Matrix-chain Multiplication

Chapter 15.2

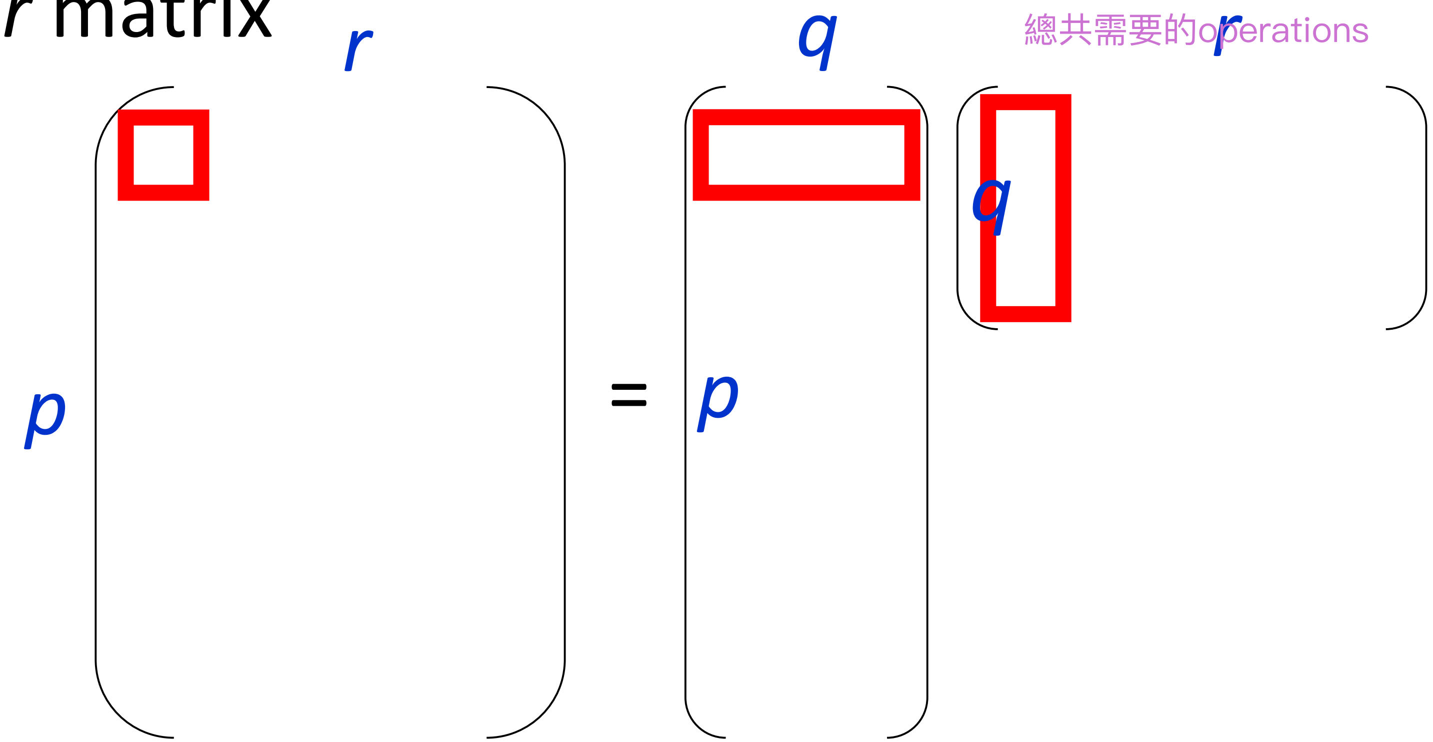
Mei-Chen Yeh

Matrix Multiplication

- ***A***: a p x q matrix
- ***B***: a q x r matrix
- ***C*** = ***AB***

How many operations do we need to get ***C*** from ***A***, ***B***? O(pqr)

