Next topic: RSA public-key cryptosystem

[Rivest-Shamir-Adelman '77]

Today: Math behind RSA

Next lecture: RSA & primality testing.

Modular arithmetic

For integer x,

$$x \bmod 2 = \text{least significant bit of } x = \begin{cases} 1 & \text{if } x \text{ is odd} \\ 0 & \text{if } x \text{ is even} \end{cases}$$

In general, for integer $N \geq 1$

$x \bmod N$ = remainder when divide x by N

= r where $x = qN + r$ for integers q, r.

which q? Doesn't matter.

Example: mod 3 has 3 equivalence classes:

$\ldots, -6, -3, 0, 3, 6, 9, \ldots$

$\ldots, -5, -2, 1, 4, 7, 10, \ldots$

$\ldots, -4, -1, 2, 5, 8, 11, \ldots$

Denote equivalence as $\equiv$

So $3 \equiv -6 \bmod 3$

& $-4 \equiv 11 \bmod 3$

$51 \equiv 1 \bmod 2$

Basic fact: if $x \equiv x' \bmod N$ & $y \equiv y' \bmod N$,

then $x + y \equiv x' + y' \bmod N$

& $xy \equiv x'y' \bmod N$

(so can replace x by x' & y by $y' \bmod N$)

Example: What's $2^{345} \bmod 31$?

hint: $345 = 5 \times 69$.

$2^{345} \equiv (2^5)^{69} \equiv (32)^{69} \equiv (1)^{69} \equiv 1 \bmod 31$

We'll work with HUGE numbers, e.g., 1000 or 4000 bit numbers.

Let n = # of bits

How long to compute $x+y$ or xy?

$x+y$: $O(n)$ time xy: $O(n^2)$ time.

Similar to multiplication, $x \div y$ takes $O(n^2)$ time,
& hence $x \bmod N$ takes $O(n^2)$ time.

For modular arithmetic,
for n-bit x, y, N,

To compute $x+y \bmod N$ (want result ~~~~ in $0, \ldots, N-1$)

1. add $x+y$
2. If $x+y \geq N$ then output $x+y-N$
   else output $x+y$.

$\Rightarrow O(n)$ time.

To compute $xy \bmod N$:

1. Compute xy (note, $xy \leq 2^{2n}$ so $\leq 2n$ bits)
2. Compute $xy \div N$ & output remainder

$\Rightarrow O(n^2)$ time since xy & N are $O(n)$ bits.

To compute $x^y \bmod N$:

Naive approach:

Compute $x \bmod N$
then $x^2 \bmod N$
then $x^3 \bmod N$
$\vdots$
$x^y \bmod N$

$\approx y$ rounds
but $y \leq 2^n$
So $O(n^2 2^n)$ time.

Better approach: Powers of 2.

Example: $y = 25 = (11001)_2$

then $x^{25} = x^{16} x^8 x \bmod N$

So compute $x \bmod N$,
then $x^2 \equiv (\downarrow)^2 \bmod N$,
then $x^4 \equiv (\downarrow) \bmod N$
$x^8 \bmod N$
$x^{16} \bmod N$

D&C view:

for even y, $x^y = \left(x^{y/2}\right)^2$

for odd y, $x^y = (x)\left(x^{\lfloor \frac{y}{2} \rfloor}\right)^2$

$\Longrightarrow$ n rounds, $O(n^2)$ time/round so $O(n^3)$ total time.

Modexp(x, y, N):

input: n-bit integers x, y, N
output: $x^y \bmod N$

if $y = 0$, return(1)
$z =$ Modexp$(x, \lfloor \frac{y}{2} \rfloor, N)$
if y is even
  then return$(z^2 \bmod N)$
  else return$(xz^2 \bmod N)$

Inverses: (Multiplicative inverses)

For real numbers $a \times \frac{1}{a} = 1$ ($\frac{1}{a}$ is the inverse of a)

What's $\frac{1}{a} \bmod N$?

Definition: z is the multiplicative inverse of $a \bmod N$ if $az \equiv 1 \bmod N$.

(Note there can be at most one such z within $0, 1, \ldots, N-1$)

denote as $z \equiv a^{-1} \bmod N$.

