Matrix-chain multiplication

[CLRS Section 15.2]

Recalling Matrix Multiplication

The product $C = AB$ of a $p \times q$ matrix A and a $q \times r$ matrix B is a $p \times r$ matrix given by

$$c[i,j] = \sum_{k=1}^{q} a[i,k]b[k,j]$$

for $1 \le i \le p$ and $1 \le j \le r$.

Example: If

$$A = \begin{bmatrix} 1 & 8 & 9 \\ 7 & 6 & -1 \\ 5 & 5 & 6 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 8 \\ 7 & 6 \\ 5 & 5 \end{bmatrix},$$

then

$$C = AB = \begin{bmatrix} 102 & 101 \\ 44 & 87 \\ 70 & 100 \end{bmatrix}.$$

$\boxed{\text{Direct Matrix multiplication } AB}$

Given a $p \times q$ matrix A and a $q \times r$ matrix B, the direct way of multiplying $C = AB$ is to compute each

$$c[i,j] = \sum_{k=1}^{q} a[i,k]b[k,j]$$

for $1 \leq i \leq p$ and $1 \leq j \leq r$.

Complexity of Direct Matrix multiplication:

Note that C has pr entries and each entry takes $\Theta(q)$ time to compute so the total procedure takes $\Theta(pqr)$ time.

Direct Matrix multiplication of ABC

Given a $p \times q$ matrix A, a $q \times r$ matrix B and a $r \times s$ matrix C, then ABC can be computed in two ways $(AB)C$ and $A(BC)$:

The number of multiplications needed are:

$$\begin{aligned} mult[(AB)C] &= pqr + prs, \\ mult[A(BC)] &= qrs + pqs. \end{aligned}$$

When $p = 5$, $q = 4$, $r = 6$ and $s = 2$, then

$$\begin{aligned} mult[(AB)C] &= 180, \\ mult[A(BC)] &= 88. \end{aligned}$$

A big difference!

Implication: The multiplication "sequence" (parenthesization) is important!!

The Chain Matrix Multiplication Problem

Given

dimensions $p_0, p_1, \ldots, p_n$

corresponding to matrix sequence A_1, A_2, ..., A_n

where A_i has dimension $p_{i-1} \times p_i$,

determine the "multiplication sequence" that minimizes the number of scalar multiplications in computing $A_1 A_2 \cdots A_n$. That is, determine how to parenthisize the multiplications.

$$
\begin{aligned}
A_1 A_2 A_3 A_4 &= (A_1 A_2)(A_3 A_4) \\
&= A_1(A_2(A_3 A_4)) = A_1((A_2 A_3) A_4) \\
&= ((A_1 A_2) A_3)(A_4) = (A_1(A_2 A_3))(A_4)
\end{aligned}
$$

Exhaustive search: [redacted].

$P(n)$: # of Parenthisizations of sequence of n matrices.

[redacted]

$$
P(n) = \begin{cases} 0 & \text{if } n = 1 \\ \sum_{k=1}^{n-1} P(k)\, P(n-k) & \text{if } n \geq 2 \end{cases}
$$

$P(n) = \Omega(2^n)$

Notation:

$$A_{i \ldots j} = A_i A_{i+1} \cdots A_j$$

$A_{i...j} = A_i A_{i+1} \cdots A_j$

For any optimal multiplication sequence, at the last step you are multiplying two matrices $A_{i..k}$ and $A_{k+1..j}$ for some k. That is,

$$A_{i..j} = (A_i \cdots A_k)(A_{k+1} \cdots A_j) = A_{i..k}A_{k+1..j}.$$

Example

$$A_{3..6} = \underbrace{(A_3(A_4A_5))}\underbrace{(A_6)} = A_{3..5}A_{6..6}.$$

Here $k = 5$.

$$A_{i..j} = (A_i \cdots A_k)(A_{k+1} \cdots A_j) = A_{i..k} A_{k+1..j}.$$

Optimal Substructure Property: If final "optimal" solution of $A_{i..j}$ involves splitting into $A_{i..k}$ and $A_{k+1..j}$ at final step then parenthesization of $A_{i..k}$ and $A_{k+1..j}$ in final optimal solution must also be optimal for the subproblems "standing alone":

For $1 \le i \le j \le n$,

let $m[i,j]$ denote the minimum number of multiplications needed to compute $A_{i \cdots j}$.

