Advanced Analysis of Algorithms - Homework II (Solutions)

K. Subramani
LCSEE,
West Virginia University,
Morgantown, WV
{ksmani@csee.wvu.edu}

1 Problems

1. Assume that you are given the following weight matrix for a four vertex graph. Compute the shortest-path distances between each pair of vertices in the graph using the Floyd algorithm. You are required to present all the intermediate distance matrices.

$$\mathbf{W} = \begin{bmatrix} 0 & 5 & \infty & 3 \\ 4 & 0 & 4 & \infty \\ 2 & 1 & 0 & \infty \\ 3 & 2 & \infty & 0 \end{bmatrix}$$

Solution: The recurrence relation of Floyd's algorithm is:

$$\begin{aligned} d_{ij}^0 &= \mathbf{W_{ij}} \\ d_{ij}^k &= \min\{d_{ij}^{k-1},\ d_{ik}^{k-1} + d_{kj}^{k-1}\},\ k > 1 \end{aligned}$$

We therefore have,

$$\mathbf{D^0} = \begin{bmatrix} 0 & 5 & \infty & 3 \\ 4 & 0 & 4 & \infty \\ 2 & 1 & 0 & \infty \\ 3 & 2 & \infty & 0 \end{bmatrix}$$

Working through the recurrence relation, we get,

$$\mathbf{D^1} = \begin{bmatrix} 0 & 5 & \infty & 3 \\ 4 & 0 & 4 & 7 \\ 2 & 1 & 0 & 5 \\ 3 & 2 & \infty & 0 \end{bmatrix}$$

$$\mathbf{D^2} = \begin{bmatrix} 0 & 5 & 9 & 3 \\ 4 & 0 & 4 & 7 \\ 2 & 1 & 0 & 5 \\ 3 & 2 & 6 & 0 \end{bmatrix}$$

$$\mathbf{D^3} = \begin{bmatrix} 0 & 5 & 9 & 3 \\ 4 & 0 & 4 & 7 \\ 2 & 1 & 0 & 5 \\ 3 & 2 & 6 & 0 \end{bmatrix}$$

$$\mathbf{D^4} = \begin{bmatrix} 0 & 5 & 9 & 3 \\ 4 & 0 & 4 & 7 \\ 2 & 1 & 0 & 5 \\ 3 & 2 & 6 & 0 \end{bmatrix}$$

□

2. Assume that you are given a graph $\mathbf{G}$ with some edges having positive weights and some edges having negative weights. How would you use the Floyd algorithm for checking whether $\mathbf{G}$ has a negative cost cycle?

   Solution: Consider the following theorem:

   Theorem 1.1 *$\mathbf{G}$ has a negative cost cycle if and only if $\mathbf{D^n_{ii}} = 0$ for some $i = 1, 2, \ldots n$.*

   Proof:

   If: Assume that $\mathbf{D^n_{ii}} < 0$ for some $i = 1, 2, \ldots n$. As per the mechanics of the Floyd algorithm, this means that the shortest path from v_i to itself, using $\{v_1, v_2, \ldots, v_n\}$ as intermediate vertices has negative cost. This shortest path is a closed walk around v_i and is either a simple cycle or constituted of simple cycles. In the former case, we have a negative cost cycle. In the latter case, at least one of the simple cycles has to have negative cost; otherwise, all the cycles have positive cost and so does their sum, which means that $\mathbf{D}^n_{ii} > 0$, contradicting the hypothesis. Thus, if $\mathbf{D}^n_{ii} < 0$, for some $i = 1, 2, \ldots n$, then $\mathbf{G}$ has a negative cost cycle.

   Only If: Assume that $\mathbf{G}$ has a negative cost cycle. Consider any vertex, say v_i, on this cycle. It is clear that the shortest path from v_i to itself using $\{v_1, v_2, \ldots, v_n\}$ as intermediate vertices has negative cost. Therefore, $\mathbf{D}^n_{ii} < 0$.
   □

   The above theorem suggests the following algorthm for checking whether an arbitrarily weighted graph $\mathbf{G}$ has a negative cost cycle:

   (i) Run Floyd's algorithm on $\mathbf{G}$ to get the distance matrix $\mathbf{D^n}$.

