Matrix Chain Multiplication

CSE 300: Technical Writing and Presentation

1805002 - A. H. M. Osama Haque
1805009 - Md. Zarzees Uddin

August 30, 2022

Contents

Description

Aim

- To find the most efficient way to multiply a sequence of matrices

Aim

- **To find the most efficient way to multiply a sequence of matrices**

A : 5x20 B: 20x10 C: 10x50

Aim

- **To find the most efficient way to multiply a sequence of matrices**

A : 5x20 B: 20x10 C: 10x50

Compute ABC

Aim

To find the most efficient way to multiply a sequence of matrices

A : 5x20 B: 20x10 C: 10x50

Compute ABC

- (AB)C

Aim

To find the most efficient way to multiply a sequence of matrices

A : 5x20 B: 20x10 C: 10x50

Compute ABC

- (AB)C
- A(BC)

Aim

To find the most efficient way to multiply a sequence of matrices

A : 5x20 B: 20x10 C: 10x50

Compute ABC

- (AB)C 5*20*10 + 5*10*50
- A(BC) 20*10*50 + 5*20*50

Aim

To find the most efficient way to multiply a sequence of matrices

A : 5x20 B: 20x10 C: 10x50

Compute ABC

- (AB)C 5*20*10 + 5*10*50 = 3500 operations
- A(BC) 20*10*50 + 5*20*50 = 15000 operations

Aim

To find the most efficient way to multiply a sequence of matrices

A : 5x20 B: 20x10 C: 10x50

Compute ABC

- (AB)C 5*20*10 + 5*10*50 = 3500 operations
- A(BC) 20*10*50 + 5*20*50 = 15000 operations

Brute Force Algorithm

Brute Force Approach :

Brute Force Approach :

- Consider all possible cases.

Brute Force Approach :

- Consider all possible cases.
- Not a good idea since the number of operations increases exponentially with the number of matrices to multiply.

Brute Force Approach :

- Consider all possible cases.
- Not a good idea since the number of operations increases exponentially with the number of matrices to multiply.
- EXPONENTIAL-TIME Algorithm
  Very slow and not practical

Effective Solution?

Dynamic Programming Algorithm

Dynamic Programming Approach

To compute $A_1 A_2 ... A_n$
n : # matrices to multiply

Dynamic Programming Approach

To compute $A_1 A_2 ... A_n$
n : # matrices to multiply

$A_i \rightarrow (d_{i-1} \times d_i)$ matrix
$A_i A_{i+1} ... A_j \rightarrow (d_{i-1} \times d_j)$ matrix
m(i,j) : Min number of operations required to compute product of A_i to A_j

Algorithm

- **Step 1:** $m(i, j) = \times$, $\forall_{i>j}$

Algorithm

- **Step 1:** $m(i, j) = \times$, $\forall_{i>j}$
- **Step 2:** $\forall_{1 \le i \le j \le n}$

$$m(i,j) = \begin{cases} 0 & \text{if } i = j \\ \min_{i \le k < j} \{ m(i,k) + m(k+1, j) + d_{i-1} d_k d_j \} & \text{if } i \neq j \end{cases}$$

Algorithm

- **Step 1:** $m(i, j) = \times$, $\forall_{i>j}$
- **Step 2:** $\forall_{1 \le i \le j \le n}$

$$m(i,j) = \begin{cases} 0 & \text{if } i = j \\ \min_{i \le k < j} \{ m(i,k) + m(k+1, j) + d_{i-1} d_k d_j \} & \text{if } i \neq j \end{cases}$$

- **Step 3:** Return $m(1, n)$

Example

Let us consider an example,
A $\rightarrow$ 40$\times$20
$B\rightarrow$ 20$\times$30
$C\rightarrow$ 30$\times$10
$D\rightarrow$ 10$\times$30

We create a 4×4 table for implementing the algorithm

Example

A	×	B	×	C	×	D
40×20		20×30		30×10		10×30

i/j	0	1	2	3
0				
1	x			
2	x	x		
3	x	x	x	

Example

A × B × C × D

40×20 20×30 30×10 10×30

Step 1 : Fill the table for i = j (All zero values)

i/j	0	1	2	3
0	0			
1	x	0		
2	x	x	0	
3	x	x	x	0

Example

A × B × C × D

40×20 20×30 30×10 10×30

i/j	0	1	2	3
0	0	24000		
1	x	0	6000	
2	x	x	0	9000
3	x	x	x	0

Step 1 : Fill the table for i = j (All zero values)

Step 2 :

m(0,1) = $40 \times 20 \times 30 = 24000$

m(1,2) = $20 \times 30 \times 10 = 6000$

m(2,3) = $30 \times 10 \times 30 = 9000$

Example

A × B × C × D

40×20 20×30 30×10 10×30

i/j	0	1	2	3
0	0	24000		
1	x	0	6000	
2	x	x	0	9000
3	x	x	x	0

Step 1 : Fill the table for i = j (All zero values)

