

动态规划篇：矩阵链乘法问题

童咏昕

北京航空航天大学
计算机学院

中国大学MOOC北航《算法设计与分析》

- 矩阵

- $p \times q$ 的矩阵 $U_{p,q}$

- 例

- 4×3 的矩阵 $U_{4,3}$

$$U = \begin{bmatrix} 2 & 1 & 3 \\ 9 & 5 & 6 \\ 0 & 8 & 1 \\ 5 & 2 & 7 \end{bmatrix}$$

- 3×2 的矩阵 $V_{3,2}$

$$V = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

- 矩阵

- $p \times q$ 的矩阵 $U_{p,q}$

- 例

- 4×3 的矩阵 $U_{4,3}$

$$U = \begin{bmatrix} 2 & 1 & 3 \\ 9 & 5 & 6 \\ 0 & 8 & 1 \\ 5 & 2 & 7 \end{bmatrix} \left. \vphantom{\begin{bmatrix} 2 & 1 & 3 \\ 9 & 5 & 6 \\ 0 & 8 & 1 \\ 5 & 2 & 7 \end{bmatrix}} \right\} p = 4$$

$\underbrace{\hspace{10em}}_{q = 3}$

- 3×2 的矩阵 $V_{3,2}$

$$V = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} \left. \vphantom{\begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}} \right\} q = 3$$

$\underbrace{\hspace{10em}}_{r = 2}$

- 矩阵

- $p \times q$ 的矩阵 $U_{p,q}$

- 例

- 4×3 的矩阵 $U_{4,3}$

$$U = \begin{bmatrix} 2 & 1 & 3 \\ 9 & 5 & 6 \\ 0 & 8 & 1 \\ 5 & 2 & 7 \end{bmatrix} \left. \vphantom{\begin{bmatrix} 2 & 1 & 3 \\ 9 & 5 & 6 \\ 0 & 8 & 1 \\ 5 & 2 & 7 \end{bmatrix}} \right\} p = 4$$

$q = 3$

- 3×2 的矩阵 $V_{3,2}$

$$V = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} \left. \vphantom{\begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}} \right\} q = 2$$

$r = 2$

问题：如何计算 $U \cdot V$ ？

- 2个矩阵相乘

- $$U = \begin{bmatrix} 2 & 1 & 3 \\ 9 & 5 & 6 \\ 0 & 8 & 1 \\ 5 & 2 & 7 \end{bmatrix}, \quad p = 4 \quad V = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}, \quad q = 3 \quad Z = UV = \begin{bmatrix} & \\ & \\ & \\ & \end{bmatrix}$$

$\underbrace{\hspace{10em}}_{q=3} \qquad \underbrace{\hspace{5em}}_{r=2}$

问题背景：基本知识



- 2个矩阵相乘

- $$U = \begin{bmatrix} 2 & 1 & 3 \\ 9 & 5 & 6 \\ 0 & 8 & 1 \\ 5 & 2 & 7 \end{bmatrix}, p = 4, \quad V = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}, q = 3, \quad Z = UV = \begin{bmatrix} 13 \\ \\ \\ \end{bmatrix}$$

$q = 3$ $r = 2$

- $$2 \times 1 + 1 \times 2 + 3 \times 3 = 13$$

- 2个矩阵相乘

- $$U = \begin{bmatrix} 2 & 1 & 3 \\ 9 & 5 & 6 \\ 0 & 8 & 1 \\ 5 & 2 & 7 \end{bmatrix}, p = 4, q = 3 \quad V = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}, q = 3, r = 2 \quad Z = UV = \begin{bmatrix} 13 & 31 \\ \vdots & \vdots \end{bmatrix}$$

- $$2 \times 4 + 1 \times 5 + 3 \times 6 = 31$$

- 2个矩阵相乘

- $$U = \begin{bmatrix} 2 & 1 & 3 \\ 9 & 5 & 6 \\ 0 & 8 & 1 \\ 5 & 2 & 7 \end{bmatrix}, p = 4 \quad V = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}, q = 3 \quad Z = UV = \begin{bmatrix} 13 & 31 \\ 37 & \end{bmatrix}$$

$q = 3$ $r = 2$

- $$9 \times 1 + 5 \times 2 + 6 \times 3 = 37$$

- 2个矩阵相乘

- $$U = \begin{bmatrix} 2 & 1 & 3 \\ 9 & 5 & 6 \\ 0 & 8 & 1 \\ 5 & 2 & 7 \end{bmatrix}, \quad p = 4 \quad V = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}, \quad q = 3 \quad Z = UV = \begin{bmatrix} 13 & 31 \\ 37 & 97 \\ 19 & 46 \\ 30 & 72 \end{bmatrix}$$

$\underbrace{\hspace{10em}}_{q=3} \quad \underbrace{\hspace{5em}}_{r=2}$

- 2个矩阵相乘

- $$U = \begin{bmatrix} 2 & 1 & 3 \\ 9 & 5 & 6 \\ 0 & 8 & 1 \\ 5 & 2 & 7 \end{bmatrix}, \quad \left. \begin{array}{c} \text{ } \\ \text{ } \\ \text{ } \\ \text{ } \end{array} \right\} p = 4$$
$$\underbrace{\hspace{1.5cm}}_{q = 3}$$
$$V = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}, \quad \left. \begin{array}{c} \text{ } \\ \text{ } \\ \text{ } \end{array} \right\} q = 3$$
$$\underbrace{\hspace{1.5cm}}_{r = 2}$$
$$Z = UV = \begin{bmatrix} 13 & 31 \\ 37 & 97 \\ 19 & 46 \\ 30 & 72 \end{bmatrix}, \quad \left. \begin{array}{c} \text{ } \\ \text{ } \\ \text{ } \\ \text{ } \end{array} \right\} p = 4$$
$$\underbrace{\hspace{1.5cm}}_{r = 2}$$

- 2个矩阵相乘

- $$U = \begin{bmatrix} 2 & 1 & 3 \\ 9 & 5 & 6 \\ 0 & 8 & 1 \\ 5 & 2 & 7 \end{bmatrix}, p = 4 \quad V = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}, q = 3 \quad Z = UV = \begin{bmatrix} 13 & 31 \\ 37 & 97 \\ 19 & 46 \\ 30 & 72 \end{bmatrix}, p = 4$$

$q = 3$ $r = 2$ $r = 2$

- 矩阵乘法的时间复杂度

- 计算1个数字： q 次标量乘法

- 2个矩阵相乘

- $$U = \begin{bmatrix} 2 & 1 & 3 \\ 9 & 5 & 6 \\ 0 & 8 & 1 \\ 5 & 2 & 7 \end{bmatrix}, \quad p = 4, \quad q = 3, \quad V = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}, \quad r = 2, \quad Z = UV = \begin{bmatrix} 13 & 31 \\ 37 & 97 \\ 19 & 46 \\ 30 & 72 \end{bmatrix}, \quad p = 4, \quad r = 2$$

- 矩阵乘法的时间复杂度

- 计算1个数字： q 次标量乘法
- 共 $p \times r$ 个数： $\Theta(pqr)$

- 2个矩阵相乘

- $$U = \begin{bmatrix} 2 & 1 & 3 \\ 9 & 5 & 6 \\ 0 & 8 & 1 \\ 5 & 2 & 7 \end{bmatrix} \left. \vphantom{\begin{bmatrix} 2 & 1 & 3 \\ 9 & 5 & 6 \\ 0 & 8 & 1 \\ 5 & 2 & 7 \end{bmatrix}} \right\} p = 4 \quad V = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} \left. \vphantom{\begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}} \right\} q = 3 \quad Z = UV = \begin{bmatrix} 13 & 31 \\ 37 & 97 \\ 19 & 46 \\ 30 & 72 \end{bmatrix} \left. \vphantom{\begin{bmatrix} 13 & 31 \\ 37 & 97 \\ 19 & 46 \\ 30 & 72 \end{bmatrix}} \right\} p = 4$$

$$\underbrace{\hspace{10em}}_{q=3} \quad \underbrace{\hspace{10em}}_{r=2} \quad \underbrace{\hspace{10em}}_{r=2}$$

- 矩阵乘法的时间复杂度

- 计算1个数字： q 次标量乘法
- 共 $p \times r$ 个数： $\Theta(pqr)$
- 上例中，标量乘法次数为： $p \times q \times r = 4 \times 3 \times 2 = 24$

- 3个矩阵相乘
 - 矩阵乘法结合率: $(UV)W = U(VW)$
 - 新问题: 矩阵乘法结合的顺序

- 3个矩阵相乘

- 矩阵乘法结合率： $(UV)W = U(VW)$
- 新问题： 矩阵乘法结合的顺序

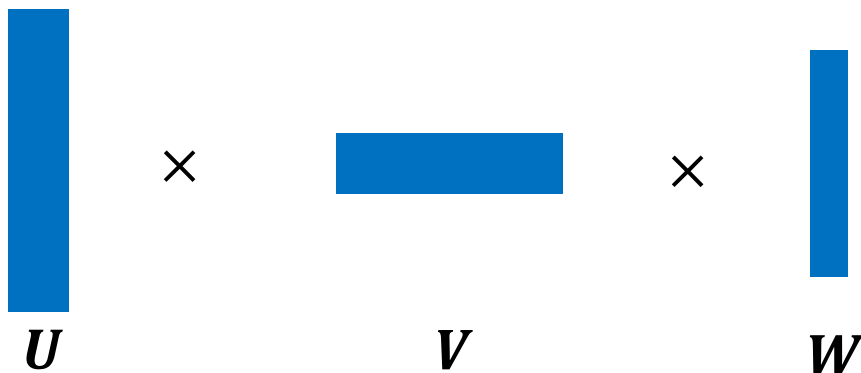
问题： 顺序不同，效率是否明显不同？

- 3个矩阵相乘

- 矩阵乘法结合率： $(UV)W = U(VW)$
- 新问题：矩阵乘法结合的顺序

问题：顺序不同，效率是否明显不同？

- 例如：矩阵维度数为 $p = 40$, $q = 8$, $r = 30$, $s = 5$

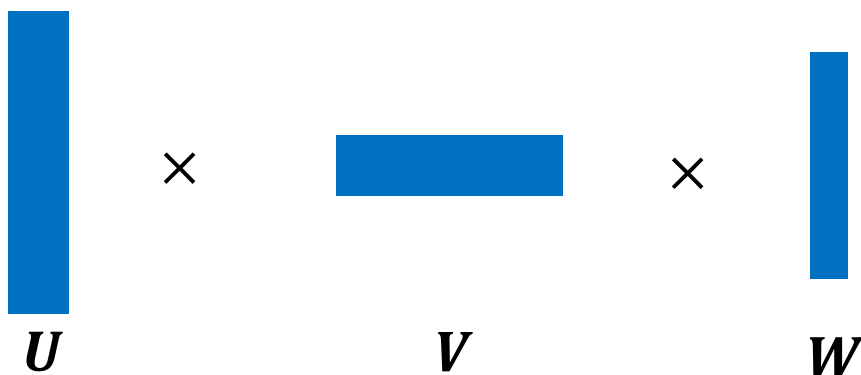


- 3个矩阵相乘

- 矩阵乘法结合率： $(UV)W = U(VW)$
- 新问题：矩阵乘法结合的顺序

问题：顺序不同，效率是否明显不同？

- 例如：矩阵维度数为 $p = 40, q = 8, r = 30, s = 5$



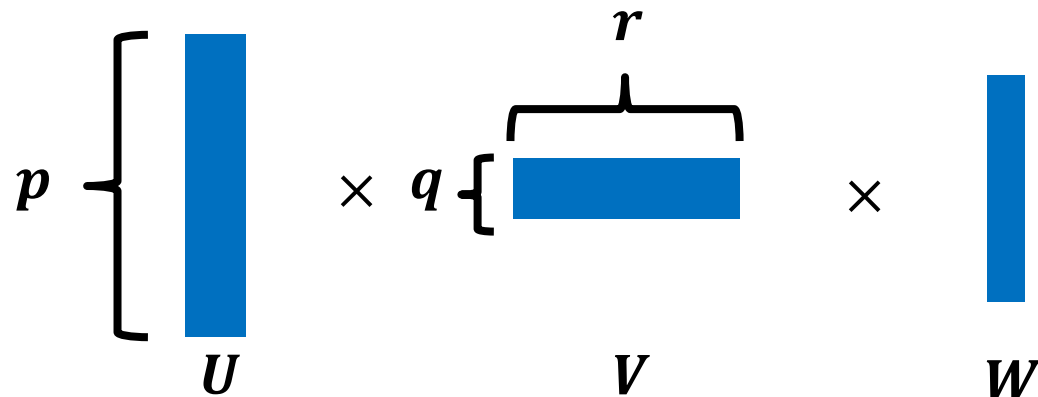
- 按 $(UV)W$ 计算，标量乘法次数：

- 3个矩阵相乘

- 矩阵乘法结合率： $(UV)W = U(VW)$
- 新问题：矩阵乘法结合的顺序

问题：顺序不同，效率是否明显不同？

- 例如：矩阵维度数为 $p = 40$, $q = 8$, $r = 30$, $s = 5$



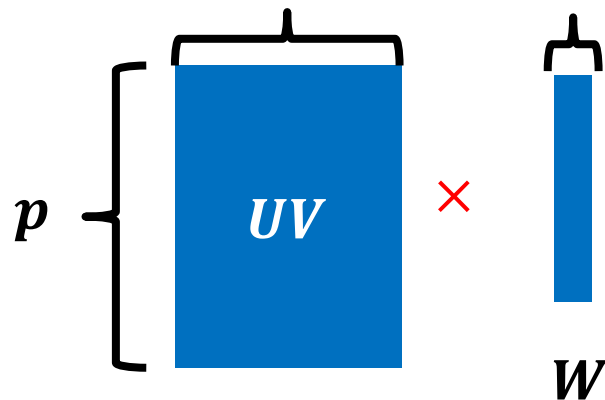
- 按 $(UV)W$ 计算，标量乘法次数： pqr

- 3个矩阵相乘

- 矩阵乘法结合率： $(UV)W = U(VW)$
- 新问题：矩阵乘法结合的顺序

问题：顺序不同，效率是否明显不同？

- 例如：矩阵维度数为 $p = 40$, $q = 8$, $r = 30$, $s = 5$



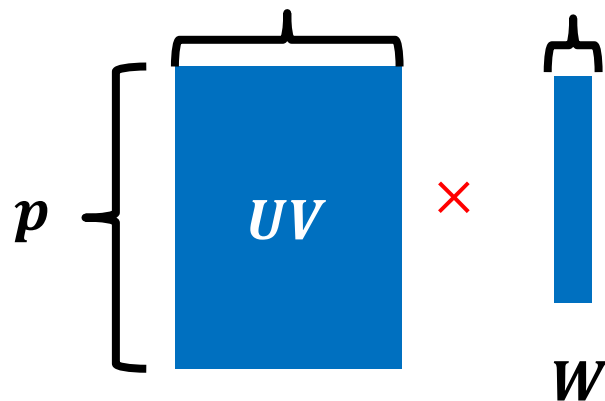
- 按 $(UV)W$ 计算，标量乘法次数： $pqr + prs$

- 3个矩阵相乘

- 矩阵乘法结合率： $(UV)W = U(VW)$
- 新问题：矩阵乘法结合的顺序

问题：顺序不同，效率是否明显不同？

- 例如：矩阵维度数为 $p = 40$, $q = 8$, $r = 30$, $s = 5$



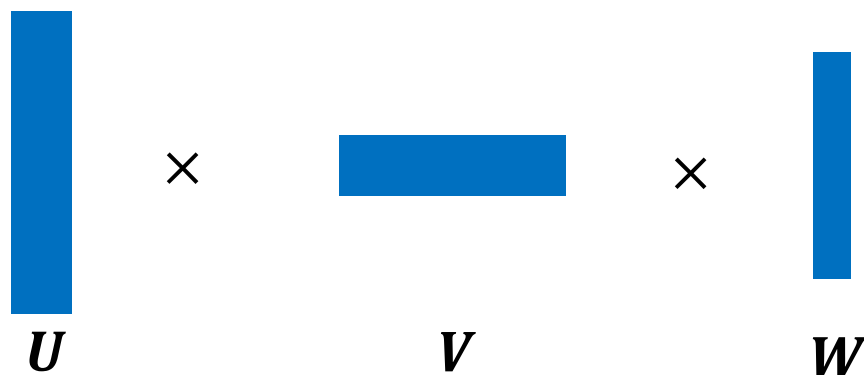
- 按 $(UV)W$ 计算，标量乘法次数： $pqr + prs = 15600$

- 3个矩阵相乘

- 矩阵乘法结合率： $(UV)W = U(VW)$
- 新问题：矩阵乘法结合的顺序

问题：顺序不同，效率是否明显不同？

- 例如：矩阵维度数为 $p = 40, q = 8, r = 30, s = 5$



- 按 $(UV)W$ 计算，标量乘法次数： $pqr + prs = 15600$
- 按 $U(VW)$ 计算，标量乘法次数：

问题背景：基本知识

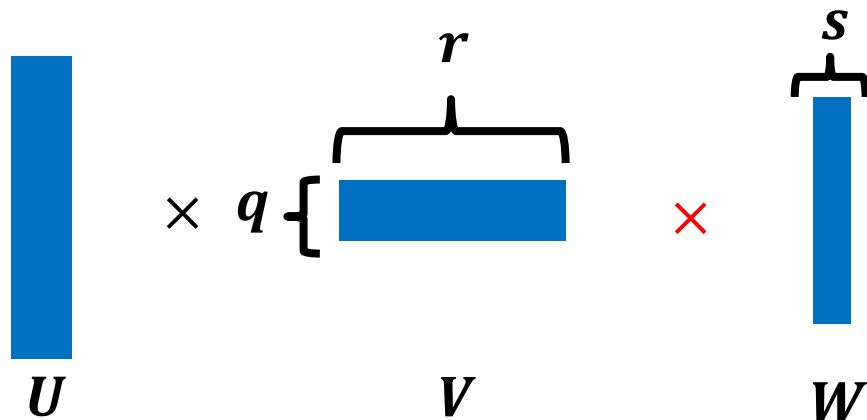


- 3个矩阵相乘

- 矩阵乘法结合率： $(UV)W = U(VW)$
- 新问题：矩阵乘法结合的顺序

问题：顺序不同，效率是否明显不同？

- 例如：矩阵维度数为 $p = 40, q = 8, r = 30, s = 5$



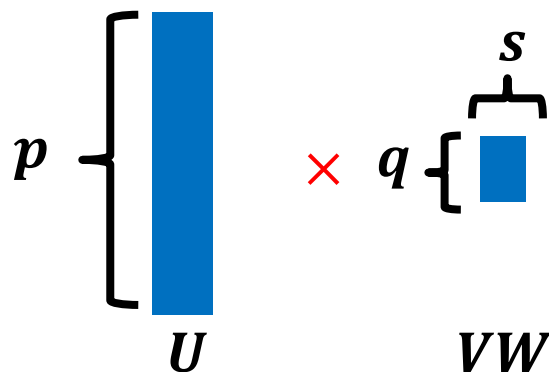
- 按 $(UV)W$ 计算，标量乘法次数： $pqr + prs = 15600$
- 按 $U(VW)$ 计算，标量乘法次数： qrs

- 3个矩阵相乘

- 矩阵乘法结合率： $(UV)W = U(VW)$
- 新问题：矩阵乘法结合的顺序

问题：顺序不同，效率是否明显不同？

- 例如：矩阵维度数为 $p = 40, q = 8, r = 30, s = 5$



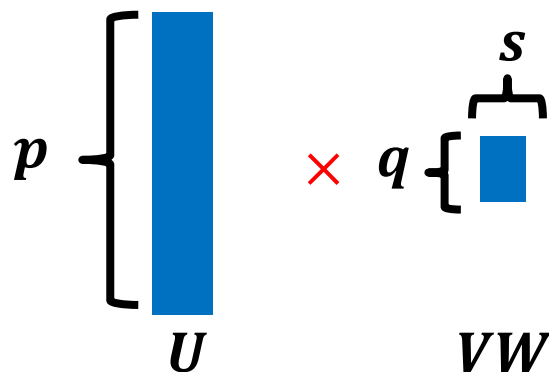
- 按 $(UV)W$ 计算，标量乘法次数： $pqr + prs = 15600$
- 按 $U(VW)$ 计算，标量乘法次数： $qrs + pqs$

- 3个矩阵相乘

- 矩阵乘法结合率： $(UV)W = U(VW)$
- 新问题：矩阵乘法结合的顺序

问题：顺序不同，效率是否明显不同？

- 例如：矩阵维度数为 $p = 40, q = 8, r = 30, s = 5$



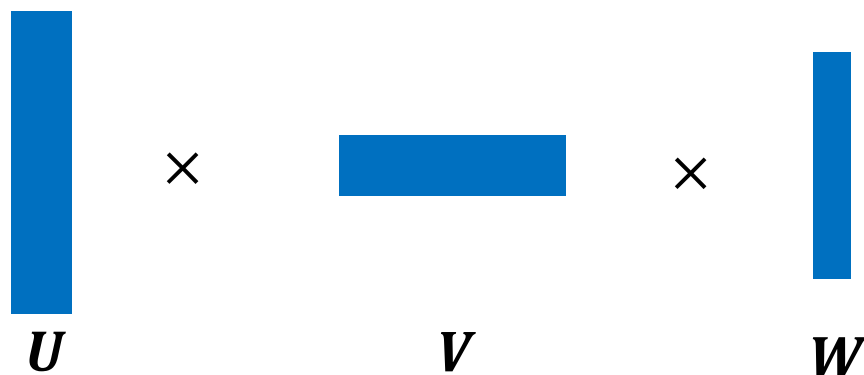
- 按 $(UV)W$ 计算，标量乘法次数： $pqr + prs = 15600$
- 按 $U(VW)$ 计算，标量乘法次数： $qrs + pqs = 2800$

- 3个矩阵相乘

- 矩阵乘法结合率： $(UV)W = U(VW)$
- 新问题：矩阵乘法结合的顺序

问题：顺序不同，效率是否明显不同？

- 例如：矩阵维度数为 $p = 40, q = 8, r = 30, s = 5$



- 按 $(UV)W$ 计算，标量乘法次数： $pqr + prs = 15600$
- 按 $U(VW)$ 计算，标量乘法次数： $qrs + pqs = 2800$

差异显著

问题背景：基本知识



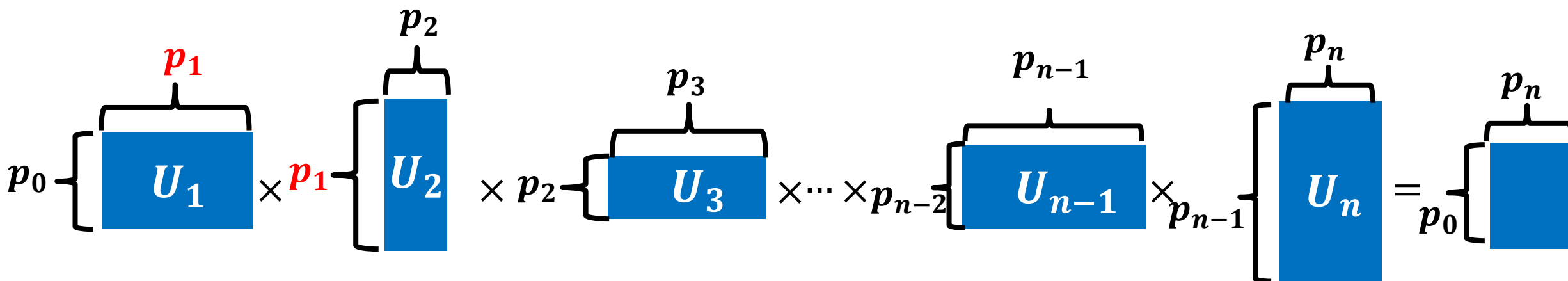
- n 个矩阵相乘
 - 有一系列矩阵按顺序排列

$$U_1 \times U_2 \times U_3 \times \cdots \times U_{n-1} \times U_n =$$

问题背景：基本知识



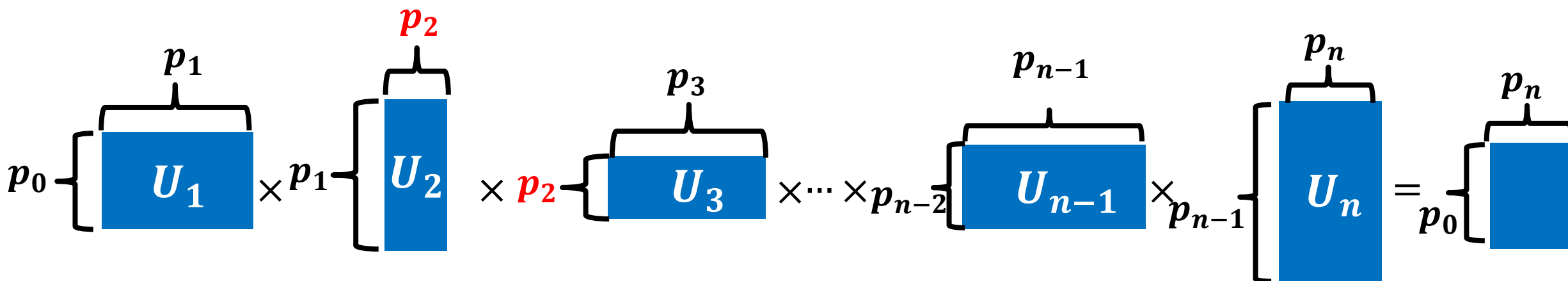
- n 个矩阵相乘
 - 有一系列矩阵按顺序排列
 - 每个矩阵的行数=前一个矩阵的列数



问题背景：基本知识



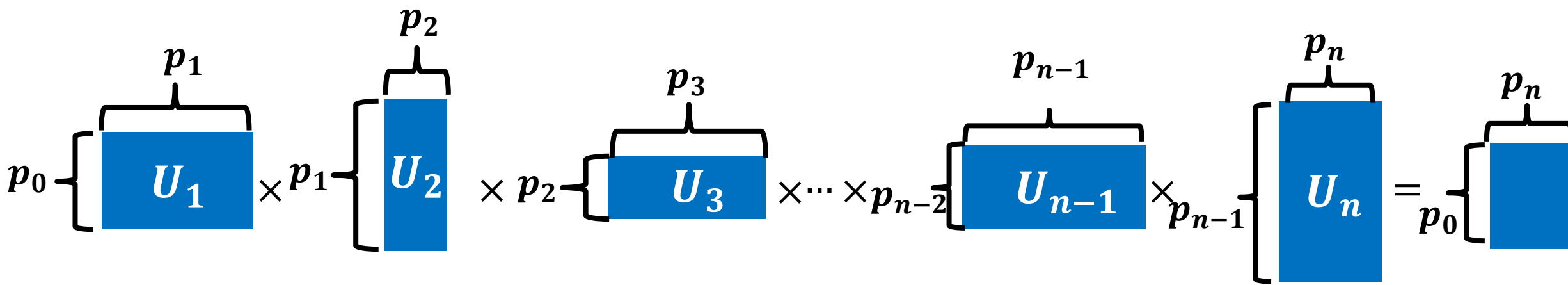
- n 个矩阵相乘
 - 有一系列矩阵按顺序排列
 - 每个矩阵的行数=前一个矩阵的列数



问题背景：基本知识



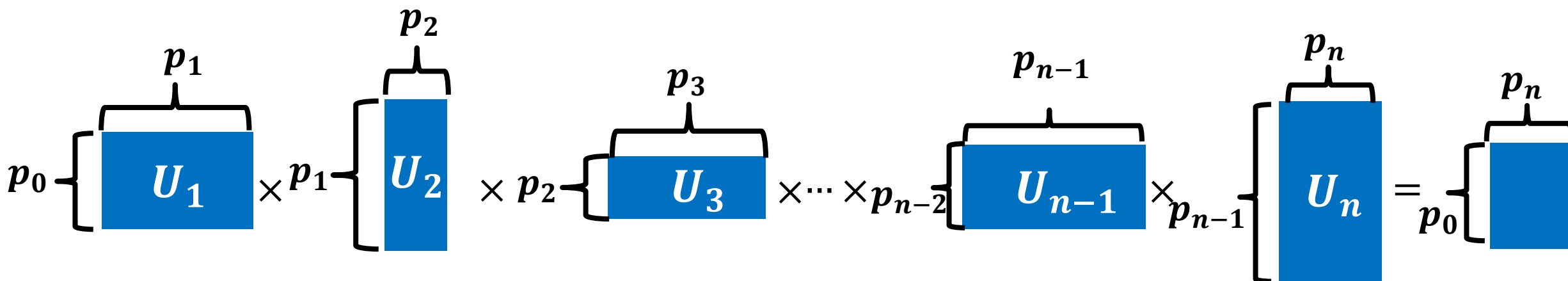
- n 个矩阵相乘
 - 有一系列矩阵按顺序排列
 - 每个矩阵的行数=前一个矩阵的列数



- n 个矩阵相乘也称为矩阵链乘法

问题背景：基本知识

- n 个矩阵相乘
 - 有一系列矩阵按顺序排列
 - 每个矩阵的行数=前一个矩阵的列数



- n 个矩阵相乘也称为**矩阵链乘法**

问题：如何确定相乘顺序（给矩阵链加括号），提高计算效率？

矩阵链乘法问题

Matrix-chain Multiplication Problem

输入

- n 个矩阵组成的矩阵链 $U_{1..n} = \langle U_1, U_2, \dots, U_n \rangle$
- 矩阵链 $U_{1..n}$ 对应的维度数分别为 p_0, p_1, \dots, p_n , U_i 的维度为 $p_{i-1} \times p_i$

