**Matrix Chain Multiplication**

It is a Method under Dynamic Programming in which previous output is taken as input for next.

Here, Chain means one matrix's column is equal to the second matrix's row [always].

In general:

If A = ⌊aij⌋ is a p x q matrix

   B = ⌊bij⌋ is a q x r matrix

   C = ⌊cij⌋ is a p x r matrix

Then

Matrix Chain Multiplication

Given following matrices {A1,A2,A3,...An} and we have to perform the matrix multiplication, which can be accomplished by a series of matrix multiplications

A1 x A2 x, A3 x.....x An

Matrix Multiplication operation is **associative** in nature rather commutative. By this, we mean that we have to follow the above matrix order for multiplication but we are free to **parenthesize** the above multiplication depending upon our need.

In general, for 1≤ i≤ p and 1≤ j ≤ r

Matrix Chain Multiplication

It can be observed that the total entries in matrix 'C' is 'pr' as the matrix is of dimension p x r Also each entry takes O (q) times to compute, thus the total time to compute all possible entries for the matrix 'C' which is a multiplication of 'A' and 'B' is proportional to the product of the dimension p q r.

It is also noticed that we can save the number of operations by reordering the parenthesis.

**Example1:** Let us have 3 matrices, A1,A2,A3 of order (10 x 100), (100 x 5) and (5 x 50) respectively.

Three Matrices can be multiplied in two ways:

1. **A1,(A2,A3):** First multiplying(A2 and A3) then multiplying and resultant withA1.
2. **(A1,A2),A3:** First multiplying(A1 and A2) then multiplying and resultant withA3.

No of Scalar multiplication in Case 1 will be:

1. (100 x 5 x 50) + (10 x 100 x 50) = 25000 + 50000 = 75000

No of Scalar multiplication in Case 2 will be:

1. (100 x 10 x 5) + (10 x 5 x 50) = 5000 + 2500 = 7500

To find the best possible way to calculate the product, we could simply parenthesis the expression in every possible fashion and count each time how many scalar multiplication are required.

Matrix Chain Multiplication Problem can be stated as "find the optimal parenthesization of a chain of matrices to be multiplied such that the number of scalar multiplication is minimized".

**Number of ways for parenthesizing the matrices:**

There are very large numbers of ways of parenthesizing these matrices. If there are n items, there are (n-1) ways in which the outer most pair of parenthesis can place.

(A1) (A2,A3,A4,................An)

Or (A1,A2) (A3,A4 .................An)

Or (A1,A2,A3) (A4 ...............An)

........................

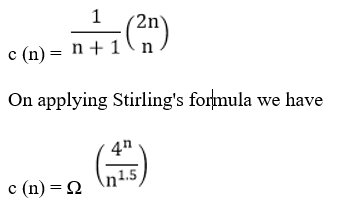
Or(A1,A2,A3.............An-1) (An)

It can be observed that after splitting the kth matrices, we are left with two parenthesized sequence of matrices: one consist 'k' matrices and another consist 'n-k' matrices.

Now there are 'L' ways of parenthesizing the left sublist and 'R' ways of parenthesizing the right sublist then the Total will be L.R:

DAA Matrix Chain Multiplication

Also p (n) = c (n-1) where c (n) is the nth **Catalon number**



Which shows that 4n grows faster, as it is an exponential function, then n1.5.

**Development of Dynamic Programming Algorithm**

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

**Dynamic Programming Approach**

Let Ai,j be the result of multiplying matrices i through j. It can be seen that the dimension of Ai,j is pi-1 x pj matrix.

Dynamic Programming solution involves breaking up the problems into subproblems whose solution can be combined to solve the global problem.

At the greatest level of parenthesization, we multiply two matrices

A1.....n=A1....k x Ak+1....n)

Thus we are left with two questions:

* How to split the sequence of matrices?
* How to parenthesize the subsequence A1.....k andAk+1......n?

