EPFL

Exercise Set VIII

These exercises are for your own benefit. Feel free to collaborate and share your answers with other students. Solve as many problems as you can and ask for help if you get stuck for too long. Problems marked * are more difficult but also more fun :).

These problems are taken from various sources at EPFL and on the Internet, too numerous to cite individually.

1 Consider an n-by-n bipartite graph $G = (X \cup Y, E)$ and let A be the adjacency matrix where we have $A_{ij} = \begin{cases} x_{ij} & \text{if } \{i,j\} \in E \\ 0 & \text{otherwise} \end{cases}$ (as in Lecture 19). In that lecture, we saw that we could replace each x_{ij} by a random number and check whether $\det(A) \neq 0$ to see whether G has a perfect matching. A beautiful alternative approach was introduced by a very influential paper by Mulmuley, Vazirani, and Vazirani'87 who considered *isolating weight functions*.

1a A weight function $w : E \to \mathbb{N}$ is *isolating* if the min-weight perfect matching is unique. Show that if w is isolating then

$$\det(\hat{A}) \neq 0 \Leftrightarrow G \text{ has a perfect matching},$$

where $\hat{A}$ is the matrix obtained from A by replacing each variable x_e by $2^{w(e)}$.

Solution: By the definition of determinant we have

$$\det(\hat{A}) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} \hat{A}_{i,\sigma(i)}$$

where S_n is the permutation group on n elements and $\operatorname{sgn}(\sigma)$ is the sign of the permutation σ (here it only matters that it is ± 1). Notice that nonzero terms in the above sum correspond to permutations σ for which each $\{i, \sigma(i)\}$ is an edge in G – i.e., to matchings. From this it is already clear that if G has no perfect matchings, then $\det(\hat{A}) = 0$. Let us prove the converse.

Note that each such nonzero term is equal to $\pm \prod_{i=1}^{n} \hat{A}_{i,\sigma(i)} = \pm \prod_{i=1}^{n} 2^{w(i,\sigma(i))} = \pm 2^{w(M)}$ where $M = \{\{i, \sigma(i)\} : i = 1, ..., n\}$ is the matching corresponding to σ. Therefore

$$\det(\hat{A}) = \sum_{M \,:\, \text{matching in } G} \operatorname{sgn}(M) 2^{w(M)}$$

where $\operatorname{sgn}(M) \in \{-1, 1\}$.

Now, if w is isolating, then there is a matching M_0 such that for each other matching $M \neq M_0$ we have $w(M_0) < w(M)$. Therefore the corresponding term $\pm 2^{w(M_0)}$ will not cancel out with any other term $\pm 2^{w(M)}$ (because the latter term is a multiple of $2^{w(M_0)+1}$; in other words, $\det(\hat{A})$ is nonzero even modulo $2^{w(M)}$). Thus $\det(\hat{A}) \neq 0$.

1b *(*)* In this exercise, we prove the "Isolation Lemma". Show that if we form the weight function w by letting, for each edge e independently, $w(e)$ be a random number in $\{1, \ldots, 2|E|\}$, then the probability that w is isolating is at least $1/2$.

(Hint: for each edge e, consider $\alpha_e = \min_{M \ni e} w(M \setminus \{e\})$ and $\beta_e = \min_{M \not\ni e} w(M)$. What is the probability that $\alpha_e + w(e) = \beta_e$?)

Solution: Note that $\alpha_e + w(e) = (\min_{M \ni e} w(M) - w(e)) + w(e) = \min_{M \ni e} w(M)$. Let us suppose that w is not isolating. Then there exists an edge e which belongs to some min-weight perfect matching but not to every min-weight perfect matching.[1] For such an edge e, $\min_{M \ni e} w(M) = \min_{M \not\ni e} w(M)$, i.e., $\alpha_e + w(e) = \beta_e$. So

$$P[w \text{ not isolating}] \leq P\left[\bigcup_e \{\alpha_e + w(e) = \beta_e\}\right] \leq \sum_e P[\alpha_e + w(e) = \beta_e].$$

But

$$P[\alpha_e + w(e) = \beta_e] = P[w(e) = \beta_e - \alpha_e] \leq \frac{1}{2|E|}$$

since both β_e and α_e are independent of $w(e)$.[2] So $P[w \text{ not isolating}] \leq 1/2$.

2 Polynomial Identity Testing. Let $p_0(\mathbf{x}), \ldots, p_m(\mathbf{x})$ be multivariate polynomials of degree at most d over n variables $\mathbf{x} = (x_1, \ldots, x_n)$. Further, let $q(\mathbf{x}, y)$ be a multivariate polynomial over $n+1$ variables $\mathbf{x}, y$ defined as

$$q(\mathbf{x}, y) = \sum_{i=0}^{m} p_i(\mathbf{x}) \cdot y^i \,.$$

In this problem, you are given *oracle access* to $q(\mathbf{x}, y)$: that is, for any $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{R}^n$ and $\gamma \in \mathbb{R}$, you can query the value $q(\boldsymbol{\alpha}, \gamma)$ and learn it in constant time.

