Today: simple examples to illustrate
- searching for a witness
- principle of deferred decisions
- fingerprinting
- probabilistic method
- derandomization
- method of conditional exps
- pairwise independence

Why randomized algs?
- often simplest or fastest
- fun!!!
Matrix Product Verification

Given $n \times n$ matrices $A, B, C$ over field $F$

Told $AB = C$

Goal: to verify this identity

Obvious method: matrix multiplication $O(n^{3.73})$

[Freivalds Alg] simple & elegant

One of first published uses of randomization in algs

Pick random vector $\mathbf{r} = (r_1, r_2, \ldots, r_n) \in \{0, 1\}^n$

Each $r_i$ indep, equally likely to be 0 or 1

Compute $A(B\mathbf{r}) = z$

If $C \mathbf{r} = z$

then output "yes, $AB = C$"

else output "no"

Running Time: $O(n^3)$

Errors: if $AB = C$ always output yes

if $AB \neq C$ may make an error
**Claim:** $Pr(\text{output an incorrect answer}) \leq \frac{1}{2}$

**Proof:** Define $D = AB - C$

Suppose $D \neq 0$

Then $\exists$ entry, say $(i,j)$ s.t. $d_{ij} \neq 0$

\[
Pr(D = 0) \leq Pr(\sum_k d_{ik} r_k = 0)
\]

\[
= Pr(d_{ij} r_j = -\sum_k d_{ik} r_k)
\]

\[
= Pr(r_j = \frac{-\sum_k d_{ik} r_k}{d_{ij}})
\]

Example of simple but powerful principle of deferred decisions

multiple r.v.'s - think of setting some of them first

and deferring setting rest until later

in analysis

Formally, use law of total probability; condition on values of vars set 1st
\[
\Pr \left( r_j = \frac{-\sum \text{dier}_k}{d_{ij}} \right) = \sum_{(x_{i_1}, x_{i_2}, x_{i_3}, \ldots, x_n) \in \{0,1\}^{n-1}} \Pr(r_j = \frac{-\sum \text{dier}_k}{d_{ij}} | (x_{i_1}, x_{i_2}, x_{i_3}, \ldots, x_n) = (x_{i_1}, x_{i_2}, x_{i_3}, \ldots, x_n)) \Pr(A) \leq \frac{1}{2}
\]

\[
\leq \sum \frac{1}{2} \Pr(A) = \frac{1}{2}
\]

If want to reduce probability of error, can do so at expense of

Small \( \uparrow \) in running time

1. Run alg \( k \) times
2. Output yes if get yes all \( k \) times

\[
\Pr(\text{error}) \leq \frac{1}{2^k}
\]

by independence of trials.
"Searching for witnesses"

- using randomization to check whether $P(r) = \text{True}$ for
  - Pick random $r$ from suitable set
  - If $P(r)$ true $\rightarrow \text{"No"}$
    Else $\rightarrow \text{"Yes"}$

- works well if density of witnesses that $P(r)$ false high enough

2 types of randomized algs

- Monte Carlo alg - halt in finite time but may output wrong answer
  - One-sided error
    - $\Pr(\text{output correct}) \geq 1 - e^{-2k^2}$
  - Two-sided error
    - $\Pr(\text{output correct}) \geq \frac{1}{2} + e^{-2k^2}$

Claim: If alg correct w.p. $\frac{1}{2} + e$ & we run $d$ times & output majority answer, probability answer correct $\geq 1 - e^{-2k^2}$

Proof:

$$\Pr(\text{majority wrong}) \leq \sum_{i=0}^{k} \frac{1}{i!} \left( \frac{1}{2} + e \right)^{i} \left( \frac{1}{2} - e \right)^{k-i}$$

$$\leq \sum_{i=0}^{k} \frac{1}{k!} \left( \frac{1}{2} + e \right)^{k} \frac{1}{2^{k-i}}$$

$$\leq \frac{1}{2} \left( \frac{1}{2} + e \right)^{k} \left( 1 - e^{k} \right) \frac{1}{2^{k-i}}$$

$$\leq \frac{1}{2} \left( \frac{1}{2} + e \right)^{k} \left( 1 - e^{k} \right) \frac{1}{2^{k-i}}$$

$$\leq \frac{1}{2} \left( \frac{1}{2} + e \right)^{k} \left( 1 - e^{k} \right) \frac{1}{2^{k-i}}$$

$$\leq \frac{1}{2} \left( \frac{1}{2} + e \right)^{k} \left( 1 - e^{k} \right) \frac{1}{2^{k-i}}$$

$$\leq e$$
Most useful approx: $1 - x \leq e^{-x}$

Los Vegas algorithms
always output correct answer. runtime is r.v.

