Dynamic Programming
Chain Matrix Multiply

(Class 19)

From Book’s Page No 373 (Chapter 14)

- Suppose we wish to multiply a series of matrices:

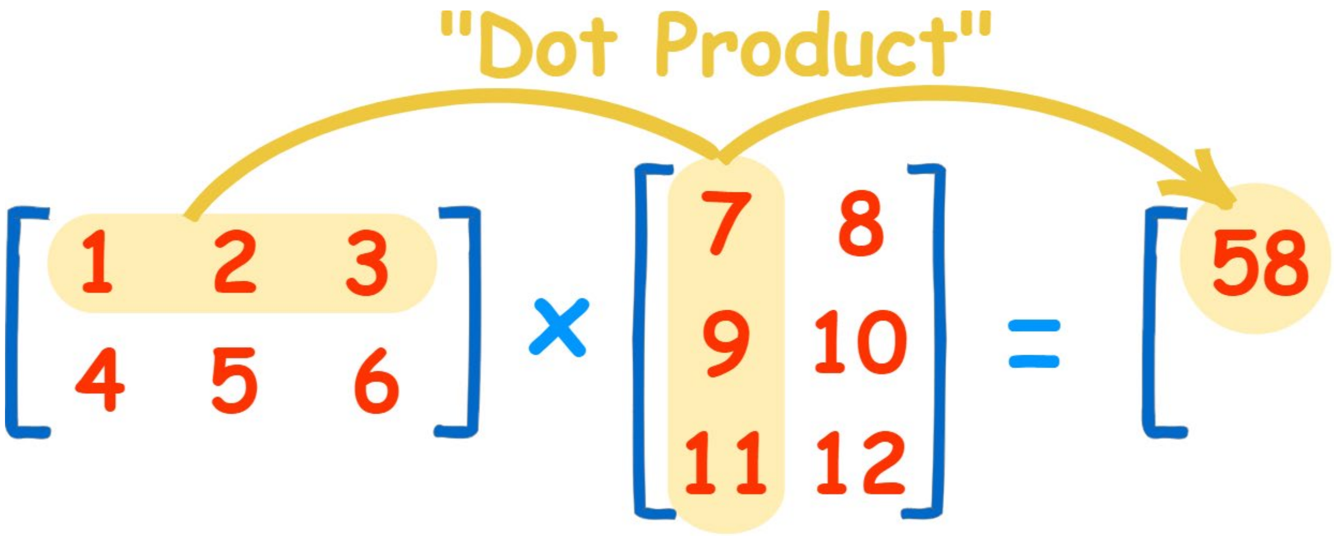
$$A_1 A_2 \ldots A_n$$

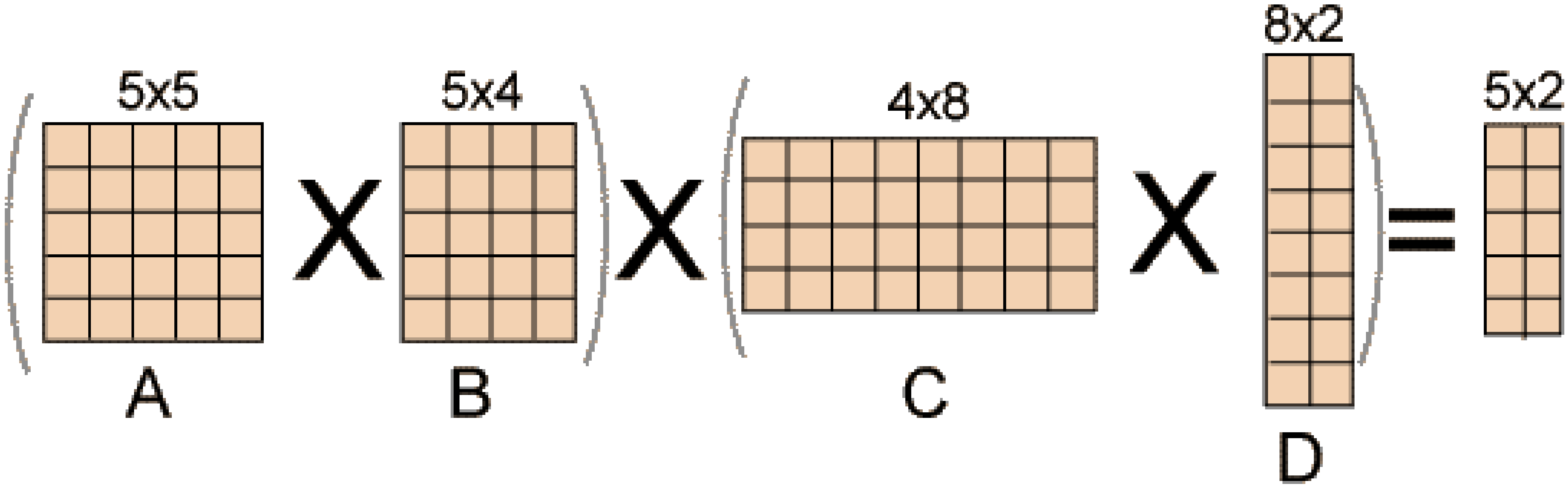
- Problem Statement: In what order should the multiplication be done?
- A $p \times q$ matrix A can be multiplied with a $q \times r$ matrix B.
- The result will be a $p \times r$ matrix C.
- For example, multiplying $A[2,3]$ with $B[3,2]$ will result in $C[2,2]$.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \text{ x } \begin{bmatrix} 10 & 11 \\ 20 & 21 \\ 30 & 31 \end{bmatrix}$$

