Last time:
- $BPP \subseteq P/poly$ (nonuniformity is at least as powerful as randomness)
- $BPP \subseteq PH$ (in fact, $BPP \subseteq \Sigma_2^P \cap \Pi_2^P$)
- start unit on complexity of counting problems

Pap 11.2, AB 7.5, Ca: 5.5

Papad. 18.1, AB 17.1–17.3

Today: More counting: #P, #P-completeness, reductions, examples, permanent.

Questions?

#SAT, #CYCLES, #PATHS: can count or obj.

φ φ φ

s.a. are "hard to find"

"easy to find"

---

① • Only makes sense to count "recognizable" objects - i.e. "NP witnesses."

② • (We'll see that even counting "easy-to-find" objects can be hard...)

#P

①

Def: A fn $f: \Sigma^* \to \mathbb{N}$ is in #P if there is a poly-time NTM M s.t. $\forall x \in \Sigma^*$,

$f(x) = \#$ of acc. comput. of M on x.

Comput. of M on x:

$f(x) = \#$ ● acc. comput. of M on x.

Equiv: $f \in \#P$ if there is an $L \in NP$, poly-time verif $V(x,w)$ for L, s.t. $\forall x$, have $f(x) = \#$ of w s.t. $V(x,w)$ accepts.

(Recall: "V is a poly-time verif for L" means: ① V runs in time poly($|x|$) for all inputs (x,w); ② $\forall x$, have $x \in L \Leftrightarrow \exists w$ s.t. $V(x,w)$ accepts.)

---

② L can be in P as well as NP & can still have that the assoc. counting problem is interesting/hard; #PATH, #CYCLE.

#3CNF: V's inputs are (φ, x) (with x, w)
V acc. (φ, x) iff x sat. φ.

Fact: Let $L \in NP$, f be corr. fn. in #P.
Have $f(x) > 0$ iff $x \in L$.

So counting problems at least as hard as assoc. dec. problems.

A different class of functions:

$FP = \{$ poly-time computable $f: \Sigma^* \to \mathbb{N} \}$.

"easy" fns; $f \in FP$ if $\exists$ polytime det TM M s.t. on input x M outputs $f(x)$ (in binary).

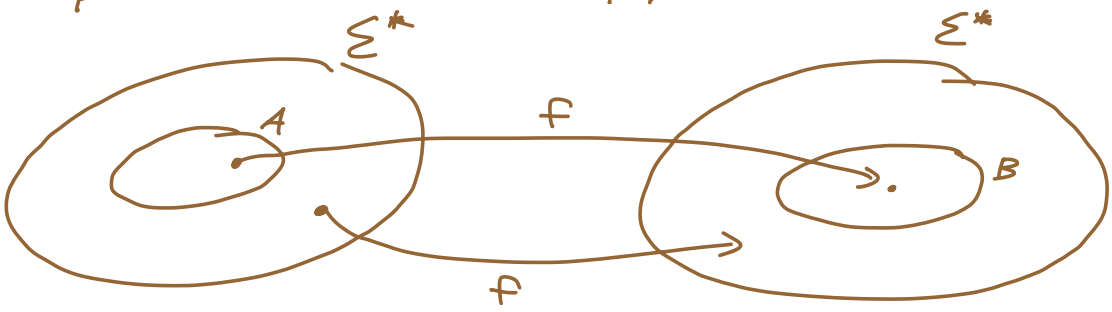
"at least as hard as NP"

Obs: If $\#P \subseteq FP$, then $P = NP$.

Completeness

Our first notion of poly-time red. for languages:

$A \leq_p B$ $\quad f$ poly-time

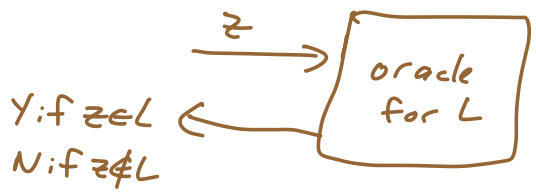


Not good for #P; our fns now output nat. #s.

Right notion for #P reductions: oracle reductions,

TM M w/ oracle for L: M^L

"magic box"



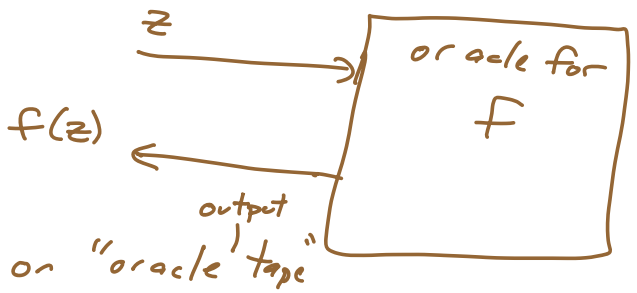
P^L: $L' \in P^L$ if $\exists$ poly-time oracle TM M s.t. M^L decides L'.

$L' \leq_p L$ means $L' \in P^L$: make 1 oracle call to L & output what it outputs.

Can do more than just this (multiple calls,

adaptive, etc); oracle reductions are more powerful/useful than $\leq_p$.

