Complexity of Counting

Lecture 22

#P: Toda’s Theorem
Last Time
Last Time

- \#P: counting problems of the form \(#R(x) = |\{w: R(x,w)=1\}|\), where \(R\) is a polynomial time relation.
#P: counting problems of the form \( \#R(x) = |\{w: R(x,w)=1\}| \), where \( R \) is a polynomial time relation

Can be hard: even \( \#\text{CYCLE} \) is not in FP (unless \( P = NP \))
Last Time

- $\#P$: counting problems of the form $\#R(x) = |\{w: R(x,w)=1\}|$, where $R$ is a polynomial time relation.

- Can be hard: even $\#\text{CYCLE}$ is not in FP (unless $P = NP$).

- $\#P \subseteq FP^{PP}$ (and $PP \subseteq P^{#P}$).
Last Time

\[ \text{\#P: counting problems of the form } \#R(x) = \left| \{ w : R(x,w)=1 \} \right|, \]
where \( R \) is a polynomial time relation

\[ \text{Can be hard: even \#CYCLE is not in FP (unless } P = NP) \]

\[ \text{\#P } \subseteq \text{ FP}^{\text{PP}} \text{ (and } \text{PP } \subseteq \text{ P}^{\text{#P}}) \]

\[ \text{\#P complete problems} \]
Last Time

- **#P**: counting problems of the form \( \#R(x) = |\{w: R(x,w)=1\}| \), where \( R \) is a polynomial time relation
  - Can be hard: even \( \#\text{CYCLE} \) is not in FP (unless \( P = NP \))
  - \( \#P \subseteq FP^{PP} \) (and \( PP \subseteq P^{#P} \))
  - \( \#P \) complete problems
  - \( \#\text{SAT} \)
Last Time

- \#P: counting problems of the form \#R(x) = |\{w: R(x,w)=1\}|, where R is a polynomial time relation
  - Can be hard: even \#CYCLE is not in FP (unless P = NP)
  - \#P \subseteq FP^{PP} (and PP \subseteq P^{#P})
  - \#P complete problems
    - \#SAT
    - Permanent
Last Time

- \#P: counting problems of the form \#R(x) = |\{w: R(x,w)=1\}|, where R is a polynomial time relation

  - Can be hard: even \#CYCLE is not in FP (unless P = NP)

- \#P \subseteq FP^{PP} (and PP \subseteq P^{\#P})

- \#P complete problems

- \#SAT

- Permanent

- Next: Toda’s Theorem: PH \subseteq P^{\#P} = P^{PP}
⊕P: parity of the number of witnesses
⊕ \textbf{P}

⊕ \textbf{P}: \textit{parity} of the number of witnesses

e.g. ⊕SAT. Least significant bit of #SAT.
⊕P

⊕P: **parity** of the number of witnesses

- e.g. ⊕SAT. Least significant bit of #SAT.
- May not be as powerful as PP (or #P)
⊕P

- ⊕P: parity of the number of witnesses
- e.g. ⊕SAT. Least significant bit of #SAT.
- May not be as powerful as PP (or #P)
- ⊕P ⊆ P may not imply NP = P
⊕P

⊕P: parity of the number of witnesses
- e.g. ⊕SAT. Least significant bit of #SAT.
- May not be as powerful as PP (or #P)
- ⊕P ⊆ P may not imply NP = P
- But it does imply NP ⊆ RP (even if only ⊕P ⊆ RP)
⊕P

⊕P: parity of the number of witnesses

- e.g. ⊕SAT. Least significant bit of #SAT.

- May not be as powerful as PP (or #P)

- ⊕P ⊆ P may not imply NP = P

- But it does imply NP ⊆ RP (even if only ⊕P ⊆ RP)

- Randomized reduction of NP to ⊕P
$\oplus P$

$\oplus P$: parity of the number of witnesses

e.g. $\oplus$SAT. Least significant bit of #SAT.

May not be as powerful as PP (or #$P$)

$\oplus P \subseteq P$ may not imply $NP = P$

But it does imply $NP \subseteq RP$ (even if only $\oplus P \subseteq RP$)

Randomized reduction of $NP$ to $\oplus P$

i.e., $\oplus P$ oracle is quite useful to randomized algorithms
$\Theta P \subseteq RP \implies NP=RP$
\( \oplus P \subseteq \text{RP} \implies \text{NP}=\text{RP} \)

Randomized reduction of NP to \( \oplus P \)
\[ \oplus P \subseteq RP \Rightarrow NP = RP \]

- Randomized reduction of NP to \( \oplus P \)

- A probabilistic polynomial time algorithm A such that
⊕P ⊆ RP \Rightarrow \text{NP}=\text{RP}

- Randomized reduction of NP to ⊕P
- A probabilistic polynomial time algorithm A such that
  \[ \varphi \notin \text{SAT} \Rightarrow \Pr[A(\varphi) \in \oplus\text{SAT}] = 0 \]
\( \oplus P \subseteq RP \implies NP=RP \)

- Randomized reduction of NP to \( \oplus P \)

- A probabilistic polynomial time algorithm \( A \) such that
  \( \varphi \notin SAT \implies \Pr[A(\varphi) \in \oplus SAT] = 0 \)

- In fact \( A(\varphi) \) will have no satisfying assignment
$\oplus P \subseteq RP \Rightarrow \text{NP} = \text{RP}$

- Randomized reduction of NP to $\oplus P$

- A probabilistic polynomial time algorithm A such that
  - $\varphi \notin \text{SAT} \Rightarrow \Pr[A(\varphi) \in \oplus \text{SAT}] = 0$
  - In fact $A(\varphi)$ will have no satisfying assignment
  - $\varphi \in \text{SAT} \Rightarrow \Pr[A(\varphi) \in \oplus \text{SAT}] \geq \varepsilon(n)$
$\oplus P \subseteq RP \Rightarrow NP=RP$

- Randomized reduction of NP to $\oplus P$

- A probabilistic polynomial time algorithm $A$ such that
  - $\varphi \notin \text{SAT} \Rightarrow Pr[A(\varphi) \in \oplus\text{SAT}] = 0$
  - In fact $A(\varphi)$ will have no satisfying assignment
  - $\varphi \in \text{SAT} \Rightarrow Pr[A(\varphi) \in \oplus\text{SAT}] \geq \varepsilon(n)$
  - With prob. $\geq \varepsilon(n)$, $A(\varphi)$ will have exactly one satisfying assignment
$\oplus P \subseteq \text{RP} \implies \text{NP} = \text{RP}$

