Density $\mu$ on space $\Omega$
Density $\mu$ on space $\Omega$

- **Sampling:** $\mathbb{P}[\text{output}] \propto \mu(\text{output})$
Density $\mu$ on space $\Omega$

- **Sampling:** $\mathbb{P}[\text{output}] \propto \mu(\text{output})$
- **Counting:** compute $\sum_x \mu(x)$
Density $\mu$ on space $\Omega$

- **Sampling:** $\mathbb{P}[\text{output}] \propto \mu(\text{output})$
- **Counting:** compute $\sum_{\chi} \mu(\chi)$
- **#P:** #accepting paths in TM
Density $\mu$ on space $\Omega$

- **Sampling:** $P[\text{output}] \propto \mu(\text{output})$
- **Counting:** compute $\sum_x \mu(x)$
- **#P:** #accepting paths in TM

- **#P-complete:**
  - Natural counting variants of known NP-complete problems.
  - Natural counting variants of some P problems too!

**Additional content:**

- Approx counting $(1 + \epsilon)$-approx in $\text{poly}(n, 1/\epsilon)$
  - $\text{FP}_R$ randomized $\text{AS/FP}_T$ deterministic $\text{AS}$

- Approx sampling $\delta$-approx in $d_{TV}$ in $\text{poly}(n, \log(1/\delta))$
  - $\text{FPAUS}$

**Self-reducibles [Jerrum-Valiant-Vazirani]:**

- Exact Counting
- Approx Counting
- Exact Sampling
- Approx Sampling
- if FPTAS
Density $\mu$ on space $\Omega$

- **Sampling:** $\mathbb{P}[\text{output}] \propto \mu(\text{output})$
- **Counting:** compute $\sum_x \mu(x)$
- **#P:** #accepting paths in TM

### Approx counting

$(1 + \epsilon)$-approx in $\text{poly}(n, 1/\epsilon)$

### Approx sampling

$\delta$-approx in $d_{TV}$ in $\text{poly}(n, \log(1/\delta))$

- **FPRAS/FPTAS**
- **FPAUS**

### #P-complete:

- Natural counting variants of known NP-complete problems.
- Natural counting variants of some P problems too!

---

**Note:**
- $\mathbb{P}$ denotes probability.
- $\mu$ denotes density.
- $\Delta$ denotes the total variation distance.
- FPRAS/FPTAS indicates Fully Polynomial Randomized Rounding (FP) Algorithms and Fully Polynomial Time Approximation Schemes (FPTAS).
- FPAUS indicates Fully Polynomial Approximation Scheme for Uniform Sampling (FPAUS).

---

2/20
Density $\mu$ on space $\Omega$

- **Sampling:** $\mathbb{P}[\text{output}] \propto \mu(\text{output})$
- **Counting:** compute $\sum_x \mu(x)$
- **#P:** #accepting paths in TM

**#P-complete:**
- Natural counting variants of known NP-complete problems.
- Natural counting variants of some P problems too!

Approx counting: $(1 + \epsilon)$-approx in $\text{poly}(n, 1/\epsilon)$

Approx sampling: $\delta$-approx in $d_{TV}$ in $\text{poly}(n, \log(1/\delta))$

- **FPRAS/FPTAS**
- **FPAUS**

Self-reducibles [Jerrum-Valiant-Vazirani]:

- **Exact Counting** $\rightarrow$ Approx Counting
  - if FPTAS

- **Exact Sampling** $\rightarrow$ Approx Sampling
DNF Counting
- Rejection sampling
- Monte Carlo estimation

Counting vs. Sampling
- Self-reducibility
- Reductions
- Total variation and coupling

Counting via Determinants if time
- Spanning trees
DNF Counting

- Rejection sampling
- Monte Carlo estimation

Counting vs. Sampling

- Self-reducibility
- Reductions
- Total variation and coupling

Counting via Determinants

- Spanning trees
DNF sampling [Karp-Luby]

\[ \phi = C_1 \lor C_2 \lor \cdots \lor C_m \]

\( A_i = \{ \text{sat assignments of } C_i \} \)

Sample u.r. \( \in A_1 \cup \cdots \cup A_m \)

\[ A_m \]
\[ \vdots \]
\[ A_2 \]
\[ A_1 \]

• reject  ○ accept
DNF sampling [Karp-Luby]

\[ \phi = C_1 \lor C_2 \lor \cdots \lor C_m \]

- \( A_i = \{ \text{sat assignments of } C_i \} \)
- Sample u.r. \( \in A_1 \bigcup A_2 \bigcup \cdots \bigcup A_m \)

Example

\[ \phi = x_1 \lor x_2 \]

\( C_1 \quad C_2 \)

\[ A_1 = \{ 10, 01 \}, \quad A_2 = \{ 01, 11 \} \]

Sample u.r. from \( \{ 10, 11, 01, 11 \} \), reject the second 11.
DNF sampling [Karp-Luby]

\[ \phi = C_1 \lor C_2 \lor \cdots \lor C_m \]

- \( A_i = \{ \text{sat assignments of } C_i \} \)
- Sample u.r. \( \in A_1 \biguplus \cdots \biguplus A_m \)

Example

\[ \phi = x_1 \lor x_2 \]

\[ C_1 \quad C_2 \]

- Goal: sample u.r. from \( A_1 \cup A_2 = \{10, 01, 11\} \)

\[ \bullet \text{ reject} \quad \bullet \text{ accept} \]
DNF sampling [Karp-Luby]

\[ \phi = C_1 \lor C_2 \lor \cdots \lor C_m \]

\[ A_i = \{ \text{sat assignments of } C_i \} \]

\[ \text{Sample u.r. } \in \bigcup_{i=1}^{m} A_i \]

**Example**

\[ \phi = x_1 \lor x_2 \]

\[ C_1 \quad C_2 \]

\[ \text{Goal: sample u.r. from } \bigcup_{i=1}^{2} A_i = \{10, 01, 11\} \]

\[ A_1 = \{10, 11\}, \quad A_2 = \{01, 11\} \]
DNF sampling [Karp-Luby]

\[ \phi = C_1 \lor C_2 \lor \cdots \lor C_m \]

\( A_i = \{ \text{sat assignments of } C_i \} \)

\( \text{Sample u.r. } \in A_1 \sqcup \cdots \sqcup A_m \)

**Example**

\[ \phi = x_1 \lor x_2 \]

\( C_1 \uparrow \quad C_2 \uparrow \)

\( A_1 \sqcup A_2 = \{10, 01, 11\} \)

\( A_1 = \{10, 11\}, A_2 = \{01, 11\} \)

\( \text{Sample u.r. from } \{10, 11, 01, 11\}, \text{ reject the second } 11 \)
DNF sampling [Karp-Luby]

\[ \phi = C_1 \lor C_2 \lor \cdots \lor C_m \]

- \( A_i = \{ \text{sat assignments of } C_i \} \)
- Sample u.r. \( \in A_1 \sqcup \cdots \sqcup A_m \)

**Example**

\[ \phi = x_1 \lor x_2 \]

\[ C_1 \quad \uparrow \quad C_2 \]

- Goal: sample u.r. from \( A_1 \cup A_2 = \{10, 01, 11\} \)
- \( A_1 = \{10, 11\}, A_2 = \{01, 11\} \)
- Sample u.r. from \( \{10, 11, 01, 11\} \), reject the second 11