Ex: randomized Quicksort

Big open question

Does randomness help in computation?

Can every poly time randomized alg be "de-randomized" with at most polynomial loss in efficiency?

Most useful approx: $1 - x \leq e^{-x}$

$e^x$ convex everywhere

$\Rightarrow$ tangent at $x=0$ lies below curve everywhere

$\Rightarrow$ $e^{-x} \geq 1 - x$
$e^x \geq 1 + x$
Fingerprinting

A & B each have large DB, separated by long distance
\[ a, b \] both n-bit strings

want to check if \( a = b \)?

Deterministically \( n \) bits of communication necessary

Next: randomized protocol that uses \( O(\log n) \) bits of communication

A picks prime \( p \in [2^n] \) u.a.r.

A sends \( (p, a \mod p) \) to B

B computes \( b \mod p \)

If \( a \mod p = b \mod p \), B sends back "yes," else "no"

Always gives right answer if \( a = b \).

may give wrong answer if \( a \neq b \)

Suppose \( a \neq b \)

\[ \Pr(a \mod p = b \mod p) = \Pr(a-b \text{ is multiple of } p) \]
\[
\frac{\text{\# distinct primes that divide } a-b}{\text{\# primes in } [2..x]} \leq n
\]

Each prime \( \geq 2 \)
Can't multiply \( > n \) together before get \( \geq n^n \)

\begin{align*}
\text{Prime \# Thm:} \\
\text{\# primes } \leq x \\
\geq \frac{x}{\ln x} \quad \forall x \geq 17
\end{align*}

Choosing \( x = c \cdot n \cdot \ln n \)

\[
\frac{n \cdot \ln x}{x} \leq \frac{1}{c} \cdot \frac{\ln x}{\ln n} = \frac{1}{c} + o(1)
\]

\[
\# \text{ bits transmitted} = 2 \cdot \log x = O(\log n)
\]

Example: \( n = 2^{32} \) \( \sim \) 1 gigabyte \( \quad x = 2^a \) (fingerprints are 64-bit words)

\[\Pr(\text{error}) < 10^{-a}\]
MaxCut \cite{MU04,CG98}:

- Simple randomized alg
- Illustration of probabilistic method

Use probabilistic argument to prove non-probabilistic mathematical thm:

**Defn:** cut in graph: partition of nodes into 2 sets $S$ and $\overline{S}$

An edge crosses cut if it has one endpoint in $S$ and one in $\overline{S}$

**Thm:**

In any graph $G=(V,E)$, $\exists$ cut $S$ s.t. at least $\frac{1}{2}$ edges cross cut.

Proof technique: show that if we pick a random cut, the expected number of edges that cross cut is $\geq \frac{1}{2}|E|$

Pick cut u.a.r. for $\forall v \in V$, flip fair coin \begin{align*} H &\rightarrow v \in S \\ T &\rightarrow v \in \overline{S} \end{align*}

Let $X_e = \begin{cases} 1 & \text{e crosses cut} \\ 0 & \text{o.w.} \end{cases}$

$$X = \sum_{e \in E} X_e \quad \text{# edges crossing cut}$$

$$E(X) = \frac{1}{2} |E|$$
\[ E(X) = E(\sum_{e \in E} x_e) = \sum_{e \in E} E(x_e) = \frac{1}{2} |E| \]

\[ \Rightarrow \text{sample space must contain at least one cut in which } \geq \frac{1}{2} \text{ edges cross cut. O.w., } E(X) \leq \frac{1}{2} |E| \]

Typical example of prob method:

- Not everybody can be below (or above) average

- Collection of objects \( \Pr( \exists \text{ object with property } P ) > 0 \)

\[ \Rightarrow \exists \text{ object in collection with property } P \]
Ideas: walk down tree, making good choice at each step

Observation:  
$$E(C(s, \overline{s}) | R_1=r_1, R_2=r_2, \ldots, R_n=r_n)$$
$$= \frac{1}{2} E(C(s, \overline{s}) | R_1=r_1, R_2=r_2, \ldots, R_n=r_n, R_{\overline{n}}=0)$$
$$+ \frac{1}{2} E(C(s, \overline{s}) | R_1=r_1, R_2=r_2, \ldots, R_n=r_n, R_{\overline{n}}=1)$$

$$\rightarrow S_i = \{ v_j \mid j < i, R_j = 0 \}$$
$$\overline{S}_i = \{ v_j \mid j < i, R_j = 1 \}$$
$$U_i = \{ v_{i+1}, \ldots, v_n \}$$
\[ E\left( \text{cut}(S, \overline{S}) \mid R, \text{cut } R, \text{cut } \overline{R} \right) = \left| \text{cut}(S, \overline{S}) \right| + \frac{1}{2} \left| \text{edges with at least one endpt in } U_i \right| \]

\[ \text{red edge, green edge, yellow edge} \]

\[ E\left( \text{cut}(S, \overline{S}) \mid R, \text{cut } R, \text{cut } \overline{R} \right) = \left| \text{cut}(S_i, \overline{S}_i) \right| + \frac{1}{2} \left| \text{edges with one endpt} \right| \]

\[ \text{in } U_i \]

\[ \Rightarrow \text{ suffice to set } r_{i+1} \text{ to maximize } \left| \text{cut}(S_i, \overline{S}_i) \right| \]

To maximize, pick biggest one \( \Rightarrow \) The greedy algorithm!