- Randomized reduction of NP to $\oplus P$

- A probabilistic polynomial time algorithm $A$ such that
  - $\varphi \not\in \text{SAT} \implies \Pr[A(\varphi) \in \oplus\text{SAT}] = 0$
  - In fact $A(\varphi)$ will have no satisfying assignment
  - $\varphi \in \text{SAT} \implies \Pr[A(\varphi) \in \oplus\text{SAT}] \geq \varepsilon(n)$
  - With prob. $\geq \varepsilon(n)$, $A(\varphi)$ will have exactly one satisfying assignment

- If an RP algorithm for $\oplus\text{SAT}$, then an RP algorithm for SAT
$\Theta P \subseteq RP \implies NP=RP$
Randomized reduction of SAT to Unique-SAT: A probabilistic polynomial time algorithm $A$ such that

$\Theta P \subseteq RP \implies NP=RP$
$\Theta P \subseteq RP \Rightarrow NP=RP$

Randomized reduction of SAT to Unique-SAT: A probabilistic polynomial time algorithm $A$ such that

- If $\varphi \in \text{SAT}$, with prob. $\geq \varepsilon(n)$, $A_\varphi$ will have exactly one satisfying assignment. Else $A_\varphi$ will have none.
\[ \Theta \mathbb{P} \subseteq \text{RP} \Rightarrow \text{NP}=\text{RP} \]

Randomized reduction of SAT to Unique-SAT: A probabilistic polynomial time algorithm \( A \) such that

- If \( \varphi \in \text{SAT} \), with prob. \( \geq \varepsilon(n) \), \( A_\varphi \) will have exactly one satisfying assignment. Else \( A_\varphi \) will have none.

- Add a filter which will pass exactly one witness (if any), with good probability: \( A_\varphi(w) = \varphi(w) \) and \( \text{filter}(w) \)
Hashing for unique preimage
Hashing for unique preimage

Let $S \subseteq X$ be a set of size $m$. Consider a pair-wise independent hash-function family $H$, from $X$ to $R$, such that $|S|/|R| \in [1/4,1/2]$. 
Let $S \subseteq X$ be a set of size $m$. Consider a pair-wise independent hash-function family $H$, from $X$ to $R$, such that $|S|/|R| \in [1/4, 1/2]$. 

Pr$_h[h(x)=0] = 1/|R| =: p$, and Pr$_h[h(x)=h(y)=0] = p^2$. $|S|p \in [1/4, 1/2]$. 

Hashing for unique preimage
Hashing for unique preimage

Let \( S \subseteq X \) be a set of size \( m \). Consider a pair-wise independent hash-function family \( H \), from \( X \) to \( R \), such that \( |S|/|R| \in [1/4,1/2] \).

- \( \Pr_h[h(x)=0] = 1/|R| =: p \), and \( \Pr_h[h(x)=h(y)=0] = p^2 \). \( |S|p \in [1/4,1/2] \).

- Let \( N := |\{x \in S|h(x)=0\}| \). \( \Pr_h[N=1] = \Pr_h[N \geq 1] - \Pr_h[N \geq 2] \).
Hashing for unique preimage

Let $S \subseteq X$ be a set of size $m$. Consider a pair-wise independent hash-function family $H$, from $X$ to $R$, such that $|S|/|R| \in [1/4,1/2]$.

Pr$_h[h(x)=0] = 1/|R| =: p$, and Pr$_h[h(x)=h(y)=0] = p^2$. \(|S|p \in [1/4,1/2]\).

Let $N := |\{x \in S | h(x)=0\}|$. Pr$_h[N=1] = Pr_h[N \geq 1] - Pr_h[N \geq 2]$.

By inclusion-exclusion: Pr$_h[N \geq 1] \geq \Sigma_x Pr_h[h(x)=0] - \Sigma_{x>y} Pr_h[h(x)=h(y)=0]$
Let $S \subseteq X$ be a set of size $m$. Consider a pair-wise independent hash-function family $H$, from $X$ to $R$, such that $|S|/|R| \in [1/4,1/2]$. 

$\Pr[H[h(x)=0] = 1/|R| =: p$, and $\Pr[H[h(x)=h(y)=0] = p^2$. $|S|p \in [1/4,1/2]$. 

Let $N := |\{x \in S | h(x)=0\}|$. $\Pr[H[N=1] = \Pr[H[N \geq 1] - \Pr[H[N \geq 2]]$ 

By inclusion-exclusion: $\Pr[H[N \geq 1] \geq \sum_x \Pr[H[h(x)=0] - \sum_{x>y} \Pr[H[h(x)=h(y)=0] \quad |S|p$
Let $S \subseteq X$ be a set of size $m$. Consider a pair-wise independent hash-function family $H$, from $X$ to $R$, such that $|S|/|R| \in [1/4, 1/2]$.

$\Pr_h[h(x)=0] = 1/|R| =: p$, and $\Pr_h[h(x)=h(y)=0] = p^2$. $|S|p \in [1/4, 1/2]$.

Let $N := |\{x \in S|h(x)=0\}|$. $\Pr_h[N=1] = \Pr_h[N\geq 1] - \Pr_h[N\geq 2]$.

By inclusion-exclusion: $\Pr_h[N\geq 1] \geq \sum_x \Pr_h[h(x)=0] - \sum_{x>y} \Pr_h[h(x)=h(y)=0]$.
Let $S \subseteq X$ be a set of size $m$. Consider a pair-wise independent hash-function family $H$, from $X$ to $R$, such that $|S|/|R| \in [1/4, 1/2]$.

$Pr_h[h(x)=0] = 1/|R| =: p$, and $Pr_h[h(x)=h(y)=0] = p^2$. $|S|p \in [1/4, 1/2]$.

Let $N := |\{x \in S | h(x)=0\}|$. $Pr_h[N=1] = Pr_h[N\geq1] - Pr_h[N\geq2]$

By inclusion-exclusion: $Pr_h[N\geq1] \geq \Sigma_x Pr_h[h(x)=0] - \Sigma_{x>y} Pr_h[h(x)=h(y)=0]$

By Union-bound: $Pr_h[N\geq2] \leq \Sigma_{x>y} Pr_h[h(x)=h(y)=0]$
Hashing for unique preimage

Let $S \subseteq X$ be a set of size $m$. Consider a pair-wise independent hash-function family $H$, from $X$ to $R$, such that $|S|/|R| \in [1/4, 1/2]$.

