PRIMES is in P

Mustafa Motiwala[1] **William He**[2]

[1]motiwala.ca (website)

[2]sites.google.com/view/wenjunhewilliam (website)

March 31, 2026

Abstract

We summarize the results first presented by Agrawal, Kayal, and Saxena in [AKS04]

1 Introduction

Prime numbers are omnipresent in mathematics and computer science ranging from being the central characters of number theory to the pillars of modern day cryptography. The problem of (efficiently) distinguishing prime numbers from composite numbers, which we denote as the PRIMES, is thus of great interest. The definition of prime (being indivisible by any smaller number) yields an obvious test – n is prime if and only if n is not divisible by any $m \leq \sqrt{n}$ – but this is inefficient, being exponential in the length of the input. In this article, we present a faster approach, first discovered by [AKS04]. We begin by summarizing previous work.

1.1 Background

Since the notions of complexity were formalised in the 1960s, the primality testing problem has been investigated intensively. It is trivial to see that PRIMES is in coNP: if n is not a prime, that can be efficiently verified with a prime factor of n. In 1974, Pratt [Pra75] proved that PRIMES is also in NP, thereby exhibiting PRIMES $\in$ NP $\cap$ coNP and raising the stakes on the question of whether PRIMES $\in$ P.

In 1975, Miller [Mil76] proved that assuming the Extended Riemann Hypothesis (ERH), PRIMES is in P, where the algorithm constructed uses a property based on Fermat's Little Theorem ($a^p = p \bmod p$ for all primes p). Since then, Rabin modified the algorithm in [Rab80] and was able to prove PRIMES $\in$ RP.

Concurrently, Solovay and Strassen [SS77] obtained in 1974 a different randomised polynomial-time algorithm for PRIMES, one that uses a property that for a prime n, $\left(\frac{a}{n}\right) = a^{\frac{n-1}{2}} (\bmod n)$ where (-) is the Jacobi symbol. Moreover, this algorithm can be made deterministic polynomial-time assuming ERH.

In 1983, a breakthrough result by Adleman, Pomerance, and Rumely [APR83] provided the first sub-exponential-time algorithm for PRIMES with time complexity $\log(n)^{\mathcal{O}(\log\log\log(n))}$.

Progress continued, eventually culminating in a randomized algorithm by Adleman and Huang [AH92] which runs in expected polynomial-time. However, the ultimate goal of an unconditional, deterministic polynomial time algorihtm for PRIMES remained elusive until 2004, when it was finally solved by [AKS04].

1.2 This work

In 2004, [AKS04] came up with a deterministic polynomial-time algorithm that runs in $\tilde{\mathcal{O}}(\log^{21/2}(n))$. With some deeper analysis, this time-bound can be improved to $\tilde{\mathcal{O}}(\log^{15/2}(n))$, and under a widely believed conjecture about the density of primes p such that $2p+1$ is also prime (known as Sophie Germain primes), the runtime improves further to $\tilde{\mathcal{O}}(\log^{6}(n))$.

The algorithm is based on an extension of Fermat's Little Theorem to polynomials over finite fields and relies on elementary results from algebra. We first fix some notation in Section 2 before summarizing the main idea of the algorithm in Section 3 and then discussing the algorithm and its correctness/complexity in Section 4.

2 Preliminaries

By $\mathbb{Z}_n$ we mean the ring of integers mod n. By $\mathbb{F}_p$, we mean the finite field with p elements where p is prime. When $h(X)$ is a degree-d polynomial irreducible over $\mathbb{F}_p$, then $\mathbb{F}_p[X]/(h(X))$ is a finite field of order p^d. We use the notation

$$f(X) = g(X) \pmod{h(X),\, n}$$

to indicate $f(X) = g(X)$ in the ring $\mathbb{Z}_n[X]/(h(X))$

Note that $\log = \log_2$. For any function $t(n)$, we use $\tilde{\mathcal{O}}(t(n))$ to denote the class $\mathcal{O}(t(n) \cdot \text{poly}(\log t(n)))$. Note that for any $\varepsilon > 0$, we have

$$\tilde{\mathcal{O}}(t(n)) = \mathcal{O}(t^{1+\varepsilon}(n))$$

Given $r \in \mathbb{N}, a \in \mathbb{Z}$ with $\gcd(a, r) = 1$, the order of a modulo r is denoted as $o_r(a)$. That is, $o_r(a)$ is the smallest $k \in \mathbb{N}$ such that $a^k = 1 \pmod{r}$.

