Problem 1

Give an $\tilde{O}(n^2)$ time algorithm for the following problem. Given a directed graph $G = (V, E)$ on $n$ nodes, with integer edge weights and a vertex $s \in V$, compute for all $u, v \in V$, the minimum last edge weight of a nondecreasing path from $u$ to $v$ passing through $s$. If no $u \rightarrow v$ nondecreasing path passes through $s$, then return $-\infty$ for $u, v$.

Recall that a nondecreasing path is a path whose consecutive edge weights form a nondecreasing sequence.

Problem 2

The equality product of two $n \times n$ integer matrices $A$ and $B$ is the matrix $C$ such that $C[i, j] = |\{k \mid A[i, k] = B[k, j]\}|$.

Recall that the dominance product of $A$ and $B$ is given by $(A \odot B)[i, j] = |\{k \mid A[i, k] \leq B[k, j]\}|$.

(a) Suppose that the dominance product of two $n \times n$ matrices can be computed in $O(n^c)$ time. Show that the equality product of two $n \times n$ matrices can then also be computed in $O(n^c)$ time.

(b) Show that given an instance of the dominance product of $n \times n$ matrices $A, B$, in $O(n^2 \log n)$ time, one can convert it into an instance of dominance product of $n \times n$ matrices $A', B'$ such that the entries of $A'$ and $B'$ are integers between 1 and $2n$, and the dominance product of $A'$ and $B'$ equals the dominance product of $A$ and $B$.

(c) Suppose that the equality product of two $n \times n$ matrices can be computed in $O(n^c)$ time. Show that the dominance product of two $n \times n$ matrices can then be computed in $O(n^c \log n)$ time. (Use part (b).)

Problem 3

Recall from lecture 5 that the following matrix product can be computed in $O(n^{(3+\omega)/2})$ time for $n \times n$ matrices:

$$(A \odot B)[i, j] = \min\{B[k, j] \mid A[i, k] = 1\}.$$ 

Consider any directed graph $G$ for which for every vertex $v$ the weights on the edges going out of $v$ take at most $L$ values. (The $L$ different values can be different for each vertex.) Then use the algorithm from lecture 5 to show that All-Pairs Shortest Paths in such an $n$-node $G$ can be computed in $\tilde{O}(\sqrt{Ln^{(9+\omega)/4}})$ time.

(That is, for any $\varepsilon > 0$, we can handle up to $n^{(3-\omega)/2-\varepsilon}$ distinct edge weights out of every node in truly subcubic time.)