PRIMES IS IN P

A presentation on work by Agrawal, Kayal and Saxena

William He, Mustafa Motiwala

2026-04-02

Outline

1. Introduction 1
   - 1.1 The PRIMES problem 2
   - 1.2 Previous Works 3
   - 1.3 This work 4
   - 1.4 Preliminaries and Notation 5
2. The Algorithm 7
   - 2.1 A polynomial primality test 8
   - 2.2 The AKS Primality Test 10
3. Correctness 11
   - 3.1 Sufficiency 12
   - 3.2 Necessity 13
   - 3.3 Introspective numbers 14
   - 3.4 Size bounds 18
4. Run Time 23
   - 4.1 An upper bound on r 24
   - 4.2 A loose time bound 26
   - 4.3 Time improvements 29
   - 4.4 Conclusion 30

1.1 The PRIMES problem

- Prime numbers have near universal importance in math and computer science.
  - Cryptographic protocols depend on large prime numbers.
- The PRIMES problem: given an integer $n > 1$, output PRIME if n is prime and COMPOSITE otherwise.
  - A standard way is to try dividing n by every number $m \leq \sqrt{n}$. If any m divides n then it is composite, otherwise it is prime.
  - This takes exponential time in the length of input, which is $\log(n)$.
  - We want to do this efficiently (in polynomial time).

1.2 Previous Works

Theorem (Pratt '75). PRIMES is in NP.

Theorem (Miller '76). Assuming Extended Riemann Hypothesis (ERH), PRIMES is in P.

- The algorithm uses a property based on Fermat's Little Theorem.

Theorem (Rabin '80). PRIMES is in RP.

Theorem (Solovay, Strassen '77). PRIMES is in RP.

- The algorithm uses a property that for a prime n, $\left(\frac{a}{n}\right) = a^{\frac{n-1}{2}} \pmod n$ where - is the Jacobi symbol.
- The algorithm can be made deterministic assuming ERH.

Theorem (Adleman, Pomerance, and Rumely '83). There is a deterministic $\log(n)^{\mathcal{O}(\log\log\log(n))}$ algorithm for PRIMES.

- The first algorithm that is sub-exponential in time.

1.3 This work

Theorem (Agrawal, Kayal, Saxena '04). PRIMES is in P.

We present a primality testing algorithm that is polynomial time. More exact time-bounds depend on further analysis.

We will prove that the algorithm is $\tilde{\mathcal{O}}\left(\log^{\frac{21}{2}}(n)\right)$ but this can, with some effort, be reduced to $\tilde{\mathcal{O}}\left(\log^{\frac{15}{2}}(n)\right)$.

Under a widely believed conjecture about the density of Sophie Germain primes (primes p such that $2p+1$ is also prime), the time complexity is reduced to $\tilde{\mathcal{O}}(\log^6(n))$.

1.4 Preliminaries and Notation

- $\mathbb{F}_p$ is the finite field with p elements, where p is prime.
- If p is prime and $h(X)$ is a polynomial of degree d and irreducible in $\mathbb{F}_p$, then $\mathbb{F}_p[X]/(h(X))$ is a finite field of order p^d.
- We denote

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

  as $f(X) = g(X)$ in the ring $\mathbb{Z}_n[X]/(h(X))$.
- For any time complexity function $t(n), \tilde{\mathcal{O}}(t(n)) = \mathcal{O}(t(n) \cdot \text{poly}(\log t(n))$
  - Note that for any $\varepsilon > 0$

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

1.4 Preliminaries and Notation

- 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 $\{a \in \mathbb{N} : a < r, \gcd(a, r) = 1\}$.