   (ii) If $\mathbf{D}^n_{ii} < 0$ for any $i = 1, 2, \ldots, n$, declare that $\mathbf{G}$ has a negative cost cycle; otherwise, declare that $\mathbf{G}$ does not have a negative cost cycle.

   □

3. Let $\mathbf{X} =< x_1, x_2, \ldots x_m >$ and $\mathbf{Y} =< y_1, y_2, \ldots, y_n >$ denote two sequences on a fixed alphabet Σ (You may assume that $\Sigma = \{0, 1\}$, if it helps). Devise an efficient algorithm to determine the length of the Longest Common Subsequence between $\mathbf{X}$ and $\mathbf{Y}$. Note that a subsequence of a sequence need not be contiguous.
   Hint: Use Dynamic Programming and derive a recurrence relation.

   Solution:

   Let $\mathbf{Z} =< z_1, z_2, \ldots, z_r >$ denote the Longest Common Subsequence (LCS) between $\mathbf{X}$ and $\mathbf{Y}$. We use the following notational scheme:

   (a) $\mathbf{X_i}$ denotes the sequence $< x_1, x_2, \ldots, x_i >$.

   (b) $\mathbf{LC}(\mathbf{X},\ \mathbf{Y})$ denotes the LCS between $\mathbf{X}$ and $\mathbf{Y}$.

   (c) $f[i][j]$ denotes the length of the LCS between $\mathbf{X_i}$ and $\mathbf{Y_j}$, i.e., $f[i][j] = |\mathbf{LC}(\mathbf{X_i},\ \mathbf{Y_j})|$.

   Consider the following cases:

   (i) $x_m = y_n$ - In this case, we must have $z_r = x_m$ and further, $\mathbf{Z_{r-1}} = \mathbf{LC}(\mathbf{X_{m-1}}, \mathbf{Y_{n-1}})$. Observe that if $z_r \neq x_m, y_n$, then the length of the LCS can be increased by at least 1, by appending x_m to $\mathbf{Z}$, thereby contradicting the optimality of $\mathbf{Z}$. Given that $z_r = x_m$, it is easy to see that $\mathbf{Z_{r-1}} = \mathbf{LC}(\mathbf{X_{m-1}},\ \mathbf{Y_{n-1}})$. If this were not the case, then we can determine $\mathbf{LC}(\mathbf{X_{m-1}},\ \mathbf{Y_{n-1}})$ and append x_m to it, thereby contradicting the optimality of $\mathbf{Z}$.

(ii) $x_m \neq y_n$ and $z_r \neq x_m$- We must have $\mathbf{Z} = \mathbf{LC}(X_{m-1}, Y_n)$. If this were not the case, then there is a subsequence between $\mathbf{X}$ and $\mathbf{Y}$ of length greater than k. However, this subsequence is also a subsequence between $\mathbf{X_{m-1}}$ and Y, thereby contradicting the optimality of $\mathbf{Z}$.

(iii) $x_m \neq y_n$ and $z_r \neq y_n$ - We must have $\mathbf{Z} = \mathbf{LC}(X_m, Y_{n-1})$. Symmetric to the case above.

From the above discussion, the following recurrence relation suggests itself for the length of the LCS beween $\mathbf{X_i}$ and $\mathbf{Y_j}$:

$$\begin{aligned} f[i][j] &= 0, \text{ if } \mathrm{i} = 0 \text{ or } \mathrm{j} = 0 \\ &= \max(f[i-1][j], f[i][j-1]), \text{ if } \mathrm{x_i} \neq \mathrm{y_j} \text{ and } \mathrm{i, j} \geq 1 \\ &= 1 + f[i-1][j-1], \text{ if } \mathrm{x_i} = \mathrm{y_j} \text{ and } \mathrm{i, j} \geq 1 \end{aligned}$$

This recurrence relation can be easily converted into an algorithm that computes $f[m][n]$ in bottom-up fashion and runs in time $O(m \cdot n)$. □

4. A Directed Acyclic Graph (DAG) is a graph without any cycles. The Longest Path problem on a graph is concerned with computing the length of the longest simple path between each pair of vertices in that graph. Given a DAG **D**, argue that the Longest Path problem does satisfy the optimal substructure property demanded by dynamic programming. Derive a recurrence relation for this problem.

