CS 263: Counting and Sampling

Nima Anari

Stanford University

slides for

Sampling vs. Counting

Review

Density μ on space Ω

Review

Density μ on space Ω

- Sampling: $\mathbb{P}[\text{output}] \propto \mu(\text{output})$

Review

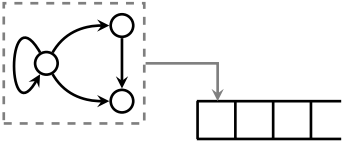
Density μ on space Ω

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

Review

Density μ on space Ω

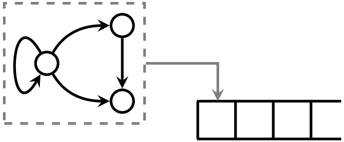
- Sampling: $\mathbb{P}[\text{output}] \propto \mu(\text{output})$
- Counting: compute $\sum_x \mu(x)$
- #P: #accepting paths in TM



Review

Density μ on space Ω

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

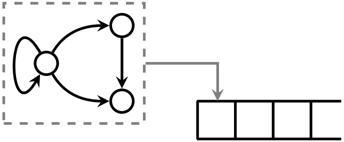


- #P-complete:
  - Natural counting variants of known NP-complete problems.
  - Natural counting variants of some P problems too!

Review

Density μ on space Ω

- 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 poly$(n, 1/\epsilon)$

FPRAS/FPTAS

randomized deterministic

Approx sampling

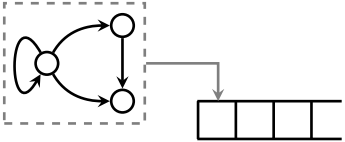
δ-approx in d_{TV} in poly$(n, \log(1/\delta))$

FPAUS

Review

Density μ on space Ω

- 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	Approx sampling
$(1+\epsilon)$-approx in poly$(n, 1/\epsilon)$	δ-approx in d_{TV} in poly$(n, \log(1/\delta))$
FPRAS/FPTAS	FPAUS

randomized (FPRAS) deterministic (FPTAS)

Self-reducibles [Jerrum-Valiant-Vazirani]:

Exact Counting → Approx Counting

Exact Counting ↓ Exact Sampling

Approx Counting → Exact Sampling (if FPTAS)

Approx Counting ↕ Approx Sampling

Exact Sampling → Approx Sampling

DNF Counting

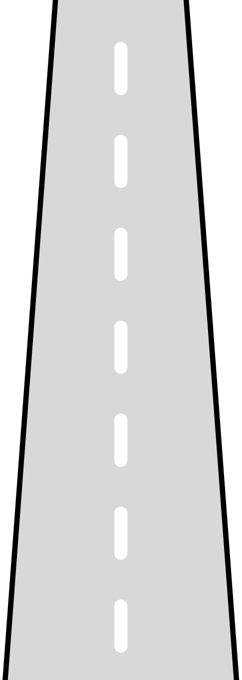
- Rejection sampling
- Monte Carlo estimation

Counting vs. Sampling

- Self-reducibility
- Reductions
- Total variation and coupling

Counting via Determinants ← if time

- Spanning trees



DNF Counting

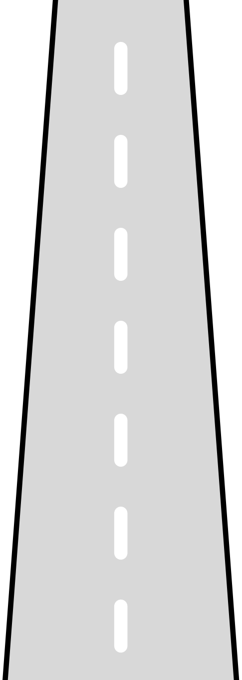
- Rejection sampling
- Monte Carlo estimation

Counting vs. Sampling

- Self-reducibility
- Reductions
- Total variation and coupling

Counting via Determinants ← if time

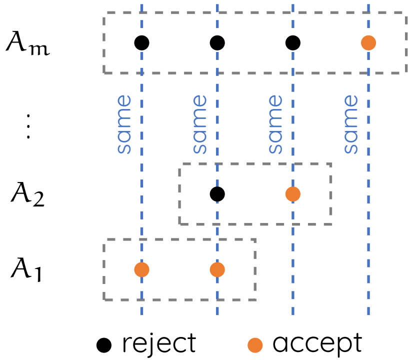
- Spanning trees



DNF sampling [Karp-Luby]

$$\phi = C_1 \vee C_2 \vee \cdots \vee C_m$$

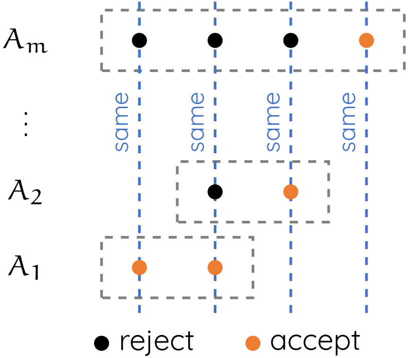
- $A_i = \{$sat assignments of $C_i\}$
- Sample u.r. $\in A_1 \sqcup \cdots \sqcup A_m$



DNF sampling [Karp-Luby]

$$\phi = C_1 \vee C_2 \vee \cdots \vee C_m$$

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



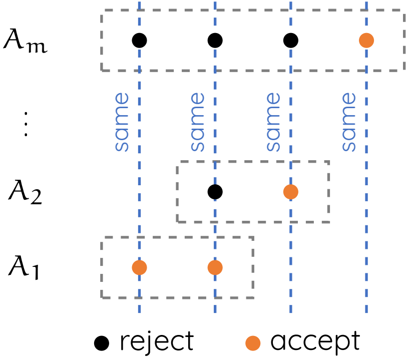
Example

$$\phi = \underbrace{x_1}_{C_1} \vee \underbrace{x_2}_{C_2}$$

DNF sampling [Karp-Luby]

$$\phi = C_1 \vee C_2 \vee \cdots \vee C_m$$

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



Example

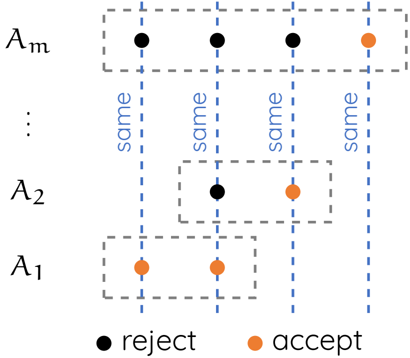
$$\phi = \underbrace{x_1}_{C_1} \vee \underbrace{x_2}_{C_2}$$