One possible answer to the first question for finding the best value of 'k' is to check all possible choices of 'k' and consider the best among them. But that it can be observed that checking all possibilities will lead to an exponential number of total possibilities. It can also be noticed that there exists only O (n2 ) different sequence of matrices, in this way do not reach the exponential growth.

**Step1: Structure of an optimal parenthesization:** Our first step in the dynamic paradigm is to find the optimal substructure and then use it to construct an optimal solution to the problem from an optimal solution to subproblems.

Let Ai....j where i≤ j denotes the matrix that results from evaluating the product

Ai Ai+1....Aj.

If i < j then any parenthesization of the product Ai Ai+1 ......Aj must split that the product between Ak and Ak+1 for some integer k in the range i ≤ k ≤ j. That is for some value of k, we first compute the matrices Ai.....k & Ak+1....j and then multiply them together to produce the final product Ai....j. The cost of computing Ai....k plus the cost of computing Ak+1....j plus the cost of multiplying them together is the cost of parenthesization.

**Step 2: A Recursive Solution:** Let m [i, j] be the minimum number of scalar multiplication needed to compute the matrixAi....j.

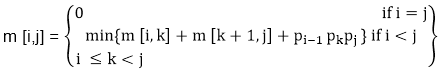
If i=j the chain consist of just one matrix Ai....i=Ai so no scalar multiplication are necessary to compute the product. Thus m [i, j] = 0 for i= 1, 2, 3....n.

If i<j we assume that to optimally parenthesize the product we split it between Ak and Ak+1 where i≤ k ≤j. Then m [i,j] equals the minimum cost for computing the subproducts Ai....k and Ak+1....j+ cost of multiplying them together. We know Ai has dimension pi-1 x pi, so computing the product Ai....k and Ak+1....jtakes pi-1 pk pj scalar multiplication, we obtain

m [i,j] = m [i, k] + m [k + 1, j] + pi-1 pk pj

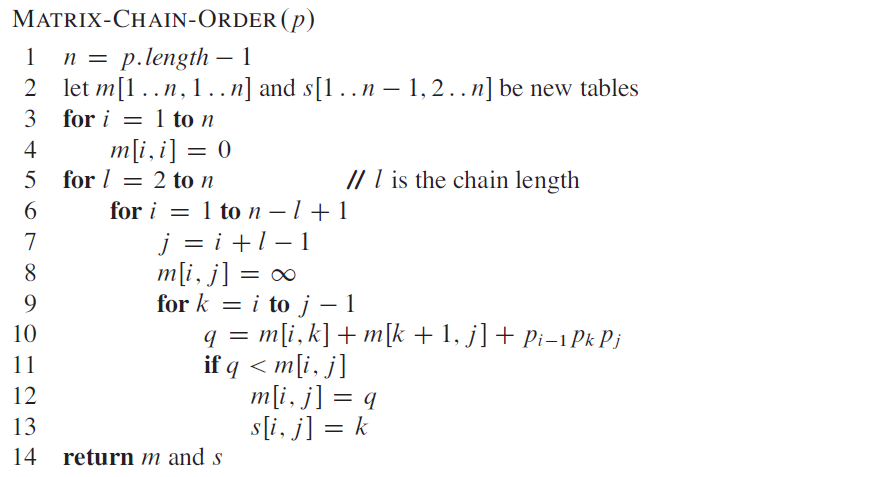
There are only (j-1) possible values for 'k' namely k = i, i+1.....j-1. Since the optimal parenthesization must use one of these values for 'k' we need only check them all to find the best.