Outline

1. Introduction 1
   - 1.1 The PRIMES problem 2
   - 1.2 Previous Works 3
   - 1.3 This work 4
   - 1.4 Preliminaries and Notation 5
2. The Algorithm 7
   - 2.1 A polynomial primality test 8
   - 2.2 The AKS Primality Test 10
3. Correctness 11
   - 3.1 Sufficiency 12
   - 3.2 Necessity 13
   - 3.3 Introspective numbers 14
   - 3.4 Size bounds 18
4. Run Time 23
   - 4.1 An upper bound on r 24
   - 4.2 A loose time bound 26
   - 4.3 Time improvements 29
   - 4.4 Conclusion 30

2.1 A polynomial primality test

The idea is inspired by the following result

Lemma. 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. For $0 < i < n$, the coefficient of x^i in $((X + a)^n - (X^n + a))$ is $\binom{n}{i} a^{n-i}$. When n is prime, $\binom{n}{i} = 0 \bmod n$ and thus all coefficients are zero. When n is not prime, there exists some prime q with $q^k \parallel n$. Since

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

and $\gcd(a, n) = 1 \implies \gcd(a, q) = 1 \implies \binom{n}{q} a^{n-q} \not\equiv 0 \bmod q^k$, X^q has non-zero coefficient $\binom{n}{q} a^{n-q} \bmod n$, so $(X + a)^n \neq X^n + a \pmod{n}$. ■

2.1 A polynomial primality test

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

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

This lemma gives us a simple test to check primality but is inefficient: we have to compute all $n = 2^{\log n}$ coefficients of these two n-degree polynomials.

We can reduce the computational demand by evaluating the equation not just modulo n, but modulo some polynomial $X^r - 1$ for a small r. In other words, we check if the following equation is satisfied

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

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

2.2 The AKS Primality Test

AKS Primality Test

procedure IsPrime(n)
  if $n = a^b$ for $a \in \mathbb{N}, b > 1$
    output COMPOSITE
  $r \leftarrow$ smallest such that $o_r(n) > \log^2 n$
  if $\exists a \leq r, 1 < \gcd(a, n) < r$
    output COMPOSITE
  if $n \leq r$
    output PRIME
  for $a = 1, ..., \left\lfloor \sqrt{\phi(r)} \log n \right\rfloor$
    if $(X + a)^n \neq X^n + a (\mathrm{mod}\, X^r - 1, n)$
      output COMPOSITE
  output PRIME

Outline

1. Introduction 1
   - 1.1 The PRIMES problem 2
   - 1.2 Previous Works 3
   - 1.3 This work 4
   - 1.4 Preliminaries and Notation 5
2. The Algorithm 7
   - 2.1 A polynomial primality test 8
   - 2.2 The AKS Primality Test 10
3. Correctness 11
   - 3.1 Sufficiency 12
   - 3.2 Necessity 13
   - 3.3 Introspective numbers 14
   - 3.4 Size bounds 18
4. Run Time 23
   - 4.1 An upper bound on r 24
   - 4.2 A loose time bound 26
   - 4.3 Time improvements 29
   - 4.4 Conclusion 30

3.1 Sufficiency

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) > log^2 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
```

Let $n > 1$. We want to show IsPrime(n) outputs PRIME if and only if n is prime.

The $(\Longleftarrow)$ direction is trivial: when n is prime, the algorithm certainly cannot return COMPOSITE on lines 3 or 6 and the for loop will never identify n as composite by the discussion above.

3.2 Necessity

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) > log^2 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
```

Suppose towards a contradiction that n is composite but the algorithm outputs PRIME.

The algorithm could not have terminated on line 8, for then the previous step would have found a non-trivial factor. Thus, the algorithm terminates at the end, after the for loop.

Let r be as chosen in Line 4. Since $o_r(n) > \log^2 n \geq 1$, we have some prime divisor $p \mid n$ with $o_r(p) > 1$. The algorithm did not terminate in lines 6 or 8, so $p > r$.

Note that we have both $n, p \in \mathbb{Z}_r^*$. Let $\ell = \left\lfloor \sqrt{\phi(r)} \log n \right\rfloor$.

3.3 Introspective numbers

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

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

Lemma. If we fix f, the set of introspective numbers for f is closed under multiplication.

