Lecture 6

Last time: primality testing.

Question: given $N = 17953$ is this a prime #?

We saw several deterministic algorithms.

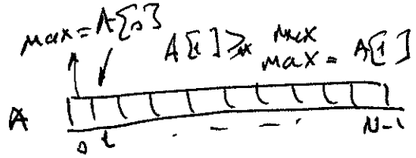
```
for i = ... N
    if i | N  return No
else YES.
```

N can be as large as 2^k if N is a k-bit #.

that were exponential time in the # of bits to represent N.

Today: Intro to Randomized Algorithms.

- Fermat's Little theorem
- obtain a randomized primality test.



Fermat's Little Theorem

If N is a prime then for all $1 \leq a \leq N-1$

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

Deterministic Alg.

Alg 4(N)

```
for a = 2 ... N-1
    if a^(N-1) ≠ 1 mod N  then No
else return YES.
```

Say N is a k-bit #. $\Rightarrow O(2^k \cdot k^3)$

Randomized Primality Test

Alg 5(N)

select an "a" at random in the range $[2, N-1]$

compute $Val = a^{N-1} \bmod N$.

if Val == 1 return TRUE (correct if actually prime, pseudoprime)

else return FALSE

Algorithm is always correct

If N is NOT prime and passes the theorem Test

then

N is ~~either~~ Charmichael # but $561 = 3 \cdot 11 \cdot 17$

eg: 561

$\gcd(a, 561) = 1$ $\quad a^{560} \equiv 1 \bmod 561$ ✓

when Alg 5 (N) returns TRUE

we have 2 scenarios

- Really prime
- or not prime

we have a mistake. error

with probability $\frac{1}{2}$. for a given a_1

lets repeat this Alg. by chossing randomly another a_2

$\frac{1}{2}$.

Question: what is the prob. that N is not prime but passes F.L.T. for both of these a_1 and a_2

$$\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2^2}$$

a_1, a_2, a_3, for all of these a values. what is the prob. of error $\frac{1}{2^3}$

$a_1 \dots a_m$ for m random a values then 2^{-m}

prob of error $\frac{1}{2^m} = 2^{-m}$

$\Rightarrow$ new Alg. Alg (N)

ran Alg 5 (N)

for m times

again N is a k-bit number $\Rightarrow O(m k^3)$

we have a poly. time primality test.

F.L.T.

$$a^{N-1} \equiv 1 \bmod N \qquad \gcd(a, N) = 1$$

N is prime.

$a > 0$

Proof sketch:

Set of residues from mod N excluding ϕ

$$\mathbb{Z}_N^* = \{1 \cdots N-1\}$$

observe that when we multiply $\mathbb{Z}_N^*$ with a constant a

$$a\,\mathbb{Z}_N^* = \{1 \cdot a, 2a \cdots N-1 a\}$$

ex: $\mathbb{Z}_5^* = \{1\ 2\ 3\ 4\}$

$a = 2$

$$2\,\mathbb{Z}_5^* = \{1 \cdot 2 \quad 2 \cdot 2 \quad 3 \cdot 2 \quad 4 \cdot 2\} \bmod 5$$

$$= \{2 \quad 4 \quad 1 \quad 3\}$$

$$\mathbb{Z}_N^* \equiv a\,\mathbb{Z}_N^*$$

$$\underbrace{1 * 2 * 3 \quad * N-1}_{(N-1)!} \equiv (a \cdot 1) * (a \cdot 2) * \cdots * (a \cdot N-1)$$

$$(N-1)! \equiv a^{N-1}\underbrace{(1 * 2 * \cdots * N-1)}_{(N-1)!}$$

divide both sides by $(N-1)!$ $\Rightarrow$ Cancellation property is based on Euclid's lemma

$$1 \equiv a^{N-1} \bmod N$$

$$a^{N-1} \equiv 1 \bmod N \quad \blacksquare$$

Euclid's lemma (division): If a prime $p \mid a \cdot b$ then p must divide at least one of them a or b.

So if any integer n s.t. $n \mid a \cdot b$ and $\gcd(a, n) = 1$ then $n \mid b \Rightarrow ax \equiv ay \bmod p$, $x \equiv y \bmod p$

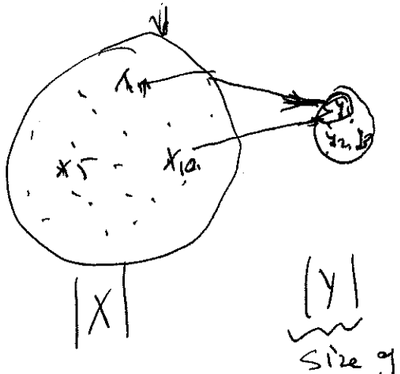
Another theorem for primality is Wilson's Thorm

Let $p > 1$ an integer. p is prime if and only if

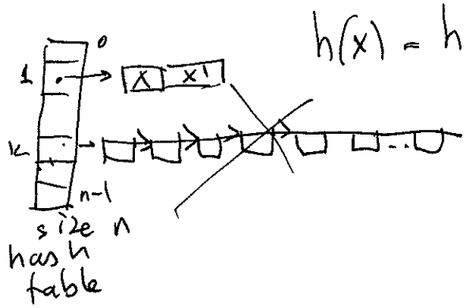
$$(p-1)! \equiv -1 \bmod p$$

Next topic: Hashing

$$h(x) = h(x')$$



- limit collisions
- hash func. must distribute data (X) to Y evenly
  ⇒ use randomization
- Consistency! each time we get the same result



$h(x) = h(x') =$ (1) index to the hash table

IP ADD: 32bits

128	32	6	2

What we need is to ensure that for a hash table with n entries. prob. of collision is $\frac{1}{n}$.

$$P(h(x) = h(x')) = \frac{1}{n}$$

the set/family of the hash functions that ensure this are called universal hash functions.