输出

- 找到一种加括号的方式, 以确定矩阵链乘法的计算顺序, 使得

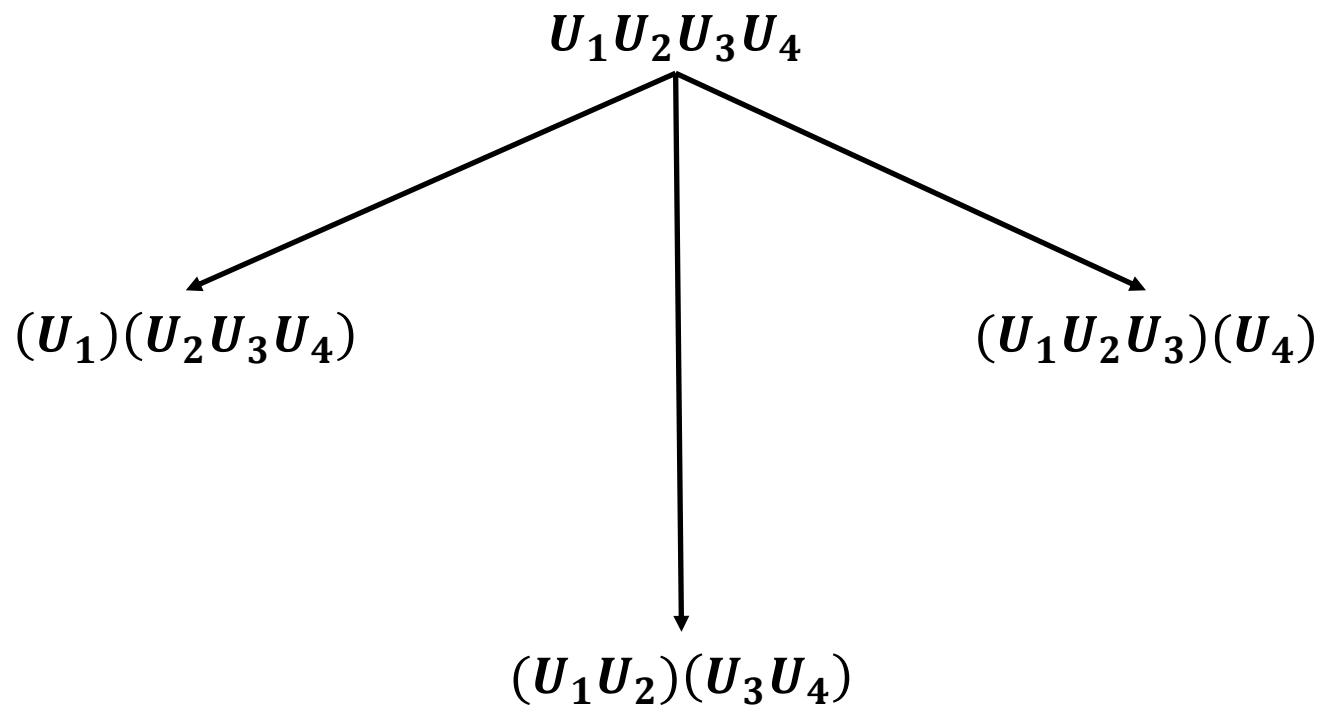
最小化矩阵链标量乘法的次数

问题示例



- 给定矩阵链： $U_{1..4} = U_1, U_2, U_3, U_4$
- 有如下加括号方式

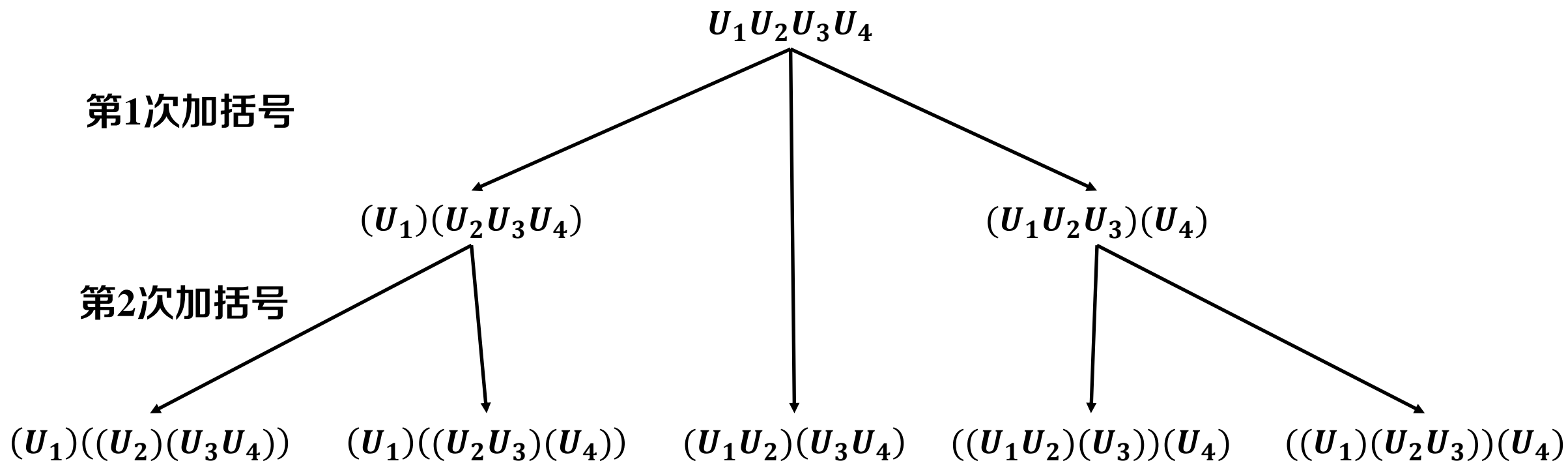
第1次加括号



问题示例

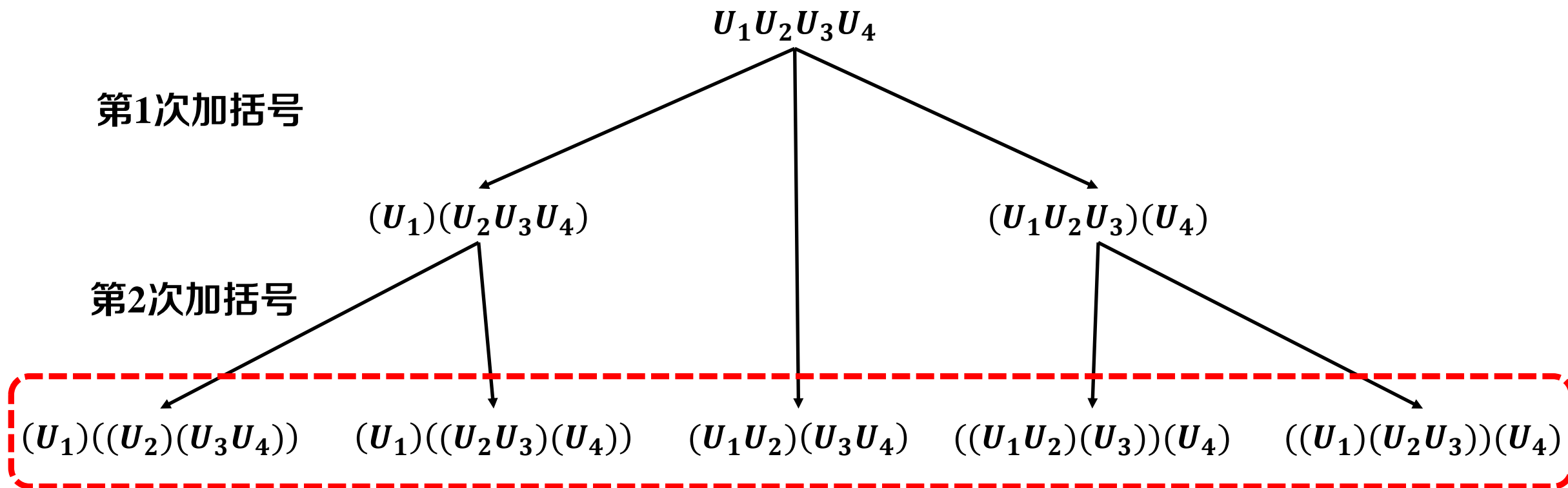


- 给定矩阵链: $U_{1..4} = U_1, U_2, U_3, U_4$
- 有如下加括号方式



问题示例

- 给定矩阵链: $U_{1..4} = U_1, U_2, U_3, U_4$
- 有如下加括号方式



找到使标量乘法次数最小的加括号方式

问题结构分析



- 给出问题表示

- $D[i, j]$: 计算矩阵链 $U_{i..j}$ 所需标量乘法的最小次数

$$U_{i..j} = p_{i-1} \underbrace{\left[\overbrace{U_i}^{p_i} \times \cdots \times p_{j-1} \overbrace{U_j}^{p_j} \right]}_{D[i, j]}$$

- 明确原始问题

- $D[1, n]$: 计算矩阵链 $U_{1..n}$ 所需标量乘法的最小次数

问题结构分析



递推关系建立



自底向上计算

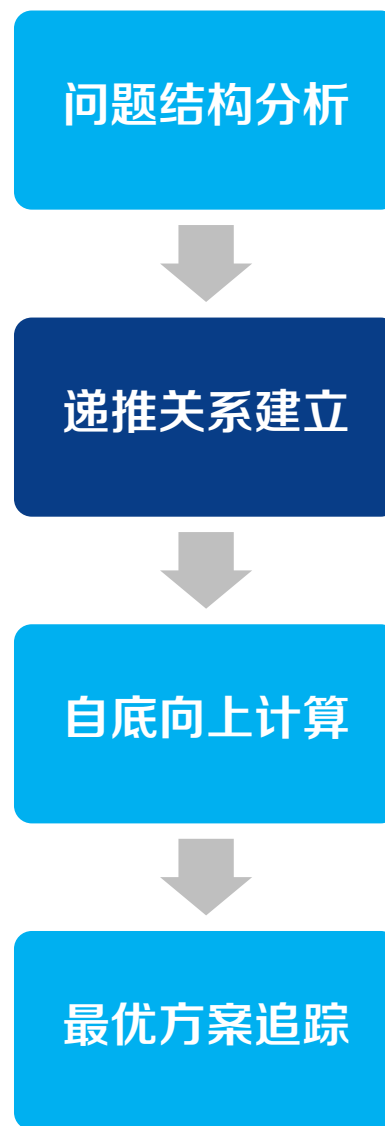


最优方案追踪

递推关系建立：分析最优（子）结构

- 对矩阵链 $U_{i..j}$ ，求解 $D[i, j]$

$$U_i \cdots U_k U_{k+1} \cdots U_j$$



递推关系建立：分析最优（子）结构

- 对矩阵链 $U_{i..j}$ ，求解 $D[i, j]$

- 加一次括号

$$U_i \dots U_k \overset{\text{某位置 } k(i \leq k < j)}{U_{k+1} \dots U_j}$$
$$\left(U_i \dots U_k \right) \times \left(U_{k+1} \dots U_j \right)$$

问题结构分析

递推关系建立

自底向上计算

最优方案追踪

递推关系建立：分析最优（子）结构

- 对矩阵链 $U_{i..j}$ ，求解 $D[i, j]$

$$\begin{array}{c} U_i \dots U_k \textcolor{red}{|} U_{k+1} \dots U_j \\ \swarrow \quad \quad \quad \searrow \\ \text{某位置 } k (i \leq k < j) \\ \left(U_i \dots U_k \right) \quad \times \quad \left(U_{k+1} \dots U_j \right) \end{array}$$

问题：如何保证不遗漏最优分割位置？

问题结构分析

递推关系建立

自底向上计算

最优方案追踪

递推关系建立：分析最优（子）结构

- 对矩阵链 $U_{i..j}$ ，求解 $D[i, j]$

$$\begin{array}{c} U_i \dots U_k \text{ } | \text{ } U_{k+1} \dots U_j \\ \swarrow \quad \quad \quad \searrow \\ \text{某位置 } k (i \leq k < j) \\ \left(U_i \dots U_k \right) \quad \times \quad \left(U_{k+1} \dots U_j \right) \end{array}$$

问题：如何保证不遗漏最优分割位置？

答案：枚举所有可能位置 $i..j-1$ ，共 $j-i$ 种

问题结构分析

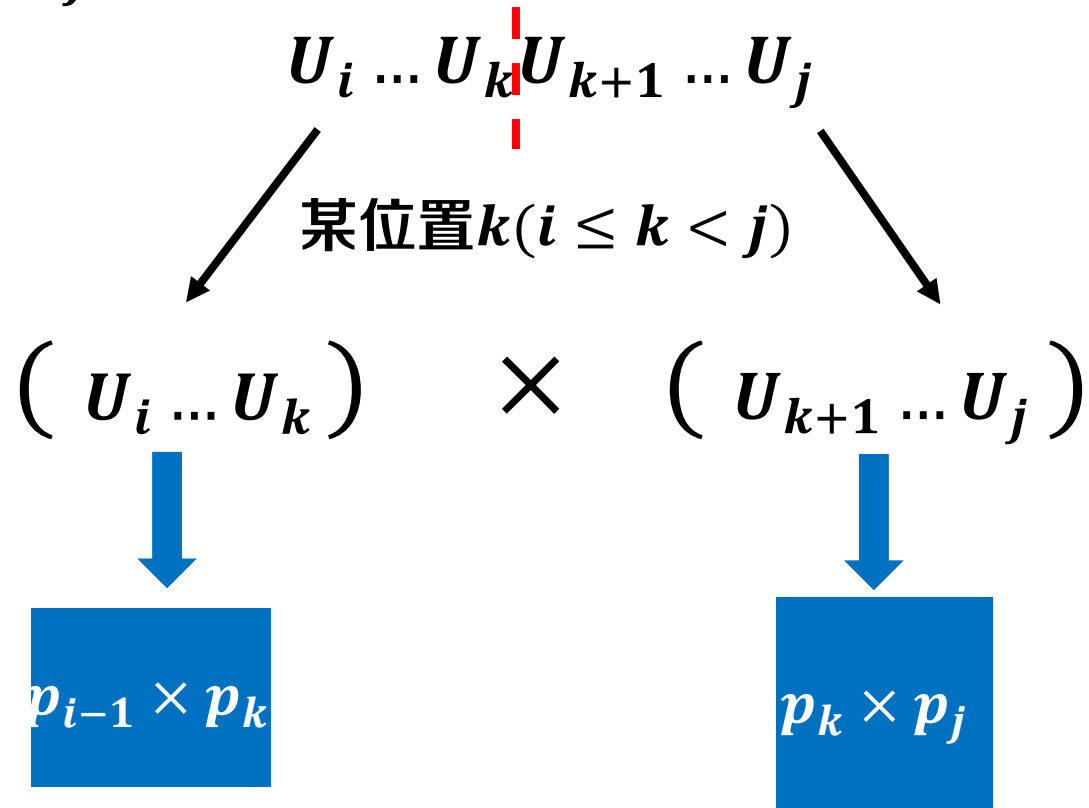
递推关系建立

自底向上计算

最优方案追踪

递推关系建立：分析最优（子）结构

- 对矩阵链 $U_{i..j}$ ，求解 $D[i, j]$



问题结构分析

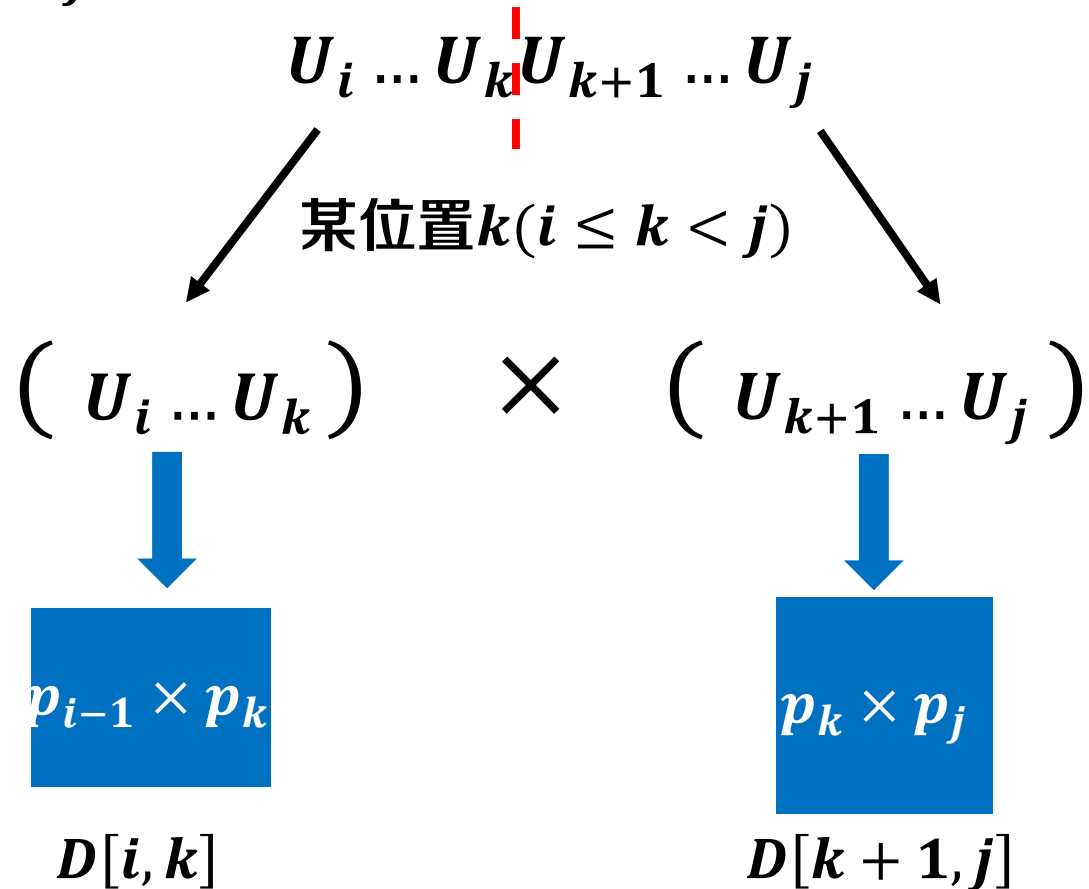
递推关系建立

自底向上计算

最优方案追踪

递推关系建立：分析最优（子）结构

- 对矩阵链 $U_{i..j}$ ，求解 $D[i, j]$



问题结构分析

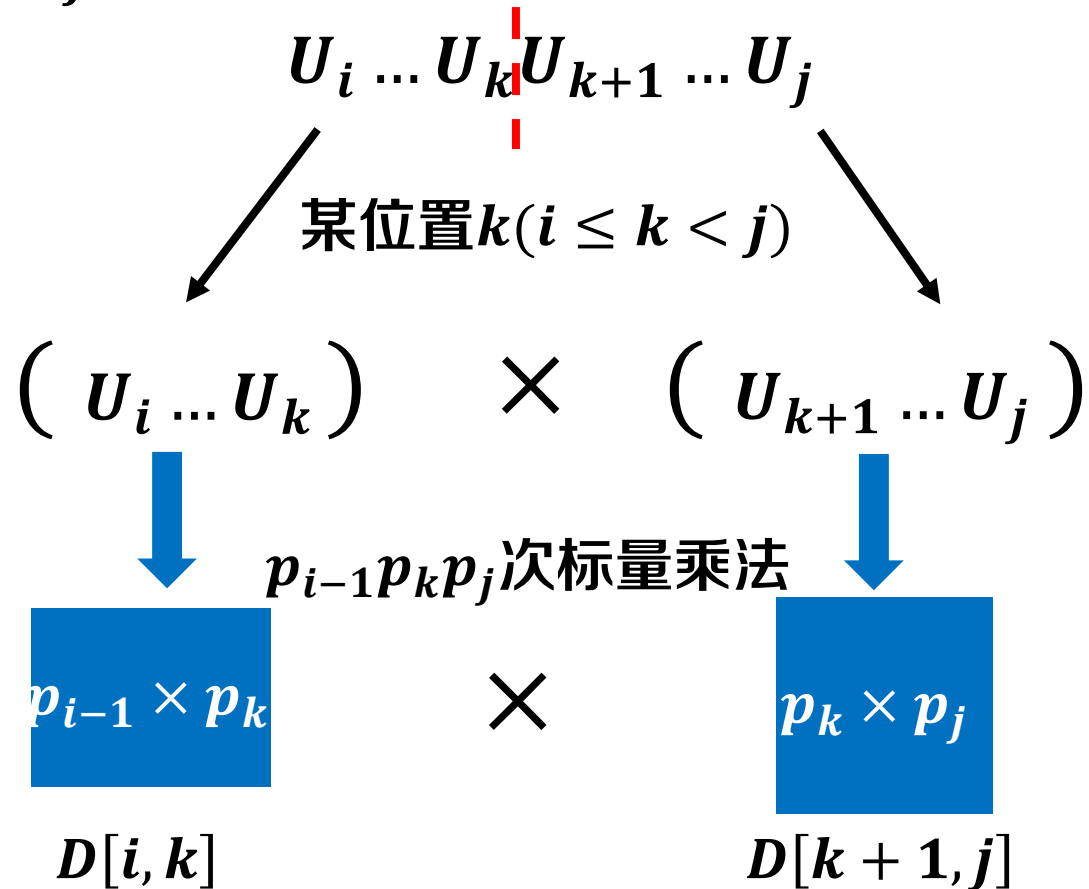
递推关系建立

自底向上计算

最优方案追踪

递推关系建立：分析最优（子）结构

- 对矩阵链 $U_{i..j}$ ，求解 $D[i, j]$



问题结构分析

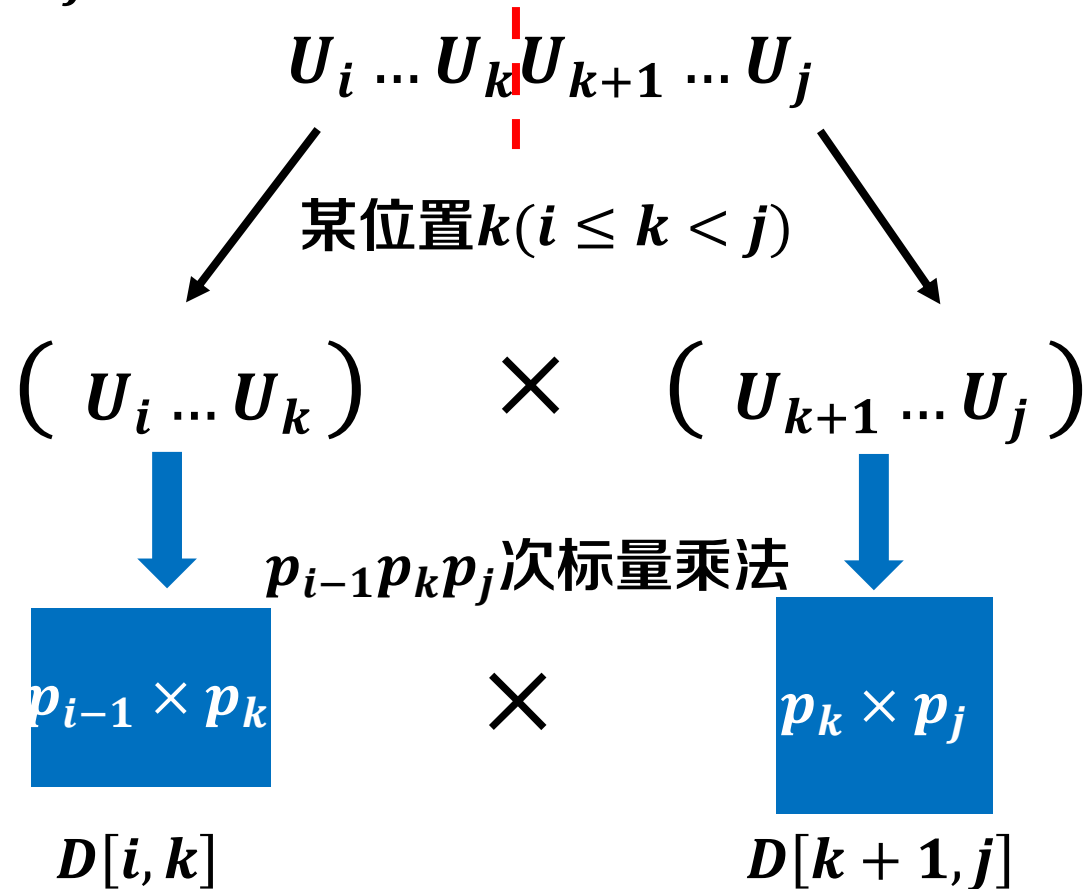
递推关系建立

自底向上计算

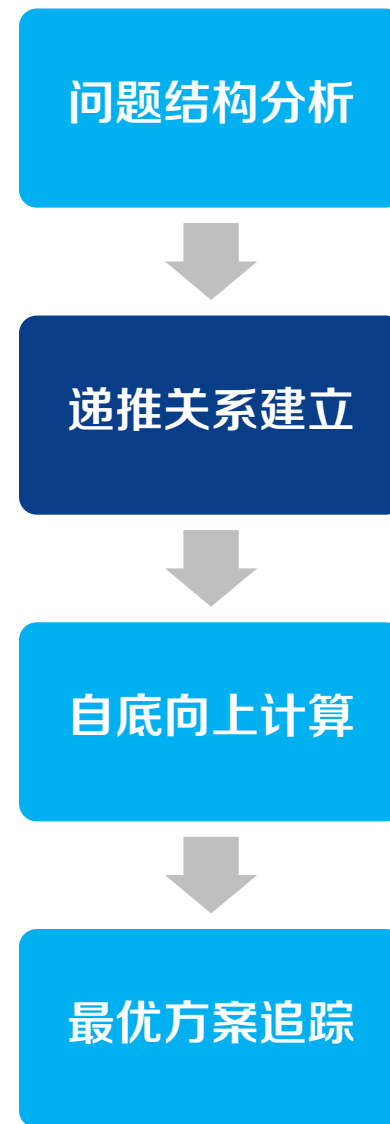
最优方案追踪

递推关系建立：分析最优（子）结构

- 对矩阵链 $U_{i..j}$ ，求解 $D[i, j]$

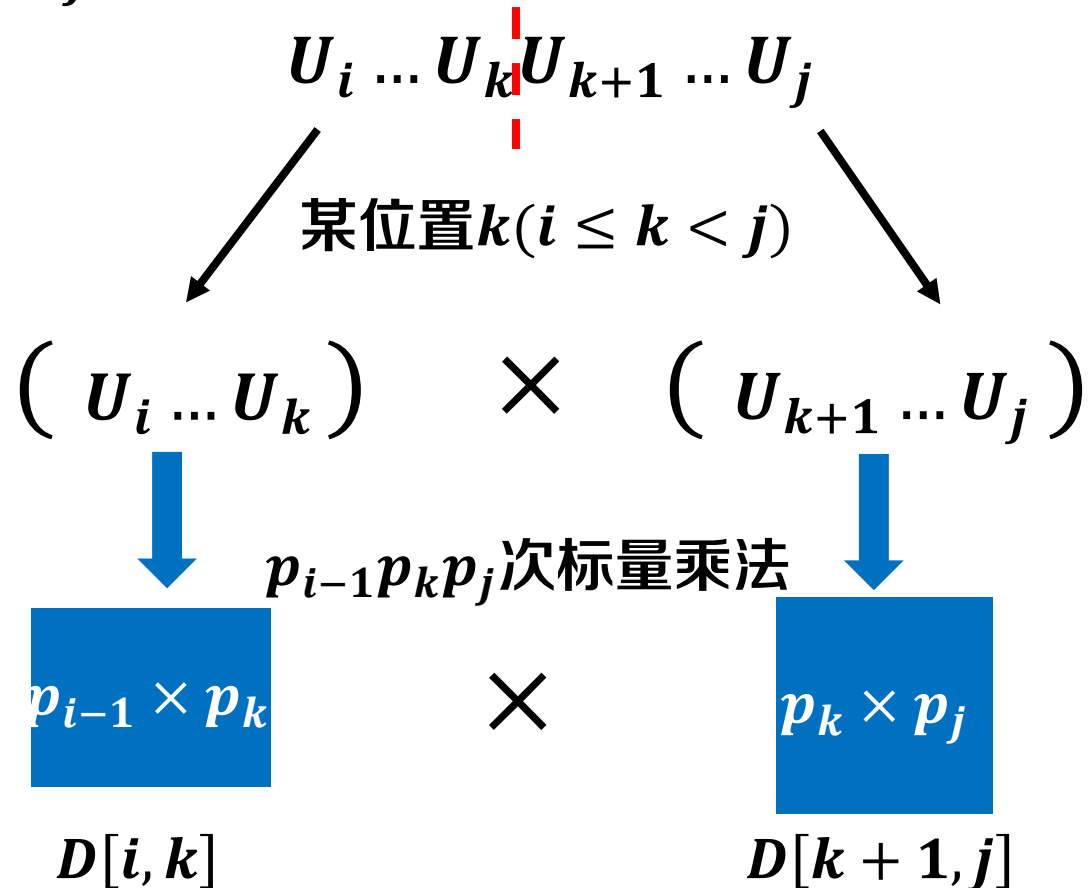


- $D[i, j] = D[i, k] + D[k+1, j] + p_{i-1}p_kp_j$



递推关系建立：分析最优（子）结构

- 对矩阵链 $U_{i..j}$ ，求解 $D[i, j]$



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

最优子结构

问题结构分析

递推关系建立

自底向上计算

最优方案追踪

递推关系建立：构造递推公式

- 对每个位置 $k(i \leq k < j)$
 - $D[i, j] = D[i, k] + D[k + 1, j] + p_{i-1}p_kp_j$
- 枚举所有 k ，得到递推式
 - $D[i, j] = \min_{i \leq k < j} (D[i, k] + D[k + 1, j] + p_{i-1}p_kp_j)$

问题结构分析



递推关系建立



自底向上计算



最优方案追踪

自底向上计算：确定计算顺序

- 初始化

- $i = j$ 时，矩阵链只有一个矩阵，乘法次数为0

$D[i, j]$	$i = 1$	2	3	$n - 1$	n
$i = 1$	0						
2		0					
3			0				
...				0			
...					0		
$n - 1$						0	
n							0

问题结构分析



递推关系建立



自底向上计算



最优方案追踪

自底向上计算：确定计算顺序

递推公式

- $$D[i, j] = \min_{i \leq k < j} (D[i, k] + D[k + 1, j] + p_{i-1}p_kp_j)$$

$D[i, j]$	$j = 1$	2	3	$n - 1$	n
$i = 1$	0						
2		0					
3			0				
...				0			
...					0		
$n - 1$						0	
n							0

$i < j$ 只用上三角

问题结构分析



递推关系建立



自底向上计算



最优方案追踪

自底向上计算：确定计算顺序

递推公式

- $$D[i, j] = \min_{i \leq k < j} (D[i, k] + D[k + 1, j] + p_{i-1}p_kp_j)$$

$j = 1$	2	3	$n - 1$	n	$D[i, j]$
0							$i = 1$
	0						2
		0					3
			0				...
				0			...
					0		$n - 1$
						0	n

问题结构分析



递推关系建立



自底向上计算



最优方案追踪

自底向上计算：确定计算顺序

递推公式

- $$D[i, j] = \min_{i \leq k < j} (D[i, k] + D[k + 1, j] + p_{i-1}p_kp_j)$$

$j = 1$	2	3	$n - 1$	n	$D[i, j]$
0							$i = 1$
	0						2
		0	$D[i, k]$	$D[i, j]$			3
			0		$D[k + 1, j]$...
				0			...
					0		$n - 1$
						0	n

问题结构分析



递推关系建立



自底向上计算



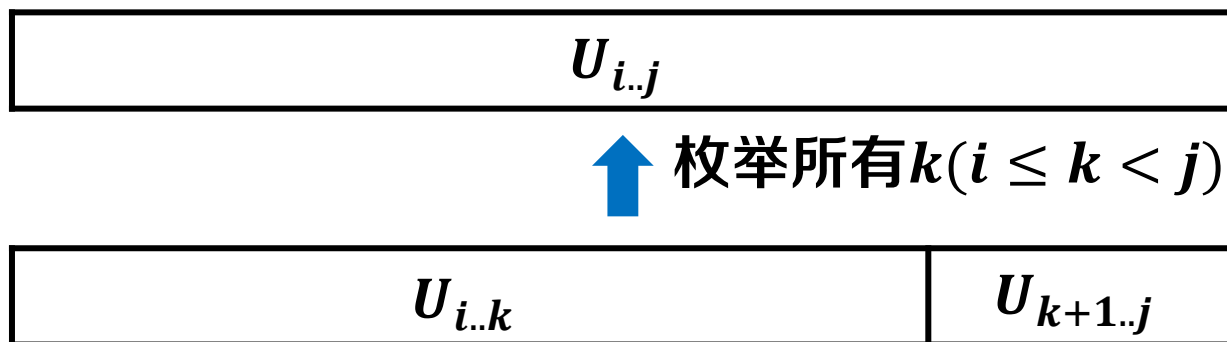
最优方案追踪

自底向上计算：确定计算顺序

- 递推公式

- $$D[i, j] = \min_{i \leq k < j} (D[i, k] + D[k + 1, j] + p_{i-1}p_kp_j)$$

- 观察枚举过程



问题结构分析

递推关系建立

自底向上计算

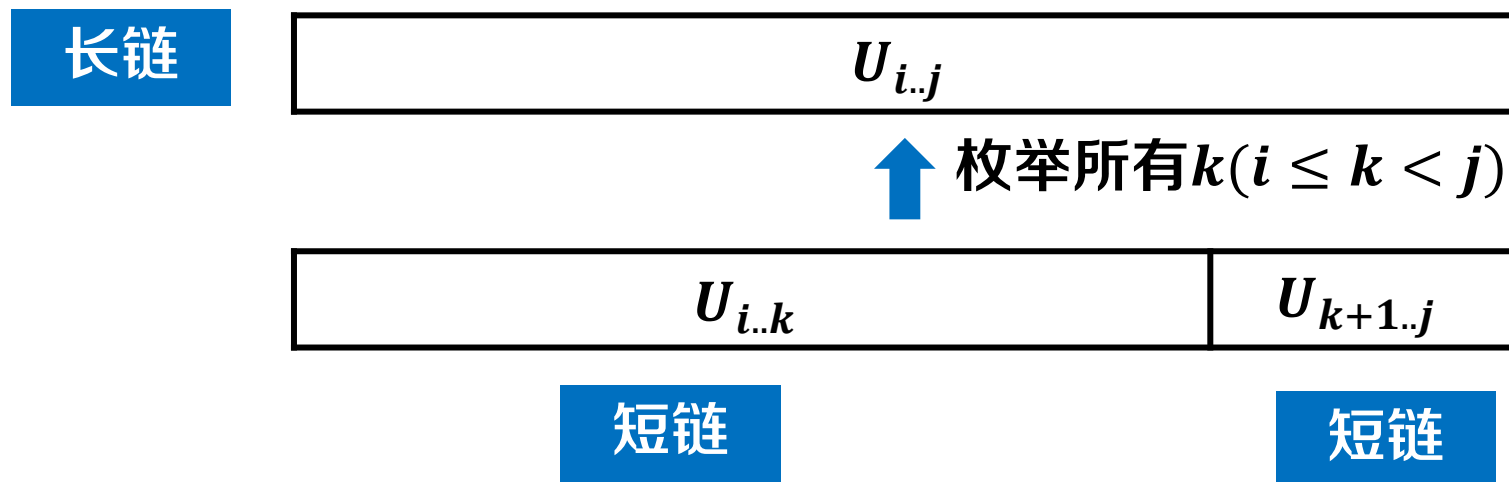
最优方案追踪

自底向上计算：确定计算顺序

- 递推公式

- $$D[i, j] = \min_{i \leq k < j} (D[i, k] + D[k + 1, j] + p_{i-1}p_kp_j)$$

- 观察枚举过程



问题结构分析

递推关系建立

自底向上计算

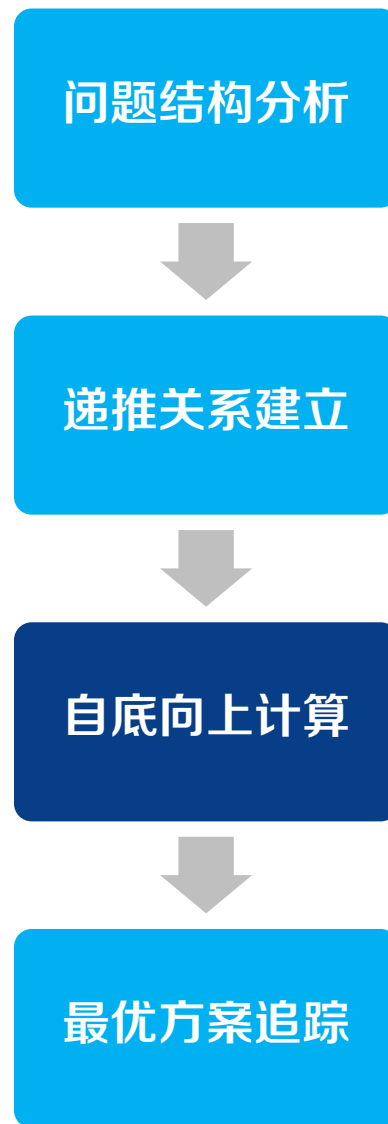
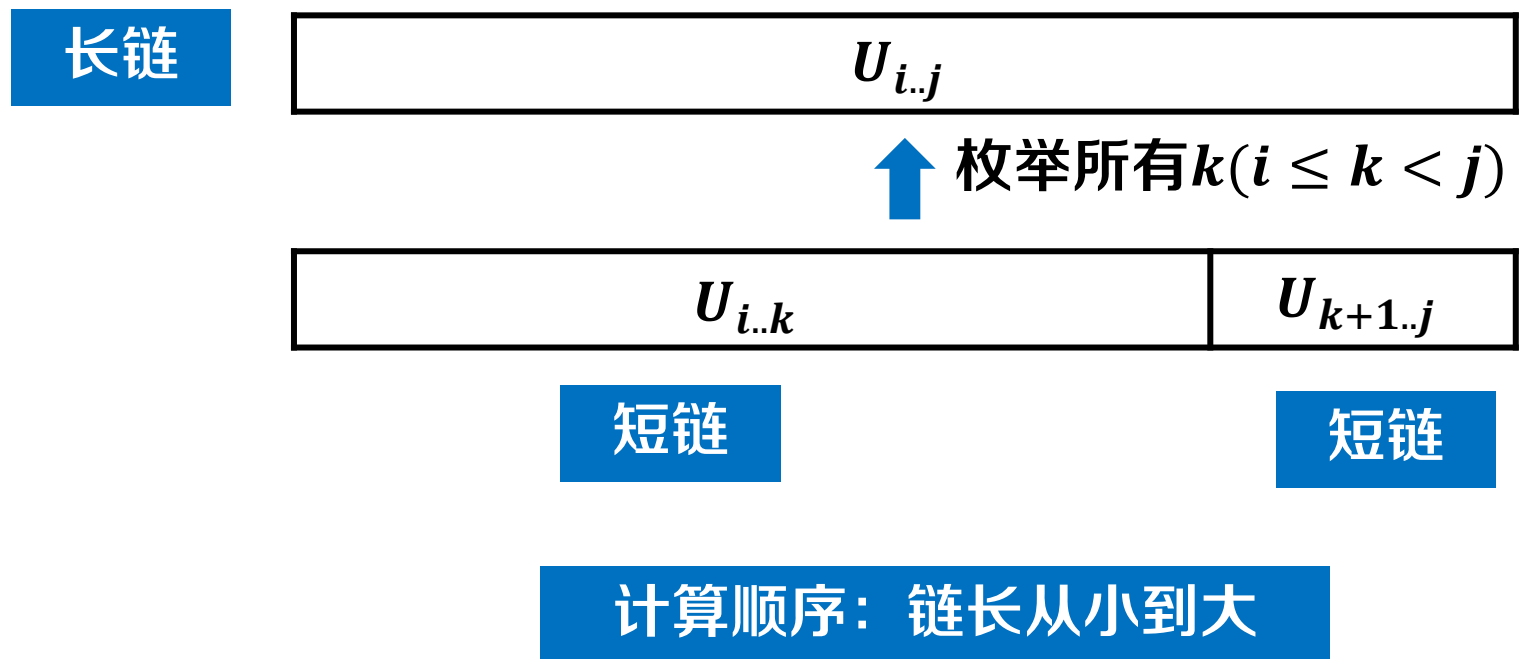
最优方案追踪

