Lecture 7

Max Cut:

- randomized algorithm
- derandomizing via pairwise independence

Generating pairwise independent bits

Pairwise Independence & Derandomization

Let's start with a simple algorithm for MaxCut:

Max Cut:

given: $G = (V, E)$

output: partition V into S, T $\}$ NP hard

to maximize $\underbrace{|\{(u,v) \mid u \in S, v \in T\}|}_{\text{size of S-T cut}}$

Randomized Algorithm:

Flip coins $r_1 \cdots r_n$

Put node i on side r_i to get S, T

if $r_i = 0$ add i to S
else $r_i = 1$ add i to T

Analysis:

u,v on opposite sides of cut

$$\text{Let } \mathbb{1}_{u,v} = \begin{cases} 1 & \text{if } r_u \neq r_v \\ 0 & \text{o.w.} \end{cases}$$

$$\text{Cut size} = \sum_{(u,v)\in E} \mathbb{1}_{u,v}$$

$$E[\text{cut size}] = E\Big[\sum_{(u,v)\in E} \mathbb{1}_{u,v}\Big]$$

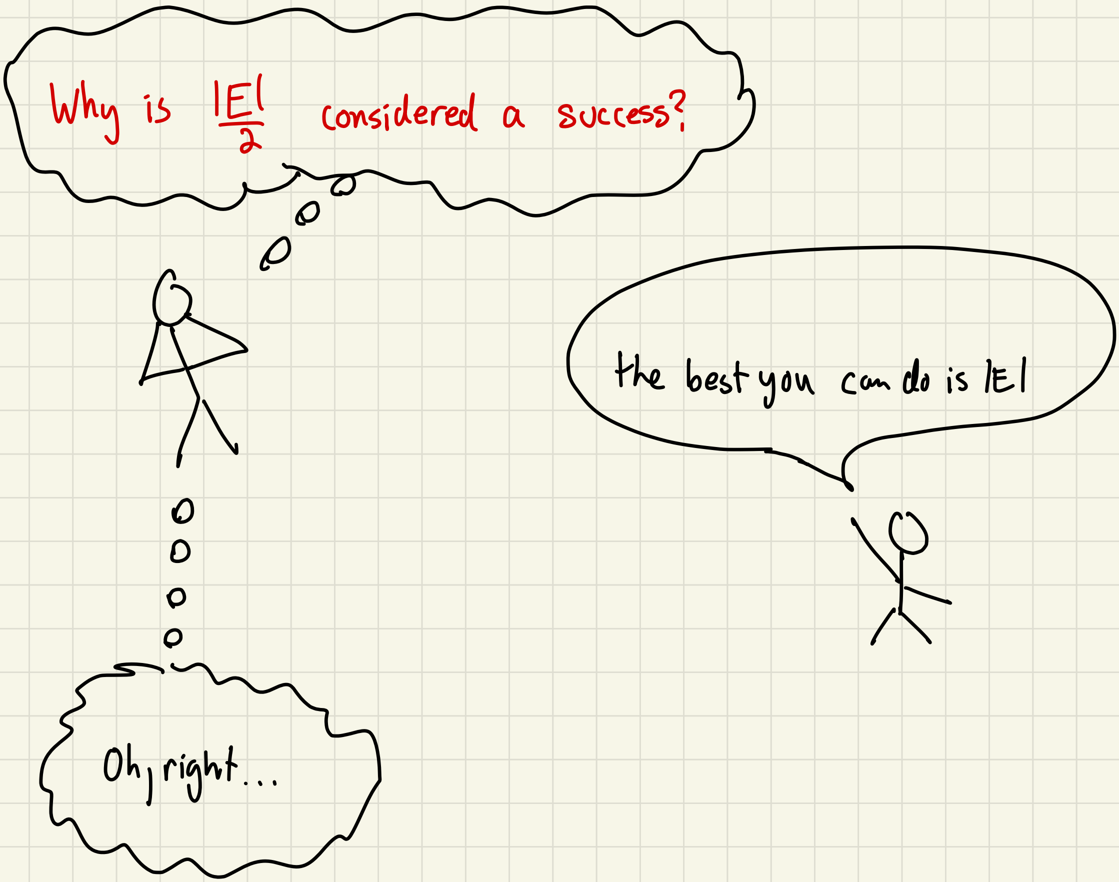
$$= \sum_{(u,v)\in E} E[\mathbb{1}_{u,v}] = \sum_{(u,v)\in E} \Pr[\mathbb{1}_{u,v} = 1]$$

