Matrix Chain Multiplication (MCM)

Problem:

Given a chain $\langle A_1, A_2, \ldots, A_n \rangle$ of *n* matrices, where for $i = 0,1,\ldots,n$, matrix A_i has dimension $p_{i-1} \times p_i$, fully parenthesize the product $A_1 A_2 \ldots A_n$ in a way that minimizes the number of scalar multiplications.

Matrix Chain Multiplication

- Suppose we have a sequence or chain A_1, A_2, …, A_n of *n* matrices to be multiplied
  - That is, we want to compute the product $A_1A_2 \ldots A_n$
- There are many possible ways (parenthesizations) to compute the product

Matrix Chain Multiplication ...contd

- Example: consider the chain A_1, A_2, A_3, A_4 of 4 matrices
  - Let us compute the product $A_1A_2A_3A_4$
- There are 5 possible ways:
  1. $(A_1(A_2(A_3A_4)))$
  2. $(A_1((A_2A_3)A_4))$
  3. $((A_1A_2)(A_3A_4))$
  4. $((A_1(A_2A_3))A_4)$
  5. $(((A_1A_2)A_3)A_4)$

Background: Matrix Multiplication

- To compute the number of scalar multiplications necessary, we must know:
  - Algorithm to multiply two matrices
  - Matrix dimensions

- Let *A* be a $p \times q$ matrix

  B be a $q \times r$ matrix.

  Then the complexity is

$$p \times q \times r$$

Algorithm: Multiply 2 Matrices

Input: Matrices $A_{p \times q}$ and $B_{q \times r}$ (with dimensions $p \times q$ and $q \times r$)

Result: Matrix $C_{p \times r}$ resulting from the product $A \cdot B$

MATRIX-MULTIPLY($A_{p \times q}$, $B_{q \times r}$)

```
1.  for i ← 1 to p
2.      for j ← 1 to r
3.          C[i, j] ← 0
4.          for k ← 1 to q
5.              C[i, j] ← C[i, j] + A[i, k] · B[k, j]
6.  return C
```

Scalar multiplication in line 5 dominates time to compute C

Number of scalar multiplications = pqr

Example: Complexity to Multiply

- Example: Consider three matrices $A_{10\times100}$, $B_{100\times5}$, and $C_{5\times50}$
- There are 2 ways to parenthesize
  - $((AB)C) = D_{10\times5} \cdot C_{5\times50}$
    - $AB \Rightarrow 10\cdot100\cdot5=5{,}000$ scalar multiplications
    - $DC \Rightarrow 10\cdot5\cdot50 =2{,}500$ scalar multiplications

    Total: 7,500
  - $(A(BC)) = A_{10\times100} \cdot E_{100\times50}$
    - $BC \Rightarrow 100\cdot5\cdot50=25{,}000$ scalar multiplications
    - $AE \Rightarrow 10\cdot100\cdot50 =50{,}000$ scalar multiplications

Total: 75,000

Recall: MCM Problem

- Given a chain of matrices $\langle A_1, A_2, \ldots, A_n \rangle$, where for i = 1, 2, …, n matrix A_i has dimensions $p_{i-1} \times p_i$, fully parenthesize the product $A_1 \cdot A_2 \cdots A_n$ in a way that minimizes the number of scalar multiplications.

$$\underset{p_0 \times p_1}{A_1} \cdot \underset{p_1 \times p_2}{A_2} \cdots\cdots \underset{p_{i-1} \times p_i}{A_i} \cdot \underset{p_i \times p_{i+1}}{A_{i+1}} \cdots\cdots \underset{p_{n-1} \times p_n}{A_n}$$

Dynamic Programming Approach

- Step 1: The structure of an optimal solution
  - Let us use the notation $A_{i..j}$ for the matrix that results from the product $A_i\, A_{i+1} \ldots A_j$
  - An optimal parenthesization of the product $A_1A_2\ldots A_n$ splits the product between A_k and A_{k+1} for some integer k where $1 \le k < n$
  - First compute matrices $A_{1..k}$ and $A_{k+1..n}$; then multiply them to get the final matrix $A_{1..n}$