$$= \begin{bmatrix} 1\text{x}10 + 2\text{x}20 + 3\text{x}30 & 1\text{x}11 + 2\text{x}21 + 3\text{x}31 \\ 4\text{x}10 + 5\text{x}20 + 6\text{x}30 & 4\text{x}11 + 5\text{x}21 + 6\text{x}31 \end{bmatrix}$$

$$= \begin{bmatrix} 10+40+90 & 11+42+93 \\ 40+100+180 & 44+105+186 \end{bmatrix} = \begin{bmatrix} 140 & 146 \\ 320 & 335 \end{bmatrix}$$


"Dot Product"
1 2 3
4 5 6
x
7 8
9 10
11 12
=
58

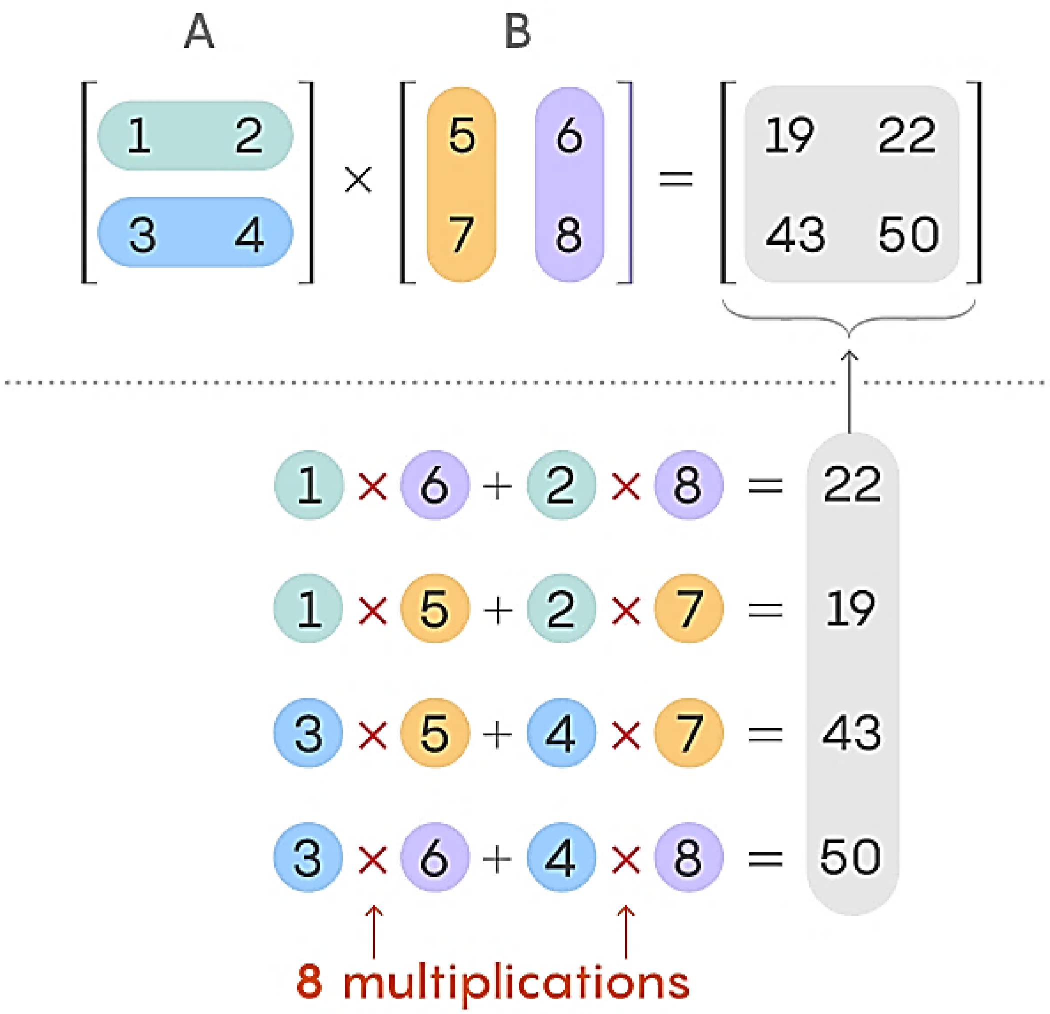

5x5
5x4
4x8
8x2
5x2
X
X
X
=
A
B
C
D

- In particular, for $1 \le i \le p$ and $1 \le j \le r$,

$$C[\mathrm{i}, \mathrm{j}] = \sum_{k=1}^{q} A[\mathrm{i}, \mathrm{k}]\mathrm{B}[\mathrm{k}, \mathrm{j}]$$

```
RECTANGULAR-MATRIX-MULTIPLY .A;B;C; p; q; r/
for i ← 1 to p
    for j ← 1 to r
        for k ← 1 to q
            c_ij ← c_ij + a_ik * b_kj
```