- $\Pr_h[h(x)=0] = 1/|R| =: p$, and $\Pr_h[h(x)=h(y)=0] = p^2$. $|S|p \in [1/4, 1/2]$.

- Let $N := |\{x \in S|h(x)=0\}|$. $\Pr_h[N=1] = \Pr_h[N \geq 1] - \Pr_h[N \geq 2]$

By inclusion-exclusion: $\Pr_h[N \geq 1] \geq \sum_x \Pr_h[h(x)=0] - \sum_{x>y} \Pr_h[h(x)=h(y)=0]$

By Union-bound: $\Pr_h[N \geq 2] \leq \sum_{x>y} \Pr_h[h(x)=h(y)=0]$
Let $S \subseteq X$ be a set of size $m$. Consider a pair-wise independent hash-function family $H$, from $X$ to $R$, such that $|S|/|R| \in [1/4, 1/2]$.

$\Pr_h[h(x)=0] = 1/|R| =: p$, and $\Pr_h[h(x)=h(y)=0] = p^2$. $|S|p \in [1/4, 1/2]$.

Let $N := |\{x \in S | h(x)=0\}|$. $\Pr_h[N=1] = \Pr_h[N \geq 1] - \Pr_h[N \geq 2]$

By inclusion-exclusion: $\Pr_h[N \geq 1] \geq \sum_x \Pr_h[h(x)=0] - \sum_{x>y} \Pr_h[h(x)=h(y)=0]$

By Union-bound: $\Pr_h[N \geq 2] \leq \sum_{x>y} \Pr_h[h(x)=h(y)=0]$

$\Pr_h[N=1] \geq |S|p - 2 \binom{|S|}{2} p^2 \geq |S|p - (|S|p)^2 \geq 3/16$
⊕ P ⊆ RP ⇒ NP=RP

Randomized reduction of SAT to Unique-SAT: A probabilistic polynomial time algorithm A such that

- If \( \varphi \in \text{SAT} \), with prob. \( \geq \varepsilon(n) \), \( A_\varphi \) will have exactly one satisfying assignment. Else \( A_\varphi \) will have none.

- Add a filter which will pass exactly one witness (if any): \( A_\varphi(w) = \varphi(w) \) and filter(w)
\oplus P \subseteq RP \Rightarrow NP=RP

Randomized reduction of SAT to Unique-SAT: A probabilistic polynomial time algorithm A such that

If \varphi \in SAT, with prob. \geq \varepsilon(n), A_\varphi will have exactly one satisfying assignment. Else A_\varphi will have none.

Add a filter which will pass exactly one witness (if any):
A_\varphi(w) = \varphi(w) and filter(w)

filter(w): a Boolean formula saying h(w)=0. (If using auxiliary variables, i.e., \exists z filter(w,z), use a parsimonious reduction.)
\[ \Theta P \subseteq RP \Rightarrow NP=RP \]

Randomized reduction of SAT to Unique-SAT: A probabilistic polynomial time algorithm \( A \) such that

- If \( \varphi \in SAT \), with prob. \( \geq \varepsilon(n) \), \( A_{\varphi} \) will have exactly one satisfying assignment. Else \( A_{\varphi} \) will have none.

- Add a filter which will pass exactly one witness (if any):
  \[ A_{\varphi}(w) = \varphi(w) \text{ and } \text{filter}(w) \]

  \( \text{filter}(w) \): a Boolean formula saying \( h(w)=0 \). (If using auxiliary variables, i.e., \( \exists z \text{ filter}(w,z) \), use a parsimonious reduction.)

- If witness \( n \)-bit long (\( |X|={0,1}^n \)), pick \( R={0,1}^k \), with \( k \) random in the range \([1,n]\)
Reducing PH to P#P
Reducing PH to $P^{#P}$

- Two steps
Reducing PH to $\text{P}^\#\text{P}$

- Two steps
  - Randomized reduction of PH to $\text{P}^{\oplus\text{P}}$
Reducing PH to $P^{\#P}$

Two steps

- Randomized reduction of PH to $P^{\oplus P}$
- Converting the probabilistic guarantee to a deterministic $\#P$ statement
Quantifier Gallery!
Quantifier Gallery!

∃
For at least one
Quantifier Gallery!

∃
For at least one

∀
For all
Quantifier Gallery!

\( \exists \)  
For at least one

\( \forall \)  
For all

\( \exists_r \)  
For at least \( r \) fraction
Quantifier Gallery!

∃
For at least one

∀
For all

∃r
For at least r fraction

∃!
For exactly one
Quantifier Gallery!

∃
For at least one

∀
For all

∃r
For at least r fraction

∃!
For exactly one

⊕
For an odd number of
QBF to $\oplus$BF
QBF to $\oplus$BF

We have a randomized reduction: $\varphi$ to $A_{\varphi}$ such that
QBF to $\oplus$BF

We have a randomized reduction: $\varphi$ to $A_{\varphi}$ such that

\[ \exists w \varphi(w) \Rightarrow \oplus_w A_{\varphi}(w) \text{ with prob. } \geq \epsilon(n) \]
QBF to $\oplus$BF

We have a randomized reduction: $\varphi$ to $A_\varphi$ such that

$\exists w \varphi(w) \Rightarrow \oplus_w A_\varphi(w)$ with prob. $\geq \varepsilon(n)$

$\forall w \text{ not } \varphi(w) \Rightarrow \text{ not } \oplus_w A_\varphi(w)$ (with prob. = 1)
QBF to $\ominus BF$

We have a randomized reduction: $\varphi$ to $A_{\varphi}$ such that

- $\exists_w \varphi(w) \Rightarrow \oplus_w A_{\varphi}(w)$ with prob. $\geq \varepsilon(n)$

- $\forall_w \neg \varphi(w) \Rightarrow \neg \oplus_w A_{\varphi}(w)$ (with prob. = 1)

i.e., with prob $\geq \varepsilon(n)$, we have $\exists_w \varphi(w) \Leftrightarrow \oplus_w A_{\varphi}(w)$ (and hence also $\forall_w \neg \varphi(w) \Leftrightarrow \neg \oplus_w A_{\varphi}(w)$)
QBF to $\oplus$BF

We have a randomized reduction: $\varphi$ to $A_\varphi$ such that

1. $\exists_w \varphi(w) \Rightarrow \oplus_w A_\varphi(w)$ with prob. $\geq \varepsilon(n)$

