Dynamic Programming (DP)

Algorithm design technique mainly used in optimization problems.

⇓

can have many feasible solutions but only one solution is optimal.

↙ ↘

Max. Min. (According to the given problem)

Steps in DP:

1. Characterize the structure of an optimal solution.
2. Recursively define the value of an optimal solution.
3. Compute the value of an optimal solution, typically in a bottom-up fashion.
4. Construct an optimal solution from computed information.

Note: If we need only the value of an optimal solution, not the solution itself, then we can omit step-4. Otherwise, we can construct optimal solution from the informations obtained in step-3.

Matrix Chain Multiplication (MCM)

Matrix multiplication:

- No. of columns of 1st matrix = No. of rows of 2nd matrix
- $(A \cdot B) \cdot C = A \cdot (B \cdot C) \Rightarrow$ Associative property.
- $A_{2\times3}, B_{3\times4} \Rightarrow P_{2\times4}$ (To get $P_{2\times4}$, we have to perform $2\times3\times4 = 24$ no. of scalar multiplications)

$$A_{2\times3} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \quad B_{3\times4} = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \end{bmatrix}$$

$$P_{2\times4} = \begin{bmatrix} a_{11} \cdot b_{11} + a_{12} \cdot b_{21} + a_{13} \cdot b_{31} & \cdots \\ \vdots & \end{bmatrix}$$

MCM Problem

$\Rightarrow$ Mimization type of optimization problem.

$\Rightarrow$ We are not concerned about the product of the matrix chain.

Our objective is to find out how we can fully parenthesize the product so that it minimizes the no. of scalar multiplications.

$$\begin{array}{ccc} A_1 & A_2 & A_3 \\ 2\times3 & 3\times4 & 4\times2 \end{array}$$

$$\underbrace{(A_1 \cdot A_2)}_{2\times4} \cdot A_3 \qquad\qquad A_1 \cdot \underbrace{(A_2 \cdot A_3)}_{3\times2}$$

$2\times3 \quad 3\times4 \quad 4\times2 \qquad\qquad 2\times3 \quad 3\times4 \quad 4\times2$

$= \underline{2\times3\times4} + \underline{0} + \underline{2\times4\times2} \qquad\qquad = 0 + \underline{3\times4\times2} + \underline{2\times3\times2}$

$= 40$ scalar multiplications $\qquad\qquad = 36$ scalar multiplications

Optimal parenthesization

$((A_1) \cdot (A_2 \cdot A_3))$

Formal definition of MCM problem: Given a chain $\langle A_1, A_2, \dots, A_n \rangle$ of n matrices, where for $i = 1, 2, \dots, n$, matrix A_i has dimension $P_{i-1} \times P_i$, fully parenthesize the product $A_1 A_2 \cdots A_n$ in a way that minimizes the no. of scalar multiplications.

Example:

A_1	A_2	A_3	A_4
5×4	4×6	6×2	2×7

$\langle 5, 4, 6, 2, 7 \rangle$

$m \rightarrow$ cost matrix

m	1	2	3	4
1	0	120	88	158
2		0	48	104
3			0	84
4				0

$\overset{A_1}{m[1,1]} = \overset{A_2}{m[2,2]} = \overset{A_3}{m[3,3]} = \overset{A_4}{m[4,4]} = 0$

$m[1,2]$

$A_1 \cdot A_2$

$5 \times 4 \quad 4 \times 6$

$5 \times 4 \times 6 = 120$

$((A_1) \cdot (A_2))$

$m[2,3]$

$A_2 \cdot A_3$

$4 \times 6 \quad 6 \times 2$

$4 \times 6 \times 2 = 48$

$m[3,4]$

$A_3 \cdot A_4$

$6 \times 2 \quad 2 \times 7$

$6 \times 2 \times 7 = 84$

$S \rightarrow$ Parenthesization matrix

s	1	2	3	4
1	0	1	1	3
2		0	2	3
3			0	3
4				0

$m[1,3]$

$A_1 \cdot A_2 \cdot A_3$

$5 \times 4 \quad 4 \times 6 \quad 6 \times 2$

$A_1 \cdot (\overset{4 \times 2}{A_2 \cdot A_3})$

$= m[1,1] + m[2,3] + 5 \times 4 \times 2$

$= 0 + 48 + 40 = 88$

$(\overset{5 \times 6}{A_1 \cdot A_2}) \cdot A_3$

$= m[1,2] + m[3,3] + 5 \times 6 \times 2$

$= 120 + 0 + 60 = \underline{\underline{180}}$

$m[2,4]$

$A_2 \cdot A_3 \cdot A_4$

$A_2 \cdot \overset{\nearrow 6\times 7}{(A_3 \cdot A_4)}$ | $\overset{\nearrow 4\times 2}{(A_2 \cdot A_3)} \cdot A_4$

