6.S078: Fixed-Parameter and Fine-Grained Computation | **MIT**
Problem Set 3 | **Due: May 4, 11:59pm (Friday)**
Ryan and Virginia | **Spring 2018**

Rules: You may collaborate with other students, but please do not consult research papers, textbooks, or the internet in general. All submitted solutions must be your own, written in your own words. Write down the set of students you collaborated with, at the top of your homework submission. Email your pset to `rrw@mit.edu` **AND** `virgi@mit.edu`

1 Linear Decision Tree for Max-Min Product

The Max-Min product of two matrices $A, B \in \mathbb{R}^{n \times n}$ is the matrix C with $C[i,j] = \max_k \min\{A[i,k], B[k,j]\}$. Give an $O(n^2 \log n)$-depth linear decision tree for the Max-Min product of two $n \times n$ matrices.

2 Counting Colorful Paths

In the lecture on algorithms for Hamiltonian Path, we saw how to use inclusion-exclusion to count the number of Hamiltonian paths in $O^\star(2^n)$ time. You can use this counting principle to get a space-efficient k-path algorithm.

(a) Given a graph with a coloring $c : V \to [k]$ on its nodes, show how to count the number of colorful k-paths in $O^\star(2^k)$ time and $\text{poly}(n,k)$ space.
(b) Use color-coding to conclude that there is a (randomized) algorithm for k-path running in $O^\star((2e)^k)$ time and $\text{poly}(n,k)$ space.

Side note: Although this counting problem is FPT, it turns out that counting *all* k-paths is W[1]-hard!

3 Dynamic Bipartite Matching

The Dynamic Bipartite Matching (DBM) problem is as follows. One is given a bipartite graph $G = (V, E)$ with $|V| = n, |E| = m$ and one can preprocess it in $p(n,m)$ time, building a data structure that can support the following:

- Edge updates: insert(u,v) that inserts an edge (u,v) (with the promise that G stays bipartite), and delete(u,v) which deletes (u,v) from E. Let's say that the time for edge updates is $u(n,m)$.
- At any point of the update sequence one can query matching-size() and the data structure needs to return the size of the maximum matching in G.

Now suppose there's a data structure for DBM with preprocessing time $p(n,m)$, update time $u(n,m)$ and query time $q(n,m)$. We'll show how to use the data structure to solve the Triangle Detection.

Suppose we're given a graph $H = (V_H, E_H)$ with $|V_H| = n, |E_H| = m$, and want to know if it's triangle-free.

You can assume that $m \geq n - 1$.

(a) Show how to construct a graph $G = (V, E)$ with $|V| = O(n)$ and $|E| = O(m)$ and a sequence of $O(n)$ edge insertions, deletions and queries to DBM so that after their execution one can determine in $O(1)$ time whether H contains a triangle.
Hints: As usual for triangle problems (e.g. as in Problem Set 1), you can assume that H is tripartite with vertex partitions A, B, C so that we want to detect whether there is a triangle with $a \in A, b \in B, c \in C$.
To create the initial graph G, take H, split part A into two copies: A_1 and A_2, where A_1 retains the edges in $A \times B$ and A_2 retains the edges in $A \times C$. Now, we have a new graph H' that has 4 layers A_1, B, C, A_2 and

the edges of H' are between consecutive layers. The problem becomes to detect some $a \in A$ so that there is a path on 3 edges between the two copies of a, $a_1 \in A_1$ and $a_2 \in A_2$.

To create G out of H', try to modify H' so that (1) each vertex v of H' is represented by an edge $e(v)$ in G, (2) $\cup_{v \in E(H')} e(v)$ forms a perfect matching in G, and (3) if we remove $e(a_1)$ and $e(a_2)$ from G for the copies $a_1 \in A_1$ and $a_2 \in A_2$ of some $a \in A$, then G has a perfect matching if and only if there is a triangle in H containing a.

The sequence of insertions and deletions performed on G should consist of stages, one for each vertex $v \in V_H$. Each stage should be of the form: "delete and insert a constant number of edges, query the size of the max matching, undo the deletions." From the size of the max matching in the stage for $v \in V_H$, you should be able to tell whether there is a triangle in H going through v.

(b) Conclude that if triangle detection requires $(n+m)^{1+\delta-o(1)}$ time for some $\delta > 0$, then

$$p(n,m) + n \cdot (u(n,m) + q(n,m)) \geq (n+m)^{1+\delta-o(1)}.$$