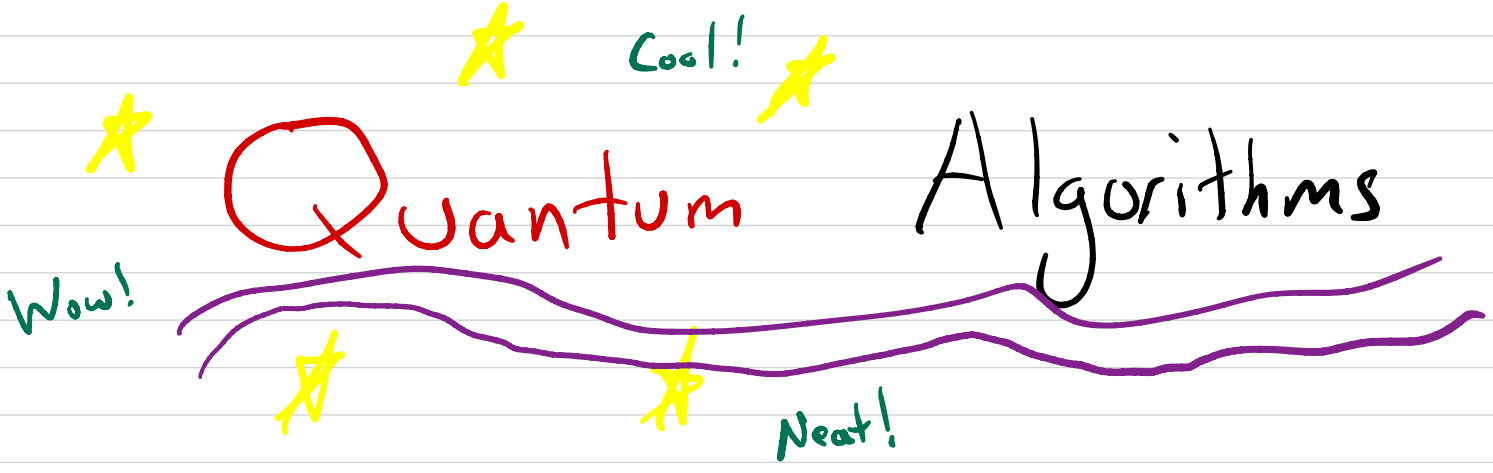
Today: Randomized Algorithms

Next class:



Randomized Algorithms

Algorithms which uses random bits to solve a problem

Allowed to fail with some probability $\leq 5\%$

Sometimes can be much faster than deterministic

Integer factorization

Given a 500 digit number N

find its prime factorization $N = p_1 p_2 \cdots p_k$

Algorithm: Check every integer $1 \leq x \leq \sqrt{N}$ to see if x divides N

Runtime:

Suppose $N \approx 10^{500}$

Then $\sqrt{N} \approx \sqrt{10^{500}} = 10^{250}$

# of atoms in universe $< 10^{106}$

N has n digits $\Rightarrow$ needs $10^{n/2}$ time

Inefficient!

General number field sieve: Factor an n bit number in time $\approx c^{n^{1/3}\log(n)^{2/3}}$

Factoring is believed to be hard!

But very important...

RSA-250: 250 digits (factored in 2020)

RSA-896: 270 digits $75,000

RSA-2048: 618 digits $200,000

(easy with a quantum computer)

Primality testing

Given an n digit number N
determine if it's prime or composite

Idea 1: Factor it! Runs in time $c^{n^{1/3}\log(n)^{2/3}}$ ☹

Idea 2: Use something else about prime numbers...

Fermat's little theorem

If N is prime then $a^{N-1} \equiv 1 \pmod{N}$
for all $a \in \{1, \ldots, N-1\}$

Fermat Test (N)

1. Pick $a \in \{1, \ldots, N-1\}$ uniformly at random
2. If $a^{N-1} \equiv 1 \pmod{N}$, output "prime"

   Otherwise, output "composite"

Fact: If N is prime, always outputs "prime"

$N = 12 = 3 \cdot 4$

not coprime: $a = 2$ — $2^{11} \not\equiv 1 \pmod{12}$ ← even

not coprime: $a = 3$ — $3^{11} \not\equiv 1 \pmod{12}$

coprime: $a = 5$ — $5^{11} \equiv 1 \pmod{12}$???