$4\times 6 \quad 6\times 2 \quad 2\times 7$ | $4\times 6 \quad 6\times 2 \quad 2\times 7$

$m[2,2] + m[3,4] + 4\times 6\times 7$ | $m[2,3] + m[4,4] + 4\times 2\times 7$

$= 0 + 84 + 168$ | $= 48 + 0 + 56$

$= 252$ | $= 104$

$A_1 \cdot A_2 \cdot A_3 \cdot A_4$

$$m[1,4] = \min \begin{cases} \overset{\nearrow 5\times 4}{A_1} \cdot \overset{\nearrow 4\times 7}{(A_2 \cdot A_3 \cdot A_4)} \\ \underline{m[1,1] + m[2,4] + 5\times 4\times 7}, \\ m[1,2] + m[3,4] + 5\times 6\times 7, \\ (A_1 \cdot A_2) \cdot (A_3 \cdot A_4) \\ m[1,3] + m[4,4] + 5\times 2\times 7 \\ (A_1 \cdot A_2 \cdot A_3) \cdot A_4 \end{cases}$$

$$= \min \{0 + 104 + 140,\ 120 + 84 + 210,\ 88 + 0 + 70\}$$

$$= \min \{244, 414, \underline{\underline{158}}\} = 158$$

So, we need 158 scalar multiplications for this matrix chain.

<u>Optimal parenthesization:</u>

$$\underline{((A_1)(A_2\ A_3))(A_4)}$$

→ dimensions of matrices

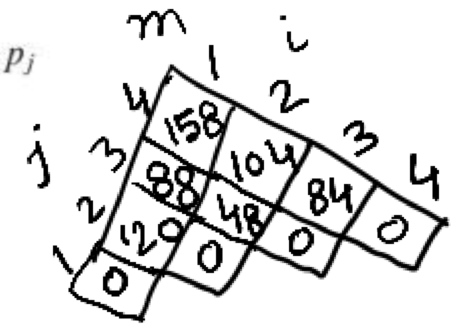
$P = \langle 5, 4, 6, 2, 7 \rangle$

p.length = 5

n^3 → Time

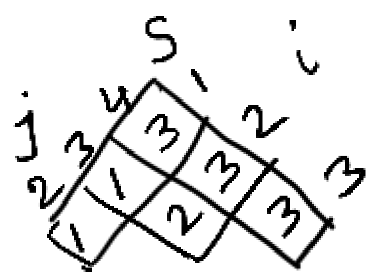
n^2 → Space

```
MATRIX-CHAIN-ORDER(p)
n = p.length - 1
let m[1..n, 1..n] and s[1..n-1, 2..n] be new tables
for i = 1 to n
    m[i,i] = 0
for l = 2 to n                  // l is the chain length
    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
```



P-O-P

```
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 ")"
```



$A_1 \; A_2 \; A_3 \; A_4$ (

P-O-P(S,1,3) — P-O-P(S,1,1)

P-O-P(S,2,3) — P-O-P(S,2,2)

P-O-P(S,3,3)

→ P-O-P(S,4,4)

Steps of DP w.r. to MCM problem:

1. **Structure of an optimal solution:**

$$A_i A_{i+1} \cdots A_j \quad , i \le j$$

To parenthesize this product, we must split it between A_k and A_{k+1} for some k in the range $i \le k < j$. Then, we first compute the matrices $A_i \cdots A_k$ and $A_{k+1} \cdots A_j$ and then multiply them together to produce $A_i \cdots A_j$. The cost of parenthesizing is the cost of computing the matrix product $A_i \cdots A_k$ plus the cost of computing $A_{k+1} \cdots A_j$ plus the cost of multiplying them together.

2. **Recursively define the value of an optimal solution**

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

3. **Computing the value of an optimal solution in bottom up fashion:**

Compute m & s tables in bottom up fashion using Matrix_Chain_Order procedure.

4. **Constructing an optimal solution (Optional step):**

Parenthesization step using Print_Optimal_Parens procedure.