POSSIBLE TOPICS

(1) 2 Coloring, 2-SAT $\Rightarrow$ 3 Coloring? 3SAT?

(2) [~~RSA~~ Primality Testing; String Matching via Hashing] Probabilistic Alg.

Usual Prime Testing

Try dividing n by each integer starting from 2 until $\sqrt{n}$ (including it), if any divide evenly $\Rightarrow$ not prime!

If n is #, it has $O(\lg n)$ digits.

1 div $\Rightarrow$ $O(\lg^2 n)$ time, up to $\sqrt{n}$ of these

runtime $\Rightarrow$ $O(\sqrt{n}\,\lg^2 n)$

if $n \sim 10^{500}$ then this is too slow

For all prime numbers p, if $\gcd(a,p)=1$, then (a isn't a multiple of p)

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

(remainder when you divide by p is 1, or p divides evenly into $a^{p-1}-1$.)

$$67^{100} \equiv 1 \pmod{101}$$ Fermat's Thm.

A fact true about <u>ALL</u> primes.

Usually if $n \neq$ Prime, and $gcd(a,n) = 1$, then $a^{n-1} \not\equiv 1 \bmod n$. (Note if $gcd(a,n) \neq 1$ and $n \nmid a$, then n can't be prime anyway.)
~~or~~ and $a \nmid n$.

Idea behind our probabilistic algorithm:
try different values of a. If it's $(1 \leq a < n-1)$, if $gcd(a,n) \neq 1 \Rightarrow n$ is composite
Calculate $a^{n-1} \bmod n$. If you ever don't get 1, we know 100% it must be composite.
If this never happens, we answer "is probably prime"

This is ALMOST what we use.
PROBLEM = Carmichael #s
↳ Composite but for all a, s.t. $gcd(a,n) = 1$
$a^{n-1} \equiv 1 \bmod n$.

1st one = 561, 2nd = 1105, ...

Miller-Rabin: Cycle length for Carmichael #s isn't $n-1$, but it's so small, it ~~of~~ lines up.

Also for primes $a^{(p-1)}$, $a^{\frac{p-1}{2}} \equiv \pm 1 \pmod p$

rewrite $p-1 = 2^k \cdot p'$, where $p' \in$ Odd

$p = 101 \quad p-1 = 100 = 2^2 \times 25, \; k=2, \; p' = 25$

a^{25}, a^{50}, a^{100}

square square

for primes either whole sequence is $1, 1, 1, \ldots$

OR $\quad u, -1, 1$

Carmichael #s this doesn't always happen.

For one value of a, test fail less than 50% ~~50%~~ of the time.

So if we run test w/ 100 values of a and it reports, "IS probably prime" then it's correct $1 - 2^{-100}$ of the time.

Hashing for the purposes of probabilistic String Matching

Text: GACCATTACCATGCATC n

Pattern: "TTACC" m

Brute Force : O(nm)

Simple hash function for a String:

large prime

```
int val = 0;
for (int i=0; i < S.length; i++)
    val = (26 * val + S.charAt(i) - 'A') % mod;
```

Some use 29

$TTACC = T\times26^4 + T\times26^3 + A\times26^2 + C\times26^1 + C\times26^0$

Idea: Calculate hash of pattern

Then, calculate hash of the ~~first~~ first m letters of text and compare. (If not equal ⇒ not match)
(If equal ⇒ double check strings!)

Then, we want to slide our window over quickly!

Known: hash GACCA ? $G\times26^4 + A\times26^3$

How Compute hash ACCAT

w/o a for loop!

X GACCA = $G \times 26^4 + A \times 26^3 + C \times 26^2 + C \times 26^1 + A \times 26^0$

Y ACCAT = $A \times 26^4 + C \times 26^3 + C \times 26^2 + A \times 26^1 + T \times 26^0$

$(X - G \times 26^4) \times 26 + \text{last term} = Y$

$O(1)$ transition

powers of 26 need to be saved!

Run-time $= O(n+m)$