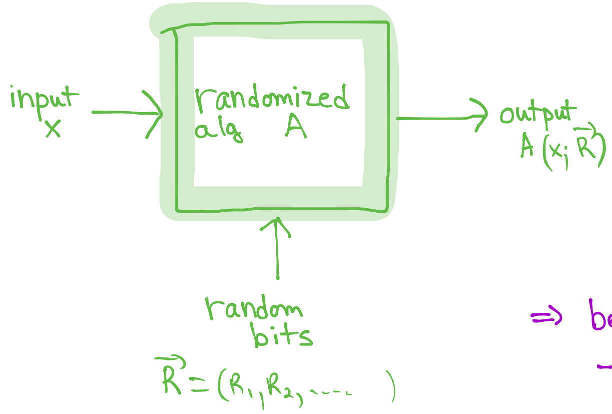
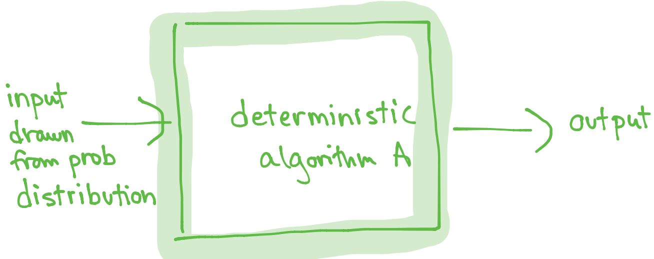
# Randomized Algorithms & Probabilistic Analysis of Algorithms....




Model of computation:

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

$\Rightarrow$ 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 $\pi$

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!!!

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

# Matrix-Product Verification

[MU] 1.3 [MR] 7.1

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

Told $AB = C$

Goal: to verify this identity

**Field $F$:**
set with 2 operations
addition, multiplication
has all the properties of $\mathbb{R}$ or rational #'s
e.g. commutativity, associativity
additive/mult. inverses
identity elts for $+, \cdot$

$\boxed{\text{Example: } GF(2)}$

addition XOR (addition mod 2)
multiplication AND

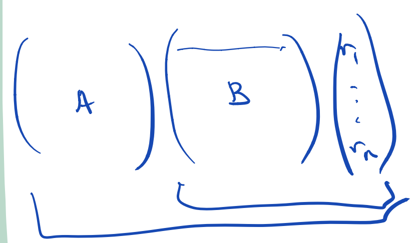
Obvious method: matrix multiplication

$O(n^{2.372})$

[Freivalds Alg] simple & elegant

one of first published uses of randomization in algs

> Pick random vector $\vec{r} = (r_1, r_2, \ldots, r_n) \in \{0,1\}^n$
>
> each $r_i$ indep, equally likely to be $\underset{\substack{\uparrow \\ \text{additive identity of field}}}{0}$ or $\underset{\substack{\uparrow \\ \text{multiplicative identity of field}}}{1}$
>
> compute $A(Br) = z$
>
> If $Cr = z$
>
> then output "yes, $AB = C$"
>
> else output "no"