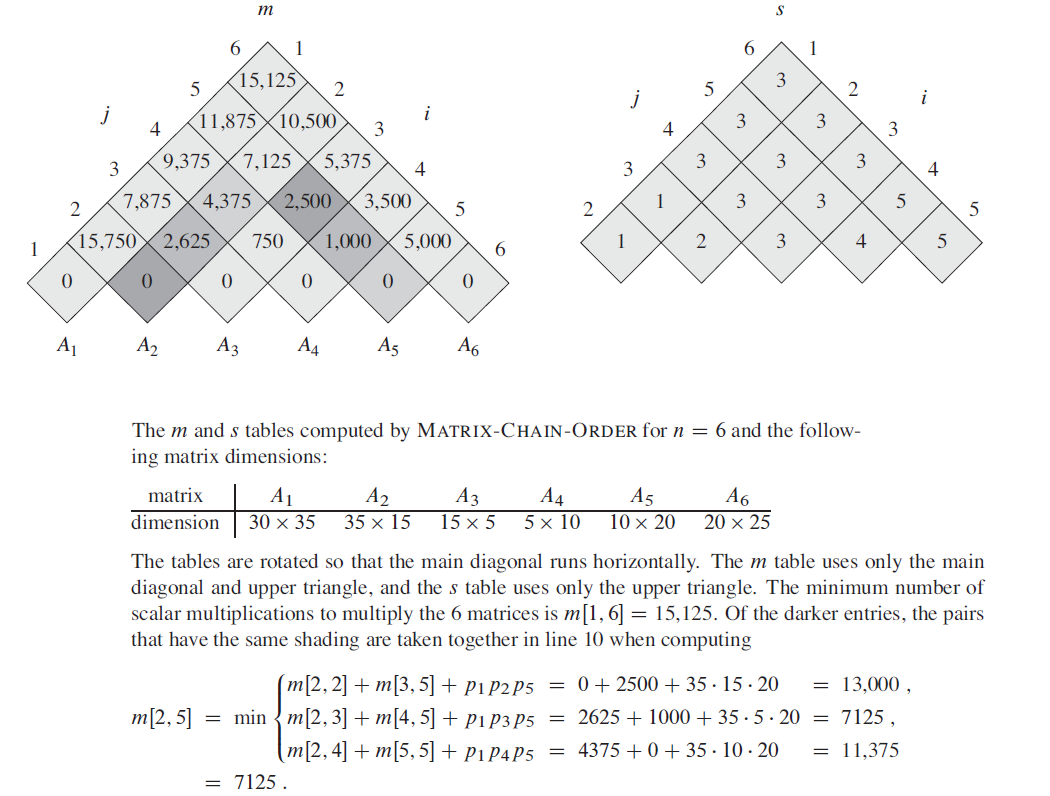
So the minimum cost of parenthesizing the product Ai Ai+1......Aj becomes

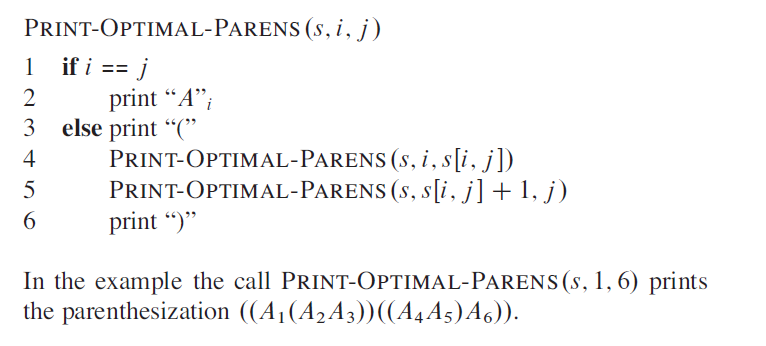


To construct an optimal solution, let us define s [i,j] to be the value of 'k' at which we can split the product Ai Ai+1 .....Aj To obtain an optimal parenthesization i.e. s [i, j] = k such that

m [i,j] = m [i, k] + m [k + 1, j] + pi-1 pk pj

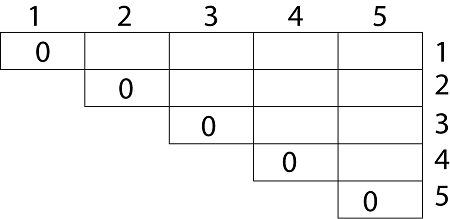




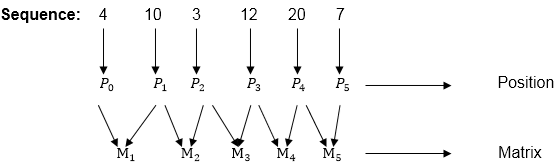


# Example of Matrix Chain Multiplication

**Example:** We are given the sequence {4, 10, 3, 12, 20, and 7}. The matrices have size 4 x 10, 10 x 3, 3 x 12, 12 x 20, 20 x 7. We need to compute M [i,j], 0 ≤ i, j≤ 5. We know M [i, i] = 0 for all i.



