CS 263: Counting and Sampling

Nima Anari

Stanford University

slides for

Det Counting; Markov Chains

Review

- DNF counting:

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

Review

- DNF counting:

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

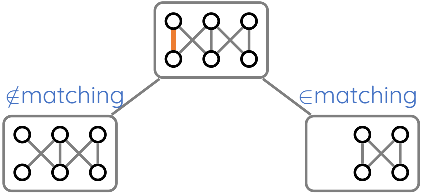
- Monte Carlo: estimate p from $\mathrm{Ber}(p)$. Need $\simeq 1/p\epsilon^2$ many.

Review

- DNF counting:

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

- Monte Carlo: estimate p from $\mathrm{Ber}(p)$. Need $\simeq 1/p\epsilon^2$ many.
- Self-reducible problems:

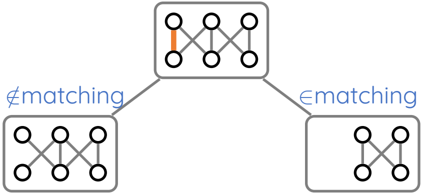


Review

- DNF counting:

$$\underset{\substack{\uparrow\\ \text{easy to compute}}}{|A_1 \sqcup \cdots \sqcup A_m|} \cdot \underset{\substack{\uparrow\\ \text{probability}}}{\frac{|A_1 \cup \cdots \cup A_m|}{|A_1 \sqcup \cdots \sqcup A_m|}}$$

- Monte Carlo: estimate p from $\mathrm{Ber}(p)$. Need $\simeq 1/p\epsilon^2$ many.
- Self-reducible problems:



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

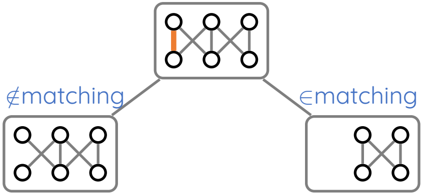
Review

- DNF counting:

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

easy to compute   probability

- Monte Carlo: estimate p from $\mathrm{Ber}(p)$. Need $\simeq 1/p\epsilon^2$ many.
- Self-reducible problems:



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

- Coupling: dist with marginals μ, ν.

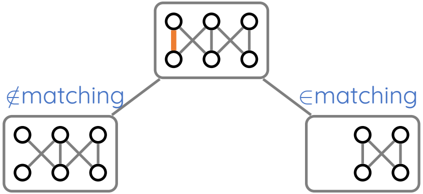
Review

- DNF counting:

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

easy to compute   probability

- Monte Carlo: estimate p from $\mathrm{Ber}(p)$. Need $\simeq 1/p\epsilon^2$ many.
- Self-reducible problems:



(FPRAS)

Exact Counting ⟶ Approx Counting

Exact Sampling ⟶ Approx Sampling

if FPTAS

(FPAUS)

- Coupling: dist with marginals μ, ν.
- Matrix-tree theorem [Kirchhoff]:

$$\#\text{spanning trees} = \det(\text{matrix})$$

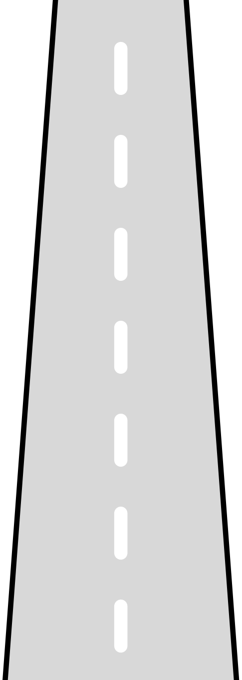
Laplacian, drop one row+col

Counting via Determinants

- Spanning trees
- Bipartite planar perfect matchings

Intro to Markov Chains

- Stationary distribution
- Fundamental theorem
- Mixing time

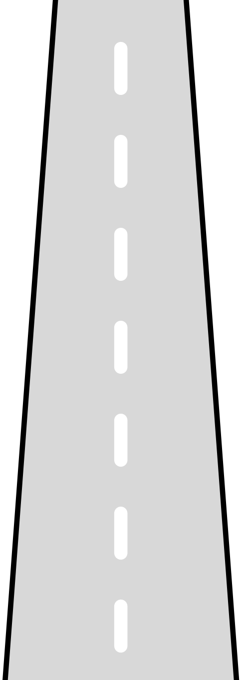


Counting via Determinants

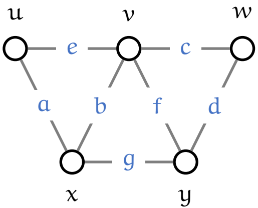
- Spanning trees
- Bipartite planar perfect matchings

Intro to Markov Chains

- Stationary distribution
- Fundamental theorem
- Mixing time



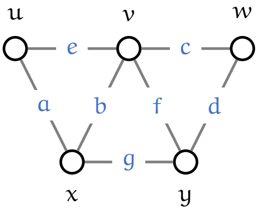
Counting spanning trees



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

vertex-edge adj matrix A

Counting spanning trees



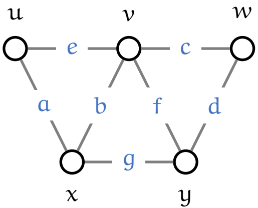
$$\begin{array}{c} \\ u \\ v \\ w \\ x \\ y \end{array} \begin{array}{c} \begin{array}{ccccccc} a & b & c & d & e & f & g \end{array} \\ \begin{bmatrix} +1 & 0 & 0 & 0 & +1 & 0 & 0 \\ 0 & -1 & +1 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & +1 & 0 & 0 & 0 \\ -1 & +1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 &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

vertex-edge adj matrix A

If we take $AA^\intercal$, we get the Laplacian:

$$(AA^\intercal)_{ij} = \begin{cases} -\mathbb{1}[i \sim j] & \text{if } i \neq j, \\ \deg(i) & \text{if } i = j. \end{cases}$$

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 A

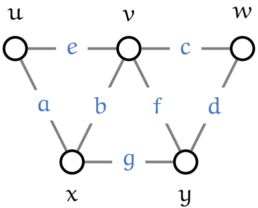
If we take $AA^\intercal$, we get the Laplacian:

$$(AA^\intercal)_{ij} = \begin{cases} -\mathbb{1}[i \sim j] & \text{if } i \neq j, \\ \deg(i) & \text{if } i = j. \end{cases}$$

Matrix-tree theorem [Kirchhoff]

det of $(n-1) \times (n-1)$ principal submatrix of Laplacian is #spanning trees.

