Design and Analysis of Algorithms

Part II: Dynamic Programming

Lecture 16: Chain Matrix Multiplication

童咏昕

北京航空航天大学
计算机学院

动态规划篇概述

- 在算法课程第二部分“动态规划”主题中，我们将主要聚焦于如下经典问题：
  - 0-1 Knapsack (0-1背包问题)
  - Maximum Contiguous Subarray II (最大连续子数组 II)
  - Longest Common Subsequences (最长公共子序列)
  - Longest Common Substrings (最长公共子串)
  - Minimum Edit Distance (最小编辑距离)
  - Rod-Cutting (钢条切割)
  - Chain Matrix Multiplication (矩阵链乘法)

动态规划篇概述

- 在算法课程第二部分“动态规划”主题中，我们将主要聚焦于如下经典问题：
  - 0-1 Knapsack (0-1背包问题)
  - Maximum Contiguous Subarray II (最大连续子数组 II)
  - Longest Common Subsequences (最长公共子序列)
  - Longest Common Substrings (最长公共子串)
  - Minimum Edit Distance (最小编辑距离)
  - Rod-Cutting (钢条切割)
  - Chain Matrix Multiplication (矩阵链乘法)

问题背景：基本知识

- 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$V_{3,2}$

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

问题背景：基本知识

- 矩阵
  - $p \times q$的矩阵$U_{p,q}$
  - 例
    - 4×3的矩阵$U_{4,3}$

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

    - 3×2的矩阵$V_{3,2}$

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

问题背景：基本知识

- 矩阵
  - $p \times q$的矩阵$U_{p,q}$
  - 例
    - 4×3的矩阵$U_{4,3}$

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

    - 3×2的矩阵$V_{3,2}$

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

问题：如何计算$U \cdot V$?

问题背景：基本知识

- **2个矩阵相乘**

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

问题背景：基本知识

- 2个矩阵相乘

$$U = \underbrace{\begin{bmatrix} \boxed{2 & 1 & 3} \\ 9 & 5 & 6 \\ 0 & 8 & 1 \\ 5 & 2 & 7 \end{bmatrix}}_{q=3} \left.\vphantom{\begin{bmatrix}1\\1\\1\\1\end{bmatrix}}\right\} p = 4 \quad V = \underbrace{\begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}}_{r=2} \left.\vphantom{\begin{bmatrix}1\\1\\1\end{bmatrix}}\right\} q = 3 \quad Z = UV = \begin{bmatrix} \boxed{13} & \\ & \\ & \end{bmatrix}$$

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

问题背景：基本知识

- 2个矩阵相乘

  - $U = \left.\underbrace{\begin{bmatrix} \boxed{2 \quad 1 \quad 3} \\ 9 \quad 5 \quad 6 \\ 0 \quad 8 \quad 1 \\ 5 \quad 2 \quad 7 \end{bmatrix}}_{q=3}\right\} p = 4 \quad V = \left.\underbrace{\begin{bmatrix} 1 & \boxed{4} \\ 2 & \boxed{5} \\ 3 & \boxed{6} \end{bmatrix}}_{r=2}\right\} q = 3 \quad Z = UV = \begin{bmatrix} 13 & \boxed{31} \\ & \\ & \\ & \end{bmatrix}$

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

问题背景：基本知识

- 2个矩阵相乘

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

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

问题背景：基本知识

- 2个矩阵相乘

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

问题背景：基本知识

- 2个矩阵相乘

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

问题背景：基本知识

- 2个矩阵相乘

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

- 矩阵乘法的时间复杂度
  - 计算1个数字：q次标量乘法

问题背景：基本知识

- 2个矩阵相乘

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

- 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：$\Theta(pqr)$

问题背景：基本知识

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

- 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：$p \times q \times r = 4 \times 3 \times 2 = 24$

问题背景：基本知识

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

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

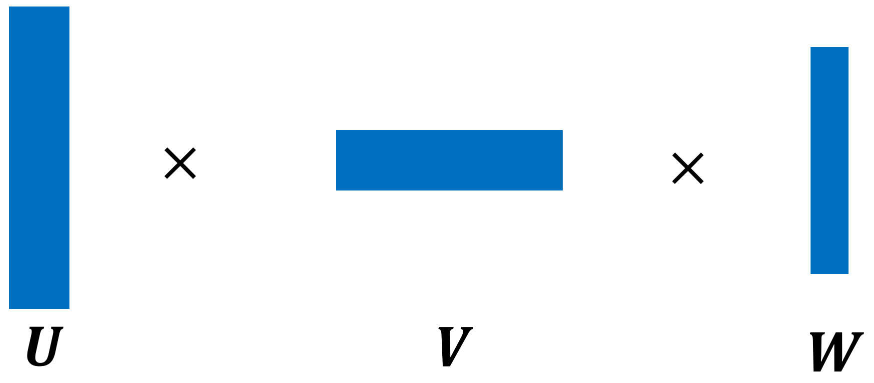
问题：顺序不同，效率是否明显不同?

问题背景：基本知识

- 3个矩阵相乘
  - 矩阵乘法结合率：$(\boldsymbol{UV})\boldsymbol{W} = \boldsymbol{U}(\boldsymbol{VW})$
  - 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$p = 40$，$q = 8$，$r = 30$，$s = 5$

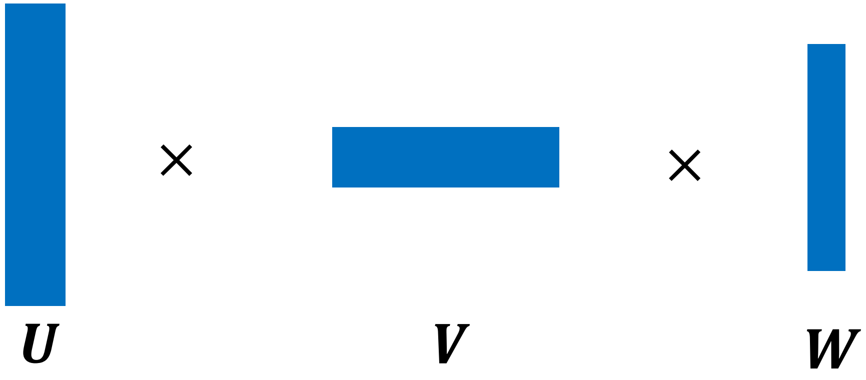


问题背景：基本知识

- 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$p = 40$，$q = 8$，$r = 30$，$s = 5$



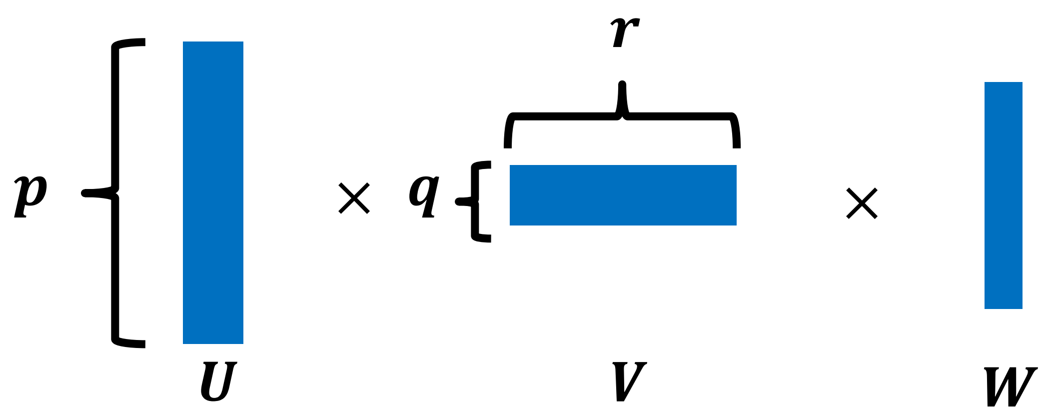
  - 按$(UV)W$计算，标量乘法次数：

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$p = 40$，$q = 8$，$r = 30$，$s = 5$



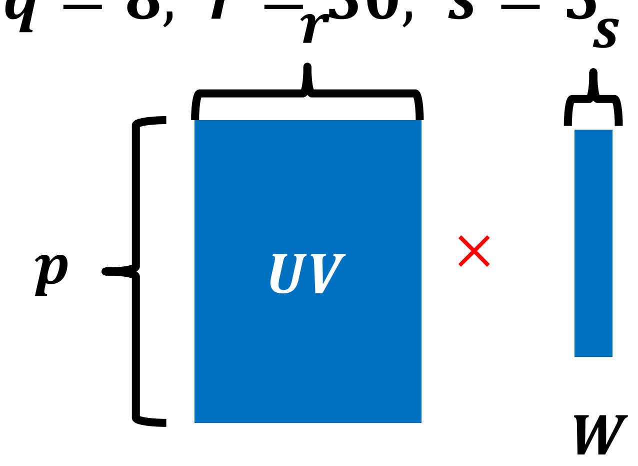
  - 按$(UV)W$计算，标量乘法次数：pqr

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$p = 40$，$q = 8$，$r = 30$，$s = 5$



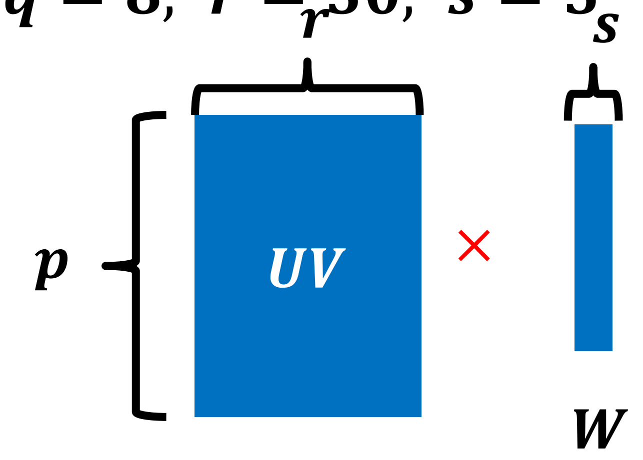
  - 按$(UV)W$计算，标量乘法次数：$pqr + prs$

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$p = 40$，$q = 8$，$r = 30$，$s = 5$



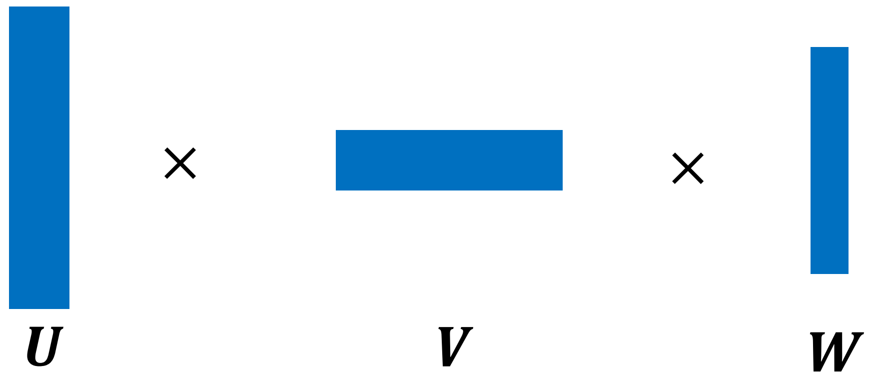
- 按$(UV)W$计算，标量乘法次数：$pqr + prs = 15600$

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$p = 40$，$q = 8$，$r = 30$，$s = 5$



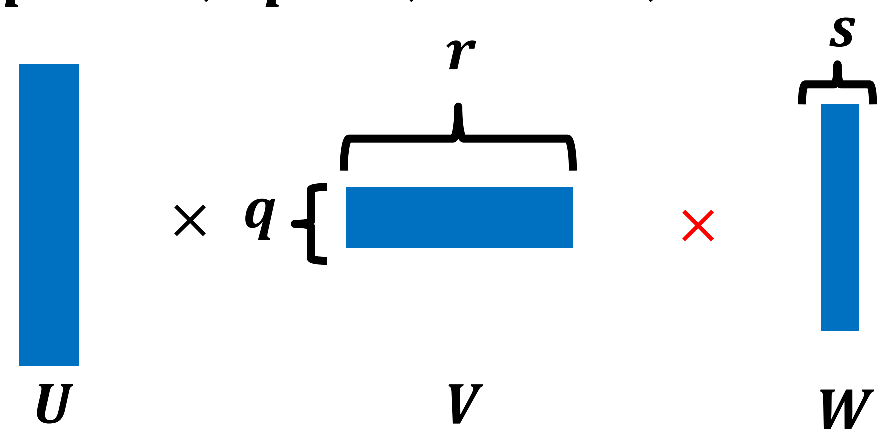
  - 按$(UV)W$计算，标量乘法次数：$pqr + prs = 15600$
  - 按$U(VW)$计算，标量乘法次数：

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$p = 40$，$q = 8$，$r = 30$，$s = 5$



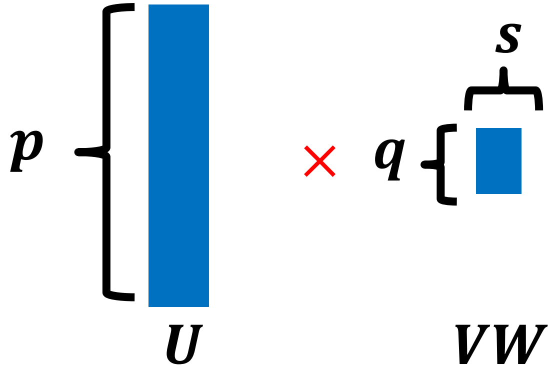
  - 按$(UV)W$计算，标量乘法次数：$pqr + prs = 15600$
  - 按$U(VW)$计算，标量乘法次数：qrs

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$p = 40$，$q = 8$，$r = 30$，$s = 5$



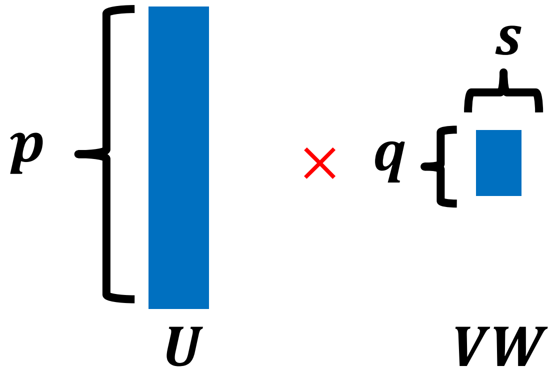
  - 按$(UV)W$计算，标量乘法次数：$pqr + prs = 15600$
  - 按$U(VW)$计算，标量乘法次数：$qrs + pqs$

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$p = 40$，$q = 8$，$r = 30$，$s = 5$



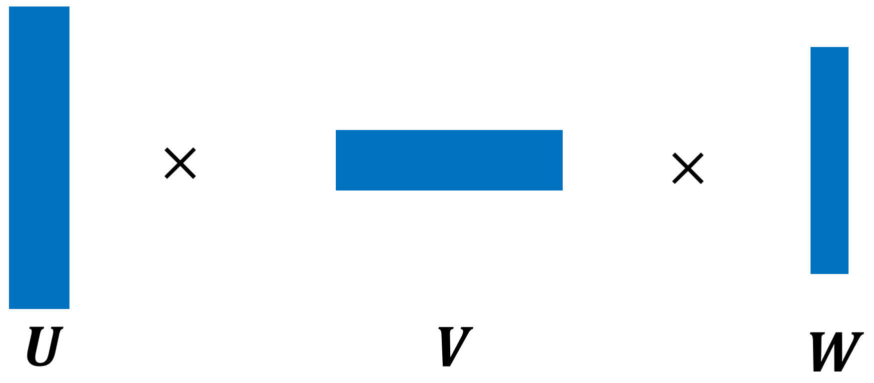
  - 按$(UV)W$计算，标量乘法次数：$pqr + prs = 15600$
  - 按$U(VW)$计算，标量乘法次数：$qrs + pqs = 2800$

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$p = 40$，$q = 8$，$r = 30$，$s = 5$

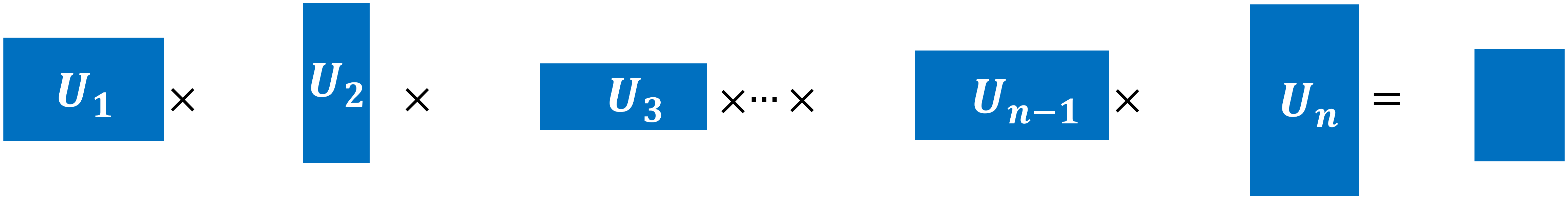


  - 按$(UV)W$计算，标量乘法次数：$pqr + prs = 15600$
  - 按$U(VW)$计算，标量乘法次数：$qrs + pqs = 2800$

差异显著

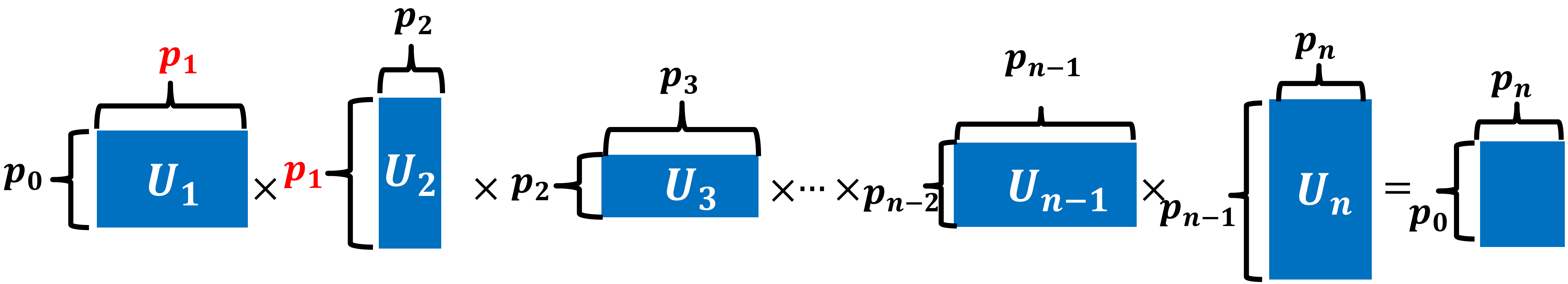
问题背景：基本知识

- n个矩阵相乘
  - 有一系列矩阵按顺序排列



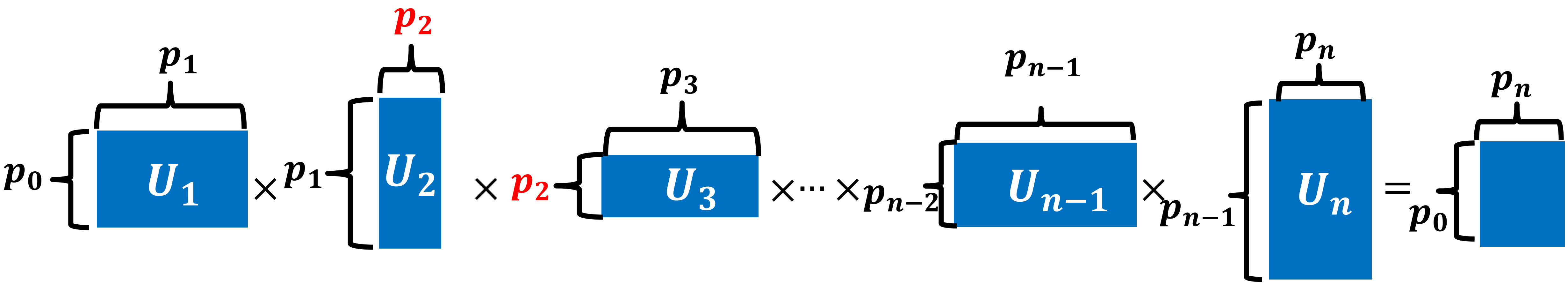
问题背景：基本知识

- 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



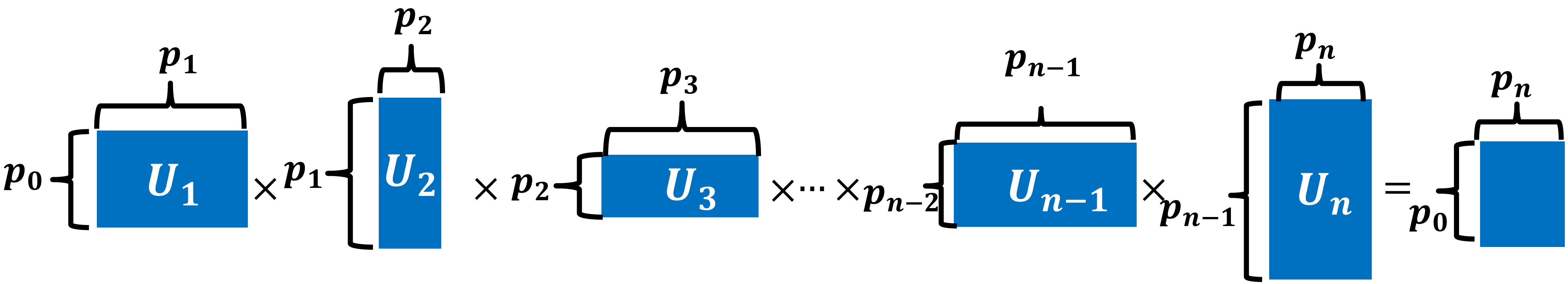
问题背景：基本知识

- 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

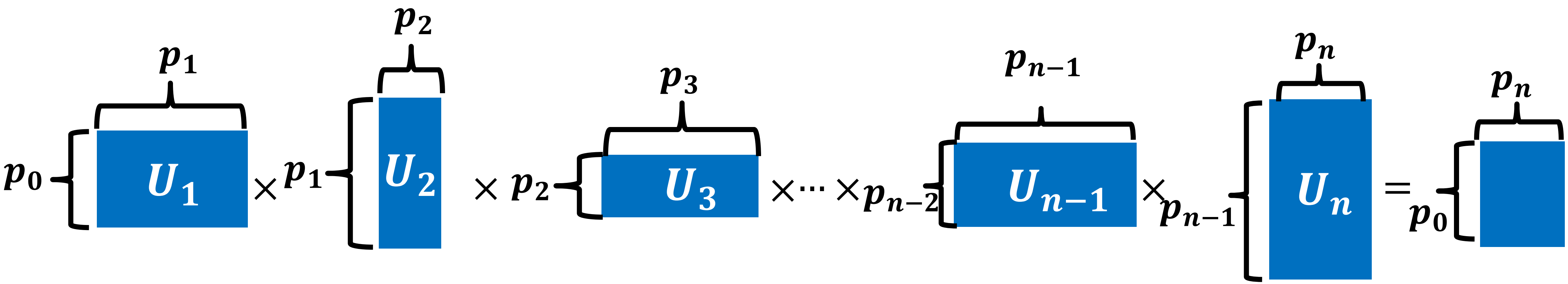
- n个矩阵相乘
  - 有一系列矩阵按顺序排列
  - 每个矩阵的行数=前一个矩阵的列数



- 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?

