3.8 The AKS Algorithm

We describe the algorithm in the version given by Lenstra/Bernstein. It is not trimmed to uttermost efficiency but aims at a transparent proof of polynomiality.

Input

An integer $n \geq 2$.

We measure the length of the input by the number ℓ of bits in the representation of n to base 2,

$$\ell = \begin{cases} \lceil \log_2 n \rceil, & \text{if } n \text{ is not a power of } 2, \\ k+1, & \text{if } n = 2^k. \end{cases}$$

Output

A Boolean value, coded as "COMPOSITE" or "PRIME".

Step 1

Catch powers of 2:

- If $n = 2$: output "PRIME", **end**.
- (Else) if n is a power of 2: output "COMPOSITE", **end**.

  We recognize this case by $\log_2 n$ being an integer.

From now on we may assume that n is not a power of 2, and $\ell = \lceil \log_2 n \rceil$.

Step 2

We precompute a big number $N \in \mathbb{N}$ as

$$N = 2n \cdot (n-1)(n^2-1)(n^3-1)\cdots(n^{4\ell^2}-1) = 2n \cdot \prod_{i=1}^{4\ell^2} (n^i - 1).$$

This number is huge, but more importantly:

- The number $4\ell^2$ of multiplications is polynomial in ℓ.
- From
  $$N \leq 2n \cdot n^{\sum_{i=1}^{4\ell^2} i} = 2n \cdot n^{\frac{4\ell^2(4\ell^2+1)}{2}} \leq 2n \cdot n^{16\ell^4},$$
  we conclude that
  $$k := \lceil \log_2 N \rceil \leq 1 + (16\ell^4 + 1) \cdot \ell$$
  is polynomial in ℓ.

We repeatedly use this integer k in the following. We have $N < 2^k$, and k is the smallest positive integer with this property.

Requirements

We have to find positive integers r and s that satisfy the following requirements:

1. r and n are coprime.
2. The integer interval $[1, \ldots, s]$ contains no prime divisor of n.
3. For each divisor $d \mid \frac{\varphi(r)}{q}$, where $q = \operatorname{ord}_r n$,
$$\binom{\varphi(r) + s - 1}{s} \geq n^{2d \cdot \lfloor \frac{\varphi(r)}{d} \rfloor}.$$
4. The primality criterion: For all $a = 1, \ldots, s$
$$(X + a)^n \equiv X^n + a \pmod{(n, X^r - 1)}.$$

Step 3

We choose r as the smallest prime that doesn't divide N. Then r also doesn't divide n. In particular requirement 1 is satisfied.

Why can we find r with polynomial cost?

By one of the extensions of the prime number theorem, equation (2), we have
$$\prod_{p \leq 2k,\, p \text{ prime}} p = e^{\vartheta(2k)} > 2^k > N.$$
Thus not all primes $< 2k$ divide N.

With costs that are at most quadratic in $2k$, and thus polynomial in ℓ, we get the list of all primes $\leq 2k$ (using Eratosthenes' sieve).

Step 4

Set $s := r$. Then requirement 2 is not necessarily satisfied. Hence we run through the list of primes $p < r$ that is known from step 3:

- If $p = n$: Output "PRIME", **end**.

  [This can happen only for "small" n since n grows exponentially with ℓ but r only polynomially.]

- (Else) If $p|n$: Output "COMPOSITE", **end**.

If we reach this point in the algorithm, then s satisfies requirement 2.

Requirement 3

To prove requirement 3 we start with the observation that $q := \operatorname{ord}_r n > 4\ell^2$.

> Otherwise $n^i \equiv 1 \pmod r$ for some i with $1 \le i \le 4\ell^2$, hence $r \mid n^i - 1 \mid N$, contradiction.

Now assume d divides $\frac{\varphi(r)}{q}$. Then

$$\begin{aligned} d &\le \frac{\varphi(r)}{q} < \frac{\varphi(r)}{4\ell^2}, \\ 2d \cdot \lfloor \sqrt{\frac{\varphi(r)}{d}} \rfloor &\le 2d \cdot \sqrt{\frac{\varphi(r)}{d}} = \sqrt{4d\varphi(r)} < \frac{\varphi(r)}{\ell} < \frac{\varphi(r)}{{}^2\log n}, \\ n^{2d \cdot \lfloor \sqrt{\frac{\varphi(r)}{d}} \rfloor} &< n^{\frac{\varphi(r)}{{}^2\log n}} = 2^{\varphi(r)}. \end{aligned}$$

On the other hand $\varphi(r) \ge 2$, so

$$\binom{\varphi(r) + s - 1}{s} = \binom{\varphi(r) + r - 1}{r} = \binom{2\varphi(r)}{\varphi(r) + 1} \ge 2^{\varphi(r)}.$$

Hence requirement 3 is satisfied.

Step 5

Next we check requirement 4,

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

in a loop for $a = 1, \dots, r$. The number of iterations is at most r, thus $\le 2k$, hence polynomial in ℓ. During each iteration we have two binary power computations, hence a total of at most 4ℓ multiplications, the factors being polynomials of degree $< r$—polynomial in ℓ—with coefficients of size $< n$, hence of bitlength polynomial in ℓ.

- If an a violates requirement 4, then output “COMPOSITE”, **end**.

Otherwise all a satisfy requirement 4, therefore n is a prime power by the AKS criterion.

Step 6

Finally we must decide whether n is a proper prime power. Since the primes $\le r$ don't divide n, we only have to check in a loop for t with $1 < t < \log_r n$:

- If $\sqrt[t]{n}$ is integer: Output “COMPOSITE”, **end**.

The number of iterations is $\le \ell$, and the test in each single iteration also takes polynomial cost, if we compute $\lfloor \sqrt[t]{n} \rfloor$ by a binary search in the interval $[1 \dots n - 1]$.

- If the algorithm reaches this point, output "PRIME", **end**.

This completes the proof of:

Theorem 1 *The AKS algorithm decides the primality of* n *with costs that depend polynomially on* $\log n$.