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

DNF sampling [Karp-Luby]

$$\phi = C_1 \vee C_2 \vee \cdots \vee C_m$$

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



Example

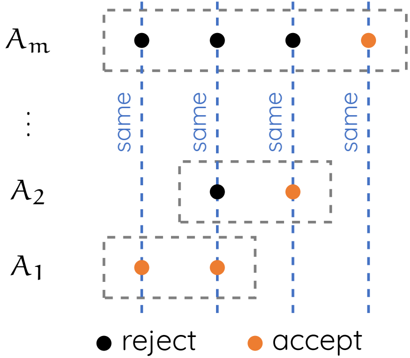
$$\phi = \underbrace{x_1}_{C_1} \vee \underbrace{x_2}_{C_2}$$

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

DNF sampling [Karp-Luby]

$$\phi = C_1 \vee C_2 \vee \cdots \vee C_m$$

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



Example

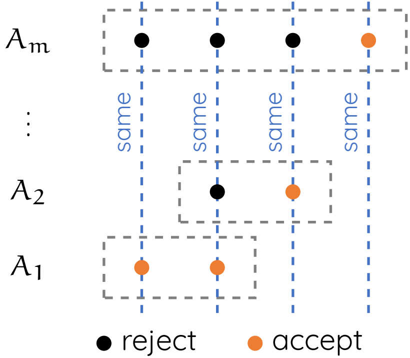
$$\phi = \underset{\substack{\uparrow \\ C_1}}{x_1} \vee \underset{\substack{\uparrow \\ C_2}}{x_2}$$

- 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

DNF sampling [Karp-Luby]

$$\phi = C_1 \vee C_2 \vee \cdots \vee C_m$$

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



Example

$$\phi = \underbrace{x_1}_{C_1} \vee \underbrace{x_2}_{C_2}$$

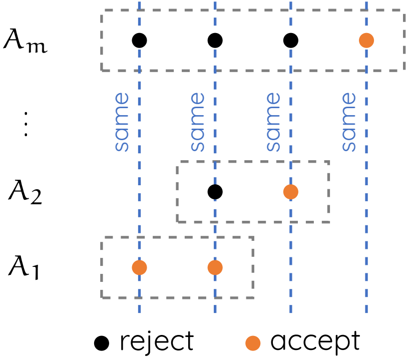
- Goal: sample u.r. from $A_1 \cup A_2 = \{10, 01, 11\}$
- $A_1 = \{10, 11\}$, $A_2 = \{01, 11\}$
- Sample u.r. from $\{10, 11, 01, 11\}$, reject the second 11

How to sample $\sim A_1 \sqcup \cdots \sqcup A_m$?

DNF sampling [Karp-Luby]

$$\phi = C_1 \vee C_2 \vee \cdots \vee C_m$$

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



Example

$$\phi = \underset{C_1}{\underset{\uparrow}{x_1}} \vee \underset{C_2}{\underset{\uparrow}{x_2}}$$

- 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

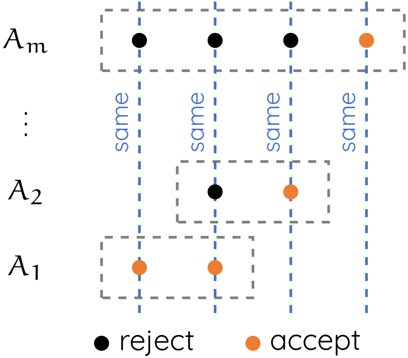
How to sample $\sim A_1 \sqcup \cdots \sqcup A_m$?

- Sample i w.p. $\propto |A_i|$

DNF sampling [Karp-Luby]

$$\phi = C_1 \vee C_2 \vee \cdots \vee C_m$$

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



Example

$$\phi = \underbrace{x_1}_{C_1} \vee \underbrace{x_2}_{C_2}$$

- 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 $\sim A_1 \sqcup \cdots \sqcup A_m$?

- Sample i w.p. $\propto |A_i|$
- Sample $x \in A_i$ u.a.r.

DNF counting [Karp-Luby]

How to count solutions?

DNF counting [Karp-Luby]

How to count solutions?

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

$$\underbrace{|A_1 \sqcup \cdots \sqcup A_m|}_{\text{easy to compute}} \cdot \underbrace{\frac{|A_1 \cup \cdots \cup A_m|}{|A_1 \sqcup \cdots \sqcup A_m|}}_{\text{accept prob}}$$

DNF counting [Karp-Luby]

How to count solutions?

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

$$|\underbrace{A_1 \sqcup \cdots \sqcup A_m}_{\text{easy to compute}}| \cdot \underbrace{\frac{|A_1 \cup \cdots \cup A_m|}{|A_1 \sqcup \cdots \sqcup A_m|}}_{\text{accept prob}}$$

- Approximate accept prob p

Monte Carlo estimation

for $i = 1, \ldots, t$ **do**
  sample $\sim A_1 \sqcup \cdots \sqcup A_m$
  and $X_i \leftarrow \mathbb{1}[\text{accept}]$

return $X = \frac{X_1 + \cdots + X_t}{t}$

DNF counting [Karp-Luby]

How to count solutions?

- Idea: Write $|A_1 \cup \dots \cup A_m|$ as

$$|A_1 \sqcup \dots \sqcup A_m| \cdot \frac{|A_1 \cup \dots \cup A_m|}{|A_1 \sqcup \dots \sqcup A_m|}$$

easy to compute   accept prob

- Approximate accept prob p

Monte Carlo estimation

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

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

DNF counting [Karp-Luby]

How to count solutions?

- Idea: Write $|A_1 \cup \dots \cup A_m|$ as

$$|A_1 \sqcup \dots \sqcup A_m| \cdot \frac{|A_1 \cup \dots \cup A_m|}{|A_1 \sqcup \dots \sqcup A_m|}$$

easy to compute   accept prob

- Approximate accept prob p

Monte Carlo estimation

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

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

DNF counting [Karp-Luby]

How to count solutions?

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

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

easy to compute   accept prob

- Approximate accept prob p

Monte Carlo estimation

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

- $\mathbb{E}[X_i] = p \quad \mathrm{Var}(X_i) = p(1-p)$
- $\mathbb{E}[X] = p \quad \mathrm{Var}(X) = p(1-p)/t$
- By Chebyshev's inequality

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

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

DNF counting [Karp-Luby]

How to count solutions?

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

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

easy to compute accept prob

- Approximate accept prob p

Monte Carlo estimation

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

- $\mathbb{E}[X_i] = p \quad \mathrm{Var}(X_i) = p(1-p)$
- $\mathbb{E}[X] = p \quad \mathrm{Var}(X) = p(1-p)/t$
- By Chebyshev's inequality

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

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

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

DNF counting [Karp-Luby]

How to count solutions?

