Analysis of Algorithms

[5CS4-05/5IT4-05]

Unit 2. Dynamic Programming

Ramakant Soni

Assistant Professor, Computer Science Department,

B K Birla Institute of Engineering & Technology, Pilani

Rajasthan

Dynamic Programming Approach

- Dynamic Programming (DP) is an algorithmic technique for **solving an optimization problem by breaking it down into simpler subproblems** and the optimal solution to the overall problem depends upon the optimal solution to its subproblems.

- Algorithm finds solutions to sub problems and stores them in memory for later use.

- Characteristics of Dynamic Programming:
    1. Overlapping Subproblems
    2. Optimal Substructure Property

Dynamic Programming Approach

1. Overlapping Subproblems:

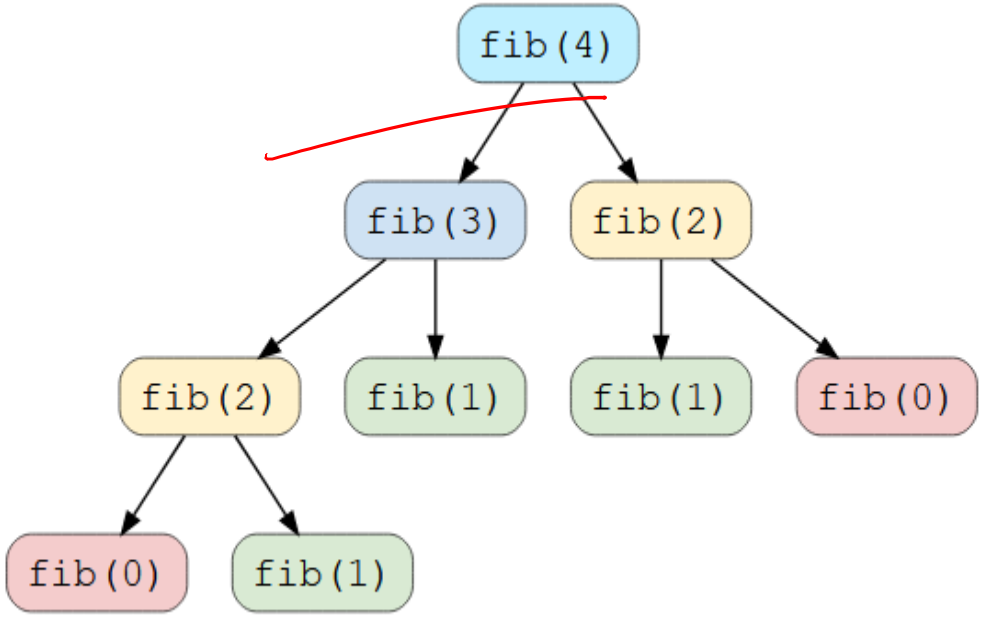
Subproblems are smaller versions of the original problem. Any problem has overlapping sub-problems if finding its solution involves solving the same subproblem multiple times.

2. Optimal Substructure Property:

Any problem has optimal substructure property if its overall optimal solution can be constructed from the optimal solutions of its subproblems.

Example: Fibonacci Series:

Fib(n) = Fib(n-1) + Fib(n-2), for n > 1



Recursion tree for calculating Fibonacci numbers

Matrix Chain Multiplication

- Engineering applications often have to multiply a long chain of matrices.
- Matrix multiplication is associative; and we can multiply only two matrices at a time.
- So, we can multiply this chain of matrices(M_1, M_2, …M_n) in many different ways, for example:

  - $((M_1 \times M_2) \times M_3) \times \ldots M_n$
  - $M_1 \times (((M_2 \times M_3) \times \ldots) \times M_n)$
  - $(M_1 \times M_2) \times (M_3 \times \ldots M_n)$ and so on.

There are numerous ways to multiply this chain of matrices.
They will all produce the same final result, however they will take more or less time to compute, based on which particular matrices are multiplied.

- If matrix A has dimensions m×n and matrix B has dimensions n×q, then matrix C=A×B will have dimensions m×q, and will require m x n x q scalar multiplications.