How to sample \( \sim A_1 \sqcup \cdots \sqcup A_m \)?
DNF sampling [Karp-Luby]

\[ \phi = C_1 \lor C_2 \lor \cdots \lor C_m \]

- \( A_i = \{ \text{sat assignments of } C_i \} \)
- Sample u.r. \( \in A_1 \biguplus \cdots \biguplus A_m \)

**Example**

\[ \phi = x_1 \lor x_2 \]
\[ C_1 \uparrow \quad C_2 \uparrow \]

- Goal: sample u.r. from \( A_1 \cup A_2 = \{10, 01, 11\} \)
- \( A_1 = \{10, 11\}, A_2 = \{01, 11\} \)
- Sample u.r. from \( \{10, 11, 01, 11\} \), reject the second 11

How to sample \( \sim A_1 \biguplus \cdots \biguplus A_m \)?
- Sample \( i \) w.p. \( \propto |A_i| \)
DNF sampling [Karp-Luby]

\[ \phi = C_1 \lor C_2 \lor \cdots \lor C_m \]

- \( A_i = \{\text{sat assignments of } C_i\} \)
- Sample u.r. \( \in A_1 \bigcup \cdots \bigcup A_m \)

Example

\[ \phi = x_1 \lor x_2 \]

- Goal: sample u.r. from 
  \( A_1 \cup A_2 = \{10, 01, 11\} \)
- \( A_1 = \{10, 11\}, A_2 = \{01, 11\} \)
- Sample u.r. from \( \{10, 11, 01, 11\} \), reject the second 11

How to sample \( \sim A_1 \bigcup \cdots \bigcup A_m \)?

- Sample \( i \) w.p. \( \propto |A_i| \)
- Sample \( x \in A_i \) u.a.r.
How to count solutions?

Idea: Write $|A_1 \cup \cdots \cup A_m|$ as $|A_1| \bigoplus \cdots \bigoplus |A_m|$ easy to compute.

Approximate accept prob

for $i = 1, \ldots, t$
do sample $\sim A_1 \bigoplus \cdots \bigoplus A_m$ and $X_i \leftarrow 1$ [accept]

return $X = X_1 + \cdots + X_t$

$E[X_i] = p$, $\text{Var}(X_i) = p(1-p)$

By Chebyshev's inequality

$P\left[X \notin [p - \epsilon p, p + \epsilon p]\right] \leq \text{Var}(X_i) \left(\epsilon p/3\right)^2$

which is $\leq 9/tp^2 \epsilon^2$.

Enough to let $t > 27/p^2 \epsilon^2$ to have success with prob $\geq 2/3$.

Lemma  To mult. estimate $p$ from $\text{Ber}(p)$ samples, $O(1/p^2 \epsilon^2)$ many enough.
How to count solutions?

Idea: Write $|A_1 \cup \cdots \cup A_m|$ as

$$|A_1 \prod \cdots \prod A_m| \cdot \frac{|A_1 \cup \cdots \cup A_m|}{|A_1 \prod \cdots \prod A_m|}$$

- easy to compute
- accept prob
DNF counting [Karp-Luby]

How to count solutions?

**Idea:** Write \(|A_1 \cup \cdots \cup A_m|\) as 
\[
|A_1 \bigcap \cdots \bigcap A_m| \cdot \frac{|A_1 \cup \cdots \cup A_m|}{|A_1 \bigcap \cdots \bigcap A_m|}
\]

- easy to compute
- accept prob

**Approximate accept prob \(p\)**

**Monte Carlo estimation**

\[
\text{for } i = 1, \ldots, t \text{ do}
\]
\[
\begin{align*}
\text{sample } &\sim A_1 \bigcap \cdots \bigcap A_m \\
\text{and } X_i &\leftarrow 1[\text{accept}]
\end{align*}
\]

\[
\text{return } X = \frac{X_1 + \cdots + X_t}{t}
\]
DNF counting [Karp-Luby]

How to count solutions?

\[ \mathbb{E}[X_i] = p \quad \text{Var}(X_i) = p(1 - p) \]

\[ \text{Idea: Write } |A_1 \cup \cdots \cup A_m| \text{ as} \]

\[ |A_1 \bigcap \cdots \bigcap A_m|: \frac{|A_1 \cup \cdots \cup A_m|}{|A_1 \bigcap \cdots \bigcap A_m|} \]

easy to compute \quad \text{accept prob}

\[ \text{Approximate accept prob } p \]

Monte Carlo estimation

\[
\begin{align*}
\text{for } i = 1, \ldots, t \text{ do} \\
\quad \text{sample } \sim A_1 \bigcap \cdots \bigcap A_m \\
\quad \text{and } X_i \leftarrow 1[\text{accept}] \\
\text{return } X = \frac{X_1 + \cdots + X_t}{t}
\end{align*}
\]
DNF counting [Karp-Luby]

How to count solutions?

- **Idea:** Write $|A_1 \cup \cdots \cup A_m|$ as $|A_1 \prod \cdots \prod A_m|$, $|A_1 \cup \cdots \cup A_m|$, easy to compute, accept prob

- **Approximate accept prob $p$**

**Monte Carlo estimation**

```plaintext
for i = 1, ..., t do
    sample $\sim A_1 \prod \cdots \prod A_m$
    and $X_i \leftarrow 1[\text{accept}]
return $X = \frac{X_1 + \cdots + X_t}{t}$
```

- $\mathbb{E}[X_i] = p$, $\text{Var}(X_i) = p(1 - p)$
- $\mathbb{E}[X] = p$, $\text{Var}(X) = p(1 - p)/t$
How to **count** solutions?

**Idea:** Write $|A_1 \bigcup \cdots \bigcup A_m|$ as

$$|A_1 \bigcap \cdots \bigcap A_m| \cdot \frac{|A_1 \cup \cdots \cup A_m|}{|A_1 \bigcap \cdots \bigcap A_m|}$$

easy to compute  
accept prob

**Approximate accept prob $p$**

**Monte Carlo estimation**

```plaintext
for i = 1, \ldots, t do
    sample \sim A_1 \bigcap \cdots \bigcap A_m
    and $X_i \leftarrow 1[\text{accept}]$
return $X = \frac{X_1 + \cdots + X_t}{t}$
```

\[ \mathbb{E}[X_i] = p \quad \text{Var}(X_i) = p(1 - p) \]

\[ \mathbb{E}[X] = p \quad \text{Var}(X) = p(1 - p)/t \]

By Chebyshev’s inequality

\[
\mathbb{P}\left[ X \notin \left[ p - \frac{ep}{3}, p + \frac{ep}{3} \right] \right] \leq \frac{\text{Var}(X)}{(ep/3)^2}
\]

which is $\leq 9/tp\epsilon^2$. 
**DNF counting [Karp-Luby]**

How to count solutions?

- **Idea:** Write $|A_1 \cup \cdots \cup A_m|$ as $|A_1 \bigcap \cdots \bigcap A_m| \cdot \frac{|A_1 \cup \cdots \cup A_m|}{|A_1 \bigcap \cdots \bigcap A_m|}$.

- Approximate accept prob $p$

---

**Monte Carlo estimation**

```plaintext
for i = 1, \ldots, t do
    sample \sim A_1 \bigcap \cdots \bigcap A_m
    and $X_i \leftarrow 1[\text{accept}]$
return $X = \frac{X_1 + \cdots + X_t}{t}$
```

- $\mathbb{E}[X_i] = p$, $\text{Var}(X_i) = p(1 - p)$
- $\mathbb{E}[X] = p$, $\text{Var}(X) = p(1 - p)/t$
- By Chebyshev’s inequality

\[
P[X \notin \left[ p - \frac{ep}{3}, p + \frac{ep}{3} \right]] \leq \frac{\text{Var}(X)}{(ep/3)^2}
\]

which is $\leq 9/tp\epsilon^2$.