问题定义

矩阵链乘法问题

Matrix-chain Multiplication Problem

输入

- n个矩阵组成的矩阵链$U_{1..n} =< U_1, U_2, \dots, U_n >$
- 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

$U_1U_2U_3U_4$

第1次加括号

$(U_1)(U_2U_3U_4)$

$(U_1U_2)(U_3U_4)$

$(U_1U_2U_3)(U_4)$

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

$U_1U_2U_3U_4$

第1次加括号

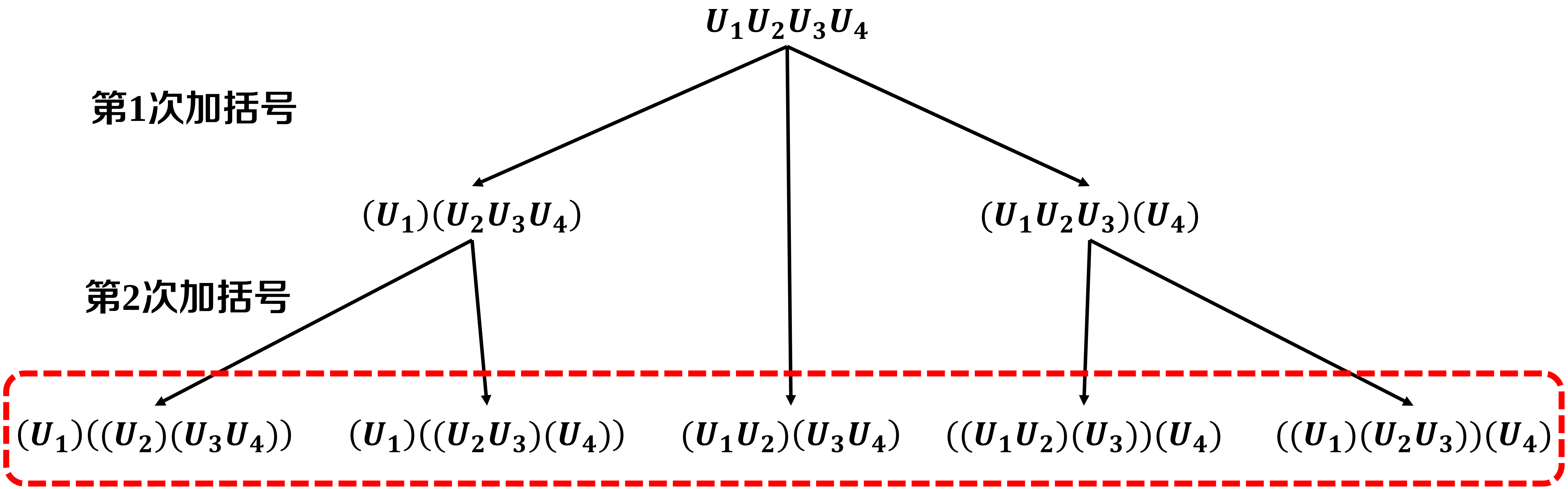
$(U_1)(U_2U_3U_4)$　　$(U_1U_2U_3)(U_4)$

第2次加括号

$(U_1)((U_2)(U_3U_4))$　　$(U_1)((U_2U_3)(U_4))$　　$(U_1U_2)(U_3U_4)$　　$((U_1U_2)(U_3))(U_4)$　　$((U_1)(U_2U_3))(U_4)$

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



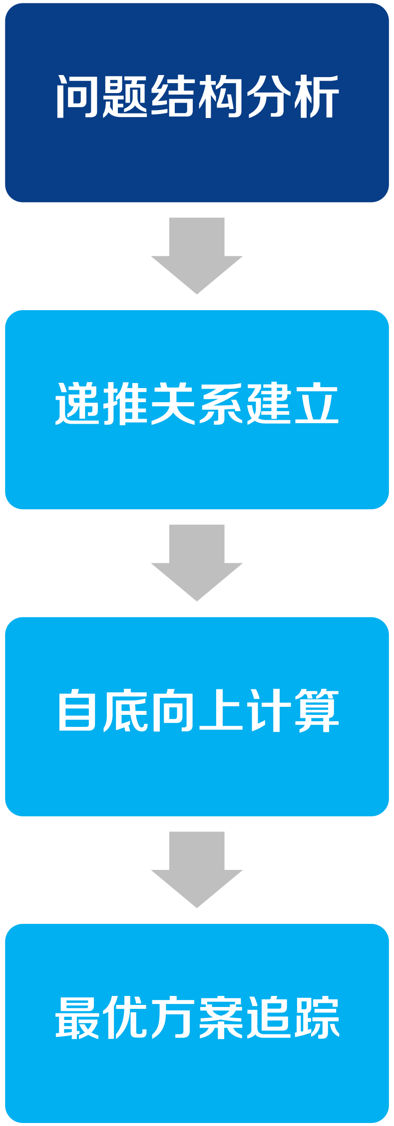
找到使标量乘法次数最小的加括号方式

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**

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

- 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**



递推关系建立：分析最优（子）结构

- 对矩阵链$U_{i..j}$，求解$D[i,j]$

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

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$D[i,j]$
  - 加一次括号

$$U_i \dots U_k \,|\, U_{k+1} \dots U_j$$

某位置$k(i \le k < j)$

$$(U_i \dots U_k) \times (U_{k+1} \dots U_j)$$

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$D[i,j]$

$$U_i \dots U_k \mid U_{k+1} \dots U_j$$

某位置$k (i \le k < j)$

$$(U_i \dots U_k) \times (U_{k+1} \dots U_j)$$

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$D[i,j]$

$$U_i \dots U_k \mid U_{k+1} \dots U_j$$

某位置$k(i \le k < j)$

$$(U_i \dots U_k) \times (U_{k+1} \dots U_j)$$

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$D[i,j]$

$$U_i \dots U_k \mid U_{k+1} \dots U_j$$

某位置$k(i \le k < j)$

$$(U_i \dots U_k) \times (U_{k+1} \dots U_j)$$

$p_{i-1} \times p_k$

$p_k \times p_j$

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$D[i,j]$

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

某位置$k (i \le k < j)$

$$(U_i \dots U_k) \times (U_{k+1} \dots U_j)$$

$p_{i-1} \times p_k$

$D[i,k]$

$p_k \times p_j$

$D[k+1,j]$

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$D[i,j]$

$$U_i \dots U_k \,|\, U_{k+1} \dots U_j$$

某位置$k(i \le k < j)$

$$(U_i \dots U_k) \times (U_{k+1} \dots U_j)$$

$p_{i-1}p_kp_j$次标量乘法

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

$$D[i,k] \qquad D[k+1,j]$$

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$D[i,j]$

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

某位置$k(i \le k < j)$

$$(U_i \dots U_k) \times (U_{k+1} \dots U_j)$$

$p_{i-1}p_kp_j$次标量乘法

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

$D[i,k]$ $\qquad$ $D[k+1,j]$

- 乘法次数：$D[i,k] + D[k+1,j] + p_{i-1}p_kp_j$

问题结构分析

递推关系建立

自底向上计算

最优方案追踪

递推关系建立：构造递推公式

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

问题结构分析

递推关系建立

自底向上计算

最优方案追踪

递推关系建立：构造递推公式

- 对每个位置$k(i \le k < j)$
  - 乘法次数：$D[i,k] + D[k+1,j] + p_{i-1}p_k p_j$

- 枚举所有k，得到递推式
  - $\boxed{D[i,j]} = \min\limits_{i \le k < j}\left(\boxed{D[i,k] + D[k+1,j]} + p_{i-1}p_k p_j\right)$

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

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

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

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

$j=1$	2	3	$\dots$	$\dots$	$n-1$	n	$D[i,j]$
0							$i=1$
	0						2
		0					3
			0				$\dots$
				0			$\dots$
					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
  - $D[i,j] = \min\limits_{i \le k < j} \left(\boxed{D[i,k]} + \boxed{D[k+1,j]} + p_{i-1}p_k p_j \right)$

$j=1$	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

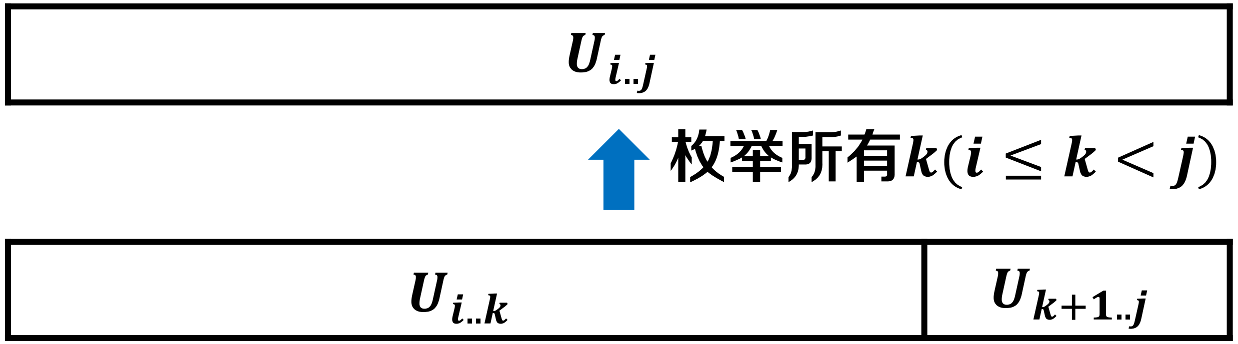
递推关系建立

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
  - $D[i,j] = \min\limits_{i \le k < j}(D[i,k] + D[k+1,j] + p_{i-1}p_k p_j)$
- 观察枚举过程



自底向上计算：确定计算顺序

- 递推公式
  - $D[i,j] = \min\limits_{i \le k < j} (D[i,k] + D[k+1,j] + p_{i-1}p_kp_j)$
- 观察枚举过程

长链

$U_{i..j}$

枚举所有$k(i \le k < j)$

$U_{i..k}$ | $U_{k+1..j}$

短链　短链

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
  - $D[i,j] = \min\limits_{i \le k < j}(D[i,k] + D[k+1,j] + p_{i-1}p_kp_j)$
- 观察枚举过程

长链

$U_{i..j}$

枚举所有$k(i \le k < j)$

$U_{i..k}$ | $U_{k+1..j}$

短链 短链

计算顺序：链长从小到大

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

$j=1$	2	3	$\dots$	$\dots$	$n-1$	n	$D[i,j]$
0							$i=1$
	0						2
		0					3
			0				$\dots$
				0			$\dots$
					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
  - $$D[i,j] = \min_{i \le k < j}(D[i,k] + D[k+1,j] + p_{i-1}p_kp_j)$$

$j=1$	2	3	$\dots$	$\dots$	$n-1$	n	$D[i,j]$
0							$i=1$
	0						2
		0		$j-i+1=2$			3
			0				$\dots$
				0			$\dots$
					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
  - $D[i,j] = \min\limits_{i \le k < j}(D[i,k] + D[k+1,j] + p_{i-1}p_kp_j)$

$j=1$	2	3	…	…	$n-1$	n	$D[i,j]$
0							$i=1$
	0						2
		0			$j-i+1=3$		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
  - $D[i,j] = \min\limits_{i \le k < j}(D[i,k] + D[k+1,j] + p_{i-1}p_kp_j)$

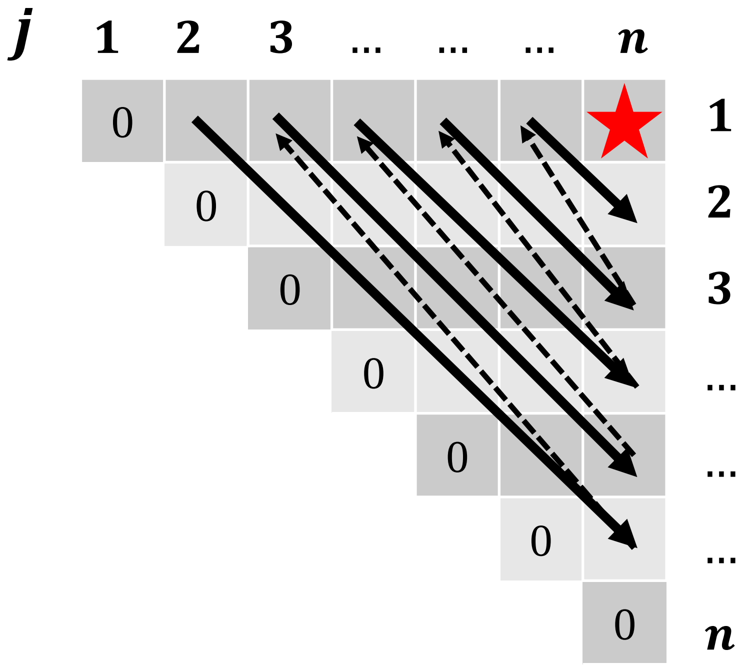
$j=1$	2	3	…	…	$n-1$	n	$D[i,j]$
0							$i=1$
	0						2
		0					3
			0				…
				0			…
					0		$n-1$
						0	n

矩阵链长：$j-i+1$

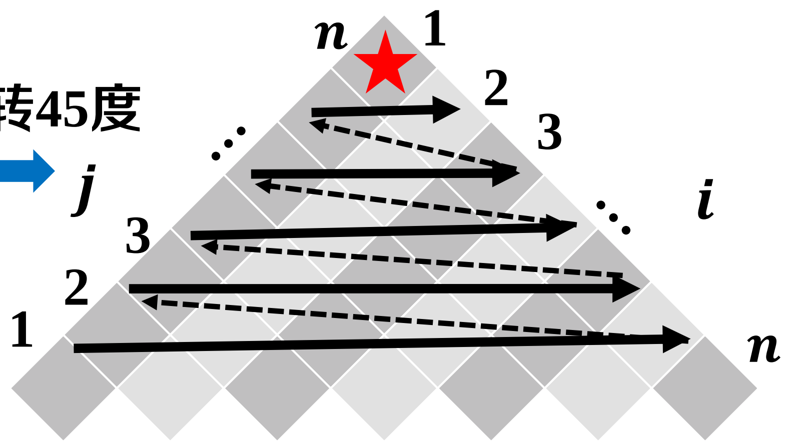
问题结构分析 → 递推关系建立 → **自底向上计算** → 最优方案追踪

自底向上计算：依次计算问题

- 递推公式
  - $D[i,j] = \min\limits_{i \le k < j}(D[i,k] + D[k+1,j] + p_{i-1}p_kp_j)$



逆时针旋转45度



问题结构分析

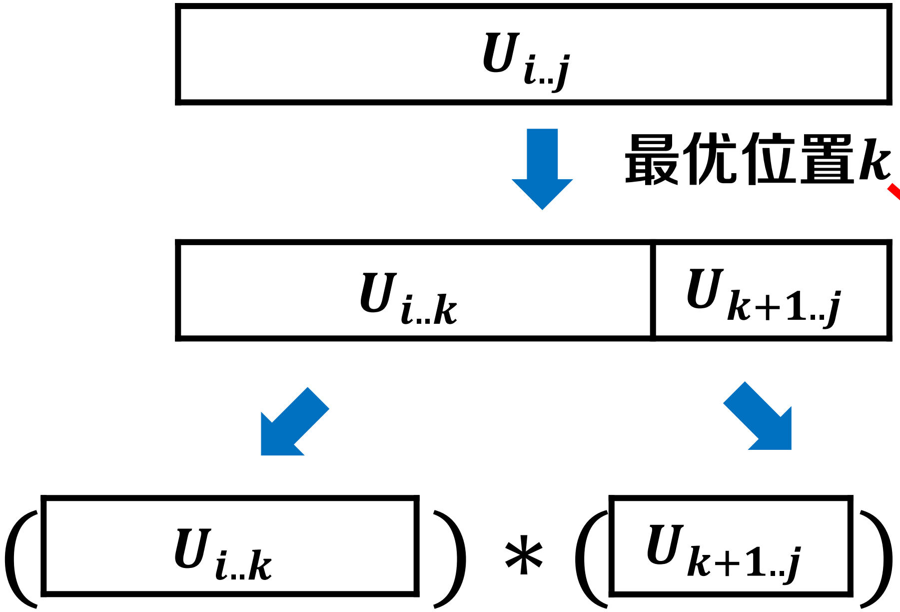
递推关系建立

自底向上计算

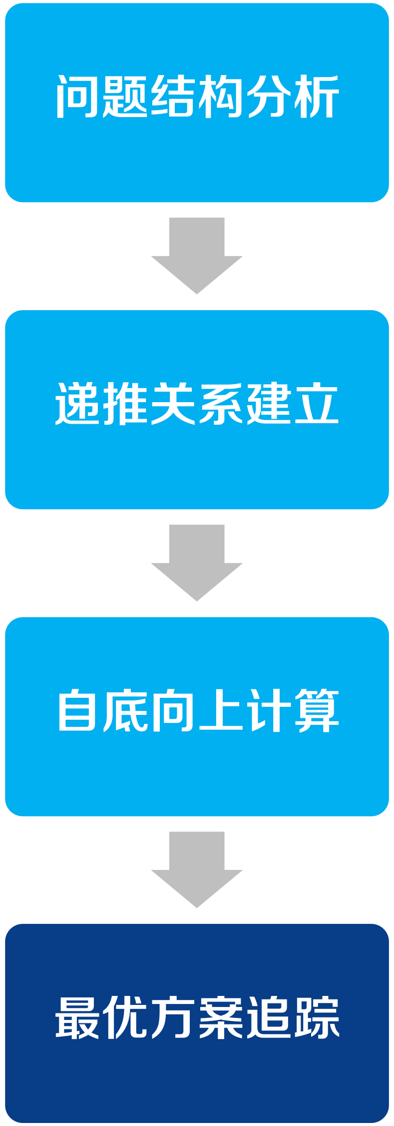
最优方案追踪

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



Rec	1	…	*j*	…	*n*
1					
…					
i			*k*		
…					
n					



最优方案追踪：输出最优方案

- 根据追踪数组，递归输出方案

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

Rec	$j=1$	2	$\dots$	k	$\dots$	$n-1$	n
$i=1$							k
2							
$\dots$							
$k+1$							
$\dots$							
$n-1$							
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

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

Rec	$j=1$	2	$\dots$	k	$\dots$	$n-1$	n
$i=1$							k
2							
$\dots$							
$k+1$							
$\dots$							
$n-1$							
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

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

Rec	$j=1$	2	$\dots$	$\boxed{k}$	$\dots$	$n-1$	n
$\boxed{i=1}$				$s \leftarrow$			k
2							
$\dots$							
$k+1$							
$\dots$							
$n-1$							
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