自底向上计算：确定计算顺序

- 递推公式

- $$D[i, j] = \min_{i \leq k < j} (D[i, k] + D[k + 1, j] + p_{i-1}p_kp_j)$$

- 观察枚举过程



自底向上计算：依次计算问题

递推公式

- $$D[i, j] = \min_{i \leq k < j} (D[i, k] + D[k + 1, j] + p_{i-1}p_kp_j)$$

$j = 1$	2	3	$n - 1$	n	$D[i, j]$
0							$i = 1$
	0						2
		0					3
			0				...
				0			...
					0		$n - 1$
						0	n

$j - i + 1 = 1$

$U_{i..j}$ 矩阵链长: $j - i + 1$

问题结构分析



递推关系建立



自底向上计算



最优方案追踪

自底向上计算：依次计算问题

递推公式

- $$D[i, j] = \min_{i \leq k < j} (D[i, k] + D[k + 1, j] + p_{i-1}p_kp_j)$$

$j = 1$	2	3	$n - 1$	n	$D[i, j]$
0							$i = 1$
	0						2
		0		$j - i + 1 = 2$			3
			0				...
				0			...
					0		$n - 1$
						0	n

矩阵链长： $j - i + 1$

问题结构分析



递推关系建立



自底向上计算



最优方案追踪

自底向上计算：依次计算问题

递推公式

- $$D[i, j] = \min_{i \leq k < j} (D[i, k] + D[k + 1, j] + p_{i-1}p_kp_j)$$

$j = 1$	2	3	$n - 1$	n	$D[i, j]$
0							$i = 1$
	0						2
		0					3
			0				...
				0			...
					0		$n - 1$
						0	n

$j - i + 1 = 3$

矩阵链长： $j - i + 1$

问题结构分析

递推关系建立

自底向上计算

最优方案追踪

自底向上计算：依次计算问题

递推公式

- $$D[i, j] = \min_{i \leq k < j} (D[i, k] + D[k + 1, j] + p_{i-1}p_kp_j)$$

$j = 1$	2	3	$n - 1$	n	$D[i, j]$
0						★	$i = 1$
	0						2
		0					3
			0				...
				0			...
					0		$n - 1$
						0	n

矩阵链长： $j - i + 1$

问题结构分析

递推关系建立

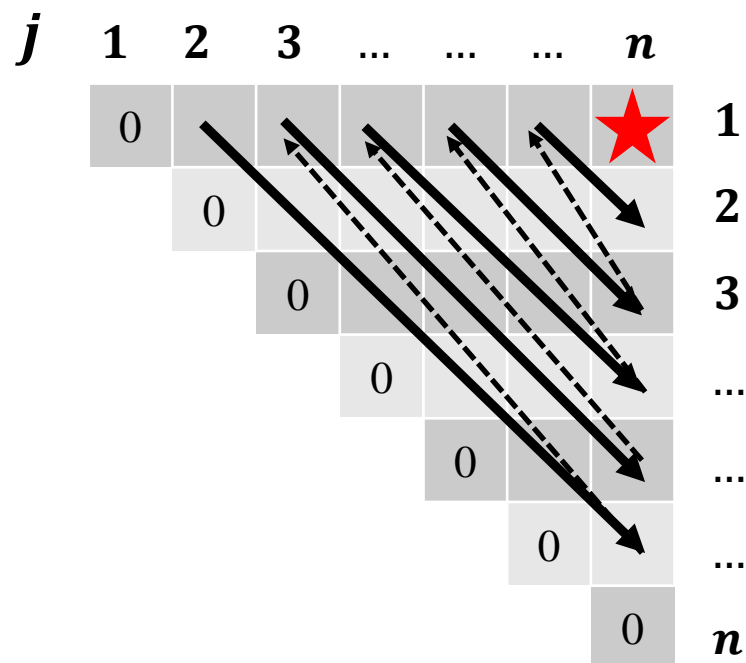
自底向上计算

最优方案追踪

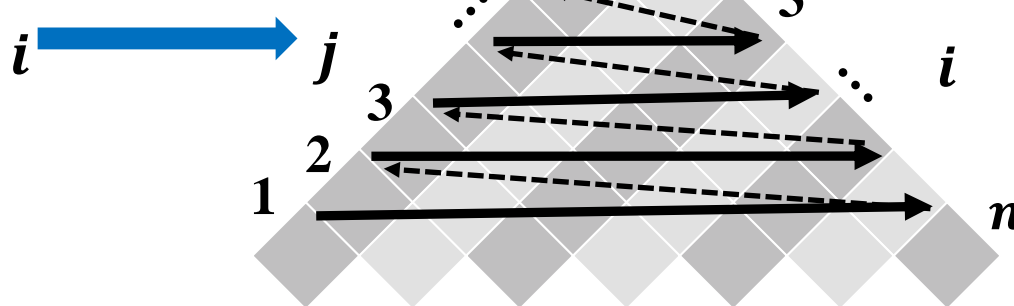
自底向上计算：依次计算问题

递推公式

- $$D[i, j] = \min_{i \leq k < j} (D[i, k] + D[k + 1, j] + p_{i-1}p_kp_j)$$



逆时针旋转45度



问题结构分析

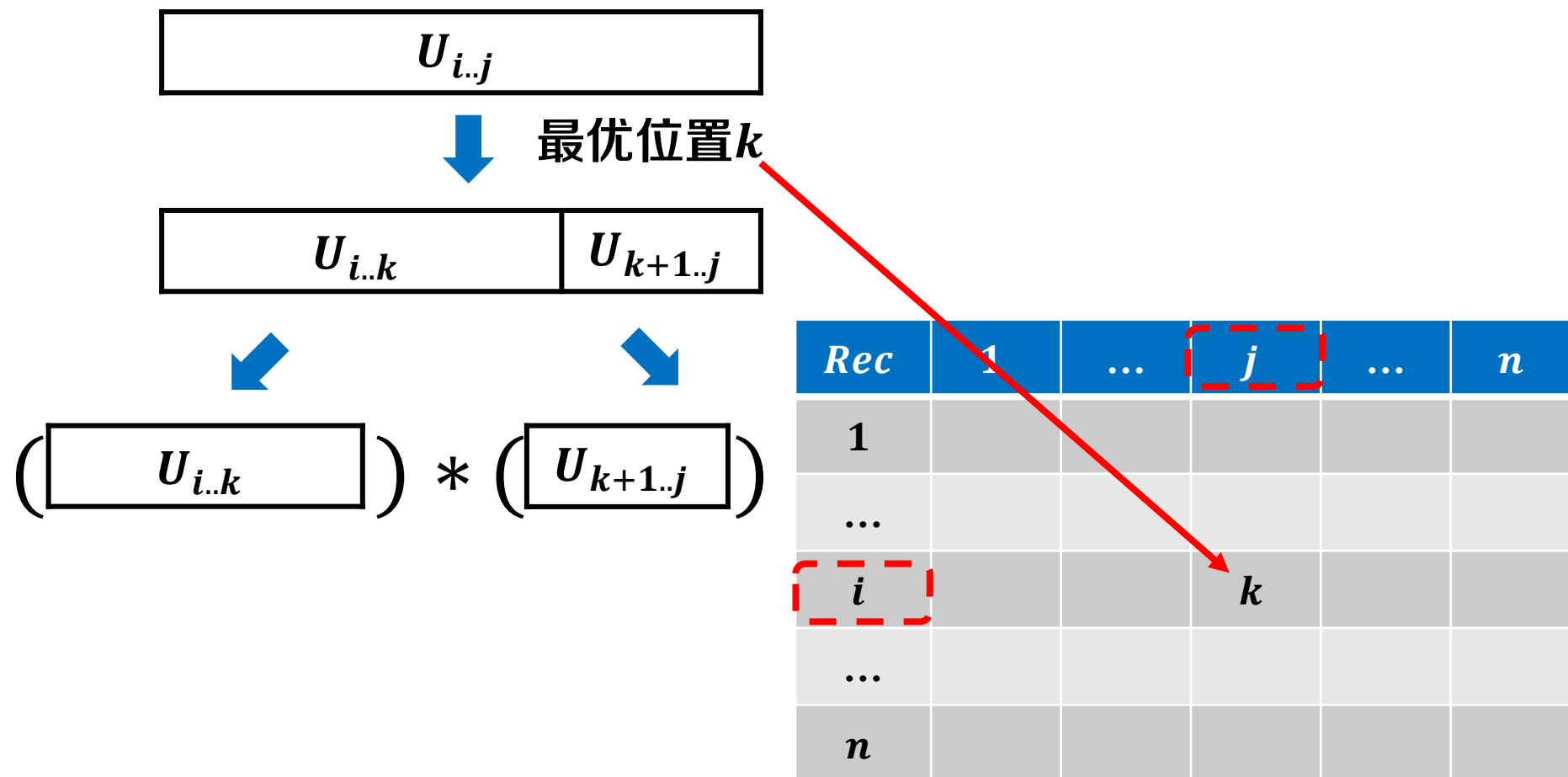
递推关系建立

自底向上计算

最优方案追踪

最优方案追踪：记录决策过程

- 构造追踪数组 $Rec[1..n, 1..n]$
- $Rec[i, j]$: 矩阵链 $U_{i..j}$ 的最优分割位置



问题结构分析

递推关系建立

自底向上计算

最优方案追踪

最优方案追踪：输出最优方案



- 根据追踪数组，递归输出方案

$$U_1 \dots U_s U_{s+1} \dots U_k U_{k+1} \dots U_t U_{t+1} \dots U_{n-1} U_n$$

<i>Rec</i>	$j = 1$	2	...	k	...	$n - 1$	n
$i = 1$							k
2							
...							
$k + 1$							
...							
$n - 1$							
n							

问题结构分析

递推关系建立

自底向上计算

最优方案追踪

最优方案追踪：输出最优方案



- 根据追踪数组，递归输出方案

$$(U_1 \dots U_s U_{s+1} \dots U_k)(U_{k+1} \dots U_t U_{t+1} \dots U_{n-1} U_n)$$

<i>Rec</i>	<i>j</i> = 1	2	...	<i>k</i>	...	<i>n</i> - 1	<i>n</i>
<i>i</i> = 1							<i>k</i>
2							
...							
<i>k</i> + 1							
...							
<i>n</i> - 1							
<i>n</i>							

问题结构分析

递推关系建立

自底向上计算

最优方案追踪

最优方案追踪：输出最优方案



- 根据追踪数组，递归输出方案

$$(U_1 \dots U_s U_{s+1} \dots U_k)(U_{k+1} \dots U_t U_{t+1} \dots U_{n-1} U_n)$$

<i>Rec</i>	$j = 1$	2	...	k	...	$n - 1$	n
$i = 1$				s			k
2							
...							
$k + 1$							
...							
$n - 1$							
n							

问题结构分析

递推关系建立

自底向上计算

最优方案追踪

最优方案追踪：输出最优方案



- 根据追踪数组，递归输出方案

$$(U_1 \dots U_s U_{s+1} \dots U_k)(U_{k+1} \dots U_t U_{t+1} \dots U_{n-1} U_n)$$

<i>Rec</i>	$j = 1$	2	...	k	...	$n - 1$	n
$i = 1$				s			k
2							
...							
$k + 1$							t
...							
$n - 1$							
n							

问题结构分析

递推关系建立

自底向上计算

最优方案追踪

最优方案追踪：输出最优方案



- 根据追踪数组，递归输出方案

$$(U_1 \dots U_s | U_{s+1} \dots U_k) (U_{k+1} \dots U_t | U_{t+1} \dots U_{n-1} U_n)$$

<i>Rec</i>	<i>j</i> = 1	2	...	<i>k</i>	...	<i>n</i> - 1	<i>n</i>
<i>i</i> = 1				<i>s</i>			<i>k</i>
2							
...							
<i>k</i>							<i>t</i>
...							
<i>n</i> - 1							
<i>n</i>							

问题结构分析

递推关系建立

自底向上计算

最优方案追踪

最优方案追踪：输出最优方案



- 根据追踪数组，递归输出方案

$$((U_1 \dots U_s)(U_{s+1} \dots U_k))(U_{k+1} \dots U_t)(U_{t+1} \dots U_{n-1}U_n))$$

<i>Rec</i>	$j = 1$	2	...	k	...	$n - 1$	n
$i = 1$				s			k
2							
...							
k							t
...							
$n - 1$							
n							

问题结构分析

递推关系建立

自底向上计算

最优方案追踪

最优方案追踪：输出最优方案



- 根据追踪数组，递归输出方案
 - 递归出口：矩阵链长为1

$$((U_1 \dots U_s)(U_{s+1} \dots U_k))(U_{k+1} \dots U_t)(U_{t+1} \dots U_{n-1}U_n))$$

<i>Rec</i>	$j = 1$	2	...	k	...	$n - 1$	n
$i = 1$				s			k
2							
...							
k							t
...							
$n - 1$							
n							

问题结构分析

递推关系建立

自底向上计算

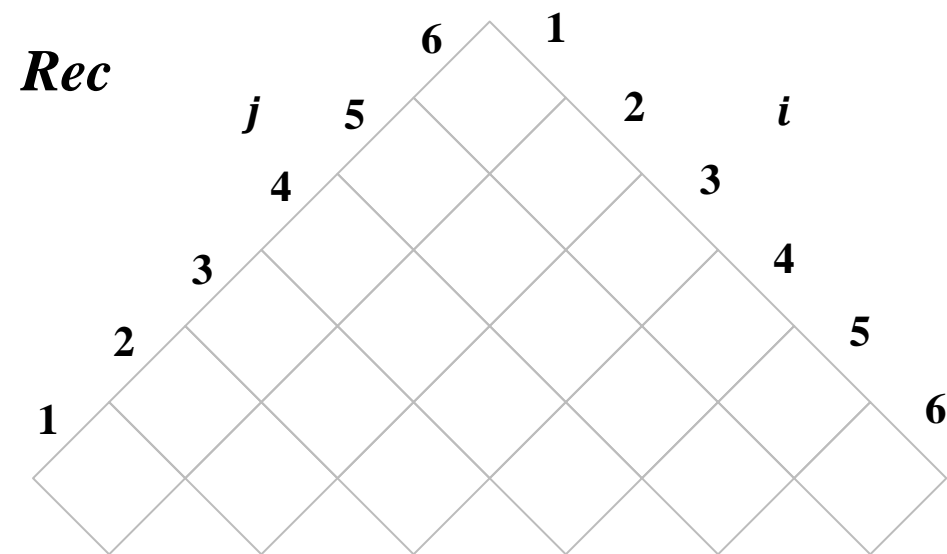
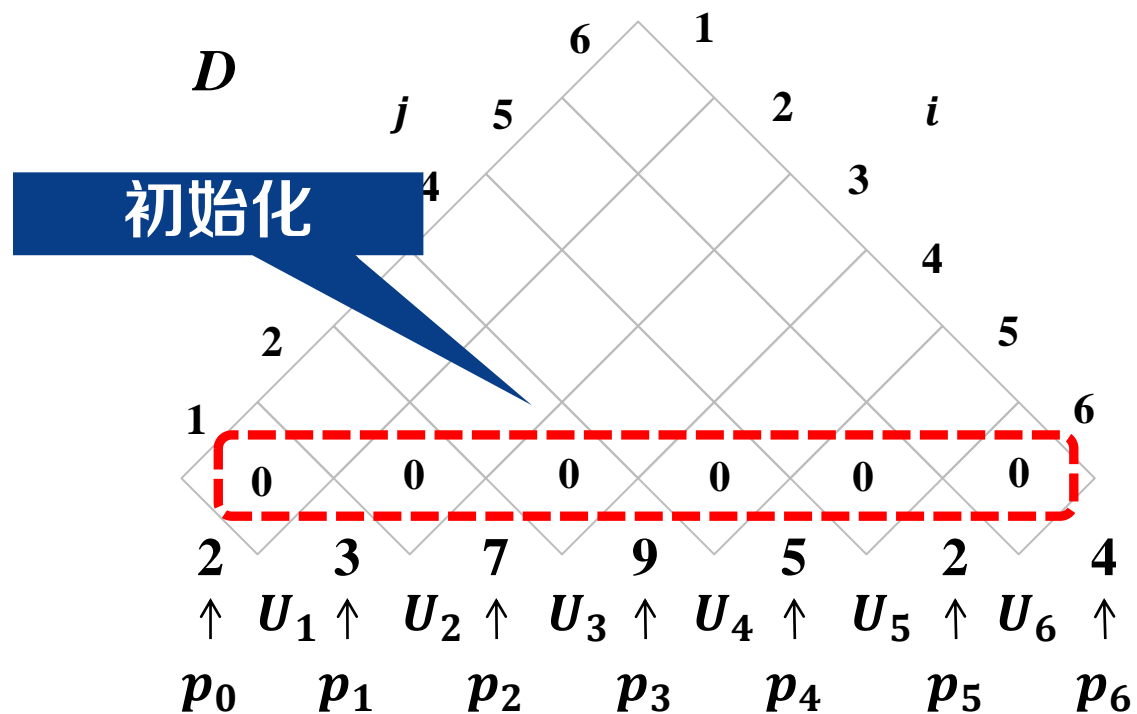
最优方案追踪

算法实例



- 给定矩阵链: $U_1 U_2 U_3 U_4 U_5 U_6$
- 对应行列数

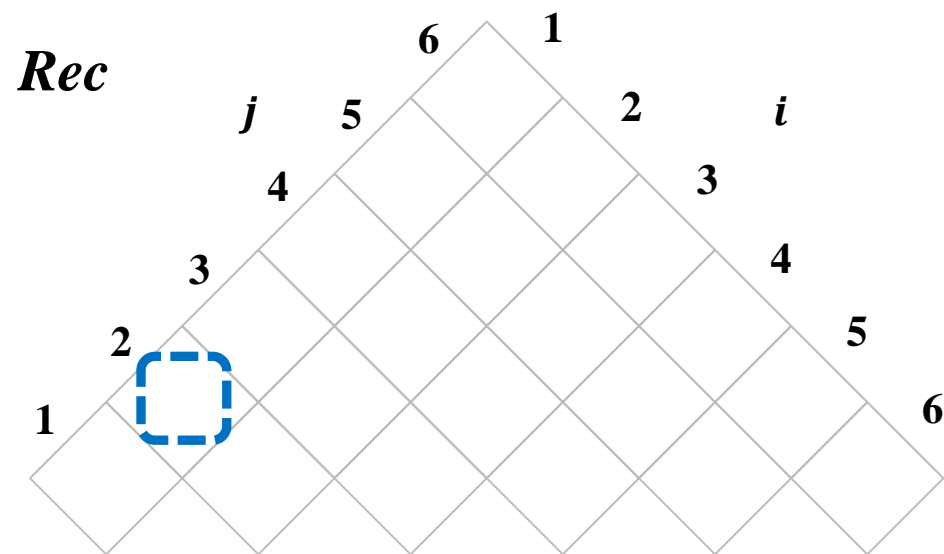
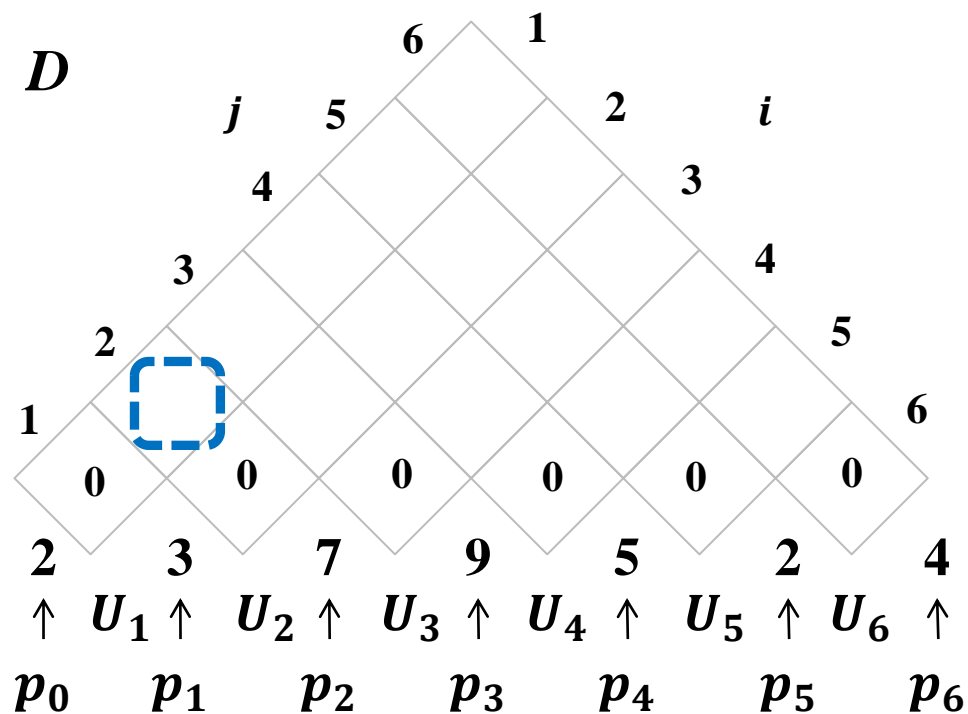
p_0	p_1	p_2	p_3	p_4	p_5	p_6
2	3	7	9	5	2	4



算法实例



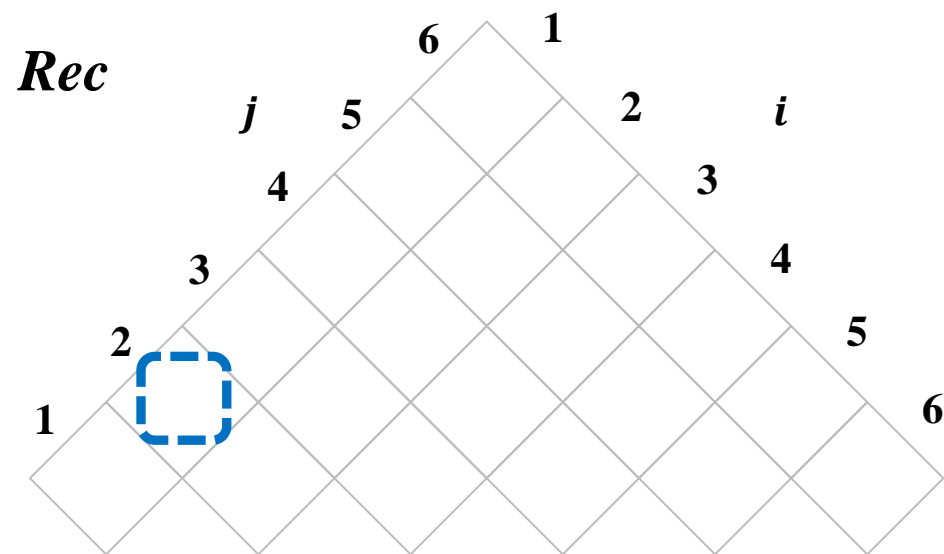
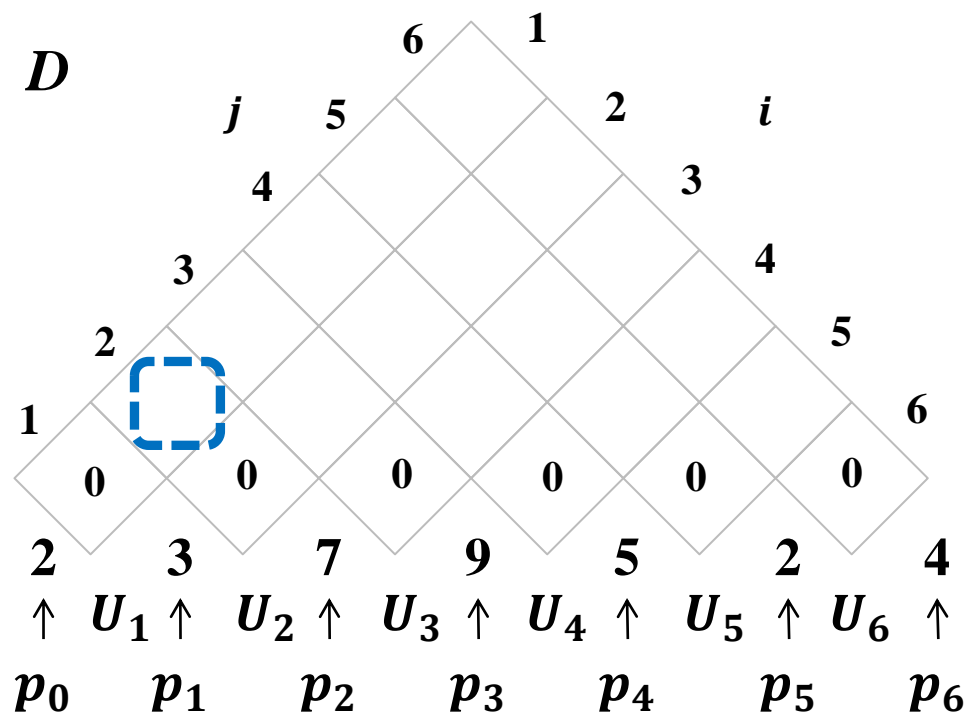
- $D[1, 2] = \min_{1 \leq k < 2} (D[1, k] + D[k + 1, 2] + p_0 p_k p_2)$



算法实例



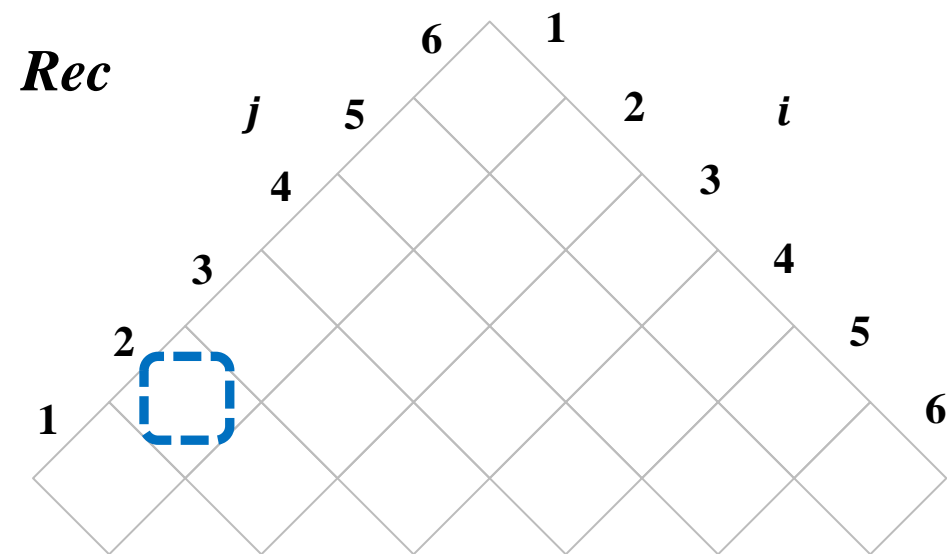
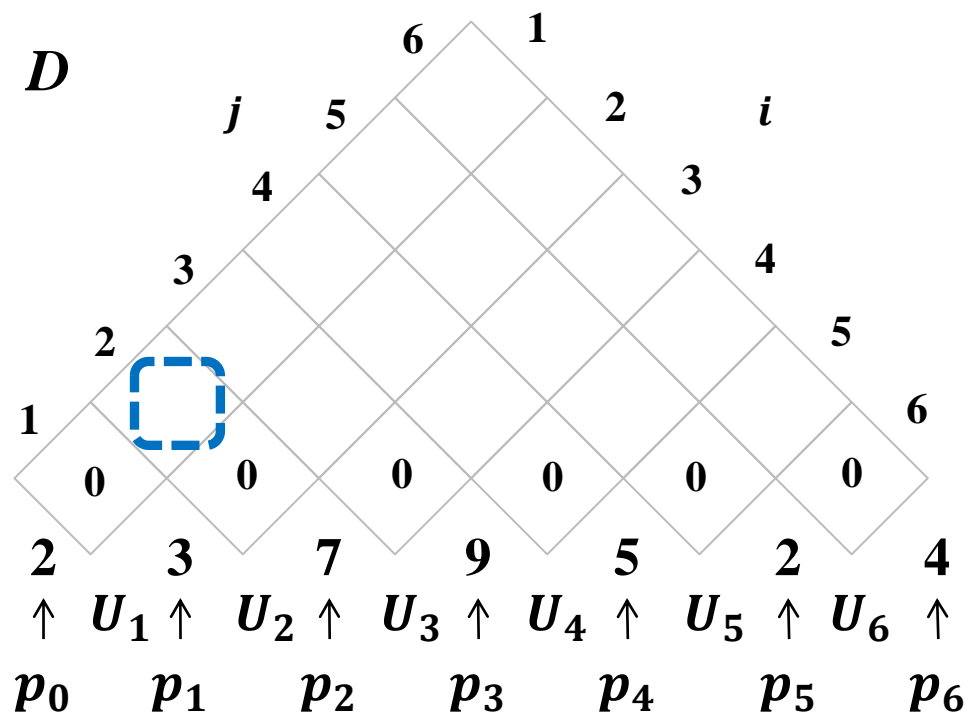
- $D[1, 2] = \min_{1 \leq k < 2} (D[1, k] + D[k + 1, 2] + p_0 p_k p_2)$



算法实例



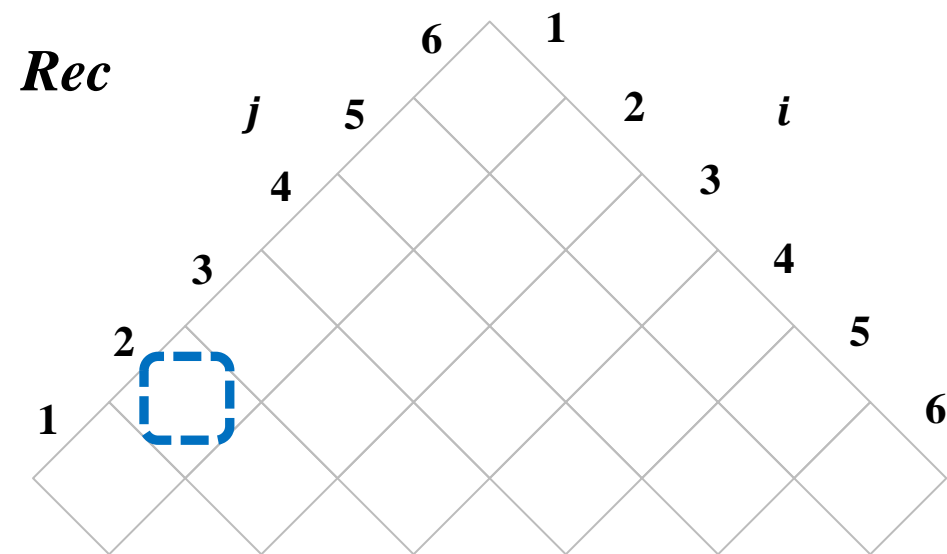
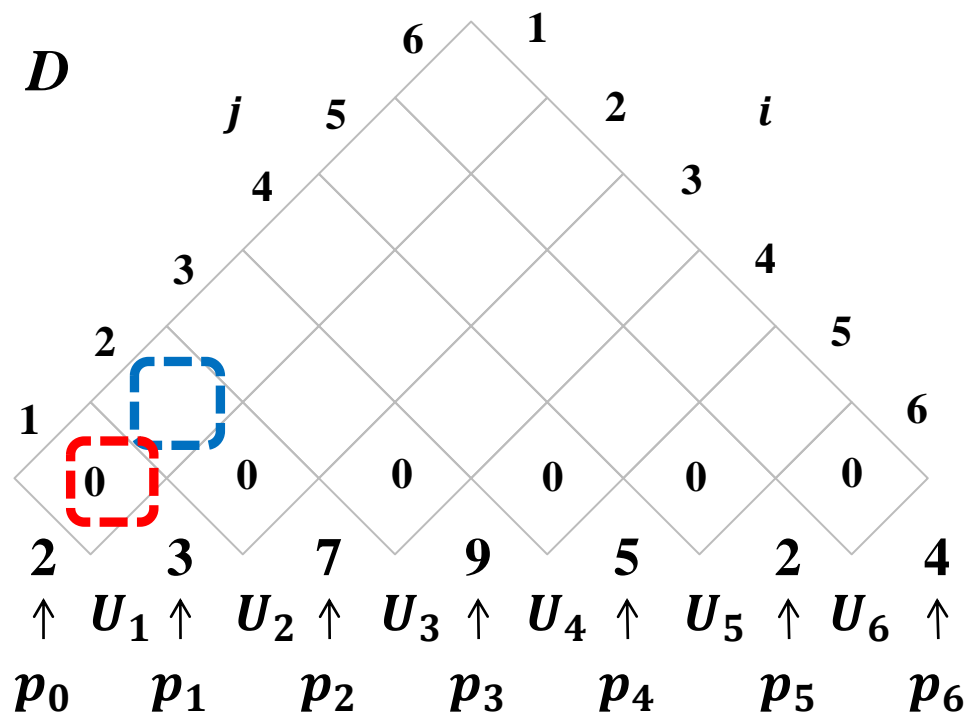
- $$D[1, 2] = \min_{1 \leq k < 2} (D[1, 1] + D[1 + 1, 2] + p_0 p_1 p_2)$$