- Enough to let $t > 27/p\epsilon^2$ to have success with prob $\geq 2/3$. 

---
DNF counting [Karp-Luby]

How to count solutions?

Idea: Write $|A_1 \cup \cdots \cup A_m|$ as $|A_1 \prod \cdots \prod A_m|/|A_1 \prod \cdots \prod A_m|$. Easy to compute and accept prob.

Approximate accept prob $p$

Monte Carlo estimation

for $i = 1, \ldots, t$ do
    sample $\sim A_1 \prod \cdots \prod A_m$ and $X_i \leftarrow 1[\text{accept}]$
return $X = \frac{X_1 + \cdots + X_t}{t}$

$\mathbb{E}[X_i] = p$  $\text{Var}(X_i) = p(1 - p)$

$\mathbb{E}[X] = p$  $\text{Var}(X) = p(1 - p)/t$

By Chebyshev’s inequality

$$\mathbb{P}\left[X \notin \left[p - \frac{ep}{3}, p + \frac{ep}{3}\right]\right] \leq \frac{\text{Var}(X)}{(ep/3)^2}$$

which is $\leq 9/tp\epsilon^2$.

Enough to let $t > 27/p\epsilon^2$ to have success with prob $\geq 2/3$.

Lemma

To mult. estimate $p$ from $\text{Ber}(p)$ samples, $O(1/p\epsilon^2)$ many enough.
Open problem: Is there an FPTAS for DNF counting?
Open problem: Is there an FPTAS for DNF counting?

[Gopalan-Meka-Reingold’12]

±\(\epsilon 2^n\) approximation in time

\(n^{\tilde{O}(\log \log n)}\)
DNF Counting
- Rejection sampling
- Monte Carlo estimation

Counting vs. Sampling
- Self-reducibility
- Reductions
- Total variation and coupling

Counting via Determinants
- Spanning trees
DNF Counting
- Rejection sampling
- Monte Carlo estimation

Counting vs. Sampling
- Self-reducibility
- Reductions
- Total variation and coupling

Counting via Determinants
- Spanning trees

if time
Self-reducible problems

advanced: measure-decomposed

Solutions of instance $I$ partitioned. Each part $\equiv$ smaller instance $I'$.

\[(x_1 \lor \overline{x}_2 \lor x_3) \land (\overline{x}_1 \lor x_4)\]

Key: branching factor, depth $\leq$ poly
Self-reducible problems

advanced: measure-decomposed

Solutions of instance $I$ partitioned. Each part $\equiv$ smaller instance $I'$.

\[(x_1 \lor \bar{x}_2 \lor x_3) \land (\bar{x}_1 \lor x_4)\]

- $x_1 \leftarrow 0$
- $x_1 \leftarrow 1$
- $x_2 \leftarrow 0$
- $x_2 \leftarrow 1$

Key: branching factor, depth $\leq \text{poly}$

Other requirements:
- Instances $I'$ produced efficiently.
- One-to-one correspondence of solutions efficiently computable.
- Base cases easy to sample/count.

Example: perfect matchings $\not\in$ matching $\in$ matching
Self-reducible problems

Solutions of instance $I$ partitioned. Each part $\equiv$ smaller instance $I'$.

\[(x_1 \lor \bar{x}_2 \lor x_3) \land (\bar{x}_1 \lor x_4)\]

- $x_1 \leftarrow 0$
- $x_1 \leftarrow 1$
- $x_2 \leftarrow 0$
- $x_2 \leftarrow 1$
- $x_4$ (false, true, ...)

Key: branching factor, depth $\leq \text{poly}$

Other requirements:
- Instances $I'$ produced efficiently.
- One-to-one correspondence of solutions efficiently computable.
- Base cases easy to sample/count.

Example: perfect matchings / matching
Self-reducible problems

Solutions of instance $I$ partitioned. Each part $≡$ smaller instance $I'$.

$$(x_1 \lor \overline{x_2} \lor x_3) \land (\overline{x_1} \lor x_4)$$

Other requirements:

- Instances $I'$ produced efficiently.
- One-to-one correspondence of solutions efficiently computable.

Key: branching factor, depth $\leq$ poly
Self-reducible problems

Solutions of instance $I$ partitioned. Each part $\equiv$ smaller instance $I'$.

\[
(x_1 \lor \overline{x_2} \lor x_3) \land (\overline{x_1} \lor x_4)
\]

$x_1 \leftarrow 0$
$x_1 \leftarrow 1$
\[
(\overline{x_2} \lor x_3)
\]
$x_2 \leftarrow 0$
$x_2 \leftarrow 1$
true
false
\[x_4\]
\[\ldots \ldots \]

Key: branching factor, depth $\leq \text{poly}$

Other requirements:
- Instances $I'$ produced efficiently.
- One-to-one correspondence of solutions efficiently computable.
- Base cases easy to sample/count.
Self-reducible problems

Solutions of instance $I$ partitioned. Each part $\equiv$ smaller instance $I'$.

\[(x_1 \lor \overline{x_2} \lor x_3) \land (\overline{x_1} \lor x_4)\]

\[
x_1 \leftarrow 0 \quad x_1 \leftarrow 1
\]

\[
(x_2 \lor x_3) \quad x_4
\]

\[
x_2 \leftarrow 0 \quad x_2 \leftarrow 1
\]

true \quad false \quad \ldots \quad \ldots

Key: branching factor, depth $\leq \text{poly}$

Other requirements:
- Instances $I'$ produced efficiently.
- One-to-one correspondence of solutions efficiently computable.
- Base cases easy to sample/count.

Example: perfect matchings

[Diagram showing matching and non-matching graphs]
Example: spanning trees

\[ \text{delete} \quad \notin \text{tree} \quad \in \text{tree} \quad \text{contract} \]

Example: independent sets

\[ \notin \text{ind set} \quad \in \text{ind set} \]

Non-example: colorings

Instance: graph \( G = (V, E) \) and \( q > 0 \)

Solutions:

\[ x \in [q] V \text{ with } x_u \neq x_v \text{ for adjacent } u, v \]

Note that

\[ \# \begin{bmatrix} v \\ u \end{bmatrix} = \# \begin{bmatrix} v \\ u \end{bmatrix} - \# \begin{bmatrix} u/v \end{bmatrix} , \]

but this is not self-reducibility.
Example: spanning trees

Example: independent sets

Solutions: $x \in [q]_V$ with $x_u \neq x_v$ for adjacent $u, v$.