2. $\forall_w \neg \varphi(w) \Rightarrow \neg \oplus_w A_\varphi(w)$ (with prob. = 1)

i.e., with prob $\geq \varepsilon(n)$, we have $\exists_w \varphi(w) \iff \oplus_w A_\varphi(w)$ (and hence also $\forall_w \neg \varphi(w) \iff \neg \oplus_w A_\varphi(w)$)

Reduction works even if $\varphi(w)$ is a quantified Boolean formula
QBF to $\oplus$BF

We have a randomized reduction: $\varphi$ to $A_{\varphi}$ such that

$\exists_w \varphi(w) \Rightarrow \oplus_w A_{\varphi}(w)$ with prob. $\geq \varepsilon(n)$

$\forall_w \text{not } \varphi(w) \Rightarrow \text{not } \oplus_w A_{\varphi}(w)$ (with prob. = 1)

i.e., with prob $\geq \varepsilon(n)$, we have $\exists_w \varphi(w) \iff \oplus_w A_{\varphi}(w)$ (and hence also $\forall_w \text{not } \varphi(w) \iff \text{not } \oplus_w A_{\varphi}(w)$)

Reduction works even if $\varphi(w)$ is a quantified Boolean formula

Can all $\exists/\forall$ be removed, by repeating, so that only $\oplus$ remain?
Some # arithmetic
Some # arithmetic

Given two boolean formulas $\varphi(x)$ and $\psi(y)$, define
Given two boolean formulas $\varphi(x)$ and $\psi(y)$, define

$F_{\varphi,\psi}(x,y): \varphi(x) \text{ and } \psi(y)$
Some # arithmetic

Given two boolean formulas $\varphi(x)$ and $\psi(y)$, define

$F_{\varphi,\psi}(x,y): \varphi(x)$ and $\psi(y)$

$\#F_{\varphi,\psi} = \#\varphi \cdot \#\psi$
Some # arithmetic

Given two boolean formulas \( \varphi(x) \) and \( \psi(y) \), define

- \( F_{\varphi \cdot \psi}(x,y) \): \( \varphi(x) \) and \( \psi(y) \)
- \( \#F_{\varphi \cdot \psi} = \#\varphi \cdot \#\psi \)
- \( F_{\varphi \lor \psi}(x,y,z) \): (\( z=0, y=0 \) and \( \varphi(x) \)) or (\( z=1, x=0 \) and \( \psi(y) \))
Given two boolean formulas $\varphi(x)$ and $\psi(y)$, define

- $F_{\varphi \cdot \psi}(x,y)$: $\varphi(x)$ and $\psi(y)$
- $#F_{\varphi \cdot \psi} = #\varphi \cdot #\psi$
- $F_{\varphi + \psi}(x,y,z)$: $(z=0,y=0$ and $\varphi(x))$ or $(z=1,x=0$ and $\psi(y))$
- $#F_{\varphi + \psi} = #\varphi + #\psi$
Some # arithmetic

Given two boolean formulas $\varphi(x)$ and $\psi(y)$, define

- $F_{\varphi.\psi}(x,y)$: $\varphi(x)$ and $\psi(y)$
  - $\#F_{\varphi.\psi} = \#\varphi \cdot \#\psi$

- $F_{\varphi+\psi}(x,y,z)$: (z=0, y=0 and $\varphi(x)$) or (z=1, x=0 and $\psi(y)$)
  - $\#F_{\varphi+\psi} = \#\varphi + \#\psi$

- $F_{\varphi+1}$: (z=0 and $\varphi(x)$) or (z=1 and x=0). $\#F_{\varphi+1} = \#\varphi + 1$
Some # arithmetic

Given two boolean formulas \( \varphi(x) \) and \( \psi(y) \), define

\[ F_{\varphi, \psi}(x, y) : \varphi(x) \text{ and } \psi(y) \]

\[ \#F_{\varphi, \psi} = \#\varphi \cdot \#\psi \]

\[ F_{\varphi+\psi}(x, y, z) : (z=0, y=0 \text{ and } \varphi(x)) \text{ or } (z=1, x=0 \text{ and } \psi(y)) \]

\[ \#F_{\varphi+\psi} = \#\varphi + \#\psi \]

\[ F_{\varphi+1} := (z=0 \text{ and } \varphi(x)) \text{ or } (z=1 \text{ and } x=0). \#F_{\varphi+1} = \#\varphi + 1 \]

Works even if \( \varphi(x), \psi(y) \) are quantified boolean formulas
Some $\oplus$ arithmetic
Some $\oplus$ arithmetic

- Boolean combinations of QBFs with $\oplus$ quantifiers
Some $\oplus$ arithmetic

Boolean combinations of QBFs with $\oplus$ quantifiers

$\oplus_x \varphi(x)$ and $\oplus_y \psi(y) \iff \oplus_{x,y} F_{\varphi,\psi}(x,y)$, i.e. $\oplus_{x,y} \varphi(x)$ and $\psi(y)$
Some $\oplus$ arithmetic

Boolean combinations of QBFs with $\oplus$ quantifiers

- $\oplus_x \varphi(x)$ and $\oplus_y \psi(y) \iff \oplus_{x,y} F_{\varphi \land \psi}(x,y)$, i.e. $\oplus_{x,y} \varphi(x) \land \psi(y)$

- not $\oplus_x \varphi(x) \iff \oplus_{x,z} F_{\varphi+1}(x,z)$. i.e. $\oplus_{x,z} (z=1, x=0)$ or $(z=0, \varphi(x))$
Some $\oplus$ arithmetic

Boolean combinations of QBFs with $\oplus$ quantifiers

- $\oplus_x \varphi(x) \text{ and } \oplus_y \psi(y) \iff \oplus_{x,y} F_{\varphi,\psi}(x,y)$, i.e. $\oplus_{x,y} \varphi(x) \text{ and } \psi(y)$

- not $\oplus_x \varphi(x) \iff \oplus_{x,z} F_{\varphi+1}(x,z)$. i.e. $\oplus_{x,z} (z=1,x=0) \text{ or } (z=0,\varphi(x))$

- $\oplus_x (\oplus_y \varphi(x,y)) \iff \oplus_{x,y} \varphi(x,y)$
Some $\oplus$ arithmetic

- Boolean combinations of QBFs with $\oplus$ quantifiers
  
  $\oplus_x \varphi(x)$ and $\oplus_y \psi(y) \iff \oplus_{x,y} F_{\varphi,\psi}(x,y)$, i.e. $\oplus_{x,y} \varphi(x) \text{ and } \psi(y)$
  
  not $\oplus_x \varphi(x) \iff \oplus_{x,z} F_{\varphi+1}(x,z)$. i.e. $\oplus_{x,z} (z=1,x=0) \text{ or } (z=0,\varphi(x))$
  
