6.890	Lecture 15	Oct. 23, 2014

NP search problem: (≈NP-relation)

- goal: instance → solution (any)
- for each instance, set of (valid/feasible) solutions
- can recognize instances & their solutions in P

- every NP problem → NP search problem
  (for every choice of YES certificates → solutions)

Counting version #A of NP search problem A
count number of solutions for given instance

- e.g. #SAT: find # satisfying assignments
  #Shakashaka: find # solutions to puzzle

#P = {#A | NP search problem A} [Valiant—TCS 1979]
= {problems solved by polynomial-time nondeterministic counting algorithms}

↳ makes guesses, at end says YES or NO (just like an NP algorithm)

↳ output = # guess paths leading to YES

#P-hard = as hard as all problems in #P via multicall (Cook-style) reductions

⇒ ∉ P unless P = NP

↳ technically, FP = poly-time computable functions

Parsimonious reduction for NP search problems

instance x of $A \xrightarrow{f}$ instance x' of B

- computable in polynomial time (like NP reduction)
- $\#$ A solutions to x = $\#$ B solutions to y

  $\Rightarrow$ decision problems ($\exists$ solution?) same answer

  $\Rightarrow$ NP reduction too

- $\#A$ is $\#P$-hard $\Rightarrow$ $\#B$ is $\#P$-hard

c-monious reduction: uniform scaling

$c \cdot \#$ A solutions to x = $\#$ B solutions to y

- preserves 0 $\Rightarrow$ NP reduction too
- $\#A$ is $\#P$-hard $\Rightarrow$ $\#B$ is $\#P$-hard

#P-complete SAT problems:

- #3SAT
- planar #3SAT
- planar monotone rectilinear #3SAT
- planar positive rectilinear #1-in-3SAT

  as in L7

- planar positive #2SAT-3

  [Xia, Zhang, Zhao – TCS 2007]

- Schaefer-style dichotomy:
  - #SAT $\in$ FP $\Leftrightarrow$ system of linear equations (mod 2)
  - #SAT #P-complete otherwise

  [Creignou & Hermann – I&C 2006]

  see [Creignou, Khanna, Sudan – SIGACT 2001]

Shakashaka: parsimonious ⇒ #P-hard
[Demaine, Okamoto, Uehara, Uno – CCCG 2013]

Hamiltonian cycles:
- old proofs not parsimonious [Lichtenstein] [Plesnik]
- parsimonious reduction from 3SAT to planar max-degree-3 Hamiltonian cycle
[Sato – senior thesis 2002]
- nonplanar case solved earlier [Valiant 1974]

Slitherlink: parsimonious ⇒ #P-hard [Yato 2000]
- here can't use grid graphs
⇒ optional vertex gadgets

<u>Determinant</u> of $n \times n$ matrix $A = (a_{ij})$ $\in P$

$$= \sum_{\text{permutation } \pi} (-1)^{\text{sign}(\pi)} \underbrace{\prod_{i=1}^{n} a_{i,\pi(i)}}_{\text{product of permutation matrix within } A}$$

<u>Permanent</u> $= \sum_{\text{permutation } \pi} \prod_{i=1}^{n} a_{i,\pi(i)}$

→ weighted directed n-node graph $w(i,j) = a_{ij}$:

$= \sum \{ \text{product of edge weights} \mid \text{cycle cover} \}$

vertex-disjoint directed cycles hitting all vertices

- #P-complete [Valiant—TCS 1979]
- c-monious reduction from #3SAT
- weight-1 edges in variable & clause gadgets
- special weight matrix X in junctions
  - perm $X = 0$ ⇒ not alone in nonzero cycle cover
    ⇒ entered & exited by bigger cycle
  - perm(X − row & col. 1) = perm(X − row & col. 4) = 0
    ⇒ can't enter & leave immediately
    ⇒ enter at one end (1 or 4), leave at other
  - perm(X − rows & cols. 1 & 4) = 0
    ⇒ can't leave interior 2x2 separate
    ⇒ must be visited between enter & exit
  - perm(X − row 1 − col. 4) = perm(X − row 4 − col. 1) = 4
    factor for each traversal

  ⇒ acts as forced edge in var. & clause gadgets

⇒ perm $= 4^{8 \cdot \#\text{clauses}} \cdot \#$ satisfying assignments

Permanent mod r also #P-hard: [Valiant—TCS 1979]

- multicall reduction from Permanent
- set $r = 2, 3, 5, 7, 11, \ldots$ until product $> M^n \cdot n!$

  largest absolute entry in matrix

- $\Rightarrow O(n \lg M + n \lg n)$ calls & max $r = O(\text{that} \ln \text{that})$
- use Chinese Remainder Theorem [Prime # theorem]