$$= \sum_{(u,v)\in E} \Pr\big[(r_u = 1 \ \& \ r_v = 0) \text{ or } (r_u = 0 \ \& \ r_v = 1)\big]$$

$$= |E| \cdot \Big[\tfrac{1}{4} + \tfrac{1}{4}\Big] = \frac{|E|}{2}$$

So expect $\frac{1}{2}$ the edges to cross cut!

$$\text{Note: } \underbrace{E[\text{cut size}]}_{\text{average cut size produced by algorithm}} = \frac{|E|}{2} \Rightarrow \exists \underbrace{\text{cut}}_{\text{must be one that is at least as big as the average}} \text{of size } \frac{|E|}{2}$$



Gives multiplicative approximation to within a factor of 2

Last time:

Derandomization via Enumeration

Given: probabilistic algorithm $\mathcal{A}$ + input x

Algorithm:

Run $\mathcal{A}$ on every possible random string of length $r(n)$

Output majority answer

Randomized Max Cut Algorithm:

Flip coins $r_1 \cdots r_n$

Put node i on side r_i to get S, T

Derandomization: first attempt

use "derandomization via enumeration"

Run $\mathcal{A}$ on every possible random string of length $r(n)$

Output majority answer

here $r(n) = n$, so need 2^n runs of $\mathcal{A}$ ☹

<u>Hope</u>: reduce $r(n)$?

still use "derandomization via enumeration"

find subset $S \subseteq \{0,1\}^{r(n)}$ of random strings that "works"

& only enumerate over S

Pairwise Independent Random Variables

Given domain T s.t. $|T| = t$

Pick n values $X_1 \cdots X_n$ s.t. each $X_i \in T$

def $X_1 \cdots X_n$ independent if $\forall\, b_1 \cdots b_n \in T^n$

$$\Pr[X_1 \cdots X_n = b_1 \cdots b_n] = \frac{1}{t^n}$$

pairwise independent if $\forall\, i \neq j$ $\quad b_i, b_j \in T^2$

$$\Pr[X_i X_j = b_i b_j] = \frac{1}{t^2}$$

k-wise independent if $\forall$ distinct $i_1 \cdots i_k$

$$b_1 \cdots b_k \in T^k$$

$$\Pr[X_{i_1} \cdots X_{i_k} = b_1 \cdots b_k] = \frac{1}{t^k}$$

Example:

total independence $\{0,1\}^3$

bits	r_1	r_2	r_3
	0	0	0
	0	0	1
	0	1	0
	0	1	1
	1	0	0
	1	0	1
	1	1	0
	1	1	1

if pick uniform setting "row" of r_i's the r_i's are Totally uniform

pairwise independence $S \subseteq \{0,1\}^3$

$|S| = 4 < 8$

bits	r_1	r_2	r_3
	0	0	0
	0	1	1
	1	0	1
	1	1	0

if pick uniform setting "row" of r_i's, the r_i's are pairwise independent

pick from these 4 rows using only 2 random bits

S is not independent:

$$\Pr_{row}[r_1 r_2 r_3 = 110] = \frac{1}{8} \neq \Pr_{row}[r_1 r_2 r_3 = 110] = \frac{1}{4}$$

