6.S078: Fixed-Parameter and Fine-Grained Computation **MIT**
Problem Set Due: April 5, 11:59pm (Thursday) **Due: April 5, 11:59pm (Thursday)**
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 both rrw and virgi at mit.edu

You are strongly encouraged to at least look at the problem set before Spring Break!

1 Listing versions of 3SUM, OV and Negative Triangle

(a) The **All-Ints** 3**SUM** problem is as follows. Given a set S of n integers, for each $s \in S$, output 1 if there exist $a, b \in S$ with $a + b + s = 0$ and 0 otherwise. (That is, the output consists of n bits, where $n = |S|$.)

Show that All-Ints 3SUM is subquadratically equivalent to 3SUM, i.e. give two fine-grained reductions, one from 3SUM to All-Ints 3SUM and one from All-Ints 3SUM to 3SUM, implying that if one of these problems has an $O(n^{2-\varepsilon})$ time algorithm for $\varepsilon > 0$, then the other problem has an $O(n^{2-\delta})$ time algorithm for some $\delta > 0$.

(b) The **All-Vectors OV** problem is as follows. Given two sets $U, V \subseteq \{0,1\}^d$ of vectors, for each $u \in U$, output 1 if there is a $v \in V$ with $u \cdot v = 0$, and 0 otherwise. (n bit output, where n is the number of vectors)

Show that All-Vectors OV is subquadratically equivalent to OV, i.e. give two fine-grained reductions, one from OV to All-Vectors OV and one from All-Vectors OV to OV, implying that if one of these problems has an $n^{2-\varepsilon}\text{poly}(d)$ time algorithm for $\varepsilon > 0$, then the other problem has an $n^{2-\delta}\text{poly}(d)$ time algorithm for some $\delta > 0$.

(c) The **All-Nodes Negative Triangle** problem is as follows. Given a graph $G = (V, E)$ with integer edge weights $w(\cdot,\cdot)$, for each $v \in V$, output 1 if there are some $a, b \in V$ so that $(a, b), (b, v), (v, a) \in E$ and $w(a, b) + w(b, v) + w(v, a) < 0$, and output 0 otherwise. (n bit output, where n is the number of nodes.)

Show that All-Nodes Negative Triangle is subcubically equivalent to Negative Triangle, i.e. give two fine-grained reductions, one from Negative Triangle to All-Nodes Negative Triangle and one from All-Nodes Negative Triangle to Negative Triangle, implying that if one of these problems has an $O(n^{3-\varepsilon})$ time algorithm for $\varepsilon > 0$, then the other problem has an $O(n^{3-\delta})$ time algorithm for some $\delta > 0$.

2 APSP is Equivalent to $(\min, +)$-Product

In class we reduced $(\min, +)$-Product to APSP, so that if APSP can be solved in $T(n)$ time in n-node graphs, $(\min, +)$-Product of $n \times n$ matrices can be solved in $O(T(n))$ time. We also reduced APSP to $(\min, +)$-Product, showing that a $T(n)$ time algorithm for $(\min, +)$-Product can be converted into a $O(T(n) \log n)$ time algorithm for APSP. Here we will remove the $\log n$ factor.

Notation: For an $n \times n$ matrix M and subsets $S, T \subseteq [n]$, let $M(X, Y)$ denote the submatrix of M composed from the rows indexed by X and columns indexed by Y.

For a graph $G = (V, E)$ and a subset $S \subseteq V$, let $G[S]$ be the subgraph of G induced by the vertices in S. Let D_G be the distance matrix of G: i.e. $D_G[u, v]$ is the distance in G between u and v.

APSP and $(\min, +)$**-Product Equivalence**: Let $G = (V, E)$ be an instance of APSP with weights $w(\cdot,\cdot)$; G is a directed graph. Let A be the generalized adjacency matrix of G with $A[u, v] = w(u, v)$ if $(u, v) \in E$, $w(u, u) = 0$ and $w(u, v) = \infty$ if $u \neq v$ and $(u, v) \notin E$.

Split V roughly in equal (disjoint) parts X and Y, so that $|X| = |Y| = |V|/2 = n/2$.

Consider the matrix $\bar{Z} = (A(X,Y) \star D_{G[Y]} \star A(Y,X) \star D_{G[X]})$. It defines a new weighted graph Z on node set X. Similarly, consider the matrix $\bar{Q} = (A(Y,X) \star D_{G[X]} \star A(X,Y) \star D_{G[Y]})$. It defines a new weighted graph Q on node set Y.

Below you will show how to compute D_G, given $A, D_{G[X]}, D_{G[Y]}, D_Z, D_Q$.

(a) Show that $D_G(X,X) = \min\{D_{G[X]}, D_{G[X]} \star D_Z\}$, where the minimum is taken componentwise.

(b) Relate $D_G(X,Y), D_G(Y,X), D_G(Y,Y)$ to $A, D_{G[X]}, D_{G[Y]}, D_Z, D_Q$.

(c) Assume that there is some constant $c > 2$ such that for all $\ell \geq 1$ and all n, $T(n/\ell) \leq T(n)/\ell^c$. Conclude that if $(\min, +)$ product is in $T(n)$ time, then APSP is in $O(T(n))$ time.

