Fri May 18, 2018 - page 1

Friday, May 18, 2018 7:35 AM

Complexity of Counting.
So far we have focused on languages (subsets of $\{0,1\}^*$)
What about computing functions?

Important type of functions: counting problems.
Given 3-CNF ϕ, how many satisfying assignments does it have?

$\#P$ = class of functions describable as:
input x $\quad f(x) = |\{y \mid (x,y) \in R\}|$ $\quad R \in P$
$\quad \phi \quad t \quad (\phi, t)$

By contrast, NP just asks $f(x) > 0$?

Examples: #SAT : given 3-cnf ϕ how many satisfying assignments?
#CLIQUE : given (G, k) how many cliques of at least size k?

Reduction $f_1 \propto f_2$ if $\exists\, Q, A$

$$\begin{tikzcd}
\text{(input to } f_1\text{)}\ x \arrow[r, "Q"] \arrow[d, dashed, "f_1"'] & y\ \text{(input to } f_2\text{)} \arrow[d, dashed, "f_2"] \\
f_1(x) & f_2(y) \arrow[l, "A"]
\end{tikzcd}$$

A function f is $\#P$-complete if
$f \in \#P$
$\forall f' \in \#P,\ f' \propto f$

#25 SAT & Clique.

$R: \phi \to (G, k)$

A parsimonious reduction is one in which the number of witnesses is preserved, so A is the identity.

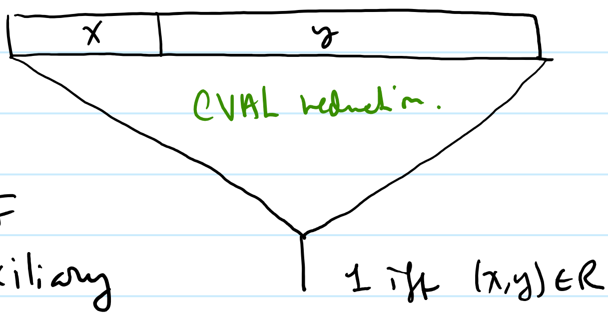
- Many standard NP-completeness reductions are parsimonious.
- If #SAT is #P-complete we get lots of #P-complete problems.

Theorem: #SAT is #P-complete.

#SAT $\in$ #P $\qquad (\phi, \vec{t}) \in R \iff \vec{t}$ satisfies ϕ.

Take $f \in \#P$ defined by $R \in P$

$f(x) = |\{y \mid (x,y) \in R\}|$



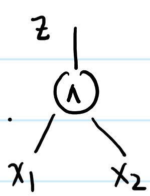
In converting circuit to 3-CNF formula ϕ we introduce auxiliary variables $\vec{z}$

$\exists \vec{z} \; \phi(x,y,z) \iff C(x,y) = 1$

Important Observation:

For fixed y, $\vec{z}_y$ is unique

$\phi_x(y,z)$ satisfied $\iff z_y = z$ and $(x,y) \in R$.



$(z \iff x_1 \wedge x_2)$

$\neg z \vee (x_1 \wedge x_2)$

$= (\neg z \vee x_1) \wedge \neg z \vee x_2)$

$z \vee \neg(x_1 \wedge x_2) = z \vee \neg x_1 \vee \neg x_2$

(z_n) New clauses

Friday, May 18, 2018 7:47 AM

So if we hardwire x:

$$\#y \text{ s.t. } C(x,y)=1 = \#(y,z) \text{ s.t. } \phi(x,y,z)=1. //$$

$$\underset{|\{y \mid (x,y)\in R\}|}{\overset{//}{}}$$

How does #P compare to other classes?

EXP
|
PSPACE
|
#P
|
PH

Use $P^{\#P}$ in order to make a class of decision problems.

NP, co-NP $\subseteq P^{\#P}$ ⤳ can use #P oracle to tell if $|\{y \mid (x,y)\in R\}| = 0$ or ≥ 1

$\#SAT \in PSPACE$ so $P^{\#P} \subseteq PSPACE$.