Your task is to design an efficient randomized algorithm which receives as input an integer $t \in \{0, \ldots, m\}$ and determines whether the t-th polynomial p_t is the zero polynomial. More precisely, your algorithm should run in time $\text{poly}(d, n, m)$ and have the following properties:

- if $p_t = 0$, then the algorithm outputs `ZERO` with probability 1;
- if $p_t \neq 0$, then the algorithm outputs `NON-ZERO` with probability at least $2/3$.

[1] There are two different matchings $M \neq M'$ which both have minimum weight $w(M) = w(M')$. Pick any edge e in their symmetric difference.

[2] For die-hard fans of formality (and conditional expectations):

$$\begin{aligned}
P[w_e = \beta_e - \alpha_e] &= \mathbb{E}[\chi_{w_e = \beta_e - \alpha_e}] \\
&= \mathbb{E}[\mathbb{E}[\chi_{w_e = \beta_e - \alpha_e}] | w(e') : e' \in E \setminus \{e\}] \\
&= \mathbb{E}\left[\begin{cases} 1/(2|E|) & \text{if } \beta_e - \alpha_e \in \{1, 2, ..., 2|E|\} \\ 0 & \text{otherwise} \end{cases}\right] \\
&\leq \mathbb{E}[1/(2|E|)] \\
&= 1/(2|E|)
\end{aligned}$$

where conditionally on $w(e') : e' \in E \setminus \{e\}$ we have that $\beta_e - \alpha_e$ is a constant (i.e., it is a function of only $w(e') : e' \in E \setminus \{e\}$ and not of $w(e)$).

You may use the fact that for any $k+1$ pairs of numbers $((u_i, v_i))_{i=0}^k$ where all the u_i's are distinct, there always exists a unique univariate polynomial $f(z) = \sum_{i=0}^k c_i z^i$ of degree at most k such that $f(u_i) = v_i$ for all $i = 0, \ldots, k$. Moreover, you can assume that given $((u_i, v_i))_{i=0}^k$ you can compute the coefficients $c_0, \ldots, c_k$ of $f(z)$ in poly(k) time.

Solution: The algorithm is the following.

1. Let X be a set of at least $3d$ distinct numbers.
2. Sample $\boldsymbol{\alpha} \sim \text{unif}(X^n)$.
3. Let $Y = \{\gamma_0, \ldots, \gamma_m\}$ be a set of $m+1$ distinct numbers.
4. For each $i = 0, \ldots, m$, let $r_i = q(\boldsymbol{\alpha}, \gamma_i)$.
5. From the pairs $\{(\gamma_i, r_i)\}_{i=0}^m$ find the coefficients $c_0, \ldots, c_m$ of the unique polynomial $q_{\boldsymbol{\alpha}}$ of degree at most m such that $q_{\boldsymbol{\alpha}}(\gamma_i) = r_i$ for all $i = 0, \ldots, m$.
6. If $c_t = 0$ then output `ZERO`, otherwise output `NON-ZERO`.

Observe that $c_t = p_t(\boldsymbol{\alpha})$ by interpolation, so the algorithm outputs `ZERO` if and only if $p_t(\boldsymbol{\alpha}) = 0$. Moreover, if $p_t = 0$ then $p_t(\boldsymbol{\alpha}) = 0$ for all $\boldsymbol{\alpha}$, otherwise if $p_t \neq 0$ then $p_t(\boldsymbol{\alpha}) \neq 0$ with probability at least $1 - d/|X| \geq 2/3$ over the choice of $\boldsymbol{\alpha}$ by the Schwartz-Zippel Lemma. Hence, the algorithm outputs `ZERO` with probability 1 if $p_t = 0$, and it outputs `NON-ZERO` with probability at least $2/3$ if $p_t \neq 0$.

We can form a set X with at least $3d$ distinct numbers so that $|X| = O(d)$ and $|X| = 2^a$ for some positive integer a, which can be done in $O(d)$ time. Then, we can sample $\boldsymbol{\alpha}$ in $O(n \log |X|) = O(n \log d)$ time. We can form the set Y in $O(m)$ time. The algorithm performs $O(m)$ queries to $q(\mathbf{x}, y)$ and does one interpolation of degree m. Overall, the running time is $O(d + n \log d + m) + \text{poly}(m) = \text{poly}(d, n, m)$.

3 Exact Spanning Tree. In the **exact spanning tree problem**, you are given an undirected connected graph G and a value k and are interested in determining whether G has a spanning tree of weight exactly k.

Design and prove the correctness of an algorithm that:

- **takes as input**
  - an undirected weighted connected graph $G = (V, E, w)$ with non-negative integer edge weights given by $w : E \to \mathbb{Z}_{\geq 0}$, and
  - a positive integer $k \in \mathbb{N}$;
- **outputs**
  - `yes` if there exists a spanning tree $T \subseteq E$ of G such that $\sum_{e \in T} w(e) = k$,
  - `no` otherwise.