Counting spanning trees



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

vertex-edge adj matrix A

If we take $AA^\intercal$, we get the Laplacian:

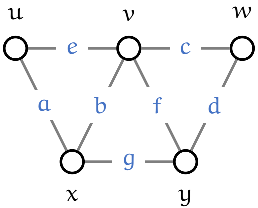
$$(AA^\intercal)_{ij} = \begin{cases} -\mathbb{1}[i \sim j] & \text{if } i \neq j, \\ \deg(i) & \text{if } i = j. \end{cases}$$

Matrix-tree theorem [Kirchhoff]

det of $(n-1) \times (n-1)$ principal submatrix of Laplacian is #spanning trees.

- Directed graphs: exercise!

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 A

If we take $AA^{\intercal}$, we get the Laplacian:

$$
(AA^{\intercal})_{ij} = \begin{cases} -\mathbb{1}[i \sim j] & \text{if } i \neq j, \\ \deg(i) & \text{if } i = j. \end{cases}
$$

Matrix-tree theorem [Kirchhoff]

det of $(n-1) \times (n-1)$ principal submatrix of Laplacian is #spanning trees.

- Directed graphs: exercise!
- Counting $\implies$ sampling.

- Runtime for counting: $O(n^{\omega})$

matrix multiplication exponent $\omega \simeq 2.37$

- Runtime for counting: $O(n^{\omega})$

  matrix multiplication exponent $\omega \simeq 2.37$

- Runtime for sampling:
  - Naïve: $m \times$ counting $= O(mn^{\omega})$
  - Smarter [Colbourn-Myrvold-Neufeld'96]: $\widetilde{O}(n^{\omega})$

- Runtime for counting: $O(n^\omega)$

  matrix multiplication exponent $\omega \simeq 2.37$

- Runtime for sampling:
  - Naïve: $m \times \text{counting} = O(mn^\omega)$
  - Smarter [Colbourn-Myrvold-Neufeld'96]: $\widetilde{O}(n^\omega)$
- Best-known (approx) counting

  [Chu-Gao-Peng-Sachdeva-Sawlani-Wang'18]:

  $\simeq m^{1+o(1)} + n^{15/8+o(1)}$

- Runtime for counting: $O(n^{\omega})$

  matrix multiplication exponent $\omega \simeq 2.37$

- Runtime for sampling:
  - Naïve: $m \times$ counting $= O(mn^{\omega})$
  - Smarter [Colbourn-Myrvold-Neufeld'96]: $\widetilde{O}(n^{\omega})$
- Best-known (approx) counting [Chu-Gao-Peng-Sachdeva-Sawlani-Wang'18]: $\simeq m^{1+o(1)} + n^{15/8+o(1)}$
- Best-known (approx) sampling [A-Liu-OveisGharan-Vinzant-Vuong'20]: $\widetilde{O}(m)$

- Runtime for counting: $O(n^{\omega})$

  matrix multiplication exponent $\omega \simeq 2.37$

- Runtime for sampling:
  - Naïve: $m \times$ counting $= O(mn^{\omega})$
  - Smarter [Colbourn-Myrvold-Neufeld'96]: $\widetilde{O}(n^{\omega})$
- Best-known (approx) counting [Chu-Gao-Peng-Sachdeva-Sawlani-Wang'18]: $\simeq m^{1+o(1)} + n^{15/8+o(1)}$
- Best-known (approx) sampling [A-Liu-OveisGharan-Vinzant-Vuong'20]: $\widetilde{O}(m)$
- Open problem: improve counting.

- Runtime for counting: $O(n^\omega)$

  matrix multiplication exponent $\omega \simeq 2.37$
- Runtime for sampling:
  - Naïve: $m \times$ counting $= O(mn^\omega)$
  - Smarter [Colbourn-Myrvold-Neufeld'96]: $\widetilde{O}(n^\omega)$
- Best-known (approx) counting [Chu-Gao-Peng-Sachdeva-Sawlani-Wang'18]: $\simeq m^{1+o(1)} + n^{15/8+o(1)}$
- Best-known (approx) sampling [A-Liu-OveisGharan-Vinzant-Vuong'20]: $\widetilde{O}(m)$
- Open problem: improve counting.
- Open problem: speedups in directed graphs?

Bipartite perfect matchings

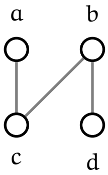
Bipartite PMs is #P-complete [Valiant].

- Count approximately ← later
- Restrict graphs ← today

Bipartite perfect matchings

Bipartite PMs is #P-complete [Valiant].

- Count approximately ⟵ later
- Restrict graphs ⟵ today



$$\begin{array}{c} \\ a \\ b \end{array}\begin{array}{c} \begin{array}{cc} c & d \end{array} \\ \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \end{array}$$

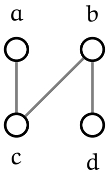
Bipartite perfect matchings

Bipartite PMs is #P-complete [Valiant].

- Count approximately $\longleftarrow$ later
- Restrict graphs $\longleftarrow$ today

- Permanent:

$$\sum_\sigma A_{1\sigma(1)} \cdots A_{n\sigma(n)}$$



$$\begin{array}{cc} & \begin{array}{cc} c & d \end{array} \\ \begin{array}{c} a \\ b \end{array} & \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \end{array}$$

Bipartite perfect matchings

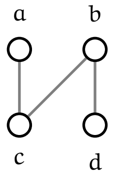
Bipartite PMs is #P-complete [Valiant].

- Count approximately ⟵ later
- Restrict graphs ⟵ today

- Permanent:

  $\sum_\sigma A_{1\sigma(1)} \cdots A_{n\sigma(n)}$

  per(bipartite adj) $=$ #PMs



$$\begin{array}{c} \\ a \\ b \end{array}\begin{array}{c} \begin{array}{cc} c & d \end{array} \\ \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \end{array}$$

Bipartite perfect matchings

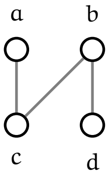
Bipartite PMs is #P-complete [Valiant].