Given $r \in \mathbb{N}$, the *Euler totient* $\phi(r)$ is the size of the set $\{a \in \mathbb{N} : a < r, \gcd(a, r) = 1\}$.

3 The Idea

The new test of primality stated in the below lemma utilises Polynomial Identity Testing. This is a generalisation of the Fermat's Little Theorem.

Lemma 3.1. *Let $n \in \mathbb{N}, n \geq 2$ and $a \in \mathbb{Z}$ such that $\gcd(a, n) = 1$. Then,*

$$n \text{ is prime} \iff (X + a)^n \equiv X^n + a \pmod{n}$$

Proof. We will do this by showing the coefficients of X^i for the polynomial $(X + a)^n - (X^n + a)$ is 0 for all $0 \leq i \leq n$. One can easily see this is always true for $i = 0$ and $i = n$.

Now, suppose $0 < i < n$. The coefficient of X^i in $((X + a)^n - (X^n + a))$ is $\binom{n}{i}a^{n-i}$ by Binomial Theorem.

Case 1: If n is prime, for any $0 < i < n$, $\binom{n}{i} = 0 \bmod n$ and thus the coefficient of X^i is 0. This is true for all $0 < i < n$.

Case 2: If n is not prime, there exists some prime q such that $q \mid n$. Take largest k such that $q^k \mid n$. We have

$$\binom{n}{q} = \frac{n(n-1)\cdots(n-(q+1))}{q!} = \underbrace{\frac{(n/q)(n-1)\cdots(n-(q+1))}{(q-1)!}}_{\text{not divisible by } q^k} \implies \binom{n}{q} \not\equiv 0 \pmod{q^k}$$

Since $q \mid n$

$$\gcd(a, n) = 1 \implies \gcd(a, q) = 1 \implies a \not\equiv 0 \pmod{q} \implies a^{n-q} \not\equiv 0 \pmod{q^k}$$

together we get $\binom{n}{q} a^{n-q} \not\equiv 0 \bmod q^k$. Hence, $\binom{n}{q} a^{n-q} \not\equiv 0 \bmod n$ and we have a non-zero coefficient at X^q. ■

This lemma gives us a simple test to check primality but is still inefficient if implemented in the algorithm: to check if $(X + a)^n \equiv X^n + a \pmod{n}$, we have to compute all $n = 2^{\log n}$ coefficients of two n-degree polynomials, which is still exponential with respect to the length of the input.

To reduce the computational demand, we will evaluate the equation not just modulo n, but modulo some polynomial $X^r - 1$ for a small $r \in \text{poly}(\log(n))$ to be chosen in the algorithm. In other words, we will check if the following equation is satisfied

$$(X + a)^n \equiv X^n + a \pmod{X^r - 1,\ n}$$

The tradeoff of this test to reach full correctness is that we need to check this equation for many values of a. We will see, however, that we can still do this in polynomial time.

4 The Algorithm

We now provide the algorithm

Algorithm 1: AKS Primality Test

```
1:  procedure IsPrime(n)
2:      if n = a^b for a ∈ ℕ, b > 1
3:          output COMPOSITE
4:      r ← smallest such that o_r(n) > log² n
5:      if ∃a ≤ r, 1 < gcd(a, n) < r
6:          output COMPOSITE
7:      if n ≤ r
8:          output PRIME
9:      for a = 1, ..., ⌊√φ(r) log n⌋
10:         if (X + a)^n ≠ X^n + a  (mod X^r − 1, n)
11:             output COMPOSITE
12:     output PRIME
```

4.1 Correctness

Before we can argue correctness, we need to be certain the algorithm actually terminates at all. This is mostly obvious, except for line 4. We will see later in Lemma 4.9 that such an r always exists and in fact $r \leq \max\{3, \lfloor \log^5(n) \rfloor\}$. For now, we take this as granted.

We aim to show $\textsc{IsPrime}(n) = \textsf{PRIME}$ if and only if n is prime. In the ($\Longleftarrow$) direction, observe that if n is prime then the algorithm certainly cannot terminate on line 3 (for n cannot be a power of another integer) or line 6 (for n cannot have $\gcd(a, n) > 1$ for any $a \leq n$). Finally, by the discussion in Section 3, the polynomial identity test on line 10 can never fail when n is prime, so the algorithm can't terminate on Line 11. Thus, we must have $\textsc{IsPrime}(n) = \textsf{PRIME}$.