A × B = C

m×n n×q m×q

Scalar products = m×n×q

Matrix Chain Multiplication

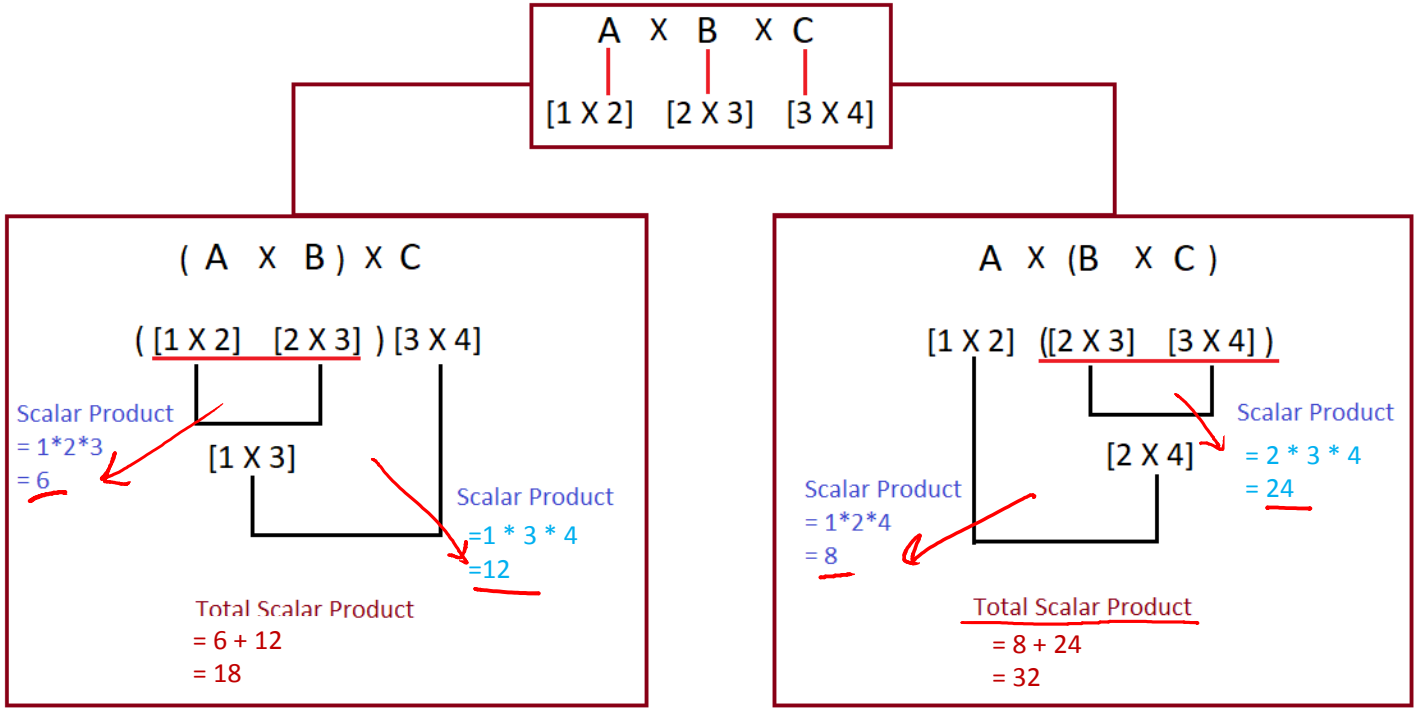
For example, let us multiply matrices A, B and C with dimensions 1 X 2, 2 X 3, and 3 X 4, respectively.

A B C

Matrix A×B×C will be of size 1 X 4 and can be calculated in two ways shown below:

1. (AxB)×C This order of matrix multiplication will require 1*2*3 + 1*3*4 = 6 + 12 = **18** scalar multiplications.

2. A×(B×C) This order of matrix multiplication will require 2*3*4 + 1*2*4 = 24 + 8 = **32** scalar multiplications.



for given example.

The First way is faster, and we should multiply the matrices using that arrangement of parenthesis.

Therefore, conclusion is that **the order of parenthesis matters, and our task is to find the optimal order of parenthesis**.

imp

Matrix Chain Multiplication

very dynamic programming approach.

