Randomized Algorithms

Randomized algorithms

Algorithm which uses **random bits** to solve a problem

RandomInt(1, 100) ← true random number

Randomized algs can be (i) faster and (ii) more elegant

Las Vegas alg: output is always correct, but runtime depends on randomness

E.x. Quicksort $E[\text{runtime}] = O(n \log n)$

Randomized median $E[\text{runtime}] = O(\log n)$

("Mayb

Monte Carlo alg: Fixed runtime, but output is sometimes incorrect

For decision (Yes/No) problems:

Two-sided error: $\Pr[\text{Alg correct}] = \frac{1}{2} + \varepsilon, \ \varepsilon > 0$

One-sided error: If Yes instance: Alg **always** outputs YES
NO instance: $\Pr[\text{Alg outputs NO}] \geq p > 0$

Boosting correctness via repetition

One-sided error: YES instance: $\Pr[\text{Output YES}] = 1$

NO instance: $\Pr[\text{NO}] \geq p > 0$ — Would like .99 (or better)

Boosted alg A': Run algorithm A t times.

If any of t outcomes are NO, output NO.

Otherwise, output YES.

YES: A' always outputs YES

NO: $\Pr[\text{A' outputs YES}] = \Pr[\text{A outputs YES} \wedge \dots \wedge \text{A outputs YES}]$

$$= \prod_{i=1}^{t} \Pr[\text{i-th A outputs YES}]$$

$$\leq (1-p)^t \leq e^{-pt} = e^{-100} \quad \text{if } t = \frac{100}{p}.$$

Integer factorization

Given a 500 digit number N
find its prime factorization $N = p_1 p_2 \cdots p_k$

Algorithm: Check every integer $2 \leq x \leq \sqrt{N}$
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 $\leq 10^{100}$

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

Inefficient!

General number field sieve: Factor an n bit integer 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: 617 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"

Goal: Fermat Test is a Monte Carlo alg with one sided error (won't quite work :)

- Prime N: $\Pr[\text{"prime"}] = 1$
- Composite N: $\Pr[\text{"prime"}] \leq \frac{1}{2}$

Repeat t times: Prime N: $\Pr[t \text{ "prime" outputs}] = 1$

Composite N: $\leq \left(\frac{1}{2}\right)^t$

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: $a = 5$ — $5^{11} \not\equiv 1 \pmod{12}$???

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

Test only good if $a^{N-1} \not\equiv 1 \pmod{N}$ 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...

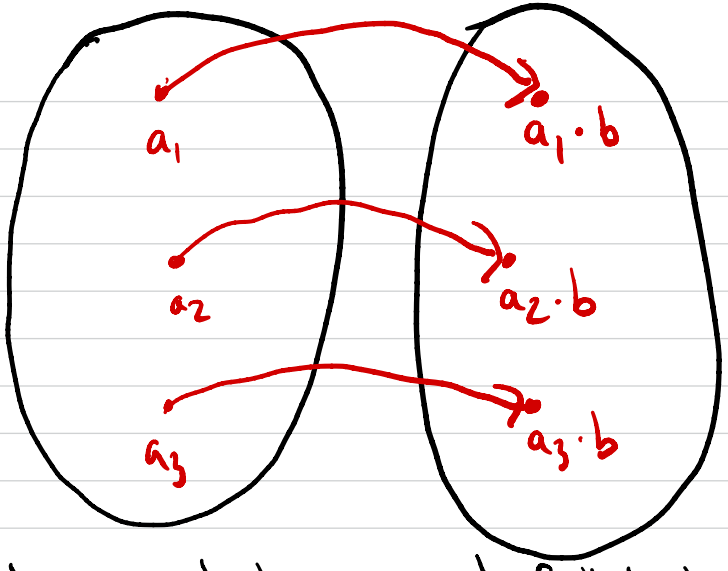
Thm: Suppose N is composite and not Carmichael.
Then $\Pr[\text{Fermat Test}(N) = \text{composite}] \geq 1/2$.

