CPSC 500

Lecture 5, Sept. 16, 2013

Scribe: Sedigheh(Nasim) Zolaktaf

Introduction

First of all, in this lecture we introduce some concepts about modularity and the Euclidean algorithm. Secondly, a review of several primality testing algorithms is given. A primality test is an algorithm for determining whether an input number is prime. Amongst other fields of mathematics, it is used for cryptography. Primality tests do not generally give prime factors, only stating whether the input number is prime or not. Some primality tests prove that a number is prime, while others like Miller–Rabin prove that a number is composite. Therefore we might call the latter compositeness tests instead of primality tests.

Definition of modularity: Modular arithmetic is a system for dealing with restricted ranges of integers. We define a modulo n to be the remainder when a is divided by n; that is, if $a = qn + r$ with $0 \leq r < n$, then a module n is equal to r. this gives an enhanced notion of equivalence between numbers: a and b are congruent modulo n if they differ by a multiple of n, or in symbols,

$a \equiv b \pmod{n} \leftrightarrow a - b$ is divisible by $n \leftrightarrow a \bmod n = b \bmod n$

For example $0 \equiv 7 \equiv 14 \equiv -7 \equiv -35 \pmod{7}$.

"a mod n" is the remainder that is left from upon division by n and "a mod n" $\epsilon \{ 0, 1, \ldots, n \}$For example:

$8 \bmod 7 = 1$, $-2 \bmod 7 = 5$ and anything mod 7 $\epsilon \{0, 1, 2, 3, 4, 5, 6\}$.

Claim 1 (This claim is false!): if $a \not\equiv b \pmod{7}$ $4a \not\equiv 4b \pmod{7}$

Example: $a = 3$, $b = 0$, $a \not\equiv b \pmod{4}$ but $4a \equiv 4b \pmod{7}$

Claim 2 (cancellation rule): If ac ≡ bc (mod m) and gcd(m,c) =1 (i.e., m and c are relatively prime), then a ≡ b (mod m)

Proof. The first part of the hypothesis says that m|(ac –bc), that is, m divided c(a –b). Since the second part of the hypothesis says that m has no prime factor in common with c, we can conclude that m must divide (a-b). [1]

If n is prime this is simply

If ac ≡ bc (mod p) and c !≡ 0 (mod p) then a ≡ b (mod p).

Example: gcd(5, 6) = 1, 5a ≡ 5b (mod 6) → a ≡ b (mod 6)

Definition of inverse: If ab ≡ 1 mod n, we say that a and b are inverses mod n. Also, b is called the inverse of a mod n. The relation is symmetric: a is also the inverse of b mod n. Moreover, if n is a prime number the following are equivalent:

1) a !≡ 0 (mod p)
2) ∋ b integer for which ab ≡ 1 (mod p), we say a^{-1} ≡ b (mod p)

Fact 1: a has an inverse mod n if and only if $\gcd(a, n) = 1$. To see this, first suppose that $\gcd(a, n) = 1$. Then according to Bezout's lemma[1] we have $1 = ax + ny$ for some x, y. Then $1 \equiv ax + ny \equiv ax \bmod n$. So x is the inverse of a mod n. Conversely, if a has an inverse b mod n, we have $ab \equiv 1 \bmod n$ so $n|(ab - 1)$ and $ab - 1 = nq$. This shows that any divisor of a and n must also divide 1. Thus $(a, n) = 1$.

This theorem is not necessarily true if a has a common factor with n other than 1. For example 3 has no inverse mod 6: (proof by contraction: if d is the inverse of 3 then 3d – 1 ≡ 6c which is impossible!)

Euclidean algorithm

[1] Bezout's lemma is a theorem in the elementary theory of numbers: let a and b be integers, not both zero, and let d be their greatest common divisor. Then there exist integers x and y such that ax + by =d. [2]

The Euclidean algorithm, also called Euclid's algorithm, is an algorithm for finding the greatest common divisor of two numbers a and b.The Euclidean algorithm is an example of a P-problem whose time complexity is bounded by a quadratic function of the length of the input values (Bach and Shallit 1996).

Let a = bq + r, then find a number u which divides both a and b (so that a = su and b = tu), then u u also divides r since

r = a – bq = su – qtu = (s – qt)u.