S is pairwise independent:

$$\Pr_{row}[r_1 r_3 = 11] = \Pr_{row}[r_1 r_3 = 10] = \frac{1}{4} = \Pr_{row}[r_1 r_3 = 11] = \Pr_{row}[r_1 r_3 = 10] = \frac{1}{4}$$

works for any $r_i r_j = b_1 b_2$ for $i \neq j \in \{1,2,3\}$ & $b_1 b_2 \in \{0,1\}^2$

Main points:

- picking from fewer rows requires fewer random bits } faster to derandomize

  so need to do fewer calls to $\mathcal{A}$ when doing "derandomization via enumeration"

- picking from smaller "subset" of rows has some of same properties as picking from <u>all</u> rows } good enough

only need pairwise independence in max cut algorithm

$\Rightarrow$ analysis of expectation still same if only use pair wise indep random bits

let's check this:

$$E[\text{cut size}] = \sum_{(u,v)\in E} E[1_{u,v}] = \sum_{(u,v)\in E} Pr[1_{u,v}=1]$$

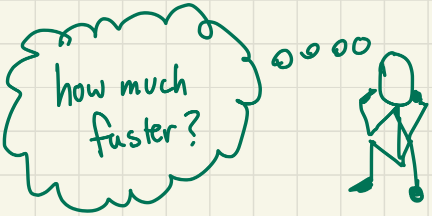
$$= \sum_{(u,v)\in E} Pr[(r_u=1 \,\&\, r_v=0) \text{ or } (r_u=0 \,\&\, r_v=1)]$$

pairwise indep enough for

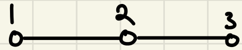
$Pr[r_u=1 \,\&\, r_v=0] = Pr[r_u=0 \,\&\, r_v=1] = 1/4$

$$= |E| \cdot \left[\tfrac{1}{4}+\tfrac{1}{4}\right] = \frac{|E|}{2}$$

$\Rightarrow$ can do "derandomization via enumeration" on pairwise independent bits



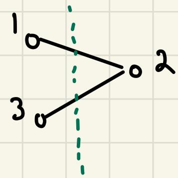
Example using our 3 pairwise indep bits:



$|E| = 2$

so looking for Max cut of size $\frac{|E|}{2} = 1$

Max cut:

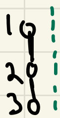


max value = 2

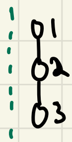
All cuts:

value = 0:

$r_1 = r_2 = r_3 = 0$

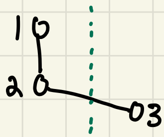


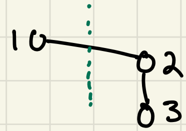
$r_1 = r_2 = r_3 = 1$



value = 1:

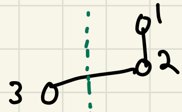
$r_1 = r_2 = 0$
$r_3 = 1$

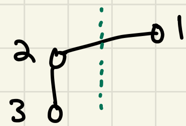




$r_1 = 0$
$r_2 = r_3 = 1$

$r_1 = r_2 = 1$
$r_3 = 0$

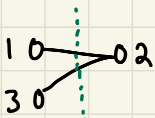


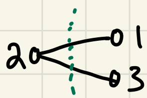


$r_1 = 1$
$r_2 = r_3 = 0$

value = 2:

$r_1 = r_3 = 0$
$r_2 = 1$



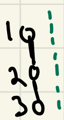


$r_1 = r_3 = 1$
$r_2 = 0$

Average value = 1
$\Rightarrow \exists$ cut of value ≥ 1

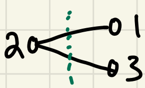
Pairwise indep cuts:

$r_1 = r_2 = r_3 = 0$



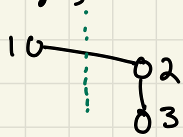
value = 0

$r_1 = r_3 = 1$
$r_2 = 0$



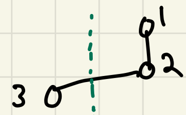
value = 2

$r_1 = 0$
$r_2 = r_3 = 1$



value = 1

$r_1 = r_2 = 1$
$r_3 = 0$



value = 1

Average value = 1 $\Rightarrow \exists$ cut of value ≥ 1

Another picture:

$b_1 \dots b_m \longrightarrow$ "randomness generator" $\longrightarrow r_1 r_2 r_3 \; \dots \; r_n$

totally independent

enumerate all 2^m choices of b_i's

picks random row in S using b_i's

output n pairwise independent random bits

in our 3-bit example:

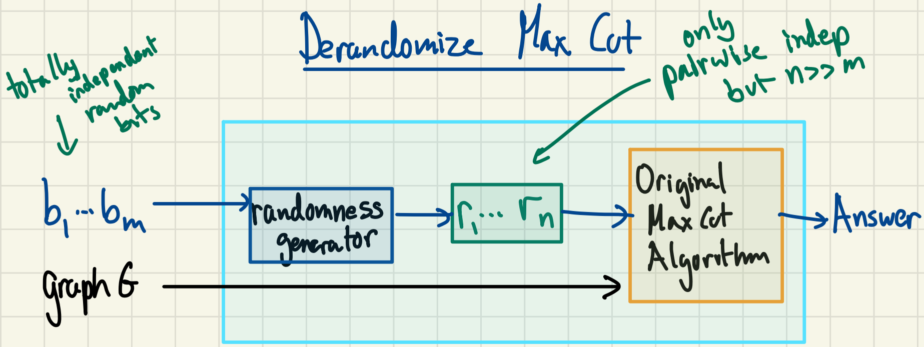
pick $b_1 b_2$	output r_1	r_2	r_3
00	0	0	0
01	0	1	1
10	1	0	1
11	1	1	0

Table lookup: e.g. if pick $b_1 b_2 = 01$ output $r_1 r_2 r_3 = 011$

Question: what do you do when you need > 3 p.i. random bits?

(to be answered soon)

Derandomize Max Cut



New Max Cut algorithm MC'
using $m << n$ random bits

} do derandomization via enumeration on this in

$O(2^m)$ calls to MC'

one for each setting of $b_1 \cdots b_m$

In words:

1. Construct MC': given m random bits $b_1 \cdots b_m$
graph G

procedure:

generates $r_1 \cdots r_n$ (pairwise indep) from $b_1 \cdots b_m$

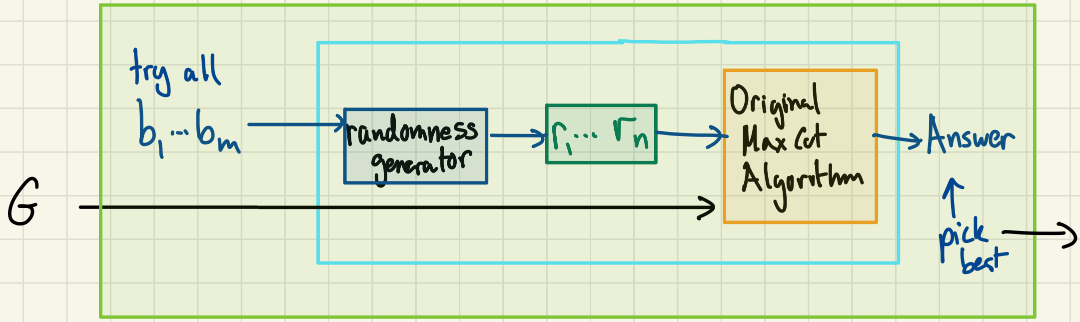
use r_i's to run Max Cut & evaluate cutsize

2. Derandomize MC':

for all choices of $b_1 \cdots b_m$

run MC' on $b_1 \cdots b_m$ & G & evaluate cut size

pick best cutsize



derandomized MC'

runtime: $2^m \times$ (time for generator + time for MC)

will show $m = O(\log n)$ + time for generator poly(n)

so total time is poly(n)