Step 2 :

m(0,1) = $40 \times 20 \times 30 = 24000$

m(1,2) = $20 \times 30 \times 10 = 6000$

m(2,3) = $30 \times 10 \times 30 = 9000$

Example

A × B × C × D

40×20 20×30 30×10 10×30

i/j	0	1	2	3
0	0	24000		
1	x	0	6000	
2	x	x	0	9000
3	x	x	x	0

Step 1 : Fill the table for i = j (All zero values)

Step 2 :

m(0,1) = $40 \times 20 \times 30 = 24000$

m(1,2) = $20 \times 30 \times 10 = 6000$

m(2,3) = $30 \times 10 \times 30 = 9000$

Example

A	×	B	×	C	×	D
40×20		20×30		30×10		10×30

i/j	0	1	2	3
0	0	24000	14000	
1	x	0	6000	
2	x	x	0	9000
3	x	x	x	0

Step 1 : Fill the table for i = j (All zero values)

Step 2 :

m(0,1) = $40 \times 20 \times 30 = 24000$

m(1,2) = $20 \times 30 \times 10 = 6000$

m(2,3) = $30 \times 10 \times 30 = 9000$

m(0,2) = min(14000, 36000) = 14000

Example

A	×	B	×	C	×	D
40×20		20×30		30×10		10×30

i/j	0	1	2	3
0	0	24000	14000	
1	x	0	6000	
2	x	x	0	9000
3	x	x	x	0

Step 1 : Fill the table for i = j (All zero values)

Step 2 :

m(0,1) = $40 \times 20 \times 30 = 24000$

m(1,2) = $20 \times 30 \times 10 = 6000$

m(2,3) = $30 \times 10 \times 30 = 9000$

m(0,2) = min(14000, 36000) = 14000

Example

A	×	B	×	C	×	D
40×20		20×30		30×10		10×30

Step 1 : Fill the table for i = j (All zero values)

Step 2 :

m(0,1) = $40 \times 20 \times 30 = 24000$

m(1,2) =$20 \times 30 \times 10 = 6000$

m(2,3) =$30 \times 10 \times 30 = 9000$

m(0,2) = min(14000, 36000) = 14000

i/j	0	1	2	3
0	0	24000	14000	
1	x	0	6000	
2	x	x	0	9000
3	x	x	x	0

Example

A × B × C × D

40×20 20×30 30×10 10×30

i/j	0	1	2	3
0	0	24000	14000	
1	x	0	6000	12000
2	x	x	0	9000
3	x	x	x	0

Step 1 : Fill the table for i = j (All zero values)

Step 2 :

m(0,1) = $40 \times 20 \times 30 = 24000$

m(1,2) = $20 \times 30 \times 10 = 6000$

m(2,3) = $30 \times 10 \times 30 = 9000$

m(0,2) = min(14000, 36000) = 14000

m(1,3) = 12000

Example

A	×	B	×	C	×	D
40×20		20×30		30×10		10×30

i/j	0	1	2	3
0	0	24000	14000	
1	x	0	6000	12000
2	x	x	0	9000
3	x	x	x	0

Step 1 : Fill the table for i = j (All zero values)

Step 2 :

m(0,1) = $40\times20\times30 = 24000$

m(1,2) =$20\times30\times10 = 6000$

m(2,3) =$30\times10\times30 = 9000$

m(0,2) = min(14000, 36000) = 14000

m(1,3) = 12000

Example

A $\times$ B $\times$ C $\times$ D

40×20 20×30 30×10 10×30

i/j	0	1	2	3
0	0	24000	14000	
1	x	0	6000	12000
2	x	x	0	9000
3	x	x	x	0

Step 1 : Fill the table for i = j (All zero values)

Step 2 :

m(0,1) = $40\times20\times30 = 24000$

m(1,2) =$20\times30\times10 = 6000$

m(2,3) =$30\times10\times30 = 9000$

m(0,2) = min(14000, 36000) = 14000

m(1,3) = 12000

Example

A	×	B	×	C	×	D
40×20		20×30		30×10		10×30

i/j	0	1	2	3
0	0	24000	14000	
1	x	0	6000	12000
2	x	x	0	9000
3	x	x	x	0

Step 1 : Fill the table for i = j (All zero values)

Step 2 :

m(0,1) = $40\times20\times30 = 24000$

m(1,2) =$20\times30\times10 = 6000$

m(2,3) =$30\times10\times30 = 9000$

m(0,2) = min(14000, 36000) = 14000

m(1,3) = 12000

Example

A × B × C × D

40×20 20×30 30×10 10×30

i/j	0	1	2	3
0	0	24000	14000	26000
1	x	0	6000	12000
2	x	x	0	9000
3	x	x	x	0

Step 1 : Fill the table for i = j (All zero values)

