CS381 **Homework 7 Sketch of Solution**

Question 1.

1. Initialize $A = M$ and then, for $j = 1, 2, \ldots, \log n$ do

$$A = A \circ A$$

It works because the fact that

$$M^{(2^{j+1})} = M^{(2^j)} \circ M^{(2^j)}$$

implies that, at the end of the kth iteration, we have $A = M^{(2^k)}$. Therefore after the last iteration we have $A = M^{(2^{\log n})} = M^{(n)}$.

2. First compute (as in the above) the sequence of $\log n$ matrices $M^{(2)}, M^{(4)}, M^{(8)}, \ldots, M^{(n)}$, using $\log n$ multiplications. Next, let the binary representation of k be $b_q b_{q-1} \cdots b_1 b_0$ where b_0 is the least significant bit and $q = \log n$. That is, we have

$$k = b_0 * 2^0 + b_1 * 2^1 + b_2 * 2^2 + \cdots + b_{\log n} * 2^{\log n}.$$

which implies that

$$M^{(k)} = M^{(b_0 * 2^0)} \circ M^{(b_1 * 2^1)} \circ M^{(b_2 * 2^2)} + M^{(b_{\log n} * 2^{(\log n)})}$$

with the convention that multiplying a matrix by M^0 is a "do nothing" operation. We already have every one of the matrices that participate in the above expression for $M^{(k)}$, because we alreadt computed all of

$$\{M^{(1)}, M^{(2)}, M^{(4)}, M^{(8)}, \ldots, M^{(n)}\}$$

and therefore $M^{(k)}$ can be obtained from these matrices with an additional (at most) $\log n - 1$ multiplications.

Comment. It is possible to reduce the space to $O(n^2)$ rather than $O(n^2 \log n)$ by avoiding the storage of the $\log n$ atrices of the form $M^{(2^j)}$. This is done by accumulating the terms of the above equation for $M^{(k)}$ in left-to-right order *during* the computation of the successive $M^{(2^j)}$ matrices (i.e., as soon as an $M^{(2^j)}$ is computed it is included or not in the accumulated product based based on whether b_j is 1 or 0), hence there is no need to store all the $\log n$ matrices of the form $M^{(2^j)}$.

3. The graph-theoretic interpretation of $M^{(k)}[i, j]$ is that it is the length of a shortest i-to-j path in G that uses at most k edges. Therefore $M^{(n)}[i, j]$ is the length of a shortest i-to-j path in G, and can be computed in $O(n^3)$ time as explained in class, e.g., by using by using the dynamic programming solution for computing the all-pairs shortest paths matrix (or, somewhat less efficiently, by using Dijkstra's algorithm n times).

Question 2. In this problem $M^{(k)}[i, j]$ has a similar meaning to the previous problem except that the cost of a path is now the max (rather than the sum) of the edge costs on it. As stated in class, in such a situation a least-cost path is one that follows the edges of a minimum cost spanning tree. As G is undirected, we can compute a minimum cost spanning tree of G by using Kruskal's algorithm; let T be such a tree, and recall from the class lectures that T can be computed in $O(n + e \log n)$ time. Next, for $i = 1, \ldots, n$, fill row i of matrix $M^{(n)}$ by traversing T starting from vertex i and computing during the traversal, for every vertex j, the maximum edge cost (call it $c_{i,j}$) on the i-to-j path in T; this traversal staring from i as root takes linear time because $c_{i,j}$ is the max of $c_{i,parent(j)}$ and the cost of edge $(parent(j), j)$. Of course, when we are done with the traversal that starts at i, we set every $M^{(n)}[i, j]$ equal to $c_{i,j}$. Because we have to do this for every row i, there is an additional $O(n^2)$ term in the overall time complexity.