Lecture 9: Matching using Matrix Methods (Tutte-Lovasz, MVV).

- One useful trick is arithmetization of the problem: cast the problem as a (low degree) polynomial and then use the properties thereof.

- Low-degree polynomials are your friend.

- Here's one fact: Low-degree polynomials have "few" zeroes.
  - (Minor Implication of) Fundamental Theorem of Algebra: Any non-zero univariate polynomial of degree $\leq d$ has $\leq d$ roots (over a field $\mathbb{F}$).
  - But with multiple variables things can be different. Clearly $P(x,y) = xy$ has infinitely many roots. Still, there are "few"; the roots lie on a "low dimensional" space. Here's one formalization of this fact.

Schwartz-Zippel Lemma (also Lipton-De Millo)

Let $P(x_1 \dots x_n)$ be a polynomial ~~over~~ in n variables over a field $\mathbb{F}$.

Let $P(X_1 \dots X_n)$ be a polynomial, and $P \not\equiv 0$ (not identically zero).

Let d be the degree of P, and

For $S \subseteq \mathbb{F}$, ~~Suppose $X_1, X_2 \dots X_n$ are picked~~ suppose $X_1, X_2 \dots X_n$ are picked uniformly ~~#roots of~~ and indep from S. then

$$\Pr\left[P(\bar{X}) = 0\right] \leq \frac{d}{|S|}.$$

Pf: by induction on n. the case $n=1$ is the F.T. of A.

Let $P(\bar{x}) = x_1^k Q(x_2 \dots x_n) + R(x_1 \dots x_n)$ where k = largest exponent of x_1 in any monomial

Now: choose $X_2 \dots X_n$ and say $\mathcal{B}$ is "bad" event that $Q(X_2 \dots X_n) = 0$.

$$\Rightarrow \Pr[P(X) = 0] = \Pr[P(X) = 0 \mid \mathcal{B}] \Pr[\mathcal{B}] + \Pr[P(X) = 0 \mid \neg\mathcal{B}] \Pr[\neg\mathcal{B}]$$

$$\leq \Pr[\mathcal{B}] + \Pr[P(X) = 0 \mid \neg\mathcal{B}]$$

$$\leq \frac{d-k}{|S|} + \frac{k}{|S|} \leq \frac{d}{|S|}$$

just a degree k univariate poly

$\blacksquare$

Hence: if $|S| = 2d \Rightarrow$ ~~w.p. 1/2 check polynomial is z~~

$$\left.\begin{array}{l} \text{if } P \equiv 0 \Rightarrow (\text{Answer} = \text{zero}) \text{ w.p. } 1 \\ \phantom{\text{if }} P \not\equiv 0 \Rightarrow (\text{Answer} = \text{zero}) \text{ w.p. } \le 1/2. \end{array}\right\} \text{can boost success probability by repetitions.}$$

A quick application [due to Lovasz]. Given bipartite graph $G = (L, R, E)$, $|L| = |R| = n$.

Consider the matrix $(\mathbb{E})_{ij} = \begin{cases} x_{ij} & \text{if } (ij) \in E \\ 0 & \text{if } (ij) \notin E. \end{cases}$

"Edmonds" matrix

Fact: consider $\det(\mathbb{E})$. this is a polynomial in $\le n^2$ variables. of degree $\le n$.

And $\det(\mathbb{E}) \not\equiv 0$ if and only if E has a perfect matching

Pf: if $\exists M$ then $\pm x^M$ is a monomial which is not cancellable by anything else.

$\Rightarrow$ [Lovasz] Also: use PolyIDTest on $\det(E)$ to check if $E \not\equiv 0$?

Requires us to compute determinants on the random entries. Can be done in $O(n^\omega)$ time. ~~Repeat $O(\log n)$ times~~ to ensure error probability $\le \frac{1}{n^3}$. $d/|S|$

use a set size $O(n^4)$

$\Rightarrow$ Thm: in time $O(n^\omega)$,

can answer "does G (bipartite) have a perfect matching"

correctly w.p. $1 - 1/n^3$.

Finding the PM: · FindPM(G) ← invariant: G has a PM (previous test said YES).

~~test if G has PM, if not, return No.~~

For $e = (u, v)$ in G, test if $G - e$ has PM.

if YES, findPM($G - e$).

else // e is in PM.