Running Time: $O(n^2)$

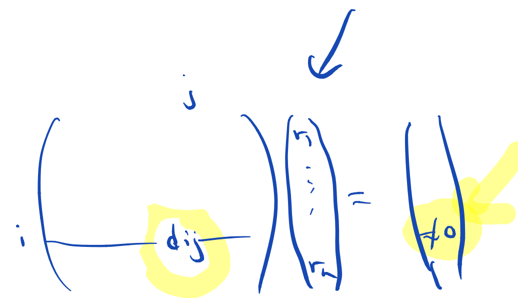
Errors: if $AB = C$ always output yes

if $AB \neq C$ may make an error $\Leftarrow$

$AB \neq C$

<u>Claim:</u> Pr(output an incorrect answer) $\leq \frac{1}{2}$

<u>Proof:</u> Define $D = AB - C$




Suppose $D \not\equiv 0$

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

$$\Pr(Dr = 0) \leq \Pr\Big(\sum_k d_{ik} r_k = 0\Big)$$

$$= \Pr\Big(d_{ij} r_j = -\sum_{k \neq j} d_{ik} r_k\Big)$$

$$= \Pr\left(r_j = \frac{-\sum_{k \neq j} d_{ik} r_k}{d_{ij}}\right)$$

Example of simple but powerful $\boxed{\text{principle of deferred decisions}}$

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

$$\Pr\left(r_j = \frac{-\sum_{k\neq j} d_{ik} r_k}{d_{ij}}\right)$$

$$= \sum_{\substack{(x_1,\ldots,x_{i-1},x_{i+1},\ldots x_n)\\ \in \{0,1\}^{n-1}}} \underbrace{\Pr\left(r_j = \frac{-\sum_{k\neq j} d_{ik} r_k}{d_{ij}} \,\middle|\, \underbrace{\begin{matrix}(r_1,\ldots,r_{i-1},r_{i+1},\ldots r_n)\\ = (x_1,\ldots,x_{i-1},x_{i+1},\ldots x_n)\end{matrix}}_{A}\right)}_{\leq \frac{1}{2}} \Pr(A)$$

$d_{ij} \rightarrow$ number

$$\leq \sum_{\substack{(x_1,\ldots,x_{i-1},x_{i+1},\ldots x_n)\\ \in \{0,1\}^{n-1}}} \frac{1}{2}\Pr(A) = \frac{1}{2}$$

---

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 $P(r) = \text{True}\ \forall r$
  - Pick random $r$ from suitable set
  - If $P(r)$ false $\longrightarrow$ "No"
    else $\longrightarrow$ "Yes"

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

---

## 2 types of randomized algs

**Monte Carlo algs** - halt in finite time but may output wrong answer

- One-sided error    $\uparrow$ confidence with repetition.

- Two-sided error    if true answer yes $\Rightarrow$ $Pr(\text{output yes}) \geq \frac{1}{2}+\varepsilon$
  no $\Rightarrow$ $Pr(\text{output no}) \geq \frac{1}{2}+\varepsilon$

  $\uparrow$ confidence with repetition & majority vote.

$Pr(\text{output correct answer}) \geq \frac{1}{2}+\varepsilon$

Claim: If $\overset{\text{decision}}{\text{alg}}$ correct w.p. $\geq \frac{1}{2}+\varepsilon$ & we run it $t$ times & output majority answer, probability answer correct $\geq 1-e^{-2t\varepsilon^2}$

Proof: $Pr(\text{majority wrong}) \leq \sum_{i=0}^{t/2} \binom{t}{i} \left(\frac{1}{2}+\varepsilon\right)^i \left(\frac{1}{2}-\varepsilon\right)^{t-i}$

$$\leq \sum_{i=0}^{t/2} \binom{t}{i} \underbrace{\left(\frac{1}{2}+\varepsilon\right)^{t/2} \left(\frac{1}{2}-\varepsilon\right)^{t/2}} = \left(\frac{1}{4}-\varepsilon^2\right)^{t/2} \underbrace{\sum_{i=0}^{t/2}\binom{t}{i}}_{\leq 2^t}$$

$$\leq \left(\frac{1}{4}-\varepsilon^2\right)^{t/2} 2^t = \left(\frac{1}{4}-\varepsilon^2\right)^{t/2} 4^{t/2} = (1-\underbrace{4\varepsilon^2}_{x})^{t/2} \leq e^{-4\varepsilon^2 t/2} = e^{-2\varepsilon^2 t}$$

---

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

**Las 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?

---

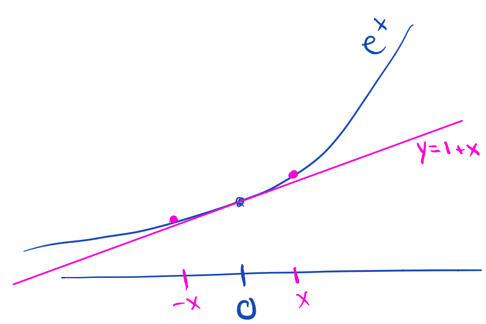
**Most useful approx: $1-x \le e^{-x}$**

$e^x$ convex everywhere

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

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

$e^x \geq 1+x$



# Fingerprinting

[MR] 7.4 [CG] 2.2.1

Alice $\qquad$ Bob

$a \in \{0,1\}^n$ $\qquad$ $b \in \{0,1\}^n$

minimize # bits they communicate.