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:
$$(a \cdot b)^{N-1} \equiv a^{N-1} \cdot b^{N-1} \pmod{N}$$
$$\equiv b^{N-1} \pmod{N}$$
$$\not\equiv 1 \pmod{N}$$

□



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

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

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

Need to check
$a_i \cdot b \not\equiv a_j \cdot b \pmod N$, $i \neq j$

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

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

□

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 alg (no deterministic)

Until...

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

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

Primality: efficient randomized algorithm first, later efficient deterministic alg.

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

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

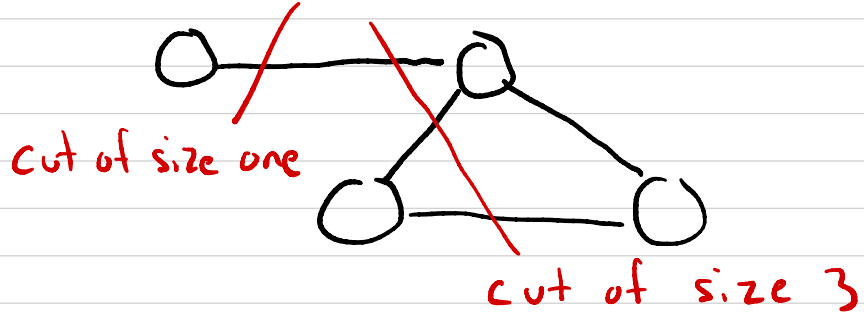
2. Some problems only have efficient randomized alg

efficient deterministic → P

v. BPP ← efficient randomized

Minimum Cut

Input: Unweighted, undirected graph $G=(V,E)$
Solution: Cut $(C,\bar{C})$ of minimum size



Idea: Use Max-Flow/Min-Cut alg

Computes min s-t cut. Which s, t to use?

Set $s=1$. Try all $t=2,\dots,n$.

Runtime: $n \cdot$ (time for max-flow alg) $\approx n \cdot O(m)$ $n=|V|$, $m=|E|$

100+ pages of math!

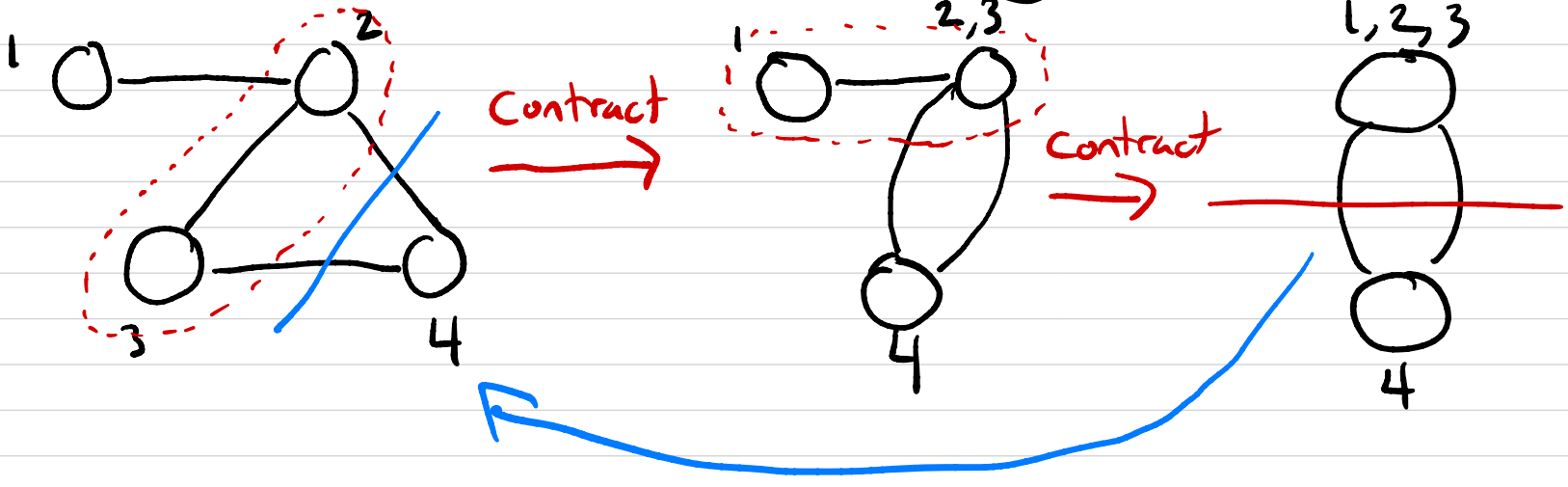
Today: elegant Monte Carlo alg, runs in time $O(n^2)$ for success prob 99%