- Count approximately ⟵ later
- Restrict graphs ⟵ today

- Permanent:

$$\sum_\sigma A_{1\sigma(1)} \cdots A_{n\sigma(n)}$$

per(bipartite adj) $=$ #PMs



- Determinant:

$$\sum_\sigma \mathrm{sign}(\sigma) A_{1\sigma(1)} \cdots A_{n\sigma(n)}$$

$$\begin{array}{cc} & \begin{array}{cc} c & d \end{array} \\ \begin{array}{c} a \\ b \end{array} & \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \end{array}$$

Bipartite perfect matchings

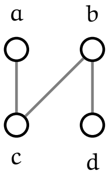
Bipartite PMs is #P-complete [Valiant].

- Count approximately ⟵ later
- Restrict graphs ⟵ today

- Permanent:

$$\sum_\sigma A_{1\sigma(1)} \cdots A_{n\sigma(n)}$$

per(bipartite adj) $=$ #PMs



- Determinant:

$$\sum_\sigma \mathrm{sign}(\sigma) A_{1\sigma(1)} \cdots A_{n\sigma(n)}$$

$$\begin{array}{cc} & \begin{array}{cc} c & d \end{array} \\ \begin{array}{c} a \\ b \end{array} & \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \end{array}$$

[Pólya]'s scheme: replace 1s with ± 1s to make all terms in sum equal-signed.

Bipartite perfect matchings

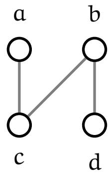
Bipartite PMs is #P-complete [Valiant].

- Count approximately ⟵ later
- Restrict graphs ⟵ today

- Permanent:

$$\sum_\sigma A_{1\sigma(1)} \cdots A_{n\sigma(n)}$$

per(bipartite adj) = #PMs



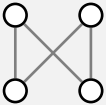
- Determinant:

$$\sum_\sigma \mathrm{sign}(\sigma) A_{1\sigma(1)} \cdots A_{n\sigma(n)}$$

$$\begin{array}{cc} & \begin{array}{cc} c & d \end{array} \\ \begin{array}{c} a \\ b \end{array} & \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \end{array}$$

[Pólya]'s scheme: replace 1s with ±1s to make all terms in sum equal-signed.

Example: $K_{2,2}$



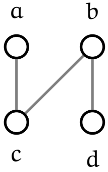
$$\det\left(\begin{bmatrix} +1 & -1 \\ +1 & +1 \end{bmatrix}\right) = \mathrm{per}\left(\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}\right)$$

Bipartite perfect matchings

Bipartite PMs is #P-complete [Valiant].

- Count approximately ⟵ later
- Restrict graphs ⟵ today
- Permanent:

$$\sum_\sigma A_{1\sigma(1)} \cdots A_{n\sigma(n)}$$



per(bipartite adj) = #PMs

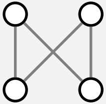
- Determinant:

$$\sum_\sigma \mathrm{sign}(\sigma) A_{1\sigma(1)} \cdots A_{n\sigma(n)}$$

$$\begin{array}{c c} & \begin{array}{cc} c & d \end{array} \\ \begin{array}{c} a \\ b \end{array} & \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \end{array}$$

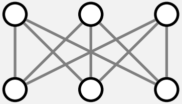
[Pólya]'s scheme: replace 1s with ± 1s to make all terms in sum equal-signed.

Example: $K_{2,2}$



$$\det\left(\begin{bmatrix} +1 & -1 \\ +1 & +1 \end{bmatrix}\right) = \mathrm{per}\left(\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}\right)$$

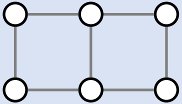
Non-example: $K_{3,3}$



Impossible! Exercise: show this.

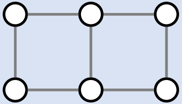
Theorem [Fisher-Kasteleyn-Temperley]

If graph is planar, [Pólya]'s scheme can be implemented. E.g., 2D lattice:



Theorem [Fisher-Kasteleyn-Temperley]

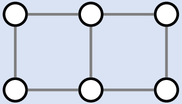
If graph is planar, [Pólya]'s scheme can be implemented. E.g., 2D lattice:



- Goal: find signing where all terms in det equal.

Theorem [Fisher-Kasteleyn-Temperley]

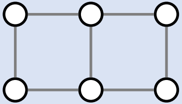
If graph is planar, [Pólya]'s scheme can be implemented. E.g., 2D lattice:



- Goal: find signing where all terms in $\det$ equal.
- Strategy: compare "neighboring" terms. Make sure equal.

Theorem [Fisher-Kasteleyn-Temperley]

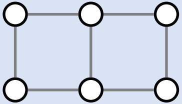
If graph is planar, [Pólya]'s scheme can be implemented. E.g., 2D lattice:

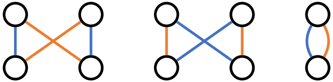


- Goal: find signing where all terms in $\det$ equal.
- Strategy: compare "neighboring" terms. Make sure equal.
- Neighboring: PMs that differ in one cycle.

Theorem [Fisher-Kasteleyn-Temperley]

If graph is planar, [Pólya]'s scheme can be implemented. E.g., 2D lattice:

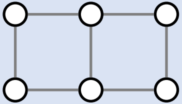




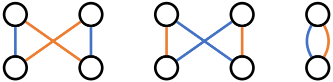
- Goal: find signing where all terms in **det** equal.
- Strategy: compare “neighboring” terms. Make sure equal.
- Neighboring: PMs that differ in one cycle.

Theorem [Fisher-Kasteleyn-Temperley]

If graph is planar, [Pólya]'s scheme can be implemented. E.g., 2D lattice:



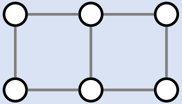
- Goal: find signing where all terms in **det** equal.
- Strategy: compare "neighboring" terms. Make sure equal.
- Neighboring: PMs that differ in one cycle.



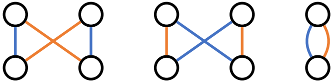
To move from orange PM to blue PM:

Theorem [Fisher-Kasteleyn-Temperley]

If graph is planar, [Pólya]'s scheme can be implemented. E.g., 2D lattice:

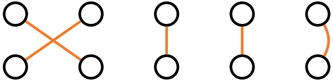


- Goal: find signing where all terms in **det** equal.
- Strategy: compare "neighboring" terms. Make sure equal.
- Neighboring: PMs that differ in one cycle.