How to compute $m[i,j]$??

$$m[i,j] = \begin{cases} 0 & \text{if } i=j \\ \min\limits_{i \le k < j} \{ m[i,k] + m[k,j] + p_i p_k p_j \}, & i<j \end{cases}$$

What should we store in DP table?

How to fill the DP table?

We should fill the table in increasing order of the length of the matrix chain.

$m[1,2], m[2,3], m[3,4], \ldots, m[n-3, n-2], m[n-2, n-1], m[n-1, n]$

$m[1,3], m[2,4], m[3,5], \ldots, m[n-3, n-1], m[n-2, n]$

$m[1,4], m[2,5], m[3,6], \ldots, m[n-3, n]$

$\vdots$

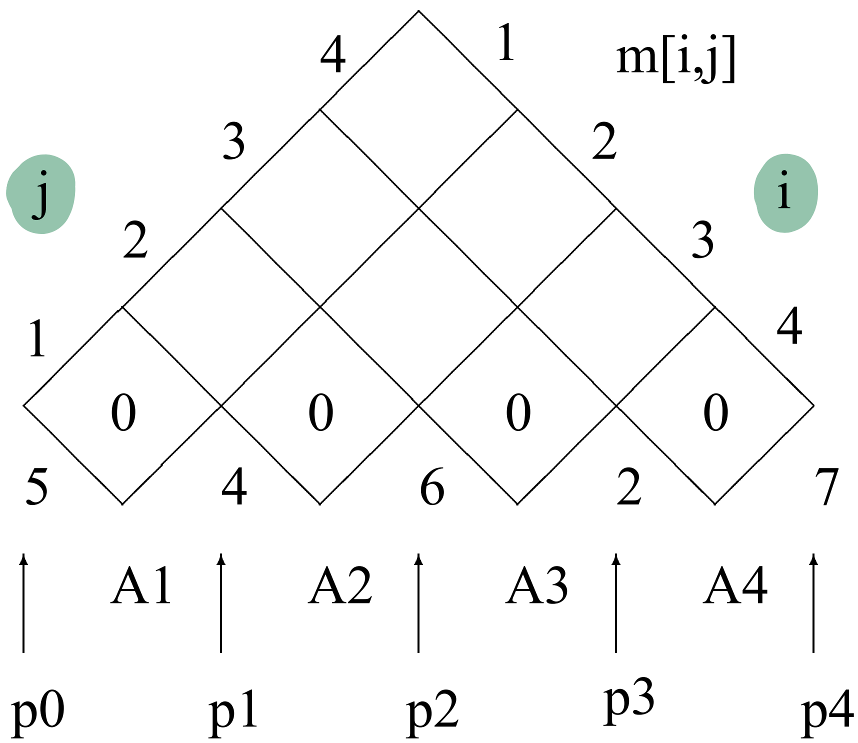
$m[1, n-1], m[2, n]$

$m[1, n]$

Example for the Bottom-Up Computation

Example: Given a chain of four matrices A_1, A_2, A_3 and A_4, with $p_0 = 5$, $p_1 = 4$, $p_2 = 6$, $p_3 = 2$ and $p_4 = 7$. Find $m[1, 4]$.