Note that $\# \begin{pmatrix} v \\ u \end{pmatrix} = \# \begin{pmatrix} v \\ u \end{pmatrix} - \# \begin{pmatrix} u/v \end{pmatrix}$, but this is not self-reducibility.
Example: spanning trees

Non-example: colorings

Instance: graph $G = (V, E)$ and $q > 0$

Solutions: $x \in [q]^V$ with $x_u \neq x_v$ for adjacent $u, v$

Note that $\# \begin{pmatrix} v \\ u \end{pmatrix} = \# \begin{pmatrix} v \\ u \end{pmatrix} - \# \begin{pmatrix} u/v \end{pmatrix}$, but this is not self-reducibility.
Example: spanning trees

- $\notin \text{tree}$ (delete)  
- $\in \text{tree}$ (contract)

Example: independent sets

- $\notin \text{ind set}$  
- $\in \text{ind set}$

Non-example: colorings

Instance: graph $G = (V, E)$ and $q > 0$
Solutions: $x \in [q]^V$ with $x_u \neq x_v$ for adjacent $u, v$

Note that

$\# \left( \begin{array}{cc} u & v \\ \end{array} \right) = \# \left( \begin{array}{cc} u & v \\ \end{array} \right) - \# \left( \begin{array}{cc} u/v \\ \end{array} \right)$,

but this is not self-reducibility.
**Theorem [Jerrum-Valiant-Vazirani]**

For “self-reducible” problems:

approx counting $\equiv$ approx sampling

(FPRAS)

Exact Counting $\rightarrow$ Approx Counting

if FPTAS

Exact Sampling $\rightarrow$ Approx Sampling

(FPAUS)

arrows are poly-time reductions
**Theorem [Jerrum-Valiant-Vazirani]**

For “self-reducible” problems:

```
approx counting \equiv \text{approx sampling}
```

Exact Counting \implies Exact Sampling

(FPRAS)

Exact Counting \implies Approx Counting

if FPTAS

Exact Sampling \implies Approx Sampling

(FPAUS)

arrows are poly-time reductions

while I not base case do

\[
\text{compute children } I_1, \ldots, I_k
\]

\[
\text{for } i = 1, \ldots, k \text{ do}
\]

\[
c_i \leftarrow \#(I_i)
\]

\[
\text{choose } i \text{ w.p. } \propto c_i
\]

\[
I \leftarrow I_i
\]

output sample for I

\[
\mathbb{P}[\text{sample}] = \frac{\#(I_i)}{\#(I)} \cdot \frac{\#(I_{ij})}{\#(I_i)} \cdot \ldots = \frac{1}{\#(I)}
\]
FPTAS $\Rightarrow$ Exact Sampling

Instead of $c_i = \#(I_i)$, compute $1 + \epsilon \approx \tilde{c}_i$.

We get $P[\text{sample}] \text{ is } (1 + \epsilon)$ depth approx to $1/\#(I_i)$.

Set $\epsilon \approx 1/\text{depth}$:

$P[\text{sample}] = \Theta(1/\#(I_i))$.

Idea: if $\nu$ is output dist, we can compute $\nu(x)$. Rejection sample this into the target dist $\mu$.

Since $\mu(x) = O(\nu(x))$ for all $x$, it takes only $O(1)$ rejections.

FPRAS $\Rightarrow$ Approx Sampling

Now there is a chance of error. But we only want $d_{TV} \leq \delta$.

Idea: cut rejection sampling after $O(\log 1/\delta)$ iterations:

$P[\text{not finishing}] \leq \delta/2$.

Total number of approx counts we need is $\text{poly}(n) \log(1/\delta)$.

Make sure each fails with prob $\leq \delta^2 \cdot 1/\text{poly}(n) \log(1/\delta)$.

Runtime: $\text{poly}(n, \log(1/\delta))$.
FPTAS $\implies$ Exact Sampling

Instead of $c_i = \#(I_i)$, compute $1 + \epsilon \text{ approx } \tilde{c}_i$. 
FPTAS $\implies$ Exact Sampling

- Instead of $c_i = \#(I_i)$, compute $1 + \epsilon$ approx $\tilde{c}_i$.
- We get $\mathbb{P}[\text{sample}]$ is $(1 + \epsilon)^{\text{depth}}$ approx to $1/\#(I)$.
FPTAS $\implies$ Exact Sampling

- Instead of $c_i = \#(I_i)$, compute $1 + \epsilon$ approx $\tilde{c}_i$.
- We get $\mathbb{P}[\text{sample}]$ is $(1 + \epsilon)^{\text{depth}}$ approx to $1/\#(I)$.
- Set $\epsilon \approx 1/\text{depth}$:

$$\mathbb{P}[\text{sample}] = \Theta\left(\frac{1}{\#(I)}\right).$$
FPTAS $\implies$ Exact Sampling

- Instead of $c_i = \#(I_i)$, compute $1 + \epsilon$ approx $\tilde{c}_i$.
- We get $\mathbb{P}[\text{sample}]$ is $(1 + \epsilon)^{\text{depth}}$ approx to $1/\#(I)$.
- Set $\epsilon \approx 1/\text{depth}$:

\[
\mathbb{P}[\text{sample}] = \Theta\left(\frac{1}{\#(I)}\right).
\]

- Idea: if $\nu$ is output dist, we can compute $\nu(x)$. Rejection sample this into the target dist $\mu$. 

FPRAS $\implies$ Approx Sampling

Now there is a chance of error. But we only want $d_{TV} \leq \delta$.

Idea: cut rejection sampling after $O(\log \frac{1}{\delta})$ iterations: $\mathbb{P}[\text{not finishing}] \leq \frac{\delta}{2}$

Total number of approx counts we need is $\text{poly}(n) \log \left(\frac{1}{\delta}\right)$.

Make sure each fails with prob $\leq \delta^2 \cdot \text{poly}(n) \log \left(\frac{1}{\delta}\right)$.

Runtime: $\text{poly}(n, \log \left(\frac{1}{\delta}\right))$.
FPTAS $\implies$ Exact Sampling

- Instead of $c_i = \#(I_i)$, compute $1 + \epsilon$ approx $\tilde{c}_i$.
- We get $\Pr[\text{sample}] \approx (1 + \epsilon)\text{depth}$ approx to $1/\#(I)$.
- Set $\epsilon \approx 1/\text{depth}$:
  \[
  \Pr[\text{sample}] = \Theta\left(\frac{1}{\#(I)}\right).
  \]

- **Idea:** if $\nu$ is output dist, we can compute $\nu(x)$. Rejection sample this into the target dist $\mu$.
- Since $\mu(x) = O(\nu(x))$ for all $x$, it takes only $O(1)$ rejections.
FPTAS $\implies$ Exact Sampling

- Instead of $c_i = \#(I_i)$, compute $1 + \epsilon$ approx $\tilde{c}_i$.
- We get $\mathbb{P}[\text{sample}]$ is $(1 + \epsilon)^{\text{depth}}$ approx to $1/\#(I)$.
- Set $\epsilon \approx 1/\text{depth}$:
  \[ \mathbb{P}[\text{sample}] = \Theta\left(\frac{1}{\#(I)}\right). \]

- Idea: if $\nu$ is output dist, we can compute $\nu(x)$. Rejection sample this into the target dist $\mu$.
- Since $\mu(x) = O(\nu(x))$ for all $x$, it takes only $O(1)$ rejections.