Return e + findPM($G - \{u, v\}$).

N.b. we call the tester on ~~$O(m)$~~ $\le m$ edges

$\Rightarrow$ by a union bound, mess up wp $\le m/n^3 \le 1/n$.

$\Rightarrow$ <u>Thm</u>: this algorithm finds a PM in a bipartite graph in time $O(mn^\omega)$.

<u>Extending to non-bipartite graphs</u>: $G=(V,E)$ $|V|=n$.

the Tutte Matrix $n \times n$

$$T_{ij} = \begin{cases} X_{ij} & \text{if } i<j \text{ and } ij \in E \\ -X_{ji} & \text{if } i>j \text{ and } ij \in E \\ 0 & \text{if } i=j \text{ or } ij \notin E \end{cases}$$

<u>Theorem</u>: det$(T) \not\equiv 0$ iff T has a perfect matching.

This does the general graph PM problem in time $O(n^\omega \cdot m)$ as well.

Here's a problem we don't know how to do deterministically, but this technique gives us a randomized algorithm.

- <u>Red Blue Matching</u>: Given a graph $G=(V,E)$ with edges colored red & blue, ~~find~~ and an integer K, ~~find~~ does $\exists$ a perfect matching containing <u>exactly</u> K red edges?

Let's assume that ~~edges have weights and~~ $\exists$ <u>unique</u> such red/blue PM (if one exists). Also <u>bipartite</u>.

Set $M_{ij} = \begin{cases} 0 & \text{if } (u_i,v_j) \notin E \\ 1 & \text{if } (u_i,v_j) \in E \text{ (blue)} \\ y & \text{if } (u_i,v_j) \in E \text{ (red)}. \end{cases}$

Consider det(M). It is a polynomial in y of degree $\le n$. And the coeff of y^K is exactly 1 iff $\exists$ such a red/blue ~~and~~ PM.

How to ~~evaluate~~ find the symbolic polynomial $\det(M)$?

Use interplation. If we know the value of polynomial at $\deg(P)+1$ points, we can find it by Lagrange interpolation, say.

Of course, we made a big assumption of uniqueness.

So, now we'll say :-

① if we assign weights randomly, whp ($\frac{1}{2}$) $\exists$ a unique min-wt red/blue PM.
from $[1..2m]$

② now define $M = \begin{cases} 0 & ij \notin E \\ 2^{w_{ij}} & ij \in \text{blue edge} \\ 2^{w_{ij}} y & ij \in \text{red edge} \end{cases}$

Now $\underbrace{\det(M)[y^k]}_{\text{coeff of } y^k} = 2^{\text{min wt red-blue}} \cdot \text{odd} \neq 0$. cannot all cancel out

$\Rightarrow$ just checking the coefficient of y^k for non-zeroness works wp $\geq \frac{1}{2}$.

And again, use interpolation to find $\det(M)$.

Numbers $\leq 2^{\text{poly}(n)} \Rightarrow \text{poly}(n)$ bits, which is OK.

Proof of ①:

- Isolation Lemma: Suppose $\mathcal{S} \subseteq 2^E$, where $|E| = m$. Assign weights from $[1..cm]$ to elements of E u.a.r. Then $\Pr[\exists \text{ a unique min wt set in } \mathcal{S}] \geq 1 - \frac{1}{c}$.

Pf: in HW.

~~Suppose~~

Previous approach: use $O(mn^{\omega})$ time. Can we do faster?

Observation: suppose $\det(\hat{\mathcal{E}}) \neq 0$. there must be at least one matching π s.t
(instantiated with random values)

$$\boxed{\hat{\mathcal{E}}_{i\pi(i)} \neq 0 \quad \forall i \in L}$$

How to find ~~such~~ a permutation? [Rabin–Vazirani $O(n^{\omega+1})$ time].

~~Compute $[\hat{\mathcal{E}}]^{-1}$. [can be done in time $O(n^{\omega})$]~~

- Find location j st $\hat{\mathcal{E}}_{1j} \neq 0$ and $\det(\hat{\mathcal{E}}$ with row 1 and col j removed) $\neq 0$.
  (the $(1,j)$ minor)
  ↳ recursively maps $\pi' : [2..n] \to [n] \setminus \{j\}$.

  and set $\pi(1) = j$
  $\pi(i) = \pi'(i) \; \forall i \in [2..n]$.