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

$$|\underbrace{A_1 \sqcup \cdots \sqcup A_m}_{\text{easy to compute}}| \cdot \underbrace{\frac{|A_1 \cup \cdots \cup A_m|}{|A_1 \sqcup \cdots \sqcup A_m|}}_{\text{accept prob}}$$

- Approximate accept prob p

Monte Carlo estimation

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

- $\mathbb{E}[X_i] = p \quad \mathrm{Var}(X_i) = p(1-p)$
- $\mathbb{E}[X] = p \quad \mathrm{Var}(X) = p(1-p)/t$
- By Chebyshev's inequality

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

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

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

Lemma

To mult. estimate p from $\mathrm{Ber}(p)$ 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]

$\pm\epsilon 2^n$ approximation in time

$$n^{\widetilde{O}(\log \log n)}$$

DNF Counting

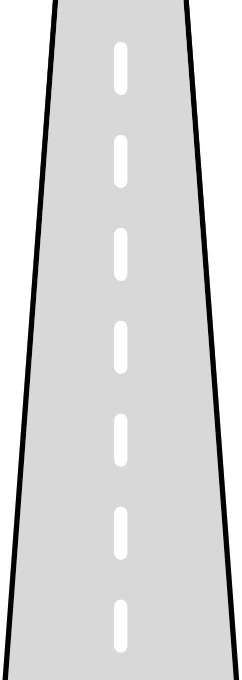
- Rejection sampling
- Monte Carlo estimation

Counting vs. Sampling

- Self-reducibility
- Reductions
- Total variation and coupling

Counting via Determinants ← if time

- Spanning trees



DNF Counting

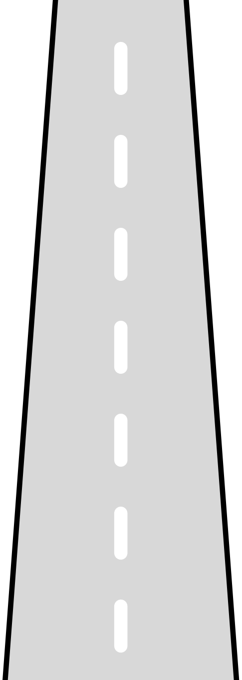
- Rejection sampling
- Monte Carlo estimation

Counting vs. Sampling

- Self-reducibility
- Reductions
- Total variation and coupling

Counting via Determinants ← if time

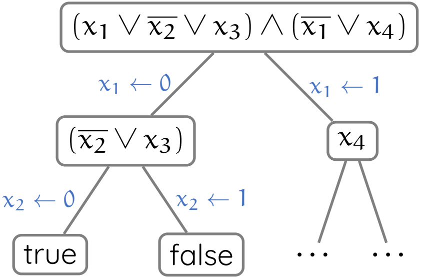
- Spanning trees



Self-reducible problems

advanced: measure-decomposed

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

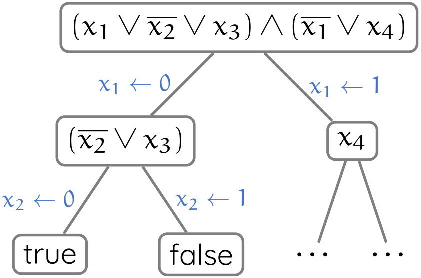


Key: branching factor, depth $\leqslant$ poly

Self-reducible problems

advanced: measure-decomposed

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



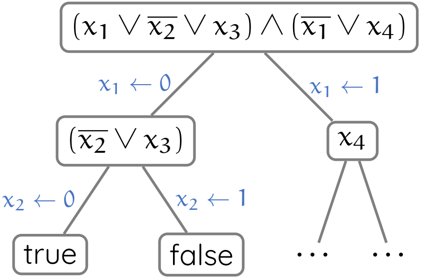
Key: branching factor, depth $\leqslant$ poly

Other requirements:

Self-reducible problems

advanced: measure-decomposed

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



Key: branching factor, depth $\leqslant$ poly

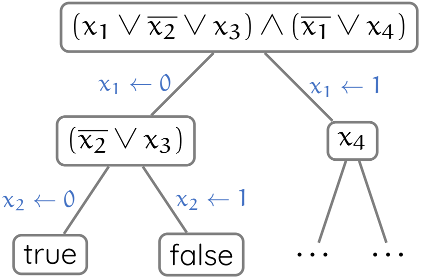
Other requirements:

- Instances I' produced efficiently.

Self-reducible problems

advanced: measure-decomposed

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



Key: branching factor, depth $\leqslant$ poly

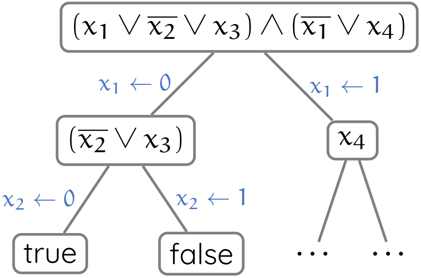
Other requirements:

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

Self-reducible problems

advanced: measure-decomposed

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



Key: branching factor, depth $\leqslant$ poly

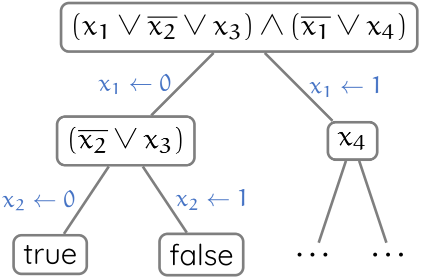
Other requirements:

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

Self-reducible problems

advanced: measure-decomposed

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

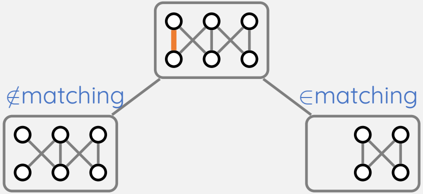


Key: branching factor, depth $\leqslant$ poly

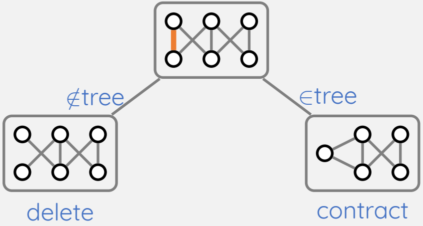
Other requirements:

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

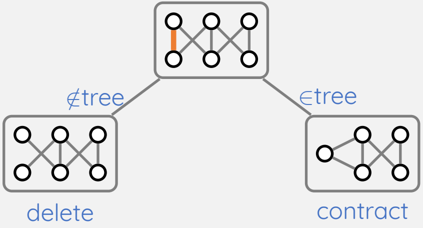
Example: perfect matchings



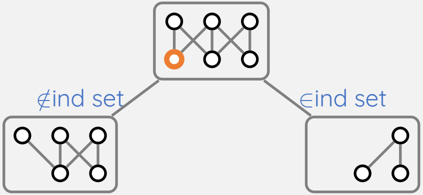
Example: spanning trees



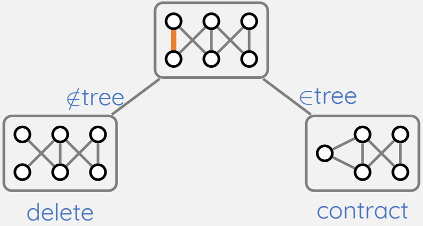
Example: spanning trees



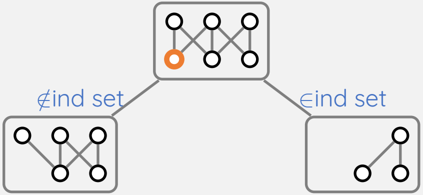
Example: independent sets



Example: spanning trees



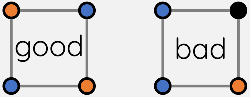
Example: independent sets



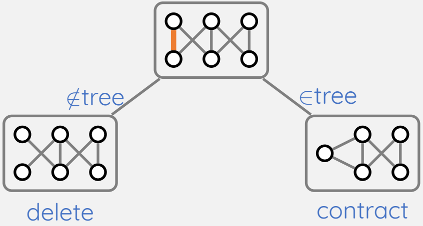
Non-example: colorings

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

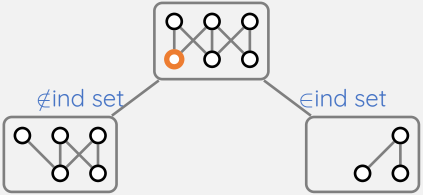
Solutions: $x \in [q]^V$ with $x_u \neq x_v$ for adjacent u, v



Example: spanning trees

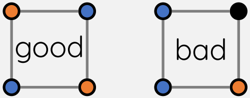


Example: independent sets

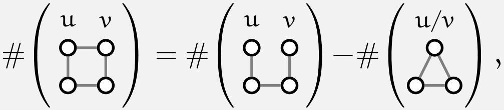


Non-example: colorings

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



but this is not self-reducibility.

Theorem [Jerrum-Valiant-Vazirani]

For "self-reducible" problems:

approx counting $\equiv$ approx sampling

$$
\begin{tikzcd}
 & (\text{FPRAS}) \\
\text{Exact Counting} \arrow[r] \arrow[d] & \text{Approx Counting} \arrow[dl, "\text{if FPTAS}"'] \arrow[d, leftrightarrow] \\
\text{Exact Sampling} \arrow[r] & \text{Approx Sampling} \\
 & (\text{FPAUS})
\end{tikzcd}
$$

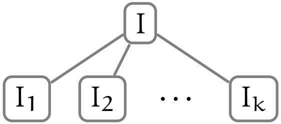
arrows are poly-time reductions

Theorem [Jerrum-Valiant-Vazirani]

For "self-reducible" problems:

approx counting $\equiv$ approx sampling

Exact Counting $\Longrightarrow$ Exact Sampling



(FPRAS)

$$
\begin{tikzcd}
\text{Exact Counting} \arrow[r] \arrow[d] & \text{Approx Counting} \arrow[d, leftrightarrow] \arrow[dl, "\text{if FPTAS}"'] \\
\text{Exact Sampling} \arrow[r] & \text{Approx Sampling}
\end{tikzcd}
$$

(FPAUS)

arrows are poly-time reductions

while I *not base case* **do**

  compute children $I_1, \dots, I_k$

  for $i = 1, \dots, k$ **do**

    $c_i \leftarrow \#(I_i)$

  choose i w.p. $\propto c_i$

  $I \leftarrow I_i$

output sample for I

$$\mathbb{P}[\text{sample}] = \frac{\#(I_i)}{\#(I)} \cdot \frac{\#(I_{ij})}{\#(I_i)} \cdots = \frac{1}{\#(I)}$$

FPTAS $\Longrightarrow$ Exact Sampling

FPTAS $\Longrightarrow$ Exact Sampling

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

FPTAS $\implies$ Exact Sampling

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

FPTAS $\implies$ Exact Sampling

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

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

FPTAS $\Longrightarrow$ Exact Sampling

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

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

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

FPTAS $\implies$ Exact Sampling

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

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

- Idea: if ν is output dist, we can compute $\nu(x)$. Rejection sample this into the target dist μ.
- 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 $\epsilon \simeq 1/\text{depth}$:

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

- Idea: if ν is output dist, we can compute $\nu(x)$. Rejection sample this into the target dist μ.
- 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 $d_{TV} \leqslant \delta$. ☺

FPTAS $\Longrightarrow$ 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 $\epsilon \simeq 1/\text{depth}$:

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

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

FPRAS $\Longrightarrow$ Approx Sampling

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

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

$$\mathbb{P}[\text{not finishing}] \leqslant \delta/2$$

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 $\epsilon \simeq 1/\text{depth}$:

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

- Idea: if ν is output dist, we can compute $\nu(x)$. Rejection sample this into the target dist μ.
- 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 $d_{TV} \leqslant \delta$. ☺

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

$$\mathbb{P}[\text{not finishing}] \leqslant \delta/2$$

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

FPTAS $\implies$ Exact Sampling

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

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

- Idea: if ν is output dist, we can compute $\nu(x)$. Rejection sample this into the target dist μ.
- 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 $d_{TV} \leqslant \delta$. ☺

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

$$\mathbb{P}[\text{not finishing}] \leqslant \delta/2$$

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

$$\leqslant \frac{\delta}{2} \cdot \frac{1}{\mathrm{poly}(n)\log(1/\delta)}$$

FPTAS $\Longrightarrow$ 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 $\epsilon \simeq 1/\text{depth}$:

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

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

FPRAS $\Longrightarrow$ Approx Sampling

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

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

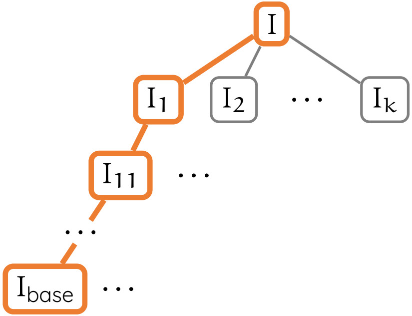
$$\mathbb{P}[\text{not finishing}] \leqslant \delta/2$$

