Polynomial-Time Primality Testing

Aaryan Sukhadia

"Advanced" Minicourse, PROMYS 2022

Contents

1 Introduction

Remark. All logarithms mentioned are base 2.

What does it mean for an algorithm to be *rapid*? For our purposes it means we want the time it takes to run to scale well with input size.

Definition. A **polynomial time** algorithm that takes an input of $\log n$ bits will terminate in less than $C(\log n)^k$ steps for all inputs, for constants C, k.

In general, factoring a number cannot be done in polynomial time. A naive algorithm factoring n would have to check up to $\sqrt{n}$ in the worst case scenario, and so the number of steps clearly cannot be logarithmic in n.

In a similar vein, it was thought for a while that testing if a number was prime was not doable in a polynomial time. This was left unexplained for several decades.

2 The AKS Algorithm

In 2002, Agrawal, Kayal and Saxena found the first known algorithm for primality testing that ran in polynomial time. The algorithm is as follows: Given a number n,

1. `Check that` n `is not a perfect power`
2. `Check that` n `has no prime factors` $\leq 100(\log n)^5$
3. `Find some` $r \leq 100(\log n)^5$ `such that the order of` n `in` $(\mathbb{Z}/r\mathbb{Z})^\times$ `is` $\geq 9(\log n)^2$
4. `Check (in` $\mathbb{Z}[x]$`) that` $(x+a)^n \equiv x^n + a \pmod{n, x^r - 1}$

Remark. Though a valid primality test, you may notice AKS is not entirely *practical* to implement due to the incredible amount of memory required to run the algorithm for decently large numbers (which is the only realm in which its even relevant. Nevertheless, it still stands as an incredible theoretical breakthrough.

3 Proof of AKS

Step 1 and 2 are fairly obvious hurdles for composite numbers, i.e if n fails step 1 or step 2 then it trivially must be composite.

3.1 Validity of Step 3

Note that step 3 requires the existence of some number r with a special property, and it's not entirely intuitive that such a number is guaranteed to exist. We first deal with this.

Lemma 3.1.1. *If n has no factors $\leq 100(\log n)^5$, then there exists some $r \leq 100(\log n)^5$ such that the order of n modulo r is $\geq 9(\log n)^2$*

Proof. Let $R = 100(\log n)^5$ and $K = 9(\log n)^2$. Assume for the sake of contradiction that $\forall r \leq R, \operatorname{ord}_r(n) < K$. This would mean that $r \mid (n^j - 1)$, for some $j < K$. If this holds for every $r \leq R$, then:

$$\operatorname{lcm}(1, 2 \ldots R) \mid \prod_{j=1}^{K-1} (n^j - 1) < \prod_{j=1}^{K-1} n = n^{K(K-1)/2}$$

Thus, we have an upper bound on $\operatorname{lcm}(1, 2 \ldots R)$. Let us now find a lower bound.

Consider the integral $I := \int_0^1 (x(1-x))^M \mathrm{d}x$, for some $M \in \mathbb{N}$. Since $\forall x \in [0,1], x(1-x) \leq 1/4$, we have that $I \leq \frac{1}{4^M}$.
Expanding out the integral, we get:

$$\begin{aligned} I &= \int_0^1 x^M - Mx^{M+1} + \cdots + (-1)^M x^{2M} \\ &= \frac{1}{M+1} - \frac{M}{M+2} + \cdots + \frac{(-1)^M}{2M+1} \end{aligned}$$

The common denominator of all those fractions is $\mathrm{lcm}(M+1 \ldots 2M+1)$, and thus the common denominator divides $\mathrm{lcm}(1, 2 \ldots 2M+1)$. Let $I' = I \cdot \mathrm{lcm}(1, 2, \ldots 2M+1) \in \mathbb{N}$. Thus, we get:

$$\begin{aligned} 1 \leq I' &\leq \frac{1}{4^M} \mathrm{lcm}(1, 2 \ldots 2M+1) \\ \implies 4^M &\leq \mathrm{lcm}(1, 2 \ldots 2M+1) \end{aligned}$$

