CS 598TMC, Fall 2020
Timothy Chan

Homework 2 (due Oct 14 Wednesday 10:00am (CT))

Instructions: see previous homework.

1. [*33 pts*] Prove the following: if $(\min, +)$-matrix multiplication of two $n \times n$ matrices (with arbitrary real values) could be computed in $O(n^\beta)$ time for some constant $2 < \beta < 3$, then APSP (with arbitrary real weights) could be solved in $O(n^\beta)$ time (without losing any logarithmic factor).

   [Hint: we mentioned this result in class, but without proof. As suggested in class, use recursion, like in Munro's transitive closure algorithm (but without strongly connected components or topological sorting). Aim for four recursive calls... Don't use "repeated squaring", which would result in an extra logarithmic factor.]

2. [*34 pts*] Consider the following variant of APSP, called ($\leq k$)-*red APSP*: We are given an unweighted directed graph $G = (V, E)$ with n vertices, where some of the edges are labelled *red*. We are also given a number k (where $1 \leq k < n$). For every pair of vertices $s, t \in V$, we want to compute the minimum length $L_k[s, t]$ over all paths from s to t that use at most k red edges.

   (a) First describe an $O(kn^2)$-time dynamic programming algorithm that solves the single-source version of the problem (i.e., for a fixed s, compute $L_k[s, t]$ for all $t \in V$).

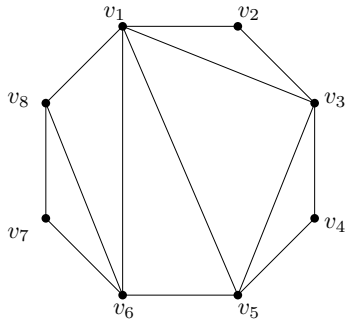
   (b) Next describe an $\widetilde{O}(kn^{2.529})$-time algorithm for ($\leq k$)-red APSP. (An $\widetilde{O}(k^2 n^{2.529})$-time solution would still be rewarded most of the points.)

   [*Hint*: modify Zwick's APSP algorithm. It may be helpful to compute not just $L_k[s, t]$ but $L_{k'}[s, t]$ for all $k' \leq k$.]

   [*Bonus research question*:[1] the above bound is subcubic only when k is less than around $n^{0.47}$; is there a truly subcubic algorithm for all k?]

[1] I don't know the answer to do this...

3. [*33 pts*] We are given a convex polygon P with vertices $v_1 v_2 \ldots v_n v_1$ in clockwise order. We are also given a weight matrix $w(v_i, v_j)$ (satisfying $w(v, v) = 0$ and $w(u, v) = w(v, u)$). Our objective is to compute a triangulation that minimizes the total weight, i.e., sum of the weights of the edges of the triangulation. (A triangulation is a collection of edges that divide P into triangles; we are not allowed to introduce additional vertices. The figure below shows one possible triangulation for a convex 8-gon.)



This problem has a standard dynamic programming algorithm: Defining $C[i, j]$ to be the weight of the minimum triangulation of the subpolygon $v_i v_{i+1} \cdots v_j v_i$ (for $j \geq i+1$), we have the recursive formula for $j \geq i + 2$:

$$C[i, j] \;=\; \min_{i<k<j} (C[i, k] + C[k, j]) + w(v_i, v_j)$$

(for the base case, $C[i, i + 1] = w(v_i, v_{i+1})$). By evaluating the formula in increasing order of $j - i$, we can compute $C[1, n]$ in $O(n^3)$ time.

Prove the following: if $(\min, +)$-matrix multiplication of two $n \times n$ matrices could be computed in $O(n^{3-\delta})$ time for some constant $0 < \delta < 1$, then the above triangulation problem could be solved in $O(n^{3-\delta'})$ time for some constant $\delta' > 0$.

[*Hint*: Let d be a parameter. First compute $C[i, j]$ for all i, j with $j - i \leq d$ by the above formula. Next for i, j with $j - i > d$, consider three cases: (a) $k - i \leq d$, (b) $j - k \leq d$, (c) $k - i > d$ and $j - k > d$.]

[*Bonus*:[2] describe a reduction from the triangulation problem to $(\min, +)$-matrix multiplication that does not hurt the time complexity, i.e., achieves $\delta' = \delta$.]

[*Bonus research question*:[3] in the case when all finite edge weights $w(v_i, v_j)$ are small integers in $[c]$ for a constant c, could the triangulation problem be solved in truly subcubic time (unconditionally) by using standard matrix multiplication?]

[2] I know how to do this...
[3] I don't know the answer to this...