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

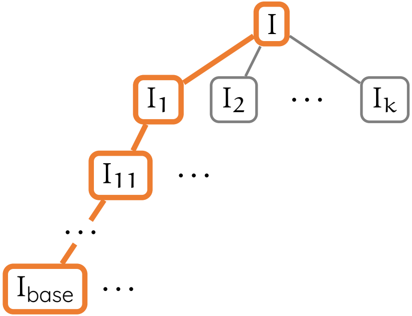
$$\leqslant \frac{\delta}{2} \cdot \frac{1}{\operatorname{poly}(n)\log(1/\delta)}$$

- Runtime: $\operatorname{poly}(n, \log(1/\delta))$ ☺

Exact Sampling $\implies$ Approx Counting

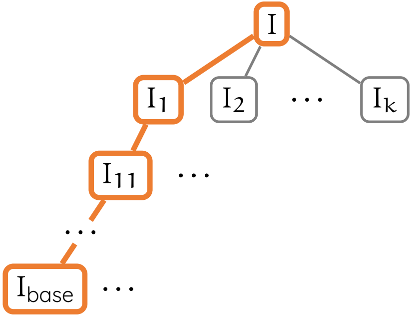


Exact Sampling $\Longrightarrow$ Approx Counting



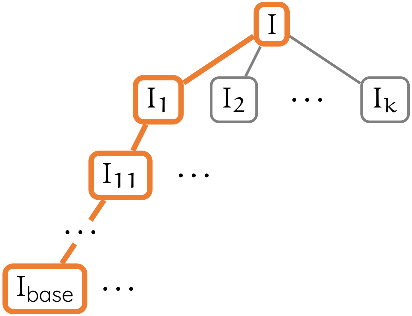
- Idea: choose root $\rightarrow$ leaf path

Exact Sampling $\Longrightarrow$ Approx Counting



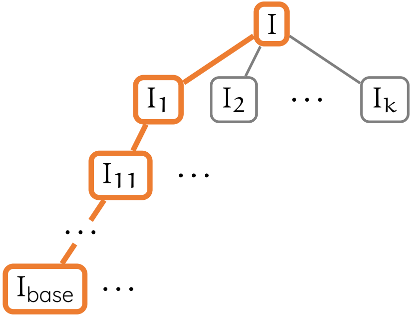
- Idea: choose root $\to$ leaf path
- Estimate $\#(I_1)/\#(I)$, $\#(I_{11})/\#(I_1)$, $\ldots$ using Monte Carlo.

Exact Sampling $\Longrightarrow$ Approx Counting



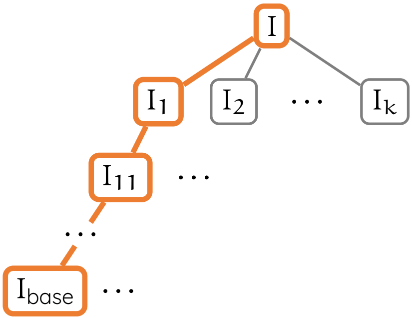
- Idea: choose root $\to$ leaf path
- Estimate $\#(I_1)/\#(I)$, $\#(I_{11})/\#(I_1)$, $\ldots$ using Monte Carlo.
- Multiply with $\#(I_{\mathrm{base}})$ and output.

Exact Sampling $\Longrightarrow$ Approx Counting



- Idea: choose root $\to$ leaf path
- Estimate $\#(I_1)/\#(I)$, $\#(I_{11})/\#(I_1)$, $\ldots$ using Monte Carlo.
- Multiply with $\#(I_{base})$ and output.
- Need $1 + \epsilon/(2 \cdot \text{depth})$ approx for each ratio.

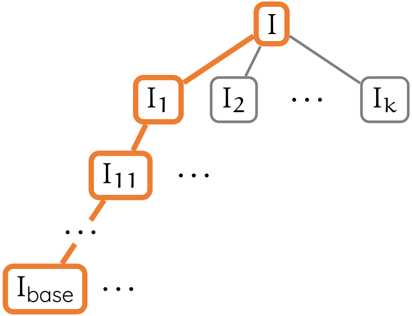
Exact Sampling $\implies$ Approx Counting



- Idea: choose root $\to$ 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 $\leqslant 1/(6 \cdot \text{depth})$.

Exact Sampling $\Longrightarrow$ Approx Counting

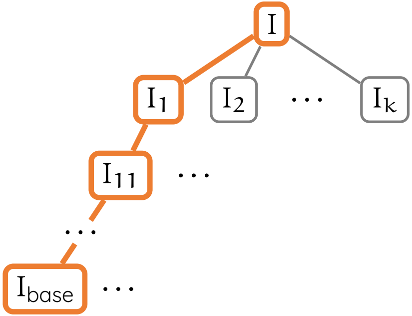


- Idea: choose root $\to$ leaf path
- Estimate $\#(\mathrm{I}_1)/\#(\mathrm{I})$, $\#(\mathrm{I}_{11})/\#(\mathrm{I}_1)$, $\ldots$ using Monte Carlo.
- Multiply with $\#(\mathrm{I}_{\text{base}})$ and output.

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

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

Exact Sampling $\implies$ Approx Counting



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

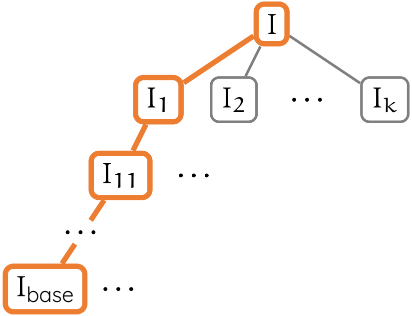
- Need $1 + \epsilon/(2 \cdot \text{depth})$ approx for each ratio.
- Set failure prob for each estimation task to $\leqslant 1/(6 \cdot \text{depth})$.
- Approx factor: 🙂

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

- Success prob: 🙂

$$\geqslant 1 - \text{depth} \cdot \frac{1}{6 \cdot \text{depth}} \geqslant \frac{5}{6}$$

Exact Sampling $\implies$ Approx Counting



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

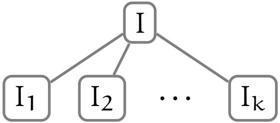
- Need $1+\epsilon/(2\cdot\text{depth})$ approx for each ratio.
- Set failure prob for each estimation task to $\leqslant 1/(6\cdot\text{depth})$.
- Approx factor: 🙂

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

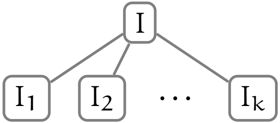
- Success prob: 🙂

$$\geqslant 1-\text{depth}\cdot\frac{1}{6\cdot\text{depth}} \geqslant \frac{5}{6}$$

- Problem: if any ratio p is small, it takes $\geqslant 1/p$ time to estimate.