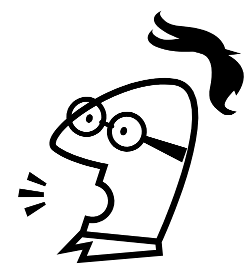
總共需要的operations



$$C[i,j] = \sum_{1 \le k \le q} A[i,k] \cdot B[k,j]$$

Matrix Multiplication

- Yes or No?
- ***A B C*** = (***A B***) ***C*** = ***A*** (***B C***) 答案會是一樣的
- Do they have the same number of operations? No!
- Example: 總共需要的operations：p*q*r 相乘
  - ***A***: $\overset{p}{10}$ x $\overset{q}{100}$, ***B***: $\overset{q}{100}$ x $\overset{r}{5}$, ***C***: 5 x 50
  - (***A B***) ***C*** => $\overset{p}{10}$ x $\overset{q}{100}$ x $\overset{r}{5}$ + 10 x 5 x 50 = 7,500
  - ***A*** (***B C***) => 100 x 5 x 50 + 10 x 100 x 50 = 75,000



Matrix-chain Multiplication Problem

- Given a chain $\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_n$ of n matrices, parenthesize the product $\boldsymbol{A}_1\boldsymbol{A}_2\ldots\boldsymbol{A}_n$ in a way that minimizes the number of operations.

Matrix-chain Multiplication Problem (Revised)

- Given a sequence of positive integers $p_0, p_1, \ldots, p_n$, where
  - p_{i-1} is the number of rows of $\boldsymbol{A}_i$
  - p_i is the number of columns of $\boldsymbol{A}_i$

我們是要看總operations的量，不用知道矩陣裡的內容，我們在乎的是矩陣的大小，所以給的input會是一串整數

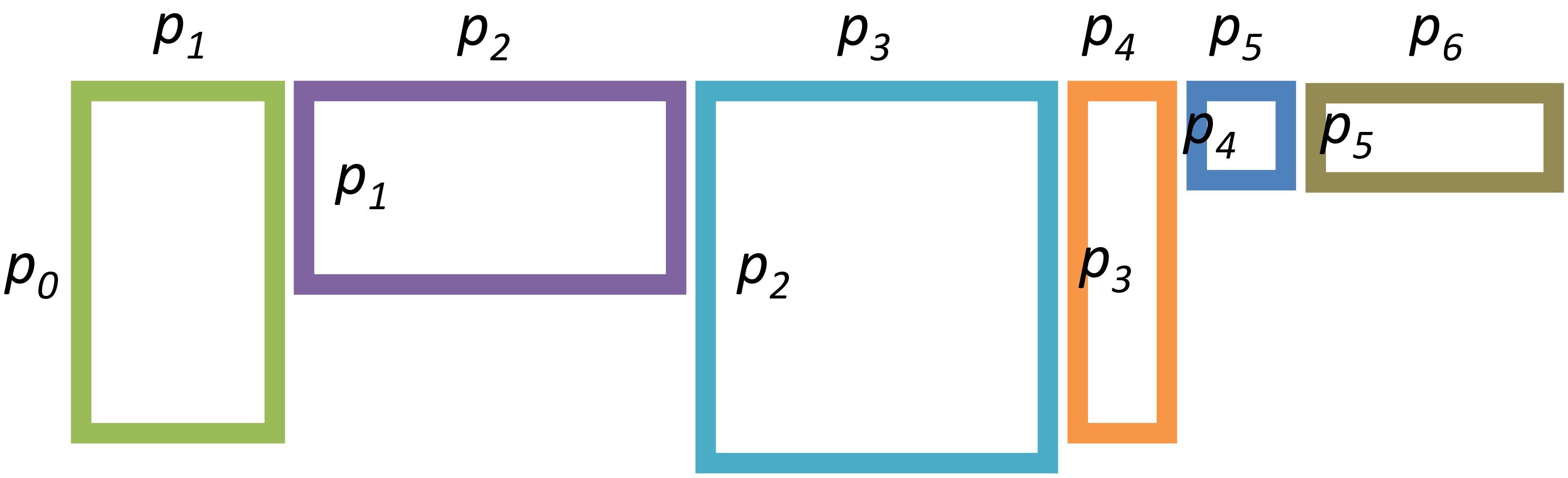
- Parenthesize the product $\boldsymbol{A}_1\boldsymbol{A}_2\ldots\boldsymbol{A}_n$ in a way that minimizes the number of operations