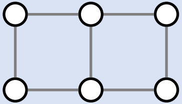
To move from orange PM to blue PM:

1

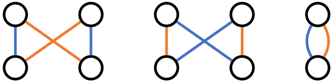


Theorem [Fisher-Kasteleyn-Temperley]

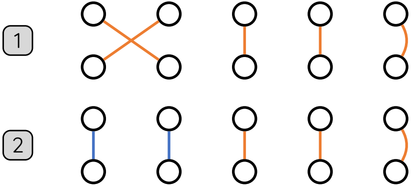
If graph is planar, [Pólya]'s scheme can be implemented. E.g., 2D lattice:



- Goal: find signing where all terms in **det** equal.
- Strategy: compare "neighboring" terms. Make sure equal.
- Neighboring: PMs that differ in one cycle.

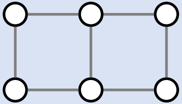


To move from orange PM to blue PM:

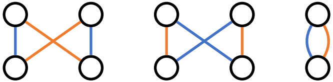


Theorem [Fisher-Kasteleyn-Temperley]

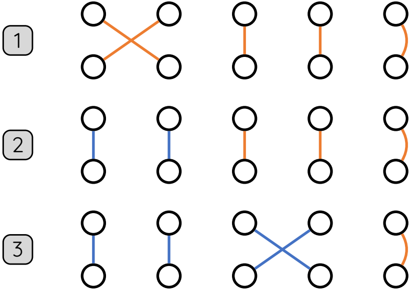
If graph is planar, [Pólya]'s scheme can be implemented. E.g., 2D lattice:



- Goal: find signing where all terms in **det** equal.
- Strategy: compare "neighboring" terms. Make sure equal.
- Neighboring: PMs that differ in one cycle.

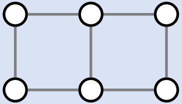


To move from orange PM to blue PM:

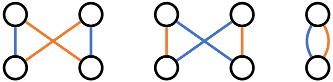


Theorem [Fisher-Kasteleyn-Temperley]

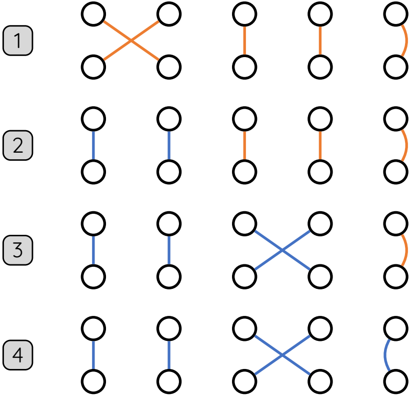
If graph is planar, [Pólya]'s scheme can be implemented. E.g., 2D lattice:



- Goal: find signing where all terms in **det** equal.
- Strategy: compare "neighboring" terms. Make sure equal.
- Neighboring: PMs that differ in one cycle.



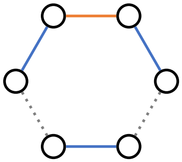
To move from orange PM to blue PM:



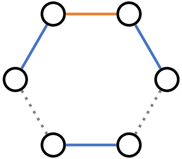
- Can move from any PM to any PM one cycle at a time.

- Can move from any PM to any PM one cycle at a time.
- Nice cycle: a cycle whose vertex-complement has a PM.

- Can move from any PM to any PM one cycle at a time.
- Nice cycle: a cycle whose vertex-complement has a PM.
- Goal: signing where nice cycles don't change term's sign.

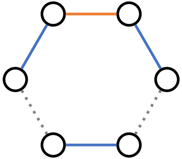


- Can move from any PM to any PM one cycle at a time.
- Nice cycle: a cycle whose vertex-complement has a PM.
- Goal: signing where nice cycles don't change term's sign.



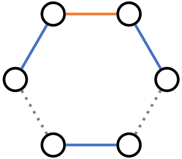
- Term is $\mathsf{sign}(\sigma)A_{1\sigma(1)} \cdots A_{n\sigma(n)}$.

- Can move from any PM to any PM one cycle at a time.
- Nice cycle: a cycle whose vertex-complement has a PM.
- Goal: signing where nice cycles don't change term's sign.



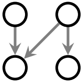
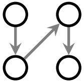
- Term is $\mathrm{sign}(\sigma)A_{1\sigma(1)}\cdots A_{n\sigma(n)}$.
- $\mathrm{sign}(\sigma)$ changes by $(-1)^{\mathrm{len}/2+1}$.

- Can move from any PM to any PM one cycle at a time.
- Nice cycle: a cycle whose vertex-complement has a PM.
- Goal: signing where nice cycles don't change term's sign.

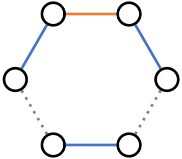


- Term is $\mathrm{sign}(\sigma)A_{1\sigma(1)}\cdots A_{n\sigma(n)}$.
- $\mathrm{sign}(\sigma)$ changes by $(-1)^{\mathrm{len}/2+1}$.

- Represent signing by orientation.
- Orient edges from one side to other. This is all $+1$ signing.

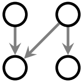
 $\begin{bmatrix} + & 0 \\ + & + \end{bmatrix}$  $\begin{bmatrix} + & 0 \\ - & + \end{bmatrix}$

- Can move from any PM to any PM one cycle at a time.
- Nice cycle: a cycle whose vertex-complement has a PM.
- Goal: signing where nice cycles don't change term's sign.

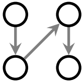


- Term is $\mathrm{sign}(\sigma) A_{1\sigma(1)} \cdots A_{n\sigma(n)}$.
- $\mathrm{sign}(\sigma)$ changes by $(-1)^{\mathrm{len}/2+1}$.

- Represent signing by orientation.
- Orient edges from one side to other. This is all $+1$ signing.



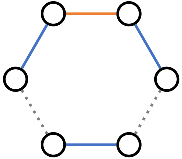
$\begin{bmatrix} + & 0 \\ + & + \end{bmatrix}$



$\begin{bmatrix} + & 0 \\ - & + \end{bmatrix}$

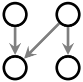
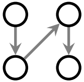
- For any cycle, #cw edges: len/2.

- Can move from any PM to any PM one cycle at a time.
- Nice cycle: a cycle whose vertex-complement has a PM.
- Goal: signing where nice cycles don't change term's sign.



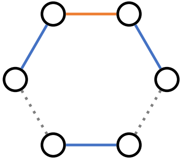
- Term is $\text{sign}(\sigma)A_{1\sigma(1)}\cdots A_{n\sigma(n)}$.
- $\text{sign}(\sigma)$ changes by $(-1)^{\text{len}/2+1}$.