算法实例



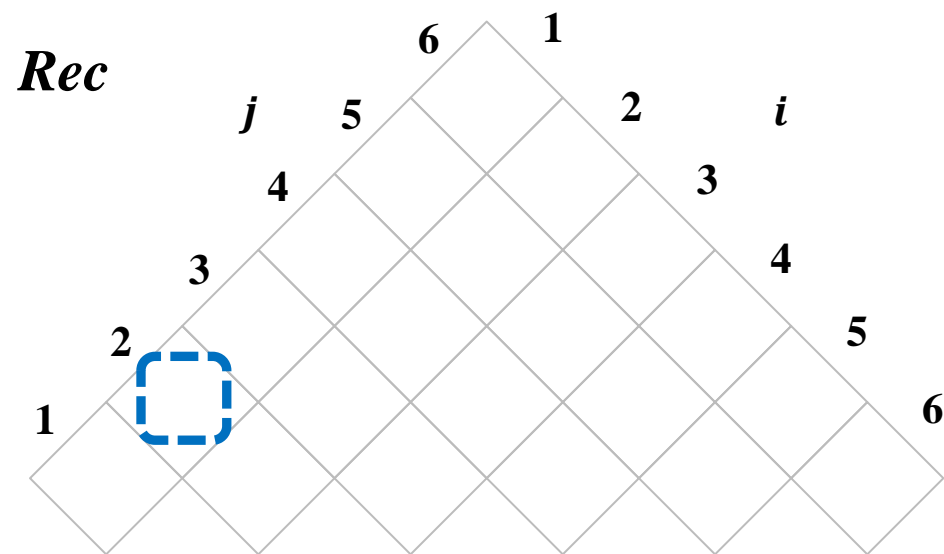
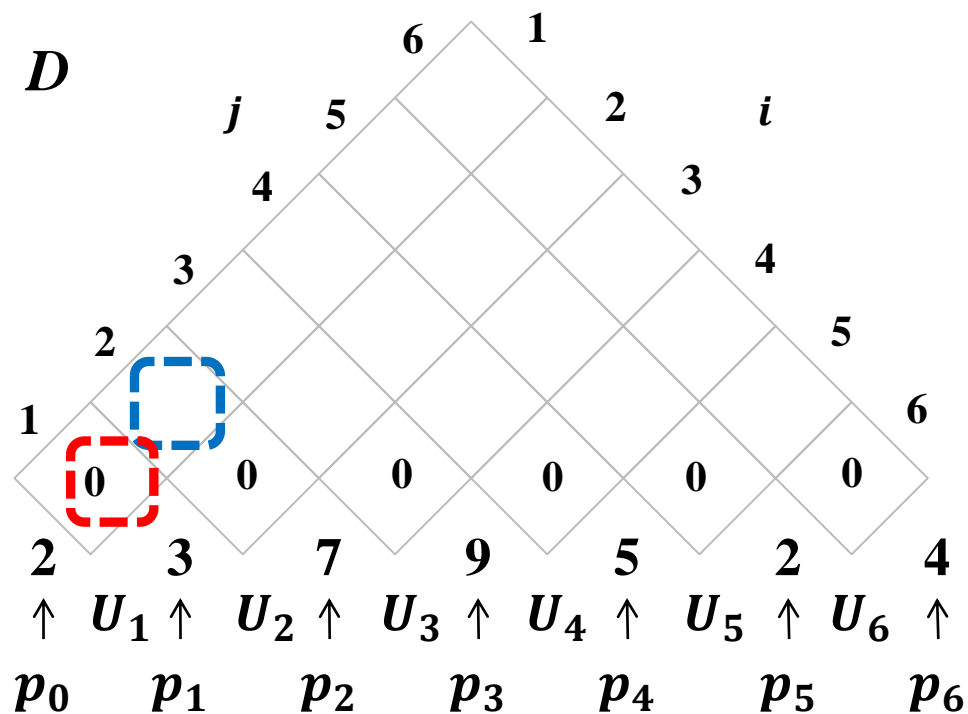
- $$D[1, 2] = \min_{1 \leq k < 2} (\textcolor{red}{D[1, 1]} + D[1 + 1, 2] + p_0 p_1 p_2)$$



算法实例



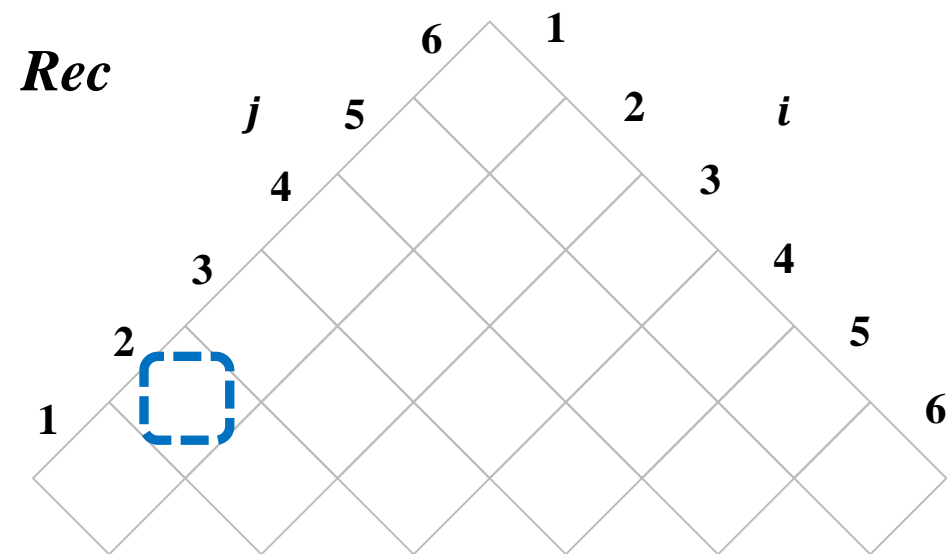
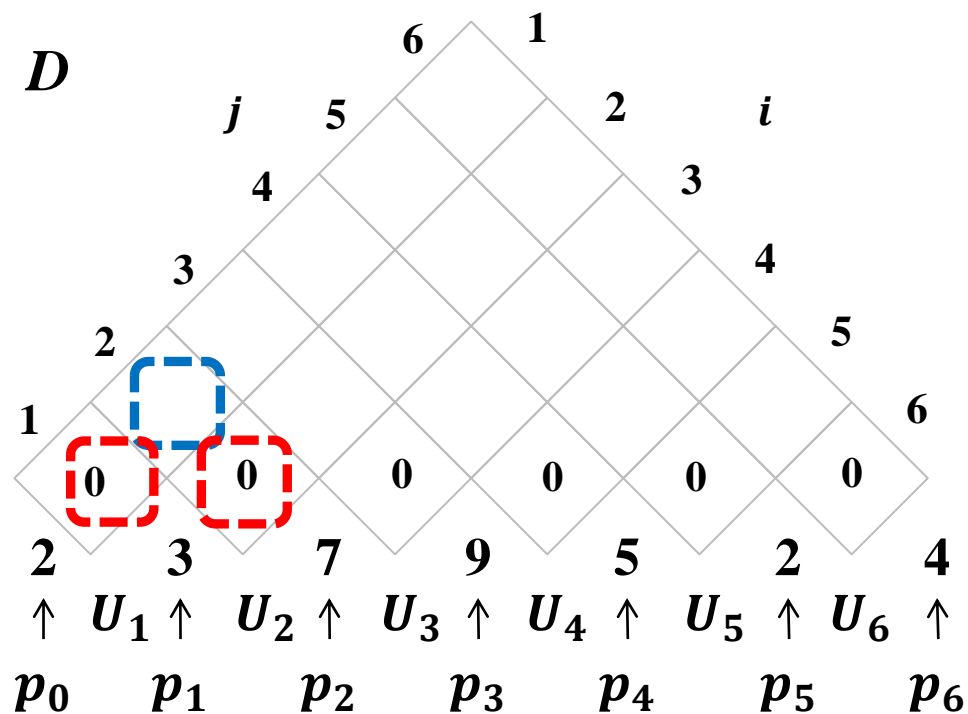
- $D[1, 2] = \min_{1 \leq k < 2} (0 + D[1 + 1, 2] + p_0 p_1 p_2)$



算法实例



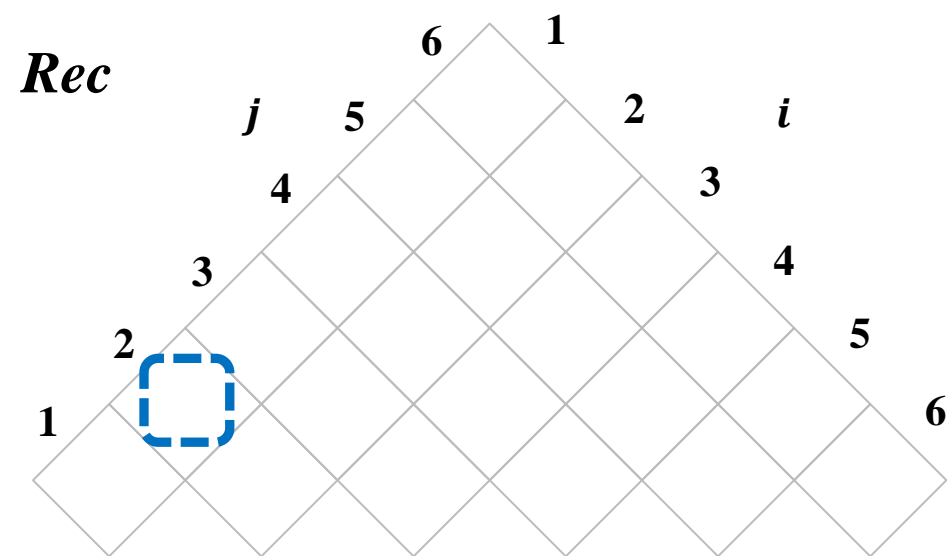
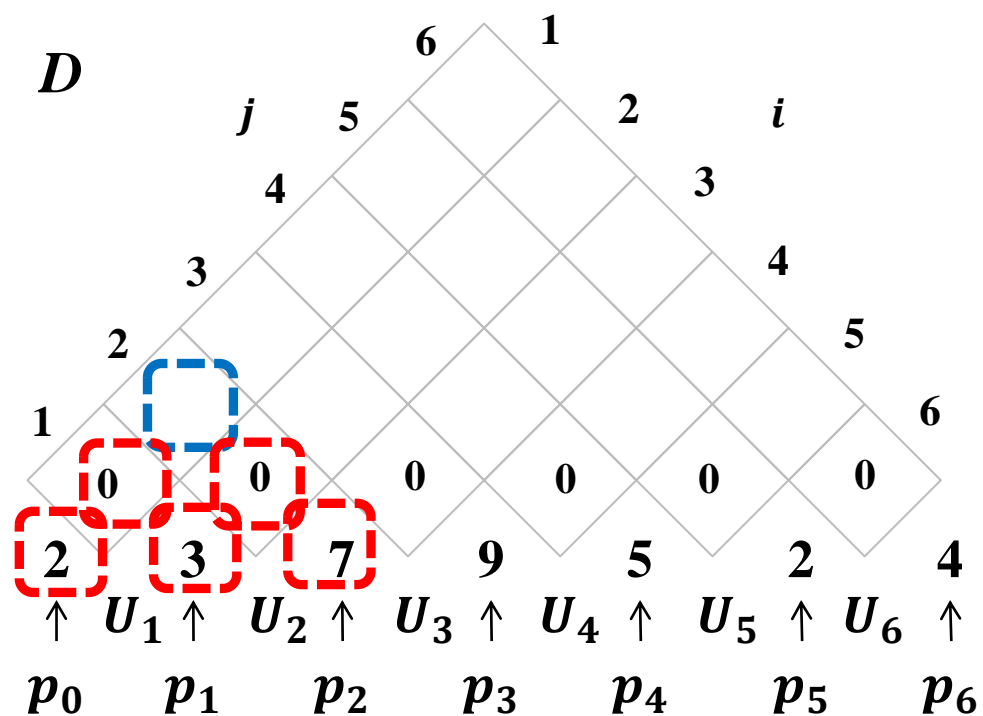
- $$D[1, 2] = \min_{1 \leq k < 2} (0 + \mathbf{D[1 + 1, 2]} + p_0 p_1 p_2)$$



算法实例



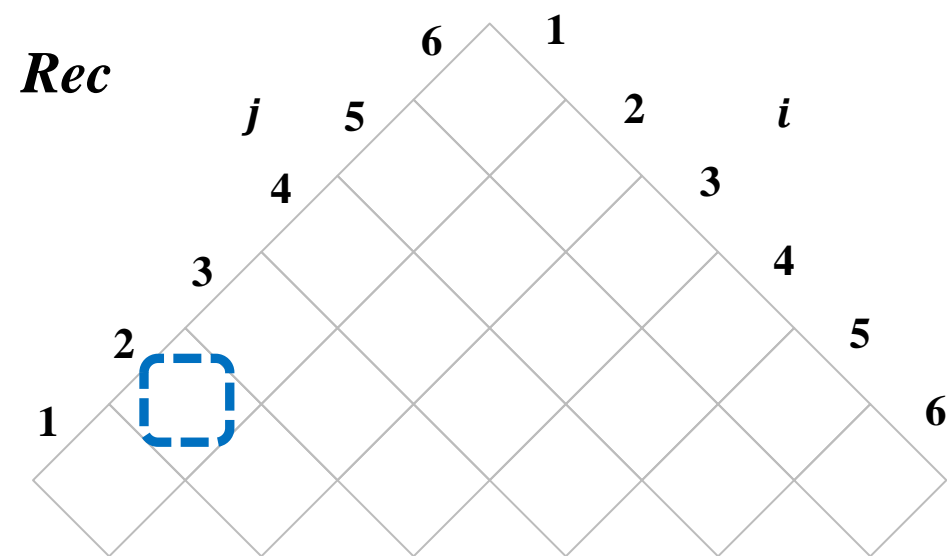
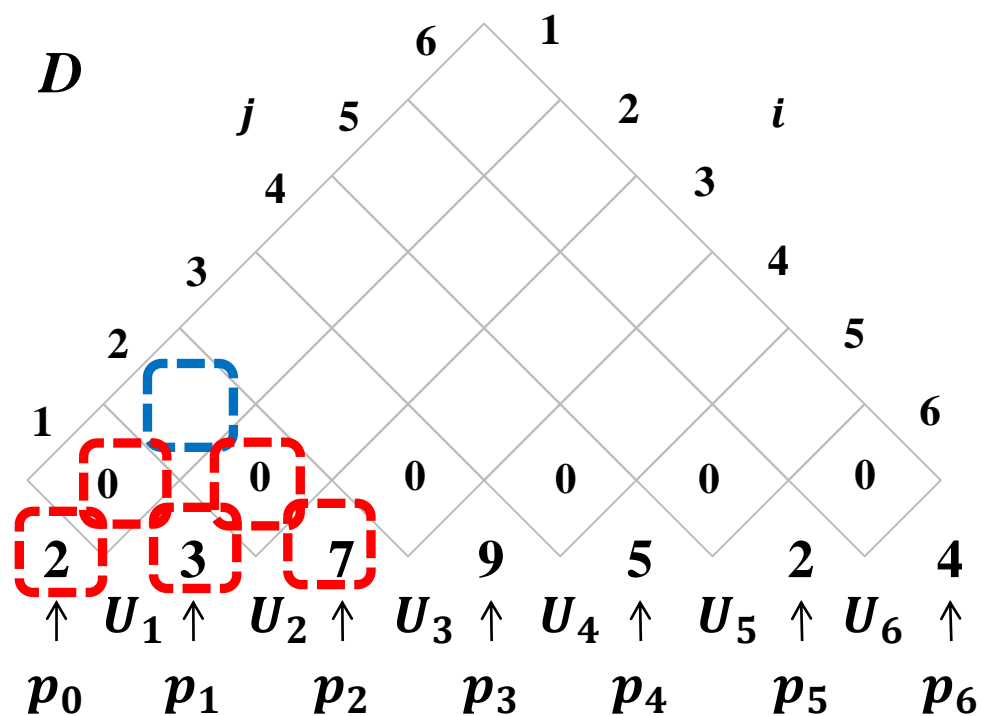
- $D[1, 2] = \min_{1 \leq k < 2} (0 + 0 + p_0 p_1 p_2)$



算法实例



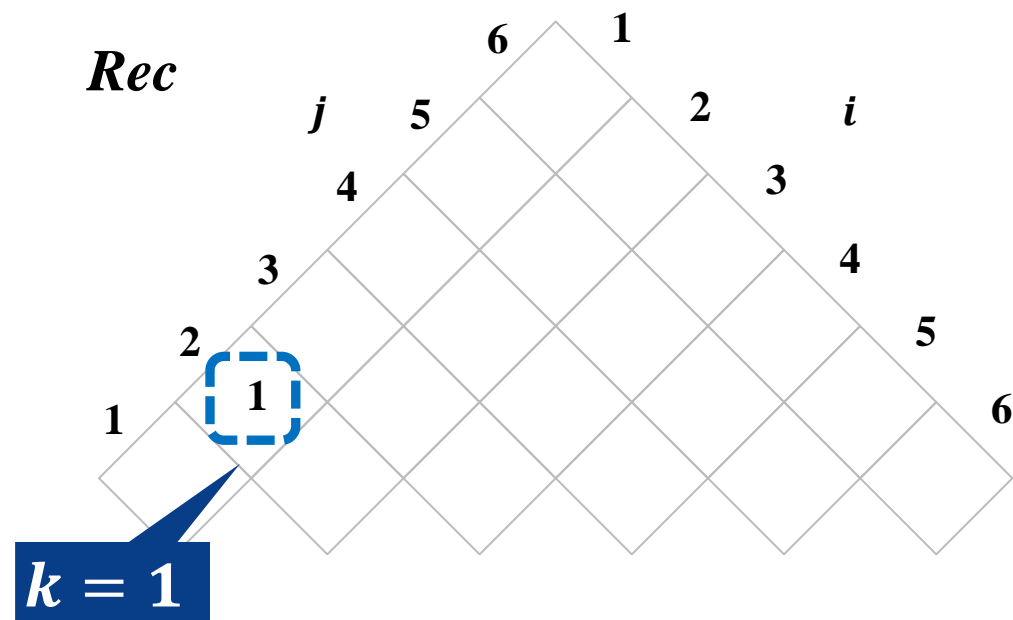
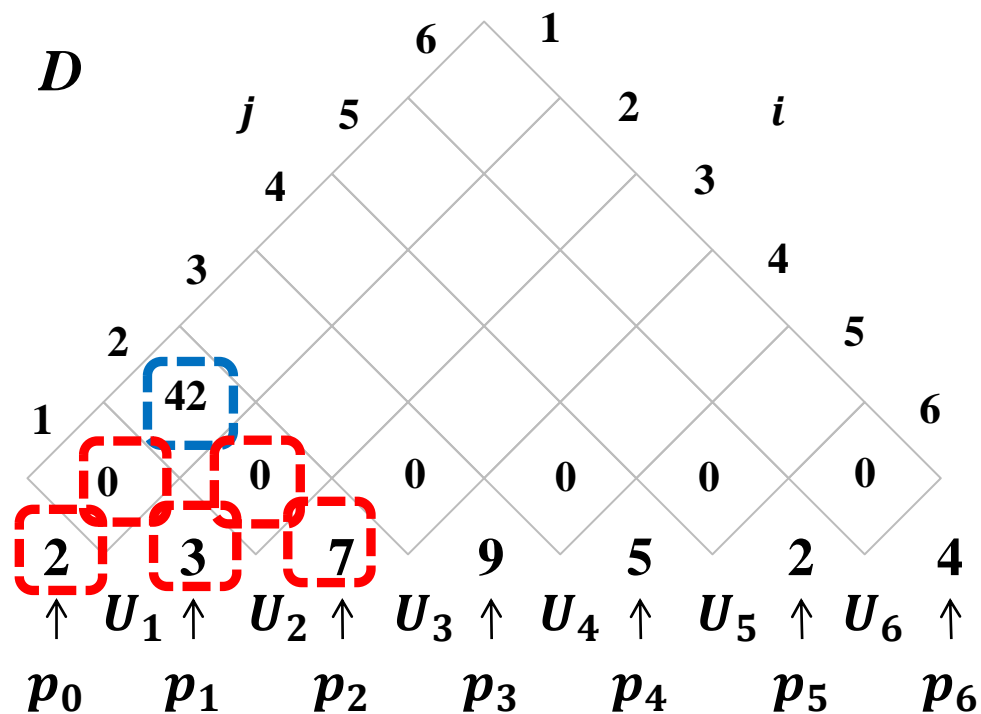
- $D[1, 2] = \min_{1 \leq k < 2} (0 + 0 + 2 \times 3 \times 7)$



算法实例



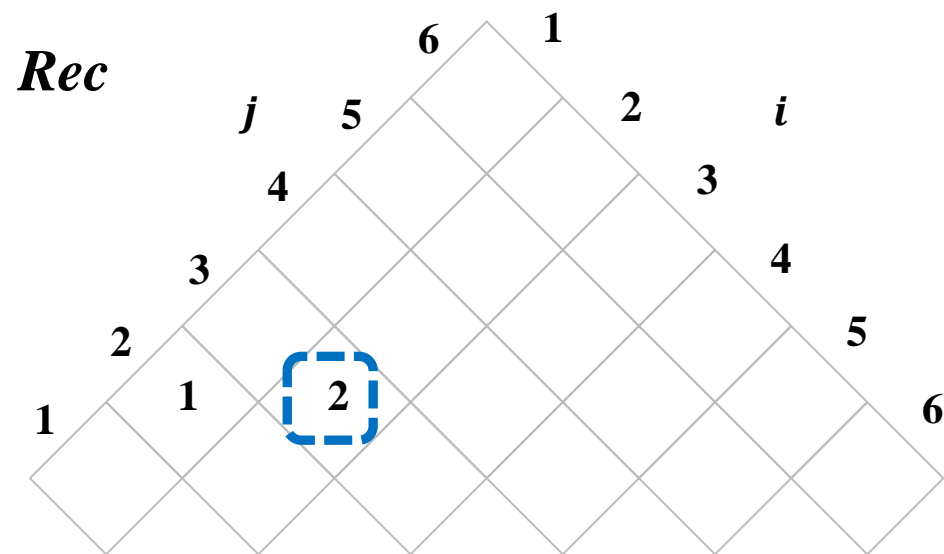
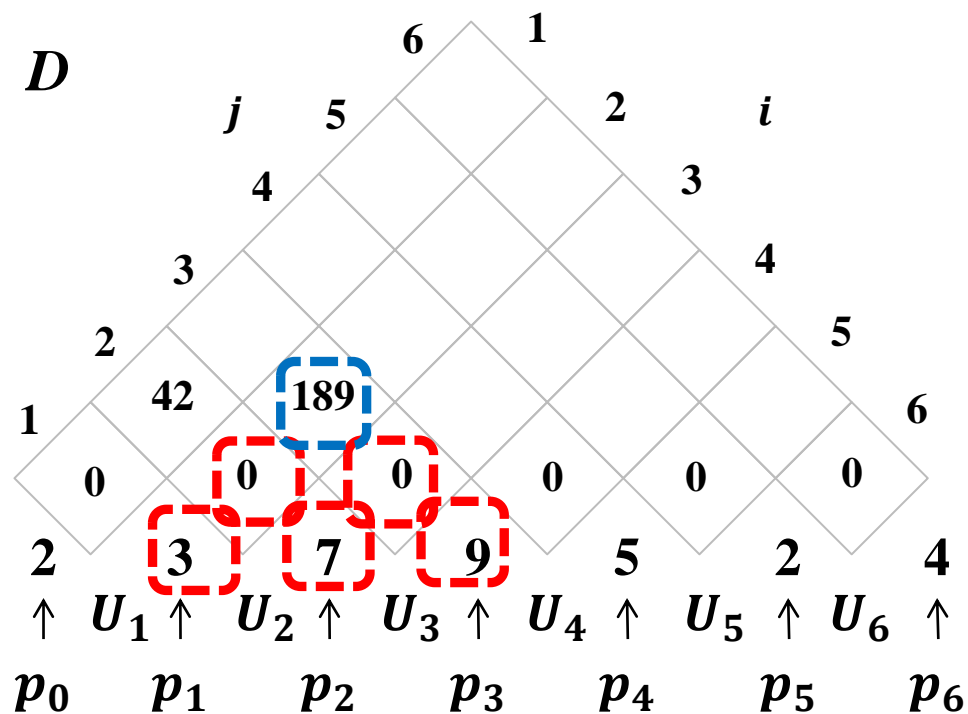
- $D[1, 2] = \min_{1 \leq k < 2} (0 + 0 + \mathbf{2 \times 3 \times 7}) = 42$



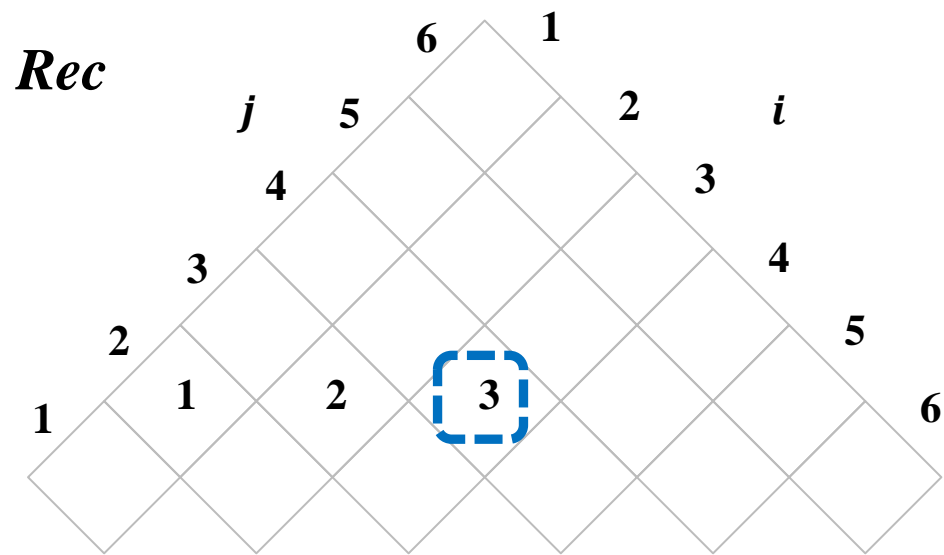
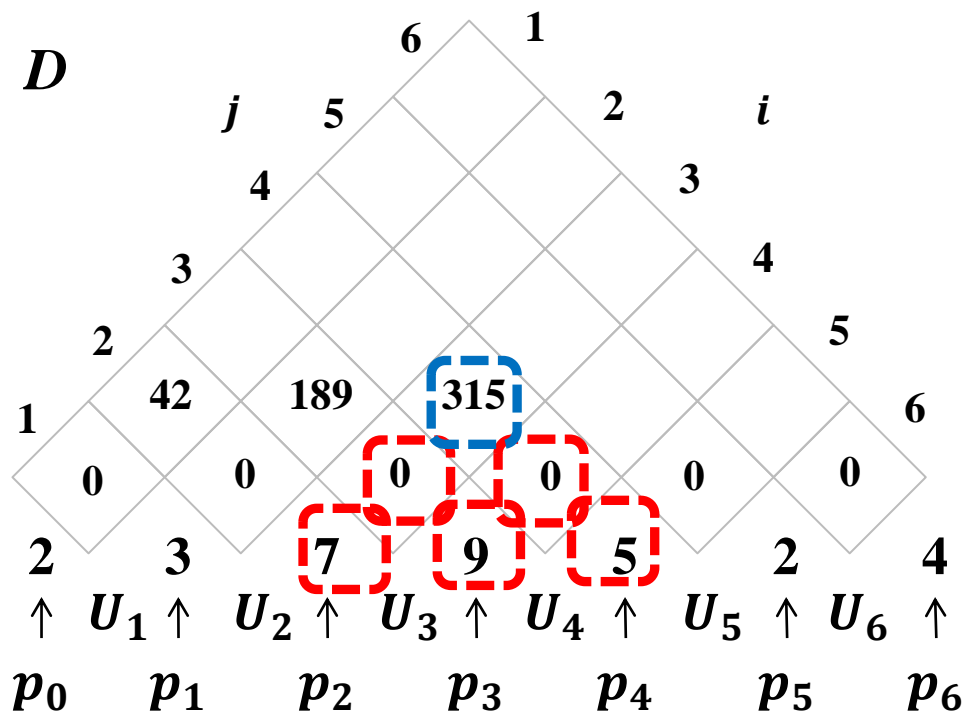
算法实例



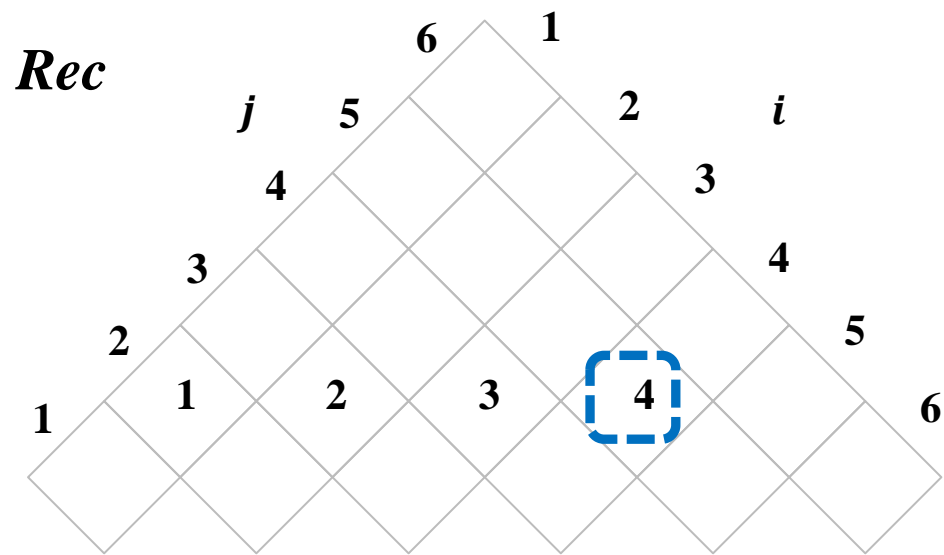
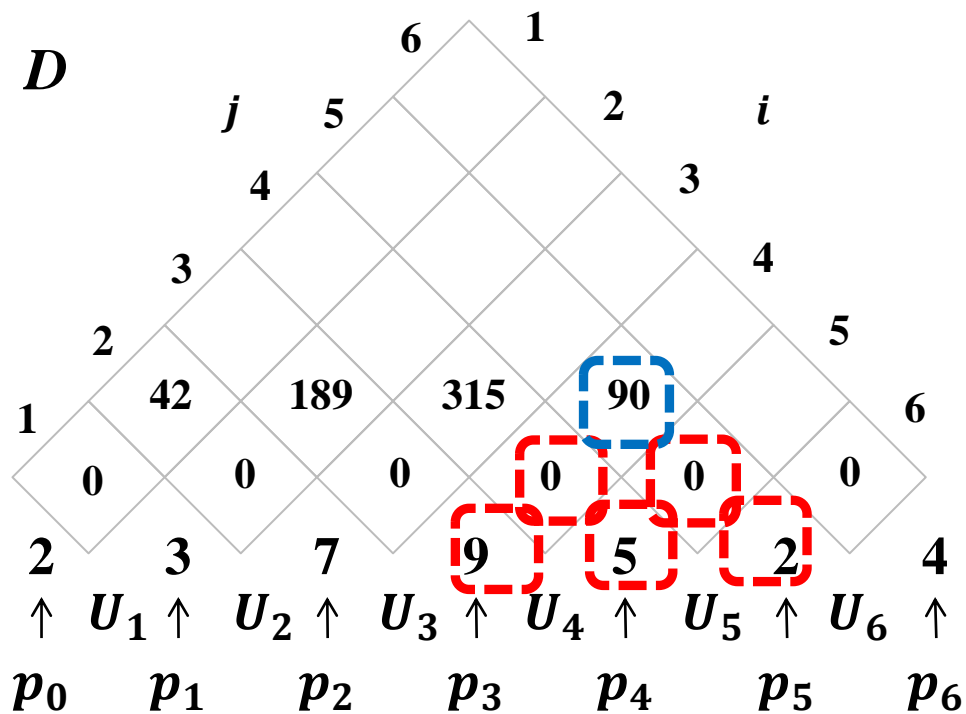
- $D[2, 3] = \min_{2 \leq k < 3} (D[2, k] + D[k + 1, 3] + p_1 p_k p_3) = 189$