The dynamic programming solution is presented below:

Let's call **m[i, j]** the minimum number of scalar multiplications needed to multiply a chain of matrices from matrix i to matrix j **(i.e. $M_i \times .... \times M_j$, i.e. i<=j).**

We split the chain at some matrix k, such that i <= k < j, and try to find out which combination produces minimum m[i, j].

$M_i \times ... \times M_k \times ... \times M_j$

The formula is:

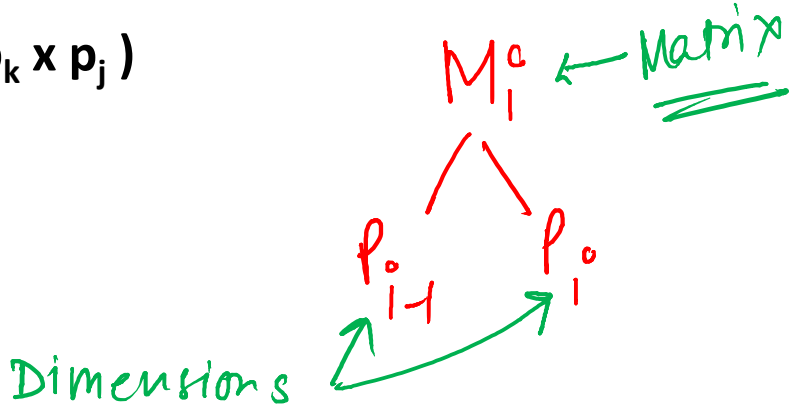
- **if** i = j, **m[i, j]**= 0
- **if** i < j, **m[i, j]=** min over all possible values of k **(m[i, k] + m[k+1, j] + $p_{i-1} \times p_k \times p_j$)**

Scalar product for multiplying matrix from M_i to M_k

$p_{i-1} \times p_k$ $\quad$ $p_k \times p_j$

where *k* ranges from *i* to *j* – 1.

- m[i, k] is the scalar multiplication for multiplying Matrices $M_i \times ... \times M_k$
- m[k+1, j] is the scalar multiplication for multiplying Matrices $M_{k+1} \times ... \times M_j$
- p_{i-1} is the row dimension of matrix i,
- p_k is the column dimension of matrix k,
- p_j is the column dimension of matrix j.



Matrix Chain Multiplication pseudocode for solving sub structures

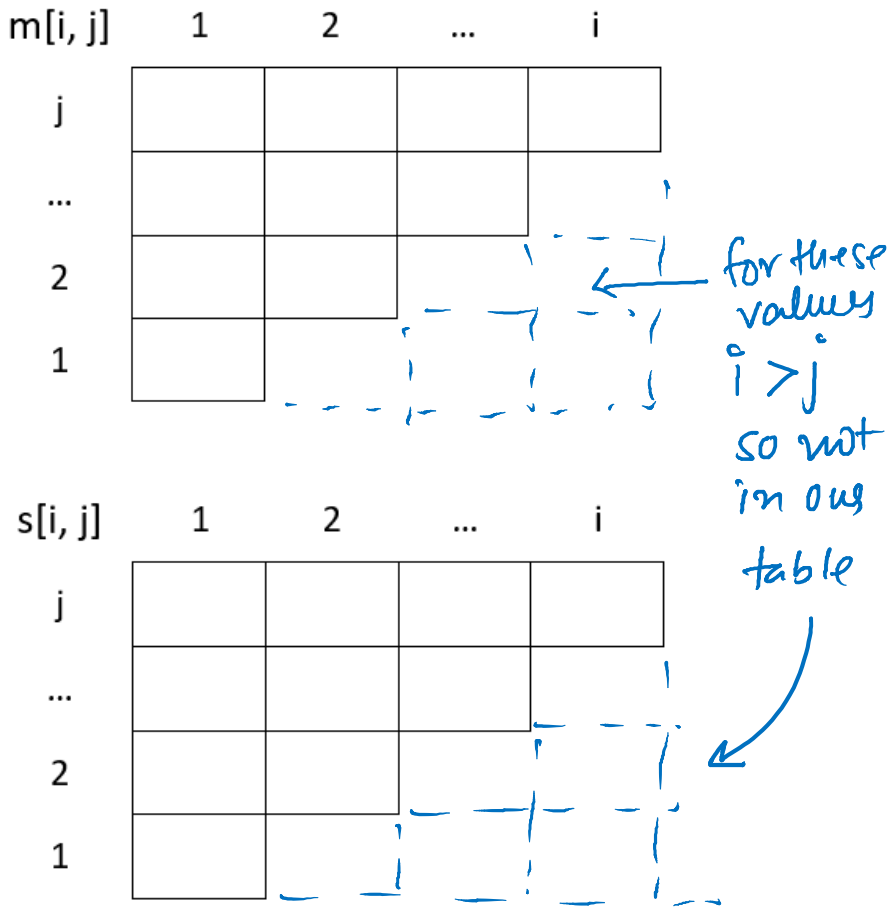
Input parameter "chain" is the chain of matrices, i.e. $M_1, M_2, \ldots M_n$:

```
function OptimalMatrixChainParenthesis(chain)
    n = length(chain)
    for i = 1, n
        m[i,i] = 0    // Since it takes no calculations to multiply one matrix
    for len = 2, n
        for i = 1, n - len + 1
            j = i + len -1
            m[i,j] = infinity      // So that the first calculation updates
            for k = i, j-1
                q = m[i, k] + m[k+1, j] + p_{i-1} * p_k * p_j
                if q < m[i, j]     // The new order of parentheses is better than what we had
                    m[i, j] = q    // Update
                    s[i, j] = k    // Record which k to split on, i.e. where to place the parenthesis
```