Similarly, find a number v which divides b and r (so that b = sv' and r = t'v), then v divides a since

a = bq + r = s'vq + t'v = (s'q + t')v

Therefore, every common divisor of a and b is a common divisor of b and r, so the procedure can be iterated as follows:

$$
\begin{array}{llll}
q_1 = \left\lfloor \frac{a}{b} \right\rfloor & a = b\,q_1 + r_1 & r_1 = a - b\,q_1 & \\
q_2 = \left\lfloor \frac{b}{r_1} \right\rfloor & b = q_2\,r_1 + r_2 & r_2 = b - q_2\,r_1 & \\
q_3 = \left\lfloor \frac{r_1}{r_2} \right\rfloor & r_1 = q_3\,r_2 + r_3 & r_3 = r_1 - q_3\,r_2 & \\
q_4 = \left\lfloor \frac{r_2}{r_3} \right\rfloor & r_2 = q_4\,r_3 + r_4 & r_4 = r_2 - q_4\,r_3 & \\
q_n = \left\lfloor \frac{r_{n-2}}{r_{n-1}} \right\rfloor & r_{n-2} = q_n\,r_{n-1} + r_n & r_n = r_{n-2} & \\
 & & -q_n\,r_{n-1} & \\
q_{n+1} = \left\lfloor \frac{r_{n-1}}{r_n} \right\rfloor & r_{n-1} = q_{n+1}\,r_n + 0 & r_n = r_{n-1}\,/\,q_{n+1} &
\end{array}
\qquad (3)
$$

For integers, the algorithm terminates when q_{n+1} divides r_{n-1} exactly, at which point r_n corresponds to the greatest common divisor of a and b, gcd(a,b) = r_n.[3]

In other words, you start by dividing n by a (integer division with remainder). You then repeatedly divide the previous divisor by the previous remainder until there is no remainder. The last remainder you divided by is the greatest common divisor. We illustrate the Euclidean

algorithm in computing gcd(77,52).

$$77 \div 52 = 1 \text{ r } 25.$$
$$52 \div 25 = 2 \text{ r } 2$$
$$25 \div 2 = 12 \text{ r } 1$$
$$2 \div 1 = 2 \text{ r } 0$$

Since the last remainder you divided by is 1, gcd(77,52)=1.

An important consequence of the Euclidean algorithm is finding integers x and y such that

$ax + by = gcd(a,b)$

This can be done by starting with the equation for r_n, substituting for r_{n-1} from the previous equation, and working upward through the equations.

Note that r_i are just remainders, so the algorithm can be easily applied by hand by repeatedly computing remainders of consecutive terms starting with the two numbers of interest (with the larger of the two written first). As an example, consider applying the algorithm to gcd(a,b) = gcd(77,52).

Paradigm: To find the inverse of 52 (mod 77):

1. Carry out the Euclidean Algorithm. (Note: the inverse only exists if the gcd is 1.)

$$77 \div 52 = 1 \text{ r } 25$$
$$52 \div 25 = 2 \text{ r } 2$$
$$25 \div 2 = 12 \text{ r } 1$$
$$2 \div 1 = 2 \text{ r } 0$$
$$gcd(52,77) = 1$$

2. Rewrite the divisions (except the last one) in multiplicative form. (This and the next are often done all at once when you are familiar with the procedure.)

$$77 = 1 \times 52 + 25$$
$$52 = 2 \times 25 + 2$$
$$25 = 12 \times 2 + 1$$

3. Solve the equalities for the remainder terms.

$$25 = 77 - 1 \times 52$$
$$2 = 52 - 2 \times 25$$
$$1 = 25 - 12 \times 2$$

4. Substitute the remainders in turn into the equality below (work up the list).

$$1 = 25 - 12 \times 2$$
$$1 = 25 - 12 \times (52 - 2 \times 25)$$
$$1 = 25 \times 25 - 12 \times 52$$
$$1 = 25 \times (77 - 1 \times 52) - 12 \times 52$$
$$1 = 25 \times 77 - 37 \times 52$$

The last line is the equation of the form $77 \times -a + 52 \times b = 1$ that we were looking for, with $a = -25$ and $b = -37$. Since this is a moderately long process, you can and should check your work by actually computing $25 \times 77 - 37 \times 52 = 1925 - 1924 = 1$. The inverse of 52 (mod 77) is then -37, or 40 (= 77 - 37).