  $\oplus_x (\oplus_y \varphi(x,y)) \iff \oplus_{x,y} \varphi(x,y)$

- $(\oplus,\exists,\forall)$-QBF can be converted to the form $\oplus_z F(z)$, where $F$ is a $(\exists,\forall)$-QBF, increasing the size by at most a constant factor, and not changing number of $\exists,\forall$-quantified variables
QBF to $\oplus$BF
QBF to $\oplus$BF

Recall: with prob $\geq \varepsilon(n)$, we have $\exists_w \varphi(w) \iff \oplus_w A_{\varphi}(w)$ (and $\forall_w \text{not } \varphi(w) \iff \text{not } \oplus_w A_{\varphi}(w)$)
QBF to $\oplus BF$

- Recall: with prob $\geq \varepsilon(n)$, we have $\exists w \varphi(w) \Leftrightarrow \oplus_w A_{\varphi}(w)$ (and $\forall w \text{ not } \varphi(w) \Leftrightarrow \text{ not } \oplus_w A_{\varphi}(w)$)

- Boosting the probability: $\varepsilon(n)$ to $1-\delta(n)$
QBF to $\oplus$BF

- Recall: with prob $\geq \varepsilon(n)$, we have $\exists_w \varphi(w) \Leftrightarrow \oplus_w A_{\varphi}(w)$ (and $\forall_w \text{not } \varphi(w) \Leftrightarrow \text{not } \oplus_w A_{\varphi}(w)$)

- Boosting the probability: $\varepsilon(n)$ to $1-\delta(n)$

  $\oplus_w A^1_{\varphi}(w)$ or $\oplus_w A^2_{\varphi}(w)$ or ... or $\oplus_w A^t_{\varphi}(w)$
QBF to $\oplusBF$

- Recall: with prob $\geq \epsilon(n)$, we have $\exists_w \varphi(w) \iff \oplus_w A_{\varphi}(w)$ (and $\forall_w \neg \varphi(w) \iff \neg \oplus_w A_{\varphi}(w)$)

- Boosting the probability: $\epsilon(n)$ to $1-\delta(n)$

  - $\oplus_w A^1_{\varphi}(w)$ or $\oplus_w A^2_{\varphi}(w)$ or ... or $\oplus_w A^t_{\varphi}(w)$

  - Can rewrite in the form $\oplus_z B_{\varphi}(z)$ where $B_{\varphi}$ has no $\oplus$
QBF to $\oplus$BF

Recall: with prob $\geq \varepsilon(n)$, we have $\exists_w \varphi(w) \iff \oplus_w A_{\varphi}(w)$ (and $\forall_w \text{not} \varphi(w) \iff \text{not} \oplus_w A_{\varphi}(w)$)

Boosting the probability: $\varepsilon(n)$ to $1-\delta(n)$

$\oplus_w A^1_{\varphi}(w) \lor \oplus_w A^2_{\varphi}(w) \lor \ldots \lor \oplus_w A^t_{\varphi}(w)$

Can rewrite in the form $\oplus_z B_{\varphi}(z)$ where $B_{\varphi}$ has no $\oplus$

In prenex form $\oplus_z B_{\varphi}(z)$ has one less $\exists/\forall$ than $\exists_w \varphi(w)$
QBF to $\oplus$BF

Recall: with prob $\geq \varepsilon(n)$, we have $\exists_w \varphi(w) \iff \oplus_w A_{\varphi}(w)$ (and $\forall_w \neg \varphi(w) \iff \neg \oplus_w A_{\varphi}(w)$)

Boosting the probability: $\varepsilon(n)$ to $1-\delta(n)$

$\oplus_w A^1_{\varphi}(w)$ or $\oplus_w A^2_{\varphi}(w)$ or ... or $\oplus_w A^t_{\varphi}(w)$

Can rewrite in the form $\oplus_z B_{\varphi}(z)$ where $B_{\varphi}$ has no $\oplus$

In prenex form $\oplus_z B_{\varphi}(z)$ has one less $\exists/\forall$ than $\exists_w \varphi(w)$

If we start from $\oplus_x \exists_w \varphi(w,x)$ we get equivalent (with probability $1-\delta(n)$) $\oplus_x \oplus_z B_{\varphi}(z,x)$
QBF to $\bigoplus\bf BF$

- Recall: with prob $\geq \varepsilon(n)$, we have $\exists_w \varphi(w) \iff \bigoplus_w A\varphi(w)$ (and $\forall_w \text{not } \varphi(w) \iff \text{not } \bigoplus_w A\varphi(w)$)

- Boosting the probability: $\varepsilon(n)$ to $1-\delta(n)$

  - $\bigoplus_w A^{1\varphi}(w)$ or $\bigoplus_w A^{2\varphi}(w)$ or ... or $\bigoplus_w A^t\varphi(w)$

  - Can rewrite in the form $\bigoplus_z B\varphi(z)$ where $B\varphi$ has no $\bigoplus$

  - In prenex form $\bigoplus_z B\varphi(z)$ has one less $\exists/\forall$ than $\exists_w \varphi(w)$

  - If we start from $\bigoplus_x \exists_w \varphi(w,x)$ we get equivalent (with probability $1-\delta(n)$) $\bigoplus_x \bigoplus_z B\varphi(z,x)$

  - By repeating, QBF can be converted to the form $\bigoplus_z F(z)$ where $F$ is unquantified, equivalent with prob. close to 1
Reducing PH to P^{#P}
Reducing PH to $\mathbf{P}^\#\mathbf{P}$

- Two steps
Reducing PH to $P^{\#P}$

- Two steps
  - Randomized reduction of PH to $P^{\oplus P}$
Reducing PH to $P^{\#P}$

- Two steps

- Randomized reduction of PH to $P^{\oplus P}$
- $\Sigma_k$SAT instance $\psi$ to $\oplus$SAT instance $\varphi_\psi$
Reducing PH to $P^{\#P}$

- Two steps

  - Randomized reduction of PH to $P^{\oplus P}$
  
    - $\Sigma_k$SAT instance $\psi$ to $\oplus$SAT instance $\varphi_\psi$
    
    - $\psi \Rightarrow \oplus \varphi_\psi$ w.p. $> 2/3$; $\neg \psi \Rightarrow \neg \oplus \varphi_\psi$ (w.p. 1)
Reducing PH to $P^{\#P}$