Matrix Chain Multiplication table genrataion

So , we have calculated values for all possible *m*[*i*, *j*], the minimum number of calculations to multiply a chain from matrix *i* to matrix *j*, and we have recorded the corresponding "split point“ *s*[*i*, *j*].

This algorithm will produce "tables" *m*[,] and *s*[,] that will have entries for all possible values of i and j. The final solution for the entire chain is m[1, n], with corresponding split at s[1, n].



Matrix Chain Multiplication print optimal parenthesis pseudocode

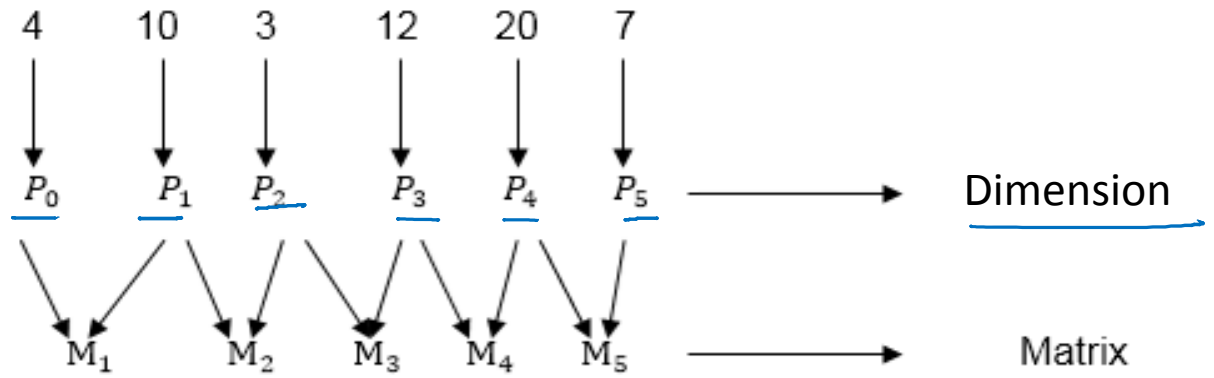
```
function PrintOptimalParenthesis (s, i, j)
    if i = j
      print " M" i
    else
      print " )"
      PrintOptimalParenthesis (s, i, s[i, j])
      PrintOptimalParenthesis (s, s[i, j] + 1, j)
      print ")"
```

← Algorithm to print the optimal parenthesis.

Matrix Chain Multiplication Example: Step 1- Single Matrix Product

Example: We are given the dimensions {4, 10, 3, 12, 20, and 7}.
Here, using 6 dimensions 5 matrices can be formed.