Your algorithm should run in time polynomial in the size of the graph G and in the maximum edge weight. In other words, the running time of your algorithm should be upper-bounded by poly$(|V|, \max_{e \in E} w(e))$.

We state here a result from Graph Theory that you are allowed to use without proof to solve this problem. Given an undirected weighted connected graph $H = (V, E, c)$ with edge weights given by $c : E \to \mathbb{R}$, the weighted Laplacian of H is the matrix $L_c \in \mathbb{R}^{V \times V}$ defined as

$$(L_c)_{u,v} \begin{cases} -c(u,v) & \text{if } (u,v) \in E \\ \sum_{(u,x) \in E} c(u,x) & \text{if } u = v \\ 0 & \text{otherwise} \end{cases} .$$

We also denote by L'_c the matrix obtained from L_c by removing its first row and column. With this notation in place, the theorem is the following.

Theorem 1 (Weighted Matrix-Tree Theorem)**.** *Let $H = (V, E, c)$ be an undirected weighted connected graph with edge weights given by $c : E \to \mathbb{R}$. Then,*

$$\sum_{\substack{T \subseteq E: \\ T \text{ is a spanning tree of } H}} \prod_{e \in T} c(e) = \det(L'_c) .$$

Solution: The input consists of an undirected weighted connected graph $G = (V, E, w)$ with non-negative integer edge weights given by $w : E \to \mathbb{Z}_{\geq 0}$ and a positive integer $k \in \mathbb{N}$. The algorithm is the following.

- Let $w_{\max} = \max_{e \in E} w(e)$.
- Let $d = (n-1) \cdot w_{\max}$.
- Let $S = \{\chi_0, \ldots, \chi_d\} \subseteq \mathbb{R}$ consisting of $d+1$ distinct positive real numbers.
- For each $i = 0, \ldots, d$ do the following.
  1. Let $c : E \to \mathbb{R}$ be a weight function defined as $c(e) = \chi_i^{w(e)}$ for each $e \in E$.
  2. Let $H = (V, E, c)$ be a graph with the same vertex and edge set as G and weight function c.
  3. Let $r_i = \det(L'_c)$.
- Interpolate the points $((\chi_i, r_i))_{i=0}^d$ to determine the coefficients $\hat{a}_0, \ldots, \hat{a}_d \in \mathbb{R}$ of the unique degree-d polynomial $\hat{p} \in \mathbb{R}[x]$ such that $\hat{p}(\chi_i) = r_i$ for all $i = 0, \ldots, d$.
- Output `yes` if $\hat{a}_k \neq 0$, otherwise output `no`.

Running time. Using for example Gaussian elimination, we can compute the determinants in time $O(|V|^3)$ and solve the linear system for interpolation in time $O(d^3)$. Hence, the running time of the algorithm is upper-bounded by $\text{poly}(|V|, k, \max_{e \in E} w(e))$.

Correctness. Let $p \in \mathbb{R}[x]$ be the polynomial defined as

$$p(x) \equiv \sum_{i=0}^{d} a_i \cdot x^i .$$

where for each $i = 0, \ldots, d$ we let

$$a_i = |\{T \subseteq E : T \text{ is a spanning tree of } G \text{ and } w(T) = i\}| .$$

Note that for any $\chi \in \mathbb{R}$ we have

$$\begin{aligned}
p(\chi) &= \sum_{i=0}^{d} a_i \cdot \chi^i \\
&= \sum_{i=0}^{d} \sum_{\substack{T \subseteq E\colon \\ T \text{ is a spanning tree of } G \\ \text{and} w(T)=i}} \chi^i \\
&= \sum_{i=0}^{d} \sum_{\substack{T \subseteq E\colon \\ T \text{ is a spanning tree of } G \\ \text{and} w(T)=i}} \prod_{e \in T} \chi^{w(e)} \\
&= \sum_{\substack{T \subseteq E\colon \\ T \text{ is a spanning tree of } G}} \prod_{e \in T} \chi^{w(e)} \\
&= \det(L'_\gamma)\,,
\end{aligned}$$

where the last equality follows by the Weighted Matrix-Tree Theorem with $\gamma : E \to \mathbb{R}$ defined as $\gamma(e) = \chi^{w(e)}$ for each $e \in E$.

By the above discussion, we conclude that for each $i = 0, \ldots, d$, the value r_i computed by the algorithm is equal to $p(\chi_i)$. Hence, we have that $\hat{p} \equiv p$, where $\hat{p}$ is the polynomial found by the algorithm through interpolation. In particular, the coefficient vector $(a_0, \ldots, a_d)$ of p equals the coefficient vector $(\hat{a}_0, \ldots, \hat{a}_d)$ of $\hat{p}$. Noticing that $a_k \neq 0$ if and only if there exists a spanning tree $T \subseteq E$ of G such that $\sum_{e \in T} w(e) = k$, we conclude that the algorithm is correct.