In the remainder of this section, we focus on proving the ($\Longrightarrow$) direction. Suppose $\textsc{IsPrime}(n) = \textsf{PRIME}$ but n is a composite number. Let r be as chosen on line 4. If IsPrime terminated on line 8, then n must be prime for otherwise line 5 would have found some factor of n. So, assume we terminate on Line 12, after checking $\ell := \lfloor \sqrt{\phi(r)} \log n \rfloor$ polynomial equations.

Write $n = p_1 p_2 \cdots p_k$ for primes p_i and $k \geq 1$. Since $o_r(n) \mid \mathrm{lcm}(o_r(p_1), ..., o_r(p_k))$, and $o_r(n) > 1$, we must have $o_r(p_i) > 1$ for some $1 \leq i \leq k$. We let $p := p_i$.

We begin with a definition that isolates the key condition checked in line 10.

Definition 4.1. *For a polynomial $f(X)$ and a natural number $m \in \mathbb{N}$, we say m is* introspective *for f if*

$$(f(X))^m = f(X^m) \pmod{X^r - 1,\ p}$$

We first observe that this introspective relation is closed under multiplication when one argument is fixed.

Lemma 4.2. *Let $f(X)$ be a polynomial. If both m, m' are introspective for f, then $m \cdot m'$ is introspective for f.*

Lemma 4.3. *Let $m \in \mathbb{N}$. If m is introspective for both $f(X), g(X)$, then m is introspective for $f(X)g(X)$.*

Next, we exhibit some explicit pairs of polynomials and natural numbers which are related this way.

Lemma 4.4. *For all $0 \leq a \leq \ell$, we have both $p, n/p$ introspective for $X + a$.*

Proof. Since p is prime, we have from Lemma 3.1 that $(X + a)^p = X^p + a \pmod{p}$ for all a so we immediately get

$$(X + a)^p = X^p + a \pmod{X^r - 1,\ p}$$

Furthermore, we have by the termination of the algorithm in line 12 that for all $0 \leq a \leq \ell$, we have

$$(X + a)^n = X^n + a \pmod{X^r - 1,\ n}$$

and since $p \mid n$, we get

$$(X + a)^n = X^n + a \pmod{X^r - 1,\ p}$$

so actually n is introspective for the $X + a$ as well. Putting these two together, we get

$$\left((X + a)^{\frac{n}{p}}\right)^p = (X + a)^n = X^n + a = \left(X^{\frac{n}{p}}\right)^p + a = \left(X^{\frac{n}{p}} + a\right)^p \pmod{X^r - 1,\ p}$$

Since $\mathbb{F}_p$ has characteristic p, we have $f^p = g^p \implies (f - g)^p = 0$. Thus, we have

$$\left((X + a)^{\frac{n}{p}} - \left(X^{\frac{n}{p}} + a\right)\right)^p = 0$$

Finally, we want to show that we can cancel pth powers. Notice that the formal derivative of $X^r - 1$ is rX^{r-1} which, as $r \nmid p$ has only 0 as a root. Thus, $\gcd(X^r - 1, rX^{r-1}) = 1$ and so $X^r - 1$ factors into irreducibles $q_1 \cdots q_k$ where the q_i are all distinct (if they were not distinct, some q_i would appear at least twice and thus divide both f and f', by the product rule). This implies that if $X^r - 1 \mid f^p$ then $X^r - 1 \mid f$ for all polynomial f. In other words, if $f^p = 0 \pmod{X^r - 1,\ p}$, then $f = 0 \pmod{X^r - 1,\ p}$. This implies $(X + a)^{\frac{n}{p}} - \left(X^{\frac{n}{p}} + a\right) = 0$ which is to say

$$(X + a)^{\frac{n}{p}} = X^{\frac{n}{p}} + a \pmod{X^r - 1,\ p}$$

and so $\frac{n}{p}$ is introspective for all $X + a$, as needed. ∎

The multiplicative closure lemmas Lemma 4.2, Lemma 4.3, together with Lemma 4.4 imply that for

$$I = \left\{ \left(\frac{n}{p}\right)^i p^j : i, j \geq 0 \right\} \qquad \text{and} \qquad P = \left\{ \prod_{a=0}^{\ell} (X+a)^{e_a} : \forall a, e_a \geq 0 \right\}$$