Examples:

$N = 14$

$1^{-1} \equiv 1 \bmod 14$
$3^{-1} \equiv 5 \bmod 14$
$5^{-1} \equiv 3 \bmod 14$
$9^{-1} \equiv 11 \bmod 14$
$13^{-1} \equiv 13 \bmod 14$

$2^{-1}, 4^{-1}, 6^{-1}, 7^{-1}, 8^{-1}, 10^{-1}, 12^{-1} \bmod 14$ do not exist

$N = 7$

$1^{-1} \equiv 1 \bmod 7$
$2^{-1} \equiv 4 \bmod 7$
$3^{-1} \equiv 5 \bmod 7$
$6^{-1} \equiv 6 \bmod 7$.

Theorem: $a^{-1} \bmod N$ exists iff $\underbrace{\gcd(a,N)=1}$

Say a & N are relatively Prime.

How to get $a^{-1} \bmod N$, if it exists?

Use the Euclid algorithm to check $\gcd(a,N)$.

Then use the Extended Euclid alg. to find $a^{-1} \bmod N$.

Euclid's algorithm computes $\gcd(x,y)$.

Based on following property:

Euclid's rule: for integers x,y where $x \geq y > 0$,

$$\gcd(x,y) = \gcd(x-y, y)$$

Proof:

if d divides x & y then it also divides $x-y$

(say $x = \alpha d$ & $y = \beta d$ then $x - y = d(\alpha - \beta)$).

Thus, $\gcd(x,y) \leq \gcd(x-y,y)$.

if d divides $x-y$ & y then it also divides x

so $\gcd(x-y,y) \leq \gcd(x,y)$.

From Euclid's rule we have:

$$\gcd(x,y) = \gcd(x \bmod y, y)$$

Why? $x \bmod y = x - ky$ where $k = \left\lfloor \frac{x}{y} \right\rfloor$

So we just apply Euclid's rule k times & we get

Now we have our gcd algorithm via D&C:

<u>Euclid(x, y):</u>

<u>input:</u> integers x, y where $x \geq y \geq 0$

<u>output:</u> $\gcd(x,y)$

if $y=0$, return(x)

else return(Euclid(y, $x \bmod y$)).

The algorithm is correct because of

What's the running time?

Each round takes $O(n^2)$ time to compute $x \bmod y$.

But how many rounds?

<u>Observation</u>: if $x \geq y$ then $x \bmod y < \frac{x}{2}$

<u>Proof:</u> if $y \leq \frac{x}{2}$ then $x \bmod y < y \leq \frac{x}{2}$ ✓

if $y > \frac{x}{2}$ then $x \bmod y = x - y < \frac{x}{2}$ ✓

∎

$(x, y) \longrightarrow (y, x \bmod y), (x \bmod y, ?)$

$\leq 1/2$

every 2 rounds goes down by factor 2

So $\leq 2n$ rounds.

Since $O(n^2)$ time/round

$\Rightarrow O(n^3)$ total time.

Suppose $\gcd(a,N)=1$, how do we get $a^{-1} \bmod N$?

Extended-Euclid is a generalization of Euclid's alg.
It returns the $\gcd(x,y)$ & it also returns integers α & β where $x\alpha+\beta y=d$ for $d=\gcd(x,y)$.

Suppose $d=1$ so $\gcd(x,y)=1$.

Then, $x\alpha+y\beta=1$

So $x\alpha+y\beta\equiv 1 \bmod y$

Since $y\beta\equiv 0 \bmod y$

$$x\alpha\equiv 1 \bmod y$$

So $\alpha\equiv x^{-1} \bmod y$

Therefore, if we run Ext-Euclid(a,N) & if $d=\gcd(a,N)$ is 1 then $\alpha\equiv a^{-1} \bmod N$ & $\beta\equiv N^{-1} \bmod a$.

Extended-Euclid algorithm is fairly simple but it's unclear why it works until you see the Proof of correctness (by induction).

Ext-Euclid(x,y):

input: integers x, y where $x \geq y \geq 0$
output: integers d, α, β where $d = \gcd(x,y)$
& $d = x\alpha + y\beta$.

if $y=0$, return$(x, 1, 0)$
$(d, \alpha', \beta') =$ Ext-Euclid$(y, x \bmod y)$
return$(d, \beta', \alpha' - \lfloor \frac{x}{y} \rfloor \beta')$

Summary:

For n-bit numbers a, b, N
in Poly(n) time we can compute:

$a^b \bmod N$

& $a^{-1} \bmod N$ if $\gcd(a,N)=1$
(if not then $a^{-1} \bmod N$ does not exist)