A & B each have large DB, separated by long distance

$\downarrow$ $\quad$ $\downarrow$

a $\quad$ b $\quad$ both n-bit strings

$\boxed{\text{want to check if } a = b?}$

Deterministically n bits of communication necessary

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

> A picks prime $p \in [2..x]$ u.a.r.
>
> $x$ → to be determined
>
> A sends $(p, a \bmod p)$ to B
>
> B computes $b \bmod p$
>
> If $a \bmod p = b \bmod p$, B sends back "yes", else "no"

u.a.r. $\equiv$ uniformly at random.

Note: need random prime ...

Always gives right answer if $a = b$.

may give wrong answer if $a \neq b$

Suppose $a \neq b$

$\Pr(a \bmod p = b \bmod p) = \Pr(a - b \text{ is multiple of } p)$

p is one of prime factors of $a - b$

$$= \frac{\#\text{ distinct primes that divide } a-b}{\#\text{ primes in } [2..x]}$$

$$\leq \frac{\#\text{ distinct primes that divide } a-b}{\#\text{ primes in } [2..x]}$$

$\leq n$

Each prime $\geq 2$
can't multiply $> n$
together before get $> 2^n$

**Prime # Thm:**

$\#$ primes $\leq x$

$\geq \frac{x}{\ln x} \quad \forall x \geq 17$

$p \leq x$

$a \bmod p$

$$\leq \frac{n \ln x}{x}$$

$\frac{\cancel{x} \ln x}{c \cancel{x} \ln n}$

choosing $x = c\, n \ln n$

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

$$\boxed{\begin{array}{l}\#\text{ bits transmitted} \\ = 2 \log x = O(\lg n)\end{array}}$$

---

Example: $n = 2^{33} \approx 1$ gigabyte $\quad x = 2^{64}$ (fingerprints are 64 bit words)

$\Pr(\text{error}) < 10^{-9}$

Max Cut [MU]6.2.1 [CG]1.4.1

simple randomized alg

illustration of $\boxed{\text{probabilistic method}}$

use probabilistic argument to prove non-probabilistic mathematical thm

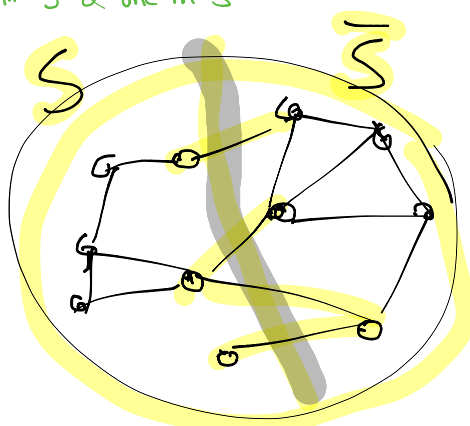
MAXCUT

Problem: Given $G=(V,E)$ find $S$ that maximizes $\mathrm{cut}(S,\bar{S})$ #edges

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

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

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




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

Pick cut u.a.r. $\forall v \in V$, flip fair coin $\begin{cases} H \to v \in S \\ T \to v \in \bar{S} \end{cases}$

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

$E(X_e) = Pr(e \text{ crosses cut}) = \frac{1}{2}$

$X = \sum_{e \in E} X_e$ # 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 $\quad Pr(\exists \text{ object with property } P) > 0$