0/1-permanent mod r: [Valiant—TCS 1979]

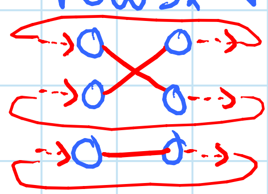
- parsimonious reduction from permanent mod r
  $\Rightarrow$ all edge weights (effectively) nonnegative
- replace weight-k edge ($k > 1$) with gadget with k loops
  - unique solution if original edge unused
  - exactly k solutions if original edge used (using exactly 1 loop)

0/1-permanent: [Valiant—TCS 1979]

- one-call reduction from 0/1-permanent mod r
- call with same input
- return output mod r

= # cycle covers in given directed graph
= # perfect matchings in given bipartite graph

(V_1 = rows, V_2 = columns, $(\underset{V_1}{i}, \underset{V_2}{j}) \in E \Leftrightarrow a_{ij} = 1$)



(balanced: $|V_1| = |V_2|$)

Bipartite # maximal matchings: [Valiant – SICOMP 1977]

- one-call reduction from bipartite # perfect matchings
- replace each vertex with n copies ($n = |V|$)
  & each edge with biclique $K_{n,n}$

  $\Rightarrow$ old matching of size i
  $\rightarrow (n!)^i$ distinct matchings of size ni
  (& preserves maximality)
- # maximal matchings

$$= \sum_{i=0}^{n/2} \underbrace{(\#\text{ orig. maximal matchings size } i)}_{\leq (n/2)! \text{ e.g. } K_{n/2,n/2}} \cdot (n!)^i$$

$\Rightarrow$ can extract # perfect matchings ($i = n/2$)

Bipartite # matchings: [Valiant – SICOMP 1977]

- multicall reduction from bipartite # perfect matchings
- $G \rightarrow G_k$: for each vertex: add k adjacent leaves
- M_r matchings of size $n/2 - r$ in G
  contained in $M_r (k+1)^r$ matchings in G_k

$\Rightarrow$ # matchings in $G_k = \sum_{r=0}^{n/2} M_r (k+1)^r$
- evaluate this polynomial for $k = 1, 2, \ldots, n/2 + 1$

$\Rightarrow$ can extract coefficients $M_0, M_1, \ldots$
- M_0 = desired # perfect matchings in G

Positive #2SAT

[Valiant – SICOMP 1977]

(cf. old VC reduction)

= # vertex covers

⇒ # cliques in complement graph

- parsimonious reduction from bipartite # matchings
- edge → variable: true = not in the matching
- 2 incident edges e & f → clause $e \vee f$

  ⇒ satisfying assignment = matching

Minimal Vertex Covers

[Valiant – SICOMP 1977]

= # maximal cliques in complement graph

= # minimal truth settings for positive 2SAT

- parsimonious reduction from bipartite # maximal matchings, as above
- minimal satisfying assignment = maximal matching

  $|E| - i$ true variables ⇐ size i

3-regular bipartite planar #Vertex Cover

= planar positive 2SAT-3 where each clause has 1 red & 1 blue variable

- #P-complete [Xia, Zhang, Zhao – TCS 2007]

(2,3)-regular bipartite #Perfect Matchings

- #P-complete [Xia, Zhang, Zhao – TCS 2007]

(note: decision versions easy)

Another Solution Problem (ASP) [Ueda & Nagao — TR 1996]

- for NP search problem A:
  ASP A: given one solution, is there another?
- useful in puzzle design: want unique solution

- e.g. ASP k-coloring $\in P$ (rotate colors)
  & ASP 3-regular Hamiltonian cycle $\in P$
  (always another solution)

ASP reduction: parsimonious reduction $A \to B$
& poly.-time bijection between solutions$_A(x)$
& solutions$_B(x')$
- induces every parsimonious reduction we've seen

$\Rightarrow$ ASP A $\to$ ASP B via NP reduction
(can map given solution to A $\to$ sol. to B)
- ASP B $\in$ P $\Rightarrow$ ASP A $\in$ P
- ASP A NP-hard $\Rightarrow$ ASP B NP-hard

ASP-hard = ASP reducible from every NP search prob.
$\Rightarrow$ NP-hard

ASP-complete = ASP-hard NP search problem
- includes planar 3SATs & Hamiltonicity today,
  Shakashaka, Slitherlink
- not c-monius reductions: 2SAT, matchings,
  permanent

6.890 Algorithmic Lower Bounds: Fun with Hardness Proofs
Fall 2014