3 Probabilistic Polynomials and Equality Product of Matrices

In lecture, we noted that using -1 for true and 1 for false could potentially yield better algorithms via probabilistic polynomials. In this problem we will work through an example of this phenomenon. (Recall a probabilistic polynomial is just a distribution of polynomials over a common set of variables.)

Let $S_m = \{0,1\}^m$. The *equality product* of two matrices $A, B \in S_m^{n \times n}$ is the $n \times n$ Boolean matrix C such that for all i, j,

$$C(i,j) = 1 \iff \text{there is a } k \text{ such that } A[i,k] = B[k,j].$$

The equality product has been a useful primitive in the design of algorithms for other graph and matrix problems. We will use probabilistic polynomials over $-1/1$ to derive an $O(n^{3-\delta})$-time algorithm for $\delta > 0$ for the problem.

(a) Define the $AND : \{-1,1\}^n \to \{0,1\}$ function, which outputs 1 if all inputs are -1, and 0 otherwise. Define a probabilistic polynomial on $x_1, \ldots, x_n$ as follows: Pick a random subset $S \subseteq [n]$, and output the polynomial

$$p_S(x_1, \ldots, x_n) = \left(1 + (-1)^{|S|} \cdot \prod_{i \in S} x_i\right).$$

Note these polynomials p_S have only two monomials! Prove:

- For all $x \in \{-1,1\}^n$, if $AND(x) = 1$ then $\Pr_S[p_S(x) = 2] = 1$.
- For all $x \in \{-1,1\}^n$, if $AND(x) = 0$ then $\Pr_S[p_S(x) = 0] \geq 1/2$.

Hint: Remember the "XOR trick"!

(b) Let's amplify part (a). Take the probabilistic polynomial defined by taking random subsets $S_1, \ldots, S_k \subseteq [n]$ and outputting

$$p_{S_1,\ldots,S_k}(x_1, \ldots, x_n) = \prod_{j=1}^{k} \left(1 + (-1)^{|S_j|} \cdot \prod_{i \in S_j} x_i\right).$$

Prove:

- For all $x \in \{-1,1\}^n$, if $AND(x) = 1$ then $\Pr_{S_1,\ldots,S_k}[p_{S_1,\ldots,S_k}(x) = 2^k] = 1$.
- For all $x \in \{-1,1\}^n$, if $AND(x) = 0$ then $\Pr_{S_1,\ldots,S_k}[p_{S_1,\ldots,S_k}(x) = 0] \geq 1 - 1/2^k$.
- The number of monomials in $p_{S_1,\ldots,S_k}$ is $O(2^k)$.

(c) Define the function $EQ : \{-1,1\}^m \times \{-1,1\}^m \to \{0,1\}$ to be 1 on input $(x,y) \in \{-1,1\}^m \times \{-1,1\}^m$ if $x = y$, and 0 otherwise. Use part (b) (and the properties of the EQ function) to design a probabilistic polynomial $\mathcal{D}$ for EQ with the properties:

- For all $x, y \in \{-1,1\}^m$, if $EQ(x,y) = 1$ then $\Pr_{p \sim \mathcal{D}}[p(x,y) = 2^k] = 1$.
- For all $x, y \in \{-1,1\}^m$, if $EQ(x,y) = 0$ then $\Pr_{p \sim \mathcal{D}}[p(x,y) = 0] \geq 1 - 1/2^k$.
- For all $p \sim \mathcal{D}$, the number of monomials in p is $O(2^k)$.

(d) The function $EQIP : (\{-1,1\}^m)^{2d} \to \{0,1\}$ (a.k.a. "equality inner product") is defined as follows: given two vectors $x = (x_1, \ldots, x_d), y = (y_1, \ldots, y_d)$ where each $x_i, y_i \in \{0,1\}^m$, output 1 if and only if there is an i such that $x_i = y_i$. Use part (c) to design a probabilistic polynomial $\mathcal{D}$ for $EQIP$ with the properties:

- For all $x, y \in (\{-1,1\}^m)^d$, if $EQIP(x,y) = 1$ then $\Pr_{p \sim \mathcal{D}}[p(x,y) \neq 0] = 1$.
- For all $x, y \in (\{-1,1\}^m)^d$, if $EQIP(x,y) = 0$ then $\Pr_{p \sim \mathcal{D}}[p(x,y) = 0] \geq 1 - d/2^k$.
- For all $p \sim \mathcal{D}$, the number of monomials in p is $O(d2^k)$.

(e) Use the probabilistic polynomial of part (d) to show that the equality product of an $n \times d$ and $d \times n$ matrix can be computed in randomized $O(M(n, O(d^2)) \log n)$ time, where $M(a,b)$ is the time complexity for matrix multiplication of $a \times b$ and $b \times a$ matrices over $\mathbb{Z}$ with $O(\log(a \cdot b))$ bit entries.
Hint: Set k from part (d) wisely, and use the evaluation lemma from lecture.

(f) Use the algorithm from part (d) to derive a subcubic algorithm for equality product of $n \times n$ matrices. How small can you make the exponent? You may assume that $M(n,n) \leq O(n^{2.4})$.