- Represent signing by orientation.
- Orient edges from one side to other. This is all $+1$ signing.

 $\begin{bmatrix} + & 0 \\ + & + \end{bmatrix}$  $\begin{bmatrix} + & 0 \\ - & + \end{bmatrix}$

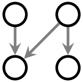
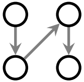
- For any cycle, #cw edges: len/2.
- Pfaffian orientation: flip some directions so that in each nice cycle, #cw edges is odd.

- Can move from any PM to any PM one cycle at a time.
- Nice cycle: a cycle whose vertex-complement has a PM.
- Goal: signing where nice cycles don't change term's sign.



- Term is $\mathsf{sign}(\sigma)A_{1\sigma(1)}\cdots A_{n\sigma(n)}$.
- $\mathsf{sign}(\sigma)$ changes by $(-1)^{\mathsf{len}/2+1}$.

- Represent signing by orientation.
- Orient edges from one side to other. This is all $+1$ signing.

 $\begin{bmatrix} + & 0 \\ + & + \end{bmatrix}$  $\begin{bmatrix} + & 0 \\ - & + \end{bmatrix}$

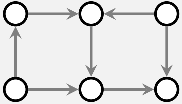
- For any cycle, #cw edges: len/2.
- Pfaffian orientation: flip some directions so that in each nice cycle, #cw edges is odd.
- This means: 🙂

$$\prod_{e\in \mathsf{cycle}} A_e = (-1)^{\mathsf{len}/2+1}$$

- Find Pfaffian orientation: nice cycles have odd #cw edges.

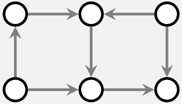
- Find Pfaffian orientation: nice cycles have odd #cw edges.

Example: lattice



- Find Pfaffian orientation: nice cycles have odd #cw edges.

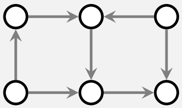
Example: lattice



1. Lemma: if all faces have odd #cw edges, so do all nice cycles.

- Find Pfaffian orientation: nice cycles have odd #cw edges.

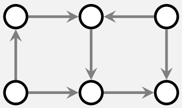
Example: lattice



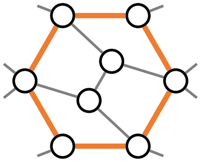
1. Lemma: if all faces have odd #cw edges, so do all nice cycles.
2. Lemma: we can find orientation with odd #cw edges per face.

- Find Pfaffian orientation: nice cycles have odd #cw edges.

Example: lattice

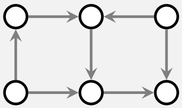


1. Lemma: if all faces have odd #cw edges, so do all nice cycles.
2. Lemma: we can find orientation with odd #cw edges per face.

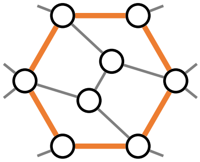


- Find Pfaffian orientation: nice cycles have odd #cw edges.

Example: lattice



1. Lemma: if all faces have odd #cw edges, so do all nice cycles.
2. Lemma: we can find orientation with odd #cw edges per face.

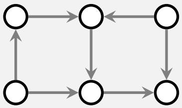


- Modulo 2, $\#(\text{cw around cycle})$ is

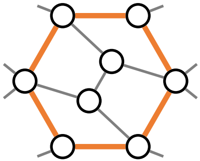
$$\equiv \sum_{\text{int face } f} \#(\text{cw around } f) + \#(\text{int edges})$$

- Find Pfaffian orientation: nice cycles have odd #cw edges.

Example: lattice



1. Lemma: if all faces have odd #cw edges, so do all nice cycles.
2. Lemma: we can find orientation with odd #cw edges per face.



- Modulo 2, #(cw around cycle) is

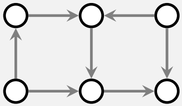
$$\equiv \sum_{\text{int face } f} \#(\text{cw around } f) + \#(\text{int edges})$$

- By Euler's formula $\#\text{verts} + \#\text{faces} - \#\text{edges} = 1$, so

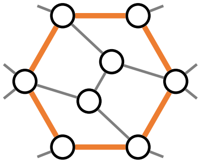
$$\#(\text{int faces}) + \#(\text{int edges}) \equiv \#(\text{int verts}) + 1$$

- Find Pfaffian orientation: nice cycles have odd #cw edges.

Example: lattice



1. Lemma: if all faces have odd #cw edges, so do all nice cycles.
2. Lemma: we can find orientation with odd #cw edges per face.



- Modulo 2, #(cw around cycle) is

$$\equiv \sum_{\text{int face f}} \#(\text{cw around f}) + \#(\text{int edges})$$

- By Euler's formula $\#\text{verts} + \#\text{faces} - \#\text{edges} = 1$, so

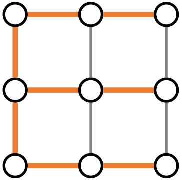
$$\#(\text{int faces}) + \#(\text{int edges}) \equiv \#(\text{int verts}) + 1$$

- Because of niceness, there are even many interior vertices. ☺

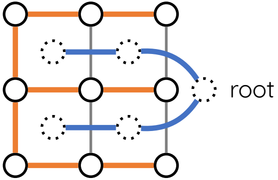
- How to make faces happy?

- How to make faces happy?
- Choose spanning tree

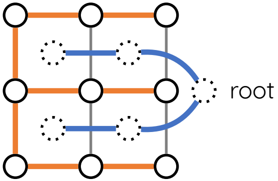
- How to make faces happy?
- Choose spanning tree
- Comes with dual spanning tree



- How to make faces happy?
- Choose spanning tree
- Comes with dual spanning tree

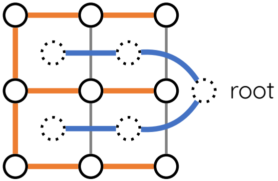


- How to make faces happy?
- Choose spanning tree
- Comes with dual spanning tree



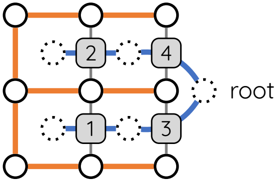
- Orient spanning tree arbitrarily.

- How to make faces happy?
- Choose spanning tree
- Comes with dual spanning tree



- Orient spanning tree arbitrarily.
- Peel leaves of dual spanning tree one-by-one. Each time, orient the single remaining edge of peeled face uniquely.