$N = p^2$, p large prime $\Rightarrow$ only $\frac{1}{p}$ of a's are not coprime with N

Test only good if $a^{N-1} \equiv 1 \pmod{N}$ for lots of coprime a

Carmichael numbers

Composite numbers N s.t.

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

for all a coprime to N

Pass the Fermat test for all coprime a

$N = 561 = 3 \cdot 11 \cdot 17$

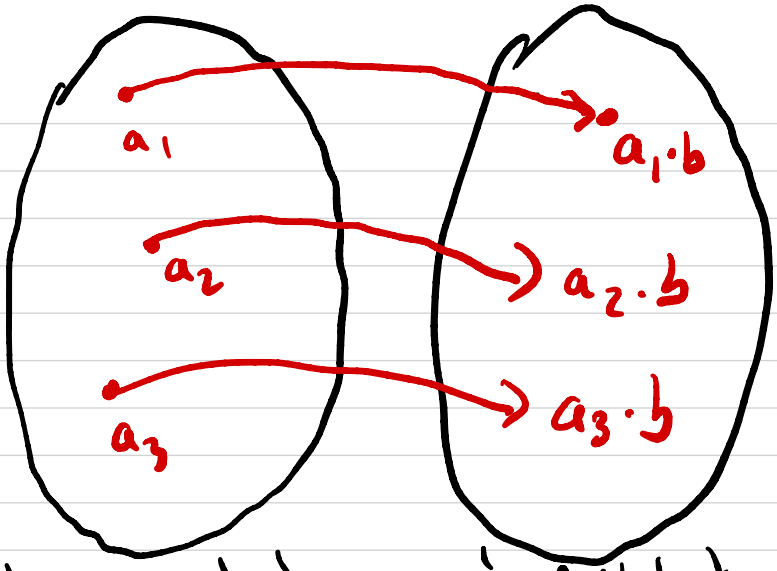
Let's pretend these don't exist for now..

Thm: Suppose N is composite and not Carmichael.
Then $\Pr[\text{Fermat Test}(N) = \text{composite}] \geq 1/2$

Pf: Not Carmichael $\Rightarrow$ coprime b s.t. $b^{N-1} \not\equiv 1 \pmod{N}$

Claim: Suppose a passes Fermat Test: $(a^{N-1} \equiv 1 \pmod{N})$
Then $a \cdot b \pmod{N}$ fails Fermat Test.

Pf:
$$\begin{aligned}(a \cdot b)^{N-1} &\equiv a^{N-1} \cdot b^{N-1} \pmod{N} \\ &\equiv b^{N-1} \pmod{N} \\ &\not\equiv 1 \pmod{N}\end{aligned}$$



a's pass test
$a^{N-1} \equiv 1 \pmod N$

a's fail test
$a^{N-1} \not\equiv 1 \pmod N$

Need to check

$a_i \cdot b \not\equiv a_j \cdot b \quad (i \neq j) \pmod N$

Fact: b is coprime
$\Rightarrow$ inverse $b^{-1} \pmod N$

$a_i b \not\equiv a_j \cdot b \pmod N$
$\cdot b^{-1} \qquad \cdot b^{-1}$

$a_i \not\equiv a_j \pmod N$

So $|\text{pass}| \leq |\text{fail}|$

□

$Pr[\text{Fermat Test}(N) = \text{composite}] \geq 1/2$ (if N is composite)

Repeat k times $\rightarrow$ detect composite w/prob $\geq 1 - \frac{1}{2^k}$

Can be very confident!

Runtime

Need to compute $a^{N-1} \pmod N$

Suppose $N-1 = 2^n$

$$a \cdot a \equiv a^2 \pmod N$$
$$a^2 \cdot a^2 \equiv a^4 \pmod N$$
$$a^4 \cdot a^4 \equiv a^8 \pmod N$$
$$\vdots$$
$$a^{2^{n-1}} \cdot a^{2^{n-1}} \equiv a^{2^n} \pmod N$$

} n steps

(Can generalize to arbitrary $N-1$)

Can add another check to detect Carmichael numbers
This gives Miller-Rabin primality test (1976)
Since then, we've only had randomized algs (no deterministic)

undergrads ← Until...

