Algorithm Theory

Chapter 7
Randomized Algorithms

Part II:
Primality Testing

Fabian Kuhn

Primality Testing

Problem: Given a natural number $n \geq 2$, is n a prime number?

Simple primality test:

1. **if** n is even **then**
2. **return** $(n = 2)$
3. **for** $i := 1$ **to** $\lfloor \sqrt{n}/2 \rfloor$ **do**
4. **if** $2i + 1$ divides n **then**
5. **return false**
6. **return true**

If n is not prime, one of the prime factors p is $p \leq \lfloor \sqrt{n} \rfloor$:

$$2i + 1 \leq \lfloor \sqrt{n} \rfloor \Longrightarrow i \leq \left\lfloor \frac{\sqrt{n}}{2} \right\rfloor$$

- **Running time:** $O(\sqrt{n})$

Size of the input $O(\log n)$ bits:

$\sqrt{n}$ is *exponential* in the size of the input.

A Better Algorithm?

- How can we test primality efficiently?
  - We need a little bit of basic number theory…

$\mathbb{Z}_p^* = \{1, \dots, p-1\}$, multiplication mod p

Square Roots of Unity: In $\mathbb{Z}_p^*$, where p is a prime, the only solutions of the equation $x^2 \equiv 1 \pmod{p}$ are $x \equiv \pm 1 \pmod{p}$

$$\begin{aligned} x^2 &\equiv 1 \pmod{p} \\ x^2 - 1 &\equiv 0 \pmod{p} \\ (x+1)\cdot(x-1) &\equiv 0 \pmod{p} \end{aligned}$$

$\Rightarrow$ $\boldsymbol{(x+1)\cdot(x-1) = c \cdot p}$

For an integer c

p is a prime factor of $x+1$ or of $x-1$:

$x + 1 \equiv 0 \pmod{p}$ or

$x - 1 \equiv 0 \pmod{p}$

$\Leftarrow$

Not true if p is not prime:

$p = 15, \qquad x = 4$

$x^2 = 16 \equiv 1 \pmod{15}$

- If we find an $x \not\equiv \pm 1 \pmod{n}$ such that $x^2 \equiv 1 \pmod{n}$, we can conclude that n is not a prime.

Algorithm Idea

Claim: Let $p > 2$ be a prime number such that $p - 1 = 2^s d$ for an integer $s \geq 1$ and some odd integer $d \geq 1$. Then for all $a \in \mathbb{Z}_p^*$,

$$a^d \equiv 1 \pmod{p} \;\; \textbf{or} \;\; a^{2^r d} \equiv -1 \pmod{p} \;\; \text{for some } 0 \leq r < s.$$

Proof:

Recall that $x^2 \equiv 1 \pmod{p} \Leftrightarrow x \equiv -1, 1 \pmod{p}$

- **Fermat's Little Theorem:**
  For every prime p and all $a \in \mathbb{Z}_p^*$: $a^{p-1} \equiv 1 \pmod{p}$
- Consider $x_0, x_1, \dots, x_s$, where $x_i = a^{\delta_i}$ for $\delta_i = \frac{p-1}{2^i} = 2^{s-i} \cdot d$

$$\underbrace{\delta_0 = p - 1}_{= 2^s \cdot d} \quad \underbrace{\delta_1 = \frac{p-1}{2}}_{= 2^{s-1} \cdot d} \quad \underbrace{\delta_2 = \frac{p-1}{4}}_{= 2^{s-2} \cdot d} \quad \cdots \quad \underbrace{\delta_{s-1} = \frac{p-1}{2^{s-1}}}_{= 2 \cdot d} \quad \underbrace{\delta_s = \frac{p-1}{2^s}}_{= d}$$

- $\forall i < s : x_i = x_{i+1}^2$, thus $x_i \equiv 1 \pmod{p} \Longrightarrow x_{i+1} \equiv -1, 1 \pmod{p}$
- Fermat's Little Theorem $\Longrightarrow x_0 \equiv 1 \pmod{p}$
- Thus: $\forall i \leq s : x_i \equiv 1 \pmod{p}$ or $\exists i \leq s : x_i \equiv -1 \pmod{p}$
  - This directly implies the claim.

Primality Test

We have: If n is an odd prime and $n - 1 = 2^s d$ for an integer $s \geq 1$ and an odd integer $d \geq 1$. Then for all $a \in \{1, \dots, n-1\}$,

$$a^d \equiv 1 \pmod{n} \;\; \textbf{or} \;\; a^{2^r d} \equiv -1 \pmod{n} \;\; \text{for some } 0 \leq r < s.$$

$(*)$

Idea: If we find an $a \in \{1, \dots, n-1\}$ such that

$$a^d \not\equiv 1 \pmod{n} \;\; \textbf{and} \;\; a^{2^r d} \not\equiv -1 \pmod{n} \;\; \text{for all } 0 \leq r < s,$$

we can conclude that n is not a prime.

$\neg(*)$

- For every odd composite $n > 2$, at least $3/4$ of all $a \in \{2, \dots, d-2\}$ satisfy the above condition $\neg(*)$.
- How can we find such a *witness* a efficiently?

Idea: pick a at random

Miller-Rabin Primality Test

- Given a natural number $n \geq 2$, is n a prime number?