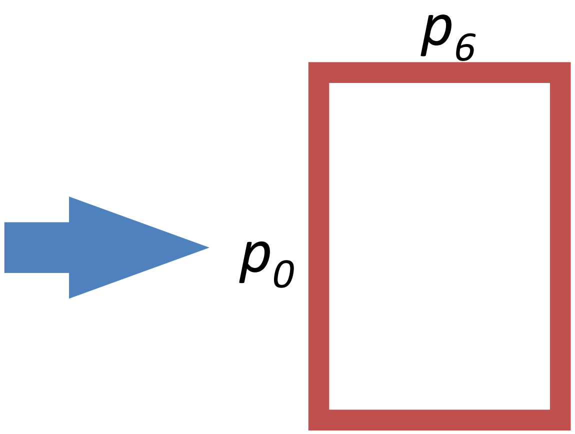
Example: $\boldsymbol{A}_1$: 10 x 100, $\boldsymbol{A}_2$: 100 x 5, $\boldsymbol{A}_3$: 5 x 50

→ Given 10, 100, 5, 50

p_0 p_1 p_2 p_3 ← input

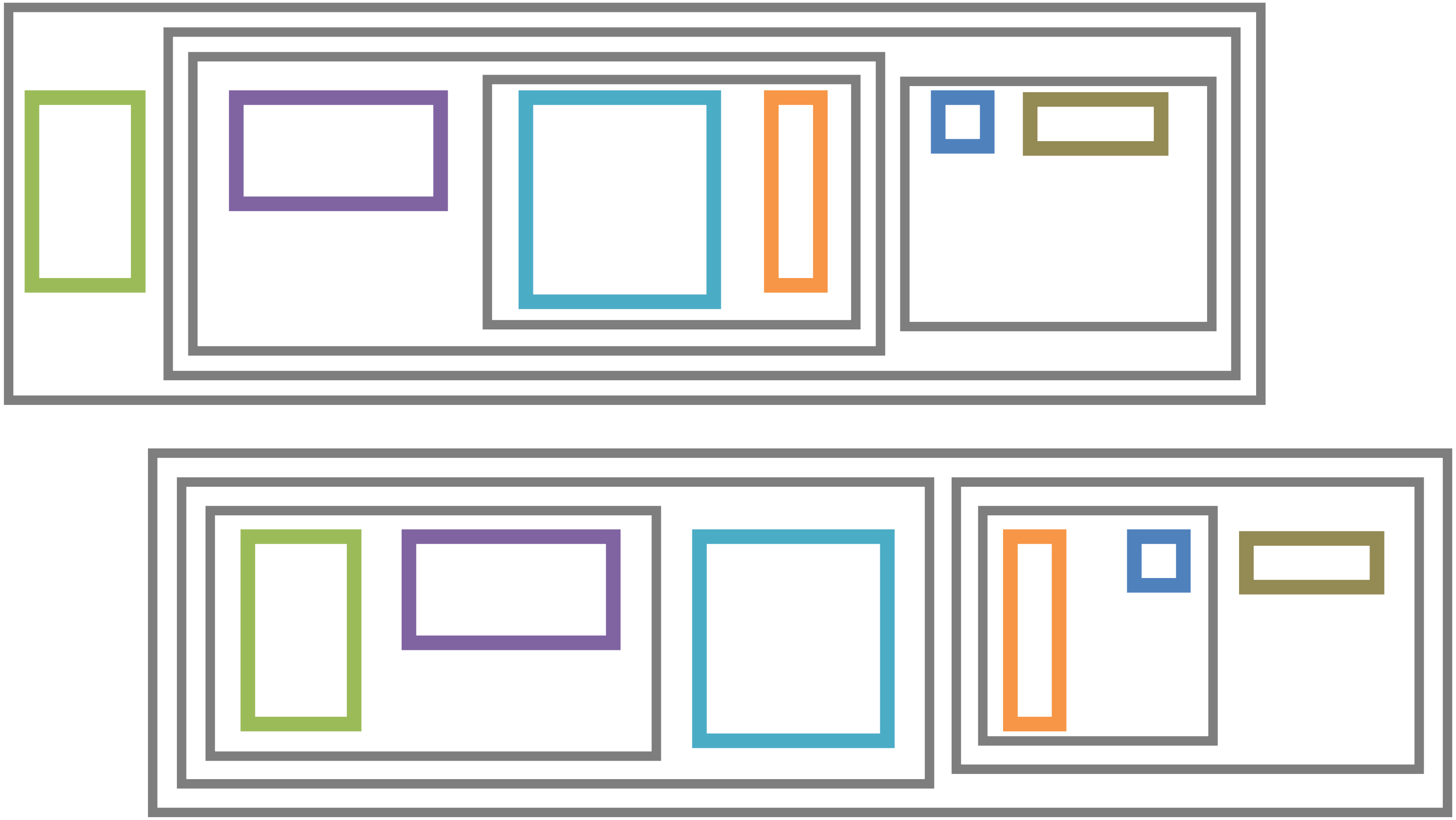
3個矩陣會轉換成4個整數，分別代表矩陣的邊長





不管怎麼乘，最後矩陣的大小會是P0*P6

只是其中一種做法，做法有很多種



Brute Force Approach

- How many possible parenthesizations? $P(n) = ?$
- Example: $\boldsymbol{A_1 A_2 A_3 A_4}$
  - $(\boldsymbol{A_1}(\boldsymbol{A_2}(\boldsymbol{A_3 A_4})))$
  - $(\boldsymbol{A_1}((\boldsymbol{A_2 A_3})\boldsymbol{A_4}))$
  - $((\boldsymbol{A_1 A_2})(\boldsymbol{A_3 A_4}))$
  - $((\boldsymbol{A_1}(\boldsymbol{A_2 A_3}))\boldsymbol{A_4})$
  - $(((\boldsymbol{A_1 A_2})\boldsymbol{A_3})\boldsymbol{A_4})$