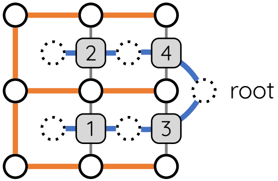
- How to make faces happy?
- Choose spanning tree
- Comes with dual spanning tree



- Orient spanning tree arbitrarily.
- Peel leaves of dual spanning tree one-by-one. Each time, orient the single remaining edge of peeled face uniquely.

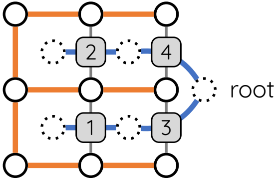
Summary: counting via dets

- How to make faces happy?
- Choose spanning tree
- Comes with dual spanning tree



- Orient spanning tree arbitrarily.
- Peel leaves of dual spanning tree one-by-one. Each time, orient the single remaining edge of peeled face uniquely.

- How to make faces happy?
- Choose spanning tree
- Comes with dual spanning tree

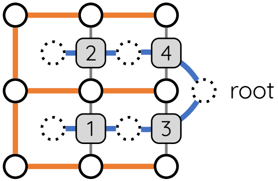


- Orient spanning tree arbitrarily.
- Peel leaves of dual spanning tree one-by-one. Each time, orient the single remaining edge of peeled face uniquely.

Summary: counting via dets

- Spanning trees:
  - Undirected: [Kirchhoff]'s matrix-tree theorem.
  - Directed: exercise!

- How to make faces happy?
- Choose spanning tree
- Comes with dual spanning tree

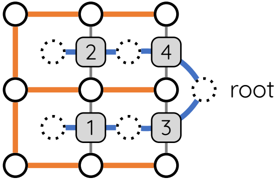


- Orient spanning tree arbitrarily.
- Peel leaves of dual spanning tree one-by-one. Each time, orient the single remaining edge of peeled face uniquely.

Summary: counting via dets

- Spanning trees:
  - Undirected: [Kirchhoff]'s matrix-tree theorem.
  - Directed: exercise!
- Planar perfect matchings:
  - Bipartite: [Fisher-Kasteleyn-Temperley]'s Pfaffian orientation.
  - Non-bipartite: exercise!

- How to make faces happy?
- Choose spanning tree
- Comes with dual spanning tree

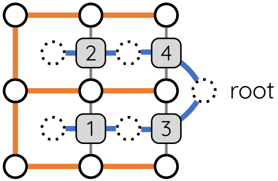


- Orient spanning tree arbitrarily.
- Peel leaves of dual spanning tree one-by-one. Each time, orient the single remaining edge of peeled face uniquely.

Summary: counting via dets

- Spanning trees:
  - Undirected: [Kirchhoff]'s matrix-tree theorem.
  - Directed: exercise!
- Planar perfect matchings:
  - Bipartite: [Fisher-Kasteleyn-Temperley]'s Pfaffian orientation.
  - Non-bipartite: exercise!
- Holographic reductions [Valiant]

- How to make faces happy?
- Choose spanning tree
- Comes with dual spanning tree

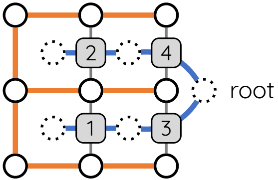


- Orient spanning tree arbitrarily.
- Peel leaves of dual spanning tree one-by-one. Each time, orient the single remaining edge of peeled face uniquely.

Summary: counting via dets

- Spanning trees:
  - Undirected: [Kirchhoff]'s matrix-tree theorem.
  - Directed: exercise!
- Planar perfect matchings:
  - Bipartite: [Fisher-Kasteleyn-Temperley]'s Pfaffian orientation.
  - Non-bipartite: exercise!
- Holographic reductions [Valiant]
- Eulerian tours: exercise!

- How to make faces happy?
- Choose spanning tree
- Comes with dual spanning tree



- Orient spanning tree arbitrarily.
- Peel leaves of dual spanning tree one-by-one. Each time, orient the single remaining edge of peeled face uniquely.

Summary: counting via dets

- Spanning trees:
  - Undirected: [Kirchhoff]'s matrix-tree theorem.
  - Directed: exercise!
- Planar perfect matchings:
  - Bipartite: [Fisher-Kasteleyn-Temperley]'s Pfaffian orientation.
  - Non-bipartite: exercise!
- Holographic reductions [Valiant]
- Eulerian tours: exercise!
- Determinantal point processes

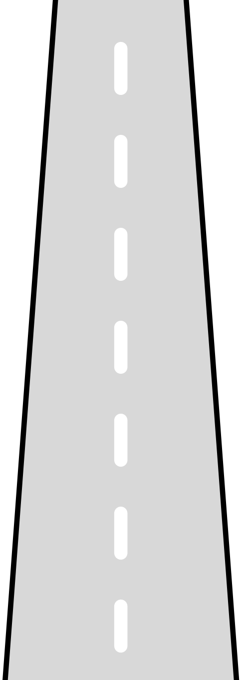
will see later

Counting via Determinants

- Spanning trees
- Bipartite planar perfect matchings

Intro to Markov Chains

- Stationary distribution
- Fundamental theorem
- Mixing time

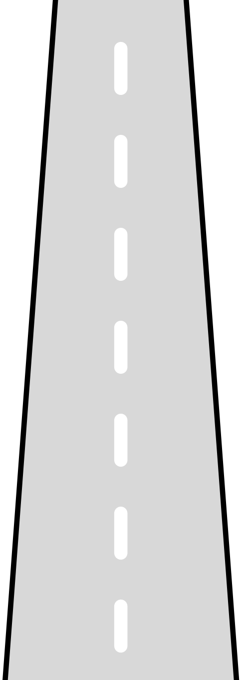


Counting via Determinants

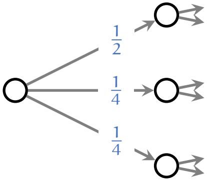
- Spanning trees
- Bipartite planar perfect matchings