Step 2 :

m(0,1) = $40 \times 20 \times 30 = 24000$

m(1,2) = $20 \times 30 \times 10 = 6000$

m(2,3) = $30 \times 10 \times 30 = 9000$

m(0,2) = min(14000, 36000) = 14000

m(1,3) = 12000

m(0,3) = min(26000, 38000) = 26000

Example

A × B × C × D

40×20 20×30 30×10 10×30

i/j	0	1	2	3
0	0	24000	14000	26000
1	x	0	6000	12000
2	x	x	0	9000
3	x	x	x	0

Step 1 : Fill the table for i = j (All zero values)

Step 2 :

m(0,1) = $40 \times 20 \times 30 = 24000$

m(1,2) = $20 \times 30 \times 10 = 6000$

m(2,3) = $30 \times 10 \times 30 = 9000$

m(0,2) = min(14000, 36000) = 14000

m(1,3) = 12000

m(0,3) = min(26000, 38000) = 26000

Step 3 : return 26000

Summery

Time Complexity : O(n^3)

Summery

Time Complexity : O(n^3)

Seems most optimal?

Summery

Time Complexity : O(n^3)

Seems most optimal?

No!!!

Summery

Time Complexity : $O(n^3)$

Seems most optimal?

No!!!

SIAM J. COMPUT.
Vol. 11, No. 2, May 1982

0097-5397/82/1102-0013 $01.00/0

COMPUTATION OF MATRIX CHAIN PRODUCTS. PART I*

T. C. HU† AND M. T. SHING†

Abstract. This paper considers the computation of matrix chain products of the form $M_1 \times M_2 \times \cdots \times M_{n-1}$. If the matrices are of different dimensions, the order in which the product is computed affects the number of operations. An optimum order is an order which minimizes the total number of operations. We present some theorems about an optimum order of computing the matrices. Based on these theorems, an $O(n \log n)$ algorithm for finding an optimum order will be presented in Part II.

Key words. matrix multiplication, polygon partition, dynamic programming

1. Introduction. Consider the evaluation of the product of $n-1$ matrices

$$M = M_1 \times M_2 \times \cdots \times M_{n-1}, \tag{1}$$

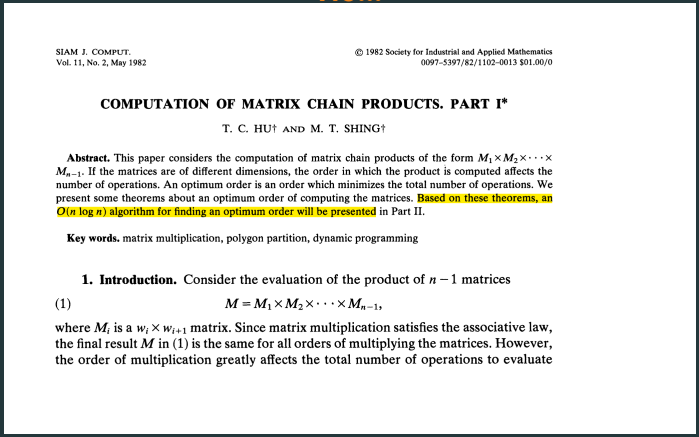
where M_i is a $w_i \times w_{i+1}$ matrix. Since matrix multiplication satisfies the associative law, the final result M in (1) is the same for all orders of multiplying the matrices. However, the order of multiplication greatly affects the total number of operations to evaluate

Summery

Time Complexity : O(n^3)

Seems most optimal?

No!!!



SIAM J. COMPUT.
Vol. 11, No. 2, May 1982

0097-5397/82/1102-0013 $01.00/0

COMPUTATION OF MATRIX CHAIN PRODUCTS. PART I*

T. C. HU† AND M. T. SHING†

Abstract. This paper considers the computation of matrix chain products of the form $M_1 \times M_2 \times \cdots \times M_{n-1}$. If the matrices are of different dimensions, the order in which the product is computed affects the number of operations. An optimum order is an order which minimizes the total number of operations. We present some theorems about an optimum order of computing the matrices. Based on these theorems, an $O(n \log n)$ algorithm for finding an optimum order will be presented in Part II.

Key words. matrix multiplication, polygon partition, dynamic programming

1. Introduction. Consider the evaluation of the product of $n-1$ matrices

$$M = M_1 \times M_2 \times \cdots \times M_{n-1}, \tag{1}$$

where M_i is a $w_i \times w_{i+1}$ matrix. Since matrix multiplication satisfies the associative law, the final result M in (1) is the same for all orders of multiplying the matrices. However, the order of multiplication greatly affects the total number of operations to evaluate

An algorithm published by Hu and Shing (1982) achieves $O(nlogn)$ computational complexity

They reduced the problem into Triangulation of a Regular Polygon.

References

- Wikipedia/Matrix Chain Multiplication
- Introduction to Algorithms by Cormen
- stackoverflow.com