Lecture 5

Last Time: relatively prime #s $\Rightarrow$ $\gcd(a,b) = 1$

$\frac{x}{a}$ mod b $\Rightarrow$ $x \cdot a^{-1}$ mod b

a^{-1} would exists if a, and b are relatively prime

how do we find multiplicative inverse of a and b?

① run Euclid's $\gcd(a,b) = d$
check if $d \neq 1$ then $\nexists$ multiplicative inverse of a.

else we find it by

② run. Extended Euclid's Alg. to extract a^{-1} from gcd alg.

Interpretation of $\gcd(a,b) = d$

① it is the largest integer that divides both a, and b.

② smallest integer d that can be written in the form of $d = ax + by$.
$\Downarrow$
a^{-1}

Algorithm for gcd:

```
gcd (a,b)
    if b=0  return a
    return  gcd(b, a mod b)
```

Last time:

$$\underbrace{\gcd(a,b)}_{LHS} = \gcd\underbrace{(b, a \bmod b)}_{RHS}.$$

$$\begin{matrix} \geq \\ \leq \end{matrix} \Rightarrow =.$$

$gcd(\underset{a}{1035}, \underset{b}{759})$ $a \geq b \geq 0$

$gcd(a, b) = gcd(b, \underbrace{a \bmod b}_{r})$

Call #	a	b	r	
1^{st}	1035	759	276	$= \boxed{1035 - 1 \cdot 759}$
2^{nd}	759	276	207	$= \boxed{759 - 2 \cdot 276}$
3^{rd}	276	207	$\boxed{69}$ gcd	$= \boxed{276 - 1 \cdot 207}$
4^{th}	207	69	0	$= 207 - 3 \cdot 69$

last non-zero reminder in gcd

Correctness is shown by

$$d \leq gcd(b, a \bmod b)$$
$$d \geq gcd(a, b) \Rightarrow d = gcd(a, b)$$

Running time: $\underbrace{\#\text{ of recursive calls}}$ $*$ $\underbrace{\text{cost of per call}}_{\text{Integer division }(q, r)}$

$O(\text{value of } a)$ $O(n^2)$

A is a number of size M

$O(M) = O(2^n) \Rightarrow$ n-bit number.

$\Rightarrow$ magnitude of the input to gcd.

hint: ① consider the value of b.

and ② how much we reduce the (a, b) both (answer: by 1/2 in every 2 calls to gcd)

$\Rightarrow$ every 2 calls the # of bits in a and b will be reduced by 1 bits. $\Rightarrow O(n)$ recursive calls

Total time: $O(n) \cdot O(n^2) = O(n^3)$

Extended Euclid's Alg
① solve $\gcd(a,b) = g$ ✓
② write $g = ax + by$.

how: by back substitution

	a	b	r
3^{rd}	276	207	69
	....		0

g

$\Rightarrow 69 = 276 - 1 \cdot 207$
$= 276 - 1(759 - 2 \cdot 276)$
$= 276 - 1(759 - 2(1035 - 1 \cdot 759))$

$$69 = \underset{x}{3} \cdot \underbrace{1035}_{a} + \underset{y}{(-4)} \cdot \underbrace{759}_{b}$$

if g ~~here~~ $= 1$ then we would extract a^{-1} as x.

Revisiting Mod. Exponentiation:

$x^y \bmod N$ — $2^{(n)}$

both x and y: $\underline{x = 2^n}$, $\underline{y = 2^n}$ $\Rightarrow$ n-bit #s

instead of $(2^n) \bmod N$

we will use square and reduce Algorithm.

mod. Exp(x, y, N) they are all n-bit #s.

$\underline{\text{phase I}}$: compute powers of x by $\underline{\text{doubling}}$. to express y

EX: $7^{327} \bmod 853$

$7^2 \bmod 853 = 49$
$7^4 \bmod 853 = 695$
$7^8 \bmod 853 = 227$
$7^{16} \bmod 853$ ⋮ mod 853
⋮

what is the time complexity of phase I.?
$O(n)$ $\underbrace{\text{multiplications}}_{\text{each costs } O(n^2)}$ $= \lfloor \log y \rfloor$

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

phase II: multiply relevant powers - 2 at a time and do mod N reduction each time.

example (cont): $(256+64+4+2+1) = 327$

$$7^{327} = 7^{256} \cdot 7^{64} \cdot 7^{4} \cdot 7^{2} \cdot 7^{1} = 7 \mod \underbrace{853}_{N}$$

$$= 298 \cdot 128 \cdot 695 \cdot 49 \cdot 7 \mod 853$$

$$= 828 \cdot 695 \cdot 49 \cdot 7$$

$$= 538 \cdot 49 \cdot 7 \qquad \mod 853$$

$$= 772 \cdot 7$$

$$7^{327} = 256 \mod 853$$

Time: $\Theta(n) \cdot \Theta(n^2) = \Theta(n^3)$

Primality Testing: $p \mid p = 1$ and $1 \mid p = p$ no other dividers then p is a prime #.

Alg1(N)

input N
output Yes/No

for $x = 2, \ldots, N-1$ — n-bit # — $N = 2^n$, $n = \log N$

if $x \mid N$ then return No

return ~~else~~ YES.

Time complexity: $O(N)$: # of times the loop is executed. each time we have a division

$$O(N) \cdot O((\log N)^2) = \underbrace{\Theta(2^n)}_{} \cdot O(n^2)$$

exponential alg. in the # of bits to represent N.

Alg2(N)

$N = 2^n$ i.e. $n = \log N$

N is n-bit number

for $x = 2, 3, \ldots \sqrt{N}$
check if $x \mid N$?

$$\underbrace{O(\sqrt{N})}_{O(2^{n/2})} \cdot O((\log N)^2)$$

$$O(2^{n/2}) \cdot O(n^2) = O(2^{n/2}) \cdot O(n^2)$$

Alg 3(N)

observation: for testing primality we need to consider only integers of the form
1) $6q+1$
2) $6q+5$ or $6q-1$

for $q = 1$ to $\frac{\sqrt{N}}{6}$
- $x = 6q+1$
- $x' = 6q-1$
- if $x \mid N$ then False/NO
- if $x' \mid N$ then NO

"check"

return YES

then we can reduce the "check" operation to 1/3

$$\underbrace{O\left(\frac{\sqrt{N}}{6}\right)}_{O\left(\frac{2^{n/2}}{6}\right)} \cdot O(n^2)$$

$$O\left(\frac{2^{n/2}}{6}\right) \cdot O(n^2) = O(2^{n/2} \cdot n^2)$$

Fermat's Last Theorem!

"It is impossible for a cube to be the sum of 2 cubes.
I discovered a demonstration but the margins in this book is too small"

$x^n + y^n = z^n$ $n > 2$

Diophantus Arithmetica

Fermat's Little Theorem

$a^{N-1} \equiv 1 \mod N$ for $a < N$

if N is a prime #.

We will prove this theorem.

Alg 4 (N)

$\boxed{N = 2^k \quad k\text{. bits \#}}$

$k = \log N$

for $a = 2 \ldots N-1$

compute $a^{N-1} \mod N = b$

if $b \neq 1$ the return No/False

return true.

Time complexity: loop will be executed $O(N)$ times

$$\underbrace{O(N)}_{} \; \theta(k^3)$$

$$O(2^k) \cdot O(k^3) = O(2^k k^3)$$

Randomization can help to reduce the time complexity

$$x \mod N \Rightarrow x = \overset{?}{q} \cdot N + \overset{?}{r}$$