Lecture 7

Definition of pairwise independence

Max Cut:

- randomized algorithm (covered last time)
- derandomizing via pairwise independence

Generating pairwise independent bits

(Last time)

Max Cut:

given: $G = (V, E)$

output: partition V into S, T } NP hard

to maximize $|\{(u,v) \mid u \in S, v \in T\}|$

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

(Last time)

Analysis:

u, v on opposite sides of cut

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

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

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

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

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

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

(last time)

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 A on every possible random string of length $r(n)$
>
> Output majority answer

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

Hope: 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 $U = \{0,1\}^3$

distribution A = Pick random row, output $r_1 r_2 r_3$

row	bits r_1	r_2	r_3
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
6	1	0	1
7	1	1	0
8	1	1	1

picking random row needs 3 random bits

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

$|S| = 4 < 8$

distribution B = Pick random row, output $r_1 r_2 r_3$

row	bits r_1	r_2	r_3
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

pick from these 4 rows needs only 2 random bits

Bits of A independent but B is not independent:

e.g. $\Pr_A[r_1 r_2 r_3 = 110] = \frac{1}{8} \neq \Pr_B[r_1 r_2 r_3 = 110] = \frac{1}{4}$

B is pairwise independent:

e.g. $\Pr_A[r_1 r_3 = 11] = \Pr_A[r_1 r_3 = 10] = \frac{1}{4} = \Pr_B[r_1 r_3 = 11] = \Pr_B[r_1 r_3 = 10] = \frac{1}{4}$

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 A when doing "derandomization via enumeration"

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

only need pairwise independence in max cut algorithm

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

let's check this:

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

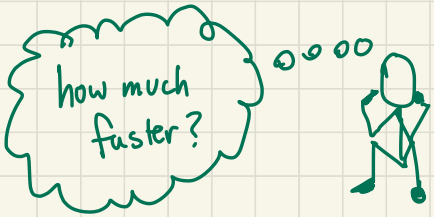
$$= \sum_{(u,v)\in E} \Pr[(r_u=1 \text{ and } r_v=0) \text{ or } (r_u=0 \text{ and } r_v=1)]$$

pairwise indep enough for

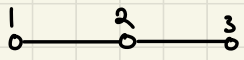
$\Pr[r_u=1 \text{ and } r_v=0] = \Pr[r_u=0 \text{ and } r_v=1]$
$= 1/4$

$$\rightarrow = |E| \cdot \left[\tfrac{1}{4} + \tfrac{1}{4}\right] = \frac{|E|}{2} \Rightarrow \exists \text{ cut of size } \geq |E|/2$$

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



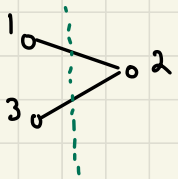
Example using our 3 pairwise indep bits:



$|E| = 2$

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

Max cut:

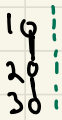


max value = 2

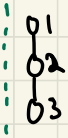
All cuts:

value = 0:

$r_1 = r_2 = r_3 = 0$

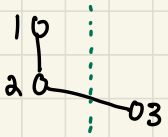


$r_1 = r_2 = r_3 = 1$

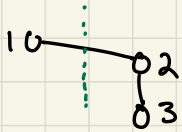


value = 1:

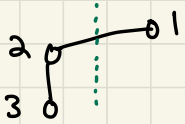
$r_1 = r_2 = 0$, $r_3 = 1$



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

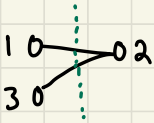


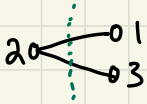
$r_1 = 0$, $r_2 = r_3 = 1$



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

value = 2:





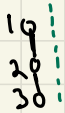
$r_1 = r_3 = 0$, $r_2 = 1$

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

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

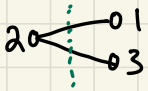
Pairwise indep cuts:

$r_1 = r_2 = r_3 = 0$



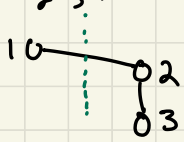
value = 0

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



value = 2

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



value = 1

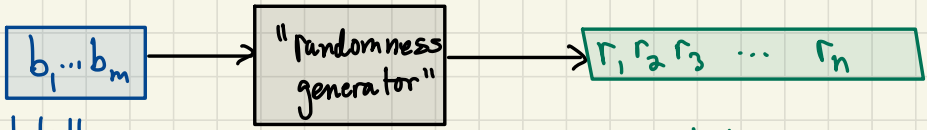
$r_1 = r_2 = 1$, $r_3 = 0$

value = 1

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

(its possible that pairwise indep misses biggest cut, but has same ave value)

Another picture:



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:

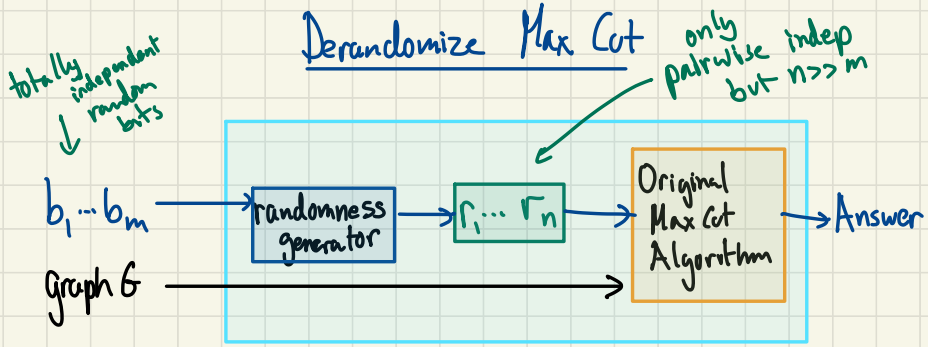
pick $b_1 b_2$	output r_1	r_2	r_3
$00 \to 1$	0	0	0
$01 \to 2$	0	1	1
$10 \to 3$	1	0	1
$11 \to 4$	1	1	0

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 $b_1 \cdots b_m$

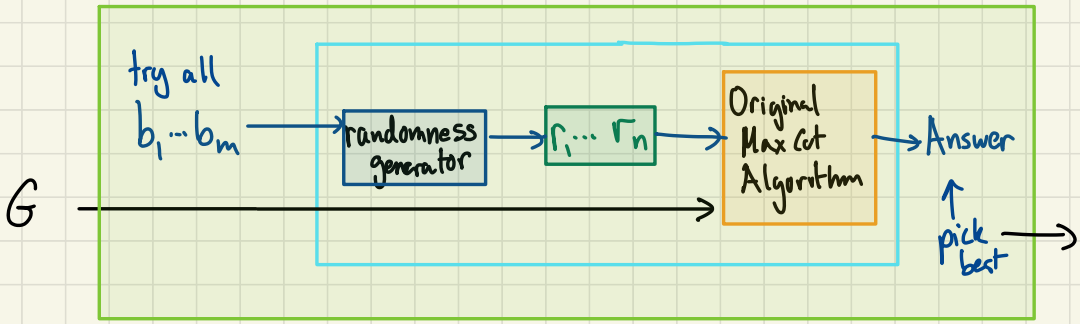
   use r_i's to run Max Cut & evaluate cutsize

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 x (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

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

proof of correctness:
upcoming homework

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

- $\log n$: bits construction
- $\log q$: # repetitions

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

- Pick $a, b \in \mathbb{Z}_q$
- $r_x \leftarrow a \cdot x + b \mod q \quad \forall x \in \{0, \dots, q-1\}$
- output $r_1 \cdots r_q$

Useful to think of construction as input/output description of a function:

This is also $\mathbb{Z}_q$

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

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

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

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

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 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

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

note: $|\mathcal{H}| = q^2$

proof of p.i.:

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

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

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

$w \neq x$

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

exactly one $h_{a,b}$ maps x to c & w to d

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