按照順序乘，4個矩陣會有5種方式

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

$\boxed{\textbf{Exponential in } \boldsymbol{n}}$

「operations的次數」跟「矩陣的個數」之間，
會是指數成長的關係，會是常數的n-次方

Dynamic Programming (1)

Dynamic Programming 就是在做表格

- $m[i, j]$ 建立一個表個叫m，把從第 i 個矩陣乘到第 j 個矩陣需要幾個 operations？把它存在 m 裡
  - the minimum number of operations required to get the product $\boldsymbol{A}_i \boldsymbol{A}_{i+1} \ldots \boldsymbol{A}_j$ 把矩陣 i 跟矩陣 j 之間存在的所有矩陣乘起來

Assume the best cut $(\boldsymbol{A}_i \ldots \boldsymbol{A}_k)\ (\boldsymbol{A}_{k+1} \ldots \boldsymbol{A}_j)$

在 i 跟 j 之間，找到一個矩陣 k，把它分成兩個大矩陣，再相乘

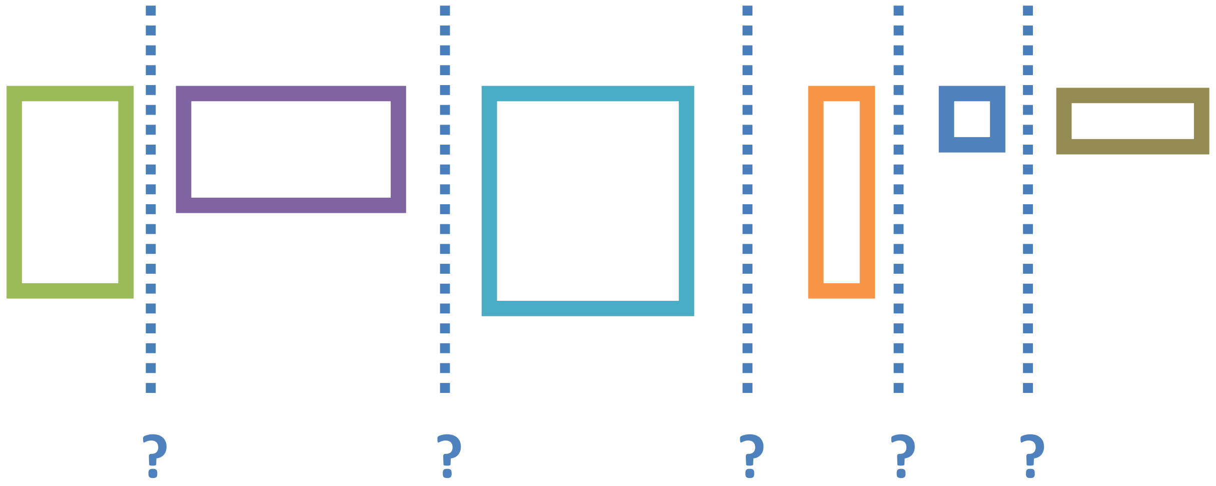
$$m[i, j] = m[i, k] + m[k+1, j] + p_{i-1}p_kp_j$$

