Puzzles

3 people brought into a room

Hat placed on each person’s head: Red or Blue equally likely

Each person sees colors of other people’s hats, but not their own

Each person, without communication says: R, B or pass

All 3 shot unless they can agree on a strategy ahead of time

• at least one doesn’t pass
• everyone who doesn’t pass is right.

Strategy 1: Each person guesses \( \Rightarrow \Pr(\text{not all shot}) = \frac{1}{8} \)

Strategy 2: 2 pass, 1 guesses \( \Rightarrow \Pr(\text{not all shot}) = \frac{1}{5} \)

Is there a better strategy?
Randomized Algorithms & Probabilistic Analysis of Algorithms...

Model of computation: standard model (TM, RAM) with additional input consisting of stream of perfectly random bits.

\[ \Rightarrow \text{behavior can vary on fixed input, running time on particular input is a random variable.} \]

Example of difference:

- quicksort with randomly selected pivots vs QS where input is random

- again, performance of algorithm is a random variable

- also other random structures: random graphs, random boolean formulas, etc.

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 $O(n^{2.376})$.

Freivalds' Algorithm simple & elegant

One of the first published uses of randomization in algorithms.

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

Each $r_i$ is independent and equally likely to be 0 or 1.

Compute $A(\vec{r})z$.

If $Cr = z$ then output "yes, $AB = C$".

Else output "no".

Running Time:

Errors:
\textbf{Claim:} \( \Pr(\text{output an incorrect answer}) \leq \frac{1}{2} \)

\textbf{Proof:} Define \( D = AB - C \)

Suppose \( D \neq 0 \)

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

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

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

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

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
\[ \Pr \left( r_j = \frac{-\sum \text{dier} \cdot r_k}{d_{ij}} \right) = \sum_{(x_{i_1}, x_{i_2}, x_{i_3}, \ldots, x_j) \in \{0, 1\}^{m-1}} \Pr(r_j = \frac{-\sum \text{dier} \cdot r_k}{d_{ij}} \mid (r_{i_1}, r_{i_2}, r_{i_3}, \ldots, r_j) = (x_{i_1}, x_{i_2}, x_{i_3}, x_j)) \Pr(A) \leq \frac{1}{2} \]

\[ \leq \sum_{(x_{i_1}, x_{i_2}, x_{i_3}, \ldots, x_j) \in \{0, 1\}^{m-1}} \frac{1}{2} \Pr(A) = \frac{1}{2} \]

If we want to reduce the probability of error, we can do so at the expense of small \( \Delta \) 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.
Fingerprinting \[ [MR] \; 7.4 \] \[ [CG] \; 2.2.1 \]

A & B each have large DB, separated by long distance
\[ \downarrow \] \[ \downarrow \]
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 \( p \in [2^{n-1}] \) \( \text{u.a.r.} \) \[ \text{to be determined} \]

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 \( > n \)

**Prime \# Thm:**
\[ \text{\# primes } \leq x \approx \frac{x}{\log x} \]
\[ \geq 1.26 \frac{x}{\ln x} \quad \forall \quad x \geq 17 \]

\[ \frac{n \ln x}{1.26 \times x} \]

choosing \( x = \frac{c}{1.26} n \ln n \)

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

\[ = 2 \log x \approx O(\log n) \]

**Example:** \( n = 2^{23} \approx 1 \text{ MByte} \quad x = 2^{32} \quad (\text{fingerprints are 32 bit words}) \)

\[ \Pr(\text{error}) < 0.0035 \]
**MaxCut**  
[MU]6.2.1  [CG]1.4.1

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 $\bar{S}$

An edge crosses cut if it has one endpoint in $S$ & one in $\bar{S}$

**Thm:**

In any graph $G = (V, E)$, $\exists$ cut st. 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., $\forall v \in V$, flip fair coin $\begin{cases} H & \rightarrow v \in S \\ \uparrow & \rightarrow v \in \bar{S} \end{cases}$

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

$X = \sum_{e \in E} X_e$  

$\#$ edges crossing cut  

$E(X) = ?$
$$E(X) = E(\sum_{e \in E} X_e) = \sum_{e \in E} E(X_e) = \frac{1}{2} |E|$$

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

Typical example of prob method:

- Not everybody can be below (or above) average.

- Collection of objects $\Pr(\exists$ object with property $P) > 0$

  $\Rightarrow$ $\exists$ object in collection with property $P$