FPRAS $\implies$ Approx Sampling

Now there is a chance of error. 😞
But we only want $d_{TV} \leq \delta$. 😊
FPTAS \implies \text{Exact Sampling}

\begin{itemize}
\item Instead of \( c_i = \#(I_i) \), compute \( 1 + \epsilon \) approx \( \tilde{c}_i \).
\item We get \( \mathbb{P}[	ext{sample}] \) is \((1 + \epsilon)^\text{depth} \) approx to \( 1/\#(I) \).
\item Set \( \epsilon \approx 1/\text{depth} \):

\[
\mathbb{P}[	ext{sample}] = \Theta\left(\frac{1}{\#(I)}\right).
\]
\end{itemize}

\begin{itemize}
\item Idea: if \( \nu \) is output dist, we can compute \( \nu(x) \). Rejection sample this into the target dist \( \mu \).
\item Since \( \mu(x) = O(\nu(x)) \) for all \( x \), it takes only \( O(1) \) rejections.
\end{itemize}

FPRAS \implies \text{Approx Sampling}

Now there is a chance of error. 😞
But we only want \( d_{TV} \leq \delta \). 😊

\begin{itemize}
\item Idea: cut rejection sampling after \( O(\log 1/\delta) \) iterations:

\[
\mathbb{P}[	ext{not finishing}] \leq \delta/2
\]
\end{itemize}

Runtime: \( \text{poly}(n, \log(1/\delta)) \)
FPTAS $\implies$ Exact Sampling

- Instead of $c_i = \#(I_i)$, compute $1 + \epsilon$ approx $\tilde{c}_i$.
- We get $P[\text{sample}]$ is $(1 + \epsilon)^{\text{depth}}$ approx to $1/\#(I)$.
- Set $\epsilon \approx 1/\text{depth}$:
  $$P[\text{sample}] = \Theta\left(\frac{1}{\#(I)}\right).$$
- Idea: if $\nu$ is output dist, we can compute $\nu(x)$. Rejection sample this into the target dist $\mu$.
- Since $\mu(x) = O(\nu(x))$ for all $x$, it takes only $O(1)$ rejections.

FPRAS $\implies$ Approx Sampling

Now there is a chance of error. 😞
But we only want $d_{TV} \leq \delta$. 😊

- Idea: cut rejection sampling after $O(\log 1/\delta)$ iterations:
  $$P[\text{not finishing}] \leq \delta/2$$
- Total number of approx counts we need is $\text{poly}(n) \log(1/\delta)$. 

Runtime: $\text{poly}(n, \log(1/\delta))$. 

12/20
FPTAS $\implies$ Exact Sampling

- Instead of $c_i = \#(I_i)$, compute $1 + \epsilon$ approx $\tilde{c}_i$.
- We get $\Pr[\text{sample}]$ is $(1 + \epsilon)^{\text{depth}}$ approx to $1/\#(I)$.
- Set $\epsilon \approx 1/\text{depth}$:

  $$\Pr[\text{sample}] = \Theta\left(\frac{1}{\#(I)}\right).$$

- **Idea**: if $\nu$ is output dist, we can compute $\nu(x)$. Rejection sample this into the target dist $\mu$.
- Since $\mu(x) = O(\nu(x))$ for all $x$, it takes only $O(1)$ rejections.

FPRAS $\implies$ Approx Sampling

Now there is a chance of error. 😞

But we only want $d_{TV} \leq \delta$. 😊

- **Idea**: cut rejection sampling after $O(\log 1/\delta)$ iterations:

  $$\Pr[\text{not finishing}] \leq \delta/2$$

- Total number of approx counts we need is $\text{poly}(n) \log(1/\delta)$.
- Make sure each fails with prob

  $$\leq \frac{\delta}{2} \cdot \frac{1}{\text{poly}(n) \log(1/\delta)}$$
FPTAS $\iff$ Exact Sampling

- Instead of $c_i = \#(I_i)$, compute $1 + \epsilon$ approx $\tilde{c}_i$.
- We get $\mathbb{P}[\text{sample}]$ is $(1 + \epsilon)^{\text{depth}}$ approx to $1/\#(I)$.
- Set $\epsilon \approx 1/\text{depth}$:
  \[
  \mathbb{P}[\text{sample}] = \Theta\left(\frac{1}{\#(I)}\right).
  \]

Idea: if $\nu$ is output dist, we can compute $\nu(x)$. Rejection sample this into the target dist $\mu$.

Since $\mu(x) = O(\nu(x))$ for all $x$, it takes only $O(1)$ rejections.

FPRAS $\iff$ Approx Sampling

Now there is a chance of error. 😞

But we only want $d_{TV} \leq \delta$. 😊

- Idea: cut rejection sampling after $O(\log 1/\delta)$ iterations:
  \[
  \mathbb{P}[\text{not finishing}] \leq \delta/2
  \]

- Total number of approx counts we need is $\text{poly}(n) \log(1/\delta)$.

- Make sure each fails with prob
  \[
  \leq \frac{\delta}{2} \cdot \frac{1}{\text{poly}(n) \log(1/\delta)}
  \]

- Runtime: $\text{poly}(n, \log(1/\delta))$. 😊
Exact Sampling $\implies$ Approx Counting

Idea: choose root $\rightarrow$ leaf path

Estimate $\#(I_1)/\#(I)$, $\#(I_{11})/\#(I_1)$, $\ldots$ using Monte Carlo.

Multiply with $\#(I_{\text{base}})$ and output.

Need $1 + \epsilon/(2 \cdot \text{depth})$ approx for each ratio.

Set failure prob for each estimation task to $\leq 1/(6 \cdot \text{depth})$.

Approx factor: $(1 + \epsilon/2 \cdot \text{depth})^{\text{depth}} \leq 1 + \epsilon$

Success prob: $\geq 1 - \text{depth} \cdot 1/6 \cdot \text{depth} \geq 5/6$

Problem: if any ratio $p$ is small, it takes $\geq 1/p$ time to estimate.
Exact Sampling $\implies$ Approx Counting

Idea: choose root $\rightarrow$ leaf path

Estimate $\#(I_1)/\#(I_1), \#(I_{111})/\#(I_1), \ldots$ using Monte Carlo. Multiply with $\#(I_{\text{base}})$ and output.

Need $1 + \epsilon/ (2 \cdot \text{depth})$ approx for each ratio.

Set failure prob for each estimation task to $\leq 1/ (6 \cdot \text{depth})$.

Approx factor: $(1 + \epsilon^2 \cdot \text{depth})^\text{depth} \leq 1 + \epsilon$

Success prob: $\geq 1 - \text{depth} \cdot 1/ (6 \cdot \text{depth}) \geq 5/6$

Problem: if any ratio $p$ is small, it takes $\geq 1/p$ time to estimate.
Exact Sampling $\implies$ Approx Counting

Idea: choose root $\rightarrow$ leaf path

- Estimate $\#(I_1)/\#(I)$, $\#(I_{11})/\#(I_1)$, $\ldots$ using Monte Carlo.

Need $1 + \epsilon/\left(2 \cdot \text{depth}\right)$ approx for each ratio.

Set failure prob for each estimation task to $\leq 1/\left(6 \cdot \text{depth}\right)$.

Approx factor: $\left(1 + \epsilon/2 \cdot \text{depth}\right)^\text{depth} \leq 1 + \epsilon$

Success prob: $\geq 1 - \text{depth} \cdot 1/6 \cdot \text{depth} \geq 5/6$.

Problem: if any ratio $p$ is small, it takes $\geq 1/p$ time to estimate.
Exact Sampling $\implies$ Approx Counting

Idea: choose root $\rightarrow$ leaf path

Estimate $\#(I_1)/\#(I)$, $\#(I_{11})/\#(I_1)$, $\ldots$ using Monte Carlo.

Multiply with $\#(I_{\text{base}})$ and output.
Exact Sampling $\implies$ Approx Counting  

<table>
<thead>
<tr>
<th>$I$</th>
<th>$I_1$</th>
<th>$I_2$</th>
<th>$\ldots$</th>
<th>$I_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$I_{11}$</td>
<td></td>
<td>$\ldots$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$I_{\text{base}}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Need** $1 + \epsilon/(2 \cdot \text{depth})$ approx for each ratio.

