Algorithm Analysis and Design (CS1.301)

Monsoon 2021, IIIT Hyderabad
09 October, Saturday (Lecture 13)

Number Theoretic Algorithms

Euclid's GCD Algorithm

Euclid's algorithm is a recursive way to find the GCD of two numbers.

```
function Euclid(a,b):
    if b = 0: return a
    return Euclid(b,a mod b)
```

Correctness

In order to prove its correctness, we only need to show that $\gcd(x, y) = \gcd(x - y, y)$. This is clear from the fact that any common factor of x and y is also a common factor of $x - y$ and y.

Analysis

Let us first prove a lemma: if $a \geq b$, then $a \mod b < \frac{a}{2}$. If this is true, then in two steps of the recursion, the first operand a reduces by at least half – two steps as it must get swapped back to the first position.

The lemma can be proved by taking two cases $b \leq \frac{a}{2}$ and $b > \frac{a}{2}$. If $b \leq \frac{a}{2}$, then $a \mod b < b \leq \frac{a}{2}$; and if $b > \frac{a}{2}$, then $a \mod b = a - b < \frac{a}{2}$.

Therefore, the algorithm runs in $2 \log a$ time, which is linear in the input size.

Euclid's Extended Algorithm

Consider the following lemma: if d is a common factor of a and b, and $d = ax+by$ for some integers x, y, then $d = \gcd(a, b)$.
For proving this, note that by the first condition, $d \leq \gcd(a, b)$, and by the second condition $\gcd(a, b) \leq d$. This proves the lemma.

Euclid's extended algorithm finds the values of x and y.

To compute $\gcd(25, 11)$, then Euclid's algorithm would go through the following steps:

$$\begin{aligned} 25 &= 2 \cdot 11 + 3 \\ 11 &= 3 \cdot 3 + 2 \\ 3 &= 1 \cdot 2 + 1 \\ 2 &= 2 \cdot 1 + 0. \end{aligned}$$

The extended algorithm, then, works like this:

```
function extended-Euclid(a,b):
    if b = 0: return (1,0,a)
    (x',y',d) = extended-Euclid(b,a mod b)
    return (y',x' - (a/b)y, d)
```

The correctness of this can be proven by noting that, by the recursion,

$$x' \cdot b + y' \cdot (a \mod b) = d.$$

But $a \mod b = a - \lfloor \frac{a}{b} \rfloor b$. This gives us

$$\begin{aligned} d &= x' \cdot b + y' \cdot (a \mod b) \\ &= x' \cdot b + y' \cdot (a - \lfloor \frac{a}{b} \rfloor b) \\ &= y' \cdot a + (x' - y' \cdot \lfloor \frac{a}{b} \rfloor). \end{aligned}$$

Multiplicative Inverses

x is the multiplicative inverse of a modulo N if $ax = 1 \mod N$. It is defined only when $\gcd(a, N) = 1$.
When it exists, the multiplicative inverse can be found in $O(n^3)$ time (where $n = \log N$). This can be done by running Euclid's extended algorithm on a and N.
Suppose we find that $ap + Nq = 1$. Taking $\mod N$, we find that $ap = 1 \mod N$; therefore p is the required number.

Public-Key Cryptography

It appears counterintuitive to be able to publish the key and yet stay secure. This can be explained intuitively by noting that although the key is published, its public *representation* makes it very hard to solve a certain essential problem, which can be solved rapidly in its private representation.

The encryption operation E_K is fast in R_2, and E_K^{-1} is very slow. This is the public representation. In the private representation R_1, the operation E_K^{-1} (decryption) is fast.

In the RSA cryptosystem, R_1 is the product of primes representation; R_2 is the decimal representation; and E_K is modular exponentiation $E_K(m) = m^e \mod K$.

The RSA System

Pick any two primes p, q and let $N = pq$. Let e be relatively prime to $(p-1)(q-1)$. The mapping $f(x) = x^e \mod N$ is a bijection on $\{0, 1, \ldots, N-1\}$.

Let d be the inverse of e modulo $(p-1)(q-1)$. Then we know that $(x^e)^d \mod N = x \mod N$.
This can be proved as follows:

$$x^{ed} - x = x^{1+k(p-1)(q-1)} - x$$

Now, Fermat's Little Theorem (proved below) tells us that $x^{p-1} = 1 \mod p$. Therefore the above quantity is divisible by p, and similarly by q. This makes it divisible by N.

Fermat's Last Theorem states that $a^{p-1} = 1 \mod p$ for $a < p$ and p prime. First, note that $a \cdot i \mod p$ are distinct for all $i < p$; from this we get

$$(p-1)! = a^{p-1} \cdot (p-1)! \mod p,$$

which means that $a^{p-1} = 1 \mod p$.