- $D[3, 4] = \min_{3 \leq k < 4} (D[3, k] + D[k + 1, 4] + p_2 p_k p_4) = 315$



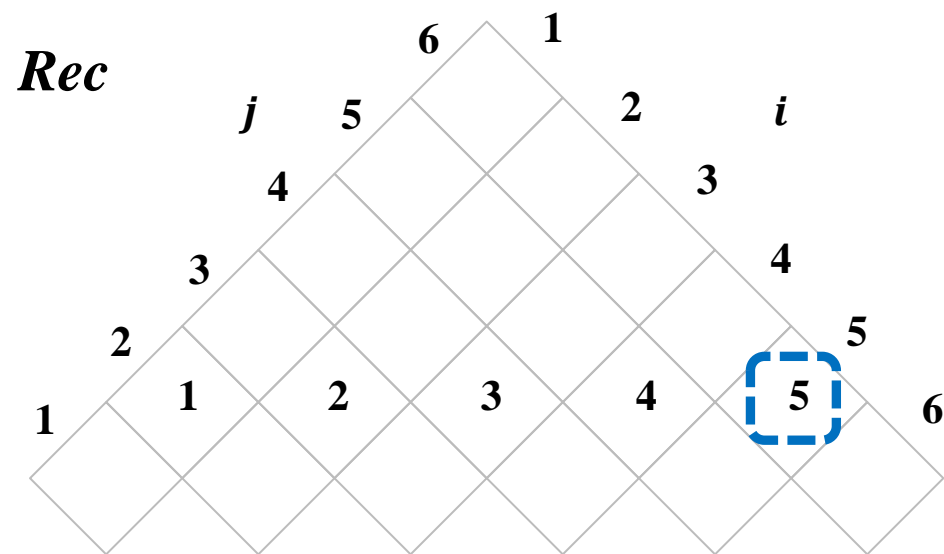
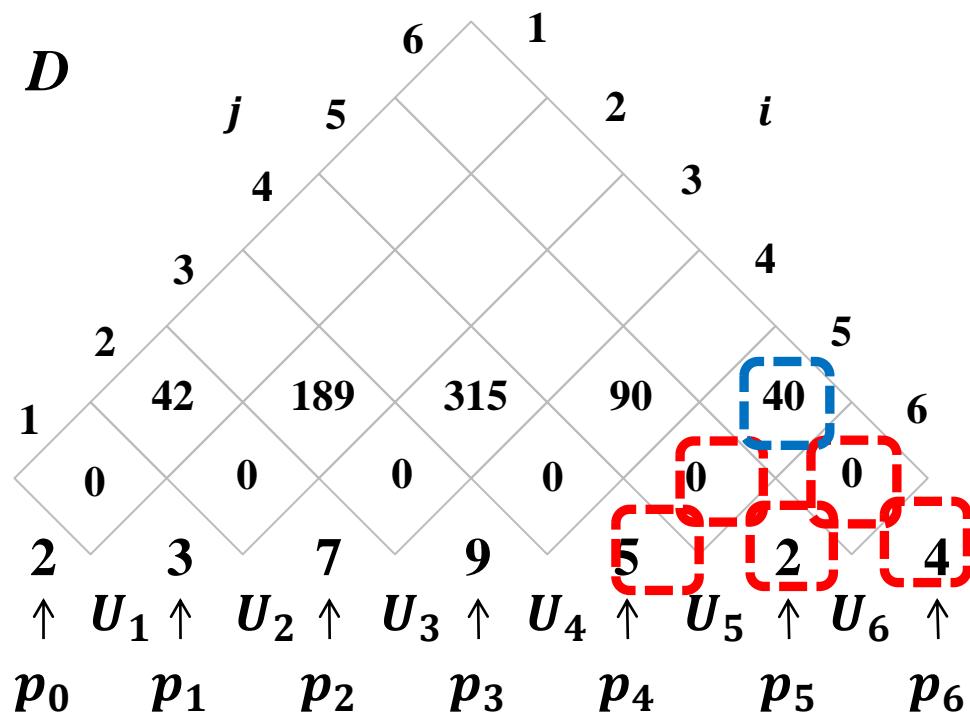
- $D[4, 5] = \min_{4 \leq k < 5} (D[4, k] + D[k + 1, 5] + p_3 p_k p_5) = 90$



算法实例

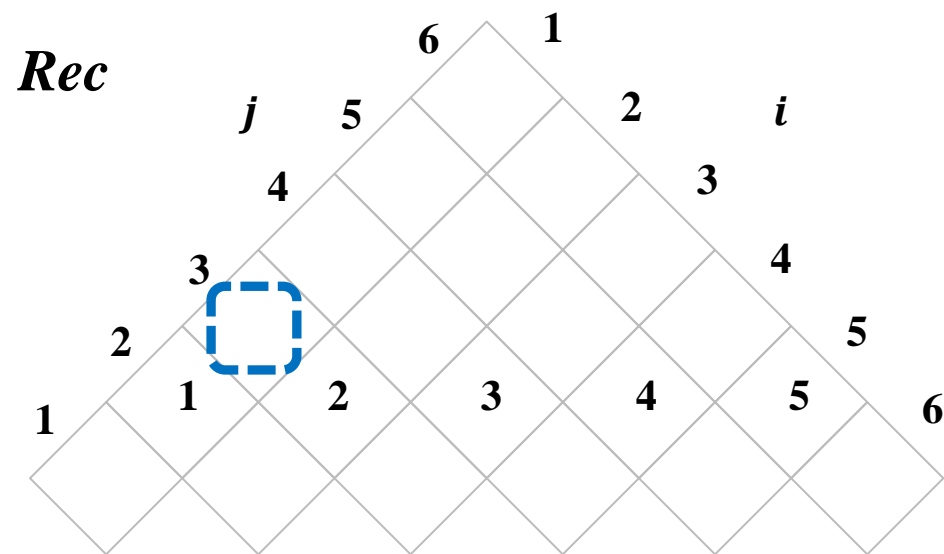
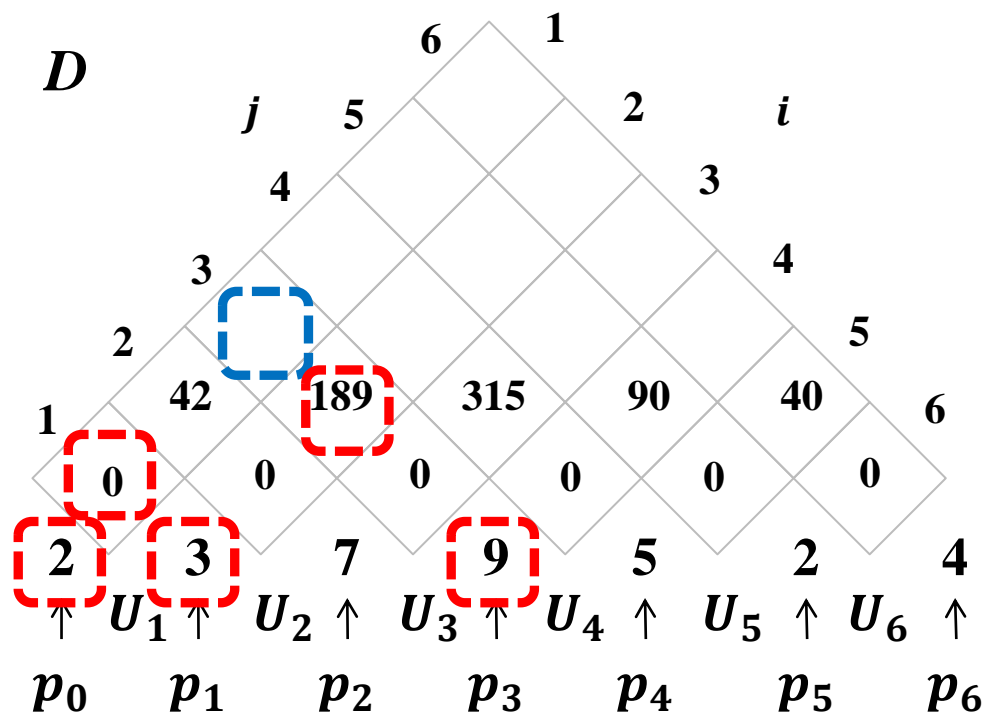


- $$D[5, 6] = \min_{5 \leq k < 6} (D[5, k] + D[k + 1, 6] + p_4 p_k p_6) = 40$$



- $$D[1, 3] = \min_{1 \leq k < 3} (D[1, k] + D[k + 1, 3] + p_0 p_k p_3)$$

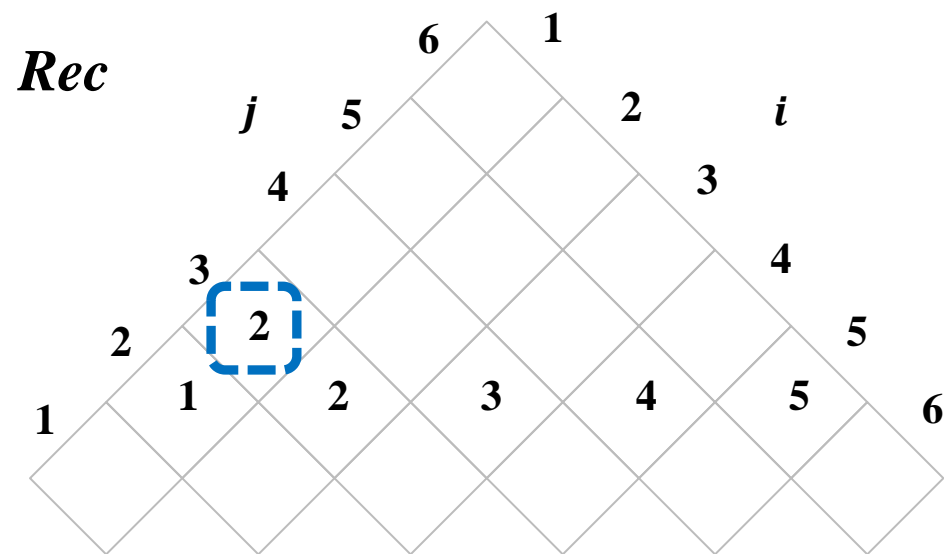
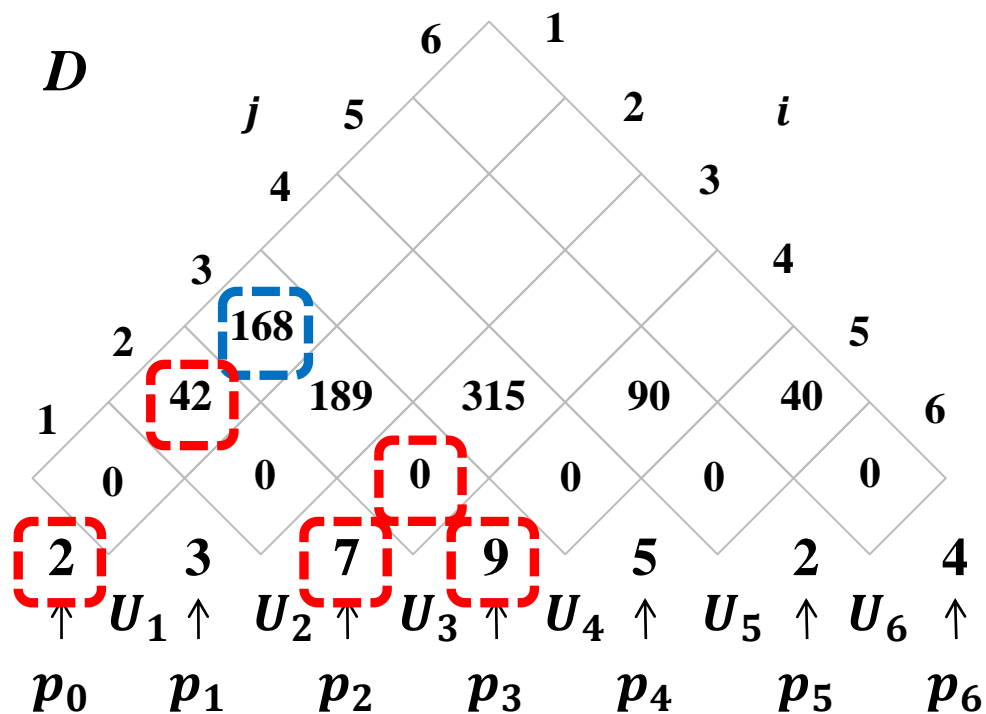
$$= \min \begin{cases} D[1, 1] + D[2, 3] + p_0 p_1 p_3 = 243 \\ D[1, 2] + D[3, 3] + p_0 p_2 p_3 = \end{cases}$$



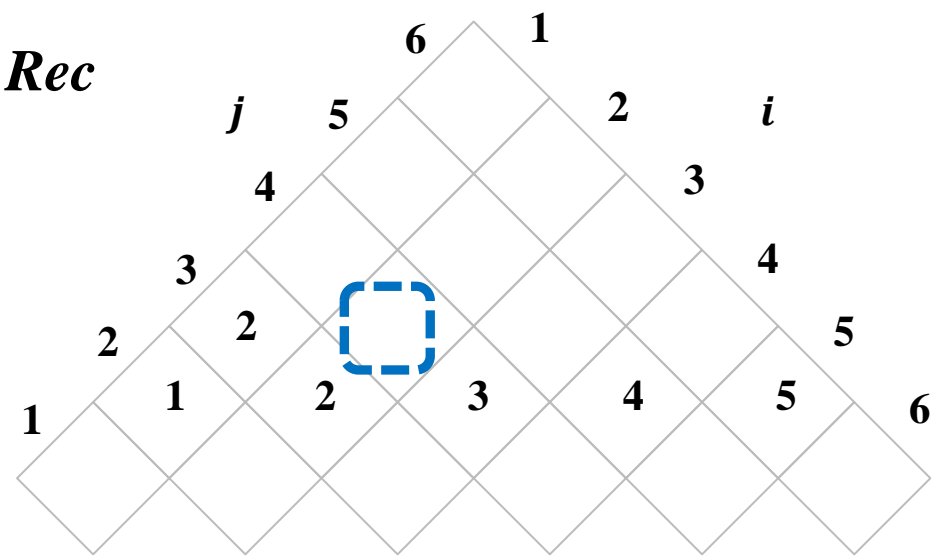
算法实例



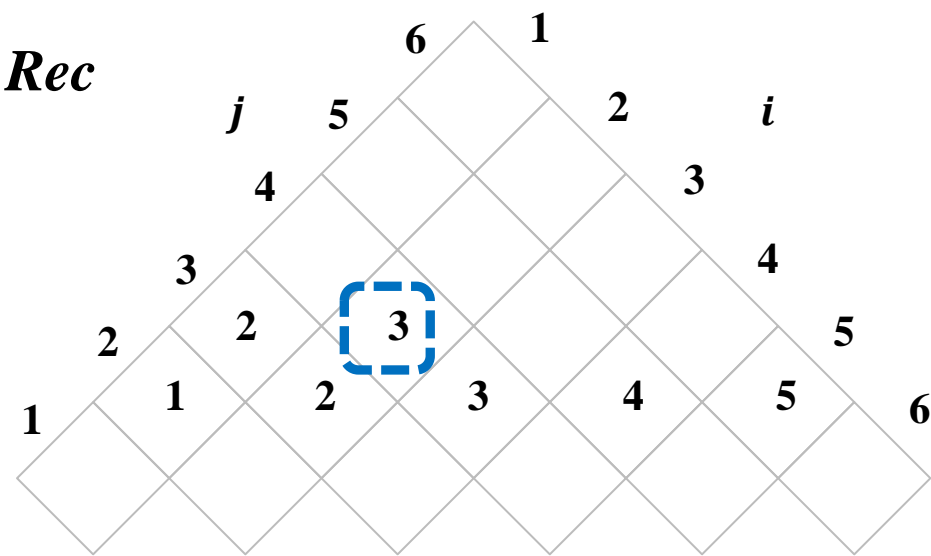
- $$D[1, 3] = \min_{1 \leq k < 3} (D[1, k] + D[k + 1, 3] + p_0 p_k p_3)$$
$$= \min \begin{cases} D[1, 1] + D[2, 3] + p_0 p_1 p_3 = 243 \\ D[1, 2] + D[3, 3] + p_0 p_2 p_3 = \mathbf{168} \end{cases}$$



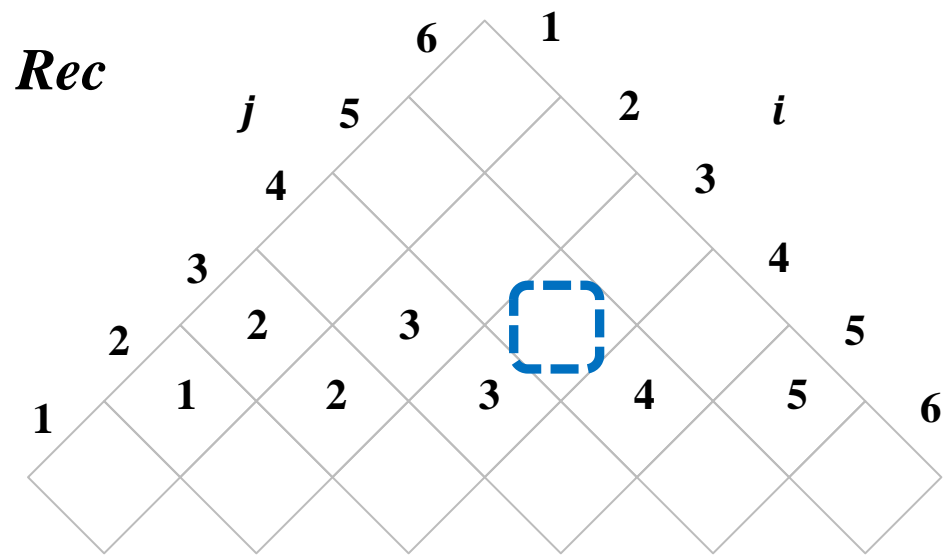
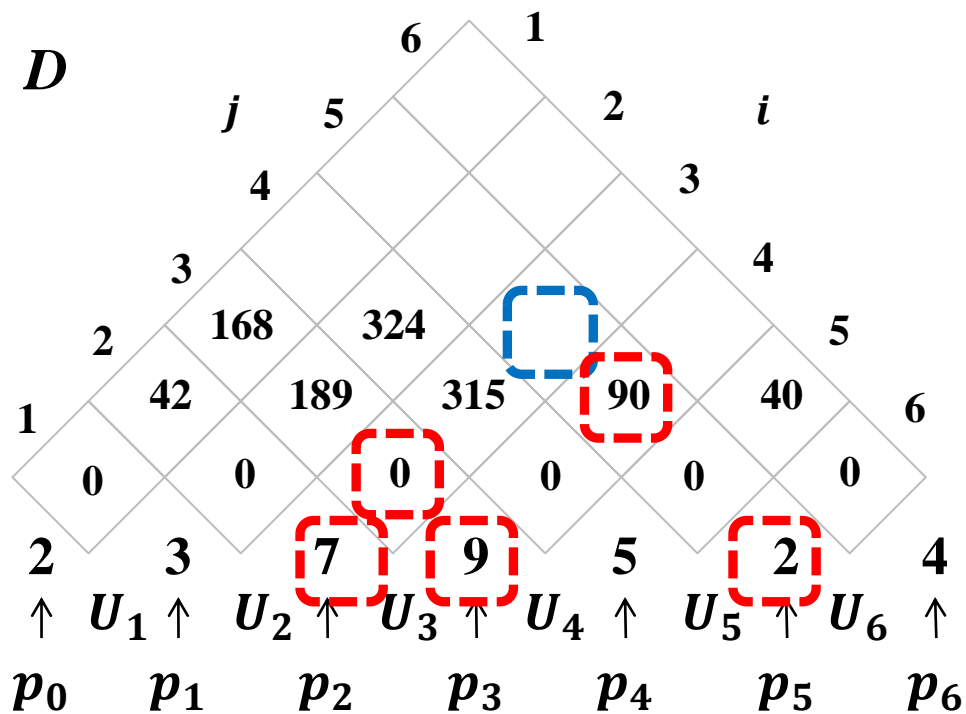
$$= \min \begin{cases} D[2, 2] + D[3, 4] + p_1 p_2 p_4 = 420 \\ D[2, 3] + D[4, 4] + p_1 p_3 p_4 = \end{cases}$$



$$= \min \begin{cases} D[2, 2] + D[3, 4] + p_1 p_2 p_4 = 420 \\ D[2, 3] + D[4, 4] + p_1 p_3 p_4 = \mathbf{324} \end{cases}$$



- $D[3, 5] = \min_{3 \leq k < 5} (D[3, k] + D[k + 1, 5] + p_2 p_k p_5)$
 $= \min \begin{cases} D[3, 3] + D[4, 5] + p_2 p_3 p_5 = 216 \\ D[3, 4] + D[5, 5] + p_2 p_4 p_5 = \end{cases}$

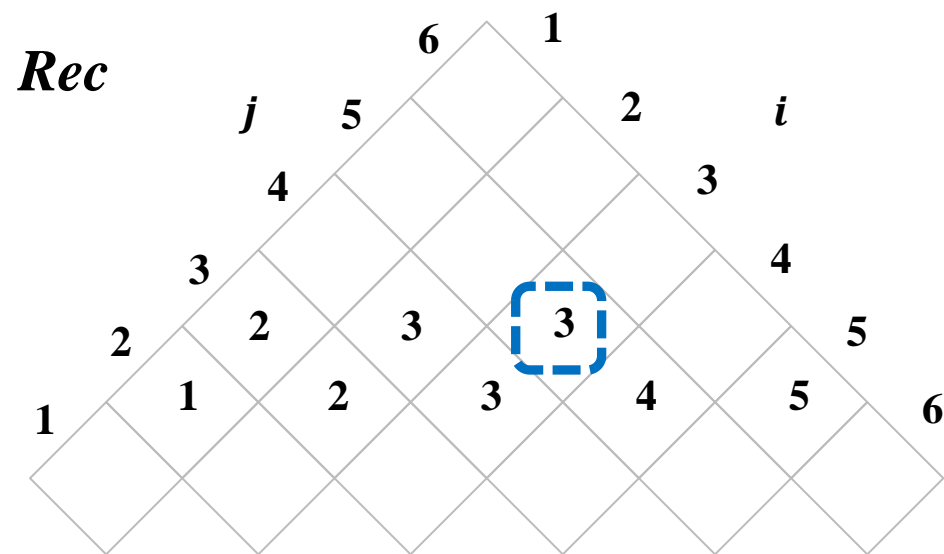
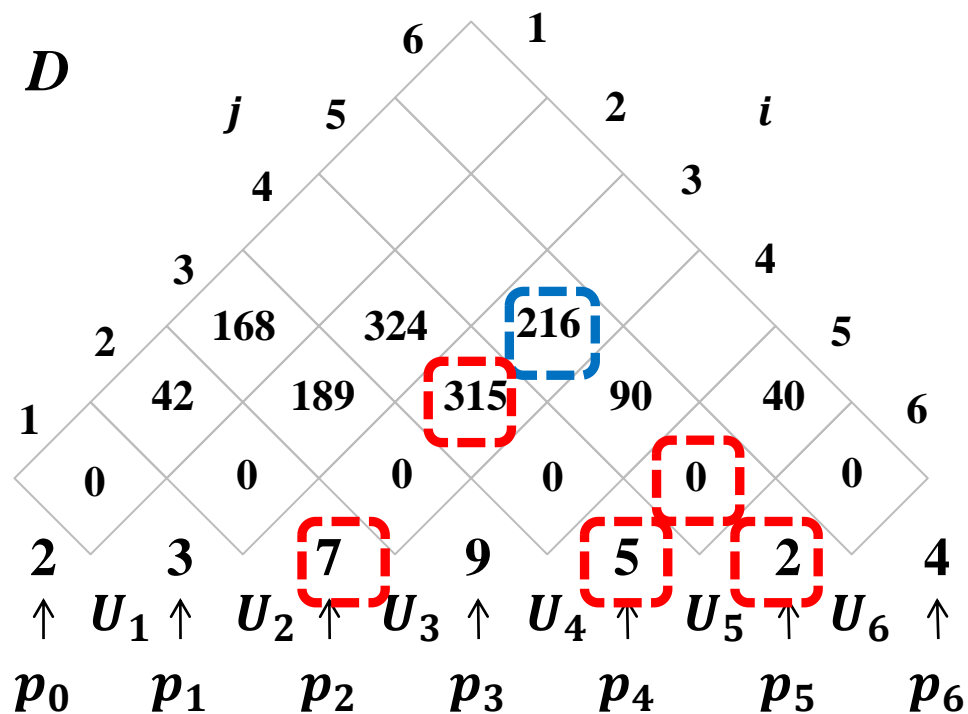


算法实例



- $$D[3, 5] = \min_{3 \leq k < 5} (D[3, k] + D[k + 1, 5] + p_2 p_k p_5)$$

$$= \min \begin{cases} D[3, 3] + D[4, 5] + p_2 p_3 p_5 = \mathbf{216} \\ D[3, 4] + D[5, 5] + p_2 p_4 p_5 = 385 \end{cases}$$

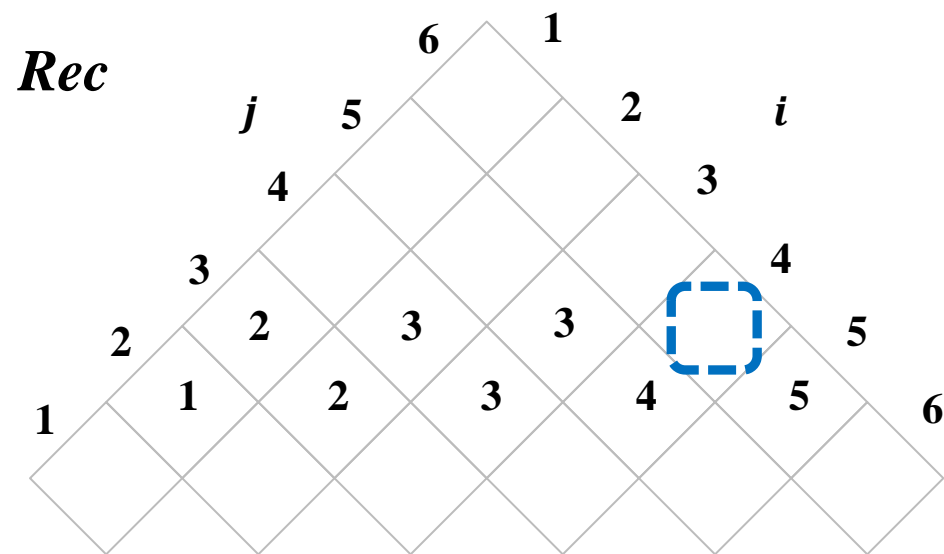
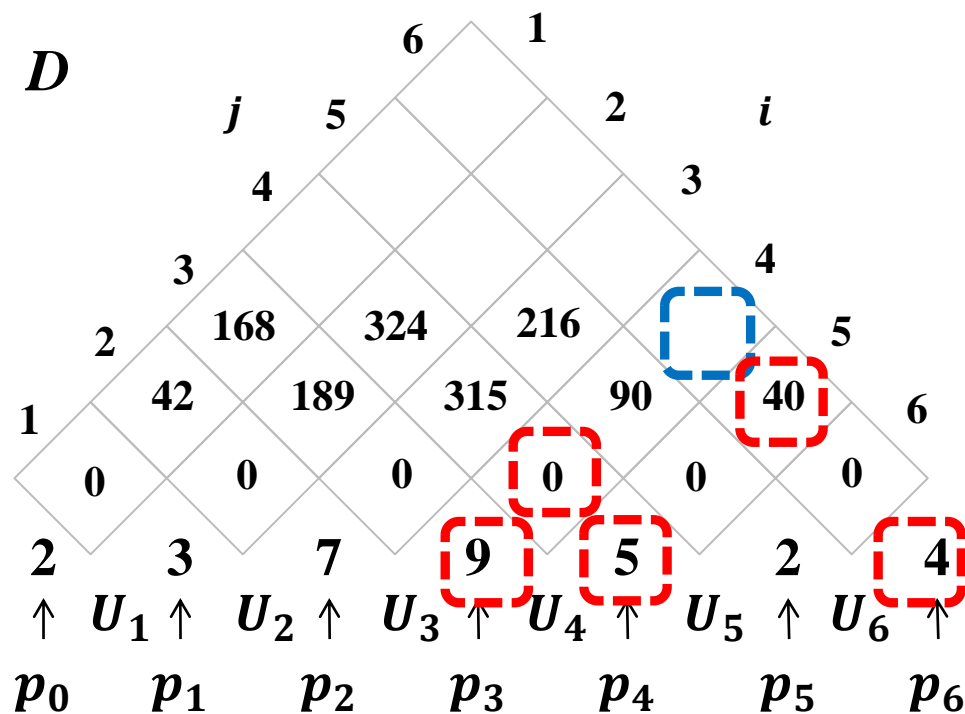


算法实例



- $$D[4, 6] = \min_{4 \leq k < 6} (D[4, k] + D[k + 1, 6] + p_3 p_k p_6)$$

$$= \min \begin{cases} D[4, 4] + D[5, 6] + p_3 p_4 p_6 = 220 \\ D[4, 5] + D[6, 6] + p_3 p_5 p_6 = \end{cases}$$

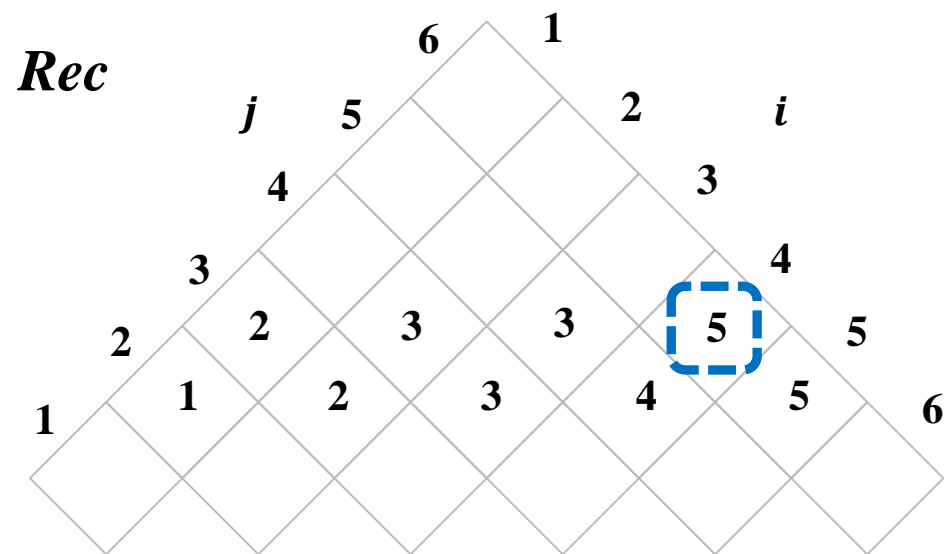
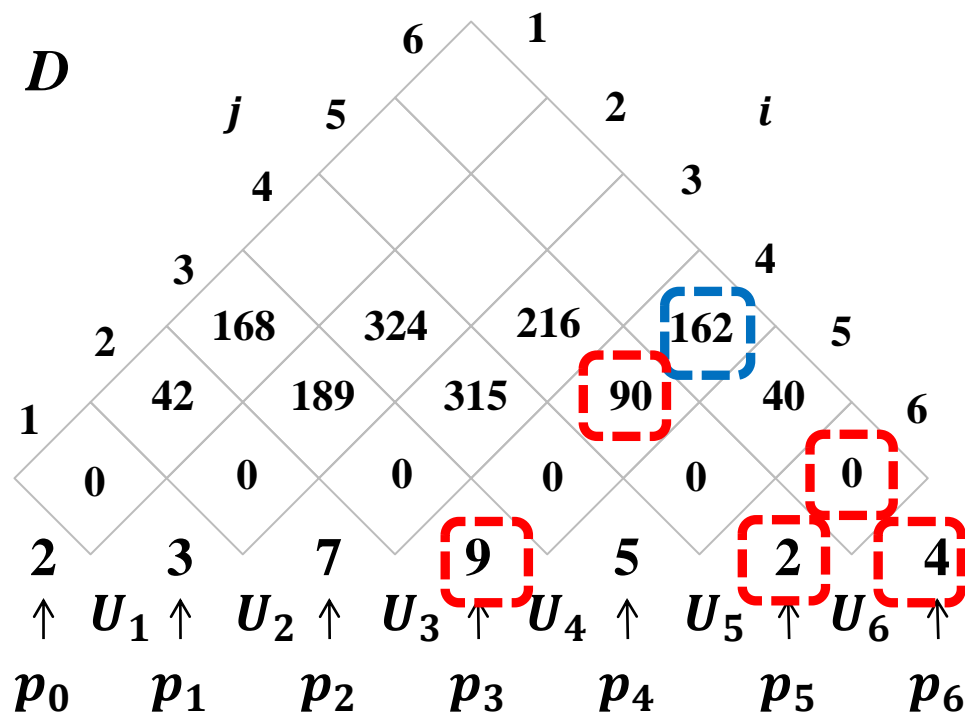