we have for every $m \in I, f(X) \in P$ that m is introspective for $f(X)$. We form groups from I and P by modular reduction. First, let $Q_r(X)$ denote the rth cyclotomic polynomial over $\mathbb{F}_p$ (so that Q_r has as roots exactly the primitive rth roots of units in $\mathbb{F}_p$). Then $Q_r(X) \mid X^r - 1$ and factors into irreducible factors of degree $o_r(p)$. We let $h(X)$ be one such irreducible factor. Now, we set

$$G = \{i \pmod r : i \in I\} \qquad \text{and} \qquad \mathcal{G} = \{f(X) \pmod{h(X), p} : f(X) \in P\}$$

so that G is a subgroup of $\mathbb{Z}_r^*$ generated by n and p and $\mathcal{G}$ is a subgroup of the multiplicative group of $\mathbb{F} \coloneqq \mathbb{F}_p[X]/(h(X))$ generated by the polynomials $X + a$ for $0 \leq a \leq \ell$.

We have an immediate lower bound on $|G|$ for since $o_r(n) > \log^2(n)$ and G is partially generated by n, we have $|G| > \log^2(n)$. We can also lower-bound the size of $\mathcal{G}$.

Lemma 4.5. $|\mathcal{G}| \geq \binom{|G|+\ell}{|G|-1}$.

Proof. We show that the mapping $P \to \mathcal{G}$ given by reduction mod h, p is injective for sufficiently low-degree polynomials. That is, if $f(X), g(X) \in P$ are distinct and have degree $< |G|$, then they are distinct as elements of $\mathbb{F}$.

Let $f(X), g(X)$ be two such polynomials and suppose towards a contradiction that they are equal in $\mathbb{F}$. Then certainly $(f(X))^m = (g(X))^m$ as elements of $\mathbb{F}$ for all $m \in \mathbb{N}$. So, for any $m \in G$, we have by the previous equation and that m is introspective for both f, g that $f(X^m) = g(X^m)$ in $\mathbb{F}$.

Thus, for all $m \in G$, the polynomial X^m is, as an element of $\mathbb{F}$, a root of the polynomial $f(Y) - g(Y)$. This contradicts $f - g \neq 0$ and $\deg(f - g) < |G|$. We conclude that f, g are distinct as elements of $\mathbb{F}$.

So, $|\mathcal{G}|$ is at least the number of polynomials in P with degree $< |G|$. We claim that there are at least $\binom{|G|+\ell}{|G|-1}$ such polynomials. Indeed, since

$$\ell = \left\lfloor \sqrt{\phi(r)} \log n \right\rfloor < \sqrt{r} \log n < r < p$$

we have $i \neq j$ in $\mathbb{F}_p$ for all $0 \leq i \neq j \leq \ell$ and thus the polynomials $X + 0, ..., X + \ell$ are distinct in $\mathbb{F}$. Moreover, $\deg h = o_r(p) > 1$ so none of the $X + a$ are zero. Thus, there are at least $\ell + 1$ polynomials of degree 1 in P and therefore $\binom{|G|+\ell}{|G|-1}$ polynomials of degree $< |G|$, which gives the stated bound. ∎

Lemma 4.6. *If $|\mathcal{G}| > n^{\sqrt{|G|}}$, then n is a power of p.*

Proof. We prove the contrapositive: Suppose n is not a power of p, we prove that $|\mathcal{G}| \leq n^{\sqrt{|G|}}$. Let $t = |G|$. Consider the subset $\hat{I} \subseteq I$ defined by

$$\hat{I} = \left\{ \left(\frac{n}{p}\right)^i p^j : 0 \leq i, j \leq \left\lfloor \sqrt{t} \right\rfloor \right\}$$

Since n is not a power of p, every distinct pair (i, j) yields a different element, so $|\hat{I}| = \left(\left\lfloor\sqrt{t}\right\rfloor + 1\right)^2 > t$. Since $\{i \bmod r : i \in I\} = |G| = t$, there exist $m_1, m_2 \in \hat{I}$ such that $m_1 = m_2 \bmod r$. WLOG, we assume $m_1 > m_2$.