Dynamic Programming Approach ...contd

Optimal

$$((A_1 A_2 \dots A_k)(A_{k+1} A_{k+2} \dots A_n))$$

Combine

Dynamic Programming Approach ...contd

- Step 2: Recursive definition of the value of an optimal solution
  - Define *m*[*i*, *j*] = Minimum number of scalar multiplications necessary to compute $A_{i..j}$

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

  - Goal *m[1, n]* = Minimum cost to compute $A_{1..n}$

Dynamic Programming Approach …contd

– Assume that the optimal parenthesization splits the product $A_i\ A_{i+1} \cdots A_j$ at k ($i \le k < j$)

– $A_{i..j} = (A_i A_{i+1} \ldots A_k) \cdot (A_{k+1} A_{k+2} \ldots A_j)$

$p_{i-1}p_kp_j$

$$= A_{i\ldots k}\ A_{k+1\ldots j} \qquad \text{for } i \le k < j$$

m[i, k] m[k+1,j]

Dynamic Programming Approach ...contd

– Cost of computing $A_{i..j}$ = cost of computing $A_{i..k}$ + cost of computing $A_{k+1..j}$ + cost of multiplying $A_{i..k}$ and $A_{k+1..j}$

$$m[i, j] = \underbrace{m[i, k]}_{\substack{\text{min \# of}\\ \text{multiplications}\\ \text{to compute } A_{i...k}}} + \underbrace{m[k+1, j]}_{\substack{\text{min \# of}\\ \text{multiplications}\\ \text{to compute } A_{k+1...j}}} + \underbrace{p_{i-1}p_k p_j}_{\substack{\text{\# of multiplications}\\ \text{to compute}\\ A_{i...k}A_{k+1...j}}}$$

– Cost of multiplying $A_{i..k}$ and $A_{k+1..j}$ is $p_{i-1}p_k\,p_j$

– $m[i, i\,] = 0$ for i=1,2,…,n

Dynamic Programming Approach ...contd

– But… optimal parenthesization occurs at one value of k among all possible $i \le k < j$
– Check all these and select the best one

$$m[i, j] = \begin{cases} 0 & \text{if } i=j \\ \min\limits_{i \le k < j} \{m[i, k] + m[k+1, j] + p_{i-1}p_k p_j\} & \text{if } i<j \end{cases}$$

Dynamic Programming Approach ...contd

- To keep track of how to construct an optimal solution, we use a table *s*

- *s*[*i*, *j*] = value of *k* at which $A_i A_{i+1} \ldots A_j$ is split for optimal parenthesization

Dynamic Programming Approach …contd

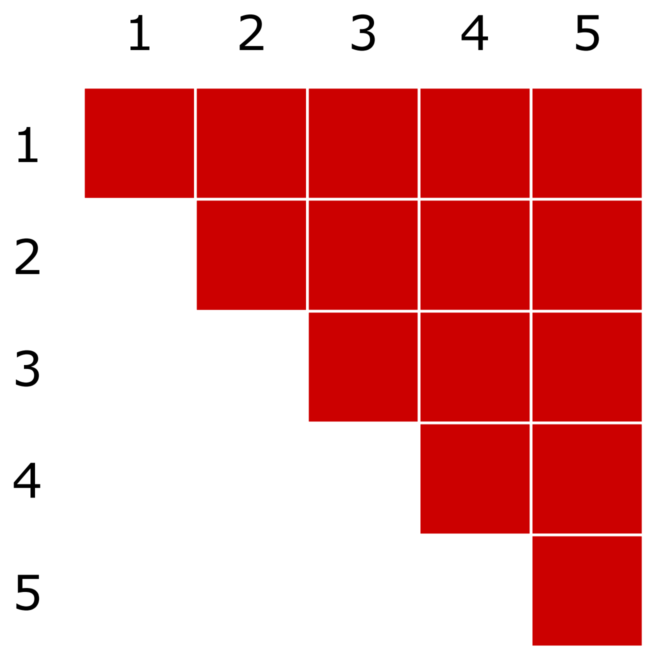
- Step 3: Computing the Optimal Costs

➢ Algorithm:

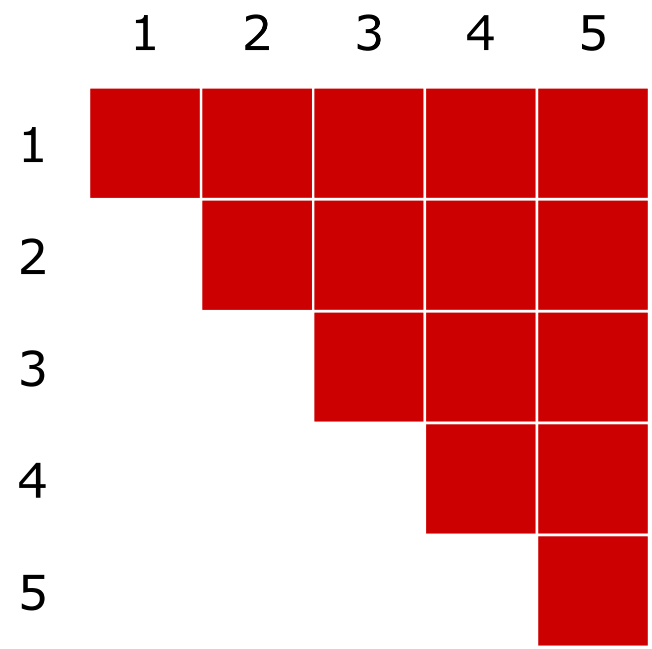
- First computes costs for chains of length l=1
- Then for chains of length l=2,3, … and so on
- Computes the optimal cost bottom-up

Example: Step 1

- $n = 5$, $p = (10, 5, 1, 10, 2, 10)$
  - $[10\times5]\times[5\times1]\times[1\times10]\times[10\times2]\times[2\times10]$



$m(i,j)$, $i \le j$



$s(i,j)$, $i \le j$

Example: Step 2

- p = [10×5]×[5×1]×[1×10]×[10×2]×[2×10]
- m(i,i) = 0

	1	2	3	4	5
1	0				
2		0			
3			0		
4				0	
5					0

	1	2	3	4	5
1					
2					
3					
4					
5					

Example: Step 3

- $p = [10 \times 5] \times [5 \times 1] \times [1 \times 10] \times [10 \times 2] \times [2 \times 10]$
- $m(i,i+1) = p_{i-1} p_i p_{i+1}$
- $s(i,i+1) = i$

	1	2	3	4	5
1	0	50			
2		0	50		
3			0	20	
4				0	200
5					0

	1	2	3	4	5
1		1			
2			2		
3				3	
4					4
5					

Example: Step 4

- $p = [10\times5]\times[5\times1]\times[1\times10]\times[10\times2]\times[2\times10]$
- $m(i,i+2) = \min\{\, m(i,i) + m(i+1,i+2) + p_{i-1}p_ip_{i+2},$
  $m(i,i+1) + m(i+2,i+2) + p_ip_{i+1}p_{i+2}\}$

	1	2	3	4	5
1	0	50			
2		0	50		
3			0	20	
4				0	200
5					0

	1	2	3	4	5
1		1			
2			2		
3				3	
4					4
5					

Example: Step 5

- $p = [10\times5]\times[5\times1]\times[1\times10]\times[10\times2]\times[2\times10]$
- $m(2,4) = \min\{\, m(2,2) + m(3,4) + p_1p_2p_4,$
  $\qquad m(2,3) + m(4,4) + p_1p_3p_4 \}$
- $m(3,5) = \ldots$

	1	2	3	4	5
1	0	50	150		
2		0	50	30	
3			0	20	40
4				0	200
5					0

	1	2	3	4	5
1		1	2		
2			2	2	
3				3	4
4					4
5					

Example: Step 6

– $p = [10 \times 5] \times [5 \times 1] \times [1 \times 10] \times [10 \times 2] \times [2 \times 10]$

– $m(i,i+3) = \min\{\, m(i,i) + m(i+1,i+3) + p_{i-1}p_ip_{i+3},$
$\quad m(i,i+1) + m(i+2,i+3) + p_{i-1}p_{i+1}p_{i+3},$
$\quad m(i,i+2) + m(i+3,i+3) + p_{i-1}p_{i+2}p_{i+3} \}$