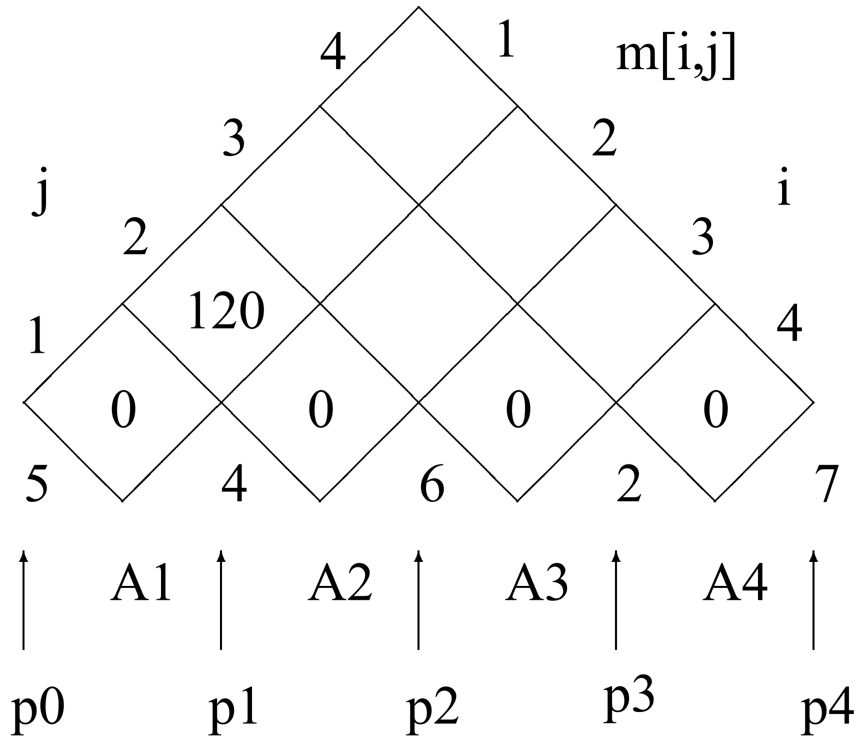
S0: Initialization



Example – Continued

Stp 1: Computing $m[1,2]$ By definition

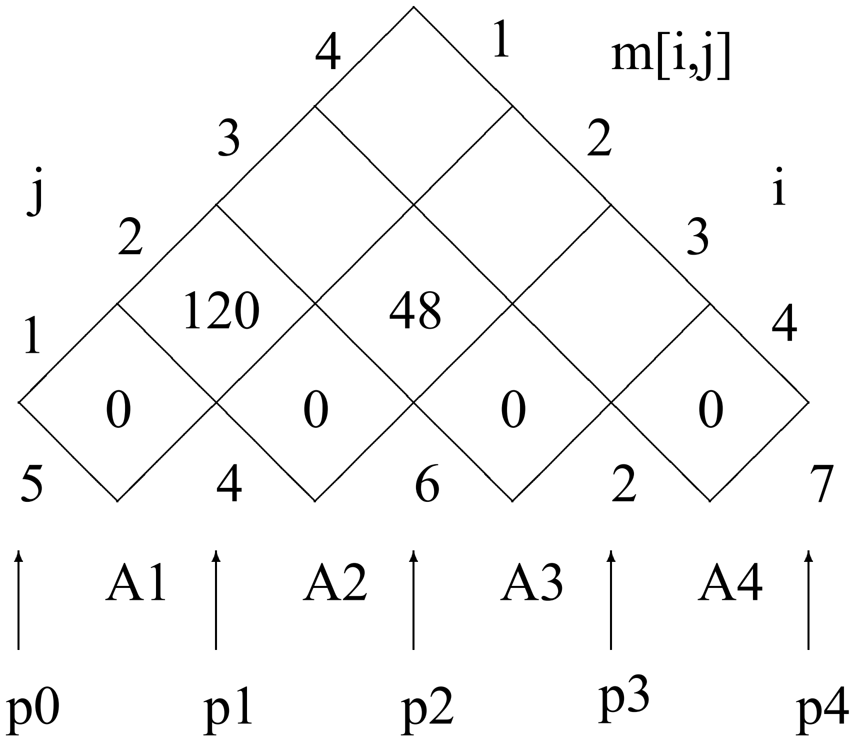
$$\begin{aligned} m[1,2] &= \min_{1 \le k < 2} (m[1,k] + m[k+1,2] + p_0 p_k p_2) \\ &= m[1,1] + m[2,2] + p_0 p_1 p_2 = 120. \end{aligned}$$