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

Rec	$j=1$	2	$\dots$	k	$\dots$	$n-1$	$\boxed{n}$
$i=1$				s			k
2							
$\dots$							
$\boxed{k+1}$							t
$\dots$							
$n-1$							
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

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

Rec	$j=1$	2	$\dots$	k	$\dots$	$n-1$	n
$i=1$				s			k
2							
$\dots$							
k							t
$\dots$							
$n-1$							
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

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

Rec	$j=1$	2	$\dots$	k	$\dots$	$n-1$	n
$i=1$				s			k
2							
$\dots$							
k							t
$\dots$							
$n-1$							
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

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

Rec	$j=1$	2	$\dots$	k	$\dots$	$n-1$	n
$i=1$				s			k
2							
$\dots$							
k							t
$\dots$							
$n-1$							
n							

问题结构分析

递推关系建立

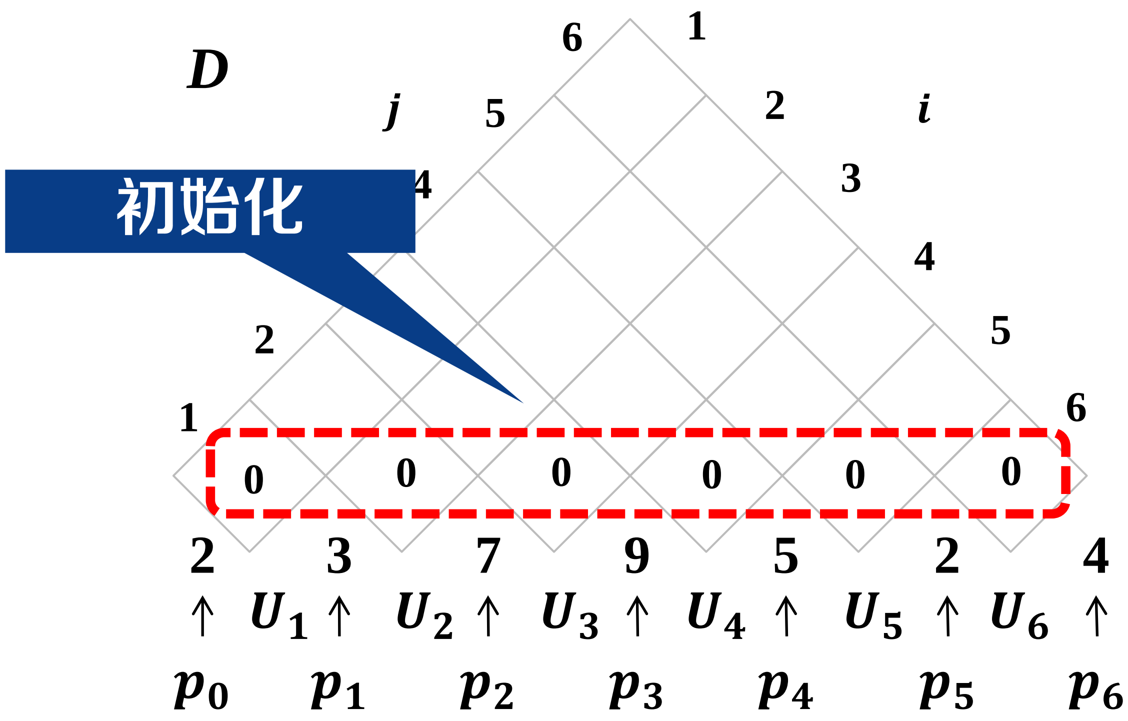
自底向上计算

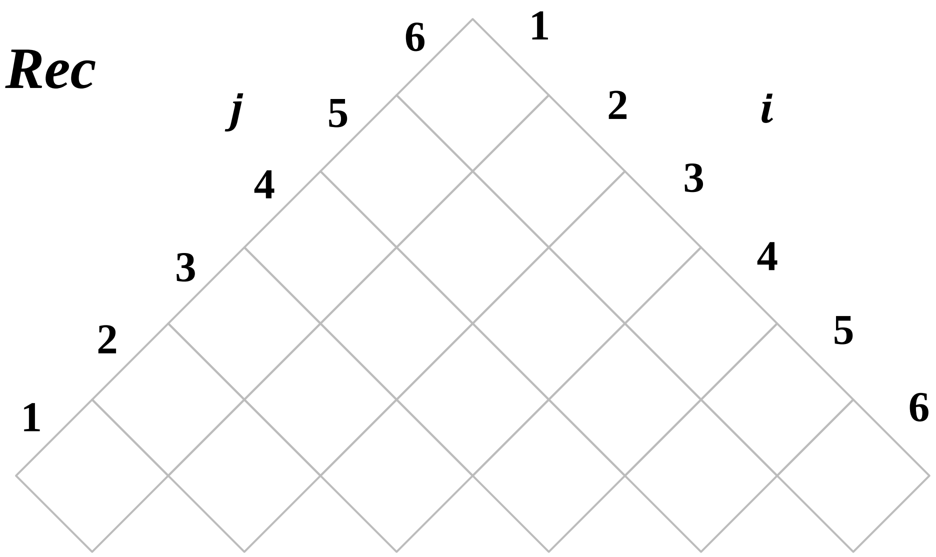
最优方案追踪

算法实例

- 给定矩阵链：$U_1U_2U_3U_4U_5U_6$
- 对应行列数

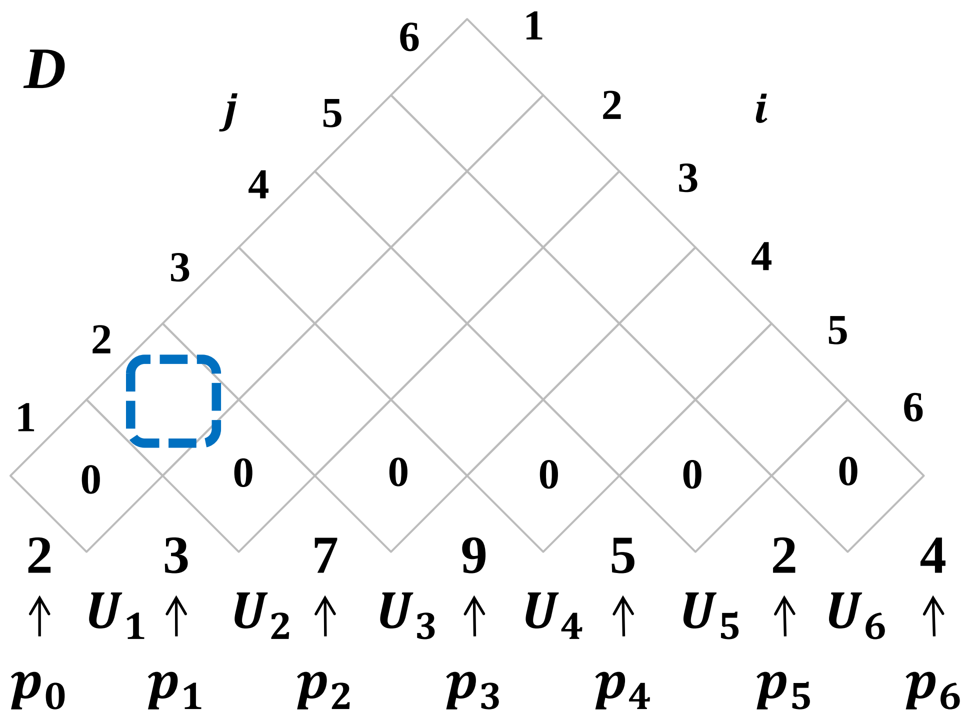
p_0	p_1	p_2	p_3	p_4	p_5	p_6
2	3	7	9	5	2	4

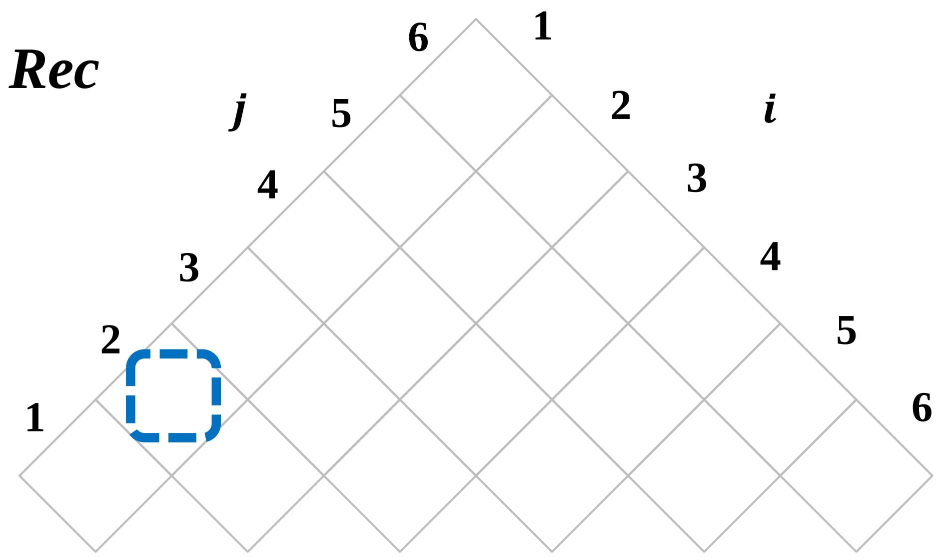




算法实例

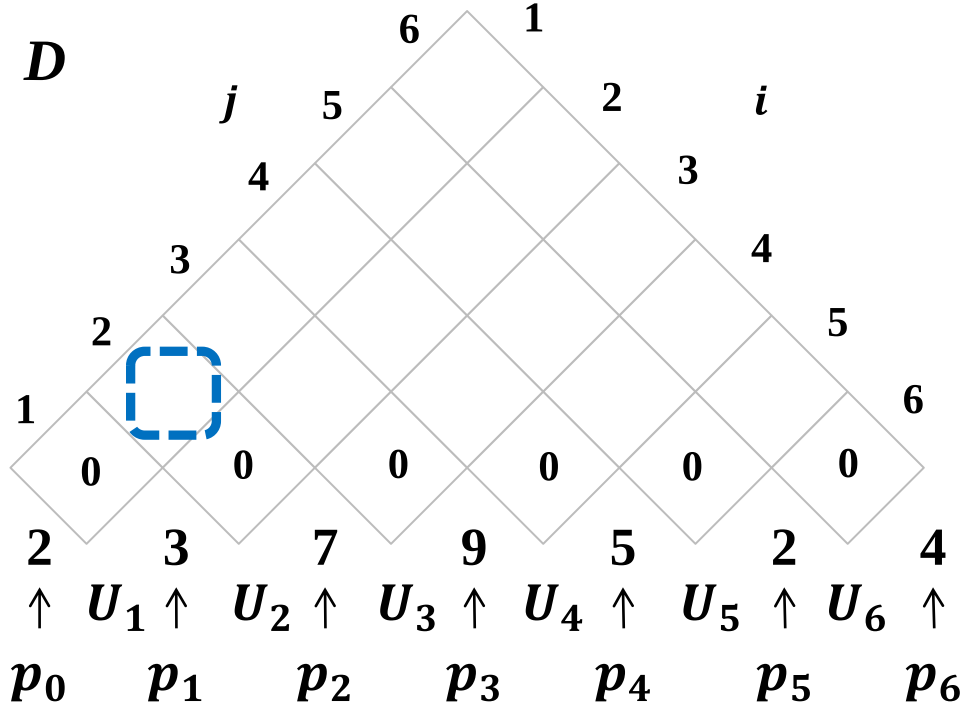
- $D[1,2] = \min\limits_{1\le k<2} (D[1,k] + D[k+1,2] + p_0 p_k p_2)$

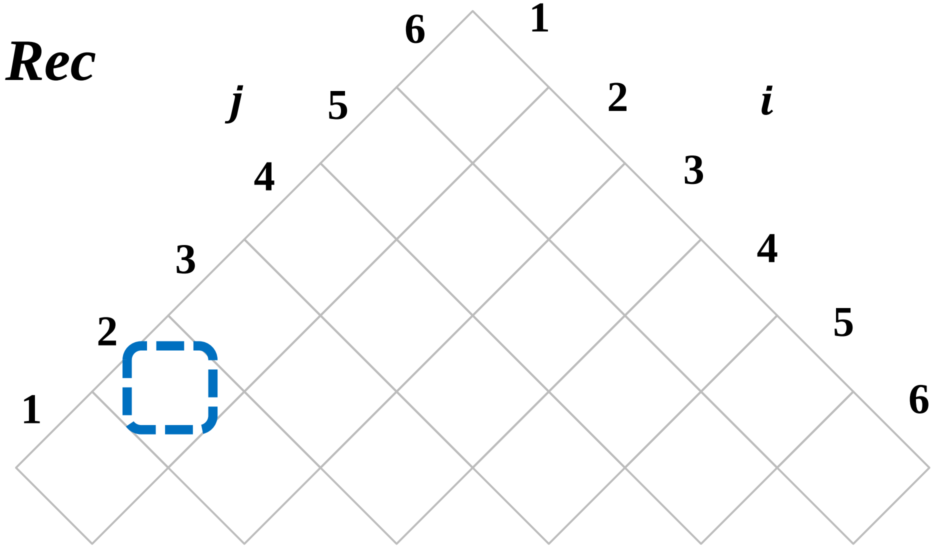




算法实例

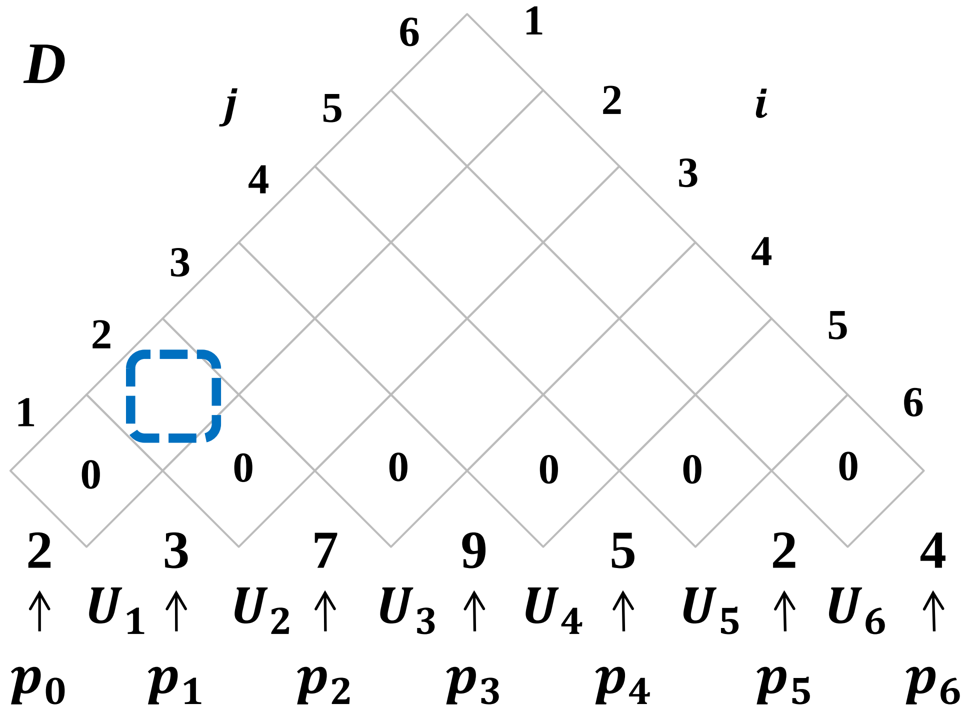
- $D[1,2] = \min_{1 \le k < 2} (D[1,k] + D[k+1,2] + p_0 p_k p_2)$

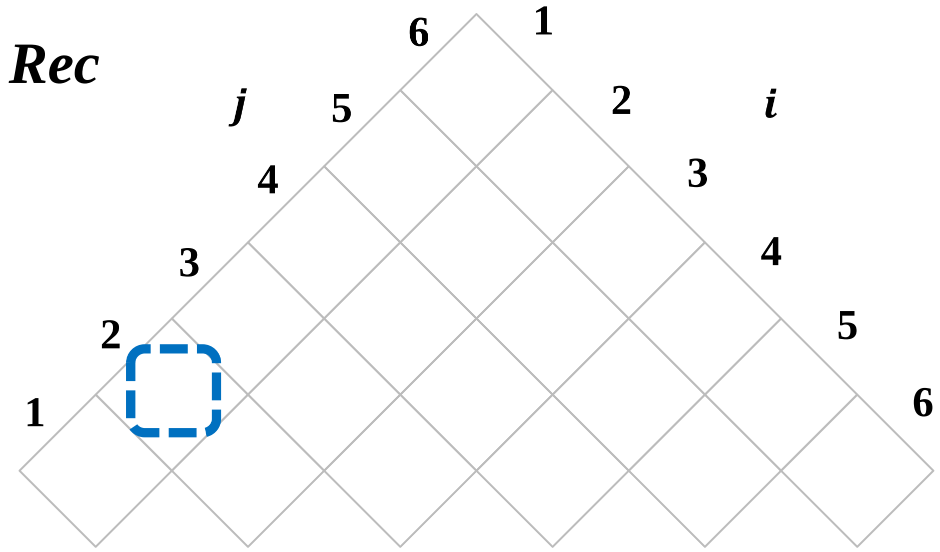




算法实例

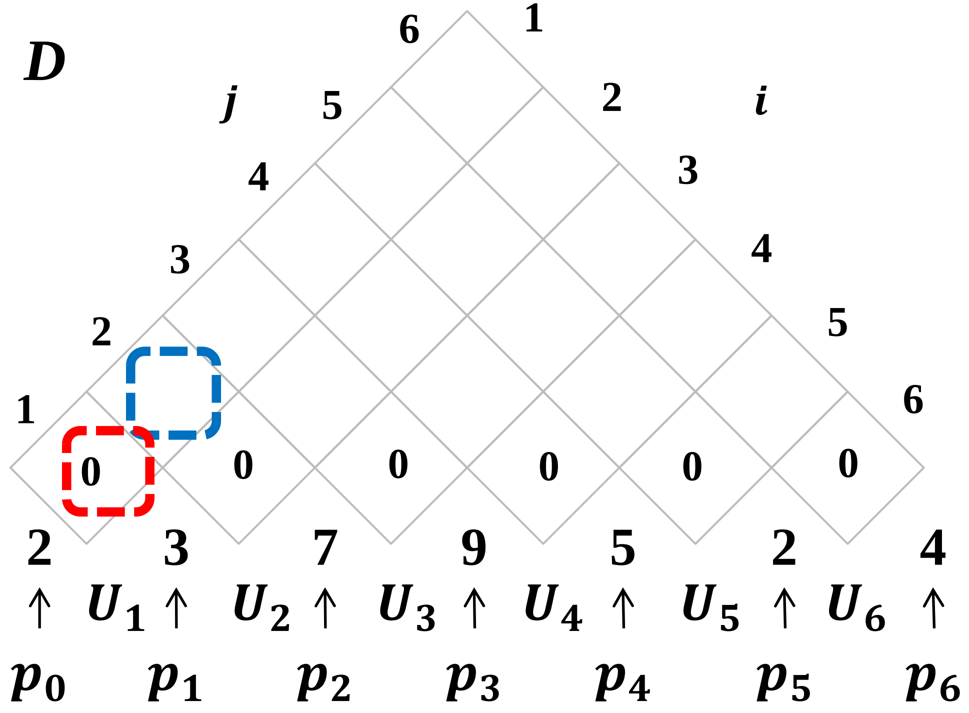
- $D[1,2] = \min_{1\le k<2}\left(D[1,1] + D[1+1,2] + p_0p_1p_2\right)$