We claim that every element of P is a root of the polynomial $Q(Y) = Y^{m_1} - Y^{m_2}$ in the field $\mathbb{F}$. First, note that from $m_1 = m_2 \bmod r$ we have

$$X^{m_1} = X^{m_2 + \frac{m_1 - m_2}{r} \cdot r} = X^{m_2} \cdot (X^r)^{\frac{m_1 - m_2}{r}} = X^{m_2} \cdot 1^{\frac{m_1 - m_2}{r}} = X^{m_2} \pmod{X^r - 1}$$

So, for any $f(X) \in P$, we have $m_1, m_2 \in \hat{I} \subseteq I$ are introspective for f and so

$$(f(X))^{m_1} = f(X^{m_1}) = f(X^{m_2}) = (f(X))^{m_2} \pmod{X^r - 1,\, p}$$

so f is a root of $Y^{m_1} - Y^{m_2}$ in the field $\mathbb{F}$.

Since $m_1 > m_2$, the polynomial $Q(y)$ have at most $m_1 \leq \left(\frac{n}{p} \cdot p\right)^{\left\lfloor\sqrt{t}\right\rfloor} \leq n^{\sqrt{t}} = n^{\sqrt{|G|}}$ roots. So $|\mathcal{G}| = |\{f \pmod{h(X),\, p} : f \in P\}| \leq |P| \leq m_1 = n^{\sqrt{|G|}}$ as needed. ∎

Now, we are able to complete the proof

Theorem 4.7. *If the algorithm returns* PRIME *then n is prime.*

Proof. Suppose that the algorithm returns PRIME, by Lemma 4.5, we have

$$\begin{aligned}
|\mathcal{G}| &\geq \binom{|G| + \ell}{|G| - 1} \quad \left[\ell = \left\lfloor\sqrt{\varphi(r)}\log(n)\right\rfloor\right] \\
&= \binom{(\ell + 1) + (|G| - 1)}{|G| - 1} \\
&\geq \binom{(\ell + 1) + \left\lfloor\sqrt{|G|}\log(n)\right\rfloor}{\left\lfloor\sqrt{|G|}\log(n)\right\rfloor}
\end{aligned}$$

where the last inequality is true as $\binom{(\ell+1)+(|G|-1)}{|G|-1}$ is increasing in $|G| - 1$. Further, we previously showed that $|G| > \log^2(n)$. This gives $|G| > \sqrt{|G|}\log(n) \implies |G| - 1 \geq \left\lfloor\sqrt{|G|}\log(n)\right\rfloor$. As G is a subgroup of $\mathbb{Z}_r^*$, we have $|G| \leq \varphi(r) \implies \ell = \left\lfloor\sqrt{\varphi(r)}\log(n)\right\rfloor \geq \left\lfloor\sqrt{|G|}\log(n)\right\rfloor$. Hence, we get

$$\binom{(\ell + 1) + \left\lfloor\sqrt{|G|}\log(n)\right\rfloor}{\left\lfloor\sqrt{|G|}\log(n)\right\rfloor} \geq \binom{2\left\lfloor\sqrt{|G|}\log(n)\right\rfloor + 1}{\left\lfloor\sqrt{|G|}\log(n)\right\rfloor}$$

Now, note that since $n \geq 2$ we have $\left\lfloor\sqrt{|G|}\log(n)\right\rfloor \geq 1$. We will argue in cases.

Case 1: Suppose $\left\lfloor\sqrt{|G|}\log(n)\right\rfloor = 1$, then we get $\log(n) < 2$. Hence we either have $n = 2$ or $n = 3$. In either cases, n is prime.

Case 2: Suppose $\left\lfloor\sqrt{|G|}\log(n)\right\rfloor \geq 2$, then by a simple induction proof, we are able to show

$$\begin{aligned}
\binom{2\left\lfloor\sqrt{|G|}\log(n)\right\rfloor + 1}{\left\lfloor\sqrt{|G|}\log(n)\right\rfloor} &> 2^{\left\lfloor\sqrt{|G|}\log(n)\right\rfloor + 1} \\
&\geq 2^{\sqrt{|G|}\log(n)} \\
&= n^{\sqrt{|G|}}
\end{aligned}$$

By Lemma 4.6, n is a power of p. Therefore, $n = p^k$ for some $k > 0$. We certainly have $k = 1$ as if $k > 1$ the algorithm would return COMPOSITE in line 2. Hence, $n = p$. ∎

4.2 Time Complexity

Lemma 4.8. ([Nai82]): *Let* $\text{LCM}(m)$ *denote the* lcm *of first* m *numbers. For* $m \geq 7$*:*

$$\text{LCM}(m) \geq 2^m$$

Using the above lemma, we can now show the existence of r that we left with, together, with the upper bound of r as well.

