[13] N. J. A. Sloane. Entry a001220 in the on-line encyclopedia of integer sequences. `https://oeis.org/A001220`, 2024. Wieferich primes.

[14] PrimeGrid. Wieferich prime search statistics, 2024. URL `https://www.primegrid.com/stats_ww.php`. Accessed: January 2024.

[15] N. A. Carella. Results for wieferich primes, 2018.

[16] Barkley Rosser. Explicit bounds for some functions of prime numbers. *American Journal of Mathematics*, 63(1):211–232, 1941. ISSN 00029327, 10806377. URL `http://www.jstor.org/stable/2371291`.

[17] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 3 edition, 2009. ISBN 978-0-262-03384-8.

[18] Swastik Kopparty and Qiang Wang. Roots and coefficients of polynomials over finite fields. *Finite Fields and Their Applications*, 29:198–201, 2014. ISSN 1071-5797. doi: https://doi.org/10.1016/j.ffa.2014.04.002. URL `https://www.sciencedirect.com/science/article/pii/S1071579714000574`.

[19] Gregory Karpilovsky. *Topics in Field Theory*. North-Holland Mathematics Studies. North-Holland, 1989. ISBN 9780444705207.

[20] David S. Dummit and Richard M. Foote. *Abstract Algebra*. John Wiley & Sons, 3 edition, 2004. ISBN 978-0-471-43334-7.

[21] Joris van Der Hoeven David Harvey. Integer multiplication in time o(n log n). *Annals of Mathematics*, 2021. doi: 10.4007/annals.2021.193.2.4.hal-02070778v2.

[22] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 3 edition, 2013. ISBN 978-1107039032.

[23] Joseph M. Shunia. A sample .net implementation of the primality test. `https://github.com/jshunia/Shunia.Primes`, 2023. Accessed: December 2023.