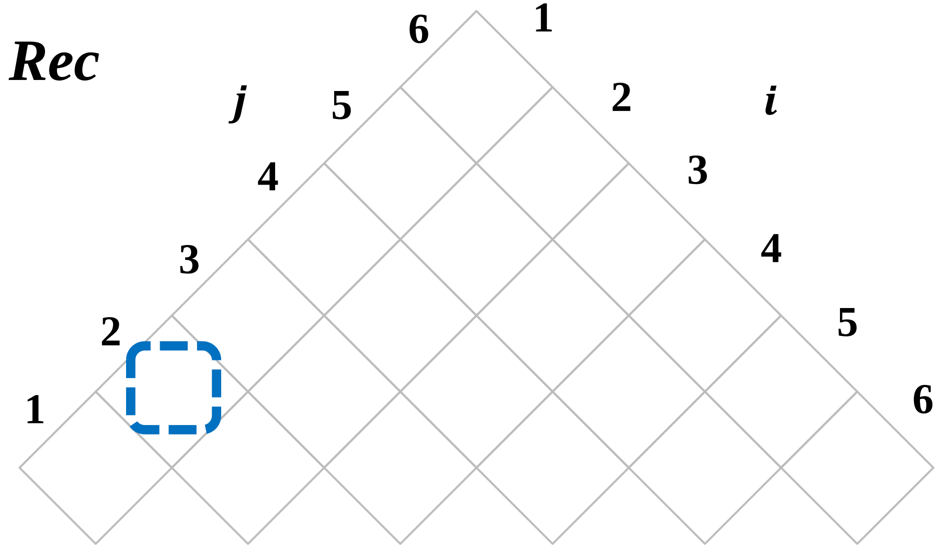




算法实例

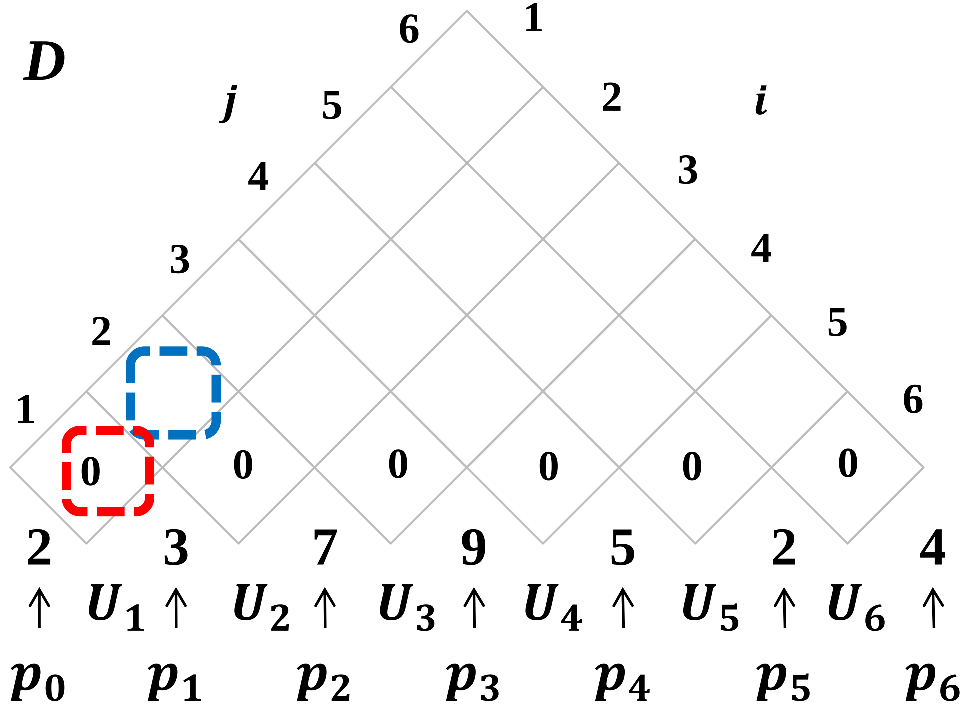
- $D[1,2] = \min\limits_{1\le k<2}\left(D[1,1] + D[1+1,2] + p_0p_1p_2\right)$

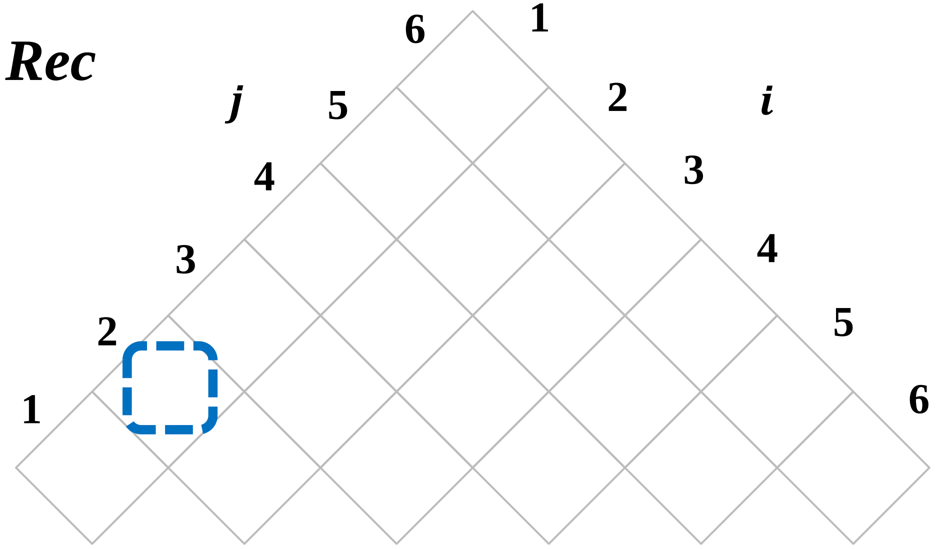




算法实例

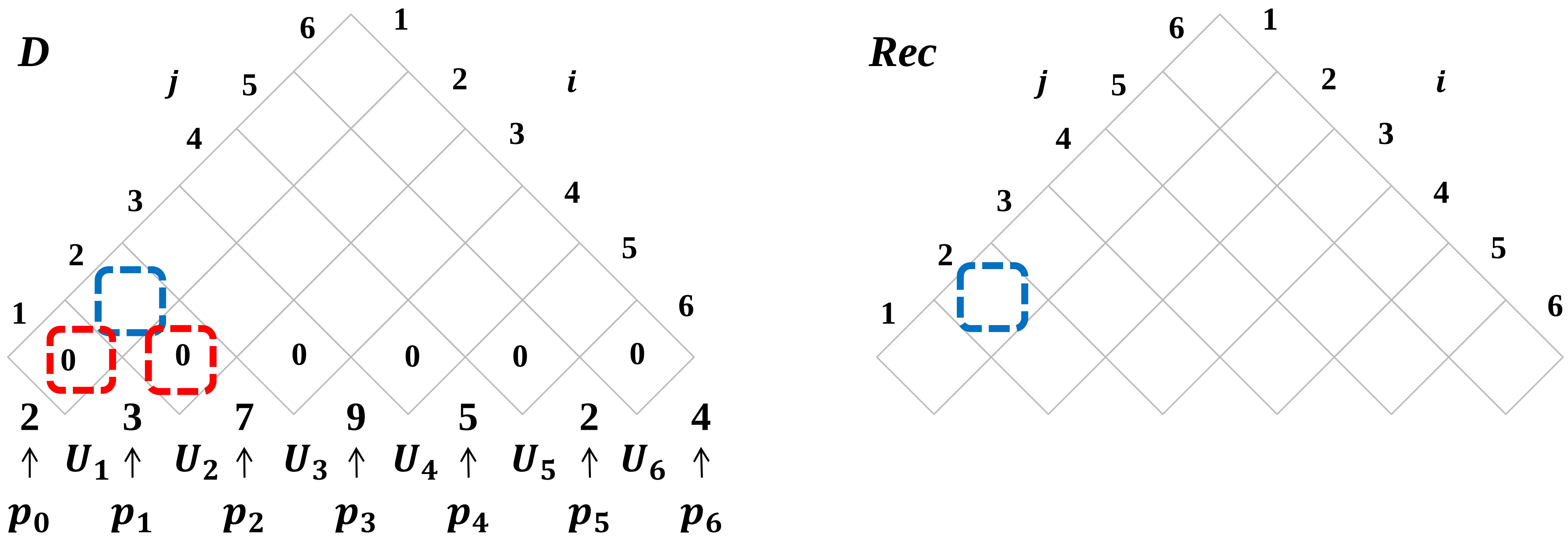
- $D[1,2] = \min_{1\le k<2}\left(0 + D[1+1,2] + p_0p_1p_2\right)$





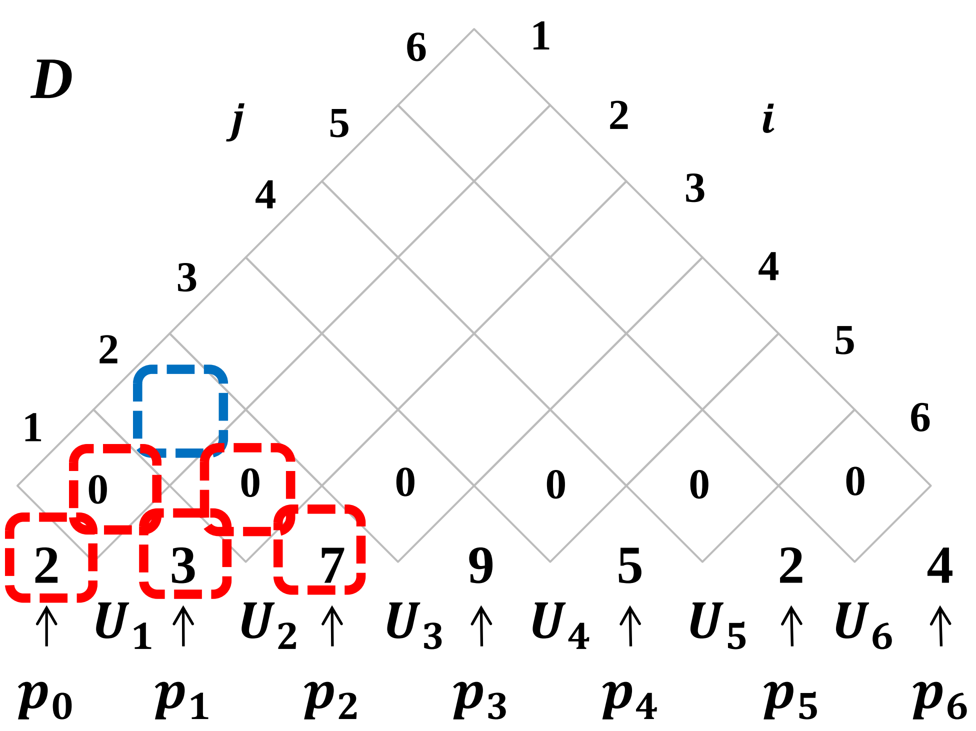
算法实例

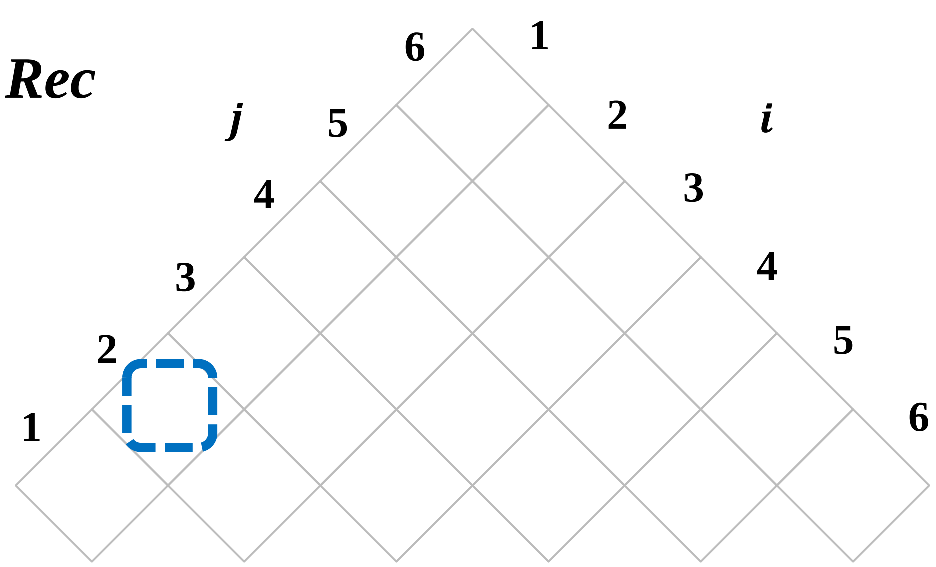
- $D[1,2] = \min_{1 \le k < 2}(0 + \color{red}{D[1+1,2]} + p_0 p_1 p_2)$



算法实例

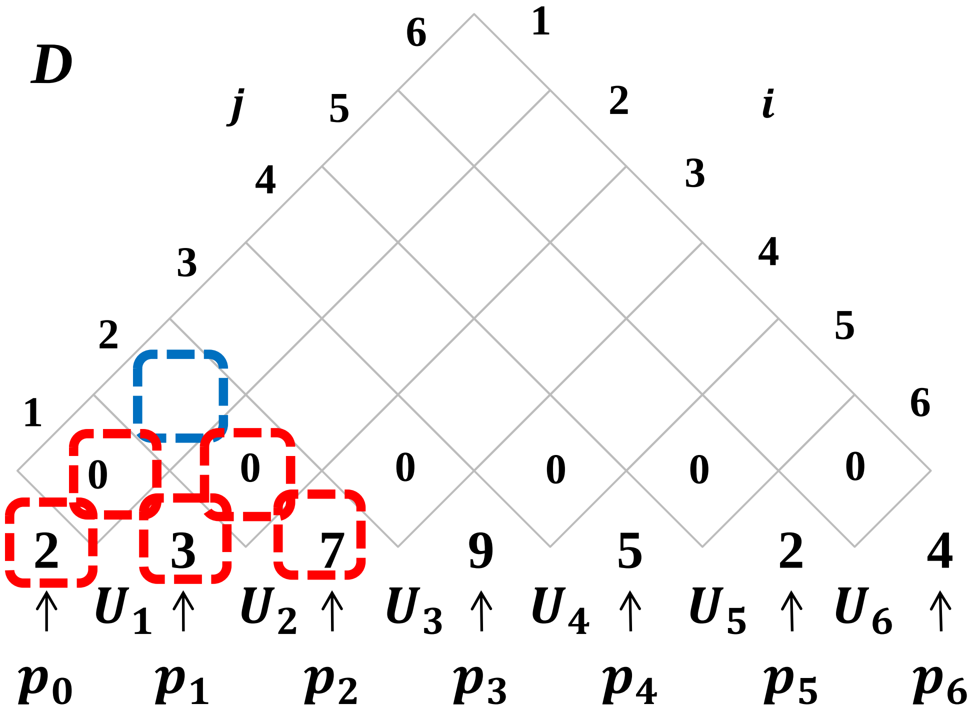
- $D[1,2] = \min\limits_{1\le k<2}(0 + 0 + p_0 p_1 p_2)$

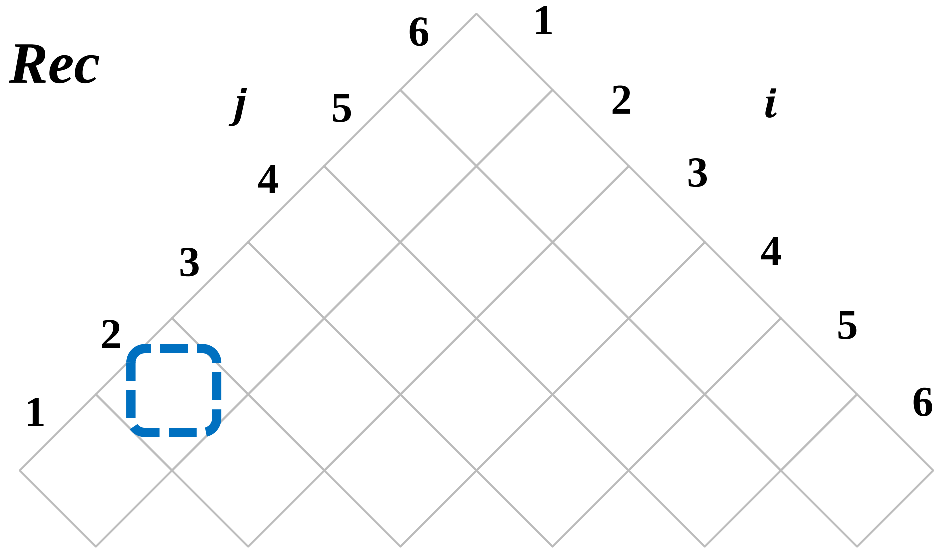




算法实例

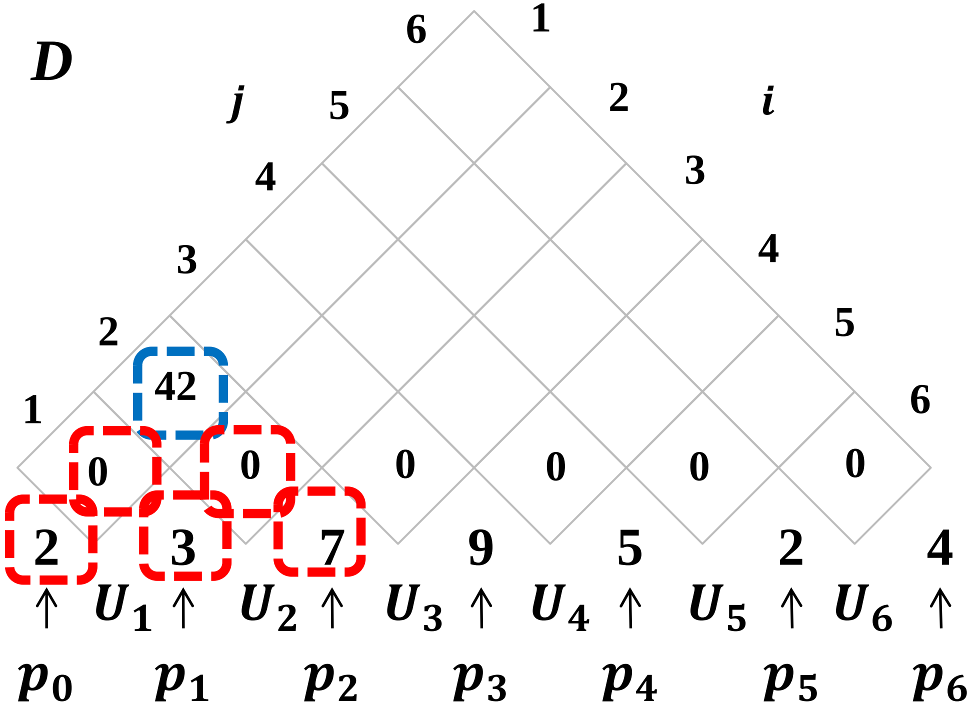
- $\boldsymbol{D}[1,2] = \min\limits_{1\le k<2}(0+0+2\times 3\times 7)$

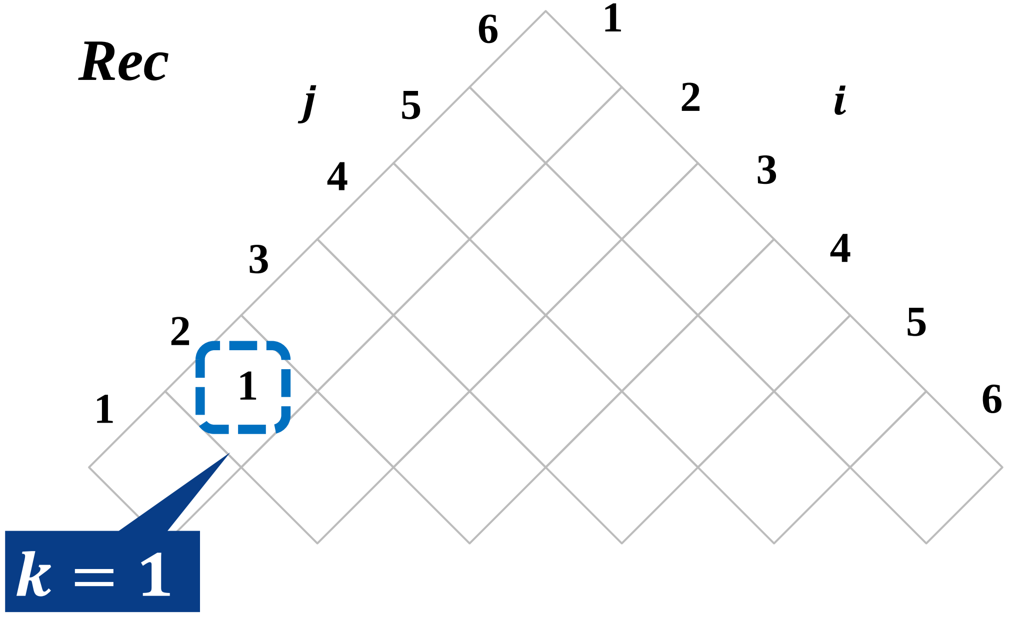




算法实例

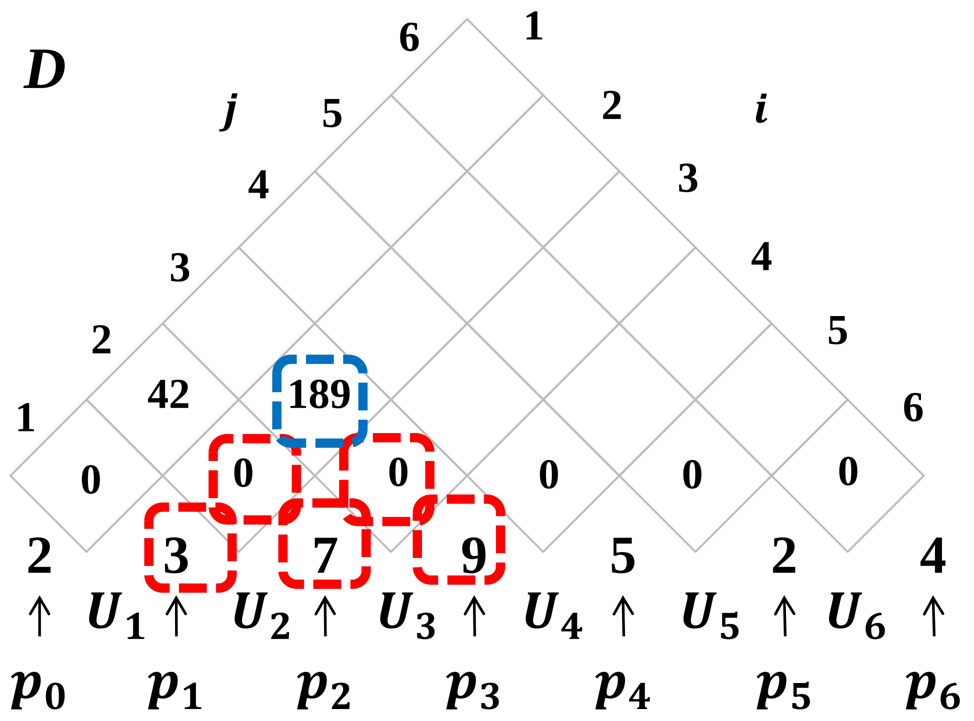
- $D[1,2] = \min\limits_{1\le k<2}(0+0+2\times3\times7) = 42$