Agrawal-Kayal-Saxena Primality test (2002)
A deterministic alg for primality testing
in $O(n^{12})$ time (later $O(n^6)$ time)

Miller 1976 "derandomization"
Try the Miller-Rabin test for all $a \leq O(n^2)$
This will detect if N is prime or composite
(assuming generalized Riemann hypothesis)

Primality: efficient randomized algorithm first, later efficient deterministic alg

Other problems: Only know efficient randomized alg
(Polynomial identity testing)

Two possible worlds: 1. Every efficient randomized alg has deterministic counterpart

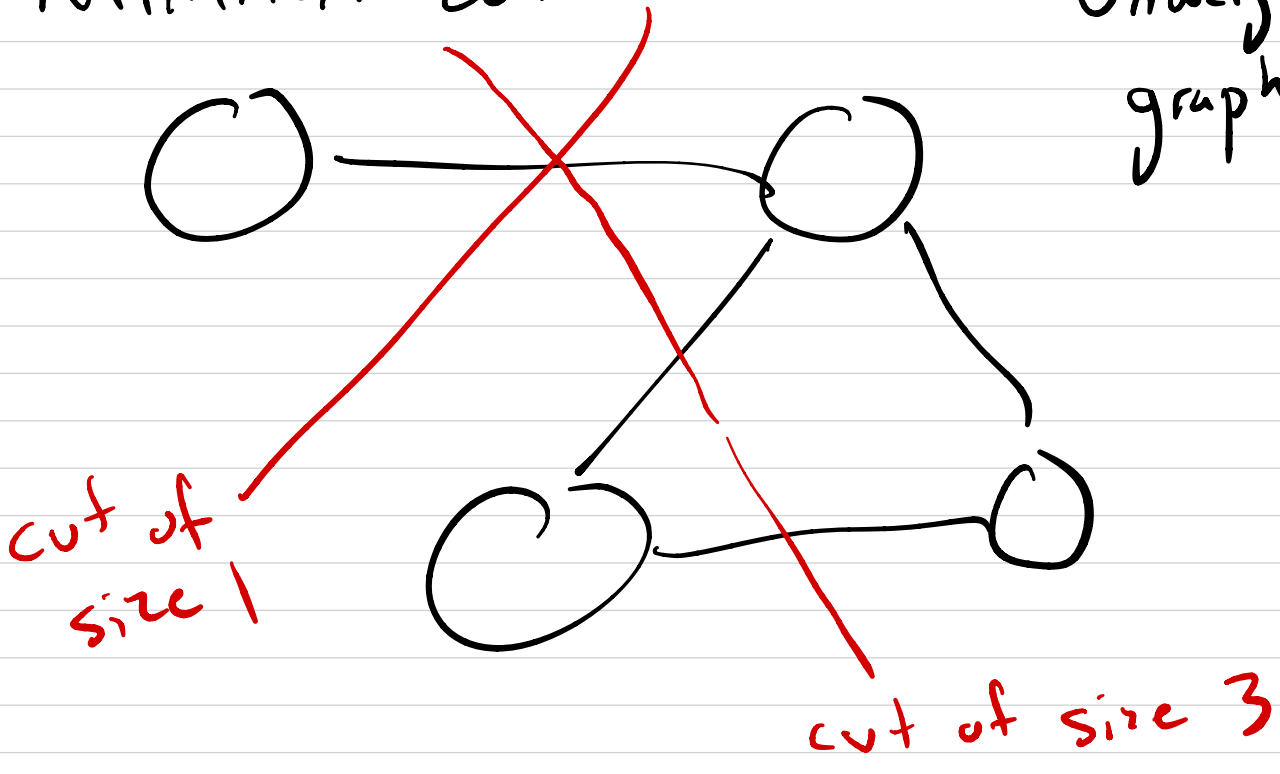
efficient deterministic → P

2. Some problems only have efficient randomized algs

v. BPP ← efficient randomized

Minimum cut

Unweight, undirected graphs $G = (V, E)$



Idea: Max flow/ min cut alg

Computes min s-t cut in time $O(n \cdot m)$

$|V|$ $|E|$

Which s, t to use?

