Analysis of Algorithms - Quiz II (Solutions)

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

1 Problems

1. **Divide-And-Conquer:** Use Strassen's matrix mutiplication algorithm to multiply

$$\mathbf{X} = \begin{bmatrix} 3 & 2 \\ 4 & 8 \end{bmatrix} \text{ and } \mathbf{Y} = \begin{bmatrix} 1 & 5 \\ 9 & 6 \end{bmatrix}.$$

Solution: We set $\mathbf{Z} = \mathbf{X} \cdot \mathbf{Y}$ and partition each matrix into four submatrices as discussed in class. Accordingly, $\mathbf{A} = [3]$, $\mathbf{B} = [2]$, $\mathbf{C} = [4]$, $\mathbf{D} = [8]$, $\mathbf{E} = [1]$, $\mathbf{F} = [5]$, $\mathbf{G} = [9]$ and $\mathbf{H} = [6]$, where,

$$\mathbf{Z} = \begin{bmatrix} \mathbf{I} & \mathbf{J} \\ \mathbf{K} & \mathbf{L} \end{bmatrix}, \mathbf{X} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \text{ and } \mathbf{Y} = \begin{bmatrix} \mathbf{E} & \mathbf{F} \\ \mathbf{G} & \mathbf{H} \end{bmatrix}$$

Applying Strassen's algorithm, we compute the following products:

(i) $S_1 = \mathbf{A} \cdot (\mathbf{F} - \mathbf{H}) = [3] \cdot ([5] - [6]) = [-3]$.
(ii) $S_2 = (\mathbf{A} + \mathbf{B}) \cdot \mathbf{H} = ([3] + [2]) \cdot [6] = [30]$.
(iii) $S_3 = (\mathbf{C} + \mathbf{D}) \cdot \mathbf{E} = ([4] + [8]) \cdot [1] = [12]$.
(iv) $S_4 = \mathbf{D} \cdot (\mathbf{G} - \mathbf{E}) = [8] \cdot ([9] - [1]) = [64]$.
(v) $S_5 = (\mathbf{A} + \mathbf{D}) \cdot (\mathbf{E} + \mathbf{H}) = ([3] + [8]) \cdot ([1] + [6]) = [77]$.
(vi) $S_6 = (\mathbf{B} - \mathbf{D}) \cdot (\mathbf{G} + \mathbf{H}) = ([2] - [8]) \cdot ([9] + [6]) = [-90]$.
(vii) $S_7 = (\mathbf{A} - \mathbf{C}) \cdot (\mathbf{E} + \mathbf{F}) = ([3] - [4]) \cdot ([1] + [5]) = [-6]$.

From the above products, we can compute **Z** as follows:

(i) $\mathbf{I} = S_5 + S_6 + S_4 - S_2 = 21$
(ii) $\mathbf{J} = S_1 + S_2 = 27$
(iii) $\mathbf{K} = S_3 + S_4 = 76$
(iv) $\mathbf{L} = S_1 - S_7 - S_3 + S_5 = 68$

The correctness can easily be verified using the naive algorithm. □

2. **Dynamic Programming:** Assume that you are given a chain of matrices $\langle A_1\ A_2\ A_3\ A_4 \rangle$, with dimensions 2×5, 5×4, 4×2 and 2×4 respectively. Compute the optimal number of multiplications required to calculate the chain product.

Solution: Let $m[i,j]$ denote the optimal number of multiplications to multiply the chain $\langle A_i, A_{i+1}, \ldots, A_j \rangle$, 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. □

3. **Shortest Paths:** The matrix $\mathbf{W}$ represents the adjacency matrix of a 4-vertex graph.

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

Run the "matrix-multiplication" algorithm on this graph to compute the All-Pairs shortest paths. Recall that in this algorithm, we increase the number of edges on the shortest path between a pair of vertices from 0 to $(n-1)$, on an n-vertex graph. You are required to show all the intermediate matrices.