Sequence: 4 10 3 12 20 7



Here P_0 to P_5 are Dimension and M_1 to M_5 are matrix of size (p_i to p_{i-1})

So, the 5 matrices will 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

for cases of $i=j$ no value of k exist because $i \le k < j$, which is not possible if $i=j$.

m[i, j]

j \ i	1	2	3	4	5
5					0
4				0	
3			0		
2		0			
1	0				

s[i, j]

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

Matrix Chain Multiplication Example: Step 2- Two Matrix Product

Using

if i < j, **m[i, j]=** $\min_{i<=k<j}$ **(m[i, k] + m[k+1, j] + p_{i-1} x p_k x p_j)**

1. **m (1,2)** =for k=1 (m[1, 1] + m[2, 2] + p_0 x p_1 x p_2)
   = 0 + 0 + 4 x 10 x 3 =120
   s (1,2) = 1

2. **m (2,3)** =for k=2 (m[2, 2] + m[3, 3] + p_1 x p_2 x p_3)
   = 0 + 0 + 10 x 3 x 12 = 360
   s (2,3) = 2

3. **m (3,4)** =for k=3 (m[3, 3] + m[4, 4] + p_2 x p_3 x p_4)
   = 0 + 0 + 3 x 12 x 20 = 720
   s (3,4) = 3

4. **m (4,5)** =for k=4 (m[4, 4] + m[5, 5] + p_3 x p_4 x p_5)
   = 0 + 0 + 12 x 20 x 7 = 1680
   s (4,5) = 4

m[i, j]

j \ i	1	2	3	4	5
5				1680	0
4			720	0	
3		360	0		
2	120	0			
1	0				

s[i, j]

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

Matrix Chain Multiplication Example: Step 3- Three Matrix Product

m (1,3) = M1 x M2 X M3

We can solve this multiplication in 2 ways: M_1+ (M_2x M_3) and (M_1 x M_2) + M_3

$$= min \begin{cases} \text{k=1 (m[1, 1] + m[2, 3] + } p_0 \text{ x } p_1 \text{ x } p_3) = 0 + 360 + 4 * 10 * 12 = 840 \\ \text{k=2 (m[1, 2] + m[3, 3] + } p_0 \text{ x } p_2 \text{ x } p_3 \text{)} = 120 + 0 + 4 * 3 * 12 = 264 \end{cases}$$

m (1,3) =264, **s (1,3)** =2

m (2,4) = M2 x M3 X M4

We can solve this multiplication in 2 ways : M_2+(M_3 x M_4) and (M_2x M_3)+M_4

$$= min \begin{cases} \text{k=2 (m[2, 2] + m[3, 4] + } p_1 \text{ x } p_2 \text{ x } p_4) = 0 + 720 + 10 * 3 * 20 = 1320 \\ \text{k=3 (m[2, 3] + m[4, 4] + } p_1 \text{ x } p_3 \text{ x } p_4) = 360 + 0 + 10 * 12 * 20 = 2760 \end{cases}$$

m (2,4) =1320, **s (2,4)** =2

m (3,5) = M3 x M4 X M5

We can solve this multiplication in 2 ways : M_3+ (M_4xM_5) and (M_3 x M_4) + M_5

$$= min \begin{cases} \text{k=3 (m[3, 3] + m[4, 5] + } p_2 \text{ x } p_3 \text{ x } p_5) = 0 + 1680 + 3 * 12 * 7 = 1932 \\ \text{k=4 (m[3, 4] + m[5, 5] + } p_2 \text{ x } p_4 \text{ x } p_5 \text{)} = 720 + 0 + 3 * 20 * 7 = 1140 \end{cases}$$

m (3,5) =1140, **s (3,5)** =4

m[i, j]

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

s[i, j]