Letting $R = 2M+1$, we get $\mathrm{lcm}(1, 2 \ldots R) \geq 4^{(R-1)/2} = 2^{R-1}$. We now have the following in terms of K and R:

$$2^{R-1} \leq (1, 2 \ldots R) < n^{K(K-1)/2}$$

Plugging in $R = 100(\log n)^5$ taking the logarithm of the LHS of the inequality, we get:

$$\begin{aligned} \log\left(2^{100(\log n)^5 - 1}\right) &= 100(\log n)^5 - 1 \\ &> \frac{81}{2}(\log n)^5 + \frac{9}{2}(\log n)^3 \\ &= \log n(9(\log n)^2 + 1)9(\log n)^2/2 \\ &> \log n(9(\log n)^2 - 1)9(\log n)^2/2 \\ \implies 2^{R-1} &> 2^{(\log n)K(K-1)/2} = n^{K(K-1)/2} \end{aligned}$$

This gives us a contradiction, and thus there must be some $r \leq R$ with the desired property. $\square$

Remark. Note the technique we used for this proof: we assumed the negation, and created a mathematical object $\mathrm{lcm}(1, 2 \ldots R)$. Then, under the assumption of the negation, we derived upper and lower bounds for this object that end up contradicting one another. This is a common method of contradiction and will come up in the next proof.

Having the criterion for step 3 ensured, we now proceed to the most difficult part - showing why step 4 works.

3.2 Validity of Step 4

Theorem 3.2.1. *If n passes through all steps 1) to 4), then n is prime*

Proof. Suppose n passes step 4 and is composite, and let p be a prime factor of n. Note that due to step 1 and 2, $n \neq p^k$ and $p > 100(\log n)^5$. We know $(x+a)^n \equiv x^n + a \pmod{(n, x^r - 1)}$. Consider:

$$\begin{aligned}(x+a)^{n^2} &= ((x+a)^n)^n \\ &\equiv (x^n + a)^n \pmod{(n, x^r - 1)} \\ &\equiv x^{n^2} + a \pmod{(n, x^{nr} - 1)}\end{aligned}$$

Note we went from line 2 to 3 by taking x^n to be a variable. However, since $(n, x^{nr} - 1) \subset (n, x^r - 1)$, and using the fact we can repeat the process above for any power of n, we get:

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

Moreover, since $p \mid n$, we have $(n, x^r - 1) \subset (p, x^r - 1)$, which gives us:

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

Since p is prime, it follows from the Binomial Theorem and Fermat's Little Theorem that:

$$(x+a)^{p^j} \equiv x^{p^j} + a \pmod{(p, x^r - 1)}$$

We define $\mathcal{M} := \{n^i p^j : i, j \geq 0\}$. Then, the above statements can be written as:

$$\forall m \in \mathcal{M}, (x+a)^m \equiv x^m + a \pmod{(p, x^r - 1)} \tag{1}$$

Let k be the order of p in $(\mathbb{Z}/r\mathbb{Z})^\times$, define $q = p^k$, and consider the field $\mathbb{F}_q$. We take some $\gamma \in \mathbb{F}_q^\times$ with order r, which we know exists because $r \mid (q-1)$.

Since $a \leq r < p$, $a \in \mathbb{F}_q$. For each $m \in \mathcal{M}$, consider $(\gamma + a)^m \in \mathbb{F}_q$. Our congruence from Equation 1 tells us that $(x+a)^m = x^m + a + pf(x) + (x^r - 1)g(x)$.
Letting $x = \gamma$, we note that $\gamma^r - 1 = 0$, and $\mathbb{F}_q$ having characteristic p implies $pf(\gamma) = 0$. Thus, we get:

$$(\gamma + a)^m = \gamma^m + a \in \mathbb{F}_q, \forall m \in \mathcal{M}, \forall a \leq r$$

Consider G, the subgroup of $\mathbb{F}_q^\times$ generated by $(\gamma + a)$, $\forall a \leq r$. We will show that if a prime factor p of n does indeed exist, then G cannot exist.

We take the elements of $\mathcal{M}$ modulo r. They form a subgroup of $H \leq (\mathbb{Z}/r\mathbb{Z})^\times$. Let $|H| = h$.