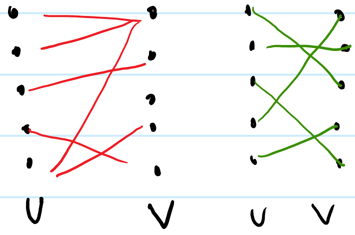
<u>Toda's Theorem</u>: $PH \subset P^{\#P}$ (won't prove here).

He actually proved: $PH \subset P^{PP}$ ⤳ accept if $\geq 1/2$ of the paths are accepting. (i.e. if first bit of count = 1).

Is #P hard because it entails <u>finding</u> NP witnesses? or is counting hard by itself?

Friday, May 18, 2018 7:35 AM

Definition: $G = (V, U, E)$ $|U| = |V|$ $E \subseteq U \times V$
bipartite graph.



A perfect matching in G is a subset $M \subseteq E$ that touch every node & no two edges share an endpoint.

MATCH $\in$ P (Given bipartite G, does G contain a perfect matching?)

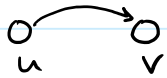
#MATCH : given bipartite G how many distinct perfect matchings does G have?

Theorem (Valiant) #MATCH is #P-complete.

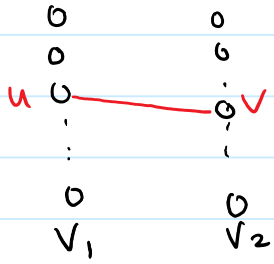
First #Cycle Cover $\propto$ # MATCH parsimonious reductions

Cycle Cover: Input directed (non-bipartite) graph G, Is there a set of cycles that visits each vertex exactly once?

Directed graph G
directed edge (u,v)



Bipartite G' two copies of G's vertices

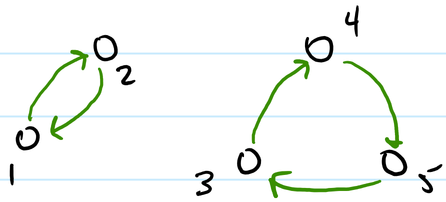


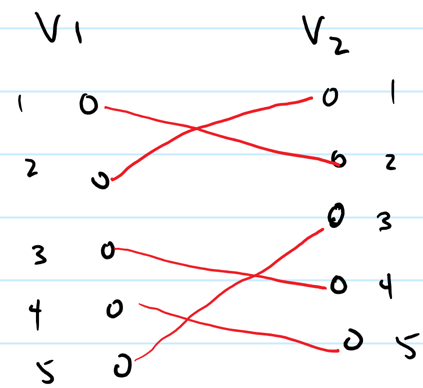
Bipartite edge
$u \in V_1$
$v \in V_2$

Fri May 18, 2018 - page 5

Friday, May 18, 2018 7:35 AM

1-1 correspondence between cycle covers in G and bipartite matchings in G'.





Given an nxn matrix:

$$\text{determinant}(M) = \sum_{\pi} \sigma(\pi) \prod_{i=1}^{n} A_{i,\pi(i)}$$

Sign of permutation π = # swaps from identity.

$$\text{permanent}(M) = \sum_{\pi} \prod_{i=1}^{n} A_{i,\pi(i)}$$

$$M = \begin{array}{cc} & v \\ u & \begin{bmatrix} & & \\ & 0/1 & \\ & & \end{bmatrix} \end{array}$$

is there an edge (u,v) in directed graph G'?

Adjacency matrix for directed graph G' (also bipartite G)

permanent(M) = # matchings in bipartite G = # cycle covers in directed G'.

Friday, May 18, 2018 7:35 AM

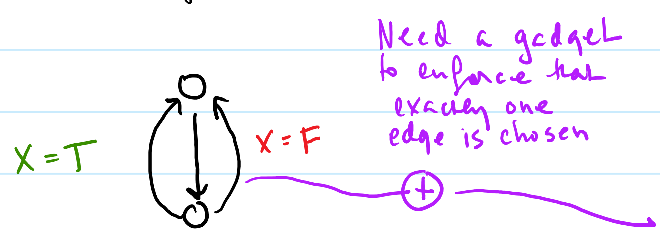
Cycle covers in directed graphs and perfect matchings in bipartite graphs define a permutation over the nodes.