	1	2	3	4	5
1	0	50	150	90	
2		0	50	30	90
3			0	20	40
4				0	200
5					0

	1	2	3	4	5
1		1	2	2	
2			2	2	2
3				3	4
4					4
5					

Example: Step 7

- $p = [10\times5]\times[5\times1]\times[1\times10]\times[10\times2]\times[2\times10]$
- $m(i,i+4) = \min\{\, m(i,i) + m(i+1,i+4) + p_{i-1}p_ip_{i+4},$
  $m(i,i+1) + m(i+2,i+4) + p_{i-1}p_{i+1}p_{i+4},$
  $m(i,i+2) + m(i+3,i+4) + p_{i-1}p_{i+2}p_{i+4},$
  $m(i,i+3) + m(i+4,i+4) + p_{i-1}p_{i+3}p_{i+4}\}$

	1	2	3	4	5
1	0	50	150	90	190
2		0	50	30	90
3			0	20	40
4				0	200
5					0

$m(i,j),\ i \le j$

	1	2	3	4	5
1		1	2	2	2
2			2	2	2
3				3	4
4					4
5					

$s(i,j),\ i \le j$

Dynamic Programming Approach …contd

Algorithm: Compute the Optimal Cost

Input: Array *p* containing matrix dimensions
Output: Minimum-cost table *m* and split table *s*

MATRIX-CHAIN-ORDER(*p*)

```
n ← length[p] – 1
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
```

Dynamic Programming Approach …contd

- Step 4: Constructing an Optimal Solution

➢ The optimal solution can be constructed from the split table **s**

– Each entry $s[i, j\ ] = k$ shows where to split the product $A_i\, A_{i+1} \ldots A_j$ for the minimum cost

Example: Step 1.1

- Optimal multiplication sequence
  - $s(1,5) = 2$
    - ▶ $A_{15} = A_{12} \times A_{35}$

	1	2	3	4	5
1	0	50	150	90	190
2		0	50	30	90
3			0	20	40
4				0	200
5					0

m(i,j), i ≤ j

	1	2	3	4	5
1		1	2	2	2
2			2	2	2
3				3	4
4					4
5					

s(i,j), i ≤ j

Example: Step 1.2

– $A_{15} = A_{12} \times A_{35}$

– $s(1,2) = 1$ ▶ $A_{12} = A_{11} \times A_{22}$

$\rightarrow A_{15} = (A_{11} \times A_{22}) \times A_{35}$

	1	2	3	4	5
1	0	50	150	90	190
2		0	50	30	90
3			0	20	40
4				0	200
5					0

	1	2	3	4	5
1		1	2	2	2
2			2	2	2
3				3	4
4					4
5					

Example: Step 1.3

– $A_{15} = (A_{11} \times A_{22}) \times A_{35}$

– $s(3,5) = 4$ ▶ $A_{35} = A_{34} \times A_{55}$

$\rightarrow A_{15} = (A_{11} \times A_{22}) \times (A_{34} \times A_{55})$

	1	2	3	4	5
1	0	50	150	90	190
2		0	50	30	90
3			0	20	40
4				0	200
5					0

	1	2	3	4	5
1		1	2	2	2
2			2	2	2
3				3	4
4					4
5					

Example: Step 1.4

– $A_{15} = (A_{11} \times A_{22}) \times (A_{34} \times A_{55})$

– $s(3,4) = 3$ ▶ $A_{34} = A_{33} \times A_{44}$

$\rightarrow A_{15} = (A_{11} \times A_{22}) \times ((A_{33} \times A_{44}) \times A_{55})$

	1	2	3	4	5
1	0	50	150	90	190
2		0	50	30	90
3			0	20	40
4				0	200
5					0

m(i,j), i ≤ j

	1	2	3	4	5
1		1	2	2	2
2			2	2	2
3				3	4
4					4
5					

s(i,j), i ≤ j

Dynamic Programming Approach …contd

Algorithm: Constructing an Optimal Solution

```
PRINT-OPTIMAL-PARENS(s, i, j)
if i == j
    print "A"_i
else print "("
    PRINT-OPTIMAL-PARENS(s, i, s[i, j])
    PRINT-OPTIMAL-PARENS(s, s[i, j] + 1, j)
    print ")"
```