Two steps

- Randomized reduction of PH to $P^{\oplus P}$
  - $\Sigma_k$SAT instance $\psi$ to $\oplus$SAT instance $\varphi_\psi$
  - $\psi \Rightarrow \oplus \varphi_\psi$ w.p. > 2/3; $\neg \psi \Rightarrow \neg \oplus \varphi_\psi$ (w.p. 1)

- Converting the probabilistic guarantee to a deterministic $\#P$ calculation
Reducing \( \text{PH} \) to \( \text{P}^{\#P} \)

Two steps

- Randomized reduction of \( \text{PH} \) to \( \text{P}^{\oplus P} \)
  - \( \Sigma_k \text{SAT} \) instance \( \psi \) to \( \oplus \text{SAT} \) instance \( \varphi_\psi \)
  - \( \psi \Rightarrow \oplus \varphi_\psi \) w.p. > 2/3; \( \neg \psi \Rightarrow \neg \oplus \varphi_\psi \) (w.p. 1)

- Converting the probabilistic guarantee to a deterministic \( \#P \) calculation
  - \( \psi \) s.t. \( \neg \oplus \varphi_\psi \Rightarrow \#\theta_\psi = 0 \) (mod \( N \))
Reducing PH to $P^{\#P}$

- Two steps

  - Randomized reduction of PH to $P^{\oplus P}$
    - $\Sigma_k$SAT instance $\psi$ to $\oplus$SAT instance $\varphi_\psi$
    - $\psi \Rightarrow \oplus\varphi_\psi$ w.p. $> 2/3$; $\neg\psi \Rightarrow \neg\oplus\varphi_\psi$ (w.p. 1)

  - Converting the probabilistic guarantee to a deterministic $\#P$ calculation

    - $\psi$ s.t. $\neg\oplus\varphi_\psi \Rightarrow \#\theta_\psi = 0 \pmod{N}$
    - $\psi$ s.t. $\oplus\varphi_\psi$ w.p. $> 2/3 \Rightarrow \#\theta_\psi \neq 0 \pmod{N}$
Reduction to \#P
Reduction to \#P

- Converting the probabilistic guarantee to a deterministic \#P calculation
Reduction to \#P

- Converting the probabilistic guarantee to a deterministic \#P calculation
- \( \psi \) s.t. \( \neg \oplus \varphi_{\psi} \Rightarrow \#\theta_{\psi} = 0 \pmod{N} \)
Reduction to #P

- Converting the probabilistic guarantee to a deterministic #P calculation
  - $\psi$ s.t. $\neg \oplus \varphi_\psi \Rightarrow \#\theta_\psi = 0 \pmod{N}$
  - $\psi$ s.t. $\oplus \varphi_\psi$ w.p. $> 2/3 \Rightarrow \#\theta_\psi \neq 0 \pmod{N}$
Reduction to \#P

- Converting the probabilistic guarantee to a deterministic \#P calculation
  - \( \psi \) s.t. \( \neg \oplus \varphi_\psi \Rightarrow \# \theta_\psi = 0 \pmod{N} \)
  - \( \psi \) s.t. \( \oplus \varphi_\psi \) w.p. > 2/3 \( \Rightarrow \# \theta_\psi \neq 0 \pmod{N} \)

- Attempt 1: let \( \varphi_\psi^r \) be the formula generated using random tape \( r \). To determine if \( \psi \) is such that number of random tapes \( r \) for which \( \oplus \varphi_\psi^r \) holds is 0 or > \( (2/3)2^m \)
Reduction to \#P

- Converting the probabilistic guarantee to a deterministic \#P calculation
  \[ \psi \text{ s.t. } \neg \oplus \varphi_\psi \Rightarrow \#\theta_\psi = 0 \pmod{N} \]
  \[ \psi \text{ s.t. } \oplus \varphi_\psi \text{ w.p. } > \frac{2}{3} \Rightarrow \#\theta_\psi \neq 0 \pmod{N} \]
  Attempt 1: let \( \varphi_\psi^r \) be the formula generated using random tape \( r \). To determine if \( \psi \) is such that number of random tapes \( r \) for which \( \oplus \varphi_\psi^r \) holds is 0 or \( > \frac{2}{3}2^m \)
  - Enough to compute \( \#_r \oplus \varphi_\psi^r \)
Reduction to $\#P$

- Converting the probabilistic guarantee to a deterministic $\#P$ calculation
  - $\psi$ s.t. $\neg \oplus \varphi_\psi \Rightarrow \#\theta_\psi = 0 \pmod{N}$
  - $\psi$ s.t. $\oplus \varphi_\psi$ w.p. $> 2/3 \Rightarrow \#\theta_\psi \neq 0 \pmod{N}$

- Attempt 1: let $\varphi_\psi^r$ be the formula generated using random tape $r$. To determine if $\psi$ is such that number of random tapes $r$ for which $\oplus \varphi_\psi^r$ holds is $0$ or $> (2/3)2^m$
  - Enough to compute $\#_r \oplus \varphi_\psi^r$
  - But $\oplus \varphi_\psi^r$ may not be in P (though $\varphi_\psi^r(x)$ is in P)
Reduction to \#P
Reduction to $\#P$

Attempt 2: If $\bigoplus_x \varphi_\psi^r = \#_x \varphi_\psi^r$ then enough to compute the number of $(x,r)$ such that $\varphi_\psi^r(x)$
Reduction to \( \#P \)

- Attempt 2: If \( \oplus_x \varphi^r_\psi = \#_x \varphi^r_\psi \) then enough to compute the number of \((x,r)\) such that \( \varphi^r_\psi(x) \)

- But \( \oplus \varphi \) is \( \# \varphi \mod 2 \)
Reduction to \#P

- Attempt 2: If $\oplus_x \varphi_\psi^r = \#_x \varphi_\psi^r$ then enough to compute the number of $(x,r)$ such that $\varphi_\psi^r(x)$

- But $\oplus \varphi$ is $\#\varphi \mod 2$

- Plan: Change sum of parities to sum of sums
Reduction to $\#P$

- Attempt 2: If $\oplus_x \varphi^r = \#_x \varphi^r$ then enough to compute the number of $(x,r)$ such that $\varphi^r(x)$

- But $\oplus \varphi$ is $\#\varphi \mod 2$

- Plan: Change sum of parities to sum of sums

- Create $\varphi' = T(\varphi)$, such that
Reduction to \#P

Attempt 2: If $\oplus_x \phi_\psi^r = \#_x \phi_\psi^r$ then enough to compute the number of $(x, r)$ such that $\phi_\psi^r(x)$