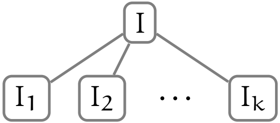


- Fix: while $\#(I_1)/\#(I)$ could be small, $\exists i$ s.t. $\#(I_i)/\#(I)$ is large.



- 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

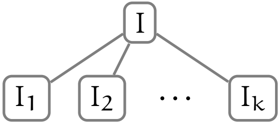
$$\frac{\#(I_i)}{\#(I)} \geqslant \frac{1}{6k \cdot \text{depth}}$$



- 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

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

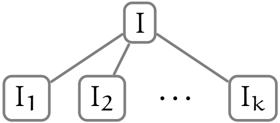
- Branch into I_i and recursively find the root $\to$ leaf path.



- 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

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

- Branch into I_i and recursively find the root $\to$ leaf path.
- Prob of wrong assumption: $\leqslant 1/6$

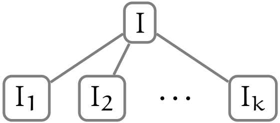


Approx Sampling $\implies$ Approx Counting

- 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

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

- Branch into I_i and recursively find the root $\to$ leaf path.
- Prob of wrong assumption: $\leqslant 1/6$



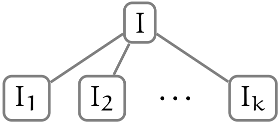
- Fix: while $\#(I_1)/\#(I)$ could be small, $\exists i$ s.t. $\#(I_i)/\#(I)$ is large.
- Take a sample x and see which I_i it belongs to. Assume

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

- Branch into I_i and recursively find the root $\rightarrow$ leaf path.
- Prob of wrong assumption: $\leqslant 1/6$

Approx Sampling $\Longrightarrow$ Approx Counting

- We have a poly-time randomized algorithm that uses samples.



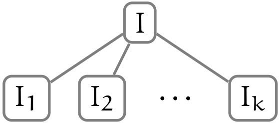
- Fix: while $\#(I_1)/\#(I)$ could be small, $\exists i$ s.t. $\#(I_i)/\#(I)$ is large.
- Take a sample x and see which I_i it belongs to. Assume

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

- Branch into I_i and recursively find the root $\to$ leaf path.
- Prob of wrong assumption: $\leqslant 1/6$

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.



- 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

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

- Branch into I_i and recursively find the root $\to$ leaf path.
- Prob of wrong assumption: $\leqslant 1/6$

Approx Sampling $\Longrightarrow$ Approx Counting

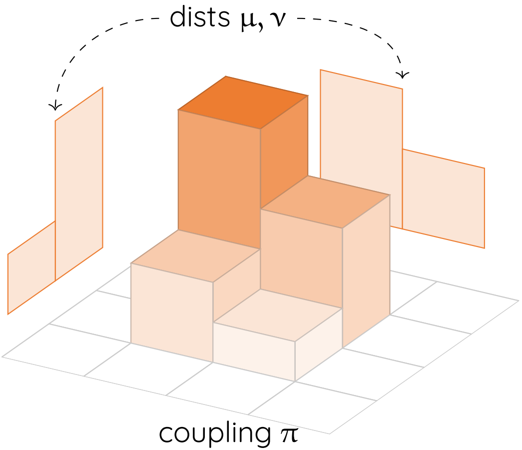
- We have a poly-time randomized algorithm that uses samples.
- In general in such algorithms, exact samplers can be replaced by approx samplers.

Lemma

In a randomized poly-time algorithm, exact samplers can be replaced by FPAUS while guaranteeing the output changes no more than δ in d_{TV} at the cost of $\mathbf{poly}(n, \mathbf{log}(1/\delta))$ in runtime.

Coupling

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



Coupling

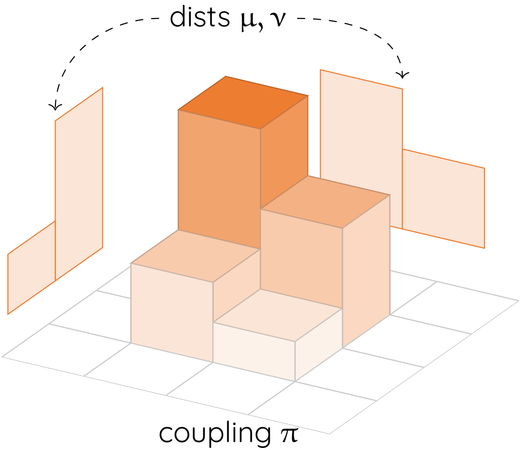
For dists μ, ν, a coupling is a joint dist π of (X, Y) where $X \sim \mu$ and $Y \sim \nu$.

Theorem

The minimum

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

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



Coupling

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

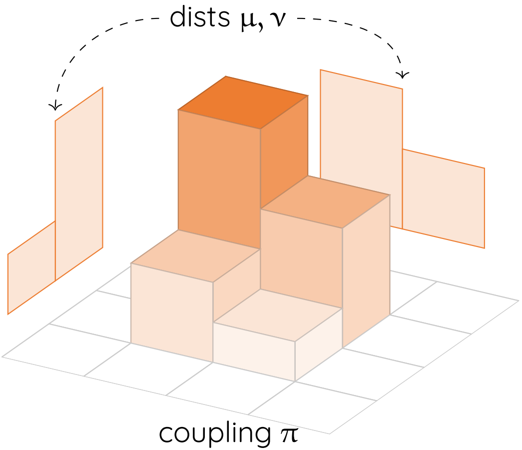
Theorem

The minimum

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

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

- Proof: exercise!



Coupling

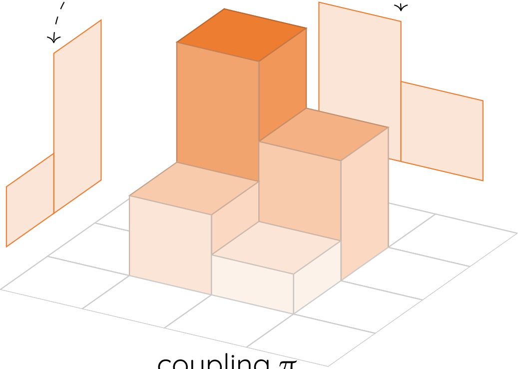
For dists μ, ν, a coupling is a joint dist π of (X, Y) where $X \sim \mu$ and $Y \sim \nu$.

