Chain Matrix Multiplication

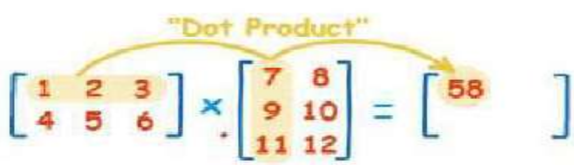
- Given a sequence or chain $A_1, A_2, \ldots, A_n$ of n matrices to be multiplied, then How to compute the product $A_1A_2\ldots A_n$

Idea

- Matrix Multiplication is not commutative. That is $AB \neq BA$.
- But Associative

  $(AB)C = A(BC)$
- But, There are many possible ways of placing parenthesis

Matrix Multiplication cost



$A_{m \times \mathrm{n}}$ and $B_{n \times r}$ (with dimensions m×n and n×r)

Number of scalar multiplications = *mnr*

Cost of Multiplication

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

Algorithm Segment

Input: Matrices $A_{m\times n}$ and $B_{n\times r}$ (with dimensions m×n and n×*r*)

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

for $i \leftarrow 1$ **to** m
    for $j \leftarrow 1$ **to** r
        $C[i, j] \leftarrow 0$
        for $k \leftarrow 1$ **to** n
            $C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$
return C

Number of scalar multiplications = *mnr*

Why Order Matters?

- Example: Consider three matrices $A_{2\times3}$, $B_{3\times4}$, and $C_{4\times5.}$
- There are 2 ways to parenthesize
  - $((AB)C) = D_{2\times4} \cdot C_{4\times5}$
    - AB $\Rightarrow$ 2 x 3 x4 = 24 scalar multiplications
    - DC $\Rightarrow$ 2 x 4 x 5 = 40 scalar multiplications
    - Total = 24 + 40 = 64 multiplications.

Another Way

- $(A(BC)) = A_{2\times4} \cdot E_{3\times5}$
  - BC $\Rightarrow$ 3 x 4 x 5 = 60 scalar multiplications
  - AE $\Rightarrow$ 2 x 3 x 5 = 30 scalar multiplications
  - Total = 60 + 30 = 90 scalar multiplications.
- So cost and order matters !!

Examples 13.9 Let us consider the following three matrices:

$$A: 2 \times 3; \quad B: 3 \times 4; \quad C: 4 \times 5$$

What are the possible orderings? What is the optimal order?

Solution Three matrices are given. Hence, two possible orderings are possible. The possible orderings for three matrices are $((AB)C)$ and $(A(BC))$. The cost of multiplying two matrices $A(i \times j)$ and $B(j \times k)$ is $i \times j \times k$.

$$[(AB)C] = (2 \times 3 \times 4) + (2 \times 4 \times 5) = 24 + 40 = 64$$
$$[A(BC)] = (3 \times 4 \times 5) + (2 \times 3 \times 5) = 60 + 30 = 90$$

Hence, the optimal order is $[(AB)C]$.

N Order

$A_1\{A_2,\ldots,A_n\}$

$\{A_1A_2\}\{A_3,\ldots,A_n\}$

$\vdots$

$\{A_1,A_2,\ldots,A_{n-1}\}A_n$

Example

- Example: consider the chain A_1, A_2, A_3, A_4 of 4 matrices. Then 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. $(((A_1A_2)A_3)A_4)$

Catalan Sequence

It can be observed that the number of possible resulting trees is a Catalan number. As discussed earlier in Chapter 6, the n^{th} Catalan number C_n is given as follows:

$$C_n = \frac{1}{n+1}\binom{2n}{n} \quad \text{for } n \geq 0$$

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

This leads to a sequence called Catalan sequence

Need for Optimization

- Optimization is necessary !
  - Given a chain $A_1, A_2, \ldots, A_n$ of n matrices, where for i=1, 2, …, n, matrix A_i has dimension $p_{i-1} \times p_i$
  - Parenthesize the product $A_1 A_2 \ldots A_n$ such that the total number of scalar multiplications is minimized

Recursive Definition

- Recursive definition of the value of an optimal solution
  - Let $C[i, j]$ be the minimum number of scalar multiplications necessary to compute $A_{i..j}$
  - Minimum cost to compute $A_{1..n}$ is $C[1, n]$
  - Suppose the optimal parenthesization of $A_{i..j}$ splits the product between A_k and A_{k+1} for some integer k where $i \leq k < j$

Why Order Matters?