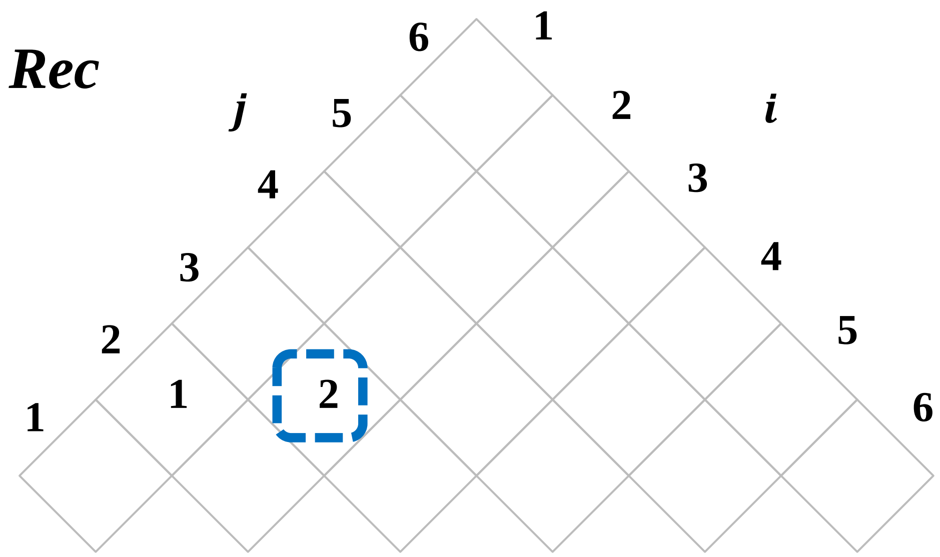




算法实例

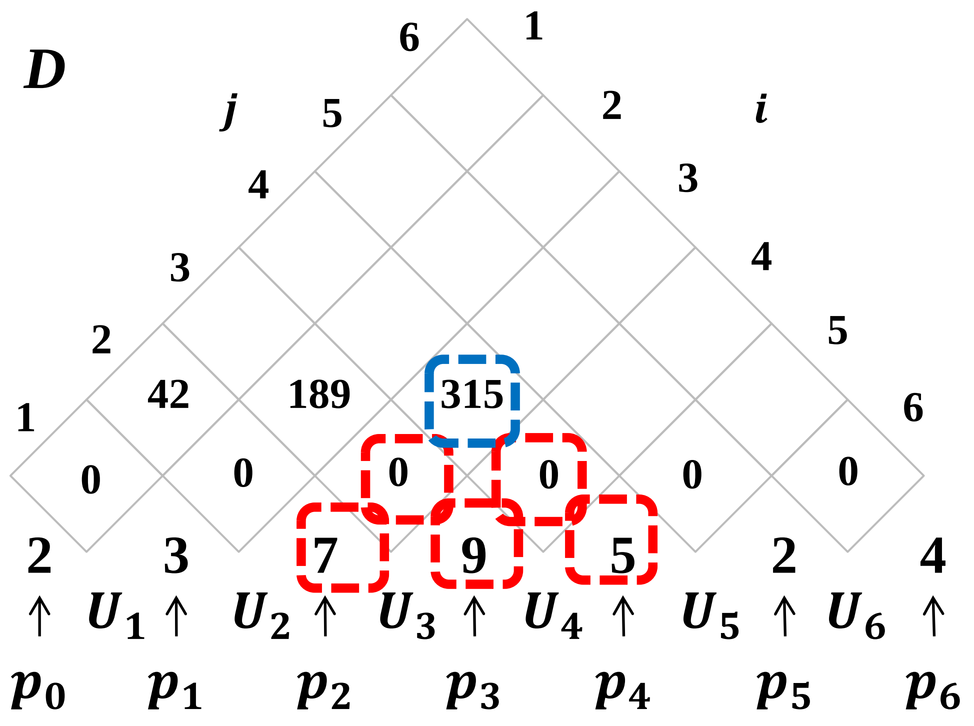
- $D[2,3] = \min\limits_{2\le k<3}(D[2,k] + D[k+1,3] + p_1 p_k p_3) = 189$

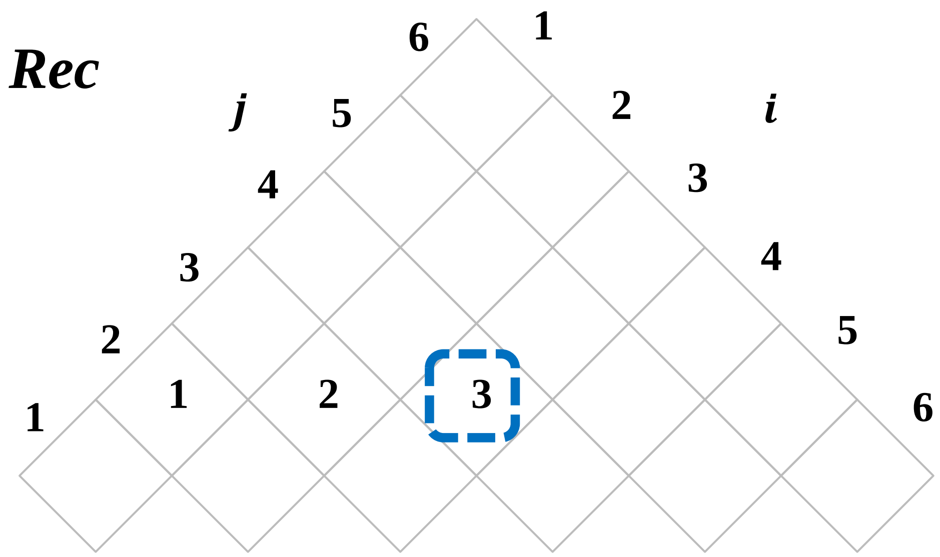




算法实例

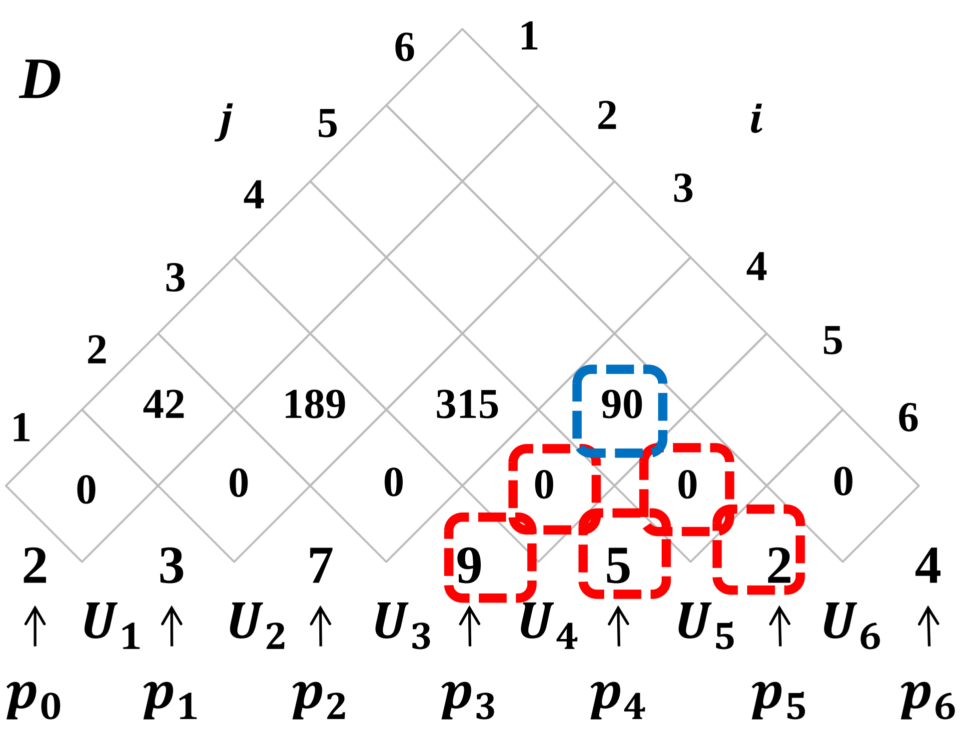
- $D[3,4] = \min\limits_{3\le k<4}(D[3,k] + D[k+1,4] + p_2 p_k p_4) = 315$

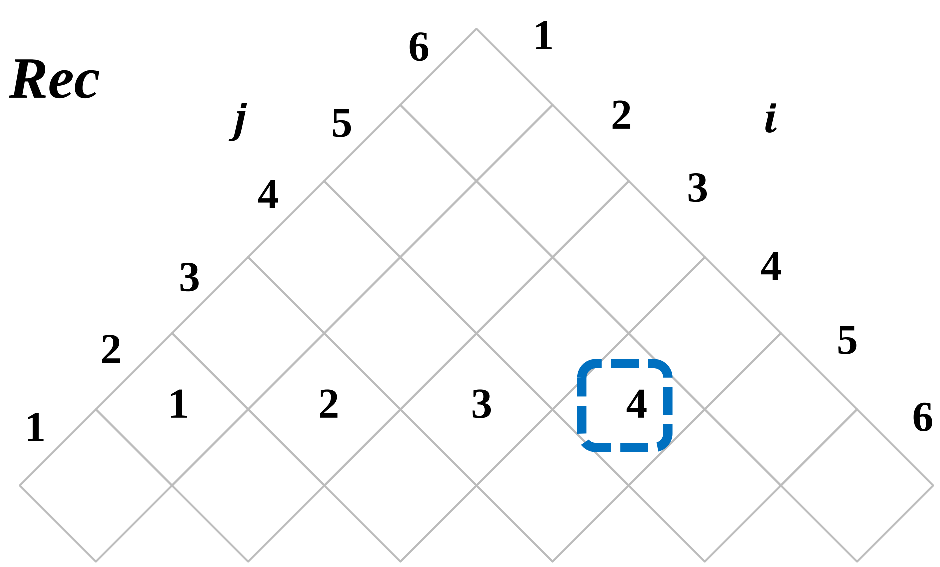




算法实例

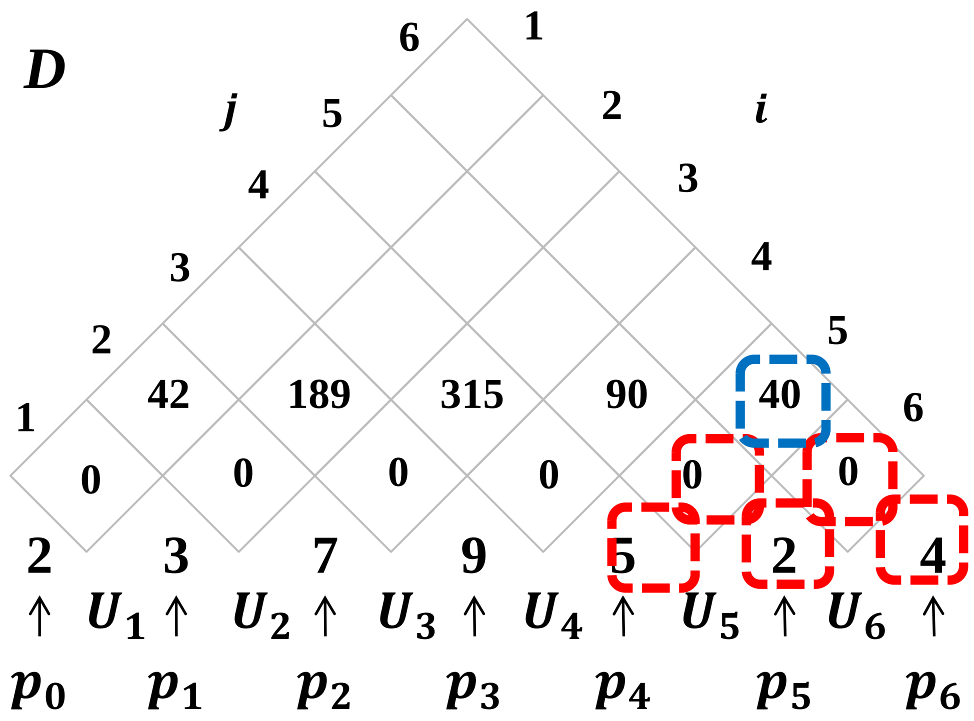
- $D[4,5] = \min_{4\le k<5} (D[4,k] + D[k+1,5] + p_3 p_k p_5) = 90$

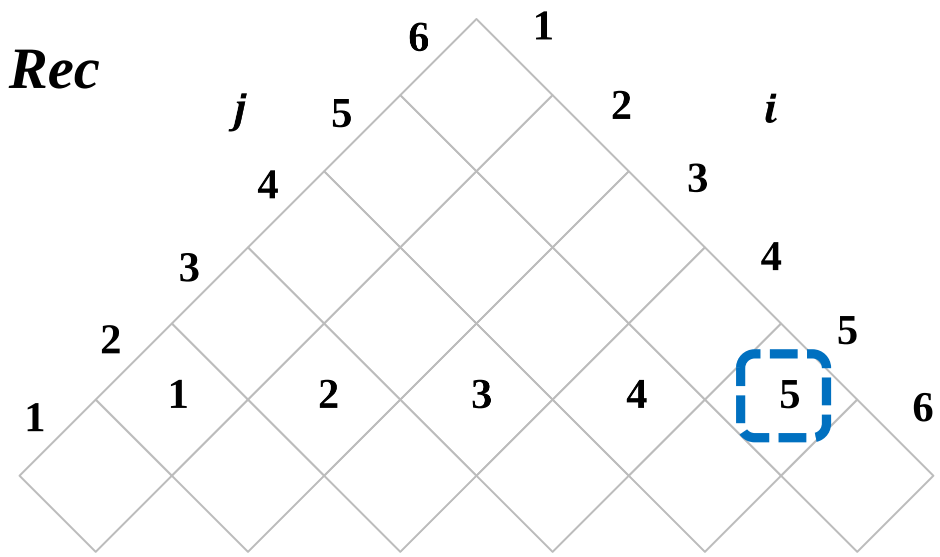




算法实例

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

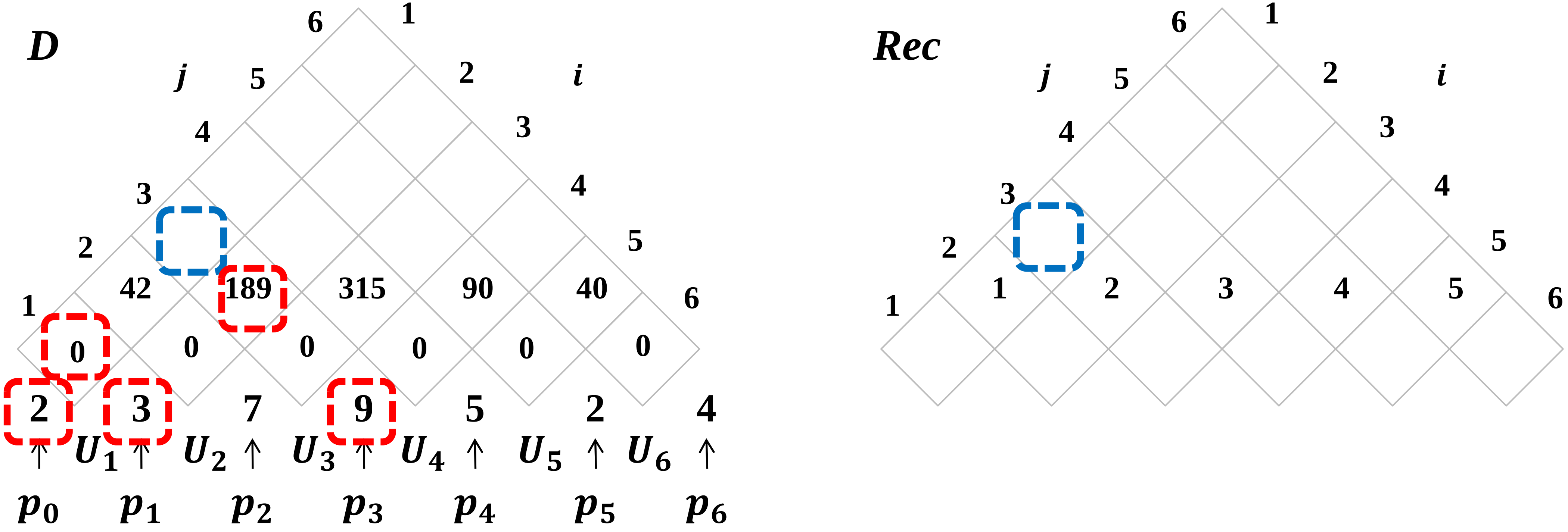




算法实例

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

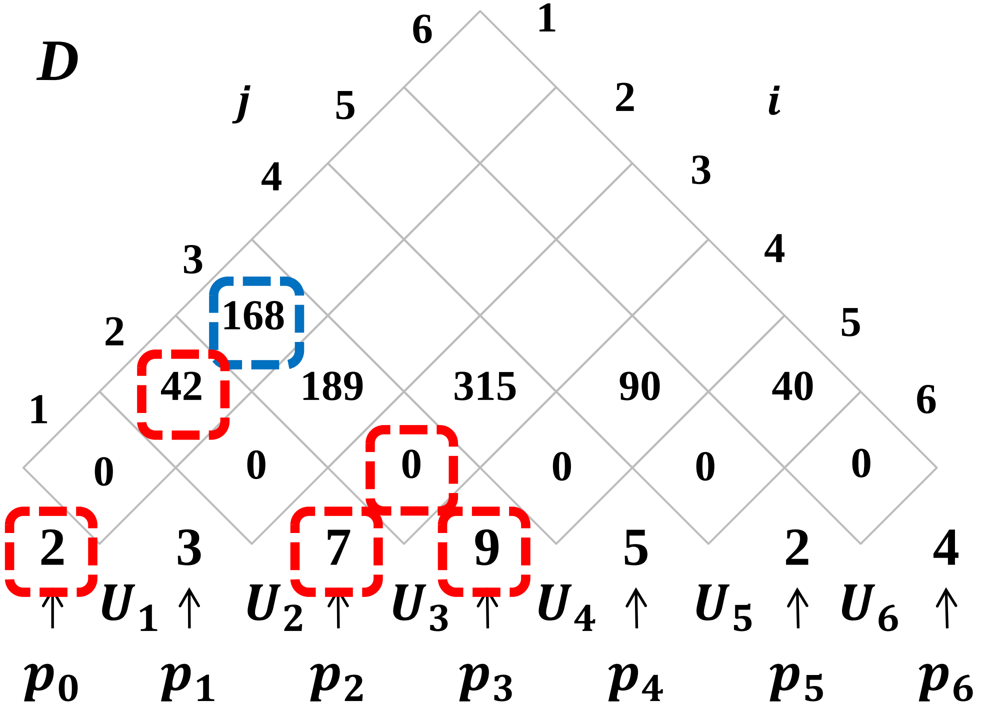
$$= \min \begin{cases} D[1,1] + D[2,3] + p_0 p_1 p_3 = 243 \\ D[1,2] + D[3,3] + p_0 p_2 p_3 = \end{cases}$$