Karger's algorithm (G)

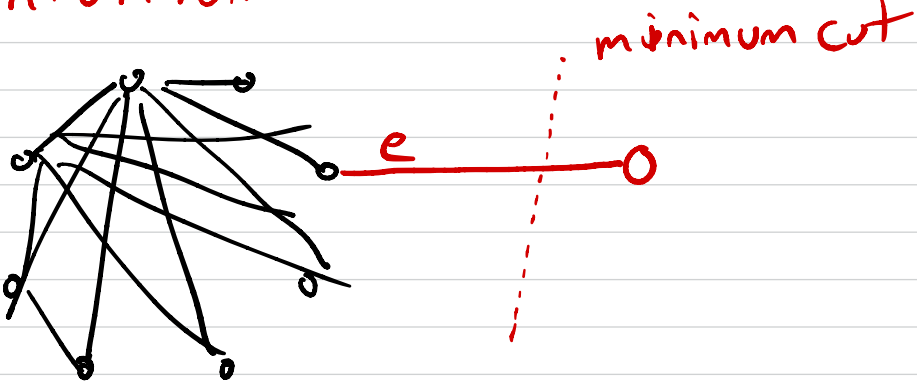
for $i = 1, \ldots, n-2$ $\quad (n = |V|)$

1. pick a uniformly random edge e_i
2. "contract" e_i

return cut specified by the remaining two supervertice



Intuition



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

But way more edges on left

Will usually pick these!

More generally: Many edges in graph, few in min cut

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: Def: G_i = graph at beginning of i-th iteration. ($G_1 = G$)

H_i = "happy" event the i-th contracted edge e_i doesn't cross cut $C = (S, \bar{S})$

$$\Pr[H_i \mid H_1, \dots, H_{i-1}] \geq \frac{n-i-1}{n-i+1}$$

$$\Pr[\text{Alg outputs } (C, \bar{C})]$$

$$= \Pr[H_1 \wedge H_2 \wedge \cdots \wedge H_{n-2}]$$

$$= \Pr[H_1] \cdot \Pr[H_2 \mid H_1] \cdot \Pr[H_3 \mid H_1, H_2] \cdots \Pr[H_{n-2} \mid H_1, \dots, H_{n-3}]$$

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

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

$$\Pr[H_i \mid H_1, \ldots, H_{i-1}] = 1 - \Pr[\text{Contract edge in } C \mid H_1, \ldots, H_{i-1}]$$

$$= 1 - \frac{\#\text{ edges across } C}{\#\text{ edges in } G_i}$$

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

close to 1 if $i=1$, $= \frac{1}{3}$ if $i = n-2$

Fact 1: Min-Cut $(G_i) \geq k$

(any cut in G_i corresponds to cut in G)

Fact 2: # 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 $G_i = \frac{1}{2}\sum_{v \in G_i} \deg(v)$

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

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

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

Succeed w/ constant prob: repeat n^2 times

(or slightly more)

Fact: # of min cut $\leq \binom{n}{2}$

$$\Pr[\text{output min cut}] \geq \sum_{\text{min cut}} \frac{1}{\binom{n}{2}} = \frac{(\#\text{ of min cuts})}{\binom{n}{2}}.$$