Upper Bound of G

Suppose we have $m_1, m_2 \in \mathcal{M}$ such that $m_1 \equiv m_2 \pmod r$, and WLOG $m_1 > m_2$. Take $g = \prod_{a=1}^r (\gamma + a)^{e_a} \in G$. Then, $g^{m_1} = \prod_{a=1}^r \left((\gamma+a)^{e_a}\right)^{m_1} = \prod_{a=1}^r (\gamma^{m_1} + a)^{e_a}$, by construction. Similarly, $g^{m_2} = \prod_{a=1}^r (\gamma^{m_2} + a)^{e_a}$.
Since $\operatorname{ord}(\gamma) = r$, we have $\gamma^{m_1} = \gamma^{m_2} \implies g^{m_1} = g^{m_2}$. Thus, every $g \in G$ is a root of $x^{m_1 - m_2} \in \mathbb{F}_q[x]$, which tells us $|G| \leq m_1 - m_2$. If we can find such m_1, m_2 reasonably close together, we can restrict the order of G from above.

Let us consider all possible powers $n^i p^j$ such that $0 \leq i, j \leq \lfloor \sqrt{h} \rfloor$. These are all distinct since n is not a perfect power of p. Note that the number of possible pairs (i, j) is given by $(\lfloor \sqrt{h} \rfloor + 1)^2 > h$. By the Pigeonhole Principle, $\exists (i_1, j_1) \neq (i_2, j_2)$ such that $n^{i_1} p^{j_1} \equiv n^{i_2} p^{j_2} \pmod r$, since $|\mathcal{M} \pmod r| = h$. These two numbers will thus be our m_1, m_2. Note that by our construction, they can differ by at most $n^{\sqrt{h}} p^{\sqrt{h}} \leq n^{2\sqrt{h}}$. Thus:

$$|G| \leq n^{2\sqrt{h}} \tag{2}$$

Lower Bound of G

Consider the set:

$$W := \left\{ \prod_{a=1}^{r} (\gamma + a)^{e_a} : e_a \geq 0, \sum_{a=1}^{r} e_a \leq h - 1 \right\} \subseteq G \subseteq \mathbb{F}_q$$

We claim that the elements of W are distinct in $\mathbb{F}_q$. Suppose that $\prod(\gamma + a)^{e_a} = \prod(\gamma + a)^{f_a} \in \mathbb{F}_q$, where $\{e_a\}$ and $\{f_a\}$ are not all the same. Consider the polynomials $\prod(x+a)^{e_a}, \prod(x+a)^{f_a} \in \mathbb{F}_p[x]$, which must be distinct by unique factorization. We now form a new polynomial:

$$\Delta(x) := \prod (x + a)^{e_a} - \prod (x + a)^{f_a}$$

Note that $\deg(\Delta) \leq h - 1$, by our construction of W. We also know that $\Delta(\gamma) = 0$, by our assumption. Now, for all $m \in \mathcal{M}$, we observe that:

$$\prod (\gamma^m + a)^{e_a} = \prod \left((\gamma + a)^{e_a}\right)^m = \prod \left((\gamma + a)^{f_a}\right)^m$$

Thus, γ^m is a root of Δ as well, $\forall m \in \mathcal{M}$. Since $\text{ord}(\gamma) = r$, there are h distinct values of γ^m. However, this means that Δ has h roots, which is a contradiction. We conclude that our assumption must have been false and so all the elements of W are distinct.

We now bound the number of ways to produce such exponents $\{e_a\}$ from below. One way is we can simply consider that each e_a is either 0 or 1. The number ways to satsify $\sum_{a=1}^{r} e_a \leq h-1$ is $\binom{r}{0} + \binom{r}{1} + \cdots + \binom{r}{h-1}$. Since $r > h$, we have that this value is greater than or equal to:

$$\binom{h+1}{0} + \binom{h+1}{1} + \cdots + \binom{h+1}{h-1} \leq |W| \leq |G|$$

This gives us our lower bound of:

$$|G| \geq 2^{h+1} - h - 2 \geq 2^h \tag{3}$$

Contradiction

Combining Equations 2 and 3 together, we get:

$$\begin{aligned} 2^h \leq & |G| \leq n^{2\sqrt{h}} \\ \implies & h \leq 2\sqrt{h}\log n \\ \implies & \sqrt{h} \leq 2\log n < 3\log n \\ \implies & h < 9(\log n)^2 \end{aligned}$$

However, since the elements of $\mathcal{M}$ are generated by n and p, $h \geq \text{ord}_r(n) \geq 9(\log n)^2$.

Thus, we arrive at a contradiction, which means n *cannot* have a prime factor, and thus n itself must be prime.