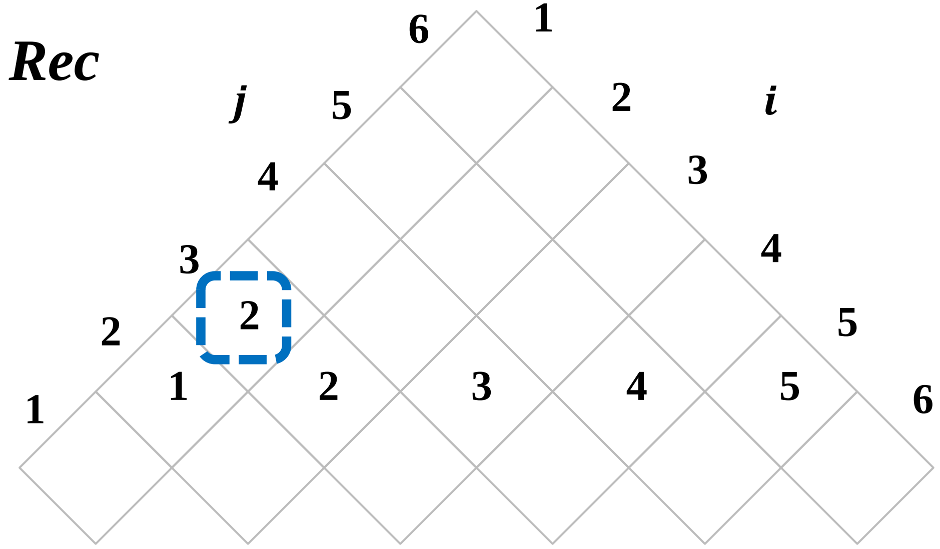


算法实例

- $D[1,3] = \min\limits_{1\le k<3}(D[1,k] + D[k+1,3] + p_0p_kp_3)$

$$= \min\begin{cases} D[1,1] + D[2,3] + p_0p_1p_3 = 243 \\ D[1,2] + D[3,3] + p_0p_2p_3 = 168 \end{cases}$$

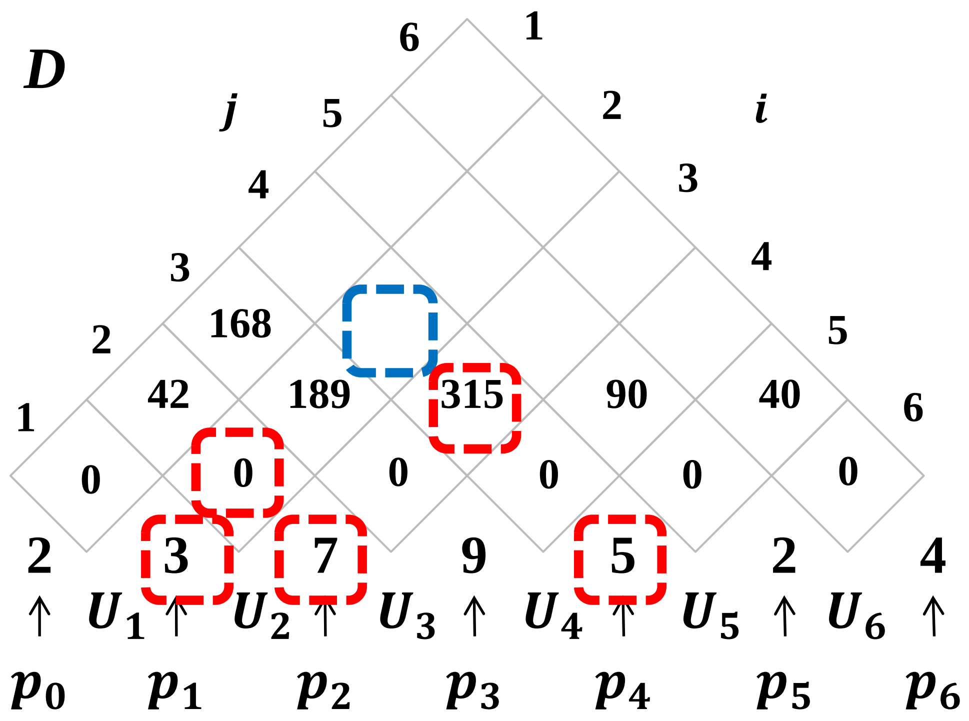


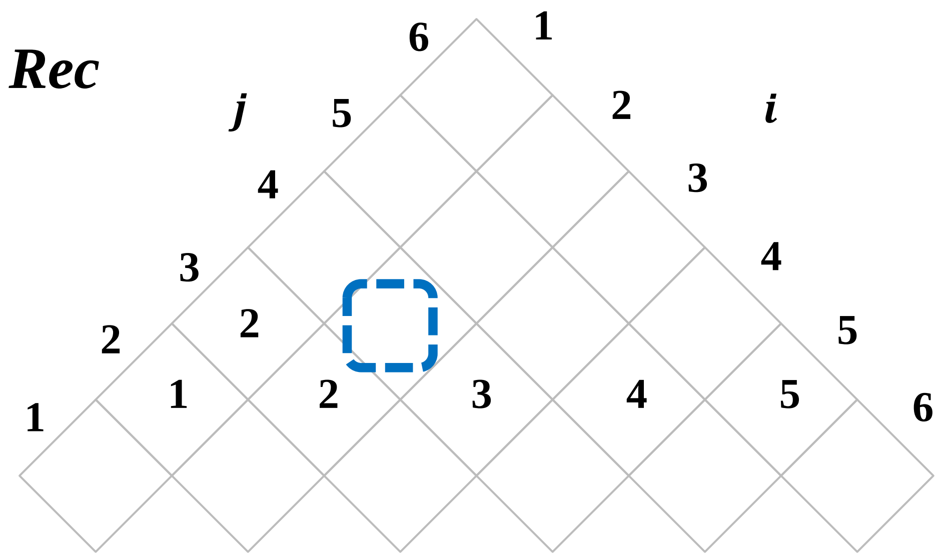


算法实例

- $D[2,4] = \min\limits_{2\le k<4}(D[2,k] + D[k+1,4] + p_1p_kp_4)$

$$= \min\begin{cases} D[2,2] + D[3,4] + p_1p_2p_4 = 420 \\ D[2,3] + D[4,4] + p_1p_3p_4 = \end{cases}$$

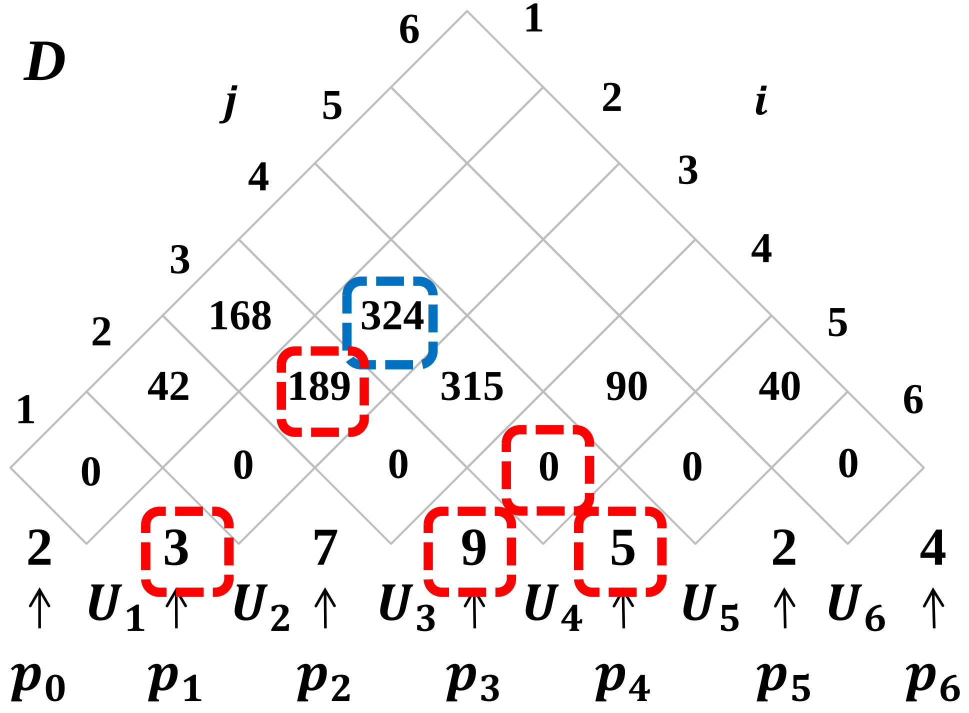


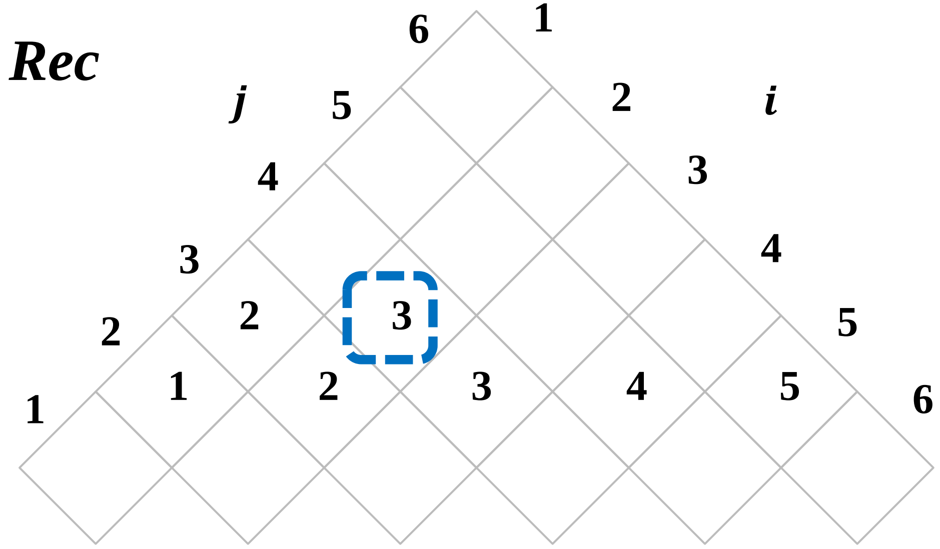


算法实例

- $D[2,4] = \min\limits_{2 \le k < 4} (D[2,k] + D[k+1,4] + p_1 p_k p_4)$

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

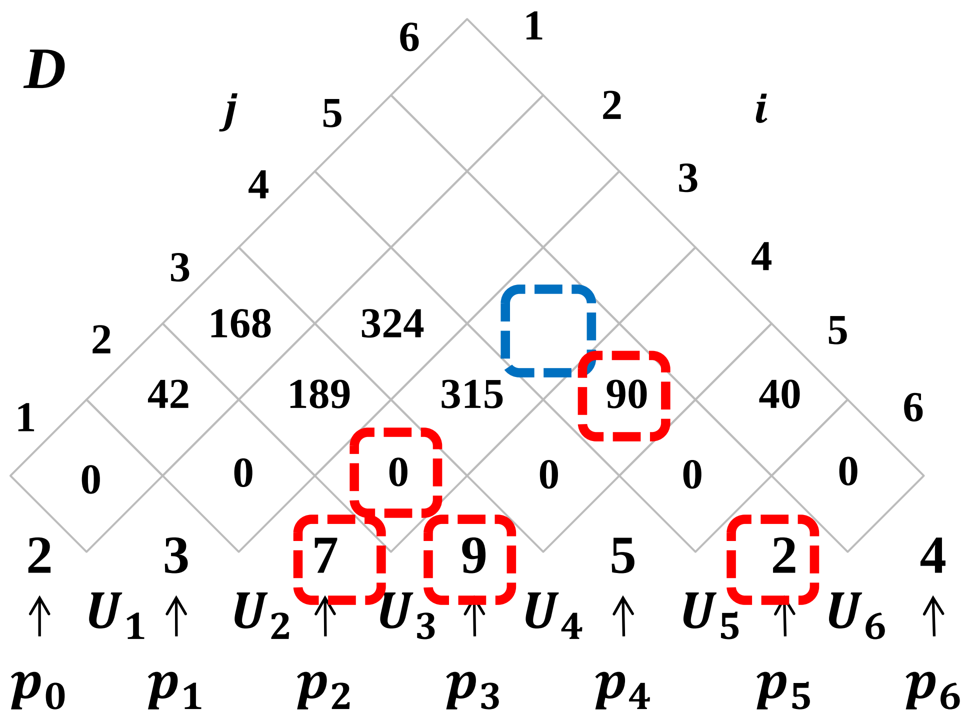


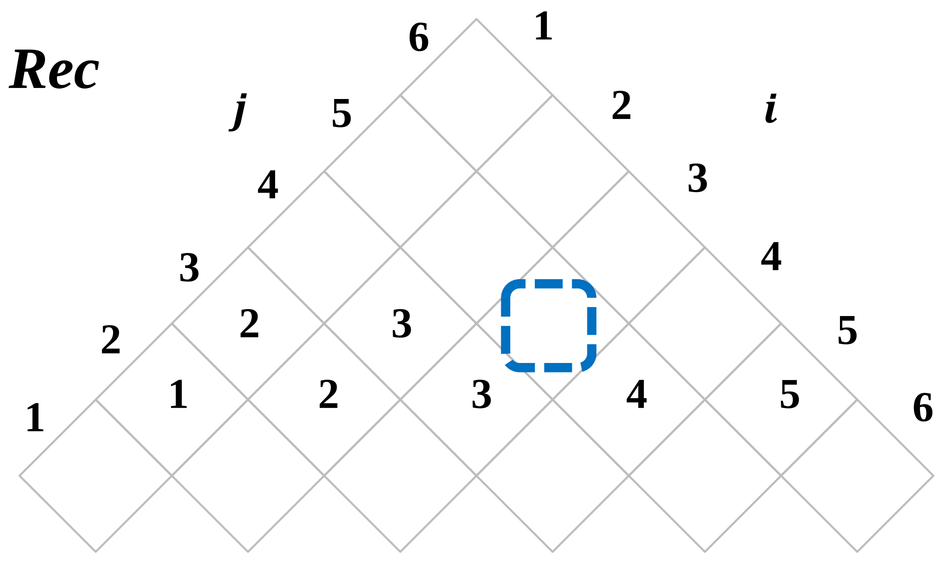


算法实例

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

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

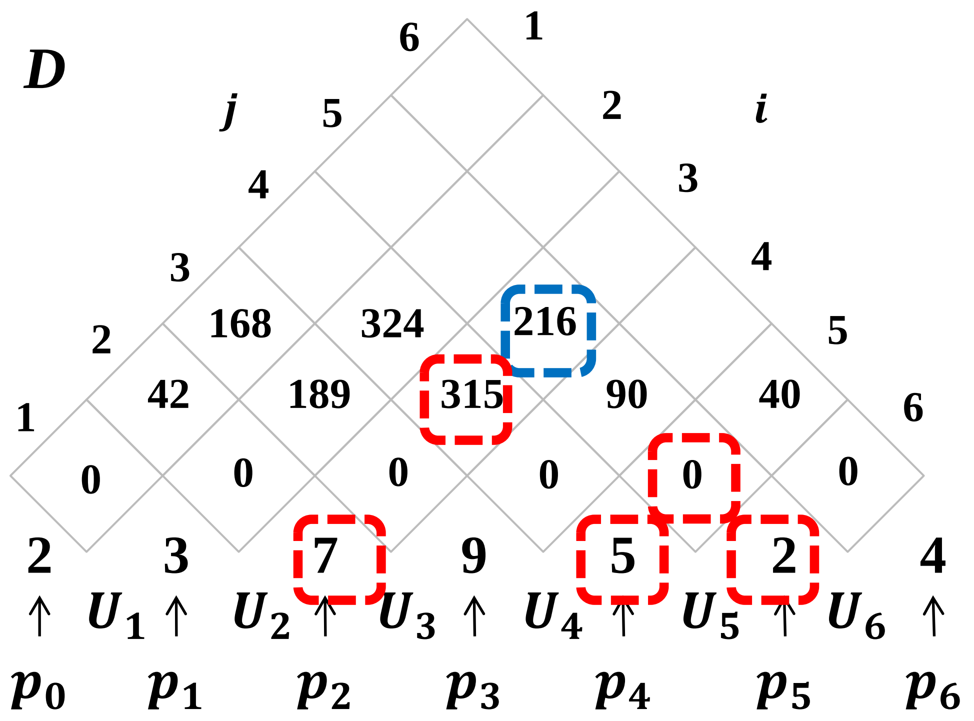


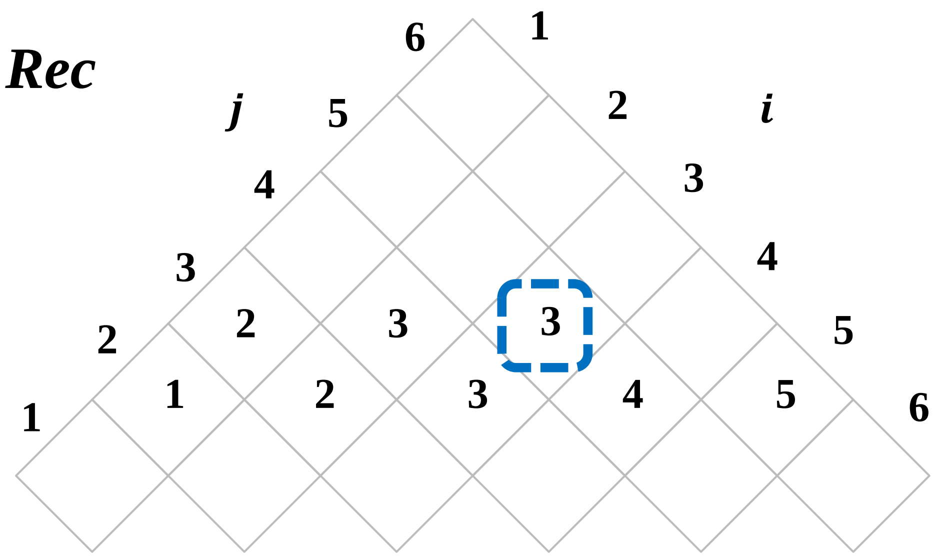


算法实例

- $D[3,5] = \min\limits_{3\le k<5}(D[3,k] + D[k+1,5] + p_2p_kp_5)$

$$= \min\begin{cases} D[3,3] + D[4,5] + p_2p_3p_5 = 216 \\ D[3,4] + D[5,5] + p_2p_4p_5 = 385 \end{cases}$$