Example – Continued

Stp 2: Computing $m[2,3]$ By definition

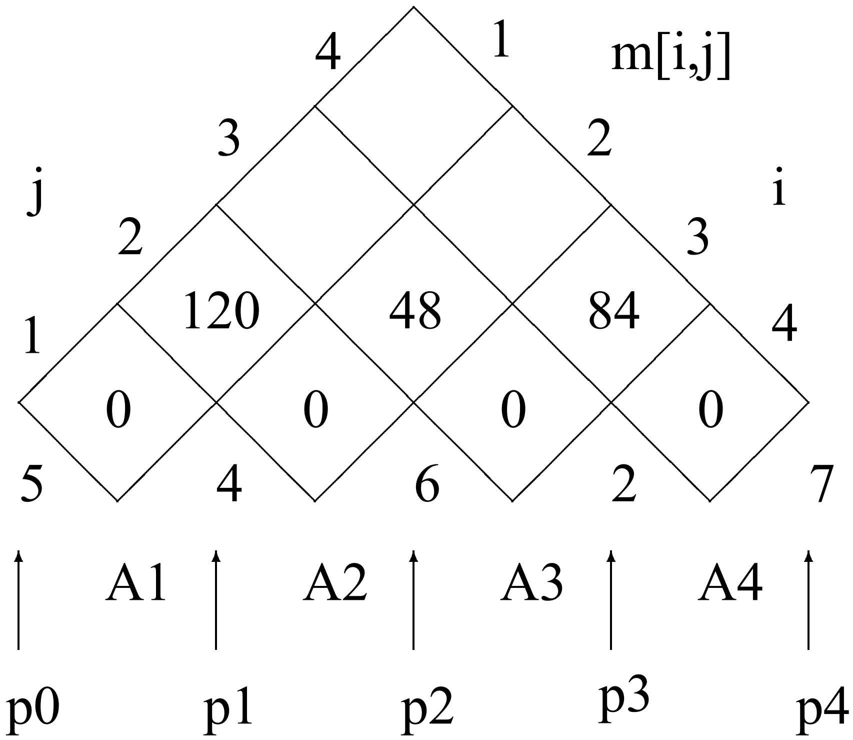
$$
\begin{aligned}
m[2,3] &= \min_{2\le k<3} (m[2,k] + m[k+1,3] + p_1 p_k p_3) \\
&= m[2,2] + m[3,3] + p_1 p_2 p_3 = 48.
\end{aligned}
$$



Example – Continued

Stp3: Computing $m[3,4]$ By definition

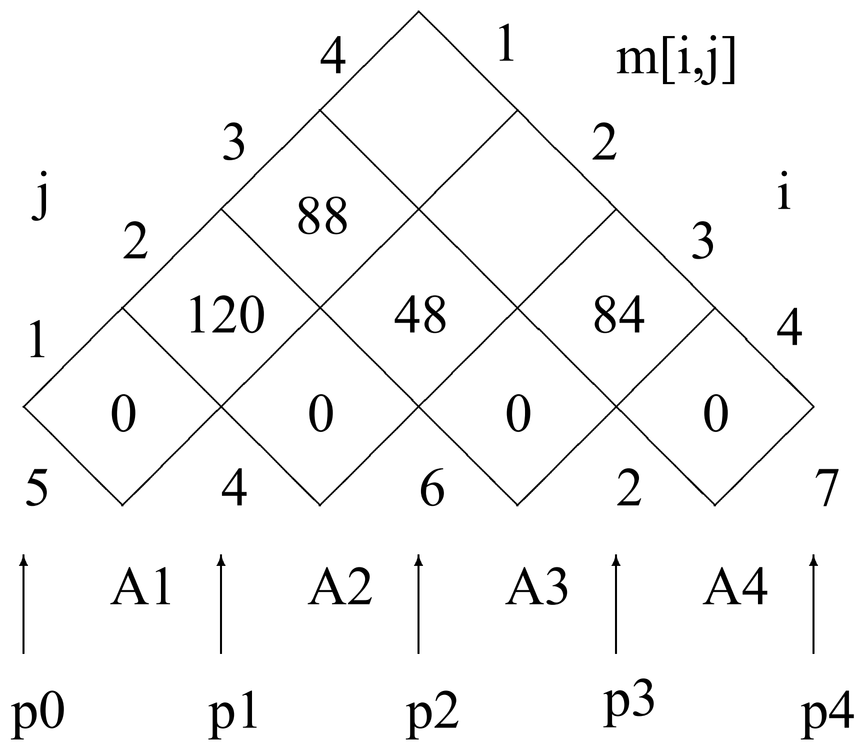
$$\begin{aligned} m[3,4] &= \min_{3 \le k < 4} (m[3,k] + m[k+1,4] + p_2 p_k p_4) \\ &= m[3,3] + m[4,4] + p_2 p_3 p_4 = 84. \end{aligned}$$