How? Naively: have to compute det for each j st $\hat{\mathcal{E}}_{1j} \neq 0$. (again $\deg(v_1)$ time).

Better: Compute $\hat{A}^{-1}$!

$$\boxed{\text{Recall} \quad (A^{-1})_{pq} = \frac{\det(A_{-q,-p}) \cdot (-1)^{p+q}}{\det(A)} \quad \forall p,q}$$

$\Rightarrow$ in one shot get all $\det(\hat{\mathcal{E}}_{-1,-j})$.

Scan to find out j st $\hat{\mathcal{E}}_{1j} \neq 0$ $\det(\hat{\mathcal{E}}_{-1,-j}) \neq 0$.

time: $O(n^{\omega})$ for inverse: $O(n)$ for scan)

recurse.

$\Rightarrow O(n^{\omega+1})$.

[Also for non-bipartite!]
(Main contrib of RV) ☺
needs another idea [MW?]

Recall: $A^{-1} = \dfrac{\text{adjugate}(A)}{\det(A)}$ ← transpose of cofactor matrix (A)

$(\text{cof}(A))_{pq} = (-1)^{p+q} \det(A_{-p,-q})$.
(minor)

Bunch and Hopcroft (Notes)

- Sps we want to compute det(A) in time $O(n^\omega)$. (Assume $\omega > 2$)

say $A = \begin{pmatrix} a_1 & v_1^T \\ u_1 & B_1 \end{pmatrix}$ then gaussian elimination makes it $\begin{pmatrix} a_1 & 0 \\ 0 & B_1 - \frac{u_1 v_1^T}{a_1} \end{pmatrix}$

(say $a_1 > 0$ so we don't have to reorder things) → get rid of a different column in row 1, and reorder etc.

[Recall that this operation does not change the determinant.]

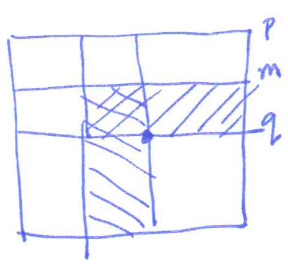
Of course this update requires n^2 time $\Rightarrow O(n^3)$ overall.

The basic idea: ~~some~~ the updates on the ~~far right columns~~ bottom rows are not required until we get to them. So "bunch" them and apply later.

Sps we don't really do any pivoting, always a_{ii} is non zero.

Then: BH(p,q)   rows $p \le q$: – will call with BH(1,n).

{
if p=q then "Lazily" do update for row/col p. (i.e. want to subtract $\frac{u_p v_p^T}{a_p}$ from B_p)
else
  $m = \frac{p+q}{2}$
  BH(p,m)
  batch apply the lazy updates for ~~A(p..n, p..n)~~ A(p..m, p..n)
  —— " —— for A(p..n, p..m)
  BH(m,q)
}



What is the batch update?

We wanted to apply $c_1 \check{u}_1 \check{v}_1^T + c_2 \check{u}_2 \check{v}_2^T \cdots + c_t \check{u}_t \check{v}_t^T$

these are subvectors of the actual vectors we wanted to do.

but this is $\begin{pmatrix} | & | & & | \\ c_1 u_1 & c_2 u_2 & \cdots & c_t u_t \\ | & | & & | \end{pmatrix} \begin{pmatrix} — & v_1^T & — \\ — & v_2^T & — \\ & \vdots & \\ — & v_t^T & — \end{pmatrix}$

can be done by fast matrix multiplication.

Now: we apply about n/x updates of "size x" i.e. multiplying $(x \times x) \times (x \times n)$ which can be done by $(x \times x)$ square matrix mult n/x times.

$$\Rightarrow \text{total work} = \sum_{x \text{ power of } 2} (n/x) \times \text{Rect mm}((x \times x) \times (x \times n))$$

$$= \sum_{x \text{ power of } 2} (n/x) \times (n/x) \times x^{\omega} = \sum_{x \text{ power of } 2} n^2 . x^{\omega-2} = O(n^{\omega})$$

if $\omega > 2$ constant.

If pivoting, need to be careful and sign will change, but idea is the same

---

[See Marcin Mucha's thesis for a readable explanation.]

— x —

How about the matrix inverse?