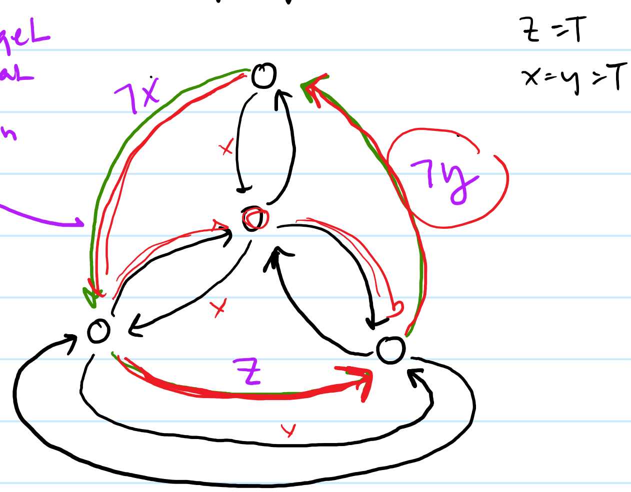
The fact that #MATCH is #P-complete means that computing the permanent is also #P-complete (even on 0/1 graphs).

Next: #SAT $\propto$ #Cycle Cover

↳ This won't be parsimonious otherwise P=NP!

Gadget for Variables: Gadget for Clauses:





Cycle chosen determines truth value of x.

Each green edge represents a literal in the clause
The green edge is in the cycle cover iff literal is false
(will enforce this)

- No cycle cover includes all green edges.
- For every proper subset of green edges → exactly one cycle cover.

Friday, May 18, 2018 7:35 AM

Exclusive OR gadget:

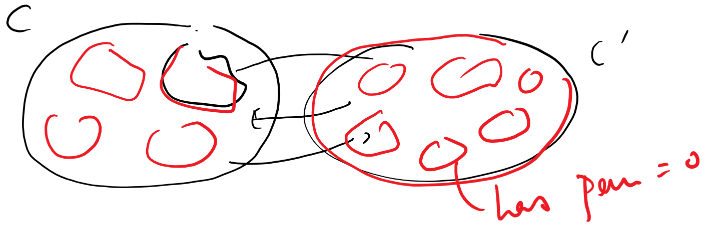
Consider a version of cycle cover with weights on the edges.

Want to compute: $\sum_{\text{cycle cover}} wt(\underline{\text{cycle cover}})$

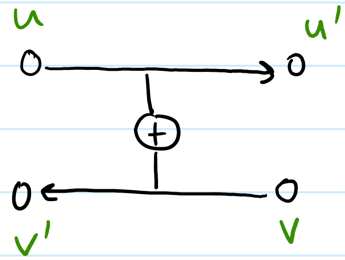
$\hookrightarrow$ product of edge weights. Non-edges have wt = 0.

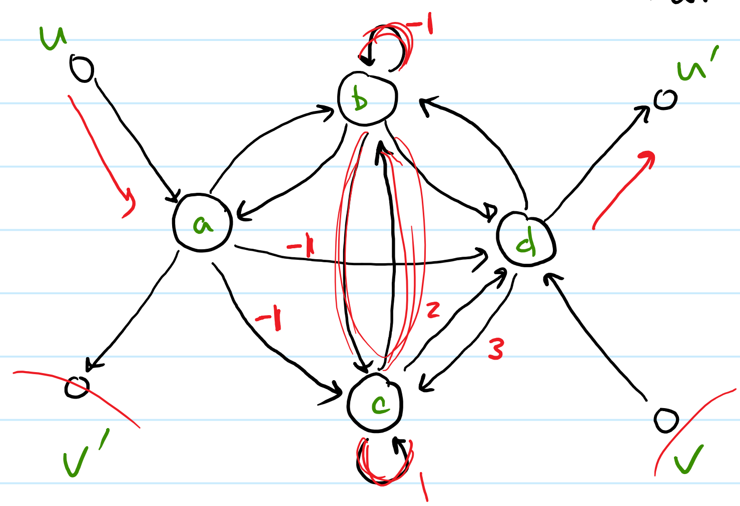
If all edges have wt = 1 then this is the same as # cycle cover. We will show:

$\#SAT \propto \#(\text{Weighted Cycle Cover}) \propto \#\text{Cycle Cover mod } N \propto \#\text{Cycle Cover}$ easy.