$A_{2\times3}$, $B_{3\times4}$, and $C_{4\times5}$. with dimensions A(P0,P1), B(P1,P2), C(P2,P3)

(A(BC)) = (k=1)

$C[1,3] = C[1,1] + C[2,3] + P_0 \cdot P_1 \cdot P_3$

- ((AB)C) = (k=2)

$C[1,3] = C[1,2] + C[3,3] + P_0 \cdot P_2 \cdot P_3$

Finally, it is a choice between k=1 and k =2
Thus, OPTIMIZATION ! As Minimize cost!

Recursive Formulation

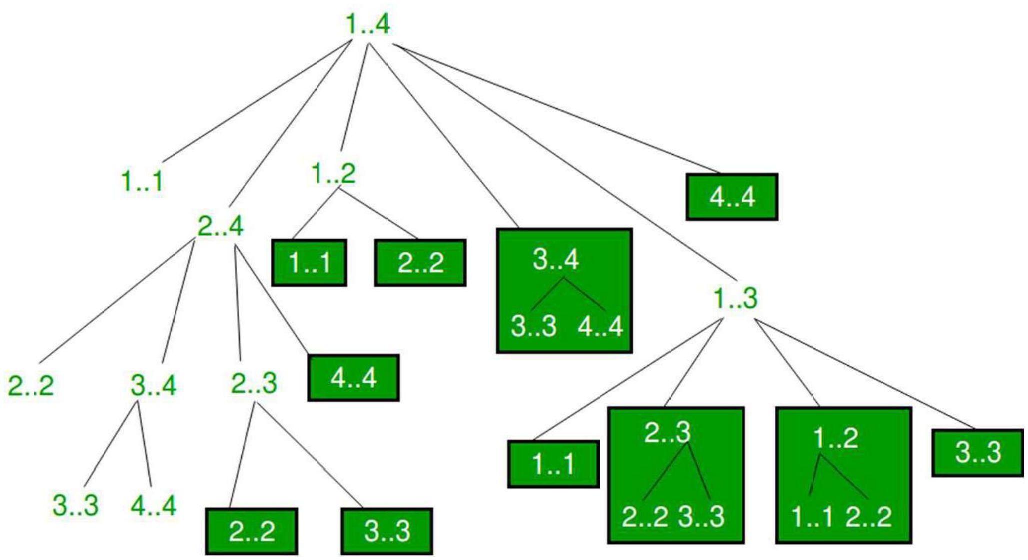
$C[i, j] = C[i, k] + C[k+1, j] + p_{i-1} p_k p_j$ for $i \leq k < j$

– $C[i, i] = 0$ for $i=1,2,\ldots,n$ *(Initial Condition)*

Optimization Problem

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

Overlapping subproblems



Informal Algorithm

- Read n chain of matrices
- Compute C[i,j] recursively and fill the table
- Compute R[i,j] to keep track of k that yields minimum cost
- Return M[1,n] as minimum cost.

Formal Algorithm

```
Algorithm dp_chainmult(p,n)
Begin
        for i = 1 to n  do
                C[ i , j ] = 0
        end for

        for diagonal = 1 to n-1
            for  i = 1  to n-diagonal
                j = i + diagonal
                    C[ i , j ] = ∞
                    for k = 1 to j-1 do
```

Formal Algorithm

```
        if C[ i , j ] < C[ i , k ]+C[ k+1 , j]+ p_{i-1} × p_k × p_j then
            C[i,j]= C[ i , k ]+C[ k+1 , j]+ p_{i-1} × p_k × p_j
            R[i,j] = k
        else
            C[i,j] = C[i,j]
            R[i,j] = k
        End if
      End for
    End for
  end for
return C[1,n]
```

Example

Perform chained matrix multiplication.