Intro to Markov Chains

- Stationary distribution
- Fundamental theorem
- Mixing time



Markov chains

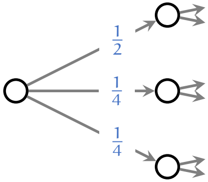


space Ω

Transition matrix: $P \in \mathbb{R}_{\geqslant 0}^{\Omega \times \Omega}$

large and implicit

Markov chains



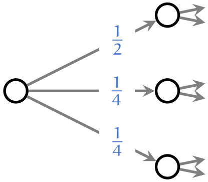
space Ω

Transition matrix: $P \in \mathbb{R}_{\geqslant 0}^{\Omega \times \Omega}$

large and implicit

- $P(x, y)$ is chance of going to y if we start from x

Markov chains



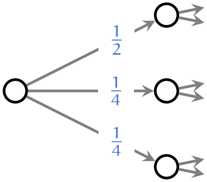
space Ω

Transition matrix: $P \in \mathbb{R}_{\geqslant 0}^{\Omega \times \Omega}$

large and implicit

- $P(x, y)$ is chance of going to y if we start from x
- $\sum_y P(x, y) = 1 \longleftarrow$ row-stochastic

Markov chains



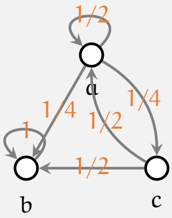
space Ω

Transition matrix: $P \in \mathbb{R}_{\geqslant 0}^{\Omega \times \Omega}$

large and implicit

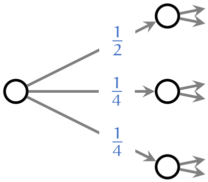
- $P(x, y)$ is chance of going to y if we start from x
- $\sum_y P(x, y) = 1 \longleftarrow$ row-stochastic

Example



$$\begin{array}{c|ccc} & a & b & c \\ \hline a & \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\ b & 0 & 1 & 0 \\ c & \frac{1}{2} & \frac{1}{2} & 0 \end{array}$$

Markov chains



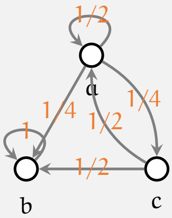
space Ω

Transition matrix: $P \in \mathbb{R}_{\geqslant 0}^{\Omega \times \Omega}$

large and implicit

- $P(x, y)$ is chance of going to y if we start from x
- $\sum_y P(x, y) = 1$ ← row-stochastic

Example



$$\begin{array}{cc} & \begin{array}{ccc} a & b & c \end{array} \\ \begin{array}{c} a \\ b \\ c \end{array} & \begin{bmatrix} \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\ 0 & 1 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix} \end{array}$$

- Given (random) start X_0, we get Markovian process:

$$X_0 \to X_1 \to X_2 \to \dots$$

transition via P　　transition via P

Fundamental theorem

Under “mild conditions”:

$$\mathrm{dist}(X_t) \to \mu$$

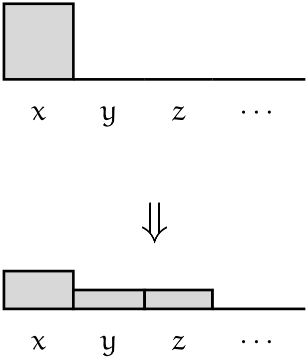
where μ is the stationary dist.

Fundamental theorem

Under “mild conditions”:

$$\mathrm{dist}(X_t) \to \mu$$

where μ is the stationary dist.

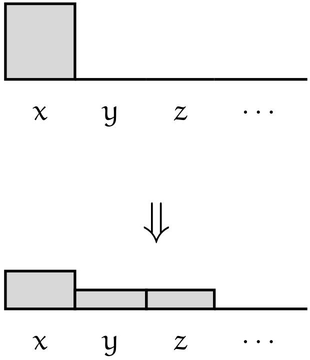


Fundamental theorem

Under "mild conditions":

$$\mathrm{dist}(X_t) \to \mu$$

where μ is the stationary dist.



- Suppose $X_0 \sim \nu$, then $X_1 \sim \nu P$

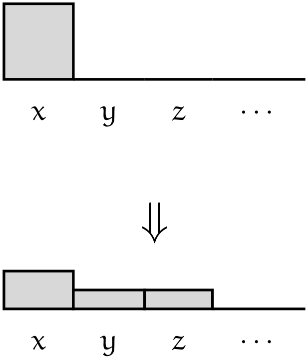
row vector transition matrix

Fundamental theorem

Under "mild conditions":

$$\text{dist}(X_t) \to \mu$$

where μ is the stationary dist.



- Suppose $X_0 \sim \nu$, then $X_1 \sim \nu P$

  row vector transition matrix

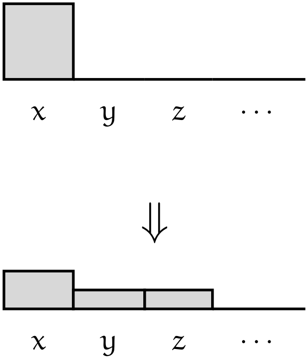
- Stationary dist: if $\mu P = \mu$, then μ is called a stationary dist.

Fundamental theorem

Under "mild conditions":

$$\mathrm{dist}(X_t) \to \mu$$

where μ is the stationary dist.



- Suppose $X_0 \sim \nu$, then $X_1 \sim \nu P$

  row vector transition matrix

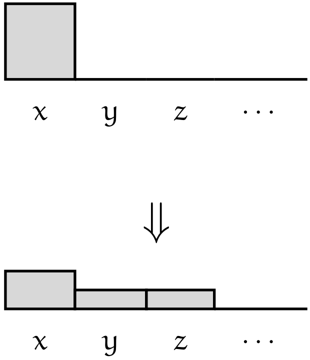
- Stationary dist: if $\mu P = \mu$, then μ is called a stationary dist.
- Note: if there is any limit, it must be stationary!

Fundamental theorem

Under "mild conditions":

$$\mathrm{dist}(X_t) \to \mu$$

where μ is the stationary dist.



- Suppose $X_0 \sim \nu$, then $X_1 \sim \nu P$

  row vector transition matrix

- Stationary dist: if $\mu P = \mu$, then μ is called a stationary dist.
- Note: if there is any limit, it must be stationary!
- Sampling via Markov chains:
  - Steps are easy ← easy
  - Correct stationary μ ← easy
  - Convergence to μ is fast

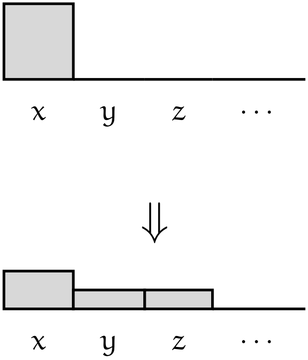
    hard

Fundamental theorem

Under "mild conditions":

$$\text{dist}(X_t) \to \mu$$

where μ is the stationary dist.