Lemma 4.9. *There is an* $r \leq \max\{3, \lceil \log^5(n) \rceil\}$ *such that* $o_r(n) > \log^2(n)$*.*

Proof. Cases for $n < 10$ can be checked manually. Assume that $n \geq 10$. For simplicity denote $B = \lceil \log^5(n) \rceil$, then $B > 10$. Consider the smallest r that does not divide the product

$$n^{\lfloor \log(B) \rfloor} \cdot \prod_{i=1}^{\lfloor \log^2(n) \rfloor} (n^i - 1)$$

We first show $r \leq B$, note that

$$n^{\lfloor \log(B) \rfloor} \cdot \prod_{i=1}^{\lfloor \log^2(n) \rfloor} (n^i - 1) \leq n^{\lfloor \log(B) \rfloor + \frac{1}{2}\log^2(n)(\log^2(n)+1)} \leq n^{\log^4(n)} = 2^{\log^5(n)} \leq 2^B$$

where the second inequality is true for all $n \geq 10$. Since $1, ..., r-1$ divides the product, $\text{LCM}(r-1) < 2^B$. Combine with Lemma 4.8 and $B \geq 10$ we get $\text{LCM}(B) \geq 2^B$. Hence, we have $r \leq B$.

Now, we show $o_r(n) > \log^2(n)$.

We first show $o_r(n)$ is defined by showing $\gcd(r, n) = 1$. Considering the prime factoring of $r = a_1^{r_1} ... a_m^{r_m}$. Assuming $a_1, ..., a_j$ are the only primes that divides n, for some $j \leq m$. We show that $j = 0$ (which implies $\gcd(r, n) = 1$). Indeed, since $r \leq B$, all prime divisors of r in the prime factoring have exponent at most $\lfloor \log(B) \rfloor$. This means $a_1^{r_1} ... a_j^{r_j} \mid n^{\lfloor \log(B) \rfloor}$. Thus, $a_{j+1}^{r_{j+1}} ... a_m^{r_m}$ does not divide the product $n^{\lfloor \log(B) \rfloor} \cdot \prod_{i=1}^{\lfloor \log^2(n) \rfloor} (n^i - 1)$. Therefore, $\frac{r}{\gcd(r,n)}$ also does not divide the product. Since r is the smallest number that divides the product, $\gcd(r, n) = 1$ as required.

Finally, it is easy to see that by definition, r does not divide $n^i - 1$ for all $1 \leq i \leq \lfloor \log^2(n) \rfloor$. Hence, we have $o_r(n) > \log^2(n)$. ∎

The proof of time complexity bound of our algorithm requires the following lemma.

Lemma 4.10. ([GG99]): *Addition, multiplication, and division operations between*

- *two* m *bits numbers can be performed in time* $\tilde{\mathcal{O}}(m)$ *steps.*
- *two degree* d *polynomials with coefficients at most* m *bits* $\tilde{\mathcal{O}}(d \cdot m)$ *steps.*

Now we can prove the time complexity of our algorithm.

Theorem 4.11. *The asymptotic time complexity of the algorithm is* $\tilde{\mathcal{O}}\left(\log^{\frac{21}{2}}(n)\right)$*.*

Proof. Line 2 can be done by brute forcing all the possible values of b. As $b \leq \log(n)$, we can brute force at most $\log(n) \in \mathcal{O}(\log(n))$ possible values of b. For each b, we can find the existence of $1 \leq a \leq n$ such that $n = a^b$ via binary search. Therefore, for each b we test at most $\log(n) \in \mathcal{O}(\log(n))$ possible values of a. Finally, for each (a, b), computing a^b starting from a can be done by repeated squaring for $\log(b)$ times. Since we perform squaring on $\log(n)$-bit numbers, computing each a^b takes

$$\log(n) \cdot \log(b) \leq \log(n) \cdot \log(\log(n)) \in \tilde{\mathcal{O}}(\log(n))$$

steps. Hence, entire steps takes $\tilde{O}(\log^3(n))$ steps.

Line 4 can be done by trying all possible integer $2 \leq r \leq \lceil \log^5(n) \rceil$ by Lemma 4.9. For each r, checking $o_r(n) > \log^2(n)$ can be done by doing $\log^2(n)$ multiplications in modulo r. This will take time $\tilde{\mathcal{O}}(\log^2(n)\log(r))$ with a particular value of r. The entire step takes $\tilde{\mathcal{O}}(\log^7(n))$ steps.