First Show: X-or gadget with weights (#SAT ≤ # WT-cycle-cover)





(u,u') but not (v,v')

OR (v,v') but not (u,u').

(unlabeled edges have wt = 1)

$$\begin{array}{cc} & \begin{array}{cccc} a & b & c & d \end{array} \quad u \quad u' \\ \begin{array}{c} a \\ b \\ c \\ d \end{array} & \left[\begin{array}{cccc} 0 & 1 & -1 & -1 \\ 1 & -1 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 0 & 1 & 3 & 0 \end{array}\right] \\ \begin{array}{c} u \\ u' \end{array} & \begin{array}{l} \times \\ \; \end{array} \end{array}$$

a) permanent = 0 (can't omit both (u,u') and (v,v'))

~~b~~b) if rows + columns for a + d are deleted, perm = 0 (can't go in by u + out by v' and in by v and out by u').

c) if first row + col are deleted perm = 0 (in by u, out by v)

d) if last row + col are deleted, perm = 0 (in by v, out by u')

⇒ The only cycle covers that contribute to the permanent go in by u + out by u' OR in by v and out by v'. (mult by 4)

Friday, May 18, 2018 7:35 AM

If we divide the graph into components C & C' and only consider cycles within C or C' then # cycle covers $= \text{Perm}(C) \cdot \text{Perm}(C')$

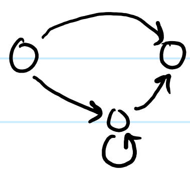
If either perm $= 0$, then these cycle covers don't contribute to $\text{Perm}(G)$.

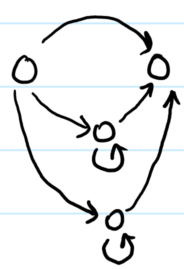
$$\#(\underline{\text{wt}}\text{-cycle cover}) = (\# \underline{\text{Sat assignments}})\, 4^{\underline{\underline{3m}}}$$

# of X-or gadgets.

How to simulate the non 0/1 entries:

For wt = 2 or 3:

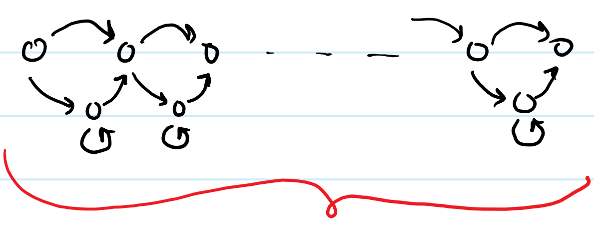




What about -1? pick $N = 2^t + 1$

edges will have weight $2^t \equiv -1 \mod N$.

Replace $\circ \xrightarrow{-1} \circ$ with:



$\times t$

Fri May 18, 2018 - page 10

Friday, May 18, 2018 8:55 AM

Pick $t = 8m$.

$$\sum_{\substack{\text{cycle}\\\text{covers}\\c}} \prod_{\substack{\text{edges in}\\c}} wt(e) = \sum_c \underbrace{wt'(c)}_{\text{wt from non -1 edges}} (\underbrace{N-1}_{2^t})^{\overset{\text{\# -1 edges in c.}}{r_c}}$$

$$\left[\sum_c wt'(c)(\underbrace{N-1}_{2^t})^{r_c}\right] \bmod N = \sum_c wt'(c)(-1)^{r_c}$$

as long as:

$$0 \le \underbrace{\sum_c wt'(c)(-1)^{r_c}}_{\le \underbrace{2^n}_{\text{\# sat assignments.}} \cdot 4^{3m} < 2^m \cdot 2^{6m} = 2^{7m}} < N = 2^{8m}$$