- Suppose $X_0 \sim \nu$, then $X_1 \sim \nu P$

  row vector transition matrix

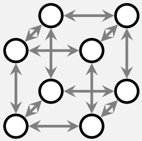
- Stationary dist: if $\mu P = \mu$, then μ is called a stationary dist.
- Note: if there is any limit, it must be stationary!
- Sampling via Markov chains:
  - Steps are easy ← easy
  - Correct stationary μ ← easy
  - Convergence to μ is fast

    hard

- Ideally, we want to stop at small t and have small d_{TV} to μ.

Example: hypercube

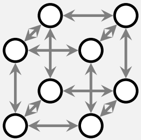
- $\Omega = \{0,1\}^n$
- Pick u.r. $i \in [n]$
- Replace coord i with $\mathrm{Ber}(\frac{1}{2})$



stationary: uniform

Example: hypercube

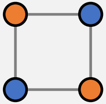
- $\Omega = \{0,1\}^n$
- Pick u.r. $i \in [n]$
- Replace coord i with Ber$(\frac{1}{2})$



stationary: uniform

Example: coloring

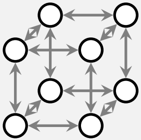
- $\Omega =$ valid colorings
- Pick u.r. vert v
- Replace v's color u.r. with valid color



stationary: uniform

Example: hypercube

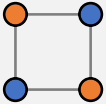
- $\Omega = \{0,1\}^n$
- Pick u.r. $i \in [n]$
- Replace coord i with Ber$(\frac{1}{2})$



stationary: uniform

Example: coloring

- $\Omega =$ valid colorings
- Pick u.r. vert v
- Replace v's color u.r. with valid color

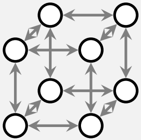


stationary: uniform

- Irreducible: possible to reach from every x to every y.

Example: hypercube

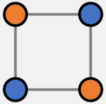
- $\Omega = \{0,1\}^n$
- Pick u.r. $i \in [n]$
- Replace coord i with Ber$(\frac{1}{2})$



stationary: uniform

Example: coloring

- $\Omega =$ valid colorings
- Pick u.r. vert v
- Replace v's color u.r. with valid color

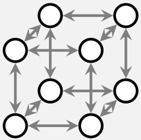


stationary: uniform

- Irreducible: possible to reach from every x to every y.
- Aperiodic: length of cycles from x to x have $\gcd = 1$.

Example: hypercube

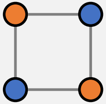
- $\Omega = \{0,1\}^n$
- Pick u.r. $i \in [n]$
- Replace coord i with Ber$(\frac{1}{2})$



stationary: uniform

Example: coloring

- $\Omega =$ valid colorings
- Pick u.r. vert v
- Replace v's color u.r. with valid color

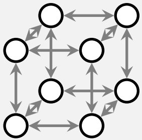


stationary: uniform

- Irreducible: possible to reach from every x to every y.
- Aperiodic: length of cycles from x to x have $\mathbf{gcd} = 1$.
- Ergodic: irreducible+aperiodic

Example: hypercube

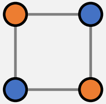
- $\Omega = \{0,1\}^n$
- Pick u.r. $i \in [n]$
- Replace coord i with Ber$(\frac{1}{2})$



stationary: uniform

Example: coloring

- $\Omega =$ valid colorings
- Pick u.r. vert v
- Replace v's color u.r. with valid color



stationary: uniform

- Irreducible: possible to reach from every x to every y.
- Aperiodic: length of cycles from x to x have $\mathsf{gcd} = 1$.
- Ergodic: irreducible+aperiodic

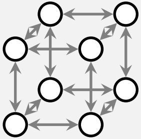
Fundamental theorem

Every ergodic chain has a unique stationary dist μ, and for any dist ν

$$\lim_{t\to\infty} \nu P^t = \mu.$$

Example: hypercube

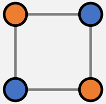
- $\Omega = \{0,1\}^n$
- Pick u.r. $i \in [n]$
- Replace coord i with Ber$(\frac{1}{2})$



stationary: uniform

Example: coloring

- $\Omega =$ valid colorings
- Pick u.r. vert v
- Replace v's color u.r. with valid color



stationary: uniform

- Irreducible: possible to reach from every x to every y.
- Aperiodic: length of cycles from x to x have $\gcd = 1$.
- Ergodic: irreducible+aperiodic

Fundamental theorem

Every ergodic chain has a unique stationary dist μ, and for any dist ν

$$\lim_{t\to\infty} \nu P^t = \mu.$$

- Note: this convergence can be very slow.

Much more useful for us:

Mixing time

For Markov chain P with stationary μ, we set

$$t_{\text{mix}}(P, \epsilon, \nu) = \min\left\{t \;\middle|\; d_{\text{TV}}(\mu, \nu P^t) \leqslant \epsilon\right\}$$

and

$$t_{\text{mix}}(P, \epsilon) = \max\{t_{\text{mix}}(P, \epsilon, \nu) \mid \nu\}$$

Much more useful for us:

Mixing time

For Markov chain P with stationary μ, we set

$$t_{\mathsf{mix}}(P, \epsilon, \nu) = \min\left\{t \;\middle|\; d_{\mathsf{TV}}(\mu, \nu P^t) \leqslant \epsilon\right\}$$

and

$$t_{\mathsf{mix}}(P, \epsilon) = \max\{t_{\mathsf{mix}}(P, \epsilon, \nu) \mid \nu\}$$

- We will see later that we don't even have to specify ϵ, and we can just talk about $t_{\mathsf{mix}}(P)$.

i.e., it's fine to set it to $1/4$

Much more useful for us:

Mixing time

For Markov chain P with stationary μ, we set

$$t_{\text{mix}}(P, \epsilon, \nu) = \min\left\{t \;\middle|\; d_{TV}(\mu, \nu P^t) \leqslant \epsilon\right\}$$

and

$$t_{\text{mix}}(P, \epsilon) = \max\{t_{\text{mix}}(P, \epsilon, \nu) \mid \nu\}$$

- We will see later that we don't even have to specify ϵ, and we can just talk about $t_{\text{mix}}(P)$.

  i.e., it's fine to set it to $1/4$

- We usually want $t_{\text{mix}}(P) = \text{poly}\log(|\Omega|)$ for efficient algs.