Matrix Chain Multiplication

- Given some matrices to multiply, determine the ***best*** order to multiply them so you minimize the number of single element multiplications.
  - i.e. Determine the way the matrices are parenthesized.

- First off, it should be noted that matrix multiplication is associative, but not commutative. But since it is associative, we always have:

- ((AB)(CD)) = (A(B(CD))), or any other grouping as long as the matrices are in the same consecutive order.

- BUT NOT: ((AB)(CD)) = ((BA)(DC))

Matrix Chain Multiplication

- It may appear that the amount of work done won’t change if you change the parenthesization of the expression, but we can prove that is not the case!

- Let us use the following example:
  - Let A be a 2x10 matrix
  - Let B be a 10x50 matrix
  - Let C be a 50x20 matrix

- But FIRST, let’s review some matrix multiplication rules…

Matrix Chain Multiplication

- Let's get back to our example: We will show that the way we group matrices when multiplying A, B, C ***matters***:
  - Let A be a 2x10 matrix
  - Let B be a 10x50 matrix
  - Let C be a 50x20 matrix

- Consider computing **A(BC):**
  - # multiplications for (BC) = 10x50x20 = 10000, creating a 10x20 answer matrix
  - # multiplications for A(BC) = 2x10x20 = 400
  - Total multiplications = 10000 + 400 = 10400.

- Consider computing **(AB)C**:
  - # multiplications for (AB) = 2x10x50 = 1000, creating a 2x50 answer matrix
  - # multiplications for (AB)C = 2x50x20 = 2000,
  - Total multiplications = 1000 + 2000 = 3000

Matrix Chain Multiplication

- Thus, our **goal** today is:
- Given a chain of matrices to multiply, determine the fewest number of multiplications necessary to compute the product.

Matrix Chain Multiplication

- Formal Definition of the problem:
  - Let $A = A_1 \cdot A_2 \cdot \ldots A_n$
  - Let $\mathbf{M_{i,j}}$ denote the minimal number of multiplications necessary to find the product:
    - $A_i \cdot A_{i+1} \cdot \ldots A_j$.
  - And let $\mathbf{p_{i-1} x p_i}$ denote the dimensions of matrix A_i.

- We must attempt to determine the minimal number of multiplications necessary($\mathbf{m_{1,n}}$) to find A,
  - assuming that we simply do each single matrix multiplication in the standard method.

Matrix Chain Multiplication

- The key to solving this problem is noticing the ***sub-problem optimality condition***:
  - If a particular parenthesization of the whole product is optimal, then any sub-parenthesization in that product is optimal as well.

- ***Say What?***
  - ***If*** (A (B ((CD) (EF)))) is optimal
  - Then (B ((CD) (EF))) is optimal as well
  - ***Proof on the next slide…***

Matrix Chain Multiplication

- Assume that we are calculating ABCDEF and that the following parenthesization is optimal:
    - (A (B ((CD) (EF))))
  - Then it is necessarily the case that
    - (B ((CD) (EF)))
  - is the optimal parenthesization of BCDEF.

- Why is this?
  - Because if it wasn't, and say (((BC) (DE)) F) was better, then it would also follow that
    - (A (((BC) (DE)) F)) was better than
    - (A (B ((CD) (EF)))),

Matrix Chain Multiplication

- Our final multiplication will ALWAYS be of the form
  - $(A_1 \cdot A_2 \cdot \ldots A_k) \cdot (A_{k+1} \cdot A_{k+2} \cdot \ldots A_n)$
- In essence, there is exactly one value of k for which we should "split" our work into two separate cases so that we get an optimal result.
  - Here is a list of the cases to choose from:
  - $(A_1) \cdot (A_2 \cdot A_3 \cdot \ldots A_n)$
  - $(A_1 \cdot A_2) \cdot (A_3 \cdot A_4 \cdot \ldots A_n)$
  - $(A_1 \cdot A_2 \cdot A_3) \cdot (A_4 \cdot A_5 \cdot \ldots A_n)$
  - ...
  - $(A_1 \cdot A_2 \cdot \ldots A_{n-2}) \cdot (A_{n-1} \cdot A_n)$
  - $(A_1 \cdot A_2 \cdot \ldots A_{n-1}) \cdot (A_n)$
- Basically, count the number of multiplications in each of these choices and **pick the minimum**.
  - One other point to notice is that you have to account for the minimum number of multiplications in each of the two products.

Matrix Chain Multiplication

- Consider the case multiplying these 4 matrices:
  - A: 2x4
  - B: 4x2
  - C: 2x3
  - D: 3x1

- 1. (A)(BCD) - This is a 2x4 multiplied by a 4x1,
  - so 2x4x1 = 8 multiplications, plus whatever work it will take to multiply (BCD).

- 2. (AB)(CD) - This is a 2x2 multiplied by a 2x1,
  - so 2x2x1 = 4 multiplications, plus whatever work it will take to multiply (AB) and (CD).

- 3. (ABC)(D) - This is a 2x3 multiplied by a 3x1,
  - so 2x3x1 = 6 multiplications, plus whatever work it will take to multiply (ABC).

Matrix Chain Multiplication

- **Our recursive formula:**
  - $M_{i,j}$ = min value of $M_{i,k} + M_{k+1,j} + p_{i-1}p_kp_j$, over all valid values of k.

