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 $2^{w(e)}$.

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 $\alpha_e + w(e) = \beta_e$?)

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 $\mathrm{poly}(d, n, m)$ 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 $2/3$.

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 $\mathrm{poly}(k)$ time.

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 $\mathrm{poly}(|V|, \max_{e \in E} w(e))$.

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,*

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