- **Idea:** choose root $\rightarrow$ leaf path
- **Estimate** $\#(I_1)/\#(I)$, $\#(I_{11})/\#(I_1)$, $\ldots$ using Monte Carlo.
- **Multiply** with $\#(I_{\text{base}})$ and output.
Exact Sampling $\implies$ Approx Counting

- Idea: choose root $\rightarrow$ leaf path
- Estimate $(I_1)/\#(I)$, $(I_{11})/\#(I_1)$, $\ldots$ using Monte Carlo.
- Multiply with $\#(I_{\text{base}})$ and output.

- Need $1 + \epsilon/(2 \cdot \text{depth})$ approx for each ratio.
- Set failure prob for each estimation task to $\leq 1/(6 \cdot \text{depth})$.

Problem: if any ratio $p$ is small, it takes $\geq 1/p$ time to estimate.
Exact Sampling $\implies$ Approx Counting

- Need $1 + \epsilon/(2 \cdot \text{depth})$ approx for each ratio.
- Set failure prob for each estimation task to $\leq 1/(6 \cdot \text{depth})$.
- Approx factor:

$$\left(1 + \frac{\epsilon}{2 \cdot \text{depth}}\right)^{\text{depth}} \leq 1 + \epsilon$$

- Idea: choose root $\rightarrow$ leaf path
- Estimate $(\#(I_1)/\#(I), (\#(I_{11})/\#(I_1), \ldots$ using Monte Carlo.
- Multiply with $\#(I_{\text{base}})$ and output.
Exact Sampling $\implies$ Approx Counting

- Need $1 + \epsilon / (2 \cdot \text{depth})$ approx for each ratio.
- Set failure prob for each estimation task to $\leq 1 / (6 \cdot \text{depth})$.
- Approx factor:
  \[
  \left(1 + \frac{\epsilon}{2 \cdot \text{depth}}\right)^{\text{depth}} \leq 1 + \epsilon
  \]
- Success prob:
  \[
  \geq 1 - \text{depth} \cdot \frac{1}{6 \cdot \text{depth}} \geq \frac{5}{6}
  \]

- Idea: choose root $\rightarrow$ leaf path
- Estimate $\#(I_1) / \#(I)$, $\#(I_{11}) / \#(I_1)$, $\ldots$ using Monte Carlo.
- Multiply with $\#(I_{\text{base}})$ and output.
Exact Sampling $\implies$ Approx Counting

- Idea: choose root $\rightarrow$ leaf path
- Estimate $\#(I_1)/\#(I)$, $\#(I_{11})/\#(I_1)$, $\ldots$ using Monte Carlo.
- Multiply with $\#(I_{\text{base}})$ and output.

- Need $1 + \epsilon/(2 \cdot \text{depth})$ approx for each ratio.
- Set failure prob for each estimation task to $\leq 1/(6 \cdot \text{depth})$.
- Approx factor:
  \[
  \left(1 + \frac{\epsilon}{2 \cdot \text{depth}}\right)^{\text{depth}} \leq 1 + \epsilon
  \]
- Success prob:
  \[
  \geq 1 - \text{depth} \cdot \frac{1}{6 \cdot \text{depth}} \geq \frac{5}{6}
  \]
- Problem: if any ratio $p$ is small, it takes $\geq 1/p$ time to estimate.
Fix: while \( \#(I_1)/\#(I) \) could be small, \( \exists i \) s.t. \( \#(I_i)/\#(I) \) is large.
Fix: while $\#(I_1)/\#(I)$ could be small, $\exists i$ s.t. $\#(I_i)/\#(I)$ is large.

Take a sample $x$ and see which $I_i$ it belongs to. Assume

$$\frac{\#(I_i)}{\#(I)} \geq \frac{1}{6k \cdot \text{depth}}$$
Fix: while \( \#(I_1)/\#(I) \) could be small, \( \exists i \) s.t. \( \#(I_i)/\#(I) \) is large.

Take a sample \( x \) and see which \( I_i \) it belongs to. Assume

\[
\frac{\#(I_i)}{\#(I)} \geq \frac{1}{6k \cdot \text{depth}}
\]

Branch into \( I_i \) and recursively find the root \( \rightarrow \) leaf path.
Fix: while $\#(I_1)/\#(I)$ could be small, $\exists i$ s.t. $\#(I_i)/\#(I)$ is large.

Take a sample $x$ and see which $I_i$ it belongs to. Assume

$$\frac{\#(I_i)}{\#(I)} \geq \frac{1}{6k \cdot \text{depth}}$$

Branch into $I_i$ and recursively find the root $\rightarrow$ leaf path.

Prob of wrong assumption: $\leq 1/6$
Fix: while \( \#(I_1)/\#(I) \) could be small, \( \exists i \) s.t. \( \#(I_i)/\#(I) \) is large.

Take a sample \( x \) and see which \( I_i \) it belongs to. Assume

\[
\frac{\#(I_i)}{\#(I)} \geq \frac{1}{6k \cdot \text{depth}}
\]

Branch into \( I_i \) and recursively find the root \( \rightarrow \) leaf path.

Prob of wrong assumption: \( \leq 1/6 \)
Fix: while \( \#(I_1)/\#(I) \) could be small, \( \exists i \) s.t. \( \#(I_i)/\#(I) \) is large.

Take a sample \( x \) and see which \( I_i \) it belongs to. Assume

\[
\frac{\#(I_i)}{\#(I)} \geq \frac{1}{6k \cdot \text{depth}}
\]

Branch into \( I_i \) and recursively find the root \( \rightarrow \text{leaf path} \).

Prob of wrong assumption: \( \leq 1/6 \)

Approx Sampling \( \implies \) Approx Counting

We have a poly-time randomized algorithm that uses samples.
Fix: while \( \#(I_1)/\#(I) \) could be small, \( \exists i \) s.t. \( \#(I_i)/\#(I) \) is large.

Take a sample \( x \) and see which \( I_i \) it belongs to. Assume

\[
\frac{\#(I_i)}{\#(I)} \geq \frac{1}{6k \cdot \text{depth}}
\]

Branch into \( I_i \) and recursively find the root \( \rightarrow \) leaf path.

Prob of wrong assumption: \( \leq 1/6 \)

Approx Sampling \( \longrightarrow \) Approx Counting

- We have a poly-time randomized algorithm that uses samples.
- In general in such algorithms, exact samplers can be replaced by approx samplers.

Lemma: In a randomized poly-time algorithm, exact samplers can be replaced by FPAUS while guaranteeing the output changes no more than \( \delta \) in \( d_{TV} \) at the cost of \( \text{poly}(n, \log(1/\delta)) \) in runtime.
Fix: while \(#(I_1)/#(I)\) could be small, \(\exists i \text{ s.t. } #(I_i)/#(I)\) is large.

Take a sample \(x\) and see which \(I_i\) it belongs to. Assume

\[
\frac{ #(I_i)}{ #(I)} \geq \frac{1}{6k \cdot \text{depth}}
\]

Branch into \(I_i\) and recursively find the root \(\rightarrow\) leaf path.

Prob of wrong assumption: \(\leq 1/6\)

---

**Approx Sampling \(\rightarrow\) Approx Counting**

- We have a poly-time randomized algorithm that uses samples.
- In general in such algorithms, **exact samplers** can be replaced by **approx samplers**.

