Matrix chain multiplication.

I/P : Matrices $A_{p\times q}$ & $B_{q\times r}$.

O/P : Matrix $C_{p\times r}$

1. for $i \leftarrow 1$ to p.
2. for $j \leftarrow 1$ to r.
3. $C[i,j] \leftarrow 0$
4. for $K \leftarrow 1$ to q.
5. $C[i,j] \leftarrow C[i,j] + A[i,K] \cdot B[K,j]$
6. return C.

No. of scalar multiplication $= pqr$.

Matrix chain multiplication is an Optimization problem. Given a sequence of matrices, we want to find most efficient way to multiply these matrices together. Formally, Given a chain $A_1, A_2, \ldots\ldots, A_n$ of n matrices, where $i = 1, 2, \ldots n$. Matrix A_i has dimensions $P_{i-1} \times P_i$. Parenthasize the product $A_1, A_2, \ldots, A_n$ such that total no. of scalar multiplication is minimized.

We have many options because matrix multiplication is associative. For eg, if we have 4 matrices A, B, C, D, then scalar multiplications will be

(A BC) D, (AB) (CD), A(BCD), A(BC) D, ...

Counting the number of parenthesization :-

Exhaustively checking all possible parenthesizations does not yield an efficient algorithm. $P(n)$ is the no. of parenthesization of a sequence of matrices. when $n=1$, then just one matrix, & there will be only one way to parenthesize the matrix. when $n \geq 2$, a fully parenthesized matrix product is the product of two fully parenthesized matrix subproducts and the split between the two subproducts may occur between k^{th} and $(k+1)^{th}$ matrices for $k = 1, 2, \ldots, n-1$

Thus we obtain the recurrence

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

The solution to a similar recurrence is the sequence of Catalan numbers.

$$C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!\, n!} \quad \text{for } n \geq 1$$

which grows as $\Omega(4^n / n^{3/2})$

We shall use the dynamic-programming method to determine how to optimally parenthesize a matrix chain. We shall follow four step sequence:

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.
4. Construct an optimal solution from computed information.

Step 1:- The structure of optimal parenthesization

Let us adopt the notation $A_{i \ldots j}$, where $i \leq j$ for the matrix that results from evaluating product $A_i A_{i+1} A_j$. If the problem is nontrivial ie $i < j$, then split $A_i A_{i+1} A_j$ into A_k and A_{k+1} for some integers k, $i \leq k < j$. Thus we first compute $A_{i \ldots k}$ and $A_{k+1 \ldots j}$ and then multiply them together to produce final result $A_{i \ldots j}$. Thus cost of optimal parenthasization = cost of computing $A_{i \ldots k}$ + cost of computing $A_{k+1 \ldots n}$ + cost of multiplying $A_{i \ldots k}$ and $A_{k+1 \ldots n}$.

step 2: A recursive Solution.

Let $m[i,j]$ be the minimum number of scalar multiplication needed to compute the matrix $A_{i \ldots j}$.

If $i=j$ (trivial case) the chain consists of just one matrix $A_{i \ldots j} = A_i$. So no scalar multiplications are needed. Then $m[i,j]=0$. For non trivial case, $i<j$, we take the advantage of optimal structure solution from step 1. Then

$m[i,j]$ = [The minimum cost of computing the subproducts $A_{i \ldots k}$ and $A_{k+1 \ldots j}$] + [The cost of multiplying these two matrices together].

Recall that each matrix $A_i = p_{i-1} \times p_i$, then computing the matrix product $A_{i \ldots k} A_{k+1 \ldots j}$ takes $p_{i-1} p_k p_j$ scalar multiplications. Then,

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

We also keep track of optimal split $s[i,j]=k$.

Step 3:- Constructing optimal solutions.

Here we implement a tabular, bottom up method for matrix chain order. This procedure assumes that matrix A has dimensions $p_{i-1} \times p_i$ for $i = 1, 2, \ldots, n$. Its input is a sequence $p = \langle p_0, p_1, \ldots, p_n \rangle$ where $p.length = n+1$. This procedure use an auxiliary table $m[1, \ldots, n, 1 \ldots n]$ for storing $m[i,j]$ costs and another auxiliary table $s[1, \ldots n, 1 \ldots n]$ that records which index of K achieved the optimal cost in computing $m[i,j]$.

MATRIX - CHAIN - ORDER(p)

```
n ← p.length - 1
Let m[1.....n, 1.....n] & s[1.....n, 1.....n] be two tables
for i = 1 to n
        m[i, j] = 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 = 1 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 & s.
```

A simple inspection of the nested loop structure of MATRIX-CHAIN-ORDER yields a running time $O(n^3)$

Step 4: Constructing an optimal solution.