A	B	C	D
4 × 5	5 × 3	3 × 2	2 × 7
P_0 P_1	P_1 P_2	P_2 P_3	P_3 P_4

Example

C[1,1] = 0 ; C[2,2] = 0 ; C[3,3] = 0; C[4,4] = 0

Table 1: Initial Table

0			
	0		
		0	
			0

Time complexity

$$C[1,2] = C[1,1] + C[2,2] + P_0 \cdot P_1 \cdot P_2$$
$$= 0 + 0 + 4 \times 5 \times 3 = 60$$
$$C[2,3] = C[2,2] + C[3,3] + P_1 \cdot P_2 \cdot P_3$$
$$= 0 + 0 + 5 \times 3 \times 2 = 30$$
$$C[3,4] = C[3,3] + C[4,4] + P_2 \cdot P_3 \cdot P_4$$
$$= 0 + 0 + 3 \times 2 \times 7 = 42$$

Table 2: After First Diagonal

0	60		
	0	30	
		0	42
			0

Example

$C[1,3] = C[1,1] + C[2,3] + P_0 \cdot P_1 \cdot P_3$
$= 0 + 30 + 4 \times 5 \times 2$
$= 30 + 40 = 70 \quad (k = 1)$
$C[1,3] = C[1,2] + C[3,3] + P_0 \cdot P_2 \cdot P_3$
$= 60 + 0 + 4 \times 3 \times 2$
$= 84 \quad (k = 2)$

The minimum is 70 when $k = 1$

$C[2,4] = C[2,2] + C[3,4] + P_1 \cdot P_2 \cdot P_4$
$= 0 + 42 + 5 \times 3 \times 7$
$= 42 + 105 = 147 \quad (k = 2)$
$C[2,4] = C[2,3] + C[4,4] + P_1 \cdot P_3 \cdot P_4$
$= 30 + 0 + 5 \times 2 \times 7$
$= 30 + 70 = 100 \quad (k = 3)$

The minimum is 100 when k = 3. The matrix now appears as Table 3.

Table 3: After second diagonal computation

0	60	70	
	0	30	100
		0	42
			0

Example

Now, C[1,4] is computed.

$$C[1,4] = C[1,1] + C[2,4] + P_0 \cdot P_1 \cdot P_4$$
$$= 0 + 10\,0 + 4 \times 5 \times 7$$
$$= 100 + 140 = 200 \qquad (k = 1)$$
$$C[1,4] = C[1,2] + C[3,4] + P_0 \cdot P_2 \cdot P_4$$
$$= 60 + 42 + 4 \times 3 \times 7$$
$$= 60 + 42 + 84 = 186 \qquad (k = 2)$$
$$C[1,4] = C[1,3] + C[4,4] + P_0 \cdot P_3 \cdot P_4$$
$$= 70 + 0 + 4 \times 2 \times 7$$
$$= 70 + 56 = 126 \qquad (k = 3)$$

The minimum is 126 and this happens when $k = 3$.

Table 4: Final Table

0	60	70	126
	0	30	100
		0	42
			0

Table 5: Table of minimum k

0	1	1	3
	0	2	3
		0	3
			0

[A (B C) D]

Example

Examples 13.11 Consider the following four matrices whose orders are given and perform chain matrix multiplication using the dynamic programming approach.

A	B	C	D
4×5	5×3	3×2	2×7
$p_0 p_1$	$p_1 p_2$	$p_2 p_3$	$p_3 p_4$

Solution Four matrices are given. A table M is created to store the intermediate results. As per the algorithm, the entries of matrix M are initialized as follows:

$$M[1, 1] = 0; M[2, 2] = 0; M[3, 3] = 0; M[4, 4] = 0$$

The resultant Table 13.16 is an initial table.

Table 13.16 Initial table

0			
	0		
		0	
			0

Now, let us compute the first super diagonal as follows:

$$M[1, 2] = M[1, 1] + M[2, 2] + p_0 \cdot p_1 \cdot p_2$$
$$= 0 + 0 + 4 \times 5 \times 3 = 60$$
$$M[2, 3] = M[2, 2] + M[3, 3] + p_1 \cdot p_2 \cdot p_3$$
$$= 0 + 0 + 5 \times 3 \times 2 = 30$$
$$M[3, 4] = M[3, 3] + M[4, 4] + p_2 \cdot p_3 \cdot p_4$$
$$= 0 + 0 + 3 \times 2 \times 7 = 42$$

The resultant table is Table 13.17.