算法实例

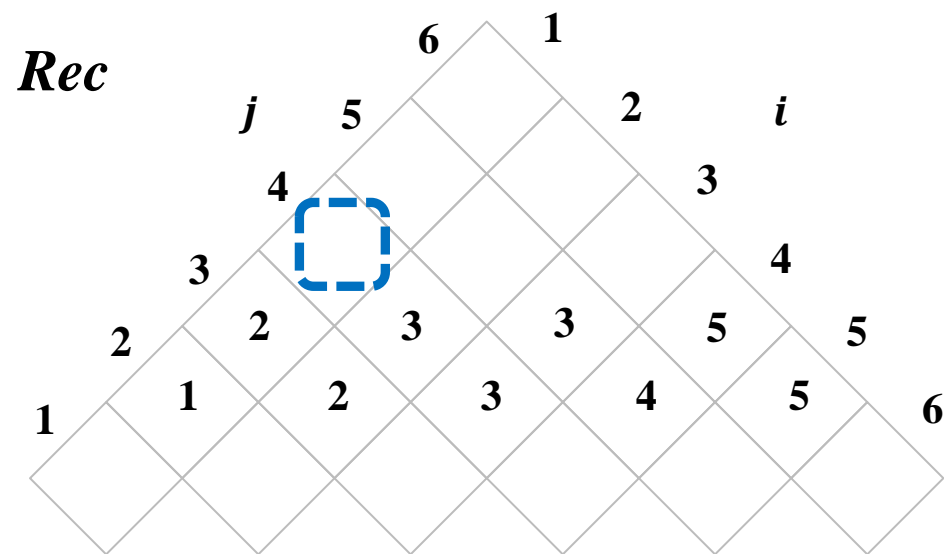
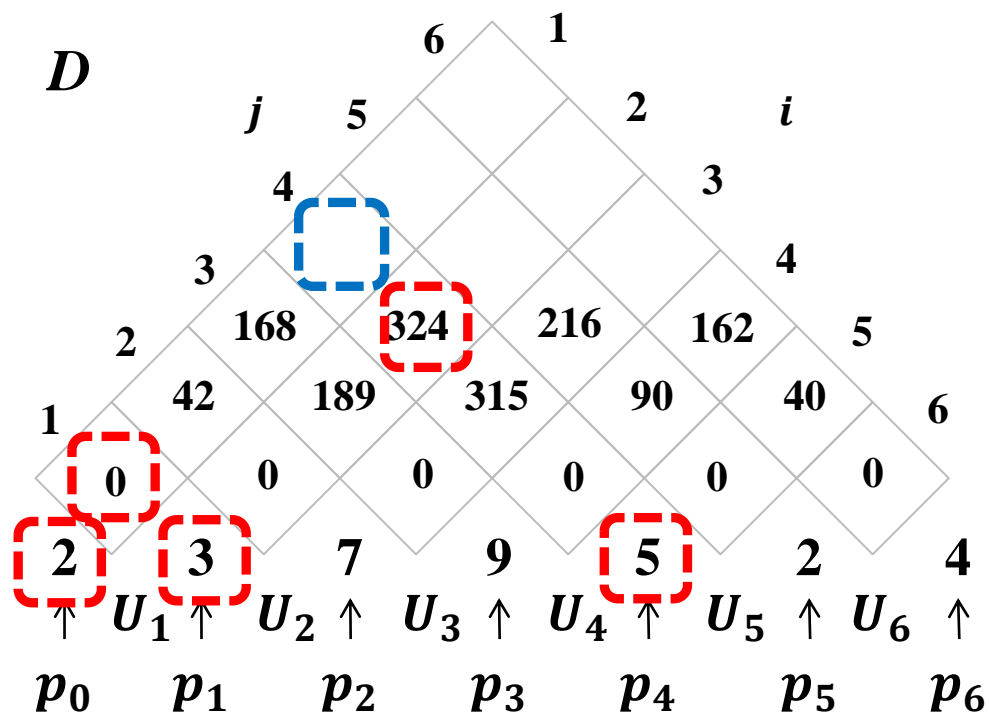


- $$D[4, 6] = \min_{4 \leq k < 6} (D[4, k] + D[k + 1, 6] + p_3 p_k p_6)$$

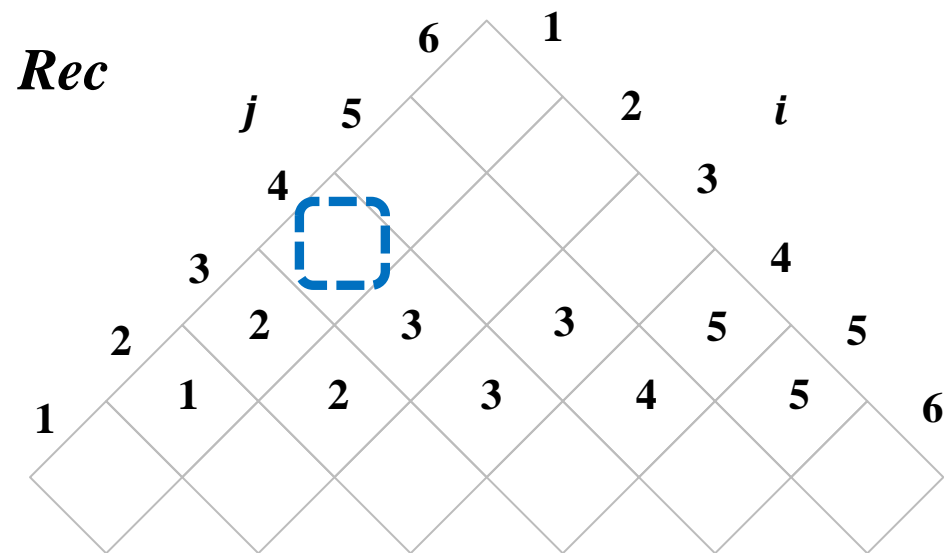
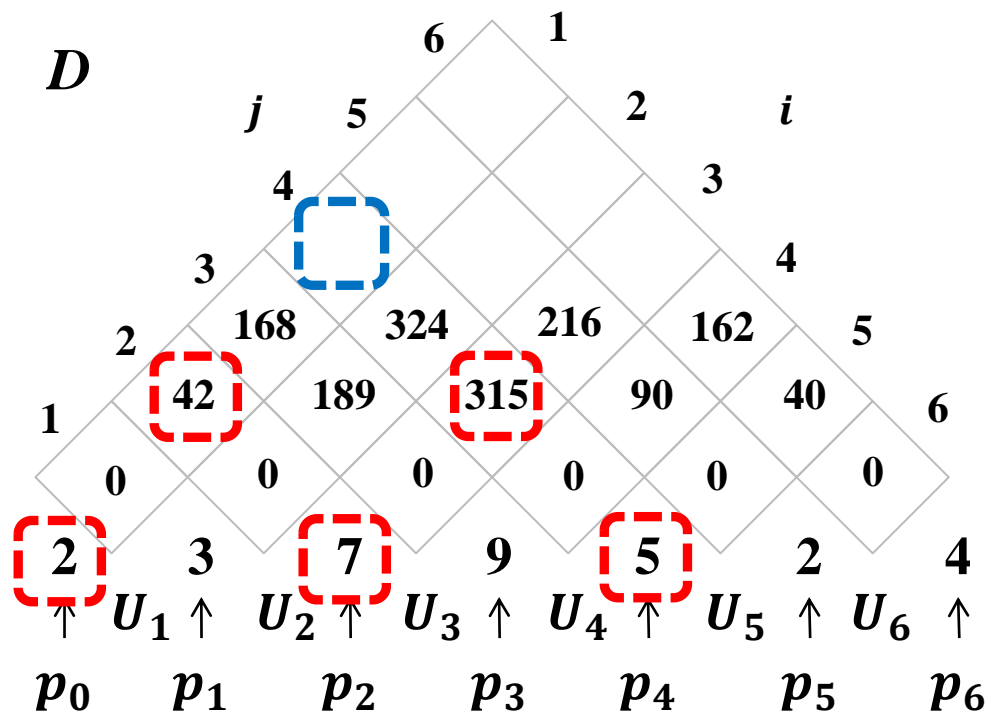
$$= \min \begin{cases} D[4, 4] + D[5, 6] + p_3 p_4 p_6 = 220 \\ D[4, 5] + D[6, 6] + p_3 p_5 p_6 = \mathbf{162} \end{cases}$$



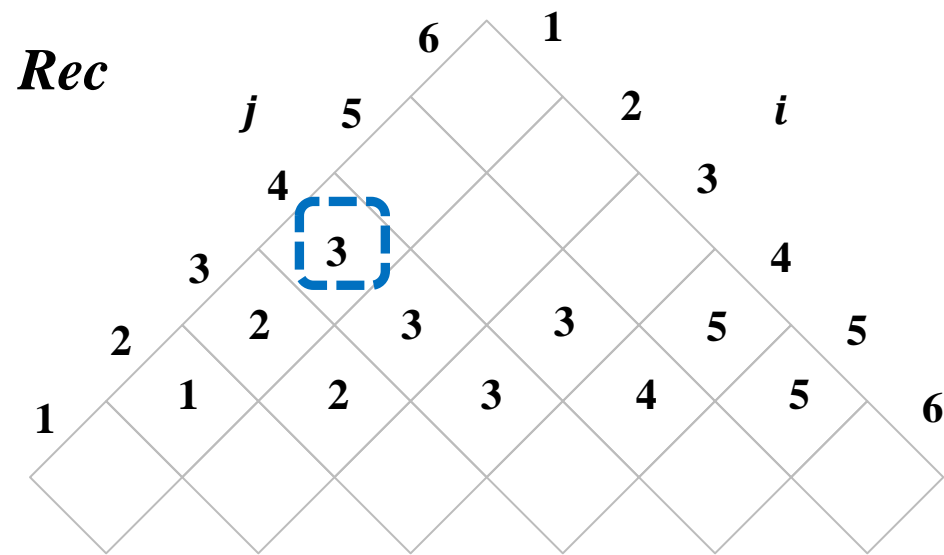
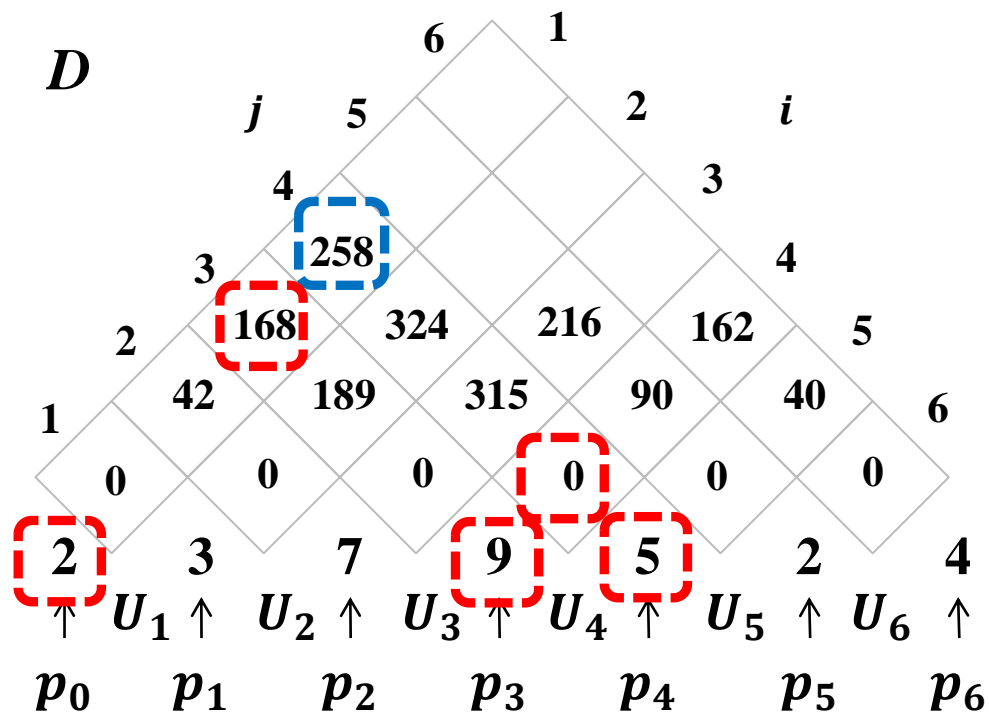
- $$D[1, 4] = \min \begin{cases} D[1, 1] + D[2, 4] + p_0 p_1 p_4 = 354 \\ D[1, 2] + D[3, 4] + p_0 p_2 p_4 = \\ D[1, 3] + D[4, 4] + p_0 p_3 p_4 = \end{cases}$$



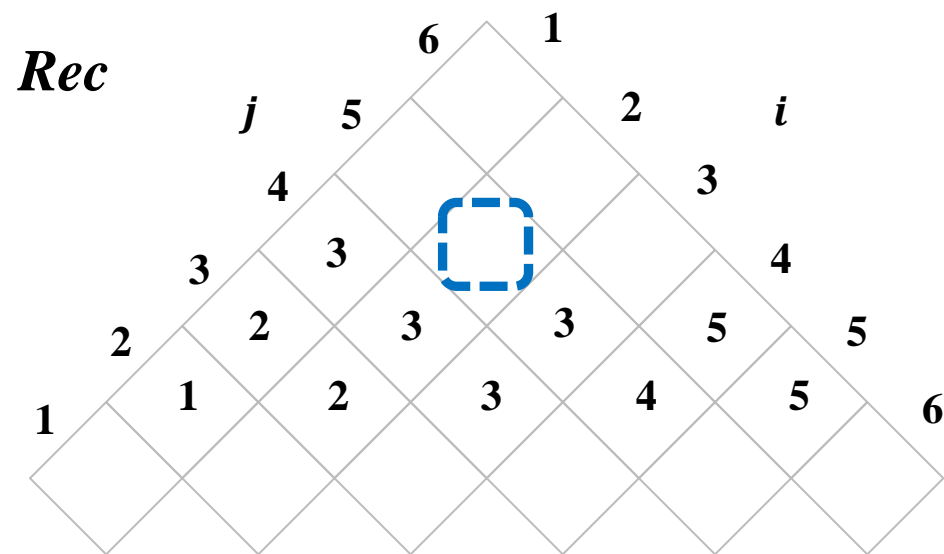
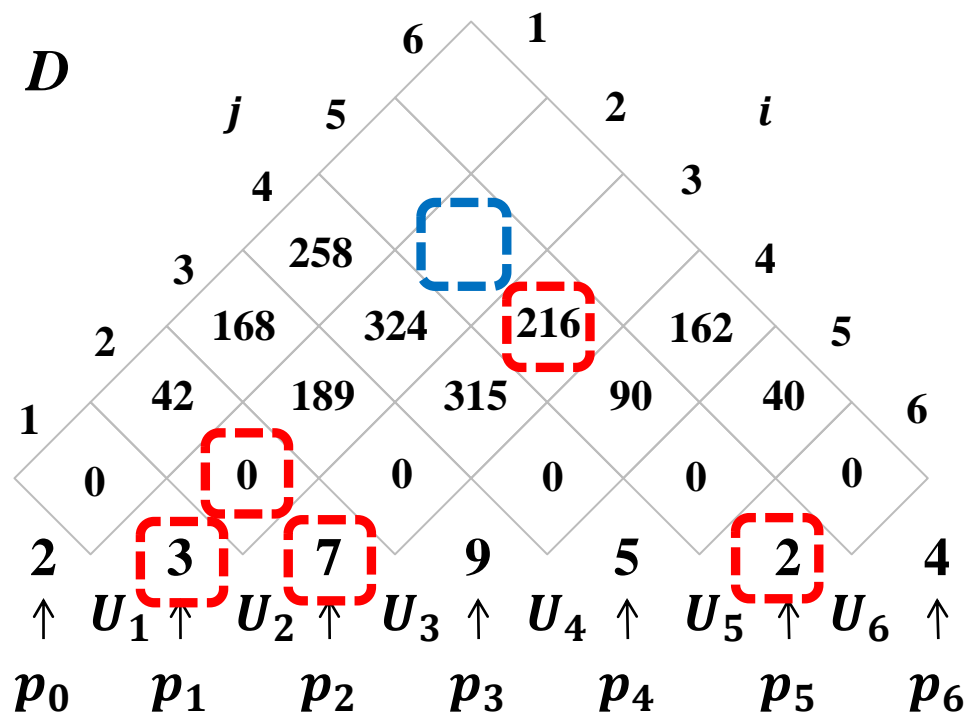
- $D[1, 4] = \min \begin{cases} D[1, 1] + D[2, 4] + p_0 p_1 p_4 = 354 \\ D[1, 2] + D[3, 4] + p_0 p_2 p_4 = 427 \\ D[1, 3] + D[4, 4] + p_0 p_3 p_4 = \end{cases}$



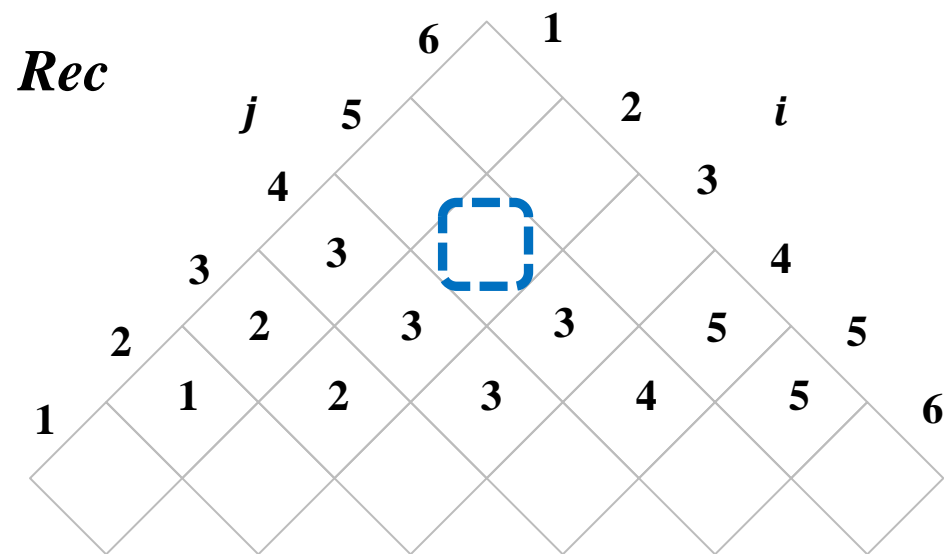
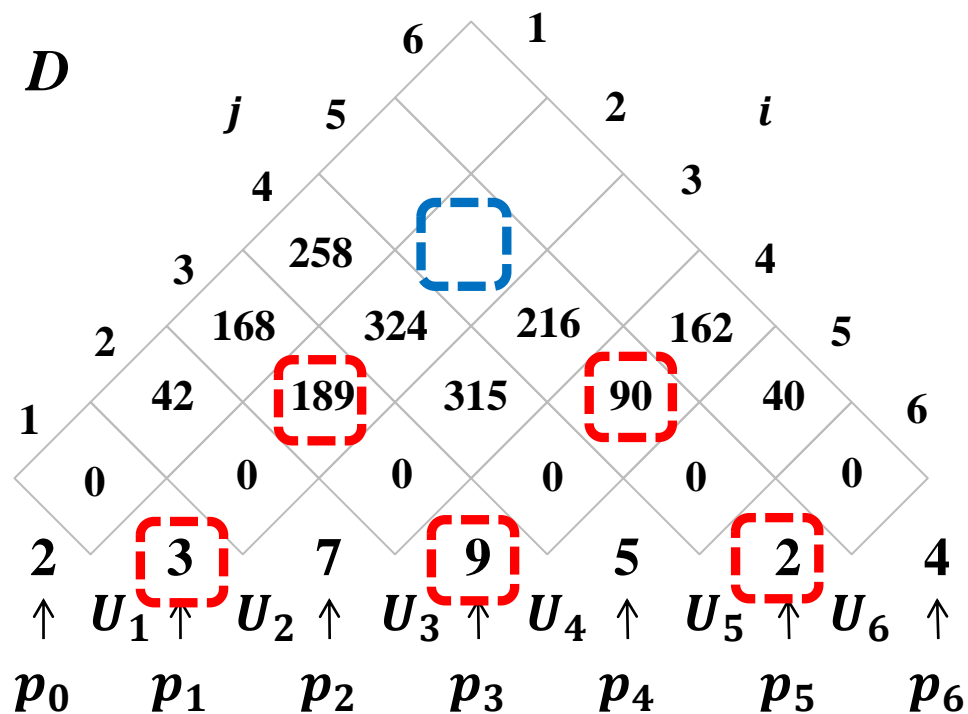
- $D[1, 4] = \min \begin{cases} D[1, 1] + D[2, 4] + p_0 p_1 p_4 = 354 \\ D[1, 2] + D[3, 4] + p_0 p_2 p_4 = 427 \\ D[1, 3] + D[4, 4] + p_0 p_3 p_4 = \mathbf{258} \end{cases}$



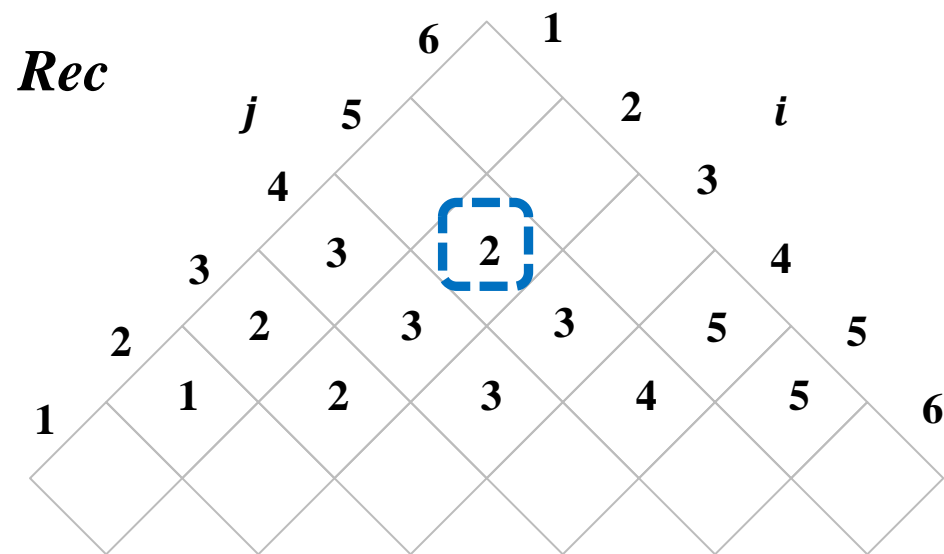
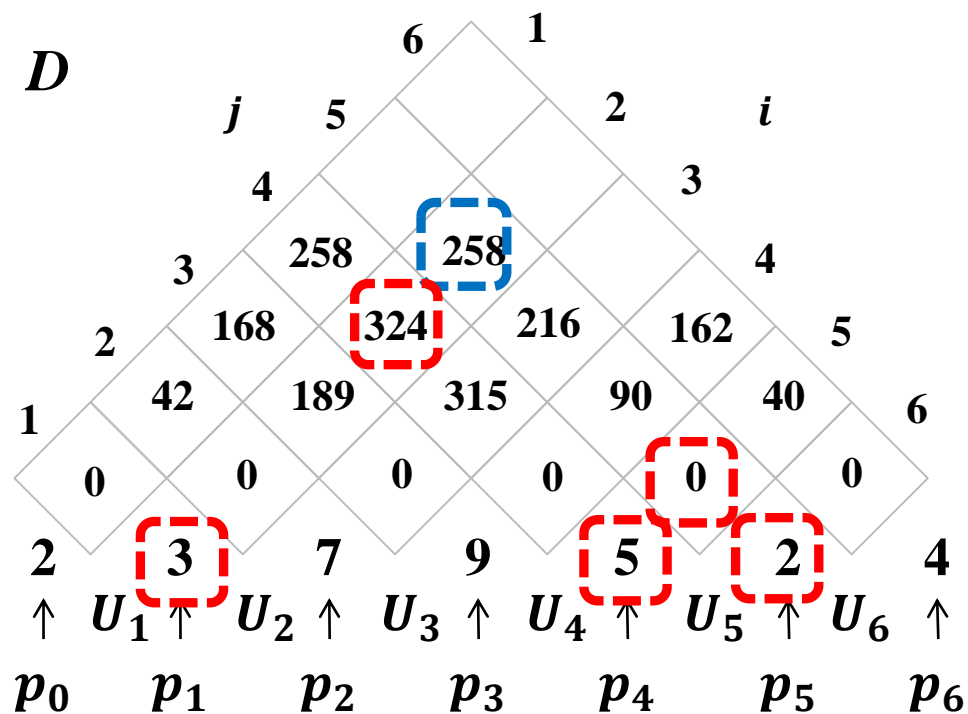
- $$D[2, 5] = \min \begin{cases} D[2, 2] + D[3, 5] + p_1 p_2 p_5 = 258 \\ D[2, 3] + D[4, 5] + p_1 p_3 p_5 = \\ D[2, 4] + D[5, 5] + p_1 p_4 p_5 = \end{cases}$$



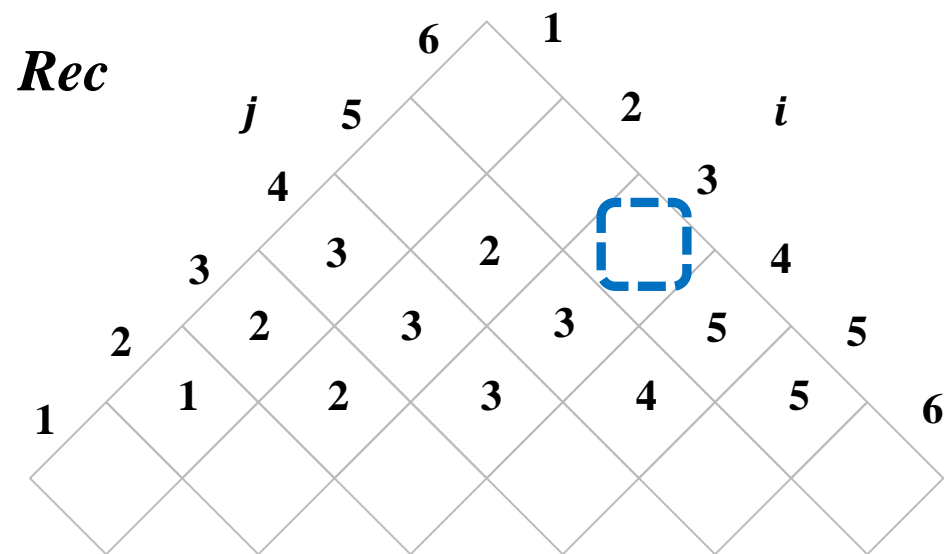
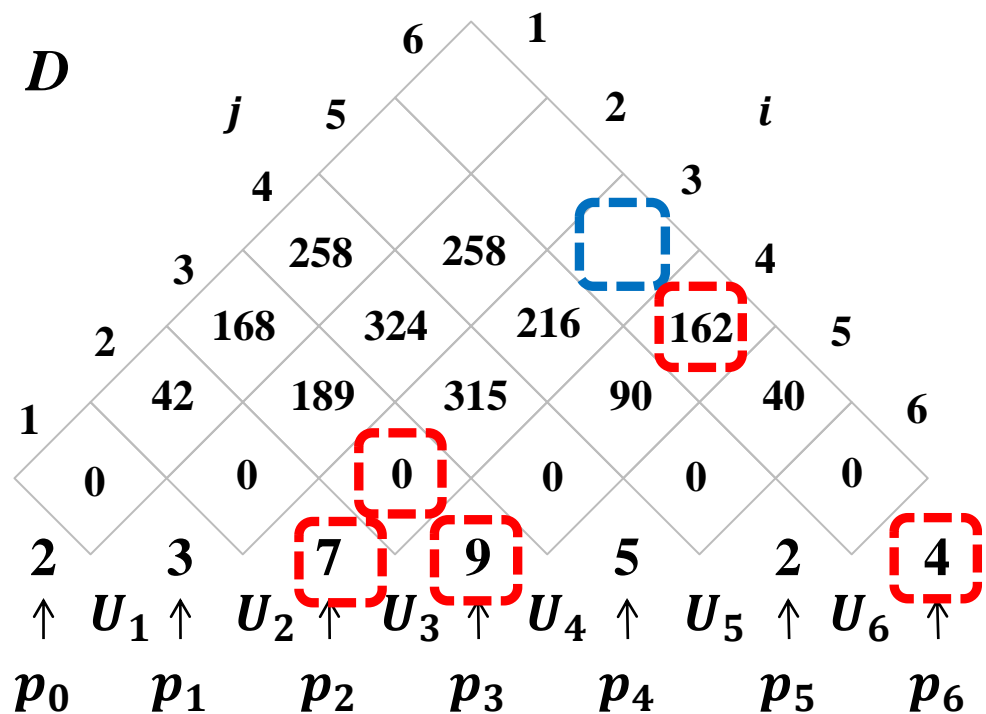
- $$D[2, 5] = \min \begin{cases} D[2, 2] + D[3, 5] + p_1 p_2 p_5 = 258 \\ D[2, 3] + D[4, 5] + p_1 p_3 p_5 = 333 \\ D[2, 4] + D[5, 5] + p_1 p_4 p_5 = \end{cases}$$



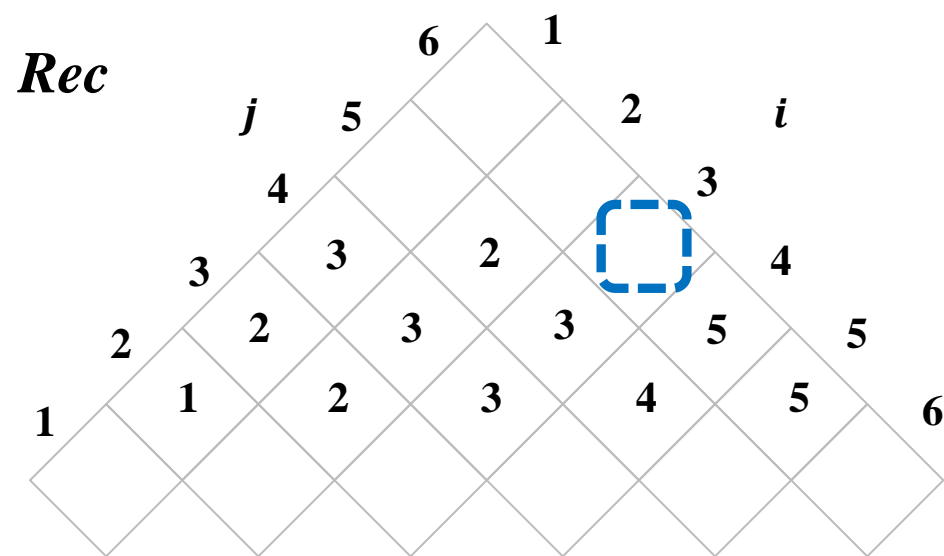
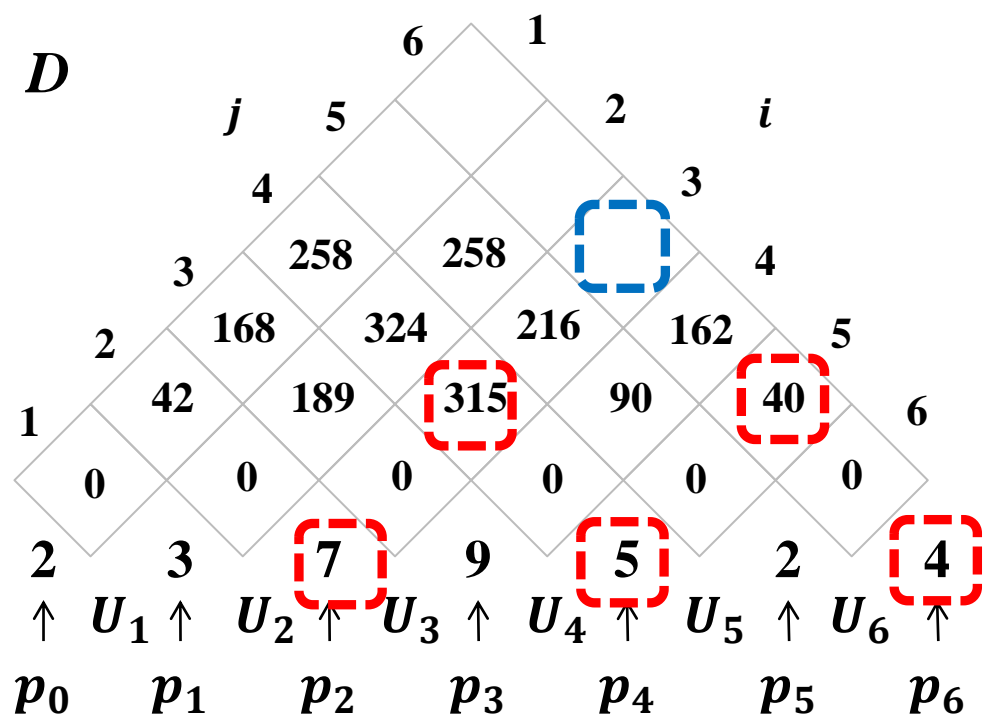
- $$D[2, 5] = \min \begin{cases} D[2, 2] + D[3, 5] + p_1 p_2 p_5 = 258 \\ D[2, 3] + D[4, 5] + p_1 p_3 p_5 = 333 \\ D[2, 4] + D[5, 5] + p_1 p_4 p_5 = 354 \end{cases}$$