Line 5 computes the $\gcd(i,n)$ for all $2 \leq i \leq r$. Each gcd computation takes time $\mathcal{O}(\log n)$. Hence, this entire step takes $\mathcal{O}(r\log(n)) = \mathcal{O}(\log^6(n))$ steps.

Line 9 verifies $\left\lfloor \sqrt{\phi(r)}\log(n) \right\rfloor$ equations. Each equation can be verified by computing two polynomials of degree at most r. By repeated squaring, computing each polynomial requires $\mathcal{O}(\log(n))$ multiplications of degree r polynomials with coefficients of size $\mathcal{O}(\log n)$. So each equation can be verified in $\tilde{\mathcal{O}}(r\log^2(n))$ steps. This entire step takes $\tilde{\mathcal{O}}\left(r\sqrt{\phi(r)}\log^3(n)\right) = \tilde{\mathcal{O}}\left(r^{\frac{3}{2}}\log^3(n)\right) = \tilde{\mathcal{O}}\left(\log^{\frac{21}{2}}(n)\right)$ steps.

Hence, the time complexity of the algorithm is $\tilde{\mathcal{O}}\left(\log^{\frac{21}{2}}(n)\right)$. ■

Time complexity can be further improved by improving the estimate of r. The following conjecture support such possibility of a better estimate.

Conjecture 4.12. *The number of primes $q \leq m$ such that $2q+1$ is also a prime is asymptotically $\frac{2C_2 m}{\ln^2(m)}$ where C_2 is the twin prime constant (estimated to be approximately 0.66).*

***Remark* 4.13.** The result of the above conjecture is that $r \in \mathcal{O}(\log^2(n))$ instead of $\mathcal{O}(\log^5(n))$. This gives a time complexity of $\tilde{\mathcal{O}}(\log^6(n))$.

Although the above conjecture is yet to be proven, the following result provides a close enough estimate of r.

Lemma 4.14. ([BH96]): *Denote $P(m)$ as the greatest prime divisor of m. There exists n_0 such that for all $x \geq n_0$,*

$$\left|\left\{q \mid q \text{ is prime}, q \leq x \text{ and } P(q-1) > q^{\frac{2}{3}}\right\}\right| \geq 0.6683\frac{x}{\ln(x)}$$

***Remark* 4.15.** The result of the above lemma is that $r \in \mathcal{O}(\log^3(n))$ instead of $\mathcal{O}(\log^5(n))$. Hence, the algorithm has a time complexity of $\tilde{\mathcal{O}}\left(\log^{\frac{15}{2}}(n)\right)$.

Bibliography

[AH92] L. M. Adleman, and M.-D. A. Huang. *Primality testing and Abelian varieties over finite fields*. Springer-Verlag, Berlin ;, 1992

[AKS04] M. Agrawal, N. Kayal, and N. Saxena. "PRIMES is in P." *Ann. Math. (2)*, 160 (2), 781–793, 2004

[APR83] L. M. Adleman, C. Pomerance, and R. S. Rumely. "On distinguishing prime numbers from composite numbers." *Ann. Math. (2)*, 117, 173–206, 1983

[BH96] R. C. Baker, and G. Harman. "The Brun-Titchmarsh theorem on average." Birkhäuser Boston, Boston, MA, 1996

[GG99] J. von zur Gathen, and J. Gerhard. *Modern computer algebra*. Cambridge University Press, New York, 1999

[Mil76] G. L. Miller. "Riemann's hypothesis and tests for primality." *Journal of Computer and System Sciences*, 13 (3), 300–317, 1976

[Nai82] M. Nair. "On Chebyshev-type inequalities for primes." *Amer. Math. Monthly*, 89 (2), 126–129, 1982

[Pra75] V. R. Pratt. "Every Prime Has a Succinct Certificate." *SIAM Journal on Computing*, 4 (3), 214–220, 1975

[Rab80] M. O. Rabin. "Probabilistic algorithm for testing primality." *Journal of Number Theory*, 12 (1), 128–138, 1980

[SS77] R. Solovay, and V. Strassen. "A Fast Monte-Carlo Test for Primality." *SIAM Journal on Computing*, 6 (1), 84–85, 1977