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

## MAXCUT

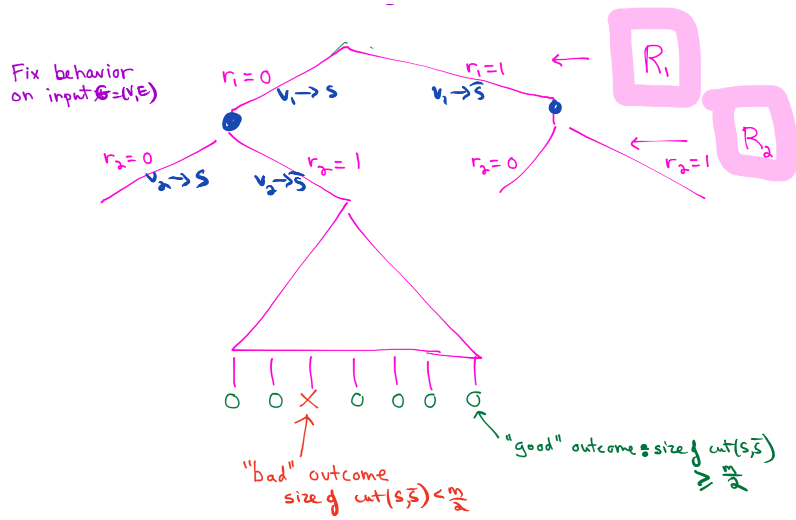
for $i := 1$ to $n$

Let $R_i \in_{\text{random}} \{0,1\}$

If $R_i = 0$, put $v_i$ in $S$, else put $v_i \in \bar{S}$

$G = (V, E)$ $\quad |V| = n \quad |E| = m$

$$\boxed{\underset{\substack{\uparrow \\ R_1, R_2, \ldots, R_n}}{E}\left[\text{Cut}(S, \bar{S})\right] \geq \frac{m}{2}}$$




Idea: walk down tree, making good choice at each step

Observation: $E\left(C(S,\bar{S}) \mid R_1 = r_1, R_2 = r_2, \ldots, R_i = r_i\right) \geq \frac{m}{2}$

$$= \frac{1}{2} E\left(C(S,\bar{S}) \mid R_1 = r_1, R_2 = r_2, \ldots, R_i = r_i, \underline{R_{i+1} = 0}\right)$$

$$+ \frac{1}{2} E\left(C(S,\bar{S}) \mid R_1 = r_1, R_2 = r_2, \ldots, R_i = r_i, \underline{R_{i+1} = 1}\right)$$

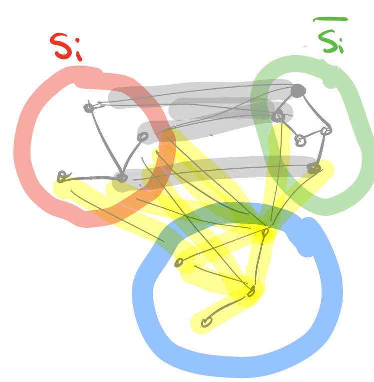
one of these is at least $\frac{m}{2}$

$v_1, \ldots, v_i \longrightarrow S, \bar{S}$

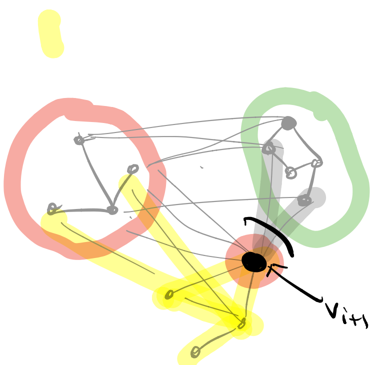
$\rightarrow S_i = \{v_j \mid j \leq i, R_j = 0\}$

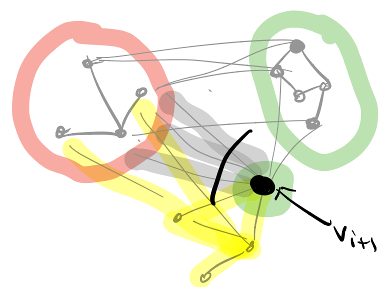
$\bar{S}_i = \{v_j \mid j \leq i, R_j = 1\}$

$U_i = \{v_{i+1}, \ldots, v_n\}$



$$E\left(\mathrm{cut}(S,\bar{S}) \mid R_1=r_1,\dots,R_i=r_i\right) = \underbrace{\left|\mathrm{cut}(S_i,\bar{S}_i)\right|}_{\text{red-green}} + \frac{1}{2}\underbrace{\left|\#\text{ edges with at least one endpt in } V_i\right|}_{\text{yellow edges}}$$




$$E\left(\mathrm{cut}(S,\bar{S}) \mid R_1=r_1, R_2=r_2,\dots, R_{i+1}=r_{i+1}\right) = \left|\mathrm{cut}(S_{i+1},\bar{S}_{i+1})\right| + \frac{1}{2}\underbrace{\left|\#\text{ edges with one endpt in } V_{i+1}\right|}_{\text{indep of } r_{i+1}}$$

$\Rightarrow$ suffices to set $r_{i+1}$ to maximize $\left|\mathrm{cut}(S_{i+1},\bar{S}_{i+1})\right|$

To maximize, pick bigger one $\Rightarrow$ The greedy algorithm!