**Corollary** The greedy alg is guaranteed to find a cut of size \( \geq \frac{|E|}{2} \)
Method of Conditional Expectation

Consider randomized alg $A$ that uses $m$ random bits. $\Pr(A(x; R_1, R_m) \text{ good}) \geq \frac{2}{3}$ at least, say.

Sequences of coin tosses $\iff$ binary tree

"Good" randomized alg $\implies$ many paths good

$$\begin{align*}
R_1, \ldots, R_m \text{ seq of unif, indep random bits} \\
\text{Define } P(r_i, r_i) &= \text{fraction of continuations that are good.} \\
&= \Pr(A(x; R_1, \ldots, R_m) \text{ good} \mid R_i = r_i, \ldots, R_i = r_i) \\
&= \frac{1}{2} P(r_i, r_i, 0) + \frac{1}{2} P(r_i, r_i, 1)
\end{align*}$$
\[ \implies \exists r_{i+1} \in \{0,1\} \text{ s.t. } P(r_{i+1}, r_i) > P(r_i, r_i) \]

To find good path, just walk down tree & pick
\[ r_{i} \in \{0,1\} \text{ for } i = 1..m \text{ s.t. } P(r_{i+1}, r_i) > P(r_i, r_i) \]

At end:
\[ P(r_{i}, r_m) > P(r_{i}, r_{m-1}) > \ldots > P(r_i) \geq P(A(x; r_{i+1}, r_m)) \geq \frac{2}{3} \]

\[ \uparrow \]

0 or 1 \implies must be 1

Issue: to do this need to be able to deterministically compute \( P(r_{i}, r_i) \); may be infeasible

but sometimes works

worked for MAX CUT
Another approach - PRGs

Recall $E(\text{cut}(s\lambda)) = \sum_{\text{random partition}} \Pr(R_i \neq R_j) = \frac{1|E|}{2}$, $R_1, \ldots, R_n$ are random bits used by algorithm

Don't need full independence of $R_i$'s
Pairwise independence suffices! $\Pr(R_i \neq R_j) = \frac{1}{2}$

Observation: Suppose $B_1, B_2, \ldots, B_n$ are $k$ indep unbiased random bits
Then $\forall S \subseteq [n] \ (s \neq \emptyset)$, the $2^k-1$ random variables $R_S = \bigoplus_{i \in S} B_i$ are pairwise indep unbiased random bits
$\bigoplus = \text{XOR}$
**Proof: Unbiased**

Pairwise-indep. Consider $S \neq T \subseteq [k]$ nonempty

- Either $S \cap T$ disjoint $\checkmark$
  - $S \cap T \neq \emptyset$ \quad $\Pr(R_T = 1 | R_S) = \frac{1}{2}$

$\Rightarrow$ Given $\lceil \log(n^2) \rceil$ indep random bits $\Rightarrow$ $n$ pairwise indep random bits

Another deterministic MAXCUT Alg

- Any sequence of bits $b_1, b_2, \ldots, b_k$ where $k = \lceil \log(n^2) \rceil$
- Run randomized MAXCUT Alg using coin tosses $(r_S = \oplus b_i)_{S \neq T}$
- Choose largest cut obtained

**Correctness:** $E(\text{cut} - 1) = \frac{|E|}{2}$

$\Rightarrow$ $\exists b_1, \ldots, b_k$ s.t. cut has size $\geq \frac{|E|}{2}$

**Running time:**
A family $\mathcal{H}$ of functions $f(x): \mathcal{H} = \{h: [n] \rightarrow [m]\}$ is pairwise independent, when $h$ is chosen uniformly at random from $\mathcal{H}$, the following conditions hold:

1. $\forall x \in [n], \ h(x)$ is uniform on $[m]$
2. $\forall x_1, x_2 \in [n], \ h(x_1)$ and $h(x_2)$ are independent

Super important: hashing & well beyond

Often modeled hash funs as truly random

Infeasible to implement

Domain is often exponentially large,
can't even write it down

Wanted explicit family, efficiently computable.