Example – Continued

Stp4: Computing $m[1,3]$ By definition

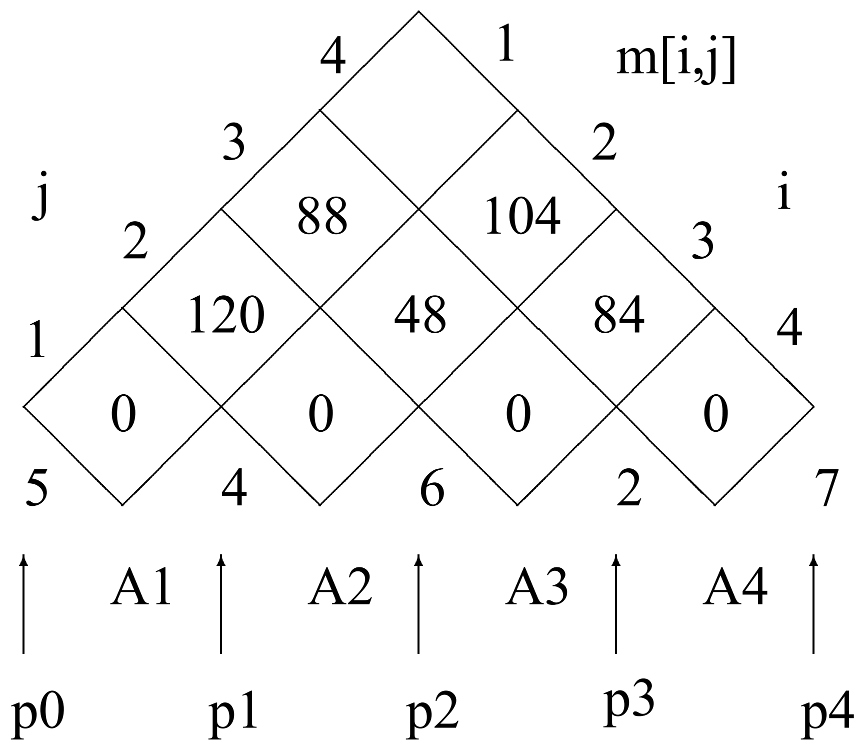
$$\begin{aligned} m[1,3] &= \min_{1\le k<3}(m[1,k]+m[k+1,3]+p_0p_kp_3) \\ &= \min\left\{\begin{array}{l} m[1,1]+m[2,3]+p_0p_1p_3 \\ m[1,2]+m[3,3]+p_0p_2p_3 \end{array}\right\} \\ &= 88. \end{aligned}$$



Example – Continued

Stp5: Computing $m[2,4]$ By definition

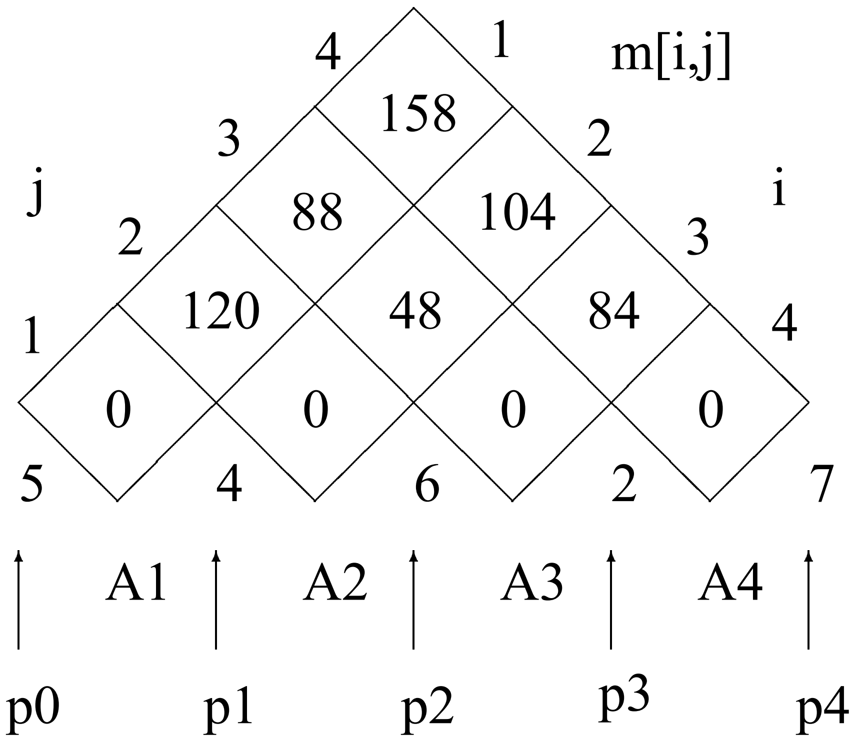
$$
\begin{aligned}
m[2,4] &= \min_{2\le k<4}(m[2,k]+m[k+1,4]+p_1p_kp_4) \\
&= \min\left\{ \begin{array}{l} m[2,2]+m[3,4]+p_1p_2p_4 \\ m[2,3]+m[4,4]+p_1p_3p_4 \end{array} \right\} \\
&= 104.
\end{aligned}
$$