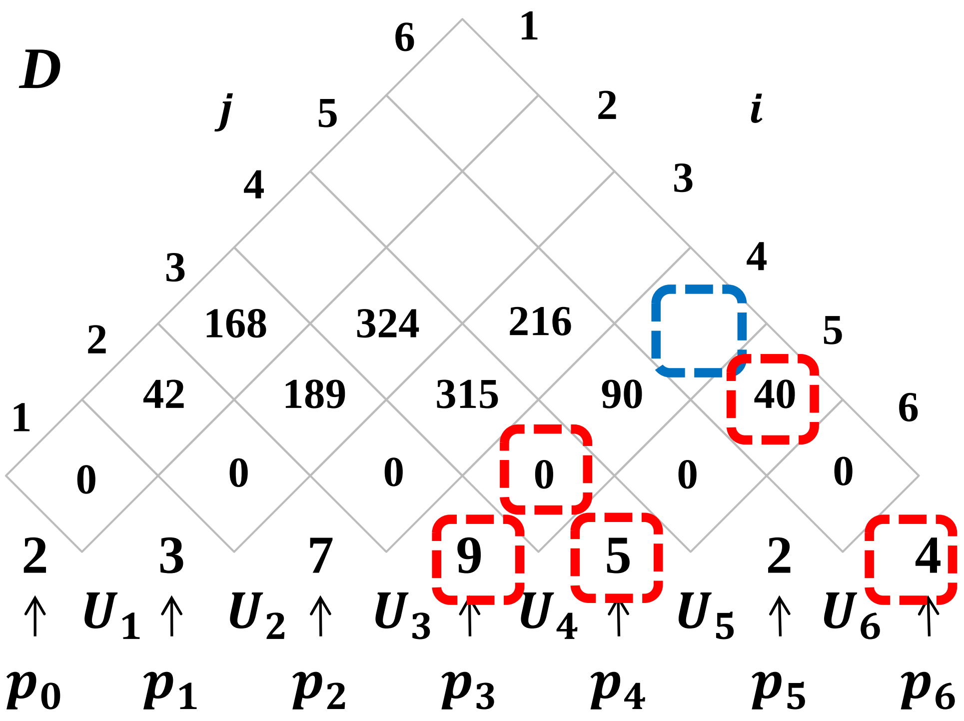


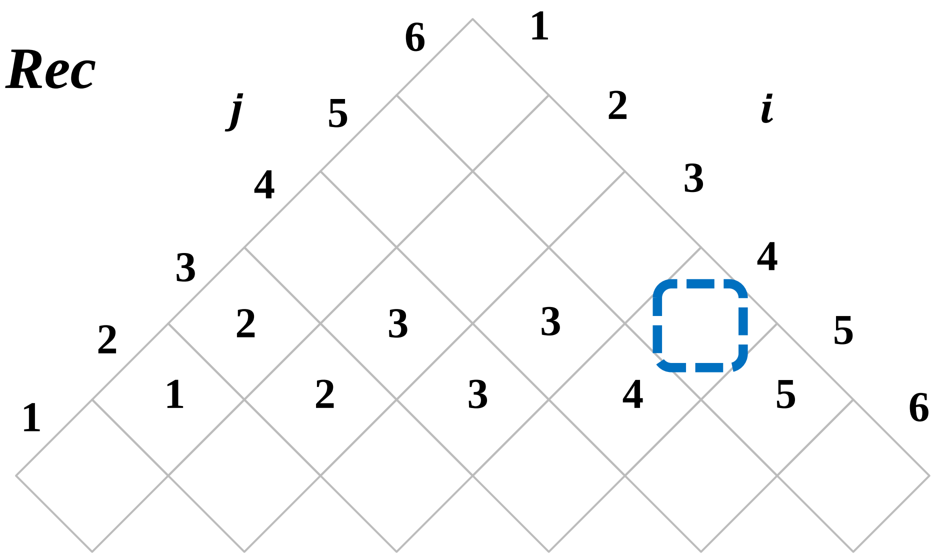


算法实例

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

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

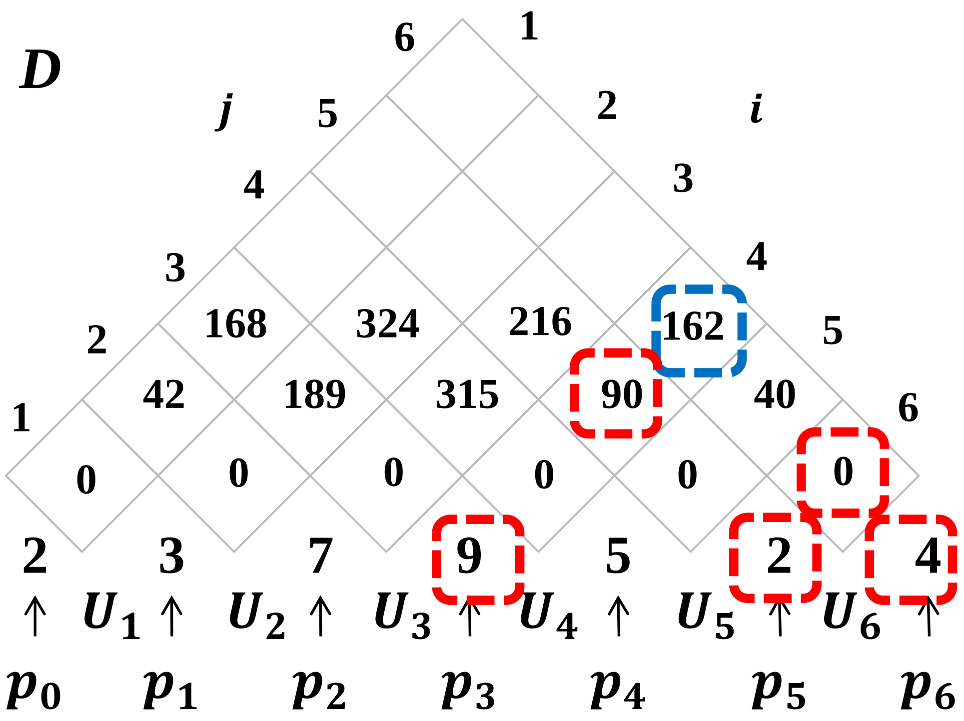


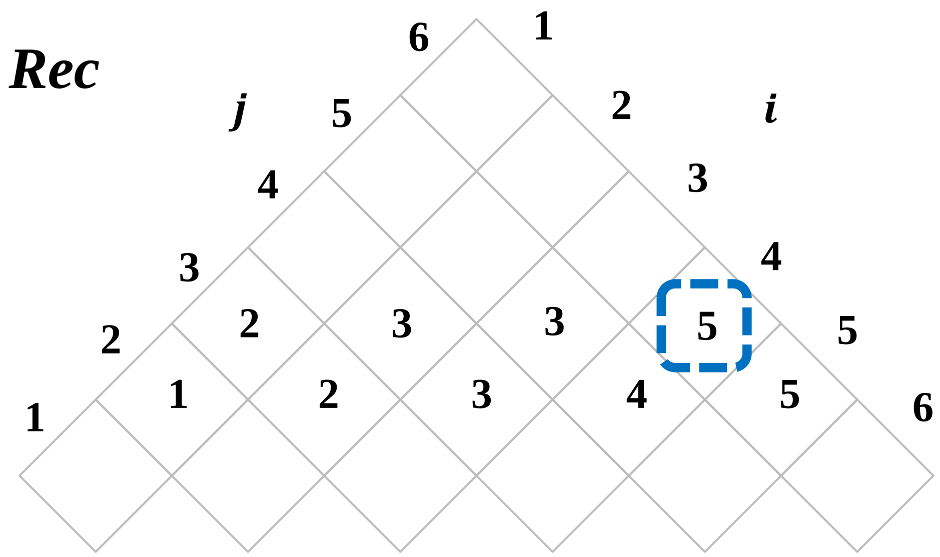


算法实例

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

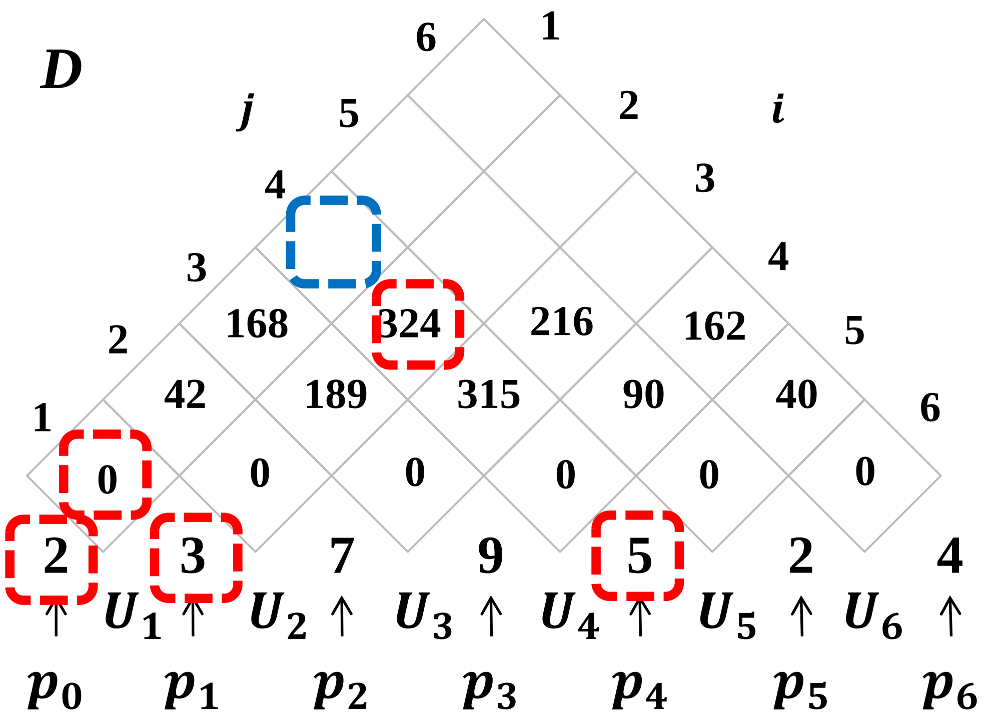
$$= \min \begin{cases} D[4,4] + D[5,6] + p_3 p_4 p_6 = 220 \\ D[4,5] + D[6,6] + p_3 p_5 p_6 = \color{red}{162} \end{cases}$$

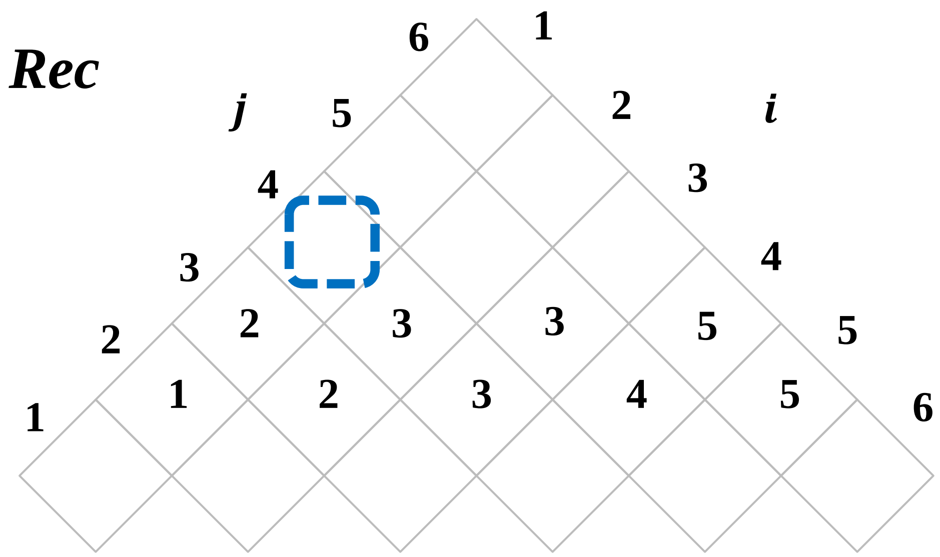




算法实例

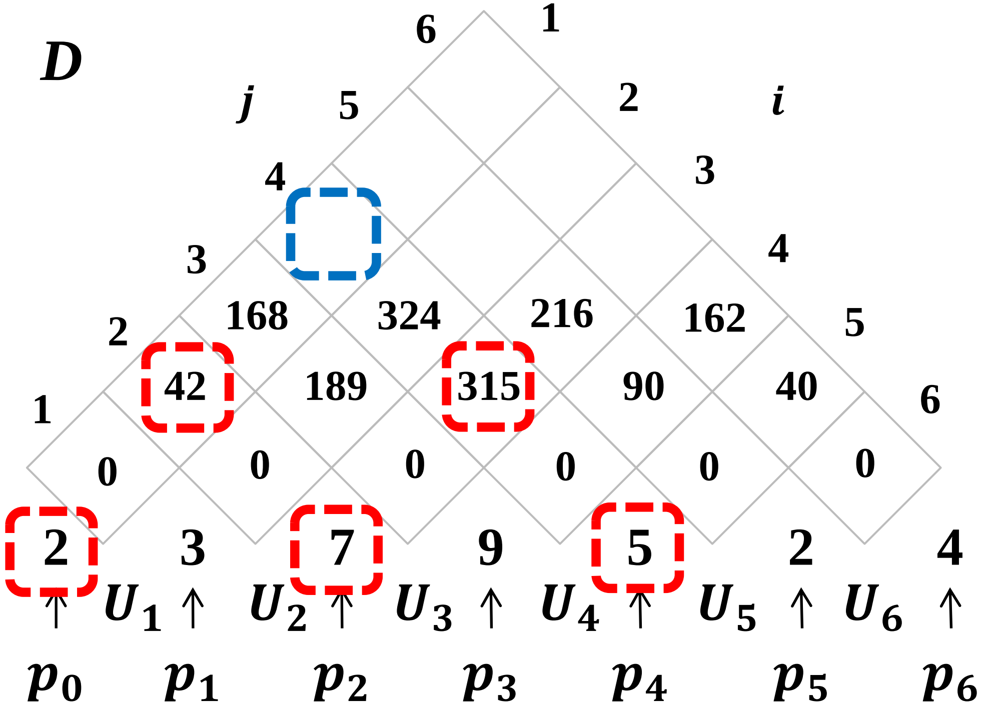
- $D[1,4] = \min \begin{cases} D[1,1] + D[2,4] + p_0 p_1 p_4 = 354 \\ D[1,2] + D[3,4] + p_0 p_2 p_4 = \\ D[1,3] + D[4,4] + p_0 p_3 p_4 = \end{cases}$

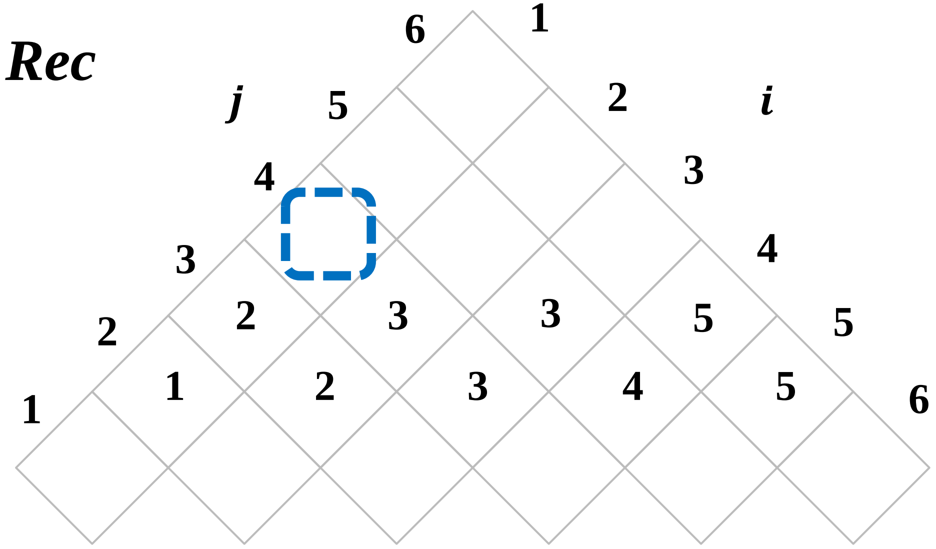




算法实例

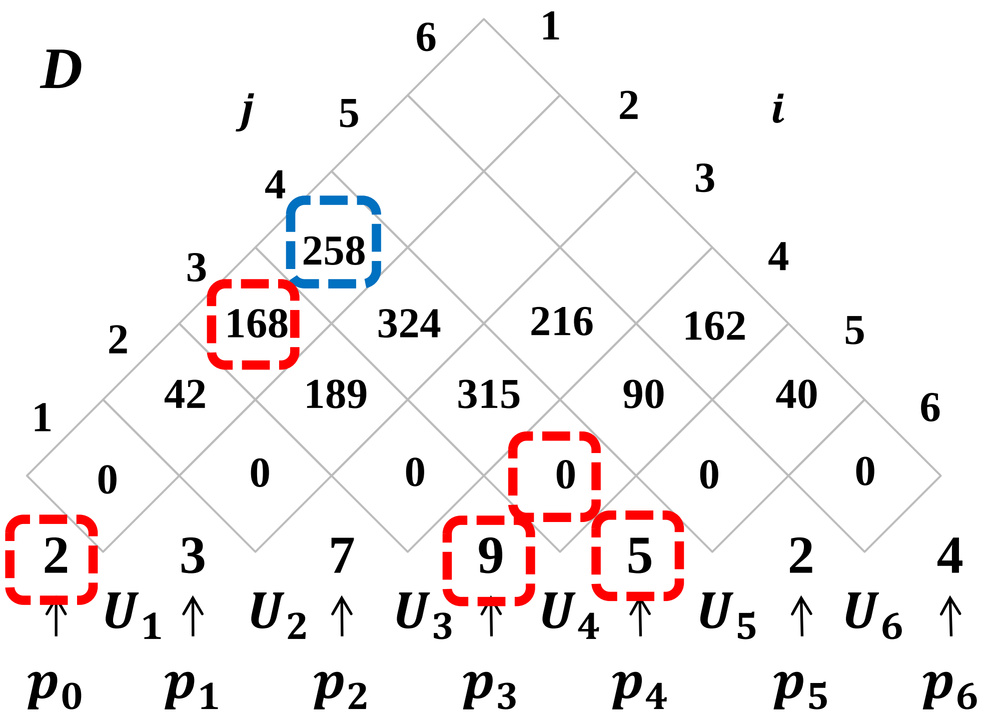
- $D[1,4] = \min \begin{cases} D[1,1] + D[2,4] + p_0p_1p_4 = 354 \\ D[1,2] + D[3,4] + p_0p_2p_4 = 427 \\ D[1,3] + D[4,4] + p_0p_3p_4 = \end{cases}$

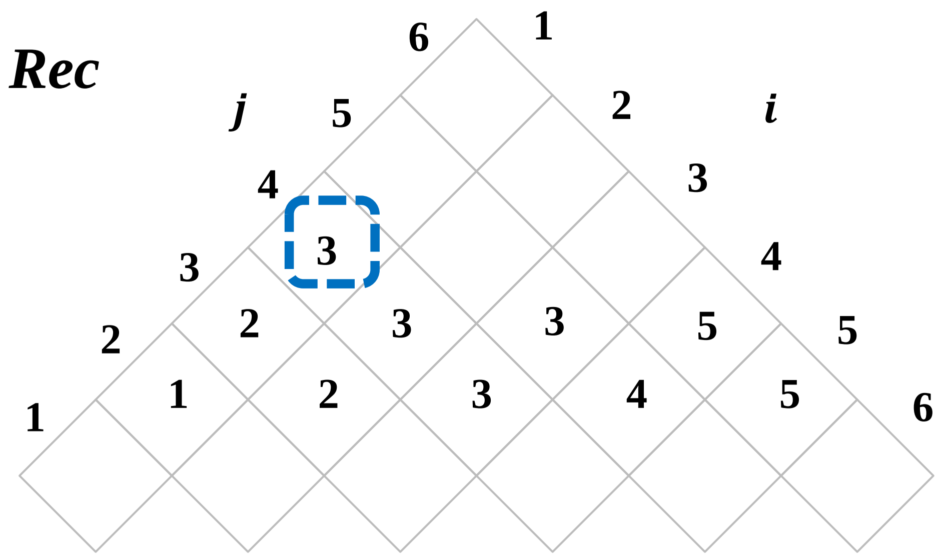




算法实例

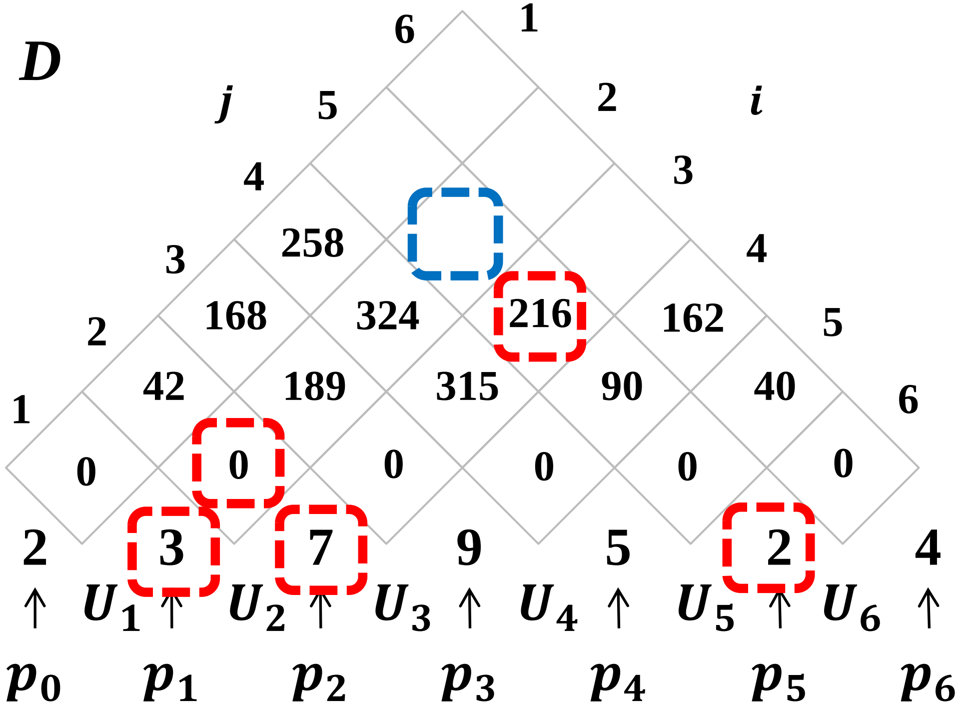
- $D[1,4] = \min\begin{cases} D[1,1] + D[2,4] + p_0p_1p_4 = 354 \\ D[1,2] + D[3,4] + p_0p_2p_4 = 427 \\ D[1,3] + D[4,4] + p_0p_3p_4 = 258 \end{cases}$