Table 13.17 After first diagonal

0	60		
	0	30	
		0	42
			0

Next, the second super diagonal needs to be computed. This implies that $M[1 \ldots 3]$ needs to be computed. Two splits are possible, with $k = 1$ and $k = 2$. The resulting computation is as follows:

$$M[1, 3] = M[1, 1] + M[2, 3] + p_0 \cdot p_1 \cdot p_3$$
$$= 0 + 30 + 4 \times 5 \times 2$$
$$= 30 + 40 = 70 \qquad (k = 1)$$
$$M[1, 3] = M[1, 2] + M[3, 3] + p_0 \cdot p_2 \cdot p_3$$
$$= 60 + 0 + 4 \times 3 \times 2$$
$$= 84 \qquad (k = 2)$$

The minimum is 70 when $k = 1$ Therefore, this must be noted in another table R. Thus, table R records k that gives the minimum cost. This process is repeated for other possibilities:

$$M[2, 4] = M[2, 2] + M[3, 4] + p_1 \cdot p_2 \cdot p_4$$
$$= 0 + 42 + 5 \times 3 \times 7$$
$$= 42 + 105 = 147 \qquad (k = 2)$$
$$M[2, 4] = M[2, 3] + M[4, 4] + p_1 \cdot p_3 \cdot p_4$$
$$= 30 + 0 + 5 \times 2 \times 7$$
$$= 30 + 70 = 100 \qquad (k = 3)$$

The minimum is 100 when $k = 3$.

The resultant matrix now appears as shown in Table 13.18. Now, $M[1, 4]$ is computed. There are three possible splits for k. The possible splits and the resultant computation are as follows:

Table 13.18 After second super diagonal computation

0	60	70	
	0	30	100
		0	42
			0

$$M[1, 4] = M[1, 1] + M[2, 4] + p_0 \cdot p_1 \cdot p_4$$
$$= 0 + 100 + 4 \times 5 \times 7$$
$$= 100 + 140 = 200 \qquad (k = 1)$$
$$M[1, 4] = M[1, 2] + M[3, 4] + p_0 \cdot p_2 \cdot p_4$$
$$= 60 + 42 + 4 \times 3 \times 7$$
$$= 60 + 42 + 84 = 186 \qquad (k = 2)$$
$$M[1, 4] = M[1, 3] + M[4, 4] + p_0 \cdot p_3 \cdot p_4$$
$$= 70 + 0 + 4 \times 2 \times 7$$
$$= 70 + 56 = 126 \qquad (k = 3)$$

It can be observed that the minimum cost is 126 and this happens when $k = 3$. The resultant matrix is given in Table 13.19.

As mentioned earlier, all values of k that yields the minimum cost is recorded in table R. The final resultant table that records the minimum k is Table 13.20.

Table 13.19 Final table

0	60	70	126
	0	30	100
		0	42
			0

Table 13.20 Table *R* of minimum *k*

0	1	1	3
	0	2	3
		0	3
			0

It can be observed that $R(1, 4)$ is 3. Hence, the split is at point $k = 3$. This gives the order $((ABC)D)$. To split ABC, check $R(1, 3)$; it is 2. Therefore, the final chain of matrix can be now represented as follows:

$[A(BC)D]$

Trace of k

Step 1: Read the trace matrix R that has minimum k, which yields the minimum cost.
Step 2: Perform recursive call as follows:

If $(i \neq j)$ then
$k = R[i, j]$
return (mult($R_{1...k}$) $\times$ mult($R_{k+1...n}$))
else
return($R(i, j)$)

Step 3: End.

Complexity analysis

There are three for-loops in the algorithm, and each loop is executed n times. Therefore, the complexity of the algorithm is $\Theta(n^3)$.

Time complexity

Takes $O(n^3)$ time

Requires $O(n^2)$ space