Let us proceed with working away from the diagonal. We compute the optimal solution for the product of 2 matrices.



Here P0 to P5 are Position and M1 to M5 are matrix of size (pi to pi-1)

On the basis of sequence, we make a formula

Example of Matrix Chain Multiplication

In Dynamic Programming, initialization of every method done by '0'.So we initialize it by '0'.It will sort out diagonally.

We have to sort out all the combination but the minimum output combination is taken into consideration.

**Calculation of Product of 2 matrices:**

1. m (1,2) = m1 x m2

= 4 x 10 x 10 x 3

= 4 x 10 x 3 = 120

2. m (2, 3) = m2 x m3

= 10 x 3 x 3 x 12

= 10 x 3 x 12 = 360

3. m (3, 4) = m3 x m4

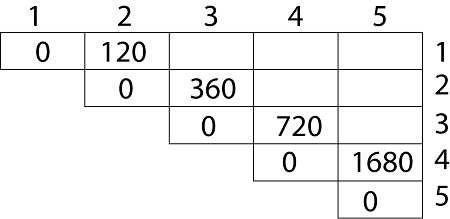
= 3 x 12 x 12 x 20

= 3 x 12 x 20 = 720

4. m (4,5) = m4 x m5

= 12 x 20 x 20 x 7

= 12 x 20 x 7 = 1680



* We initialize the diagonal element with equal i,j value with '0'.
* After that second diagonal is sorted out and we get all the values corresponded to it

Now the third diagonal will be solved out in the same way.

**Now product of 3 matrices:**

M [1, 3] = M1 M2 M3

1. There are two cases by which we can solve this multiplication: ( M1 x M2) + M3, M1+ (M2x M3)
2. After solving both cases we choose the case in which minimum output is there.

Example of Matrix Chain Multiplication

**M [1, 3] =264**

As Comparing both output **264** is minimum in both cases so we insert **264** in table and ( M1 x M2) + M3 this combination is chosen for the output making.

M [2, 4] = M2 M3 M4

1. There are two cases by which we can solve this multiplication: (M2x M3)+M4, M2+(M3 x M4)
2. After solving both cases we choose the case in which minimum output is there.

DAA Example of Matrix Chain Multiplication

**M [2, 4] = 1320**

As Comparing both output **1320** is minimum in both cases so we insert **1320** in table and M2+(M3 x M4) this combination is chosen for the output making.

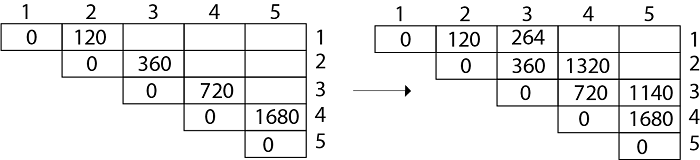
M [3, 5] = M3 M4 M5

1. There are two cases by which we can solve this multiplication: ( M3 x M4) + M5, M3+ ( M4xM5)
2. After solving both cases we choose the case in which minimum output is there.

Example of Matrix Chain Multiplication

M [3, 5] = 1140

As Comparing both output **1140** is minimum in both cases so we insert **1140** in table and ( M3 x M4) + M5this combination is chosen for the output making.