__Corollary__ The greedy alg is guaranteed to find a cut of size $\geq \frac{|E|}{2}$

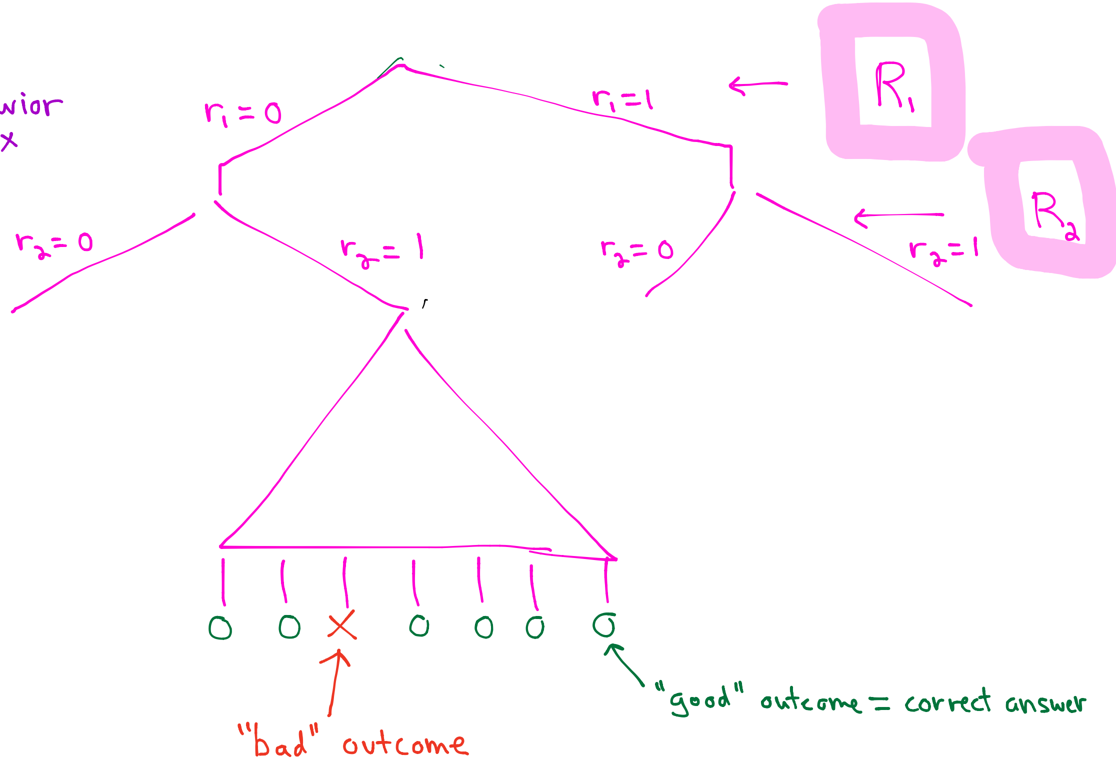
# Method of Conditional Expectation

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

Sequences of coin tosses $\Leftrightarrow$ binary tree

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

Fix behavior on input $x$




$R_1,..,R_m$ seq of unif, indep random bits

Define $P(r_1,..,r_i)$ = fraction of continuations that are good.

$$= \Pr(A(x; R_1,..,R_m) \text{ good} \mid R_1=r_1, \ldots R_i=r_i)$$

$$= \frac{1}{2}P(r_1,..,r_i,0) + \frac{1}{2}P(r_1,..,r_i,1)$$

$\Rightarrow \exists\, r_{i+1} \in \{0,1\}$ s.t. $\quad P(r_1,\dots,r_{i+1}) \geq P(r_1,\dots,r_i)$

To find good path, just walk down tree & pick

$r_i \in \{0,1\}$ for $i = 1..m$ s.t. $P(r_1,\dots,r_{i+1}) \geq P(r_1,\dots,r_i)$

At end:

$$\underset{\substack{\uparrow \\ 0 \text{ or } 1}}{P(r_1,\dots,r_m)} \geq P(r_1,\dots,r_{m-1}) \geq \dots \geq P(r_1) \geq P(\underset{\text{good}}{A(x; R_1,\dots,R_m)}) \geq \frac{2}{3}$$