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 $X_1, \ldots, X_m$.

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.

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 $\mathbb{P}[X_i \neq Y_i] \leqslant \delta/m$.

Replacing exact samples with approx samples

- Suppose alg uses samples $X_1, \dots, X_m$.
- Instead feed it samples $Y_1, \dots, Y_m$ from FPAUS.
- Couple each X_i and Y_i so that $\mathbb{P}[X_i \neq Y_i] \leqslant \delta/m$.
- Chance of deviation (using Xs vs Ys):

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

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 $\mathbb{P}[X_i \neq Y_i] \leqslant \delta/m$.
- Chance of deviation (using Xs vs Ys):

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

- Alg's output changes no more than δ in d_{TV}. ☺

DNF Counting

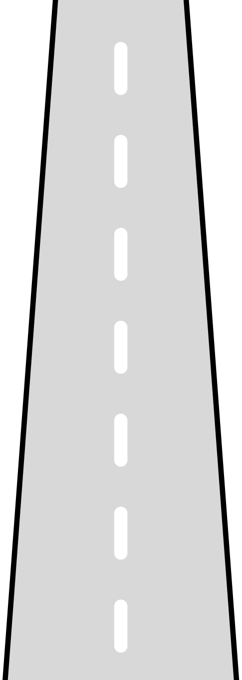
- Rejection sampling
- Monte Carlo estimation

Counting vs. Sampling

- Self-reducibility
- Reductions
- Total variation and coupling

Counting via Determinants ← if time

- Spanning trees



DNF Counting

- Rejection sampling
- Monte Carlo estimation

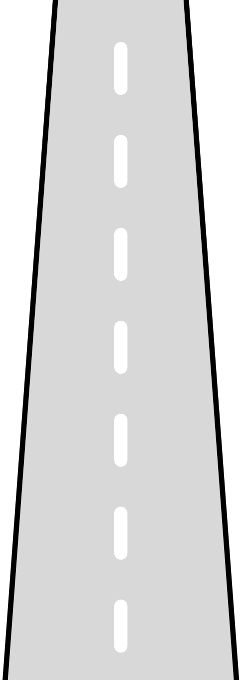
Counting vs. Sampling

- Self-reducibility
- Reductions
- Total variation and coupling

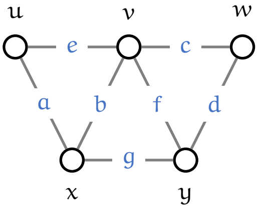
Counting via Determinants

if time

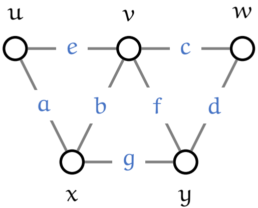
- Spanning trees



Counting spanning trees



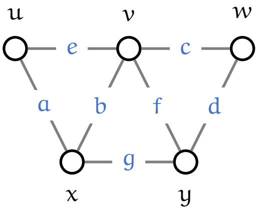
Counting spanning trees



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

vertex-edge adj matrix

Counting spanning trees

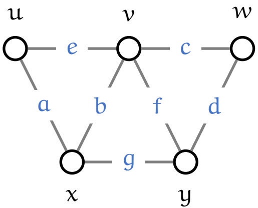


$$
\begin{array}{c|c}
 & \begin{array}{ccccccc} a & b & c & d & e & f & g \end{array} \\
\begin{array}{c} u \\ v \\ w \\ x \\ y \end{array} &
\begin{bmatrix}
+1 & 0 & 0 & 0 & +1 & 0 & 0 \\
0 & -1 & +1 & 0 & -1 & -1 & 0 \\
0 & 0 & -1 & +1 & 0 & 0 & 0 \\
-1 & +1 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & -1 & 0 & +1 & +1
\end{bmatrix}
\end{array}
$$

vertex-edge adj matrix

- Sum of rows $= 0$

Counting spanning trees

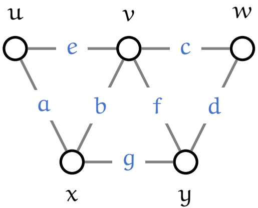


$$
\begin{array}{c|ccccccc|}
 & a & b & c & d & e & f & g \\
\hline
u & +1 & 0 & 0 & 0 & +1 & 0 & 0 \\
v & 0 & -1 & +1 & 0 & -1 & -1 & 0 \\
w & 0 & 0 & -1 & +1 & 0 & 0 & 0 \\
x & -1 & +1 & 0 & 0 & 0 & 0 & -1 \\
y & 0 & 0 & 0 & -1 & 0 & +1 & +1
\end{array}
$$

vertex-edge adj matrix

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

Counting spanning trees

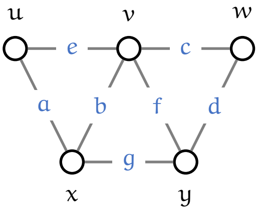


$$\begin{array}{c@{}c}
 & \begin{array}{ccccccc} a & b & c & d & e & f & g \end{array} \\
\begin{array}{c} u \\ v \\ w \\ x \\ y \end{array} &
\begin{bmatrix}
+1 & 0 & 0 & 0 & +1 & 0 & 0 \\
0 & -1 & +1 & 0 & -1 & -1 & 0 \\
0 & 0 & -1 & +1 & 0 & 0 & 0 \\
-1 & +1 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & -1 & 0 & +1 & +1
\end{bmatrix}
\end{array}$$

vertex-edge adj matrix

- Sum of rows $= 0$
- $n \times n$ submatrices have $\det = 0$
- How about $(n-1) \times (n-1)$?