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

不知道 k 切在哪裡最好，所以每個都切切看（會需要 for loop 來做）

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

切開成兩半邊後，各自會成為一個大矩陣，把兩個大矩陣乘起來，還是會需要大小pqr的運算



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

Where is the solution?

Initialization of the table?

	1	2	3	4	5	6	…	*n*
1								
2								
3								
4								
5								
6								
:								
n								

i

j

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

代表從 第一個矩陣乘 到 第一個矩陣（自己乘自己），只有自己一個矩陣所以沒辦法乘，故為 0

How to fill the table?

	1	2	3	4	5	6	…	*n*
1	0							$\boxed{}$
2		0						
3			0					
4				0				
5					0			
6						0		
:							0	
n								0

i ↓ j →

答案會在這，這格要最後填

右上角這格會是最小的

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

從第一個乘到第三個

從第二個乘到第四個

⋮

所以這個斜排的運算量都是把三個矩陣乘在一起

	1	2	3	4	5	6	…	n
1	0							$\boxed{}$
2		0						
3			0					
4				0				
5					0			
6						0		
:							0	
n								0

i ↓ , j →

chain len. = n

⋮

chain len. = 4

chain len. = 3

chain len. = 2

斜著填

$$m[\underset{3}{i}, \underset{6}{j}] = \begin{cases} 0 & \text{if } i \geq j \\ \min\limits_{i \leq k < j}\{m[i, k] + m[k+1, j] + p_{i-1}p_kp_j\} & \text{if } i < j \end{cases}$$

k: 3, 4, 5

(*k*=3)

m[3, 3]

m[4, 6]

(*k*=4)

m[3, 4]

m[5, 6]

(*k*=5)

m[3, 5]

m[6, 6]

	1	2	3	4	5	6	…	*n*
1	0							
2		0						
3			3	4	5			
4				0		3		
5					0	4		
6						5		
⋮							0	
n								0

要計算粉紅色這格，會用到我們已經算好的那六格

Dynamic Programming (cont.)

- $s[i, j]$
  - the index k that achieves the optimal cost in computing $m[i, j]$

Example

matrix	**A_1**	**A_2**	**A_3**	**A_4**	**A_5**	**A_6**
dim.	30x35	35x15	15x5	5x10	10x20	20x25

要計算上面的測資，最少會需要 15125 的運算量

m	1	2	3	4	5	6
1	0	15750	7875	9375	11875	15125
2		0	2625	4375	7125	10500
3			0	750	2500	5375
4				0	1000	3500
5					0	5000
6						0

s	1	2	3	4	5	6
1	0	1	1	3	3	3
2		0	2	3	3	3
3			0	3	3	3
4				0	4	5
5					0	5
6						0

要切在第五個矩陣

The best parenthesization? $(A_1(A_2A_3))((A_4A_5)A_6)$

```
MATRIX-CHAIN-ORDER(p)
n = p.length - 1
for i = 1 to n
    m[i, i] = 0
for l = 2 to n   // chain length
    for i = 1 to n – l + 1
        j = i + l – 1
        m[i, j] = ∞
        for k = i to j-1  // find min
            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
```

Time: $O(n^3)$

Space: $O(n^2)$

n^2

n^3

n