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

Matrix Chain Multiplication Example: Step 4- Four Matrix Product

m (1,4) = M1 x M2 X M3 X M4

We can solve this multiplication in 3 ways: M_1 x(M_2 x M_3 x M_4) , (M_1 xM_2) x (M_3 x M_4) and (M_1 x M_2 x M_3) M_4

$$= min \begin{cases} k=1\ (\ m[1,1] + m[2,4] + p_0 \times p_1 \times p_4) = 0 + 1320 + 4 * 10 * 20 = 2120 \\ k=2\ (\ m[1,2] + m[3,4] + p_0 \times p_2 \times p_4) = 120 + 720 + 4 * 3 * 20 = 1080 \\ k=3\ (\ m[1,3] + m[4,4] + p_0 \times p_3 \times p_4) = 264 + 0 + 4 * 12 * 20 = 1224 \end{cases}$$

m (1,4) =1080, **s (1,4)** =2

m (2,5) = M2 x M3 X M4 X M5

We can solve this multiplication in 3 ways: M_2 x(M_3 x M_4 x M_5) , (M_2 x M_3)x (M_4 x M_5) and (M_2 x M_3 x M_4)x M_5

$$= min \begin{cases} k=2\ (\ m[2,2] + m[3,5] + p_1 \times p_2 \times p_5) = 0 + 1140 + 10 * 3 * 7 = 1350 \\ k=3\ (\ m[2,3] + m[4,5] + p_1 \times p_3 \times p_5) = 360 + 1680 + 10 * 12 * 7 = 2880 \\ k=4\ (\ m[2,4] + m[5,5] + p_1 \times p_4 \times p_5) = 1320 + 0 + 10 * 20 * 7 = 2720 \end{cases}$$

m (2,5) =1320, **s (2,5)** =2

m[i, j]

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

s[i, j]

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

Matrix Chain Multiplication Example: Step 5- Five Matrix Product

m (1,5) = M1 x M2 X M3 X M4 X M_5

We can solve this multiplication in 4 ways: M_1 x(M_2 x M_3 x M_4 x M_5) , (M_1 xM_2) x (M_3 x M_4 x M_5) and (M_1 x M_2 x M_3) x (M_4 x M_5) and (M_1 x M_2 x M_3 x M_4) x M_5

$$= min \begin{cases} \text{k=1 (m[1, 1] + m[2, 5] + } p_0 \times p_1 \times p_5) = 0 + 1350 + 4 * 10 * 7 = 1630 \\ \text{k=2 (m[1, 2] + m[3, 5] + } p_0 \times p_2 \times p_5) = 120 + 1140 + 4 * 3 * 7 = 1344 \\ \text{k=3 (m[1, 3] + m[4, 5] + } p_0 \times p_3 \times p_5) = 264 + 1680 + 4 * 12 * 7 = 2016 \\ \text{k=3 (m[1, 4] + m[4, 5] + } p_0 \times p_4 \times p_5) = 264 + 0 + 4 * 20 * 7 = 1544 \end{cases}$$

m (1,5) =1344, **s (1,5)** =2

Total Minimum scalar products

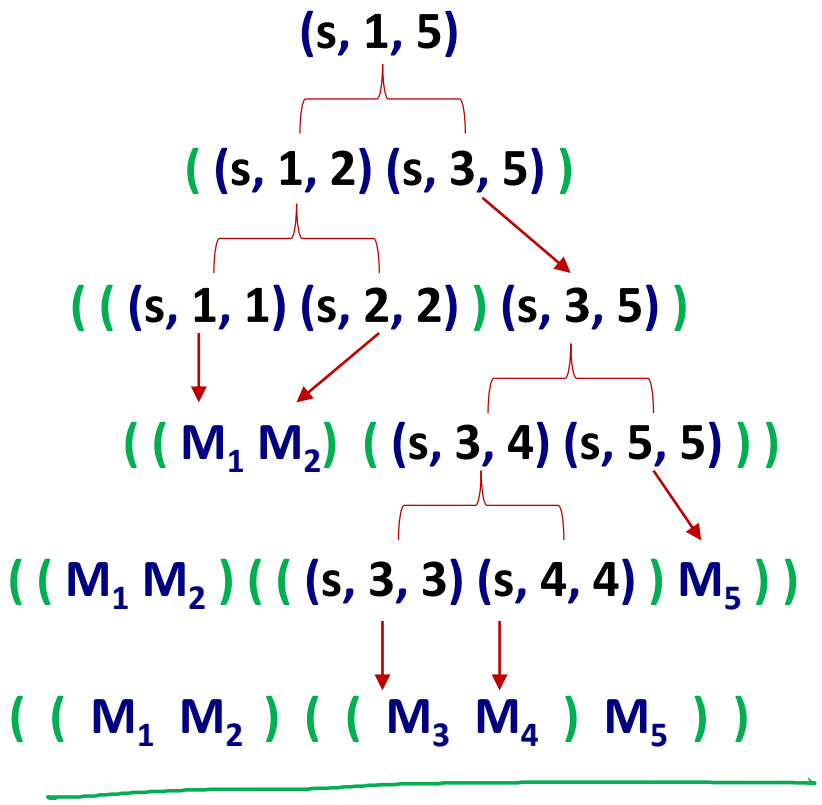
m[i, j]