0 or 1 $\Rightarrow$ must be 1

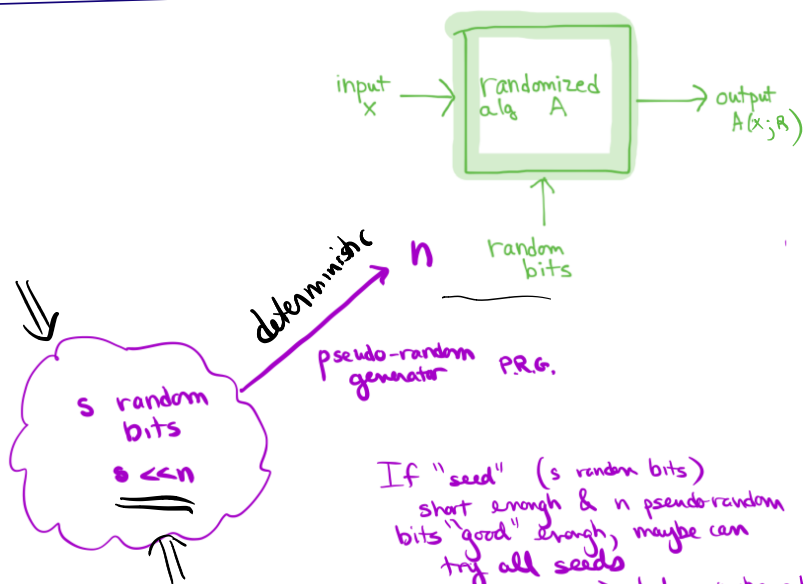
Issue: to do this need to be able to deterministically

compute $P(r_1,\dots,r_i)$ ; may be infeasible

but sometimes works

worked for MAX CUT

## Another approach – PRGs




If "seed" (s random bits) short enough & n pseudo-random bits "good" enough, maybe can try all seeds
$\Rightarrow$ deterministic alg

---

## Back to MAXCUT

Recall $\underset{\text{random partition}}{E}(|cut(S)|) = \sum_{(i,j)\in E} \underbrace{Pr(R_i \neq R_j)}_{\text{used this } = \frac{1}{2}} = \frac{|E|}{2}$

$R_1, \ldots, R_n$ are random bits used by algorithm

Don't need full independence of $R_i$'s

Pairwise independence suffices!

$Pr(R_i \neq R_j) = \frac{1}{2}$ ✓

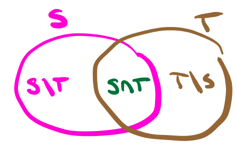
**Observation:** Suppose $B_1, B_2, \ldots, B_k$ are $k$ indep unbiased random bits

Then $\forall\ S \subset [k]\ (S \neq \phi)$, the $2^k - 1$ random variables $R_S = \bigoplus_{i\in S} B_i$ are pairwise indep unbiased random bits

$\oplus \equiv$ XOR

**Proof**: Unbiased ✓ . .

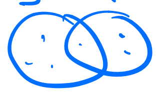
Pairwise-indep: Consider $S \neq T \subseteq [k]$ nonempty




either $S, T$ disjoint ✓

$S \subsetneq T$.

$S$ $T$



$\Pr(R_T = 1 \mid R_S) = \frac{1}{2}$

$\Rightarrow$ Given $\lceil \log_2(n+1) \rceil$ indep random bits $\Rightarrow$ $n$ pairwise indep random bits

Another deterministic MAXCUT Alg

$\forall$ sequences of bits $b_1, b_2, \dots, b_k$ where $k = \lceil \log_2(n+1) \rceil$

run randomized MAXCUT Alg using coin tosses $\left(r_S = \bigoplus_{i \in S} b_i\right)_{S \neq \emptyset}$

choose largest cut obtained

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

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

**Running time:**

A family $\mathcal{H}$ of fns: $\mathcal{H} = \{h : [n] \to [m]\}$

is pairwise indep if, when h is chosen

u.a.r. from $\mathcal{H}$ the following conditions hold:

(1) $\forall x \in [n]$, $h(x)$ is uniform on $[m]$

(2) $\forall x_1 \neq x_2 \in [n]$, $h(x_1)$ and $h(x_2)$ are independent

Super important: hashing & well beyond

Often model hash fns as truly random

infeasible to implement

domain often exponentially large, can't even write it down

Want explicit family efficiently computable.