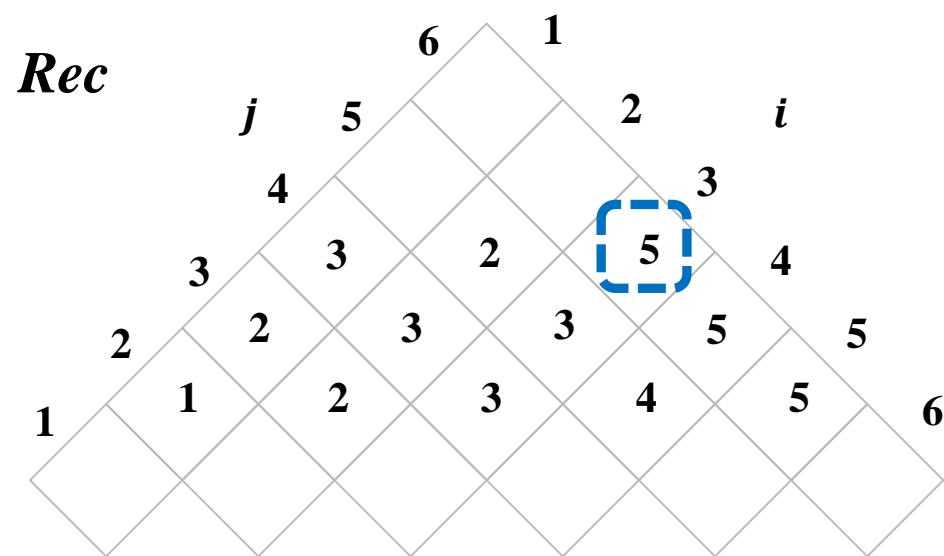
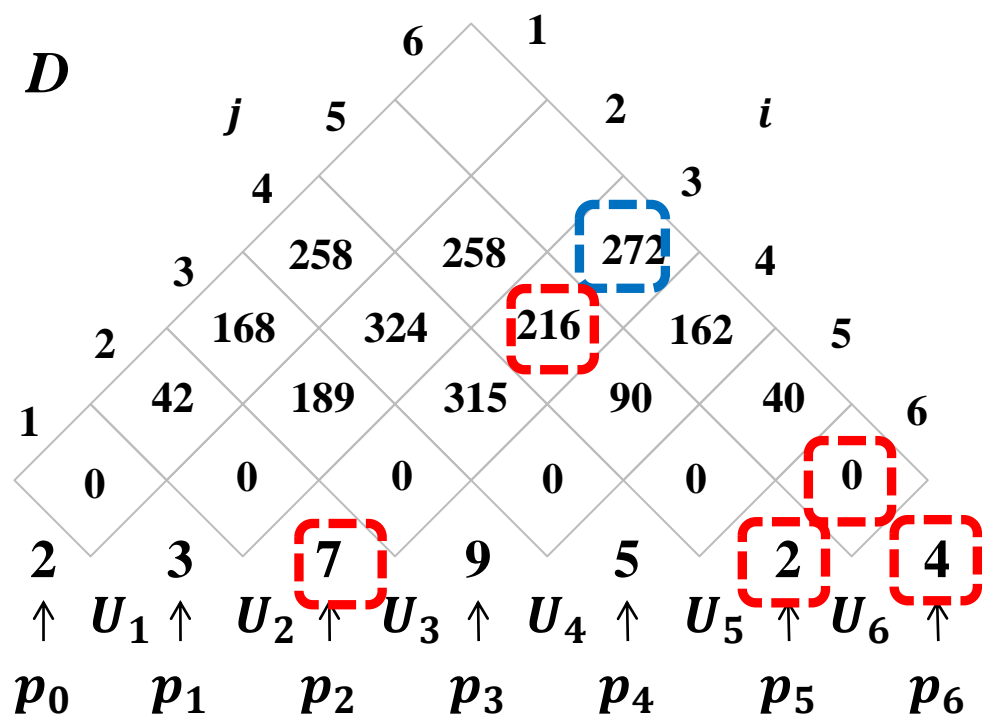
- $$D[3, 6] = \min \begin{cases} D[3, 3] + D[4, 6] + p_2 p_3 p_6 = 414 \\ D[3, 4] + D[5, 6] + p_2 p_4 p_6 = \\ D[3, 5] + D[6, 6] + p_2 p_5 p_6 = \end{cases}$$



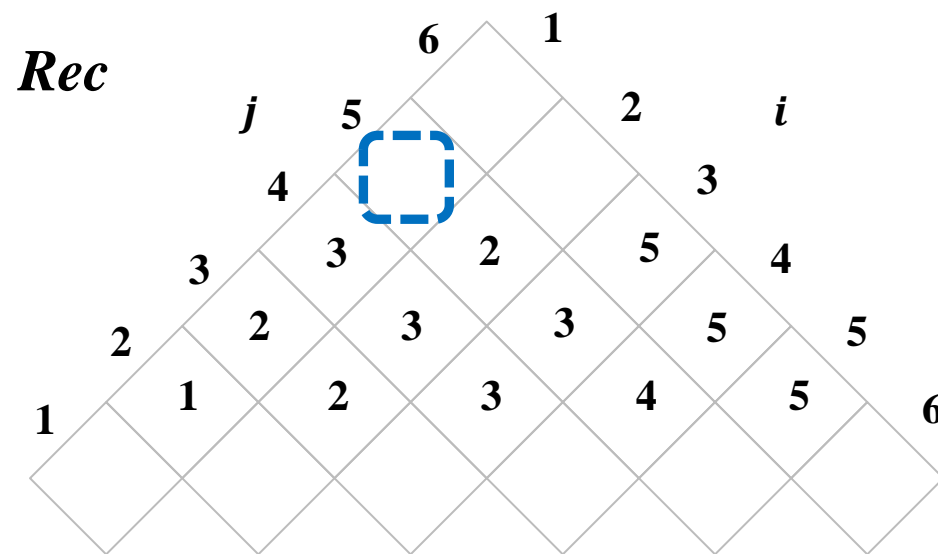
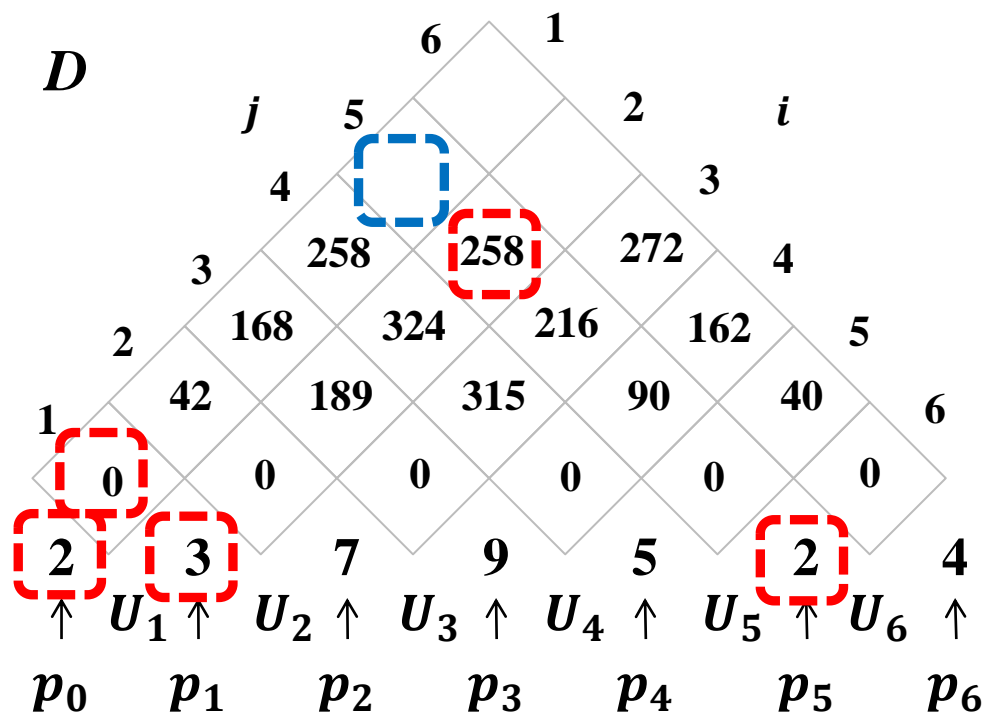
- $$D[3, 6] = \min \begin{cases} D[3, 3] + D[4, 6] + p_2 p_3 p_6 = 414 \\ D[3, 4] + D[5, 6] + p_2 p_4 p_6 = 495 \\ D[3, 5] + D[6, 6] + p_2 p_5 p_6 = \end{cases}$$



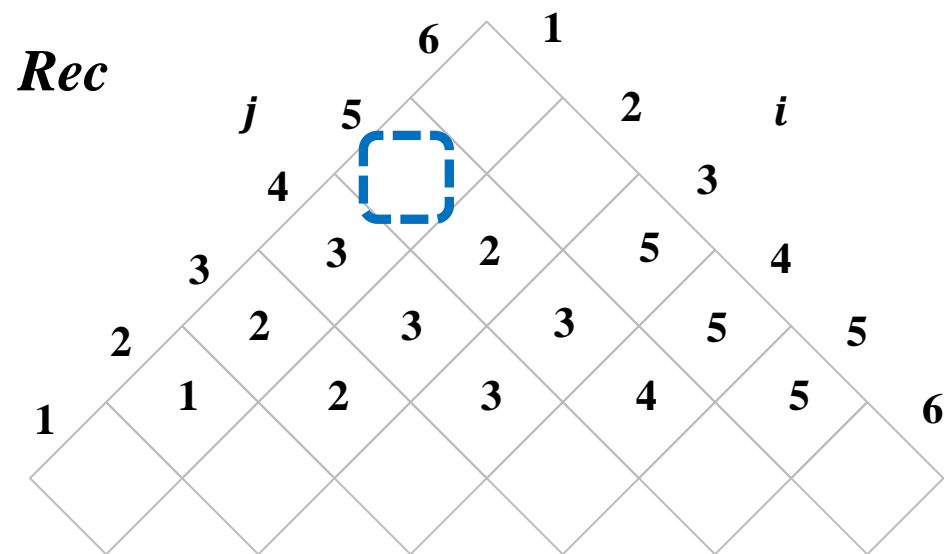
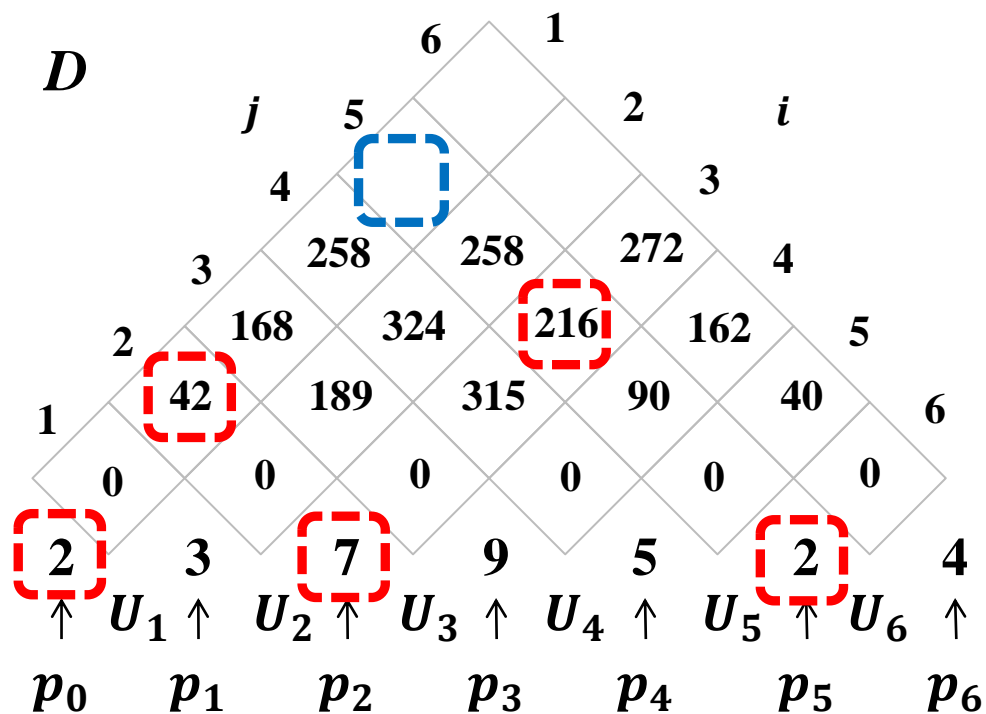
- $$D[3, 6] = \min \begin{cases} D[3, 3] + D[4, 6] + p_2 p_3 p_6 = 414 \\ D[3, 4] + D[5, 6] + p_2 p_4 p_6 = 495 \\ D[3, 5] + D[6, 6] + p_2 p_5 p_6 = \mathbf{272} \end{cases}$$



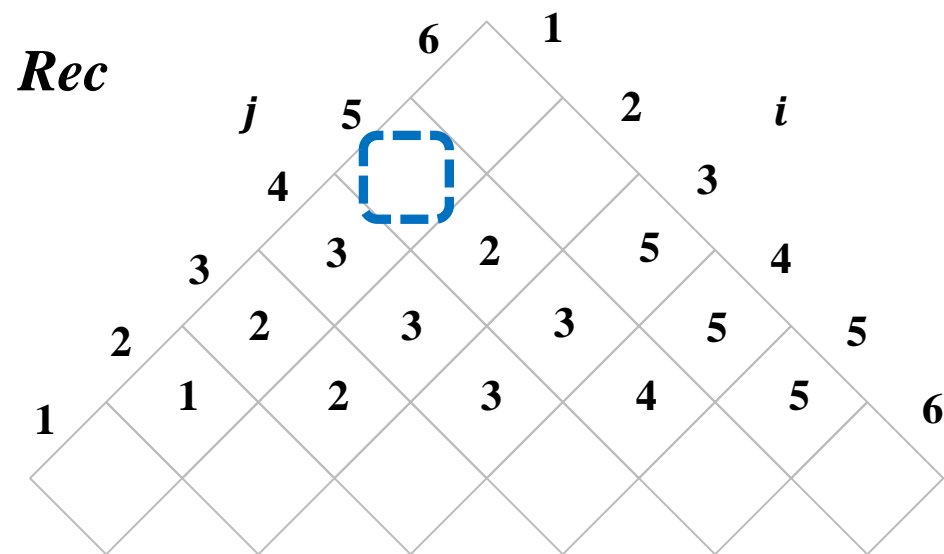
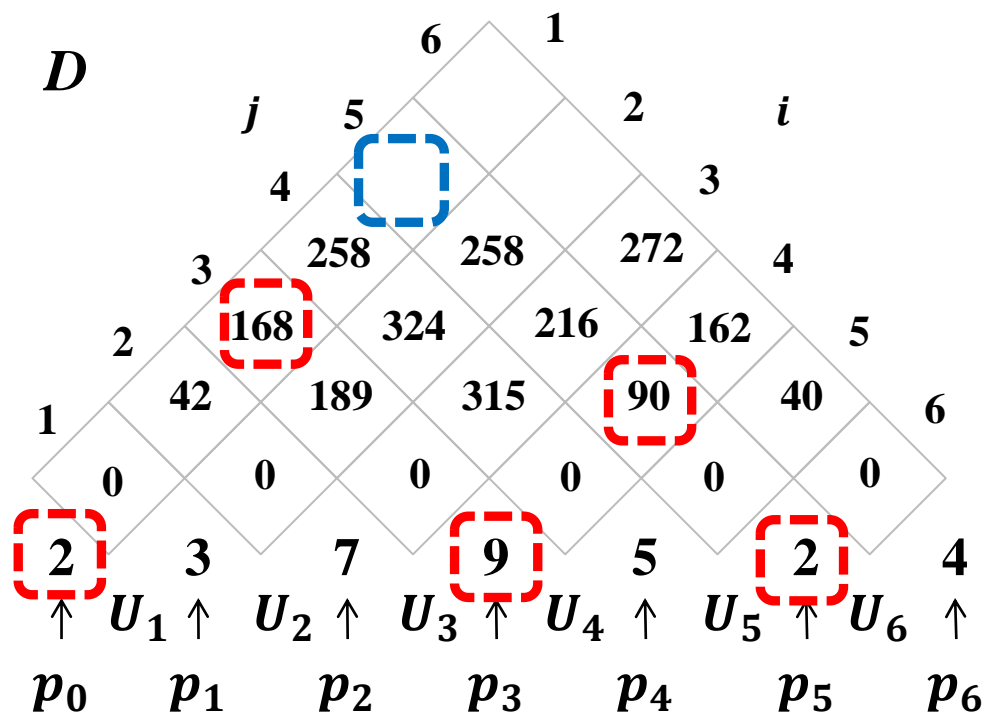
- $$D[1, 5] = \min \begin{cases} D[1, 1] + D[2, 5] + p_0 p_1 p_5 = 270 \\ D[1, 2] + D[3, 5] + p_0 p_2 p_5 = \\ D[1, 3] + D[4, 5] + p_0 p_3 p_5 = \\ D[1, 4] + D[5, 5] + p_0 p_4 p_5 = \end{cases}$$



- $$D[1, 5] = \min \begin{cases} D[1, 1] + D[2, 5] + p_0 p_1 p_5 = 270 \\ D[1, 2] + D[3, 5] + p_0 p_2 p_5 = 286 \\ D[1, 3] + D[4, 5] + p_0 p_3 p_5 = \\ D[1, 4] + D[5, 5] + p_0 p_4 p_5 = \end{cases}$$

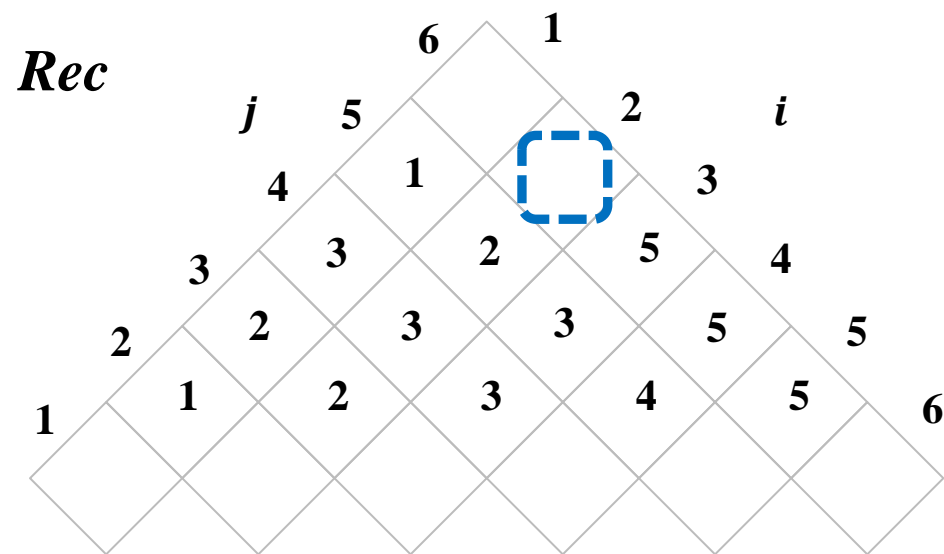
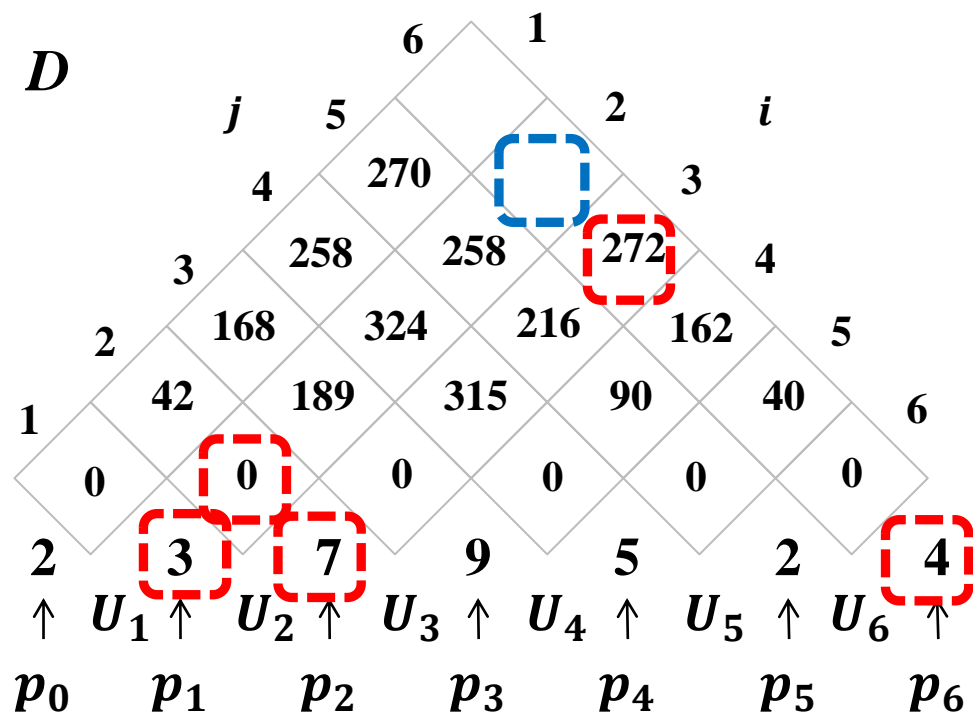


- $$D[1, 5] = \min \begin{cases} D[1, 1] + D[2, 5] + p_0 p_1 p_5 = 270 \\ D[1, 2] + D[3, 5] + p_0 p_2 p_5 = 286 \\ D[1, 3] + D[4, 5] + p_0 p_3 p_5 = 294 \\ D[1, 4] + D[5, 5] + p_0 p_4 p_5 = \end{cases}$$

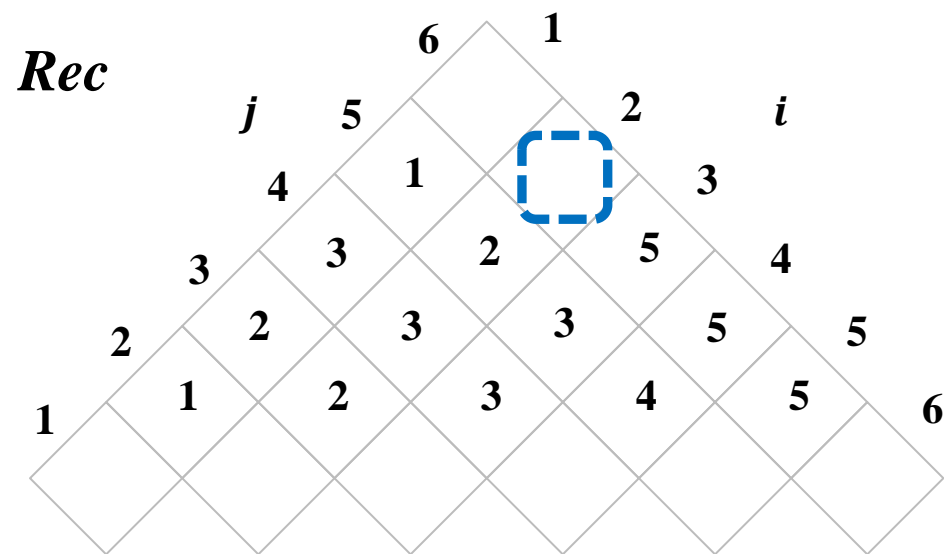
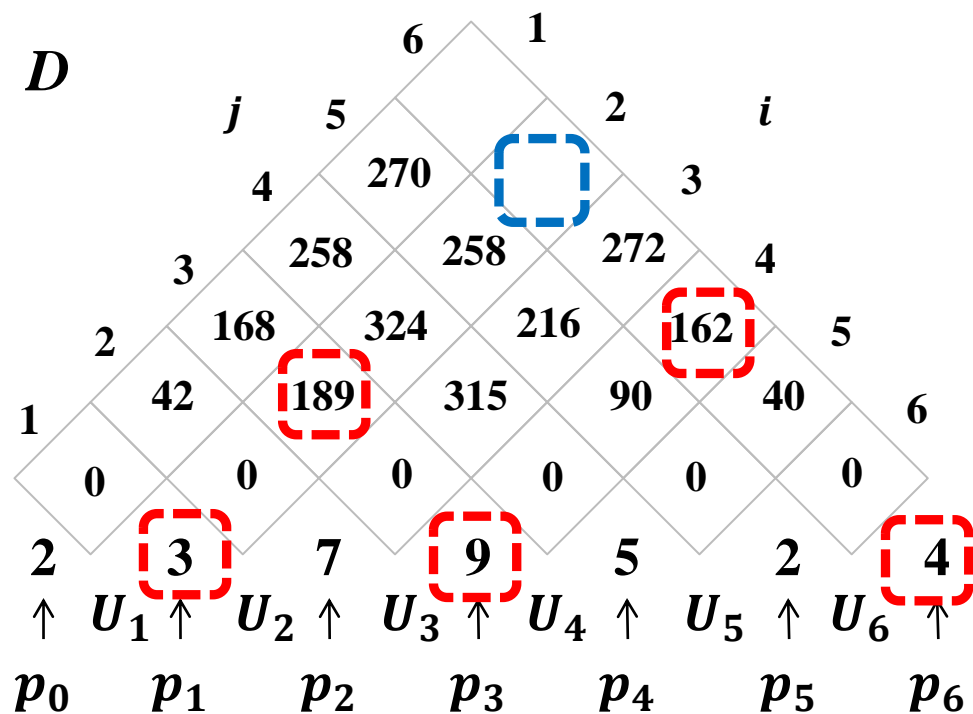




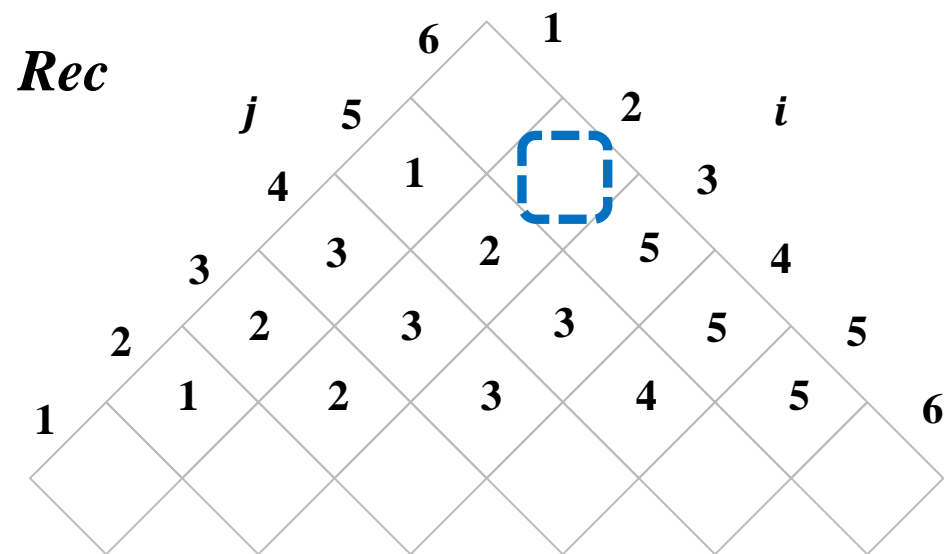
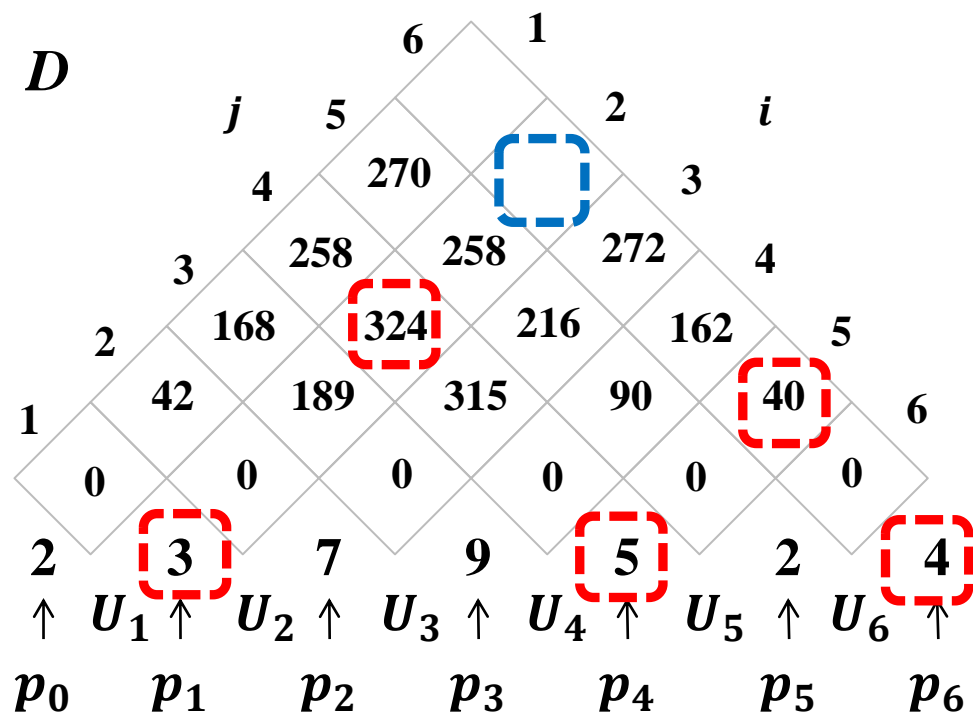
- $$D[2, 6] = \min \begin{cases} D[2, 2] + D[3, 6] + p_1 p_2 p_6 = 356 \\ D[2, 3] + D[4, 6] + p_1 p_3 p_6 = \\ D[2, 4] + D[5, 6] + p_1 p_4 p_6 = \\ D[2, 5] + D[6, 6] + p_1 p_5 p_6 = \end{cases}$$



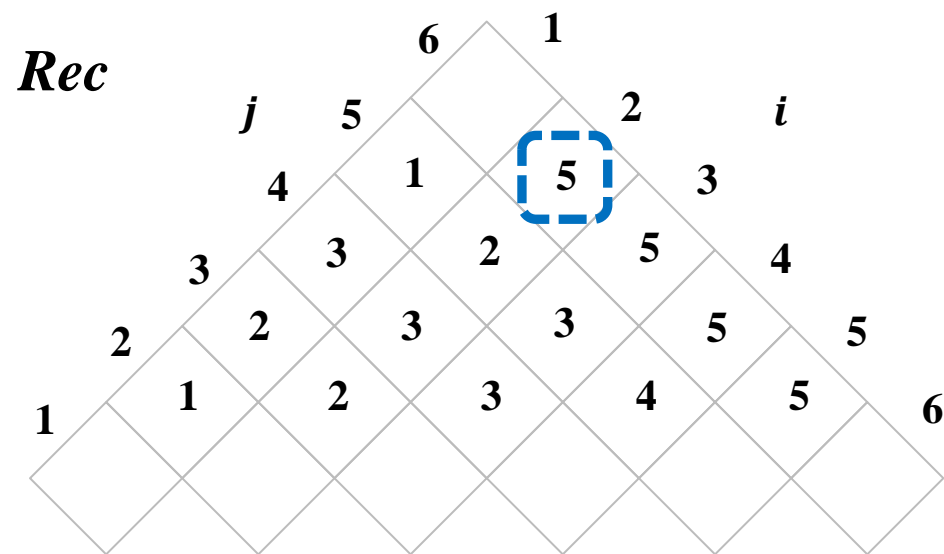
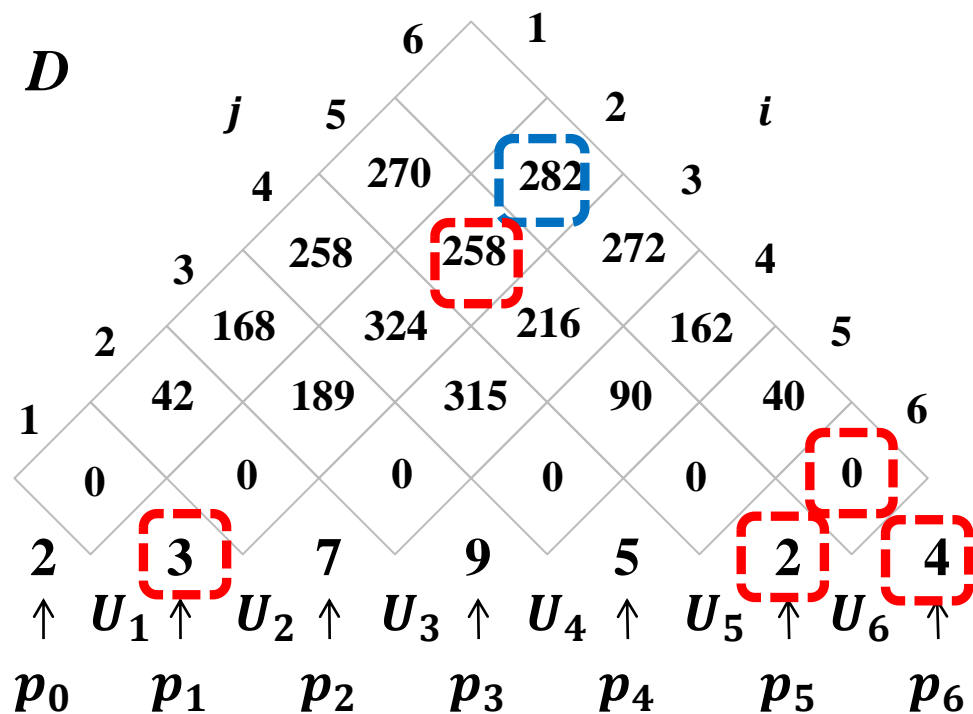
- $$D[2, 6] = \min \begin{cases} D[2, 2] + D[3, 6] + p_1 p_2 p_6 = 356 \\ D[2, 3] + D[4, 6] + p_1 p_3 p_6 = 459 \\ D[2, 4] + D[5, 6] + p_1 p_4 p_6 = \\ D[2, 5] + D[6, 6] + p_1 p_5 p_6 = \end{cases}$$