Comments:

if derandomize MC by enumeration, you end up trying all cuts $\Rightarrow$ get OPT

here, we are trying very few cuts so no guarantee of getting OPT. just $\frac{OPT}{2}$

Generating Pairwise Independent Random Variables:

1. Bits

- choose k truly random bits $b_1 .. b_k$

$$\forall S \subseteq [k] \quad \text{s.t.} \quad S \neq \emptyset$$

$$\text{set} \quad C_S \equiv \bigoplus_{i \in S} b_i$$

- output all C_S

k truly random bits $\rightarrow$ $2^k - 1$ p.i. bits

$\log n$ $\qquad\qquad$ $n-1$

proof of correctness:

upcoming homework

2. Integers in $[0, \ldots, q-1]$ q prime

1st idea: if $q < 2^{\ell}$ can be represented via ℓ bits

repeat "bits" construction independently for each position q_i in $1..\ell$

Uses $O(\log n \cdot \log q)$ bits of true randomness

$\uparrow$ bits construction

$\uparrow$ # repetitions

Slightly better idea: $O(\log q)$ bits of randomness

- Pick $a, b \in \mathbb{Z}_q$
- $r_i \leftarrow a \cdot i + b \mod q$ $\forall i \in \{0, \ldots, q-1\}$
- output $r_1 \cdots r_q$

Useful to think of construction as

input/output description of a

function:

$$h_{a,b}: \{0, \ldots, q-1\} \to \mathbb{Z}_q$$

& family of fctns $\mathcal{H} = \{ h_{a,b} \mid a,b \in \mathbb{Z}_q \}$

Family of fctns $\mathcal{H} = \{h_1, h_2 \ldots\}$

for $h_i : [N] \to [M]$ is

"pairwise independent" if

when $h \in_u \mathcal{H}$

any locn x is mapped uniformly

(1) $\forall x \in [N]$, $h(x) \in_u [M]$

any pair of locns $x_1 \neq x_2$ mapped uniformly & independently

(2) $\forall x_1 \neq x_2 \in [N]$, $(h_1(x), h_2(x)) \in_u [M]^2$

equivalently:

$$\forall\, x_1 \neq x_2 \in [N]$$
$$\forall\, y_1, y_2 \in [M]$$
$$\Pr_{h \in \mathcal{H}}\left[h(x_1) = y_1 \;\&\; h(x_2) = y_2 \right] = \frac{1}{M^2}$$

Comments:

- no single fctn is p.i. on its own - need to pick a random fctn from a collection of fctns.
- given h & $x \in [N]$, $h(x)$ should be computable in time $\mathrm{poly}(N, \log M)$ — don't have to compute all $h(x)$ at once
- also called "strongly 2-universal hash fctns"

Why is our example p.i.?

Our family:

$$\mathcal{H} = \{ h_{a,b} \mid \mathbb{Z}_q \to \mathbb{Z}_q \}$$ recall q is prime

$$h_{a,b}(x) = ax + b \bmod q$$

proof of p.i.:

$\forall\, x \neq w,\ c, d$

$$\Pr_{a,b}[\underbrace{ax+b}_{h_{a,b}(x)} = c \ \wedge\ \underbrace{aw+b}_{h_{a,b}(w)} = d] = \frac{1}{q^2}$$

$$\underbrace{\begin{pmatrix} x & 1 \\ w & 1 \end{pmatrix}}_{w \neq x} \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} c \\ d \end{pmatrix}$$

so $\det \begin{pmatrix} x & 1 \\ w & 1 \end{pmatrix} \neq 0$ $\Rightarrow$ unique soln

so nonsingular

how many truly random bits?

pick a, b uniformly, needs $2 \log q$ random bits

More comments:

Can construct for all finite fields, even when domain & range have different sizes

Original motivation: hashing

Choose hash fctns from p.i. family instead of totally random fctns.

why?

how to store random fctn?

need $|\text{domain}| \cdot \log|\text{range}|$ bits to write down input/output table

how to store p.i. hash fctn?

write down a & b