Example – Continued

St6: Computing $m[1,4]$ By definition

$$
\begin{aligned}
m[1,4] &= \min_{1\le k<4}(m[1,k]+m[k+1,4]+p_0p_kp_4) \\
&= \min\left\{\begin{array}{l} m[1,1]+m[2,4]+p_0p_1p_4 \\ m[1,2]+m[3,4]+p_0p_2p_4 \\ m[1,3]+m[4,4]+p_0p_3p_4 \end{array}\right\} \\
&= 158.
\end{aligned}
$$



We are done!

Developing a Dynamic Programming Algorithm

Step 4: Construct an optimal solution from computed information – extract the actual sequence.

Idea: Maintain an array $s[1..n, 1..n]$, where $s[i, j]$ denotes k for the optimal splitting in computing $A_{i..j} = A_{i..k}A_{k+1..j}$. The array $s[1..n, 1..n]$ can be used recursively to recover the multiplication sequence.

How to Recover the Multiplication Sequence?

$s[1, n]$	$(A_1 \cdots A_{s[1,n]})(A_{s[1,n]+1} \cdots A_n)$
$s[1, s[1, n]]$	$(A_1 \cdots A_{s[1,s[1,n]]})(A_{s[1,s[1,n]]+1} \cdots A_{s[1,n]})$
$s[s[1, n] + 1, n]$	$(A_{s[1,n]+1} \cdots A_{s[s[1,n]+1,n]})\times$ $(A_{s[s[1,n]+1,n]+1} \cdots A_n)$
$\vdots$	$\vdots$

Do this recursively until the multiplication sequence is determined.

Developing a Dynamic Programming Algorithm

Step 4: Construct an optimal solution from computed information – extract the actual sequence.

Example of Finding the Multiplication Sequence:
Consider $n = 6$. Assume that the array $s[1..6, 1..6]$ has been computed. The multiplication sequence is recovered as follows.

$$\begin{array}{ll} s[1,6] = 3 & (A_1A_2A_3)(A_4A_5A_6) \\ s[1,3] = 1 & (A_1(A_2A_3)) \\ s[4,6] = 5 & ((A_4A_5)A_6) \end{array}$$

Hence the final multiplication sequence is

$$(A_1(A_2A_3))((A_4A_5)A_6).$$

The Dynamic Programming Algorithm

```
Matrix-Chain(p, n)
{   for (i = 1 to n) m[i, i] = 0;
    for (l = 2 to n)
    {
        for (i = 1 to n − l + 1)
        {
            j = i + l − 1;
            m[i, j] = ∞;
            for (k = i to j − 1)
            {
                q = m[i, k] + m[k + 1, j] + p[i − 1] ∗ p[k] ∗ p[j];
                if (q < m[i, j])
                {
                    m[i, j] = q;
                    s[i, j] = k;
                }
            }
        }
    }
    return m and s; (Optimum in m[1, n])
}
```

Complexity: The loops are nested three deep.

Each loop index takes on $\leq n$ values.

Hence the time complexity is $O(n^3)$. Space complexity $\Theta(n^2)$.