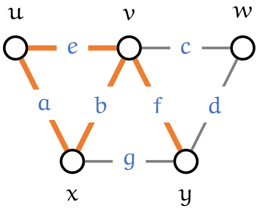
Counting spanning trees



$$\begin{array}{cc} & \begin{matrix} \;\;a & \;\;b & \;\;c & \;\;d & \;\;e & \;\;f & \;\;g \end{matrix} \\ \begin{matrix} u \\ v \\ w \\ x \\ y \end{matrix} & \begin{bmatrix} +1 & 0 & 0 & 0 & +1 & 0 & 0 \\ 0 & -1 & +1 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & +1 & 0 & 0 & 0 \\ -1 & +1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & +1 & +1 \end{bmatrix} \end{array}$$

vertex-edge adj matrix

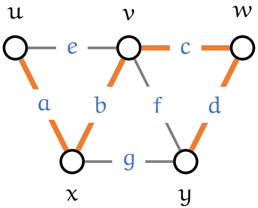
- Sum of rows $= 0$
- $n \times n$ submatrices have **det** $= 0$
- How about $(n-1) \times (n-1)$?
- If cycle exists, **det** $= 0$:



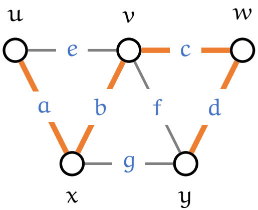
For some choice of signs:

$$\pm(\text{col } a) \pm (\text{col } b) \pm (\text{col } e) = 0$$

Otherwise, columns are a spanning tree. In this case $\mathbf{det} \in \{\pm 1\}$. Sketch:



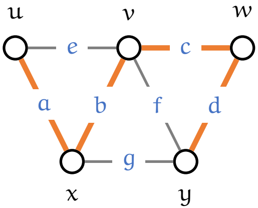
Otherwise, columns are a spanning tree. In this case **det** $\in \{\pm 1\}$. Sketch:



submatrix

$$\begin{array}{c c} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} u \\ v \\ w \\ x \end{array} & \begin{bmatrix} +1 & 0 & 0 & 0 \\ 0 & -1 & +1 & 0 \\ 0 & 0 & -1 & +1 \\ -1 & +1 & 0 & 0 \end{bmatrix} \end{array}$$

Otherwise, columns are a spanning tree. In this case $\mathbf{det} \in \{\pm 1\}$. Sketch:



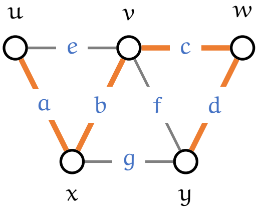
submatrix

$$\begin{array}{c c} & \begin{matrix} a & b & c & d \end{matrix} \\ \begin{matrix} u \\ v \\ w \\ x \end{matrix} & \begin{bmatrix} +1 & 0 & 0 & 0 \\ 0 & -1 & +1 & 0 \\ 0 & 0 & -1 & +1 \\ -1 & +1 & 0 & 0 \end{bmatrix} \end{array}$$

added row u to x

$$\begin{array}{c c} & \begin{matrix} a & b & c & d \end{matrix} \\ \begin{matrix} u \\ v \\ w \\ x \end{matrix} & \begin{bmatrix} +1 & 0 & 0 & 0 \\ 0 & -1 & +1 & 0 \\ 0 & 0 & -1 & +1 \\ 0 & +1 & 0 & 0 \end{bmatrix} \end{array}$$

Otherwise, columns are a spanning tree. In this case $\mathbf{det} \in \{\pm 1\}$. Sketch:



submatrix

$$\begin{array}{c|cccc|} & a & b & c & d \\ \hline u & +1 & 0 & 0 & 0 \\ v & 0 & -1 & +1 & 0 \\ w & 0 & 0 & -1 & +1 \\ x & -1 & +1 & 0 & 0 \end{array}$$

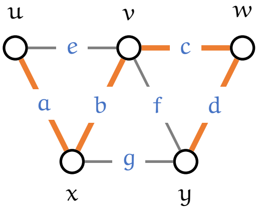
added row u to x

$$\begin{array}{c|cccc|} & a & b & c & d \\ \hline u & +1 & 0 & 0 & 0 \\ v & 0 & -1 & +1 & 0 \\ w & 0 & 0 & -1 & +1 \\ x & 0 & +1 & 0 & 0 \end{array}$$

added row x to v

$$\begin{array}{c|cccc|} & a & b & c & d \\ \hline u & +1 & 0 & 0 & 0 \\ v & 0 & 0 & +1 & 0 \\ w & 0 & 0 & -1 & +1 \\ x & 0 & +1 & 0 & 0 \end{array}$$

Otherwise, columns are a spanning tree. In this case **det** $\in \{\pm 1\}$. Sketch:



submatrix

$$\begin{array}{c} \\ u \\ v \\ w \\ x \end{array}\begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} +1 & 0 & 0 & 0 \\ 0 & -1 & +1 & 0 \\ 0 & 0 & -1 & +1 \\ -1 & +1 & 0 & 0 \end{bmatrix} \end{array}$$

added row u to x

$$\begin{array}{c} \\ u \\ v \\ w \\ x \end{array}\begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} +1 & 0 & 0 & 0 \\ 0 & -1 & +1 & 0 \\ 0 & 0 & -1 & +1 \\ 0 & +1 & 0 & 0 \end{bmatrix} \end{array}$$

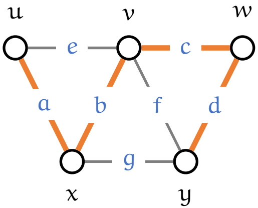
added row x to v

$$\begin{array}{c} \\ u \\ v \\ w \\ x \end{array}\begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} +1 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 \\ 0 & 0 & -1 & +1 \\ 0 & +1 & 0 & 0 \end{bmatrix} \end{array}$$

added row v to w

$$\begin{array}{c} \\ u \\ v \\ w \\ x \end{array}\begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} +1 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & +1 \\ 0 & +1 & 0 & 0 \end{bmatrix} \end{array}$$

Otherwise, columns are a spanning tree. In this case **det** $\in \{\pm 1\}$. Sketch:



submatrix

$$\begin{array}{c} \\ u \\ v \\ w \\ x \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} +1 & 0 & 0 & 0 \\ 0 & -1 & +1 & 0 \\ 0 & 0 & -1 & +1 \\ -1 & +1 & 0 & 0 \end{bmatrix} \end{array}$$

added row u to x

$$\begin{array}{c} \\ u \\ v \\ w \\ x \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} +1 & 0 & 0 & 0 \\ 0 & -1 & +1 & 0 \\ 0 & 0 & -1 & +1 \\ 0 & +1 & 0 & 0 \end{bmatrix} \end{array}$$

added row x to v

$$\begin{array}{c} \\ u \\ v \\ w \\ x \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} +1 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 \\ 0 & 0 & -1 & +1 \\ 0 & +1 & 0 & 0 \end{bmatrix} \end{array}$$

added row v to w

$$\begin{array}{c} \\ u \\ v \\ w \\ x \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} +1 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & +1 \\ 0 & +1 & 0 & 0 \end{bmatrix} \end{array}$$

permuted and fixed signs

$$\begin{array}{c} \\ u \\ x \\ v \\ w \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} +1 & 0 & 0 & 0 \\ 0 & +1 & 0 & 0 \\ 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & +1 \end{bmatrix} \end{array}$$

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

arbitrary

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

  arbitrary
- We get

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

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

  arbitrary
- We get

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

- Next lecture: other determinant-based counting algs.