**Lemma**

In a randomized poly-time algorithm, exact samplers can be replaced by FPAUS while guaranteeing the output changes no more than \(\delta\) in \(d_{TV}\) at the cost of \(\text{poly}(n, \log(1/\delta))\) in runtime.
For dists $\mu$, $\nu$, a coupling is a joint dist $\pi$ of $(X, Y)$ where $X \sim \mu$ and $Y \sim \nu$. The minimum
\[
\min \{ P(X, Y) \sim \pi \mid X \neq Y \} \mid \text{coupling } \pi
\]
is $d_{TV}(\mu, \nu)$.
Proof: exercise!
Useful mindset: think of coupling as an alg to produce $X, Y$.
Compose these algs together.
For dists \( \mu, \nu \), a coupling is a joint dist \( \pi \) of \((X, Y)\) where \( X \sim \mu \) and \( Y \sim \nu \).

**Theorem**

The minimum

\[
\min \left\{ \mathbb{P}_{(X,Y) \sim \pi}[X \neq Y] \mid \text{coupling } \pi \right\}
\]

is \( d_{TV}(\mu, \nu) \).
Coupling

For dists $\mu, \nu$, a coupling is a joint dist $\pi$ of $(X, Y)$ where $X \sim \mu$ and $Y \sim \nu$.

**Theorem**

The minimum

$$\min\{\mathbb{P}_{(X,Y) \sim \pi}[X \neq Y] \mid \text{coupling } \pi\}$$

is $d_{TV}(\mu, \nu)$.

Proof: exercise!
For dists $\mu, \nu$, a coupling is a joint dist $\pi$ of $(X, Y)$ where $X \sim \mu$ and $Y \sim \nu$.

**Theorem**

The minimum

$$\min \{ \mathbb{P}_{(X,Y) \sim \pi}[X \neq Y] \mid \text{coupling } \pi \}$$

is $d_{TV}(\mu, \nu)$.

- Proof: exercise!
- **Useful mindset**: think of coupling as an alg to produce $X, Y$. Compose these algs together.
Replacing exact samples with approx samples

- Suppose alg uses samples $X_1, \ldots, X_m$.

  - Instead feed it samples $Y_1, \ldots, Y_m$ from FPAUS.

  - Couple each $X_i$ and $Y_i$ so that $P[X_i \neq Y_i] \leq \delta/m$.

  - Chance of deviation (using $X$s vs $Y$s): $\delta + \delta + \cdots + \delta \leq \delta$.

  - Alg’s output changes no more than $\delta$ in TV.
Suppose alg uses samples $X_1, \ldots, X_m$. Instead feed it samples $Y_1, \ldots, Y_m$ from FPAUS.
Suppose alg uses samples $X_1, \ldots, X_m$.
Instead feed it samples $Y_1, \ldots, Y_m$ from FPAUS.
Couple each $X_i$ and $Y_i$ so that $\Pr[X_i \neq Y_i] \leq \delta/m$. 

Chance of deviation (using $X$s vs $Y$s):

$\delta + \delta + \cdots + \delta \leq \delta$.

Alg’s output changes no more than $\delta$ in $d_{TV}$. 

Replacing exact samples with approx samples
Suppose alg uses samples $X_1, \ldots, X_m$.

Instead feed it samples $Y_1, \ldots, Y_m$ from FPAUS.

**Couple** each $X_i$ and $Y_i$ so that $P[X_i \neq Y_i] \leq \delta/m$.

Chance of deviation (using $X$s vs $Y$s):

$$\frac{\delta}{m} + \frac{\delta}{m} + \cdots + \frac{\delta}{m} \leq \delta.$$
Suppose alg uses samples $X_1, \ldots, X_m$. Instead feed it samples $Y_1, \ldots, Y_m$ from FPAUS. 

**Couple** each $X_i$ and $Y_i$ so that $P[X_i \neq Y_i] \leq \delta/m$. 

Chance of deviation (using Xs vs Ys): 

$$\frac{\delta}{m} + \frac{\delta}{m} + \cdots + \frac{\delta}{m} \leq \delta.$$ 

Alg’s output changes no more than $\delta$ in $d_{TV}$.😊
DNF Counting
  - Rejection sampling
  - Monte Carlo estimation

Counting vs. Sampling
  - Self-reducibility
  - Reductions
  - Total variation and coupling

Counting via Determinants
  - Spanning trees

if time
DNF Counting

- Rejection sampling
- Monte Carlo estimation

Counting vs. Sampling

- Self-reducibility
- Reductions
- Total variation and coupling

Counting via Determinants

- Spanning trees

if time
Counting spanning trees

\[
\begin{pmatrix}
 a & b & f & d \\
 e & c & & \\
 u \quad v \quad w
\end{pmatrix}
\]

vertex-edge adj matrix

Sum of rows = 0

\(n \times n\) submatrices have \(\det = 0\)

How about \((n - 1) \times (n - 1)\)?

If cycle exists, \(\det = 0\):

For some choice of signs: 

\[\pm (\text{col } a) \pm (\text{col } b) \pm (\text{col } e) = 0\]
Counting spanning trees

\begin{align*}
&\begin{array}{c}
u \\
\end{array} & \begin{array}{c}
v \\
\end{array} & \begin{array}{c}
w \\
\end{array} \\
&\begin{array}{c}
a \\
b \\
f \\
g \\
x \\
y \\
\end{array} & \begin{array}{c}
e \\
c \\
d \\
\end{array} & \\
\end{align*}

vertex-edge adj matrix

\[
\begin{bmatrix}
a & b & c & d & e & f & g \\
+1 & 0 & 0 & 0 & +1 & 0 & 0 \\
0 & -1 & +1 & 0 & -1 & -1 & 0 \\
0 & 0 & -1 & +1 & 0 & 0 & 0 \\
-1 & +1 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & -1 & 0 & +1 & +1 \\
\end{bmatrix}
\]
Counting spanning trees

\[ \begin{bmatrix}
  a & b & c & d & e & f & g \\
  u & +1 & 0 & 0 & 0 & +1 & 0 & 0 \\
  v & 0 & -1 & +1 & 0 & -1 & -1 & 0 \\
  w & 0 & 0 & -1 & +1 & 0 & 0 & 0 \\
  x & -1 & +1 & 0 & 0 & 0 & 0 & -1 \\
  y & 0 & 0 & 0 & -1 & 0 & +1 & +1 \\
\end{bmatrix} \]

vertex-edge adj matrix

Sum of rows = 0
Counting spanning trees

- Sum of rows = 0
- $n \times n$ submatrices have $\det = 0$

Vertex-edge adj matrix:

\[
\begin{bmatrix}
    +1 & 0 & 0 & 0 & +1 & 0 & 0 \\
    0 & -1 & +1 & 0 & -1 & -1 & 0 \\
    0 & 0 & -1 & +1 & 0 & 0 & 0 \\
    -1 & +1 & 0 & 0 & 0 & 0 & -1 \\
    0 & 0 & 0 & -1 & 0 & +1 & +1 \\
\end{bmatrix}
\]
Counting spanning trees

Sum of rows = 0

\( n \times n \) submatrices have \( \text{det} = 0 \)

How about \( (n - 1) \times (n - 1) \)?