Lemma. If we fix m, the set of polynomials f for which m is introspective is closed under multiplication.

We claim that $p, n/p$ are introspective for all polynomials $X + a$, where $0 \leq a \leq \ell$.

The algorithm verifies that for every $0 \leq a \leq \ell$, we have

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

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

3.3 Introspective numbers

Now, as p is prime, we have

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

Putting both equations together, we have

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

Since $p \nmid r$, the polynomial $X^r - 1$ is square-free and thus the ring of polynomials mod $X^r - 1$ is reduced. So, we can cancel p'th roots and obtain

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

Thus, both $p, n/p$ are introspective for all $(X+a)$ where $0 \le a \le \ell$.

3.3 Introspective numbers

Our multiplicative closure lemmas then imply that for

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

every number in I is introspective for every polynomial in P.

3.3 Introspective numbers

Now consider the following two groups

- $G = \{i \bmod r : i \in I\}$
  - G is generated by $\frac{n}{p}, p$ in modulo r
  - This is a subgroup of $\mathbb{Z}_r^*$
  - $|G| > \log^2(n)$ as $o_r(n) > \log^2(n)$.
- $\mathcal{G} = \{f \bmod h(X), p : f \in P\}$ where $h(X)$ is an irreducible factor of degree $o_r(p)$ in $Q_r(X)$, where $Q_r(X)$ is the rth cyclotomic polynomial over $\mathbb{F}_p$ that divides $X^r - 1$.
  - This is generated by $X, ..., X + \ell$ in $\mathbb{F} = \mathbb{F}_p[X]/h(X)$.
  - This is a subgroup of the multiplicative group of $\mathbb{F}$.

3.4 Size bounds

We'll first establish two facts relating $|\mathcal{G}|$ and $|G|$.

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

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

We will conclude by showing $\binom{|G|+\ell}{|G|-1} > n^{\sqrt{|G|}}$ so that n is a power of p. This implies $n = p$ as if $n = p^k$ where $k \geq 2$, this will contradict IsPrime$(n) =$ PRIME as the algorithm would have found $n = p^k$ in the first step.

3.4 Size bounds

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

Proof. We show that any two distinct polynomials with degree $< |G|$ in P correspond to distinct elements of $\mathcal{G}$, and that there are at least $\binom{|G|+\ell}{|G|-1}$ such polynomials.

Let $f(X), g(X) \in P$ be distinct and have degree $< |G|$. Suppose towards a contradiction that $f(X) = g(X)$ in $\mathbb{F}$.

For any $m \in G$, we also have $f(X)^m = g(X)^m$ in $\mathbb{F}$ and thus, as m is introspective for f, g, we have $f(X^m) = g(X^m)$ in $\mathbb{F}$.

So, for all $m \in G$, X^m is a root of the polynomial $f(Y) - g(Y)$ in $\mathbb{F}$. This is a contradiction, as $\deg(f(Y) - g(Y)) < |G|$. So, $f(X) \neq g(X)$ in $\mathbb{F}$.

3.4 Size bounds

So, distinct polynomials of degree $< |G|$ in P correspond to distinct elements in $\mathcal{G}$. How many such polynomials are there?

Since $\ell = \left\lfloor \sqrt{\phi(r)} \log n \right\rfloor < \sqrt{r} \log n < r < p$, each of the elements $X, \ldots, X + \ell$ are distinct in $\mathbb{F}$. Also, $\deg h(X) > 1$, so none of the $X + a$ are zero in $\mathbb{F}$.

Thus, we can choose up to any $|G|$ of them (with multiplicity) to obtain a polynomial of degree $< |G|$; that is, there are at least $\binom{|G|+\ell}{|G|-1}$ polynomials of degree $< |G|$.

So, $|\mathcal{G}| \geq \binom{|G|+\ell}{|G|-1}$. ■

3.4 Size bounds

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

Proof. We prove the contrapositive: if n is not a power of p then $|\mathcal{G}| \leq n^{\sqrt{|G|}}$.