- Now let's turn this recursive formula into a dynamic programming solution
  - Which sub-problems are necessary to solve first?

  - Clearly it's necessary to solve the smaller problems before the larger ones.
    - In particular, we need to know $m_{i,i+1}$, the number of multiplications to multiply any adjacent pair of matrices before we move onto larger tasks.
    - Similarly, the next task we want to solve is finding all the values of the form $m_{i,i+2}$, then $m_{i,i+3}$, etc.

Matrix Chain Multiplication

```
MATRIX-CHAIN-ORDER(p)
n ← length[p] − 1
for i ← 1 to n
     do m[i, i] ← 0
for l ← 2 to n                    ▷ l is the chain length.
     do for i ← 1 to n − l + 1
              do j ← i + l − 1
                  m[i, j] ← ∞
                  for k ← i to j − 1
                      do q ← m[i, k] + m[k + 1, j] + p_{i−1} p_k p_j
                         if q < m[i, j]
                            then m[i, j] ← q
                                 s[i, j] ← k
return m and s
```

- Basically, we're checking different places to "split" our matrices by checking different values of k and seeing if they improve our current minimum value.

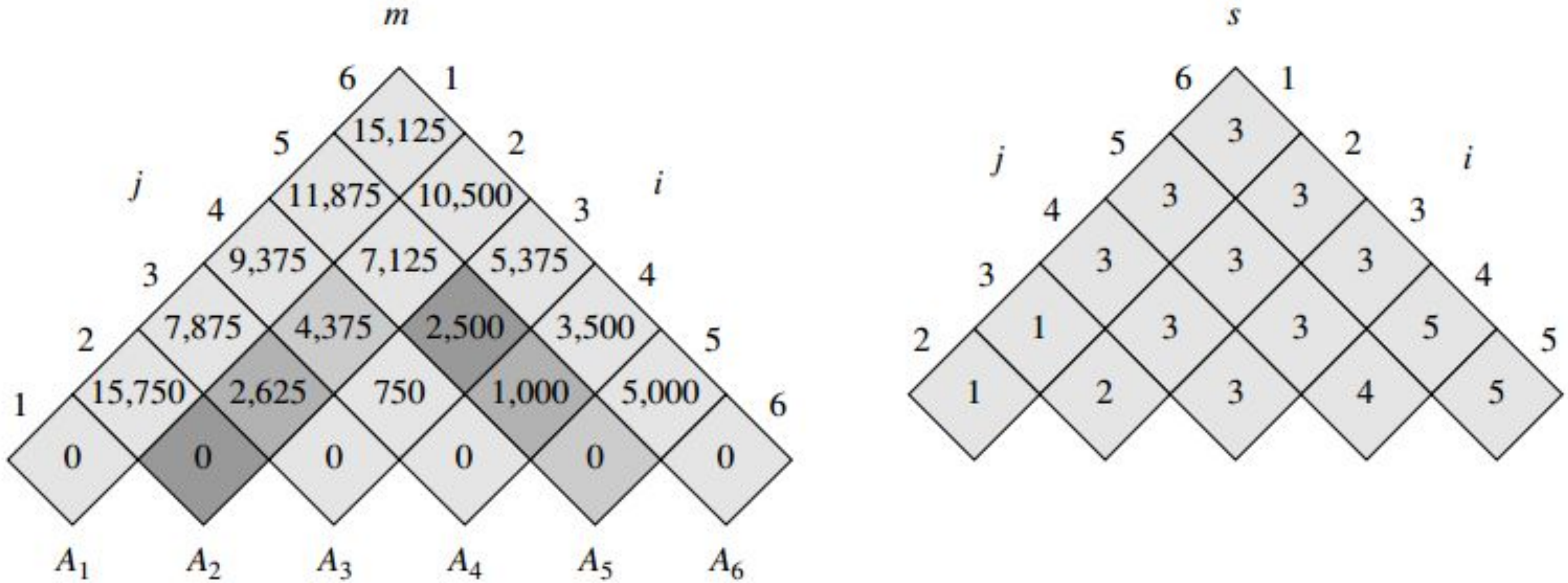

Figure 15.3 The m and s tables computed by MATRIX-CHAIN-ORDER for $n = 6$ and the following matrix dimensions:

matrix	dimension
A_1	30×35
A_2	35×15
A_3	15×5
A_4	5×10
A_5	10×20
A_6	20×25

A1 = 30 X 35
A2 = 35 X 15
A3 = 15 X 05
A4 = 05 X 10
A5 = 10 X 20
A6 = 20 X 25

$$m[i,j] = m[i,k] + m[k+1,j] + P_{i-1}*P_k*P_j$$

$$m[2,5] = \min \begin{cases} m[2,2] + m[3,5] + p_1 p_2 p_5 = 0 + 2500 + 35 \cdot 15 \cdot 20 & = 13000\,, \\ m[2,3] + m[4,5] + p_1 p_3 p_5 = 2625 + 1000 + 35 \cdot 5 \cdot 20 & = 7125\,, \\ m[2,4] + m[5,5] + p_1 p_4 p_5 = 4375 + 0 + 35 \cdot 10 \cdot 20 & = 11375 \end{cases}$$

$$= 7125\,.$$