set $s = 1$, try all $t = 2, \ldots, n$

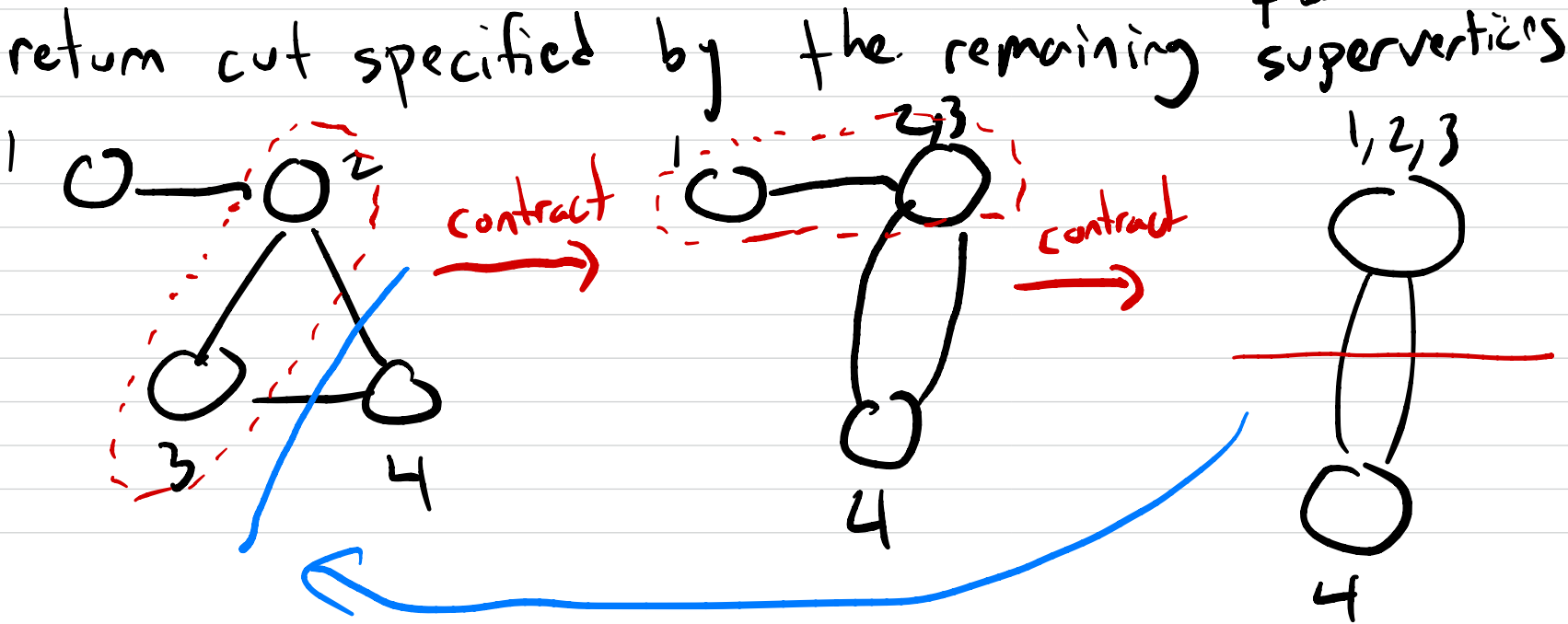
$O(n^2 \cdot m)$ time

Karger's algorithm(G)