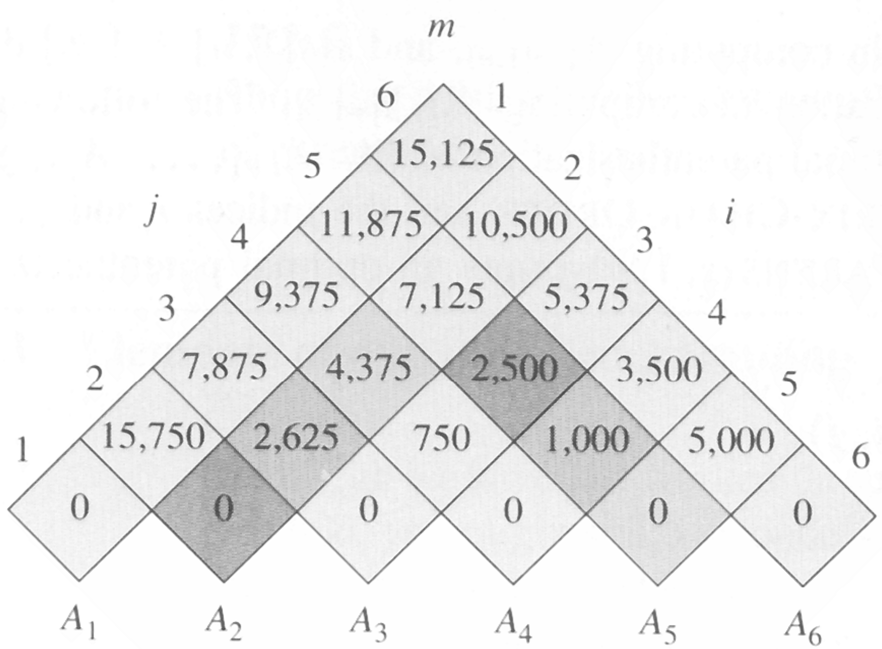
Practice Example (Book: Cormen; Page-376)

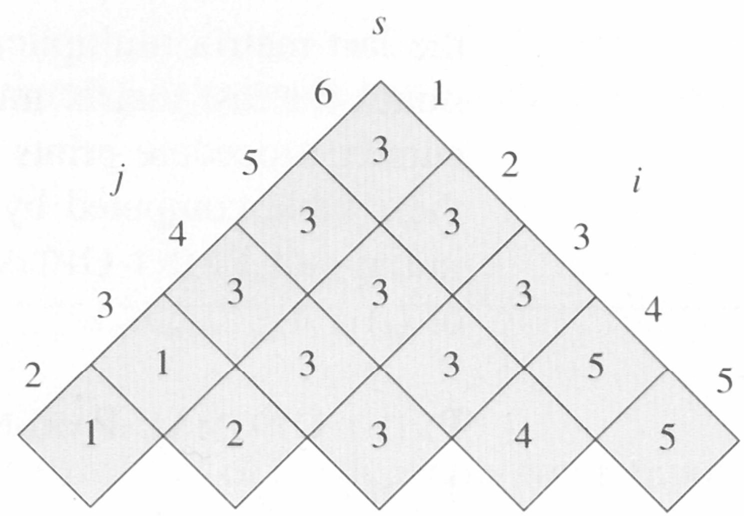
- Show how to multiply this matrix chain optimally

$$
\begin{array}{lll}
A_1 & 30\times 35 & = p_0 \times p_1 \\
A_2 & 35\times 15 & = p_1 \times p_2 \\
A_3 & 15\times 5 & = p_2 \times p_3 \\
A_4 & 5\times 10 & = p_3 \times p_4 \\
A_5 & 10\times 20 & = p_4 \times p_5 \\
A_6 & 20\times 25 & = p_5 \times p_6
\end{array}
$$

- Solution:
  - Minimum cost 15,125
  - Optimal parenthesization $((A_1(A_2A_3))((A_4A_5)A_6))$

The m and s table computed by MATRIX-CHAIN-ORDER





Practice Example: Solution Sample

$\boldsymbol{m}[2,5]=$

$\min\{$

$\quad \boldsymbol{m[2,2]+m[3,5]+p_1p_2p_5=0+2500+35\times15\times20=13000},$

$\quad \boldsymbol{m[2,3]+m[4,5]+p_1p_3p_5=2625+1000+35\times5\times20=7125},$

$\quad \boldsymbol{m[2,4]+m[5,5]+p_1p_4p_5=4375+0+35\times10\times20=11374}$

$\}$

$\boldsymbol{=7125}$

Thank You

Stay Safe