Solution: As per the algorithm discussed in class,

$$l_{ij}^k = \min_{1 \leq k \leq n} (l_{ik}^{k-1} + w_{kj})$$

where l_{ij} represents the length of the shortest path from v_i to v_j using at most k edges.

Using the trick discussed in class,

$$\begin{aligned} \mathbf{L^1} &= \mathbf{W} \\ \mathbf{L^2} &= \mathbf{W} \otimes \mathbf{W} \\ &= \begin{bmatrix} 0 & -1 & -3 & -2 \\ \infty & 0 & -2 & -1 \\ \infty & 2 & 0 & 1 \\ \infty & \infty & \infty & 0 \end{bmatrix} \\ \mathbf{L^4} &= \mathbf{L^2} \otimes \mathbf{L^2} \\ &= \begin{bmatrix} 0 & -1 & -3 & -2 \\ \infty & 0 & -2 & -1 \\ \infty & 2 & 0 & 1 \\ \infty & \infty & \infty & 0 \end{bmatrix} \end{aligned}$$

$\mathbf{L^4}$ represents the matrix of shortest path distances; note that the matrix multiplication is not traditional matrix multiplication, but the funny matrix multiplication that we discussed in class. □

4. **Shortest Paths:** In class, we discussed how the Floyd-Warshall algorithm is used to compute the length of the shortest path between all vertex-pairs of the input graph; the algorithm can effect this computation *only* if there are no negative cost cycles in the graph. How would you use the Floyd-Warshall approach to declare that the input graph has a negative cost cycle?

Solution: As per the discussion in class,

$$d_{ij}^k = \min\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\}$$

where d_{ij}^k represents the length of the shortest path from vertex v_i to vertex v_j, with all intermediate vertices in the set $S = \{v_1, v_2, \ldots, v_k\}$. Likewise, $\mathbf{D}^k = [d_{ij}]_{i,j=1,2,\ldots n}$. The crucial observation is that there exists a negative cycle in the input graph if and only if there is a vertex v_i such that $d_{ii}^k < 0$, for some $k = 1, 2, \ldots, n$. Since the distance of a vertex to itself is monotonically decreasing over the iterations of Floyd-Warshall, we only need to check whether $d_{ii}^n < 0$, for any vertex v_i and if so, declare that the input graph has a negative cost cycle. In the event that $d_{ii}^n \geq 0,\ \forall i = 1, 2, \ldots n$, the graph does not contain a negative cost cycle. □

5. **String Algorithms:** Draw a compressed trie for the following set of strings:

$$\{abab, baba, ccccc, bbaaa, caa, bbaacc, cbcc, cbca\}.$$

Solution: □

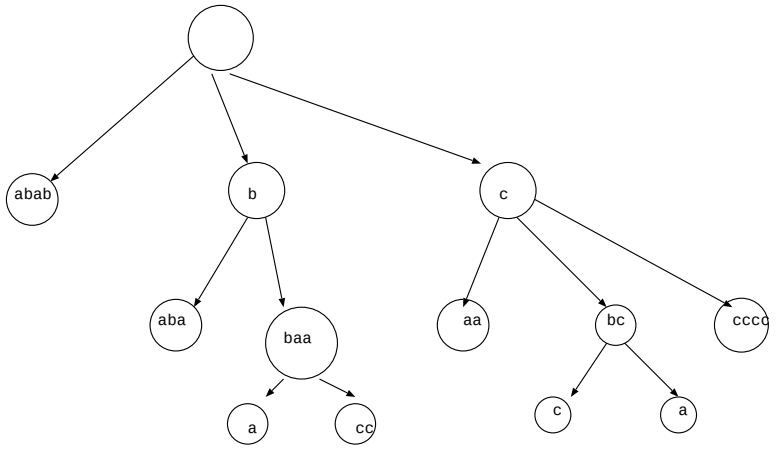

Figure 1: Compressed Trie