Miller-Rabin Test:

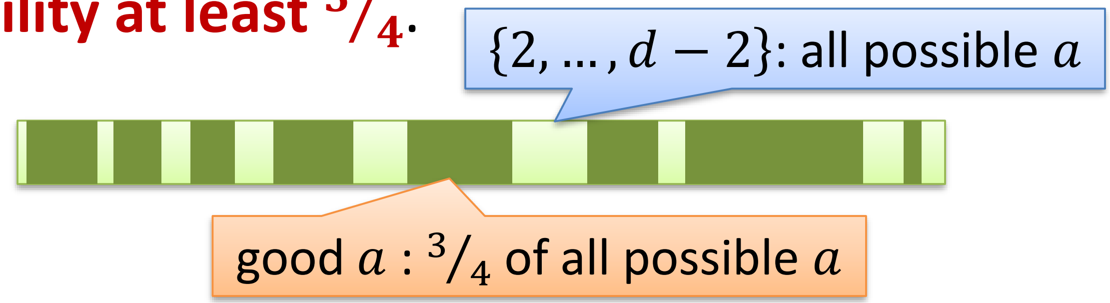
if n is even **then return** $(n = 2)$
compute s, d such that $n - 1 = 2^s d$;
choose $a \in \{2, \dots, n-2\}$ uniformly at random;
$x := a^d \bmod n$;
if $x = 1$ **or** $x = n - 1$ **then return probably prime;**
for $r := 1$ **to** $s - 1$ **do**
    $x := x^2 \bmod n$;
    if $x = n - 1$ **then return probably prime;**
return composite;

$(*)$ holds

Analysis

Theorem:

- If n **is prime**, the Miller-Rabin test **always** returns **probably prime**.
- If n **is composite**, the Miller-Rabin test returns **composite** with **probability at least** $^3/_4$.



Proof:

- If n is prime, the test works for all values of a
- If n is composite, we need to pick a good witness a

Corollary: If the Miller-Rabin test is repeated k times, it fails to detect a composite number n with probability at most 4^{-k}.

Running Time

Cost of Modular Arithmetic:

- Representation of a number $x \in \mathbb{Z}_n$: $O(\log n)$ bits
- Cost of adding two numbers $x + y \bmod n$:

  Time: $O(\log n)$

- Cost of multiplying two numbers $x \cdot y \bmod n$:
  - Done naively, this takes time $O(\log^2 n)$
  - It's like multiplying degree $O(\log n)$ polynomials
    → use FFT to compute $z = x \cdot y$

Time: $O(\log n \cdot \log\log n \cdot \log\log\log n)$

Running Time

Cost of exponentiation $x^d \bmod n$:

- Can be done using $O(\log d)$ multiplications
- Base-2 representation of d: $d = \sum_{i=0}^{\lfloor \log_2 d \rfloor} d_i 2^i$
- **Fast exponentiation:**
  1. $y \coloneqq 1$;
  2. **for** $i \coloneqq \lfloor \log_2 d \rfloor$ **to** 0 **do**
  3. $\quad y \coloneqq y^2 \bmod n$;
  4. $\quad$ **if** $d_i = 1$ **then** $y \coloneqq y \cdot x \bmod n$;
  5. **return** y;
- **Example:** $d = 22 = 10110_2$

$$x^{22} = \left(\left(\left(\left(1^2 \cdot x\right)^2\right)^2 \cdot x\right)^2 \cdot x\right)^2$$

Running Time

Theorem: One iteration of the Miller-Rabin test can be implemented with running time $O(\log^2 n \cdot \log\log n \cdot \log\log\log n)$.

1. **if** n is even **then return** $(n = 2)$
2. compute s, d such that $n - 1 = 2^s d$;
3. choose $a \in \{2, \dots, n-2\}$ uniformly at random;

Time $O(\log n)$ (lines 1–3)

4. $x \coloneqq a^d \bmod n$; $\quad O(\log d) = O(\log n)$ multiplications
5. **if** $x = 1$ **or** $x = n - 1$ **then return probably prime;**
6. **for** $r \coloneqq 1$ **to** $s - 1$ **do** $\quad s = O(\log n)$ iterations
7. $\quad x \coloneqq x^2 \bmod n$; $\quad$ 1 multiplication per iteration
8. $\quad$ **if** $x = n - 1$ **then return probably prime;**
9. **return composite;**

$O(\log n)$ multiplications $\Longrightarrow$ time $O(\log^2 n \cdot \log\log n \cdot \log\log\log n)$

Deterministic Primality Test

- If a conjecture called the generalized Riemann hypothesis (GRH) is true, the Miller-Rabin test can be turned into a polynomial-time, deterministic algorithm

  → It is then sufficient to try all $a \in \{1, \dots, 2\ln^2 n\}$

- It has long not been proven whether a deterministic, polynomial-time algorithm exists

hides factors polynomial in $\log \log n$

- In 2002, Agrawal, Kayal, and Saxena gave an $\tilde{O}(\log^{12} n)$-time deterministic algorithm
  - Has been improved to $\tilde{O}(\log^6 n)$

- In practice, the randomized Miller-Rabin test is still the fastest algorithm