 But $\oplus \phi$ is $\#\phi \mod 2$

Plan: Change sum of parities to sum of sums

Create $\phi' = T(\phi)$, such that

For each $r$, $\neg \oplus_x \phi \Rightarrow \#_x \phi' = 0 \mod N$
Reduction to \#P

Attempt 2: If $\oplus_x \varphi_{\psi^r} = \#_x \varphi_{\psi^r}$ then enough to compute the number of $(x, r)$ such that $\varphi_{\psi^r}(x)$

But $\oplus \varphi$ is $\# \varphi \mod 2$

Plan: Change sum of parities to sum of sums

Create $\varphi' = T(\varphi)$, such that

For each $r$, $\neg \oplus_x \varphi \Rightarrow \#_x \varphi' = 0 \mod N$

For each $r$, $\oplus_x \varphi \Rightarrow \#_x \varphi' = -1 \mod N$
Reduction to \#P

Attempt 2: If $\oplus_x \varphi^r = \#_x \varphi^r$ then enough to compute the number of $(x,r)$ such that $\varphi^r(x)$

- But $\oplus \varphi$ is $\# \varphi \mod 2$
- Plan: Change sum of parities to sum of sums
- Create $\varphi' = T(\varphi)$, such that
  - For each $r$, $\neg \oplus_x \varphi \Rightarrow \#_x \varphi' = 0 \mod N$
  - For each $r$, $\oplus_x \varphi \Rightarrow \#_x \varphi' = -1 \mod N$

$N > 2^m$ so that for $(2/3)2^m < R \leq 2^m$ we have $R(-1) \not\equiv 0 \mod N$
Reduction to \( \#P \)

Attempt 2: If \( \oplus_x \varphi_{\psi^r} = \#_x \varphi_{\psi^r} \) then enough to compute the number of \((x,r)\) such that \( \varphi_{\psi^r}(x) \)

- But \( \oplus \varphi \) is \( \#\varphi \mod 2 \)
- Plan: Change sum of parities to sum of sums
- Create \( \varphi' = T(\varphi) \), such that

- For each \( r \), \( -\oplus_x \varphi \Rightarrow \#_x \varphi' = 0 \mod N \)
- For each \( r \), \( \oplus_x \varphi \Rightarrow \#_x \varphi' = -1 \mod N \)

- \( N > 2^m \) so that for \((2/3).2^m < R \leq 2^m \) we have \( R.(-1) \neq 0 \mod N \)
- Let \( \theta_{\psi}(x,r) = T(\varphi_{\psi^r})(x) \). Use \( \#\theta_{\psi} \mod N \) to check if w.h.p. \( \oplus \varphi \)
Reduction to $\mathcal{NP}$
Reduction to \#P

Remains to do: Given \( \varphi \), create \( \varphi' \) such that for \( N = 2^{2^k} \), where \( k = O(\log m) \)
Reduction to \#P

Remains to do: Given \( \varphi \), create \( \varphi' \) such that for \( N=2^{2^k} \), where \( k = O(\log m) \)

\[ \neg \oplus \varphi \Rightarrow \#\varphi' = 0 \mod N \]
Reduction to \#P

Remains to do: Given $\varphi$, create $\varphi'$ such that for $N = 2^{2^k}$, where $k = O(\log m)$

- $\neg \oplus \varphi \Rightarrow \#\varphi' = 0 \mod N$
- $\oplus \varphi \Rightarrow \#\varphi' = -1 \mod N$
Reduction to \#P

Remains to do: Given $\varphi$, create $\varphi'$ such that for $N=2^{2^k}$, where $k=O(\log m)$

- $\neg \oplus \varphi \Rightarrow \#\varphi' = 0 \mod N$
- $\oplus \varphi \Rightarrow \#\varphi' = -1 \mod N$

Initially true for $N = 2 \left(2^{2^i}, i=0\right)$
Reduction to $\#P$

Remains to do: Given $\varphi$, create $\varphi'$ such that for $N=2^{2^k}$, where $k = O(\log m)$

- $\neg \oplus \varphi \Rightarrow \#\varphi' = 0 \mod N$
- $\ominus \varphi \Rightarrow \#\varphi' = -1 \mod N$

Initially true for $N = 2$ ($2^{2^i}$, $i=0$)

$\varphi_{i+1} = F_4(\varphi_i)^3 + 3(\varphi_i)^4$ so that $\#\varphi_{i+1} = 4(\#\varphi_i)^3 + 3(\#\varphi_i)^4$
Reduction to \( \#P \)

Remains to do: Given \( \varphi \), create \( \varphi' \) such that for \( N=2^{2^k} \), where \( k = O(\log m) \)

\[
\neg \oplus \varphi \Rightarrow \#\varphi' = 0 \mod N
\]

\[
\oplus \varphi \Rightarrow \#\varphi' = -1 \mod N
\]

Initially true for \( N = 2 \ (2^{2^i}, \ i=0) \)

\[
\varphi_{i+1} = F_4(\varphi_i)^3 + 3(\varphi_i)^4 \text{ so that } \#\varphi_{i+1} = 4(\#\varphi_i)^3 + 3(\#\varphi_i)^4
\]

\[
\#\varphi_i = -1 \mod 2^{2^i} \text{ implies } \varphi_{i+1} = -1 \mod 2^{2^{i+1}} \text{ (for } i \geq 0)\]
Reduction to \#P

Remains to do: Given \( \varphi \), create \( \varphi' \) such that for \( N=2^{2^k} \), where \( k = O(\log m) \)

\[ \neg \oplus \varphi \Rightarrow \#\varphi' = 0 \mod N \]
\[ \oplus \varphi \Rightarrow \#\varphi' = -1 \mod N \]

Initially true for \( N = 2 \ (2^{2^i}, i=0) \)

\[ \varphi_{i+1} = F_4(\varphi_i)^3 + 3(\varphi_i)^4 \] so that \( \#\varphi_{i+1} = 4(\#\varphi_i)^3 + 3(\#\varphi_i)^4 \)

\[ \#\varphi_i = -1 \mod 2^{2^i} \] implies \( \varphi_{i+1} = -1 \mod 2^{2^{i+1}} \) (for \( i \geq 0 \))

Clearly \( \#\varphi_i = 0 \mod 2^{2^i} \) implies \( \varphi_{i+1} = 0 \mod 2^{2^{i+1}} \)
$\text{PH} \subseteq \text{P}^{\#P}$
$\text{PH} \subseteq \text{P}^\text{#P}$