We'll now consider oracles for functions $f: \Sigma^* \to \mathbb{N}$:



value $f(z)$ provided on "oracle tape" in 1 time step when machine M^f enters "oracle state" with z on "oracle input tape".

Analogous to above, have M^f; P^f (class of lang. s.t. some poly-time TM w/ f-oracle decides it); FP^f (class of functions). This is our reduc. notion for #P.

Def: A fn $f: \Sigma^* \to \mathbb{N}$ is **#P-complete** if (a) $f \in \#P$, & (b) $\forall g \in \#P$, have $g \in FP^f$.

If $f \in FP$, then $FP^f = FP$, so

Cor: If f is #P-complete & $f \in FP$,

then $FP = \#P$.

Q: are there #P-complete fns ?

A: Yes:

input: a 3CNF : output: # sat assts.

Thm: #3CNF is #P-complete.

Pf: Clear that ⓐ #3CNF is in #P.

Now ⓑ: for every $g \in \#P$, have $g \in FP^{\#3CNF}$.

Fix any $g \in \#P$; let M be the corr. NTM, L be the lang. M decides.

We reduce g to #3CNF in two steps:

1) C-L thm ($L \leq_p$ CIRCUIT-SAT)
2) CIRCUIT-SAT $\leq_p$ 3CNF.

Each of these steps preserves # of acc. comput.:

- C-L reduc. R_{CL} from L to CKT-SAT:

given an input x, reduc. constructs ckt C s.t.

assts to inputs of C $\overset{\text{bij.}}{\Longleftrightarrow}$ seq. of nondet. choices M makes when run on x.

$C = 1$ on assignment to its input vars iff

the corr. seq. of nondet steps causes M to accept x.

So $g(x) = \#$ acc. comput. of M on x
$= $ s.a. of C.

- Second reduc. R_{3CNF} from CKT-SAT to 3CNF: given as input a ckt C on inputs x, it outputs a 3CNF $\phi(x,y)$ s.t.
  - $C(x) = 1 \Rightarrow$ there is a unique y s.t. $\phi(x,y) = 1$;
  - $C(x) = 0 \Rightarrow$ " " no y s.t. $\phi(x,y) = 1$.

[Sipser Thm. 9.34]

So # s.a. of ϕ is $\equiv$ # s.a. of C.

So, $g \in FP^{\#3CNF}$: on input x, perform R_{CL} to get ckt C; then perform $R_{\#3CNF}$ on C to get a 3CNF ϕ; call #3CNF oracle on ϕ, & that gives $g(x)$. ∎

3CNF: decision is hard (NP-hard)
counting " " (#P-hard).

For other problems,
decision is easy (in P ☺) but
counting " " (#P-hard).

A 3DNF : an OR of ANDs of size ≤3 ("terms" of size ≤3)
↑ ≤3 literals

Ex:
$(x_1 \wedge \bar{x}_3 \wedge x_4) \vee (x_2 \wedge x_3 \wedge x_4) \vee (x_4 \wedge x_5 \wedge x_6) \vee (\bar{x}_4 \wedge x_6 \wedge x_7)$

Decision version of 3CNF (is there a s.a.?)
trivial: if any term satisfiable $(\bar{x}_2 \wedge x_3 \wedge \bar{x}_4)$, Y.
If every term unsat. $(\bar{x}_1 \wedge x_1 \wedge x_3)$, N.

#3DNF: input is a 3DNF; output = # s.a.

Thm: #3DNF is #P-complete.
Pf: Clear that #3DNF is in #P.
We reduce from #3CNF.
Given $\phi(x_1, \ldots, x_n)$ a 3CNF, consider $\bar{\phi}$: this is a 3DNF.
Ex: $\phi = (x \vee y \vee z) \wedge (x \vee \bar{y} \vee \bar{z})$ Then

$\overline{\phi} = (\bar{x} \wedge \bar{y} \wedge \bar{z}) \vee (\bar{x} \wedge y \wedge z)$

S.A. of the 3DNF $\overline{\phi}$ is $2^n - (\#$ S.A. of $\phi)$.

So here's an alg for #3CNF if we have a #3DNF oracle:

input: ϕ a 3CNF (n vars)
write down $\overline{\phi}$, a 3DNF, & call #3DNF oracle, get $0 \le N \le 2^n$;
return $2^n - N$

Another ex:

Def: A Bool. formula over vars $x_1, \ldots, x_n$ is monotone if only gates it contains are $\wedge$, $\vee$ - no $\neg$.

$\phi = ((x \wedge y) \vee (z \vee w \vee x)) \wedge (y \wedge (z \vee y \vee (v \wedge w)))$
T T T T T T T T T T T T T T

#MON: input is an n-var. mon. formula
q: # s.a. ?

$\forall$ mon. formula ϕ, $\#MON(\phi) \ge 1$.

Thm: #MON is #P-complete.

[Idea: given a 3CNF ϕ (may have negated vars), cook up two mon. formulas $\phi_1 + \phi_2$, designed s.t.

$$(\#\text{s.a. for } \phi_1) - (\#\text{s.a. for } \phi_2) = (\#\text{s.a. for } \phi).$$

2 oracle calls.]