Solution:

Let D_{ij}^k denote the length of the longest path between vertex v_i and vertex v_j with all intermediate vertices drawn from the set $S_k = \{v_1, v_2, \ldots, v_k\}$.

Claim 1.1

$$\begin{aligned} D_{ij}^k &= W[i][j], \ \mathrm{k} = 0 \\ &= \max(D_{ij}^{k-1}, D_{ik}^{k-1} + D_{kj}^{k-1}) \end{aligned}$$

Proof: Clearly, $D_{ij}^0 = W[i][j]$, i.e., if no intermediate vertices are allowed on the longest path between v_i and v_j, then $W[i][j]$ indeed denotes the length of the longest path between v_i and v_j; indeed it denotes the length of the shortest path between these vertices as well!

Assume that the above recurrence relation is valid whenever the intermediate vertices are drawn from the set $S_{k-1} = \{v_1, v_2, \ldots, v_{k-1}\}$. Now consider the case in which we are allowed to draw the intermediate vertices from the set $S_k = \{v_1, v_2, \ldots, v_{k-1}, v_k\}$. Let us study the structure of the longest path p between v_i and v_j. We need to consider the following two cases:

(a) v_k is not on p - In this case, $D_{ij}^k = D_{ij}^{k-1}$.

(b) v_k is on p - Let p_{ik} denote the portion of p between v_i and v_k; likewise, let p_{kj} denote the portion of p between v_k and v_j. We claim that p_{ik} is the longest path between v_i and v_k with all the intermediate vertices drawn from the set $S_{k-1} = \{v_1, v_2, \ldots v_{k-1}\}$. If this were not the case, then there is a path p'_{ik} between v_i and v_k of length greater than p_{ik}. Now this path *must* have some vertices in common with the path p_{kj}. Otherwise, p'_{ik} can be combined with p_{kj} to get a path between v_i and v_j, which is longer than p, contradicting the optimality of p. However, if p'_{ij} has vertices in common with p_{kj}, then there *must* be a cycle in $\mathbf{D}$, contradicting the fact that $\mathbf{D}$ is a DAG! (Convince yourself this is true.)

Likewise, the path between v_k and v_j is the longest such path, with all intermediate vertices drawn from the set $S_{k-1} = \{v_1, v_2, \ldots, v_{k-1}\}$, thereby establishing the correctness of the recurrence relation in Claim 1.1.

□

□

5. Assume that you are given a chain of matrices $< A_1\ A_2\ A_3\ A_4 >$, with dimensions 2×5, 5×4, 4×2 and 2×4 respectively. Compute the optimal number of multiplications required to calculate the chain product and also indicate what the optimal order of multiplication should be using parentheses.

   Solution: Let $m[i, j]$ denote the optimal number of multiplications to multiply the chain $< A_i, A_{i+1}, \ldots, A_j >$, where matrix A_i has dimensions $d_{i-1} \times d_i$. As per the discussion in class, we know that

$$\begin{aligned} m[i,j] &= 0, \text{ if j} \leq \text{i} \\ &= \min_{k:i \leq k < j} m[i,k] + m[k+1,j] + d_{i-1} \cdot d_k \cdot d_j \end{aligned}$$

   Computing $\mathbf{M} = [m[i, j]], i = 1, 2, 3, 4;\ j = i, i + 1, \ldots, 4$, in bottom-up fashion, we get

$$\mathbf{M} = \begin{bmatrix} 0 & 40 & 56 & 72 \\ 0 & 0 & 40 & 80 \\ 0 & 0 & 0 & 32 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

   As per the above table, the optimal number of multiplications to multiply the given chain is 72; by recording the split values, we see that the optimal order of multiplication is $(((A_1 \cdot A_2) \cdot A_3) \cdot A_4)$. □