PRINT-OPTIMAL-PARENS (s, i, j)

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

So $s[1,n] = (A_1 A_2 \cdots A_{s[1,n]})(A_{s[1,n]+1} \cdots A_n)$

$s[1, s[1,n]] = (A_1 \cdots A_{s[1, s[1,n]]})(A_{s[1,s[1,n]]+1} \cdots A_{s[1,n]})$

$s[s[1,n]+1, n] = (A_{s[1,n]+1} \cdots A_{s[s[1,n]+1,n]})$

$(A_{s[s[1,n]+1,n]+1} \cdots A_n)$

Let the matrix chain —

	A_1	A_2	A_3	A_4	A_5.
Dimension :	4×10	10×3	3×12	12×20	20×7
	p_0 p_1	p_1 p_2	p_2 p_3	p_3 p_4	p_4 p_5

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

$m[1,2] = 120 \longrightarrow k=1$

$m[2,3] = 360 \longrightarrow k=2$

$m[3,4] = 720 \longrightarrow k=3$

$m[4,5] = 1680 \longrightarrow k=4$

$m[1,3] = 264 \longrightarrow k=2$

$m[2,4] = 1320 \longrightarrow k=2$

$m[3,5] = 1140 \longrightarrow k=4$

$m[1,4] = 1080 \longrightarrow k=2$

$m[2,5] = 1350 \longrightarrow k=2$

$m[1,5] = 1344 \longrightarrow k=2$

i \ j	1	2	3	4	5
1	0	120	264	1080	1344
2	X	0	360	1320	1350
3	X	X	0	720	1140
4	X	X	X	0	1680
5	X	X	X	X	0

m table and s table

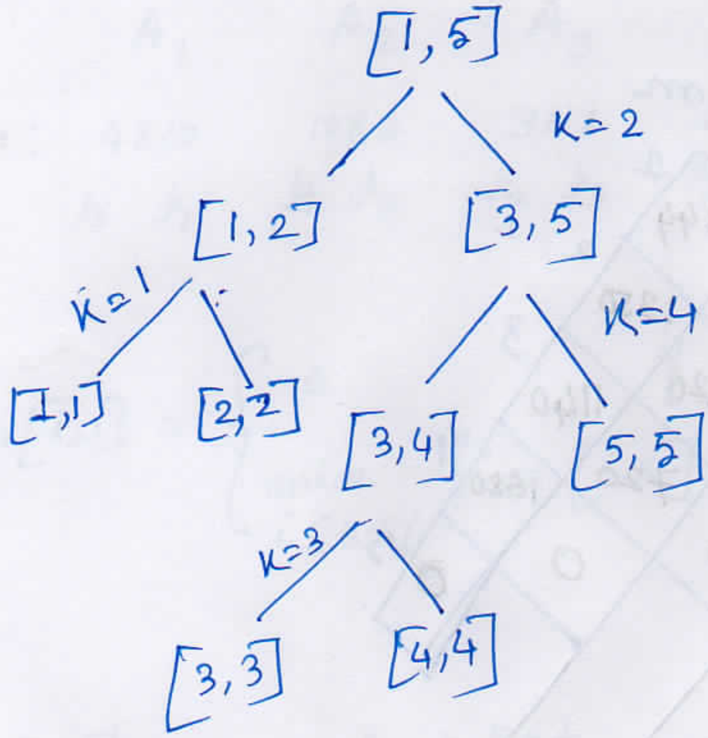
m

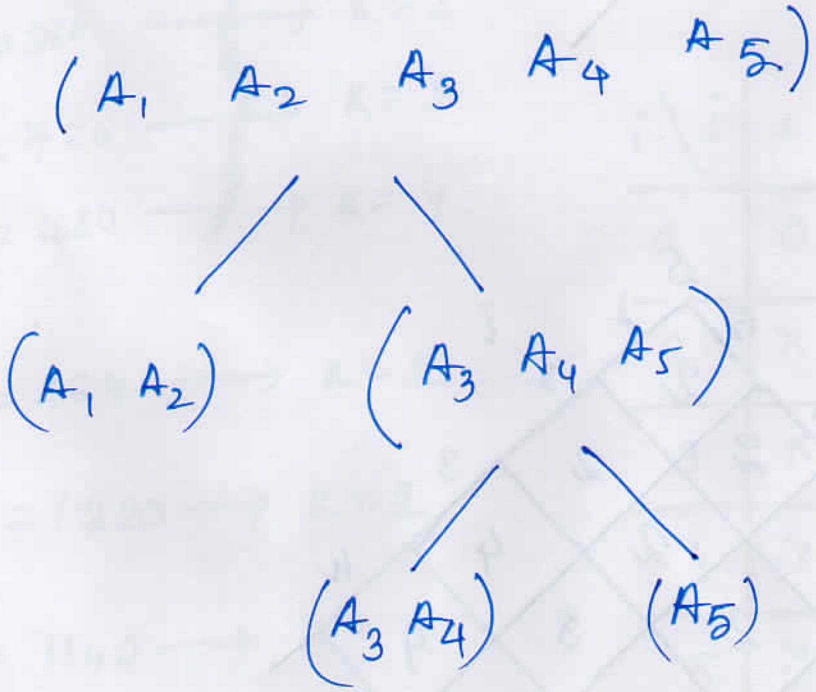
i \ j	1	2	3	4	5
1	0	120	264	1080	1344
2		0	360	1320	1350
3			0	720	1140
4				0	1680
5					0

s

i \ j	2	3	4	5
1	1	2	2	2
2		2	2	2
3			3	4
4				4

optimal parenthesization





$\therefore (A_1 A_2)((A_3 A_4) A_5).$