Fact: there are nontrivial, non-hard counting problems.

#SPANTREE: input: undir. n-node G.

q: # spanning trees in G?

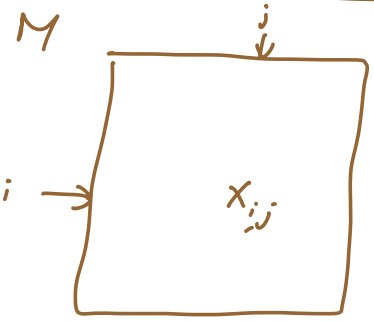
Kirchhoff's matrix tree theorem:

$$\#\text{SPANTREE} \in \text{FP}.$$

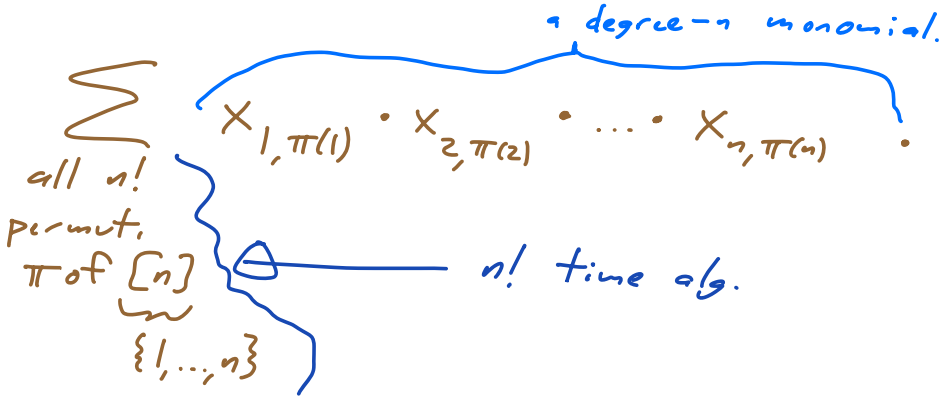
A famous #P-completeness result: the permanent.

Let M be $n \times n$ matrix

x_{ij} = (i,j) entry.

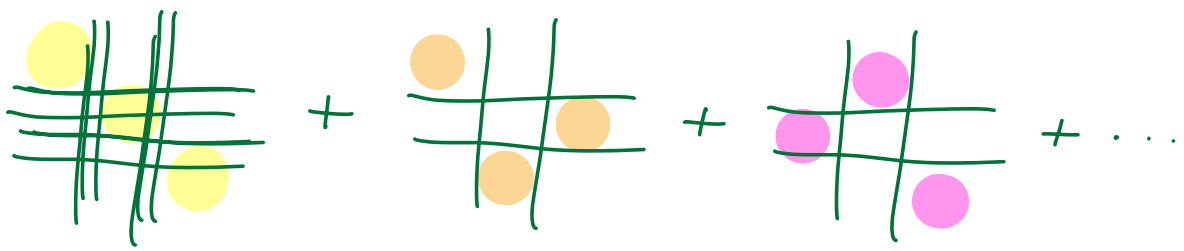


Def Per(M) (permanent of M) is



Ex: $M = \begin{array}{|c|c|c|}\hline x_{11} & x_{12} & x_{13} \\ \hline x_{21} & x_{22} & x_{23} \\ \hline x_{31} & x_{32} & x_{33} \\ \hline \end{array}$

$$PER(M) = x_{11}x_{22}x_{33} + x_{11}x_{23}x_{32} + x_{12}x_{21}x_{33} + x_{12}x_{23}x_{31} + x_{13}x_{22}x_{31} + x_{13}x_{21}x_{32}$$



Comes up a lot. 1800's on: efforts to eff. compute PER(M).

Looks very similar to DET(M):

+1 or -1 value, dep. on π

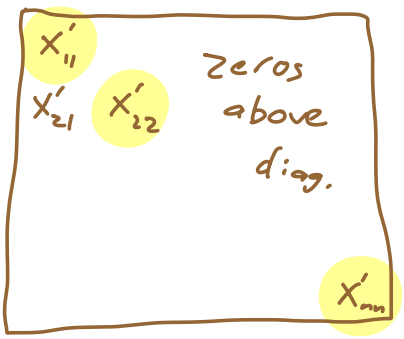
$$DET(M) = \sum_{\substack{\text{all } n! \\ \text{permut.} \\ \pi \text{ of } [n]}} sign(\pi)\, x_{1,\pi(1)} \cdot x_{2,\pi(2)} \cdot \ldots \cdot x_{n,\pi(n)}$$

$[n] = \{1, \ldots, n\}$

Fact: $DET(M)$ is in FP.

(Gaussian elim / row op. to convert M into a diag. mtx M' with same det:

$M' =$



$Det(M') = x'_{11} \cdot \ldots \cdot x'_{nn}$

prod. of diag. elts.

Thm (Valiant): PER is #P-complete.

Next time: • show that PER of 0/1 matr. is in #P.

• Random self reducibility:

PER's worst-case hardness $\Rightarrow$ avg-case hardness.

- approximate counting?