□

4 Running Time Analysis

We go through step by step of the algorithm to show that it can and does indeed run in polynomial time. We first go through some facts of how many steps it takes to perform basic arithmetic operations.

- Adding or subtracting two n-bit numbers takes about n steps. This can be seen through basic algorithms like column addition

- Considering column multiplication and long division, doing these operations on n-bit numbers takes at most n^2 steps. However, using some tricks due to Gauss and Karatsuba, we can reduce the exponent to something along the lines of $n^{1.5}$ steps.
- Exponentiation in modular arithmetic can be done relatively fast due to repeated squaring and reducing after every multiplication. In particular, raising an n-bit number to an n-bit power modulo some n-bit number can be done in $< n^3$ steps.

Thus, we see how these operations are all *rapid*, in the sense that they can be performed in a time that is polynomial in the input size.

Definition (Big O notation)**.** Given functions $f(x)$ and $g(x)$, we say $f(x) = O(g(x))$ if $|f(x)| \leq Mg(x)$ for some constant M, and for all x greater than some finite x_0.

Thus, in the following running time analysis calculations, we will omit all constants and simply calculate the time complexity in terms of some polynomial of $\log n$.

Step 1

To show n is not a perfect power, we have to see if $n = m^k$ for $k \in [2, \log n]$. For each power k, we can perform a binary search from 1 to n to see if anything to the power of k equals n. Each exponentiation calculation will take $(\log n)^3$ steps, and each binary search will take $(\log n)$ steps. Doing this for every power of k, we get something less than $(\log n)^5$ steps.

Step 2

To check that n has no factors less than $100(\log n)^5$, we divide n by each k such that $k \in [2, 100(\log n)^5]$. Each division would take $(\log n)(\log k) \leq (\log n)^2$ steps. Doing this for every k, it would take at most $C(\log n)^7$ steps, for some constant $C < 1$.

Step 3

We want to find some $r \leq 100(\log n)^5$ such that the order of n modulo r is greater than $9(\log n)^2$. We run over all $9(\log n)^2 \leq r \leq 100(\log n)^5$, and check that none of $n, n^2, n^3 ... n^{9(\log n)^2}$ are 1.

To reduce each n modulo r, it takes $(\log n)(\log r) \leq (\log n)^2$ steps. Each multiplication, after reduction, takes $\leq (\log r)^2$. Thus, checking this property for a single r takes $(\log r)^2(\log n)^2 \leq (\log n)^3$ steps.
Doing this for all possibilites of r, we have at most $(\log n)^8$ steps.

Step 4

We need to check $\forall a \leq r$ whether $(x+a)^n \equiv x^n + a \pmod{(n, x^{r-1})}$. Recall that exponentiation is fairly rapid via repeated squaring and reducing modulo $(n, x^r - 1)$ every time.

Given polynomials $f, g \pmod{(n, x^r - 1)}$, to find $fg\ ((n, x^r-1))$ takes r^2 multiplications of coefficients $\leq n$. Recall that each such multiplication takes $\leq (\log n)^2$ steps, and reduction is very fast, so we can say the total time per multiplication and reduction is $(\log n)^2$. Thus, in total we have $\leq r^2 (\log n)^2 \leq (\log n)^{13}$ steps.

Doing this for every single $a \leq r \leq 100(\log n)^5$, we have $(\log n)^{18}$ steps.
Thus, the overall time complexity of AKS is of the order $O(\log n)^{18}$

5 Other Primality Tests

AKS is the only known polynomial time that is both *deterministic* and *general-purpose*, that is, it can definitively show a given number is composite or prime in any scenario. There are however, other, faster primality tests that are either probabilistic or only work for specific types of primes.

- The **Miller-Rabin** test is a probabilistic polynomial time test - that is, it determines whether a number is *likely* to be prime in polynomial time, but cannot confirm it in polynomial time. If the Riemann Hypothesis is true, it is known that the test can be done in polynomial time, in particular $O((\log n)^4)$ at worst.
- The **Lucas-Lehmer** test is deterministic, but only for potential Mersenne primes. This test can be done in $O((\log n)^3)$ steps.
- The **Elliptic-Curve Primality Test** is deterministic and general purpose, but its worst-case time complexity is currently unknown. It is heuristically suspected to be $O((\log n)^{5+\epsilon})$, but it is not even confirmed whether it always runs in polynomial time or not.

References

Mafs tingz.