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

s[i, j]

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

Matrix Chain Multiplication Example: Step 6- Print optimal parenthesis



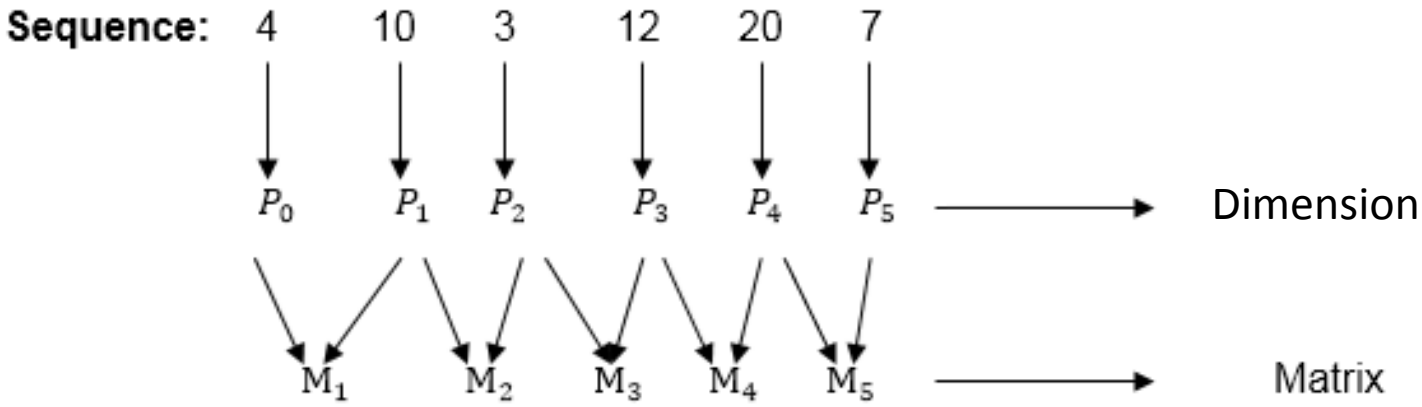
	s[i, j]	i = 1	2	3	4	5
j	5	2	2	4	4	x
	4	2	2	3	X	
	3	2	2	X		
	2	1	X			
	1	X				

```
function PrintOptimalParenthesis (s, i, j)
    if i = j
      print " M" i
    else
      print " ("
      PrintOptimalParenthesis (s, i, s[i, j])
      PrintOptimalParenthesis (s, s[i, j] + 1, j)
      print ")"
```

Verify result

← To check if we have done calculations right. ← Must to be done

((M_1 M_2) ((M_3 M_4) M_5))



Step 1: $M_3 * M_4 = M_{34}$

Scalar products=3 *12*20 =720

Dimension of M_{34} = 3 X 20

((M_1 M_2) (M_{34} M_5))

Step 2: $M_{34} * M_5 = M_{35}$

Scalar products = 3 *20*7 = 420

Dimension of M_{35} = 3 X 7

((M_1 M_2) M_{345})

Step 3: $M_1 * M_2 = M_{12}$

Scalar products = 4 *10*3 = 120

Dimension of M_{12} = 4 X 3

(M_{12} M_{345})

Step 4: $M_{12} * M_{35} = M_{15}$

Scalar products = 4 *3*7 = 84

Correct

Total Scalar Product = 720+420+120+84 = **1344**

Queries ?