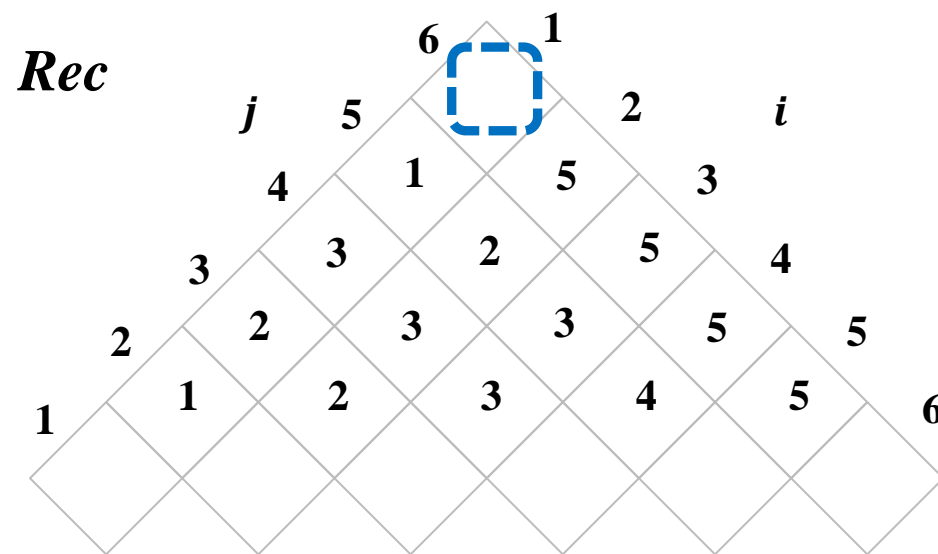
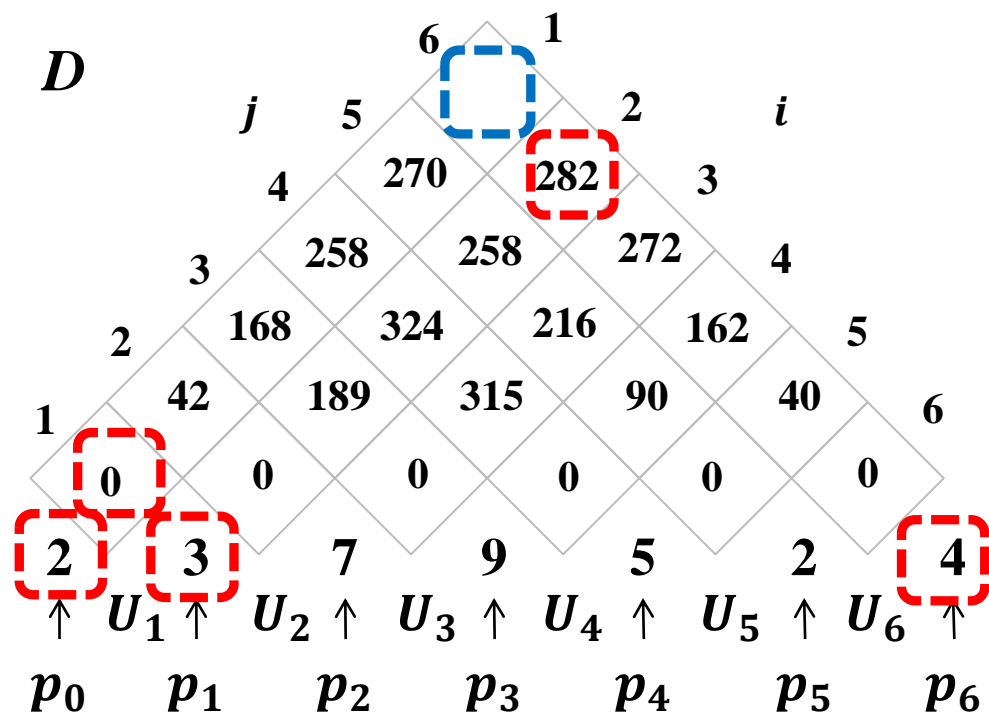
- $$D[2, 6] = \min \begin{cases} D[2, 2] + D[3, 6] + p_1 p_2 p_6 = 356 \\ D[2, 3] + D[4, 6] + p_1 p_3 p_6 = 459 \\ D[2, 4] + D[5, 6] + p_1 p_4 p_6 = 424 \\ D[2, 5] + D[6, 6] + p_1 p_5 p_6 = \end{cases}$$

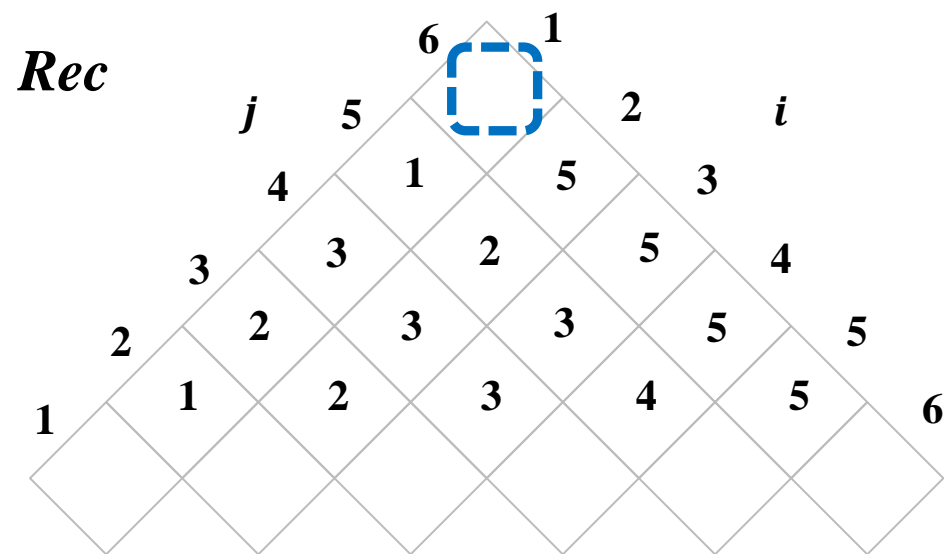


- $$D[2, 6] = \min \begin{cases} D[2, 2] + D[3, 6] + p_1 p_2 p_6 = 356 \\ D[2, 3] + D[4, 6] + p_1 p_3 p_6 = 459 \\ D[2, 4] + D[5, 6] + p_1 p_4 p_6 = 424 \\ D[2, 5] + D[6, 6] + p_1 p_5 p_6 = \mathbf{282} \end{cases}$$



$$D[1, 6] = \min \begin{cases} D[1, 1] + D[2, 6] + p_0 p_1 p_6 = 306 \\ D[1, 2] + D[3, 6] + p_0 p_2 p_6 = \\ D[1, 3] + D[4, 6] + p_0 p_3 p_6 = \\ D[1, 4] + D[5, 6] + p_0 p_4 p_6 = \\ D[1, 5] + D[6, 6] + p_0 p_5 p_6 = \end{cases}$$





●



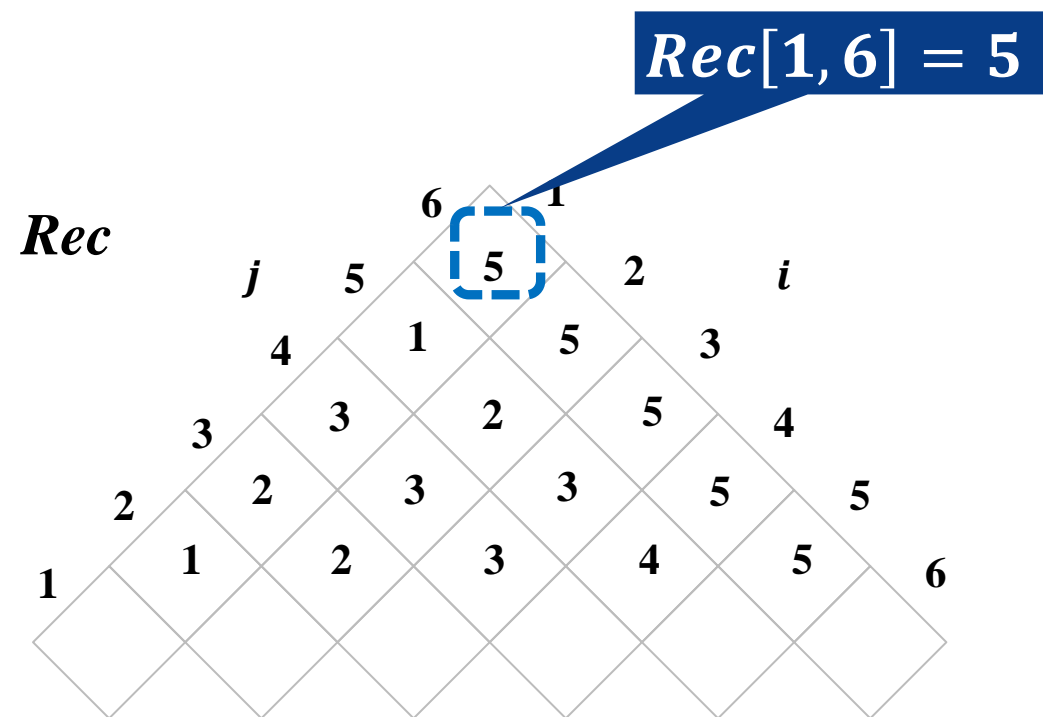
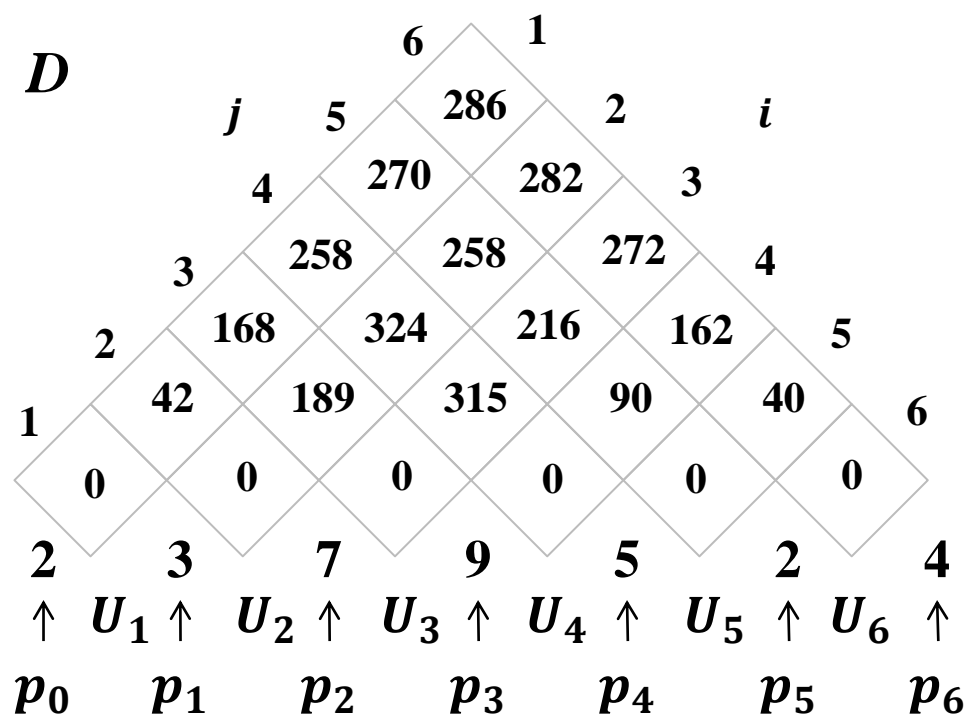




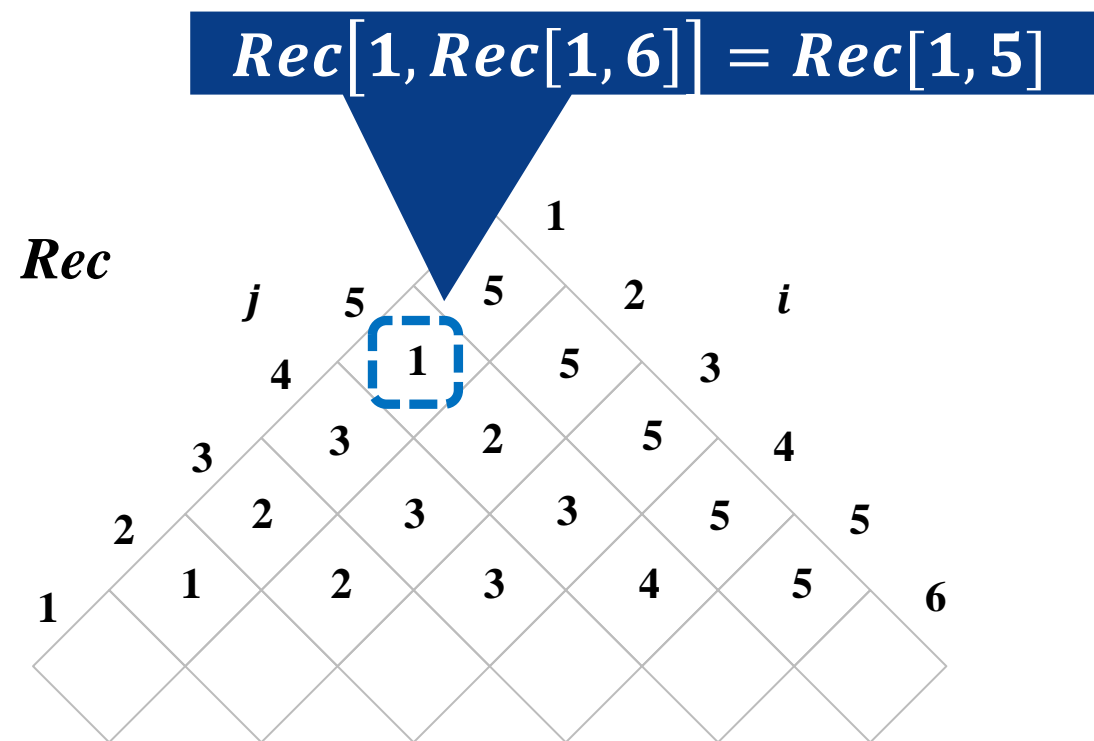
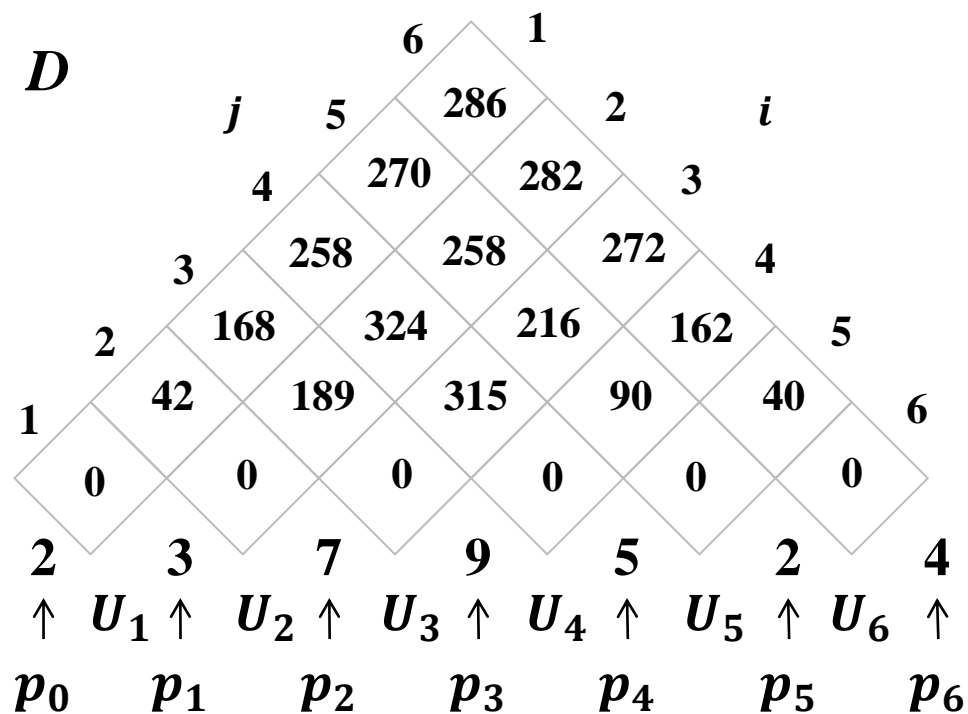
算法实例



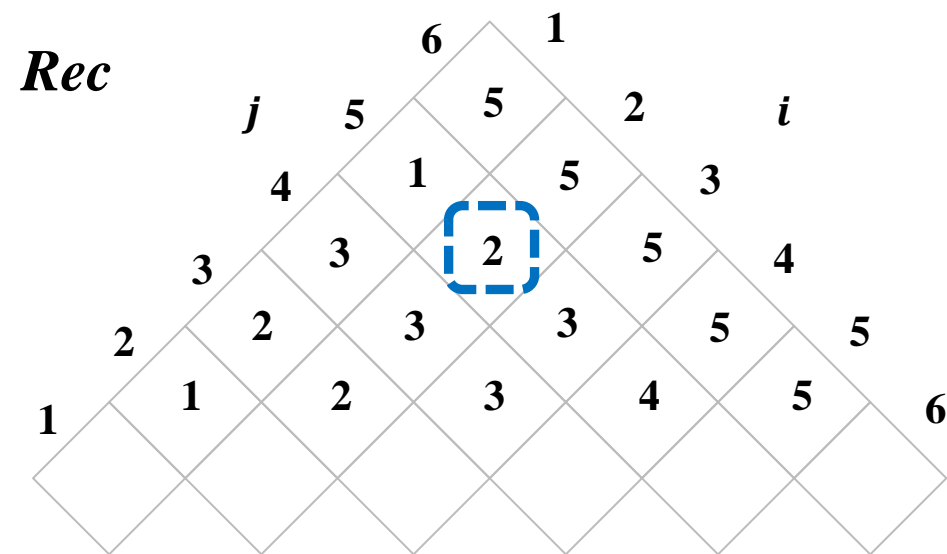
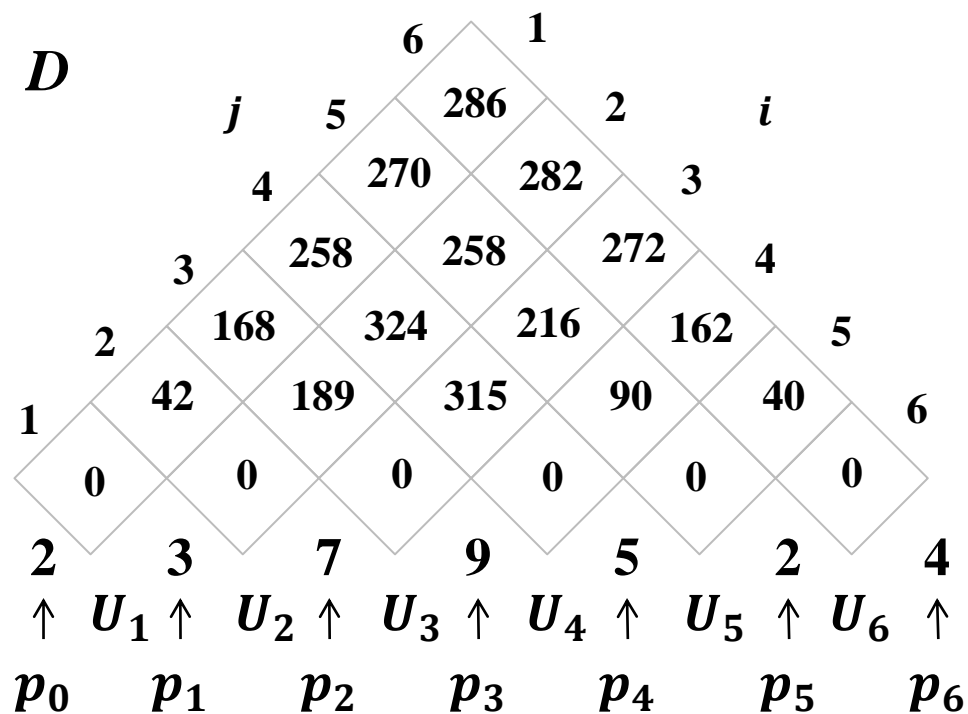
- $U_1 U_2 U_3 U_4 U_5 U_6$
- $\rightarrow (U_1 U_2 U_3 U_4 U_5)(U_6)$



- $U_1 U_2 U_3 U_4 U_5 U_6$
- $\rightarrow (U_1 U_2 U_3 U_4 U_5)(U_6)$



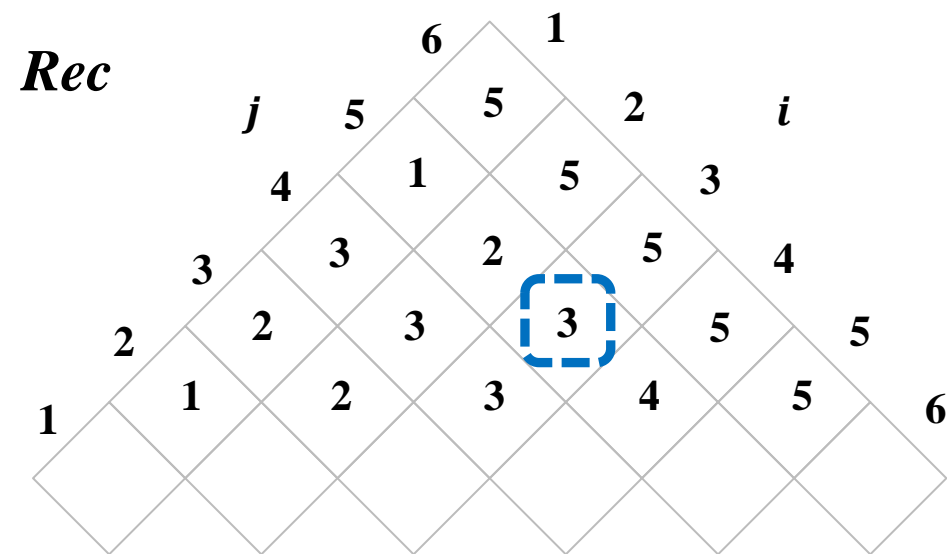
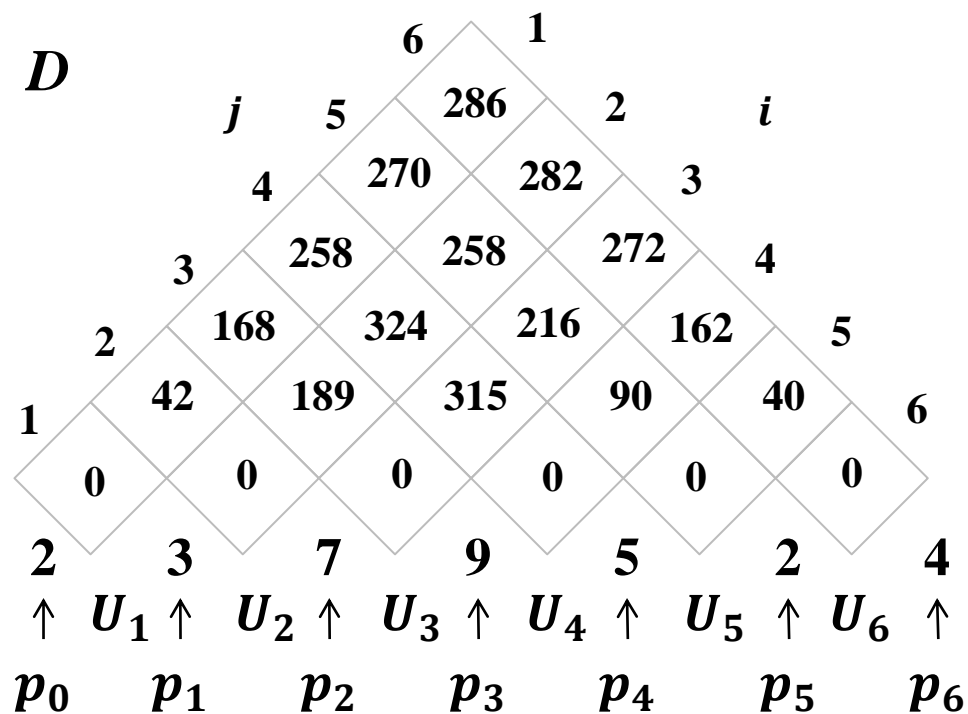
- $U_1 U_2 U_3 U_4 U_5 U_6$
- $\rightarrow (U_1 U_2 U_3 U_4 U_5)(U_6)$
- $\rightarrow (U_1 (U_2 U_3 U_4 U_5))(U_6)$



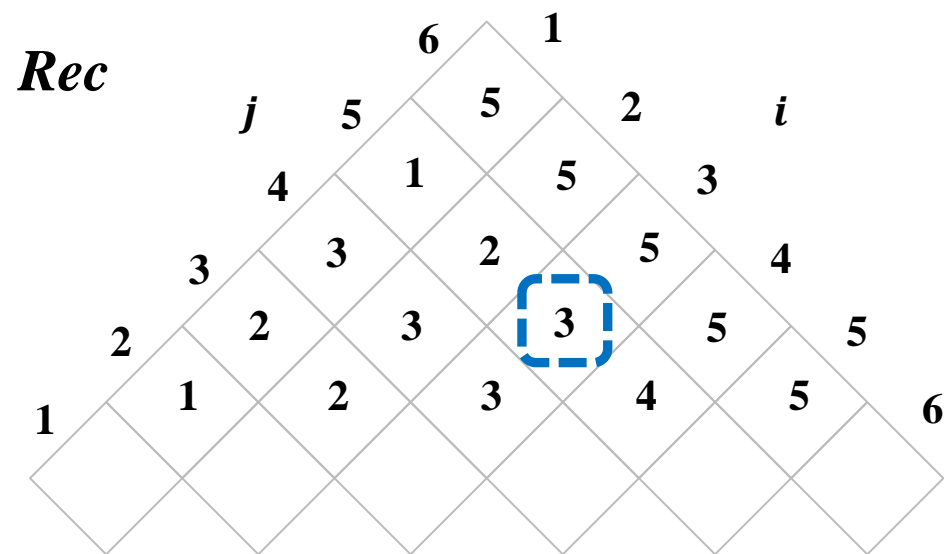
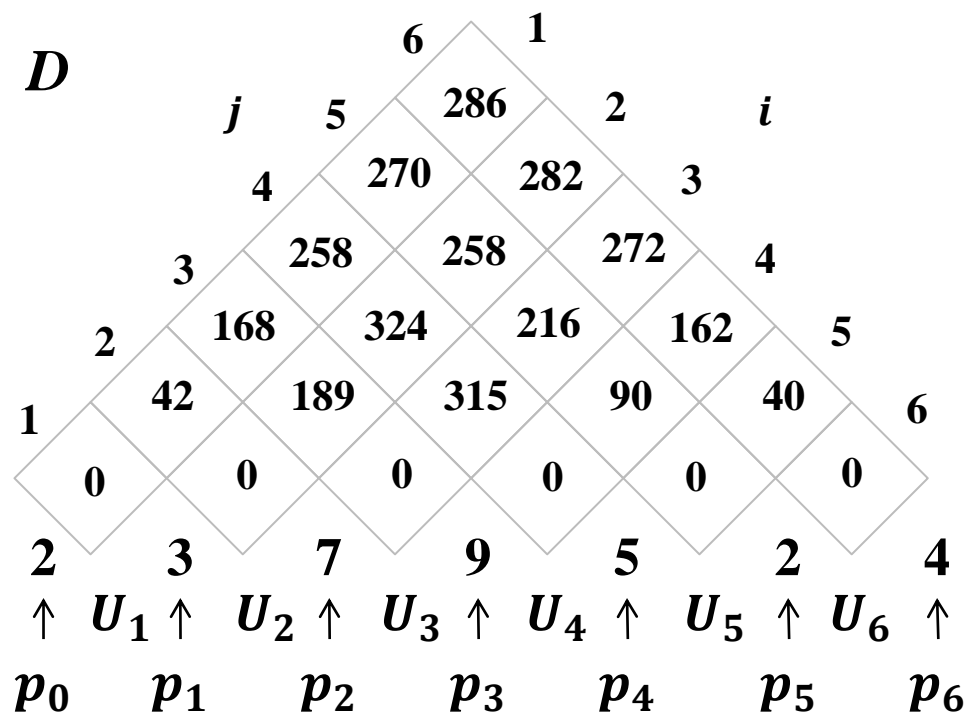
算法实例



- $U_1 U_2 U_3 U_4 U_5 U_6$
- $\rightarrow (U_1 U_2 U_3 U_4 U_5)(U_6)$
- $\rightarrow (U_1(U_2 U_3 U_4 U_5))(U_6)$
- $\rightarrow (U_1(U_2(U_3 U_4 U_5)))(U_6)$



- $U_1 U_2 U_3 U_4 U_5 U_6$
- $\rightarrow (U_1 U_2 U_3 U_4 U_5)(U_6)$
- $\rightarrow (U_1(U_2 U_3 U_4 U_5))(U_6)$
- $\rightarrow (U_1(U_2(U_3 U_4 U_5)))(U_6) \rightarrow (U_1(U_2((U_3)(U_4 U_5))))(U_6)$



- Matrix-Chain-Multiply(p, n)

输入: 矩阵维度数组 p , 矩阵的个数 n

输出: 最小标量乘法次数, 分割方式追踪数组 Rec

新建二维数组 $D[1...n, 1...n], Rec[1...n, 1...n]$

//初始化

$D \leftarrow \infty$

for $i \leftarrow 1$ to n do

| $D[i, i] \leftarrow 0$

end

初始化

- Matrix-Chain-Multiply(p, n)

//动态规划

for $l \leftarrow 2$ to n do

区间长度从小到大

for $i \leftarrow 1$ to $n - l + 1$ do

$j \leftarrow i + l - 1$

 for $k \leftarrow i$ to $j - 1$ do

$q \leftarrow D[i, k] + D[k + 1, j] + p[i - 1] * p[k] * p[j]$

 if $q < D[i, j]$ then

$D[i, j] \leftarrow q$

$Rec[i, j] \leftarrow k$

 end

 end

end

end

return $D[1, n], Rec$

- Matrix-Chain-Multiply(p, n)

//动态规划

for $l \leftarrow 2$ to n do

for $i \leftarrow 1$ to $n - l + 1$ do

$j \leftarrow i + l - 1$

 for $k \leftarrow i$ to $j - 1$ do

$q \leftarrow D[i, k] + D[k + 1, j] + p[i - 1] * p[k] * p[j]$

 if $q < D[i, j]$ then

$D[i, j] \leftarrow q$

$Rec[i, j] \leftarrow k$

 end

 end

 end

end

return $D[1, n], Rec$

依次计算子问题

- Matrix-Chain-Multiply(p, n)

//动态规划

```
for  $l \leftarrow 2$  to  $n$  do
```

```
  for  $i \leftarrow 1$  to  $n - l + 1$  do
```

```
     $j \leftarrow i + l - 1$ 
```

```
    for  $k \leftarrow i$  to  $j - 1$  do
```

```
       $q \leftarrow D[i, k] + D[k + 1, j] + p[i - 1] * p[k] * p[j]$ 
```

```
      if  $q < D[i, j]$  then
```

```
         $D[i, j] \leftarrow q$ 
```

```
         $Rec[i, j] \leftarrow k$ 
```

```
      end
```

```
    end
```

```
  end
```

```
end
```

```
return  $D[1, n], Rec$ 
```

枚举所有分割位置

- Matrix-Chain-Multiply(p, n)

//动态规划

```
for  $l \leftarrow 2$  to  $n$  do
  for  $i \leftarrow 1$  to  $n - l + 1$  do
     $j \leftarrow i + l - 1$ 
    for  $k \leftarrow i$  to  $j - 1$  do
       $q \leftarrow D[i, k] + D[k + 1, j] + p[i - 1] * p[k] * p[j]$ 
      if  $q < D[i, j]$  then
         $D[i, j] \leftarrow q$ 
         $Rec[i, j] \leftarrow k$ 
      end
    end
  end
end
return  $D[1, n], Rec$ 
```

计算最少乘法次数

- Matrix-Chain-Multiply(p, n)

//动态规划

```
for  $l \leftarrow 2$  to  $n$  do
  for  $i \leftarrow 1$  to  $n - l + 1$  do
     $j \leftarrow i + l - 1$ 
    for  $k \leftarrow i$  to  $j - 1$  do
       $q \leftarrow D[i, k] + D[k + 1, j] + p[i - 1] * p[k] * p[j]$ 
      if  $q < D[i, j]$  then
         $D[i, j] \leftarrow q$ 
         $Rec[i, j] \leftarrow k$ 
      end
    end
  end
end
return  $D[1, n], Rec$ 
```

记录最优决策

最优方案追踪：伪代码



- Print-Matrix-Chain(U, Rec, i, j)

初始调用: Print-Matrix-Chain($U, Rec, 1, n$)

输入: 矩阵链 $U_{1..n}$, 追踪数组 $Rec[1..n, 1..n]$, 位置索引 i, j

输出: 矩阵链的加括号方式

```
if  $i = j$  then  
    print  $U_i$   
    return  
end
```

链长为1直接输出

```
print "("
```

```
Print-Matrix-Chain( $U, Rec, i, Rec[i, j]$ )
```

```
print ")("
```

```
Print-Matrix-Chain( $U, Rec, Rec[i, j] + 1, j$ )
```

```
print ")"
```

```
return
```

最优方案追踪：伪代码



- Print-Matrix-Chain(U, Rec, i, j)

初始调用：Print-Matrix-Chain($U, Rec, 1, n$)

输入：矩阵链 $U_{1..n}$, 追踪数组 $Rec[1..n, 1..n]$, 位置索引 i, j

输出：矩阵链的加括号方式

if $i = j$ then

 print U_i

 return

end

print “(”

Print-Matrix-Chain($U, Rec, i, Rec[i, j]$)

print “)”

Print-Matrix-Chain($U, Rec, Rec[i, j] + 1, j$)

print “)”

return

递归输出左侧加括号方式

最优方案追踪：伪代码



- Print-Matrix-Chain(U, Rec, i, j)

初始调用：Print-Matrix-Chain($U, Rec, 1, n$)

输入：矩阵链 $U_{1..n}$, 追踪数组 $Rec[1..n, 1..n]$, 位置索引 i, j

输出：矩阵链的加括号方式

if $i = j$ then

 | print U_i

 | return

end

print “(”

Print-Matrix-Chain($U, Rec, i, Rec[i, j]$)

print “)(”

Print-Matrix-Chain($U, Rec, Rec[i, j] + 1, j$)

print “)”

return

递归输出右侧加括号方式

- Matrix-Chain-Multiply(p, n)

//动态规划

```
for  $l \leftarrow 2$  to  $n$  do
  for  $i \leftarrow 1$  to  $n - l + 1$  do
     $j \leftarrow i + l - 1$ 
     $D[i, j] \leftarrow \infty$ 
    for  $k \leftarrow i$  to  $j - 1$  do
       $q \leftarrow D[i, k] + D[k + 1, j] + p[i - 1] * p[k] * p[j]$ 
      if  $q < D[i, j]$  then
         $D[i, j] \leftarrow q$ 
         $Rec[i, j] \leftarrow k$ 
      end
    end
  end
end
return  $D[1, n], Rec$ 
```

时间复杂度: $O(n^3)$

$O(n)$ $O(n^2)$ $O(n^3)$