Summary:
\[ \text{PH} \subseteq \text{P}^{\text{#P}} \]

Summary:

- First, randomized reduction of PH to \( \text{P}^{\oplus \text{P}} \)
\[
\text{PH} \subseteq \text{P}^{\#P}
\]

Summary:
- First, randomized reduction of PH to \( P^{\oplus P} \)
- \( \Sigma_k \text{SAT} \) instance \( \psi \) to \( \oplus \text{SAT} \) instance \( \varphi_\psi \)
$\text{PH} \subseteq \text{P}^{\#P}$

Summary:

- First, randomized reduction of PH to $\text{P}^{\oplus P}$
- $\Sigma_k$SAT instance $\psi$ to $\oplus$SAT instance $\varphi_\psi$
- $\psi \Rightarrow \oplus \varphi_\psi$ w.p. $> 2/3$; $\neg \psi \Rightarrow \neg \oplus \varphi_\psi$ (w.p. 1)
$\text{PH} \subseteq \text{P}^{\#P}$

**Summary:**

First, randomized reduction of PH to $\text{P}^{\oplus P}$

- $\Sigma_k \text{SAT}$ instance $\psi$ to $\oplus \text{SAT}$ instance $\varphi_\psi$

  $\psi \Rightarrow \oplus \varphi_\psi \text{ w.p. } > 2/3$; $\neg \psi \Rightarrow \neg \oplus \varphi_\psi$ (w.p. 1)

- Converting the probabilistic guarantee to a deterministic $\#P$ calculation
PH ⊆ P^{#P}

Summary:

- First, randomized reduction of PH to $P^{⊕P}$
  - $Σ_k$SAT instance $ψ$ to $⊕$SAT instance $φ_ψ$
  - $ψ \Rightarrow ⊕φ_ψ$ w.p. $> 2/3$; $¬ψ \Rightarrow ¬⊕φ_ψ$ (w.p. 1)
- Converting the probabilistic guarantee to a deterministic $#P$ calculation
  - $ψ$ s.t. $¬⊕φ_ψ \Rightarrow #θ_ψ = 0$ (mod $N$)
\( \text{PH} \subseteq \text{P}^{\#P} \)

Summary:

- First, randomized reduction of \( \text{PH} \) to \( \text{P}^{\oplus} \)
  - \( \Sigma_k \text{SAT} \) instance \( \psi \) to \( \oplus \text{SAT} \) instance \( \varphi_\psi \)
  - \( \psi \Rightarrow \oplus \varphi_\psi \) w.p. > 2/3; \( \neg \psi \Rightarrow \neg \oplus \varphi_\psi \) (w.p. 1)

- Converting the probabilistic guarantee to a deterministic \( \#P \) calculation
  - \( \psi \) s.t. \( \neg \oplus \varphi_\psi \Rightarrow \#\theta_\psi = 0 \) (mod \( N \))
  - \( \psi \) s.t. \( \oplus \varphi_\psi \) w.p. > 2/3 \( \Rightarrow \#\theta_\psi \neq 0 \) (mod \( N \))
Approximation for #P
Approximation for $\#P$

$\alpha$-approximation of $f$: estimate $f(x)$ within a factor $\alpha$
Approximation for \#P

- \(\alpha\)-approximation of \(f\): estimate \(f(x)\) within a factor \(\alpha\)
- Randomized approximation ("PAC"): answer is within a factor \(\alpha\) with probability at least \(1-\delta\)
Approximation for $\#P$

- $\alpha$-approximation of $f$: estimate $f(x)$ within a factor $\alpha$

- Randomized approximation ("PAC"): answer is within a factor $\alpha$ with probability at least $1-\delta$

- $\#CYCLE$ is hard to even approximate unless $P=NP$
Approximation for \#P

- $\alpha$-approximation of $f$: estimate $f(x)$ within a factor $\alpha$

- Randomized approximation ("PAC"): answer is within a factor $\alpha$ with probability at least $1-\delta$

- $\#CYCLE$ is hard to even approximate unless $P=NP$

- If $P=NP$, every problem in $\#P$ can be "well approximated"
Approximation for #$P$

- $\alpha$-approximation of $f$: estimate $f(x)$ within a factor $\alpha$
- Randomized approximation ("PAC"): answer is within a factor $\alpha$ with probability at least $1-\delta$
- #$\text{CYCLE}$ is hard to even approximate unless P=NP
  - If P=NP, every problem in #$P$ can be "well approximated"
- Permanent has an FPRAS
Approximation for \#P

- $\alpha$-approximation of $f$: estimate $f(x)$ within a factor $\alpha$

- Randomized approximation ("PAC"): answer is within a factor $\alpha$ with probability at least $1-\delta$

- \#CYCLE is hard to even approximate unless $P=NP$

  - If $P=NP$, every problem in \#P can be "well approximated"

- Permanent has an FPRAS

  - For any $\epsilon, \delta > 0$, $\alpha$-approximation for $\alpha = 1-\epsilon$ in time $\text{poly}(n, \log 1/\epsilon, \log 1/\delta)$
Approximation for \#P

- \( \alpha \)-approximation of \( f \): estimate \( f(x) \) within a factor \( \alpha \)
- Randomized approximation (“PAC”): answer is within a factor \( \alpha \) with probability at least \( 1-\delta \)
- \#CYCLE is hard to even approximate unless \( P=NP \)
  - If \( P=NP \), every problem in \#P can be “well approximated”
- Permanent has an FPRAS
  - For any \( \epsilon, \delta > 0 \), \( \alpha \)-approximation for \( \alpha = 1-\epsilon \) in time \( \text{poly}(n, \log 1/\epsilon, \log 1/\delta) \)
- Technique: Monte Carlo Markov Chain (MCMC)
Approximation for #$P$

- $\alpha$-approximation of $f$: estimate $f(x)$ within a factor $\alpha$
- Randomized approximation ("PAC"): answer is within a factor $\alpha$ with probability at least $1-\delta$
- #$CYCLE$ is hard to even approximate unless P=NP
  - If P=NP, every problem in #$P$ can be "well approximated"
- Permanent has an FPRAS
  - For any $\epsilon, \delta > 0$, $\alpha$-approximation for $\alpha = 1-\epsilon$ in time $\text{poly}(n, \log 1/\epsilon, \log 1/\delta)$
- Technique: Monte Carlo Markov Chain (MCMC)
  - Very useful for sampling. Turns out counting $\approx$ sampling!