For some choice of signs:

\[ \pm (\text{col}_a) \pm (\text{col}_b) \pm (\text{col}_e) = 0 \]

vertex-edge adj matrix

\[
\begin{bmatrix}
+1 & 0 & 0 & 0 & +1 & 0 & 0 \\
0 & -1 & +1 & 0 & -1 & -1 & 0 \\
0 & 0 & -1 & +1 & 0 & 0 & 0 \\
-1 & +1 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & -1 & 0 & +1 & +1 \\
\end{bmatrix}
\]
Counting spanning trees

\[ \begin{bmatrix}
  a & b & c & d & e & f & g \\
  u & +1 & 0 & 0 & 0 & +1 & 0 & 0 \\
  v & 0 & -1 & +1 & 0 & -1 & -1 & 0 \\
  w & 0 & 0 & -1 & +1 & 0 & 0 & 0 \\
  x & -1 & +1 & 0 & 0 & 0 & 0 & -1 \\
  y & 0 & 0 & 0 & -1 & 0 & +1 & +1
\end{bmatrix} \]

vertex-edge adj matrix

- Sum of rows = 0
- \( n \times n \) submatrices have \( \det = 0 \)
- How about \((n - 1) \times (n - 1)\)?
- If cycle exists, \( \det = 0 \):

For some choice of signs:

\[ \pm (\text{col } a) \pm (\text{col } b) \pm (\text{col } e) = 0 \]
Otherwise, columns are a spanning tree. In this case $\det \in \{\pm 1\}$. Sketch:

```
O----e----O
 |     |     |
 a-----b-----f-----d
 |     |     |     |
O----g----O
 |     |
 x-----y
```
Otherwise, columns are a spanning tree. In this case \( \det \in \{\pm 1\} \). Sketch:

\[
\begin{bmatrix}
\begin{array}{cccc}
a & b & c & d \\
+1 & 0 & 0 & 0 \\
0 & -1 & +1 & 0 \\
0 & 0 & -1 & +1 \\
-1 & +1 & 0 & 0
\end{array}
\end{bmatrix}
\]
Otherwise, columns are a spanning tree. In this case \( \det \in \{ \pm 1 \} \). Sketch:

\[
\begin{align*}
\text{submatrix} & \\
\begin{bmatrix}
a & b & c & d \\
u & +1 & 0 & 0 & 0 \\
v & 0 & -1 & +1 & 0 \\
w & 0 & 0 & -1 & +1 \\
x & -1 & +1 & 0 & 0
\end{bmatrix} & & \begin{bmatrix}
a & b & c & d \\
u & +1 & 0 & 0 & 0 \\
v & 0 & -1 & +1 & 0 \\
w & 0 & 0 & -1 & +1 \\
x & 0 & +1 & 0 & 0
\end{bmatrix}
\end{align*}
\]
Otherwise, columns are a spanning tree. In this case $\det \in \{\pm 1\}$. Sketch:

<table>
<thead>
<tr>
<th>Submatrix</th>
<th>Added row $u$ to $x$</th>
<th>Added row $x$ to $v$</th>
</tr>
</thead>
</table>
| \[
\begin{pmatrix}
a & b & c & d \\
u & +1 & 0 & 0 & 0 \\
v & 0 & -1 & +1 & 0 \\
w & 0 & 0 & -1 & +1 \\
x & -1 & +1 & 0 & 0
\end{pmatrix}
| \[
\begin{pmatrix}
a & b & c & d \\
u & +1 & 0 & 0 & 0 \\
v & 0 & -1 & +1 & 0 \\
w & 0 & 0 & -1 & +1 \\
x & 0 & +1 & 0 & 0
\end{pmatrix}
| \[
\begin{pmatrix}
a & b & c & d \\
u & +1 & 0 & 0 & 0 \\
v & 0 & 0 & +1 & 0 \\
w & 0 & 0 & -1 & +1 \\
x & 0 & +1 & 0 & 0
\end{pmatrix}
|
Otherwise, columns are a spanning tree. In this case $\det \in \{\pm 1\}$. Sketch:

**Submatrix**

\[
\begin{bmatrix}
a & b & c & d \\
u & +1 & 0 & 0 \\
v & 0 & -1 & +1 \\
w & 0 & 0 & -1 \\
x & -1 & +1 & 0 \\
\end{bmatrix}
\]

**Added row $u$ to $x$**

\[
\begin{bmatrix}
a & b & c & d \\
u & +1 & 0 & 0 \\
v & 0 & -1 & +1 \\
w & 0 & 0 & -1 \\
x & 0 & +1 & 0 \\
\end{bmatrix}
\]

**Added row $x$ to $v$**

\[
\begin{bmatrix}
a & b & c & d \\
u & +1 & 0 & 0 \\
v & 0 & 0 & +1 \\
w & 0 & 0 & -1 \\
x & 0 & +1 & 0 \\
\end{bmatrix}
\]

**Added row $v$ to $w$**

\[
\begin{bmatrix}
a & b & c & d \\
u & +1 & 0 & 0 \\
v & 0 & 0 & +1 \\
w & 0 & 0 & 0 \\
x & 0 & +1 & 0 \\
\end{bmatrix}
\]
Otherwise, columns are a spanning tree. In this case \( \det \in \{ \pm 1 \} \). Sketch:

\[
\begin{bmatrix}
+1 & 0 & 0 & 0 \\
0 & -1 & +1 & 0 \\
0 & 0 & -1 & +1 \\
-1 & +1 & 0 & 0
\end{bmatrix}
\]

\[
\begin{bmatrix}
+1 & 0 & 0 & 0 \\
0 & 0 & +1 & 0 \\
0 & 0 & 0 & +1 \\
0 & +1 & 0 & 0
\end{bmatrix}
\]
Determinants tell us which subsets are spanning trees ...
Determinants tell us which subsets are spanning trees …

How to sum?
Determinants tell us which subsets are spanning trees ...

How to sum?

[Cauchy-Binet]

If $A$ is $n \times m$ and $B$ is $m \times n$:

$$\det(AB) = \sum_{S \in \binom{[m]}{n}} \det(A_{\text{cols}=S}) \det(B_{\text{rows}=S}).$$
Determinants tell us which subsets are spanning trees ... How to sum?

[Cauchy-Binet]

If $A$ is $n \times m$ and $B$ is $m \times n$:

$$\det(AB) = \sum_{S \in \binom{[m]}{n}} \det(A_{\text{cols}=S}) \det(B_{\text{rows}=S}).$$

Let $A = B^T$ be vertex-edge adj matrix with one row removed.
Determinants tell us which subsets are spanning trees ...

How to sum?

[Cauchy-Binet]

If $A$ is $n \times m$ and $B$ is $m \times n$:

$$\det(AB) = \sum_{S \in \binom{[m]}{n}} \det(A_{\text{cols}=S}) \det(B_{\text{rows}=S}).$$

Let $A = B^T$ be vertex-edge adj matrix with one row removed.

We get

$$\det(AA^T) = \sum_S (\pm 1 [S \text{ spanning tree}])^2 = \# \text{spanning trees}.$$
Determinants tell us which subsets are spanning trees...

How to sum?

**[Cauchy-Binet]**

If $A$ is $n \times m$ and $B$ is $m \times n$:

$$\det(AB) = \sum_{S \in \binom{[m]}{n}} \det(A_{\text{cols}=S}) \det(B_{\text{rows}=S}).$$

Let $A = B^T$ be vertex-edge adj matrix with one row removed.

We get

$$\det(AA^T) = \sum_{S} (\pm 1 [S \text{ spanning tree}])^2 = \#\text{spanning trees}.$$

Next lecture: other determinant-based counting algs.