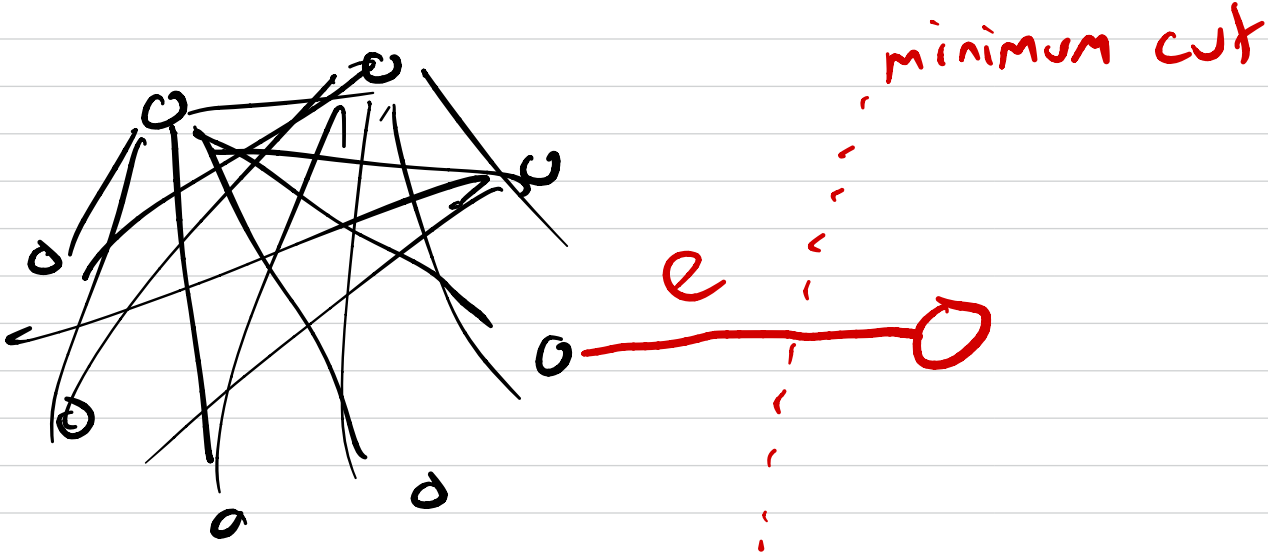
for $i = 1, \ldots, n-2$ $(n = |V|)$

1. pick a uniformly random edge e
2. contract e

return cut specified by the remaining two supervertices



Intuition



Karger's alg finds min cut if it never contracts e

But way more edges on left

Will usually pick there!

Thm: Let $C = (S, \bar{S})$ be a min cut of size k.

$$\Pr[\text{Karger's alg outputs } C] \geq \frac{1}{\binom{n}{2}} = \frac{2}{n(n-1)}$$

Pf: Let G_i for graph at beginning of i^{th} iteration ($G_1 = G$)

Fact 1: Min-Cut in $G_i \geq k$.

(any cut corresponds to cut in G)

Fact 2: # of vertices in $G_i = n - (i-1)$

$= n - i + 1$

Fact 3 : degree of each vertex in $G_i \geq k$

Fact 4 : # of edges in G_i

$$= \frac{1}{2} \sum_{v \in G_i} d_v \geq \frac{1}{2} \sum_{v \in G_i} k$$

$$= \frac{1}{2} k \cdot |G_i|$$

$$= \frac{1}{2} k \cdot (n - i + 1)$$

Suppose at G_i, haven't contracted edge in C yet.

$$\Pr[\text{don't contract an edge in } C]$$

$$= 1 - \Pr[\text{contract an edge in } C]$$

$$\geq 1 - \frac{k}{\frac{1}{2} \cdot k \cdot (n-i+1)} = 1 - \frac{2}{(n-i+1)}$$

$$\Pr[\text{never contract edge in } C]$$

$$\geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\left(1 - \frac{3}{n-2}\right) \cdots \left(1 - \frac{2}{3}\right)$$

$$= \left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right)\left(\frac{n-4}{n-2}\right) \cdots \left(\frac{2}{4}\right)\left(\frac{1}{3}\right)$$

$$= \frac{2}{n(n-1)}$$

$$\Pr[\text{succeeds}] \geq \frac{1}{\binom{n}{2}} \approx \frac{2}{n^2}$$

succeed w/ constant prob: repeat n^2 times (or slightly more)

$$\#\text{ of min cuts} \leq \binom{n}{2}$$

$$\Pr(\text{output min cut}) \geq \sum_{\text{min cut}} \frac{1}{\binom{n}{2}} =$$