Randomized Primality Testing

COMP 3711H - HKUST
Version of 22/12/2014
M. J. Golin

Introduction

- Many algorithms require large primes,
  e.g., Universal Hashing and RSA public key cryptography.
  How can we find them?

- Known (Lagrange Prime Number Theorem) that a random n bit number has around a $1/n$ chance of being prime. So, if looking for a random n bit prime, can just choose a random 1000 bit number and check if it's prime. After average $O(n)$ steps will find a prime.

- How can we check if it's prime?
  Standard *Sieve of Eratosthenes* requires $O(\sqrt{N})$ time to check number N. If number has 1000 bits, that's $2^{\sqrt{N}}$ time.
  Much too slow to be useful.

- In this class we will see a *Randomized Algorithm* for checking primality that will run in $O(\log N)$ time (or $O(\log^3 N)$ bit operations).
  Until 2002, only randomized algorithms were known.
  Deterministic algorithms developed since then are still not as simple as the randomized ones, so randomized ones are still used.

Las Vegas vs. Monte Carlo

There are two very different types of Randomized Algorithms

- Las-Vegas Algorithms: Algorithms always give correct answer but their running time is random.

  All randomized algroithms we have seen so far are Las-Vegas.

- Monte Carlo Algorithms: Algorithm is deterministic but only has a given probability of being correct.

  Can run algorithm many times to push probability of correctness higher.

  The Rabin-Miller primality testing algorithm we will see, will be a Monte Carlo Algorithm.

$Prime(p, a)$

$Prime(p, a)$:
Input is p the number to check, and a, integer $1 < a < p$.

- If p prime, then $Prime(p, a)$ always returns True.
- If p composite, then $Prime(p, a)$ may return False or True.
  - If it returns False, a is a *proof* of compositeness.
  - Less than $1/4$ of a's will return True.

Lemma: If p is composite then

$$\left|\{a : 1 < a < p \text{ and } Prime(p, a) = True\}\right| \leq \frac{1}{4}.$$

The Algorithm

$Prime(p, a)$:
Input is p the number to check, and a, integer $1 < a < p$.

Algorithm:
For $i = 1$ to k
    Choose a at random from $\{2, 3, \ldots, p-1\}$
    If $Prime(p, a) == False$
        Return(p is composite with proof a).
Return(p is Prime).

If algorithm returns composite, the number is composite.
If algorithm returns prime the algorithm is only wrong with probability $\left(\frac{1}{4}\right)^k$.

For reference, if $k = 100$, program has higher chance of being wrong due to cosmic ray hitting computer memory than from always choosing bad a.

$Prime(p, a)$

Write $p - 1$ in form $2^t u$

Can be done in $O(\log p)$ time by successive division by 2
(or looking at trailing 0s in base 2 representation)

Calculate $a^u \bmod p$

$O(\log u) = O(\log p)$ time using repeated squarings,

and then, using $O(t) = O(\log p)$ more squarings, calculate sequence

$$a^u \bmod p,\ a^{2u} \bmod p,\ a^{2^2 u} \bmod p,\ \ldots,\ a^{2^t u} = a^{p-1} \bmod p$$

(i) If $a^{p-1} \bmod p \neq 1$
$\Rightarrow Prime(p, a) = False$

(ii) If $a^{p-1} \bmod p == 1$
and if $\exists s \geq 1$ s.t. $a^{2^{s-1} u} \bmod p \not\equiv 1, -1$ and $a^{2^s u} \bmod p == 1$
$\Rightarrow Prime(p, a) = False$

Else $Prime(p, a) = True$

Why Should This Work (i)

$$a^u \bmod p,\ a^{2u} \bmod p,\ a^{2^2 u} \bmod p,\ \ldots,\ a^{2^t u} = a^{p-1} \bmod p$$

(i) If $a^{p-1} \bmod p \neq 1$
$\Rightarrow Prime(p, a) = False$

Fermat's Little Theorem is that,

if p is prime $\Rightarrow \forall a < p$, $a^{p-1} \bmod p = 1$.

$\Rightarrow$ if $a^{p-1} \bmod p \neq 1$, a is a witness that p is not prime.

Why Should This Work (ii)

$$a^u \bmod p,\ a^{2u} \bmod p,\ a^{2^2 u} \bmod p,\ \ldots,\ a^{2^t u} = a^{p-1} \bmod p$$

Unfortunately the first condition is not sufficient. There are some composite numbers, p, such that $a^{p-1} = 1 \bmod p$ for all $1 < a < p$. These numbers are called Carmichael numbers. While relatively "rare", there are still an infinite number of them.

We therefore use the second condition:

(ii) If $a^{p-1} \bmod p == 1$
and if $\exists s \geq 1$ s.t. $a^{2^{s-1}u} \bmod p \neq 1, -1$ and $a^{2^s u} \bmod p == 1$
$\Rightarrow Prime(p, a) = False$

This works because if p is prime and $x^2 = 1 \bmod p$ then

p divides $x^2 - 1 = (x-1)(x+1)$,

i.e., p divides $(x+1)$ or p divides $(x-1)$, i.e., $x = \pm 1 \bmod p$.

So if red condition is true $\Rightarrow$ a is a witness that p is not prime.
Because for $x = a^{2^{s-1}u} \bmod p$, $x \neq \pm 1 \bmod p$, but $x^2 = 1 \bmod p$.

Example

$p = 561$ is a Carmichael number.
$561 = 3 \cdot 7 \cdot 11$ so it is not prime.
Yet, for every $2 < a < 561$, $a^{560} = 1 \bmod p$.

$p - 1 = 2^4 \cdot 35$.
If we choose $a = 7$ then, $\bmod 561$, we calculate

$$a^{35} = 241, \quad a^{2 \cdot 35} = 298, \quad a^{4 \cdot 35} = 166, \quad a^{8 \cdot 35} = 67, \quad a^{16 \cdot 35} = 1.$$

This provides a proof of compositeness since
$x = 67 \neq \pm 1 \bmod 561$ but $x^2 = 1 \bmod 561$.

- We just saw that both conditions (i) and (ii) provide witness a that p is not prime
- The last piece is that it is possible to prove that, if p is composite, then at least $3/4$ of the numbers a between 2 and $p-1$ are witnesses from condition (i) or condition (ii).

This implies the lemma that was the source of the probabilistic gurantee of correctness of the algorithm.

Lemma: If p is composite then

$$\left|\{a \,:\, 1 < a < p \text{ and } Prime(p,a) = True\}\right| \leq \frac{1}{4}.$$

Wrap Up

- Developed a $O(\log p)$ procedure that checks to see if $1 < a < p$ is a *witness* that p is not prime
- Two cases
  - If p prime, no a is a witness
  - If p not prime, at least $3/4$ possible $a's$ are witnesses
- Pick k random a's and run test with them
  - If one of the a's is a witness, then p is absolutely not prime
  - If none of the $a's$ are witnesses, then p is prime with probability of error being at most $\frac{1}{4^k}$.