- There are $(p \cdot r)$ total entries in C and each takes $O(q)$ to compute.
- Thus, the total number of multiplications is $p \cdot q \cdot r$.



- Consider the case of 3 matrices:

$$A_1 = 5 \times 4$$

$$A_2 = 4 \times 6$$

$$A_3 = 6 \times 2$$

- The multiplication can be carried out either as $((A_1 A_2) A_3)$ or $(A_1 (A_2 A_3))$.
- The cost of the two is:

$$\big((A_1 A_2) A_3\big) = (5 \cdot 4 \cdot 6) + (5 \cdot 6 \cdot 2) = 180$$
$$\big(A_1 (A_2 A_3)\big) = (5 \cdot 4 \cdot 2) + (4 \cdot 6 \cdot 2) = 88$$

- There is considerable savings achieved even for this simple example.
- In general, however, in what order should we multiply a series of matrices $A_1 A_2 \dots A_n$?
- Matrix multiplication is associative but not commutative operation.
- We are free to add parenthesis the above multiplication, but the order of matrices cannot be changed.

- The *Chain Matrix Multiplication* problem is stated as follows:

"Given a chain of n matrices $A_1, A_2, \dots, A_n$ and dimensions $p_0, p_1, \dots, p_n$,
where matrix A_i has dimension $p_{i-1} \times p_i$, determine the order of multiplication that minimizes the number of scalar multiplications."

Counting the Number of Parenthesizations

- We could write a procedure that tries all possible parenthesizations.
- Unfortunately, the number of ways of parenthesizing an expression is very large.
- If there are n items, there are $n - 1$ ways in which outer most pair of parentheses can placed.

$$
\begin{array}{ll}
 & (A_1)(A_2A_3A_4\ldots A_n) \\
\text{or} & (A_1A_2)(A_3A_4\ldots A_n) \\
\text{or} & (A_1A_2A_3)(A_4\ldots A_n) \\
\ldots & \ldots \\
\text{or} & (A_1A_2A_3A_4\ldots A_{n-1})(A_n)
\end{array}
$$

- Once we split just after the k^{th} matrix, we create two sub lists to be parenthesized.
- One with k and other with $n - k$ matrices.

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

- We could consider all the ways of parenthesizing these two.

- Since these are independent choices, if there are L ways of parenthesizing the left sublist and R ways to parenthesize the right sublist, then the total is $L \cdot R$.
- For which value of k the result $L \cdot R$ will be minimum?
- This suggests the following recurrence for $P(n)$, the number of different ways of parenthesizing n items:

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

- This is related to a function called the *Catalan Numbers*.
- Catalan numbers are related the number of different binary trees on n nodes.
- We will get the running time:

$$P(n) \in \mathrm{O}(4^{\mathrm{n}}/\mathrm{n}^{3/2})$$

$$\boldsymbol{P(n) \in O(4^{n})}$$

- The dominating term is the exponential 4^n thus $P(n)$ will grow large very quickly.
- So, this brute-force method of exhaustive search makes for a poor strategy when determining how to optimally parenthesize a matrix chain.

Chain Matrix Multiplication - Dynamic Programming Formulation

- The dynamic programming solution involves breaking up the problem into subproblems whose solutions can be combined to solve the global problem.
- Let $A_{i..j}$ be the result of multiplying matrices i through j.
- It is easy to see that $A_{i..j}$ is a $p_{i-1} \times p_j$ matrix.

$$\underset{4\times 5}{A_3} \quad \underset{5\times 2}{A_4} \quad \underset{2\times 8}{A_5} \quad \underset{8\times 7}{A_6} = \underset{4\times 7}{A_{3..6}}$$

- At the highest level of parenthesization, we multiply two matrices:

$$A_{1..n} = A_{1..k} \cdot A_{k+1..n} \quad where\ 1 \leq k \leq n-1$$

- The question is what is the optimum value of k for the split?
- And how do we parenthesis the sub-chains $A_{1..k}$ and $A_{k+1..n}$.

- We cannot use divide and conquer because we do not know that what is the optimum value of k.
- We will have to consider all possible values of k and take the best of them.
- We will apply this strategy to solve the subproblems optimally.
- We will store the solutions to the subproblem in a table and build the table bottom-up.