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 $|G| = t$, there exist $m_1 > m_2 \in \hat{I}$ with $m_1 = m_2 \bmod r$.

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

$$X^{m_1} = X^{m_2} \pmod{X^r - 1}$$

3.4 Size bounds

So, for any $f(X) \in P$, we have $m_1, m_2 \in \hat{I} \subseteq I$ 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 $\mathbb{F}$.

There are at most $m_1 \leq \left(\frac{n}{p} \cdot p\right)^{\lfloor \sqrt{t} \rfloor} \leq n^{\sqrt{t}}$ such roots so $|\mathcal{G}| \leq |P| \leq n^{\sqrt{t}} = n^{\sqrt{|G|}}$ as needed. ■

To conclude, we have by our first lemma that $|\mathcal{G}| \geq \binom{|G|+\ell}{|G|-1}$. Some routine arithmetic then shows $\binom{|G|+\ell}{|G|-1} > n^{\sqrt{|G|}}$. By our second lemma, n is a power of a prime. By the first step of our algorithm, we're done!

Outline

1. Introduction 1
   - 1.1 The PRIMES problem 2
   - 1.2 Previous Works 3
   - 1.3 This work 4
   - 1.4 Preliminaries and Notation 5
2. The Algorithm 7
   - 2.1 A polynomial primality test 8
   - 2.2 The AKS Primality Test 10
3. Correctness 11
   - 3.1 Sufficiency 12
   - 3.2 Necessity 13
   - 3.3 Introspective numbers 14
   - 3.4 Size bounds 18
4. Run Time 23
   - 4.1 An upper bound on r 24
   - 4.2 A loose time bound 26
   - 4.3 Time improvements 29
   - 4.4 Conclusion 30

4.1 An upper bound on r

Lemma. Let $\mathrm{LCM}(m)$ denote the lcm of first m numbers. For $m \geq 7$:

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

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

Proof (modified). $n < 10$ can be checked manually. Assume that $n \geq 10$. Then $B = \lceil \log^5(n) \rceil > 10$. Consider the smallest r that does not divide the product

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

Since r does not divide $n^i - 1$ for all $1 \leq i \leq \lfloor \log^2(n) \rfloor, o_r(n) > \log^2(n)$. Hence,

4.1 An upper bound on r

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

Now, by minimality of r, the numbers $1, \ldots, r-1$ divide the product. With the above inequality we get $\mathrm{LCM}(r-1) < 2^B$. Combine with the lemma that $\mathrm{LCM}(B) \geq 2^B$, this implies $r \leq B$, as needed. ■

4.2 A loose time bound

Lemma (Joachim von zur Gathen, Jürgen Gerhard '99). Addition, multiplication, and division operations between two m bits numbers (two degree d polynomials with coefficients at most m bits) can be performed in time $\tilde{\mathcal{O}}(m)$ ($\tilde{\mathcal{O}}(d \cdot m)$) steps.