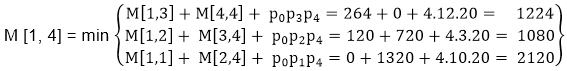
Now Product of 4 matrices:

M [1, 4] = M1 M2 M3 M4

There are three cases by which we can solve this multiplication:

1. ( M1 x M2 x M3) M4
2. M1 x(M2 x M3 x M4)
3. (M1 xM2) x ( M3 x M4)

After solving these cases we choose the case in which minimum output is there



**M [1, 4] =1080**

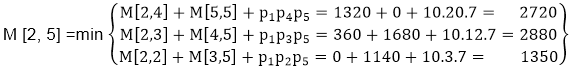
As comparing the output of different cases then '**1080**' is minimum output, so we insert 1080 in the table and (M1 xM2) x (M3 x M4) combination is taken out in output making,

M [2, 5] = M2 M3 M4 M5

There are three cases by which we can solve this multiplication:

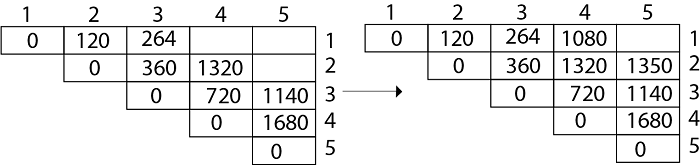
1. (M2 x M3 x M4)x M5
2. M2 x( M3 x M4 x M5)
3. (M2 x M3)x ( M4 x M5)

After solving these cases we choose the case in which minimum output is there



M [2, 5] = 1350

As comparing the output of different cases then '**1350**' is minimum output, so we insert 1350 in the table and M2 x( M3 x M4 xM5)combination is taken out in output making.



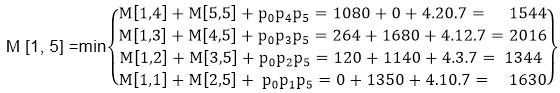
**Now Product of 5 matrices:**

M [1, 5] = M1 M2 M3 M4 M5

There are five cases by which we can solve this multiplication:

1. (M1 x M2 xM3 x M4 )x M5
2. M1 x( M2 xM3 x M4 xM5)
3. (M1 x M2 xM3)x M4 xM5
4. M1 x M2x(M3 x M4 xM5)

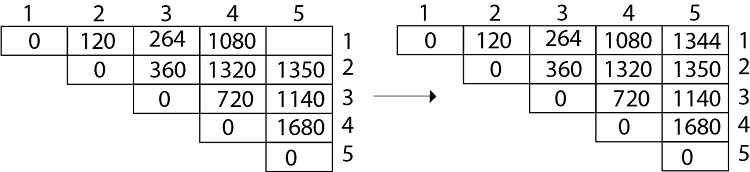
After solving these cases we choose the case in which minimum output is there



M [1, 5] = 1344

As comparing the output of different cases then '**1344**' is minimum output, so we insert 1344 in the table and M1 x M2 x(M3 x M4 x M5)combination is taken out in output making.

**Final Output is:**



**Step 3: Computing Optimal Costs:** let us assume that matrix Ai has dimension pi-1x pi for i=1, 2, 3....n. The input is a sequence (p0,p1,......pn) where length [p] = n+1. The procedure uses an auxiliary table m [1....n, 1.....n] for storing m [i, j] costs an auxiliary table s [1.....n, 1.....n] that record which index of k achieved the optimal costs in computing m [i, j].

The algorithm first computes m [i, j] ← 0 for i=1, 2, 3.....n, the minimum costs for the chain of length 1.

**RELEVANT READING MATERIAL AND REFERENCES:**

**Source Notes:**

1. <https://www.javatpoint.com/matrix-chain-multiplication-example>

**Lecture Video:**

1. <https://youtu.be/prx1psByp7U>
2. <https://www.youtube.com/watch?v=_WncuhSJZyA>

**Online Notes:**

1. <http://vssut.ac.in/lecture_notes/lecture1428551222.pdf>

**Text Book Reading:**

1. Cormen, Leiserson, Rivest, Stein, “*Introduction to Algorithms*”, Prentice Hall of India, 3rd edition 2012. problem, Graph coloring.

**In addition: PPT can be also be given.**