The trickiest part about this process is that some numbers are placeholders and should be left alone while other numbers are coefficients and should be computed with. In going from the first line to the third line in step 4, we are going from writing 1 as a linear combination of 2 and 25 to a linear combination of 25 and 52. So we substitute in for 2 in the second line and then collect like terms of 25 and 52. It is important in doing this that we don't actually multiply anything by the 25 or the 52, they are just placeholders. It gets worse when you operate on the third line where there are two 25s, the one on the left a coefficient and the one on the right a placeholder. [4]

Primality testing

There are lots of methods for testing whether n is prime or not. One approach is to continue dividing n by each number between 2 and n1/2 inclusive. If any of them divide evenly, then n is not prime because you found a factor. If n has no factors less than its square root, then n is prime. It is sufficient to check only for divisors less than or equal to n1/2 because if n = a*b, then a and b can't both exceed the square root of n. However for a large n the running time of the algorithm would be too much. In the following, we introduce Fermat's little algorithm and Miller-Rabin primality test, which are faster than the method described above but not as accurate.

Fermat's little theorem

Fermat's little theorem states that if p is a prime number, then for any integer a, the number $a^p - a$ is an integer multiple of p. In the notation of modular arithmetic, this is expressed as $a^p \equiv a \pmod{p}$.

For example, if a = 2 and p = 7, $2^7 = 128$, and $128 - 2 = 7 \times 18$ is an integer multiple of 7.

Fact 2: If a is not divisible by p and p is a prime number, Fermat's little theorem is equivalent to the statement that $a^{p-1} - 1$ is an integer multiple of p:

$a^{p-1} \equiv 1 \pmod{p}$

For example, if a = 2 and p = 7, $2^6 = 64$, and $64 - 1 = 63 = 7 \times 9$.

Fermat's little theorem is the basis for the Fermat primality test and is one of the fundamental results of elementary number theory. The theorem is named after Pierre de Fermat, who stated it in 1640. It is called the "little theorem" to distinguish it from Fermat's last theorem.

Fermat primality test:

The Fermat primality test is a probabilistic test to determine if a number is probable prime.

Inputs: n: a value to test for primality; k: a parameter that determines the number of times to test for primality
Output: composite if n is composite, otherwise probably prime
repeat k times:
pick a randomly in the range [1, n − 1]

if $a^{n-1} \not\equiv 1 \pmod{n}$, then return composite

return probably prime

The problem is that Fermats theorem is not an if-and-only-if condition; it doesn't say what happens when N is not prime. In fact, it is possible for a composite number N to pass Fermat's test (that is, $a^{N-1} \equiv 1 \bmod N$) for certain choices of a. for instance, 341 = 11. 31 is not prime, and yet $2^{340} \equiv 1 \bmod 341$. [5]

Miller-Rabin primality test

A primality test that provides an efficient probabilistic algorithm for determining if a given number is prime. It is based on the properties of strong pseudoprimes[2].

The algorithm proceeds as follows. Given an odd integer n, let $n = 2^r s + 1$ with odd. Then choose a random integer a with $1 \le a \le n-1$. If $a^s \equiv 1 \pmod{n}$ or $a^{2js} \equiv -1 \pmod{n}$ for some $0 \le j \le r-1$, then n passes the test. A prime will pass the test for all a.

The test is very fast and requires no more than $(1 + o(1)) \lg n$ multiplications (mod n), where lg is the logarithm base 2. Unfortunately, a number which passes the test is not necessarily prime. Monier (1980) and Rabin (1980) have shown that a composite number passes the test for at most 1/4 of the possible bases a. If n multiple independent tests are performed on a composite number, then the probability that it passes each test is $1/4^N$ or less. [6]

Conclusion

There are several methods for testing primality of integers. The best choice depends on the circumstances. Some of the methods are faster than others, while some popular tests like Fermats primality test and Miller-Rabin are actually only probabilistic algorithms that will occasionally falsely characterize a number as prime or composite.

---

[2] A strong pseudoprime to a base a is an odd composite number n with $n - 1 = d \cdot 2^s$ (for d odd) for which either

$a^d \equiv 1 \pmod{n}$ or $a^{d \cdot 2^r} \equiv -1 \pmod{n}$ for some $r = 0, 1, \ldots, s-1$

References

[1] Mathworld, “Cancellation Law”. http://mathworld.wolfram.com/CancellationLaw.html

[2] Wikipedia, “Bezouts identity”. http://en.wikipedia.org/wiki/B%C3%A9zout%27s_identity

[3] Mathworld, “Euclidean Algorithm”. http://mathworld.wolfram.com/EuclideanAlgorithm.html

[4] KSU, “The Euclidean Algorithm”. http://www.math.ksu.edu/~bennett/gc/831.html

[5] Wikipedia, “Fermat’s primality test”. http://en.wikipedia.org/wiki/Fermat_primality_test

[6] Wikipedia, “Miller-Rabin test”. http://en.wikipedia.org/wiki/Miller%E2%80%93Rabin_primality_test