4.2 A loose time bound

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) > log^2 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
```

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

Proof. Line 2 takes asymptotic time $\tilde{\mathcal{O}}(\log^3(n))$ by brute force search over $b = \mathcal{O}(\log n)$ and the previous lemma.

Line 4 can be done by trying $\mathcal{O}(\log^5(n))$ different r's. Each r needs at most $\mathcal{O}(\log^2(n))$ multiplications in modulo r, and so will take time $\tilde{\mathcal{O}}(\log^2(n)\log(r))$. This entire step takes $\tilde{\mathcal{O}}(\log^7(n))$ steps.

Line 5 computes the gcd of r numbers. Each gcd computation takes time $\mathcal{O}(\log n)$. This entire step takes $\mathcal{O}(r\log(n)) = \mathcal{O}(\log^6(n))$ steps.

4.2 A loose time bound

AKS Primality Test

1: **procedure** IsPrime(n)
2: **if** $n = a^b$ for $a \in \mathbb{N}, b > 1$
3: **output** COMPOSITE
4: $r \leftarrow$ smallest such that $o_r(n) > \log^2 n$
5: **if** $\exists a \leq r, 1 < \gcd(a, n) < r$
6: **output** COMPOSITE
7: **if** $n \leq r$
8: **output** PRIME
9: **for** $a = 1, ..., \left\lfloor \sqrt{\phi(r)} \log n \right\rfloor$
10: **if** $(X + a)^n \neq X^n + a \pmod{X^r - 1, n}$
11: **output** COMPOSITE
12: **output** PRIME

Line 9 verifies $\sqrt{\phi(r)} \log(n)$ equations.

Each equation 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 $\tilde{\mathcal{O}}\left(r \log^2(n)\right)$ 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 $\tilde{\mathcal{O}}\left(\log^{\frac{21}{2}}(n)\right)$.

4.3 Time improvements

Conjecture. 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. If the above conjecture holds then $r = \mathcal{O}\big(\log^2(n)\big)$ instead of $\mathcal{O}\big(\log^5(n)\big)$. This gives a time complexity of $\tilde{\mathcal{O}}\big(\log^6(n)\big)$.

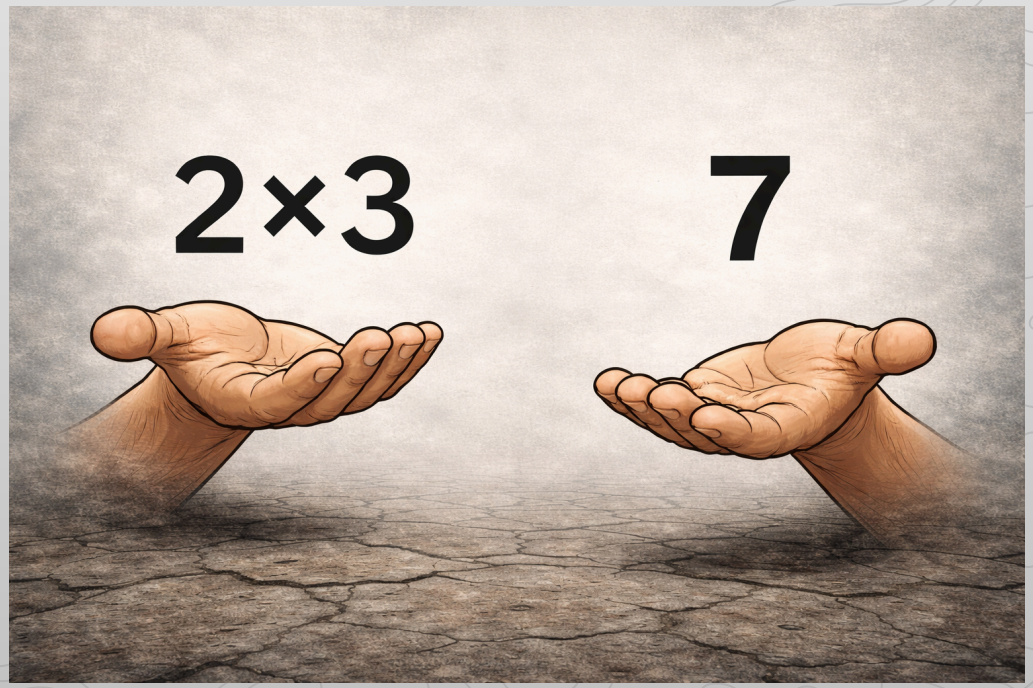
Lemma. (Baker, Harman '96) Denote $P(m)$ as the greatest prime divisor of m. There exists n_0 such that for all $x \geq n_0$,

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

Remark. The above lemma shows the algorithm has a time complexity of $\tilde{\mathcal{O}}\Big(\log^{\frac{15}{2}}(n)\Big)$.

4.4 Conclusion

We have presented a polynomial time algorithm that can test whether a number is prime.



Thank you to Professor Shubhangi Saraf for her invaluable insight during the semester and to our classmates for the many illuminating discussions.