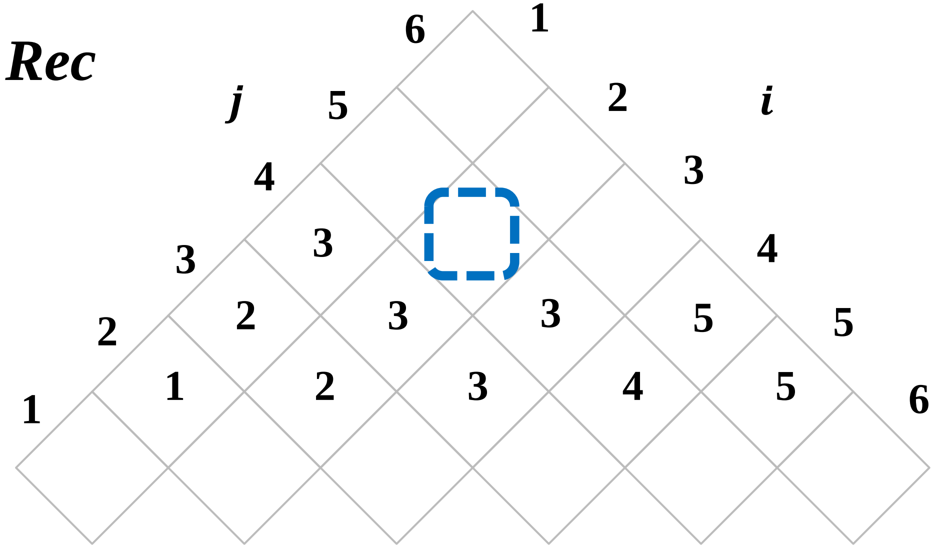




算法实例

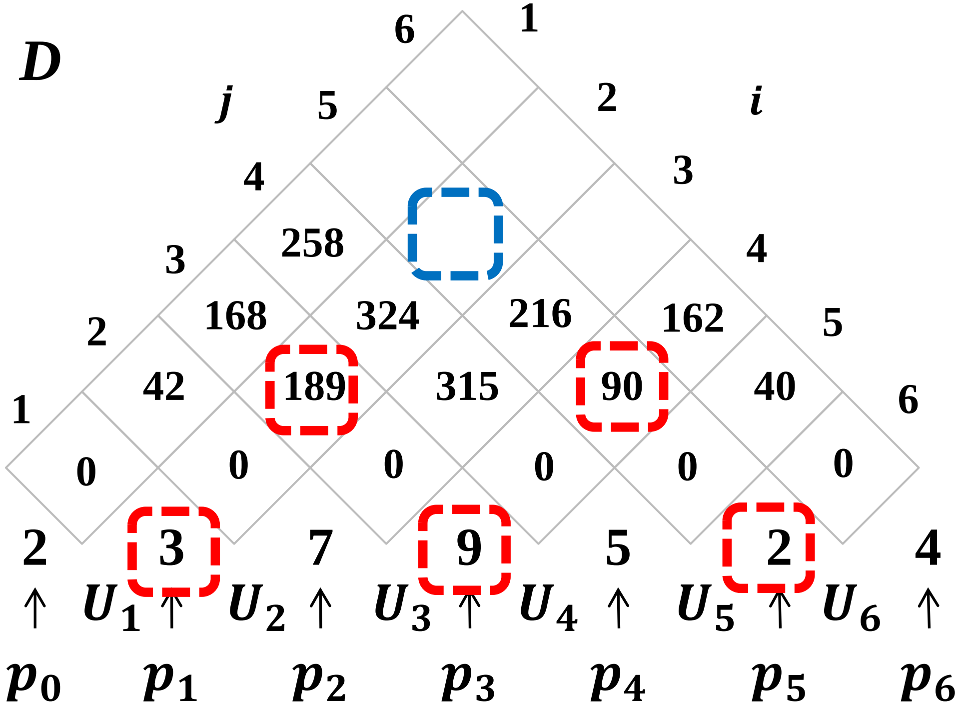
- $D[2,5] = \min\begin{cases} D[2,2] + D[3,5] + p_1p_2p_5 = 258 \\ D[2,3] + D[4,5] + p_1p_3p_5 = \\ D[2,4] + D[5,5] + p_1p_4p_5 = \end{cases}$

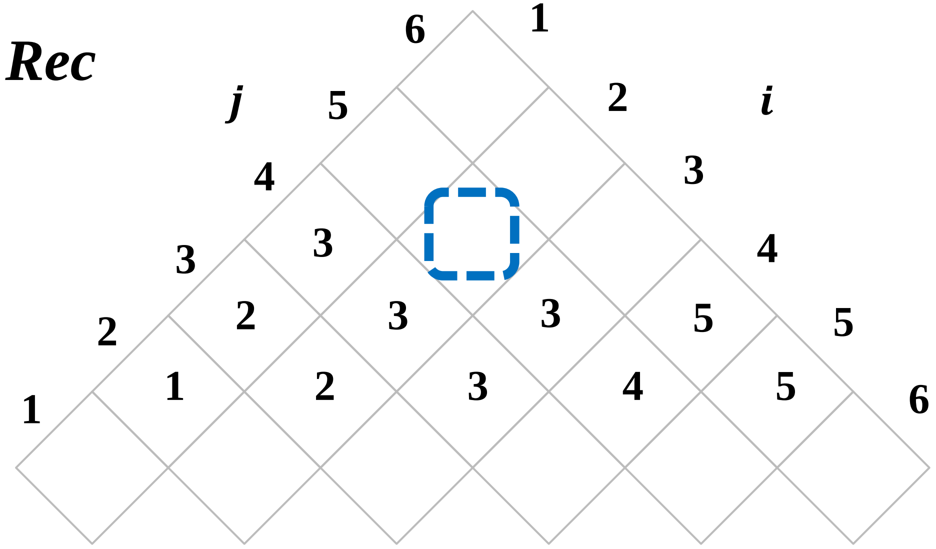




算法实例

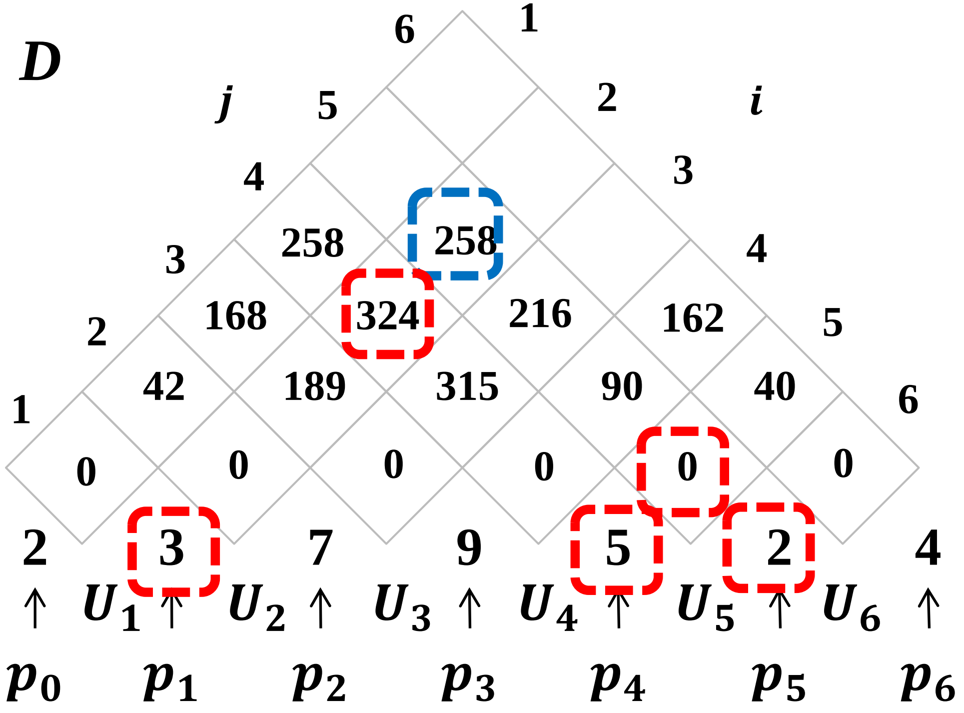
- $$D[2,5] = \min \begin{cases} D[2,2] + D[3,5] + p_1p_2p_5 = 258 \\ D[2,3] + D[4,5] + p_1p_3p_5 = 333 \\ D[2,4] + D[5,5] + p_1p_4p_5 = \end{cases}$$

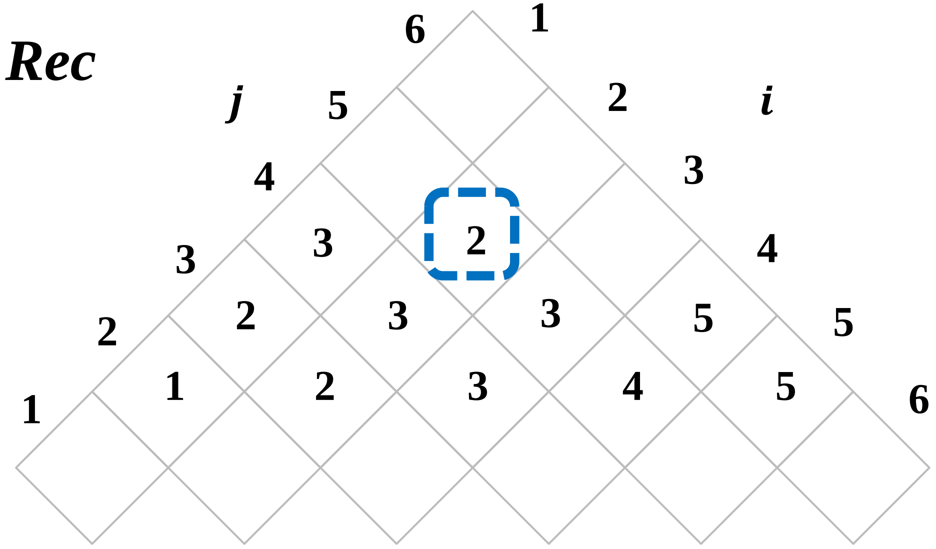




算法实例

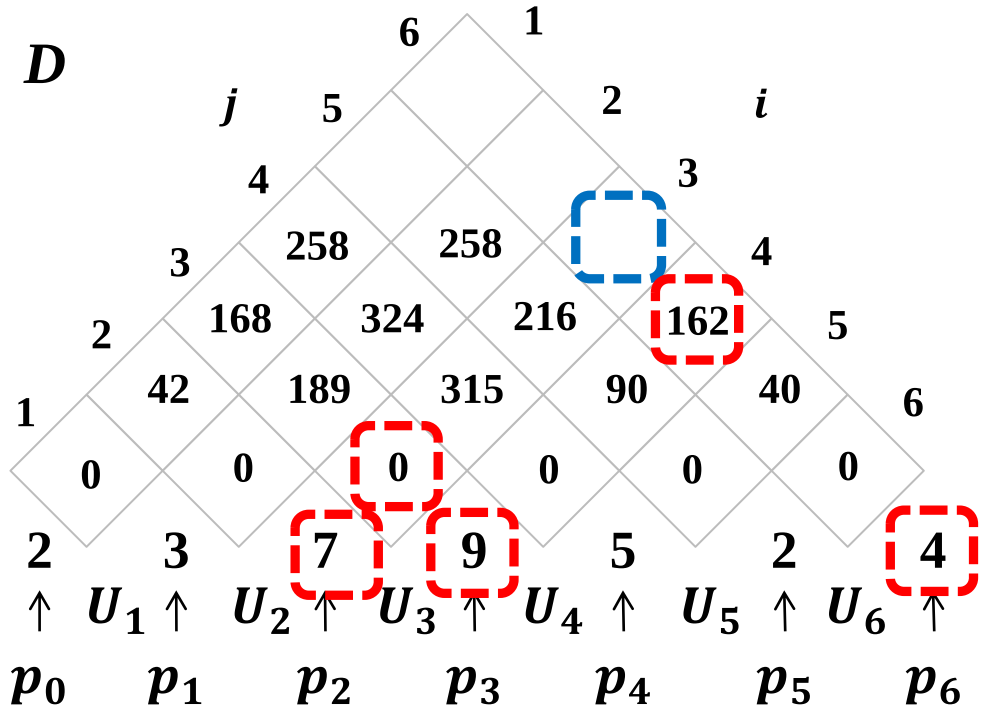
- $D[2,5] = \min \begin{cases} D[2,2] + D[3,5] + p_1 p_2 p_5 = 258 \\ D[2,3] + D[4,5] + p_1 p_3 p_5 = 333 \\ D[2,4] + D[5,5] + p_1 p_4 p_5 = 354 \end{cases}$

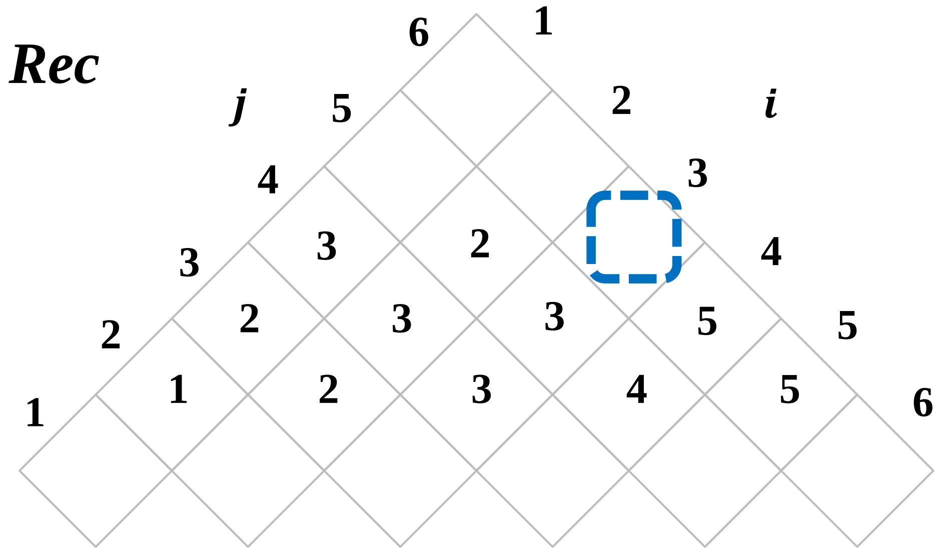




算法实例

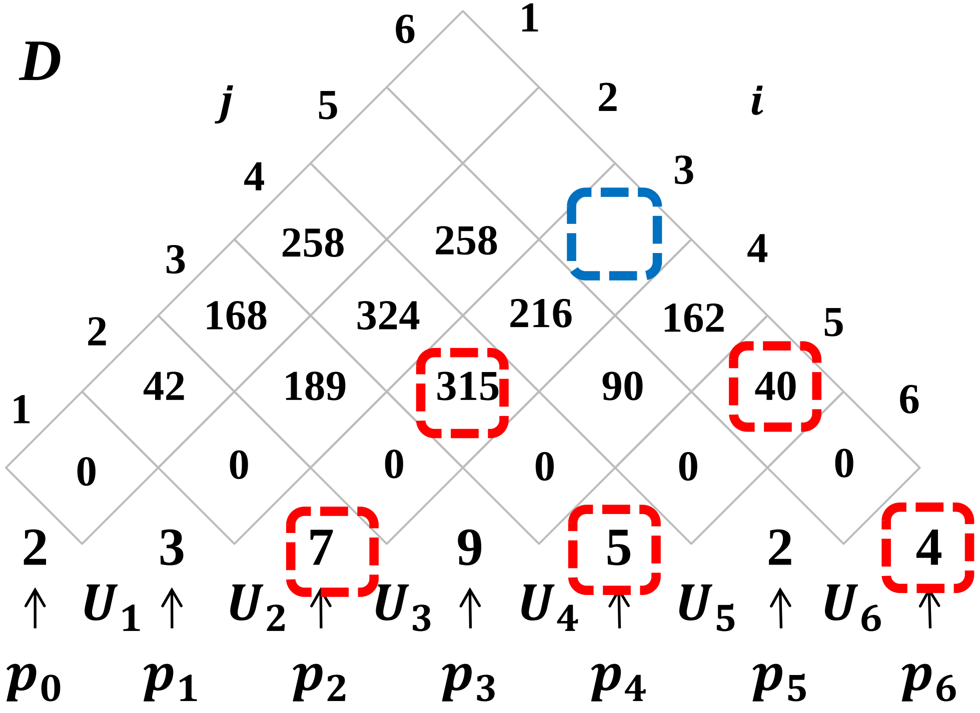
- $D[3,6] = \min \begin{cases} D[3,3] + D[4,6] + p_2p_3p_6 = \mathbf{414} \\ D[3,4] + D[5,6] + p_2p_4p_6 = \\ D[3,5] + D[6,6] + p_2p_5p_6 = \end{cases}$

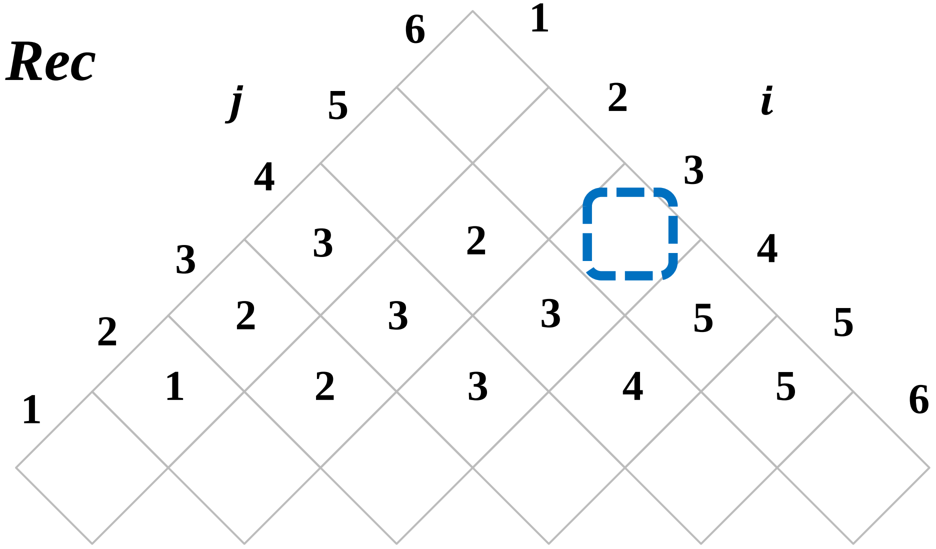




算法实例

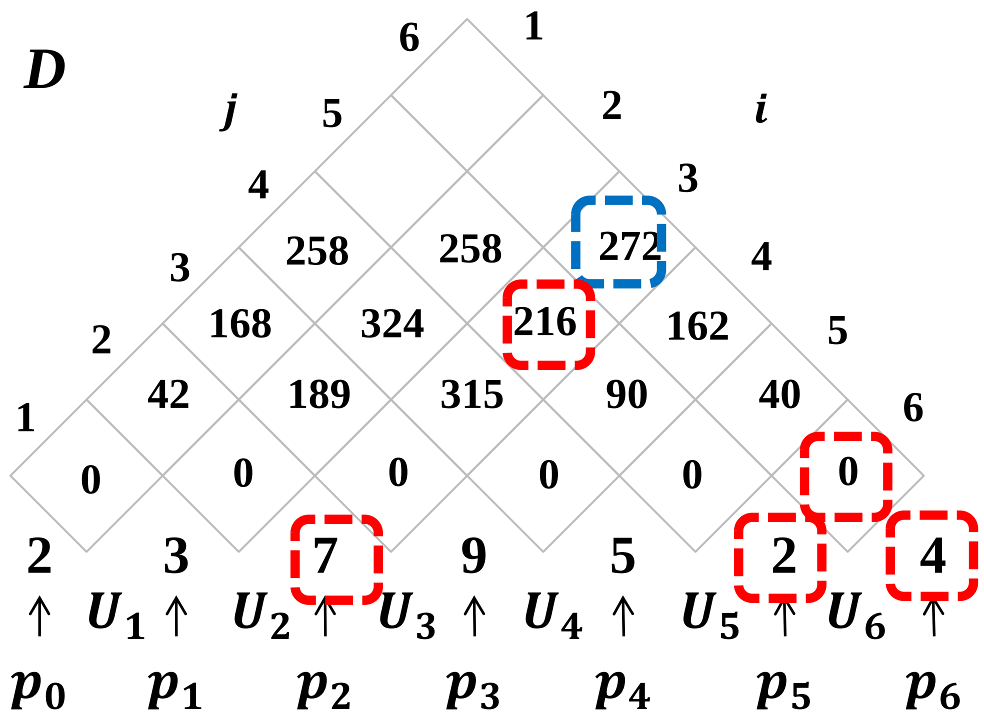
- $D[3,6] = \min \begin{cases} D[3,3] + D[4,6] + p_2p_3p_6 = 414 \\ D[3,4] + D[5,6] + p_2p_4p_6 = 495 \\ D[3,5] + D[6,6] + p_2p_5p_6 = \end{cases}$

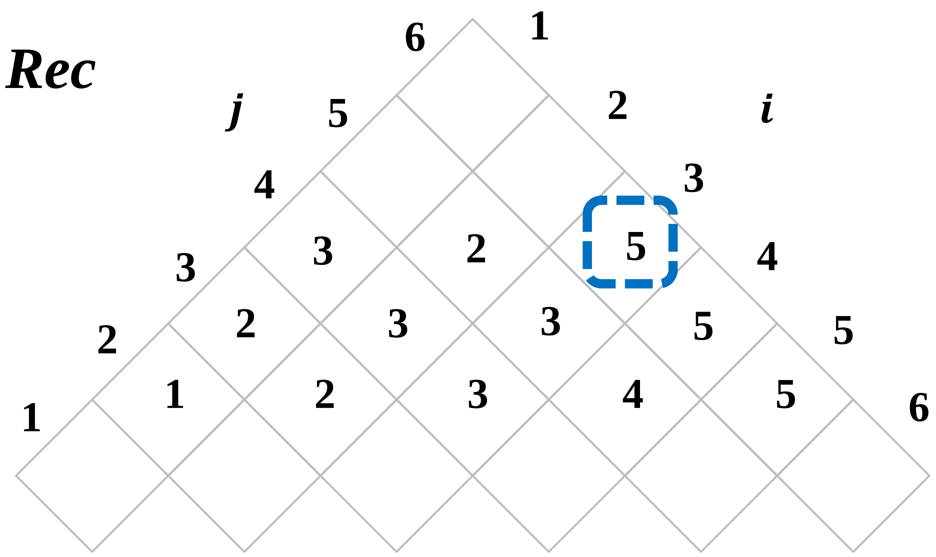




算法实例

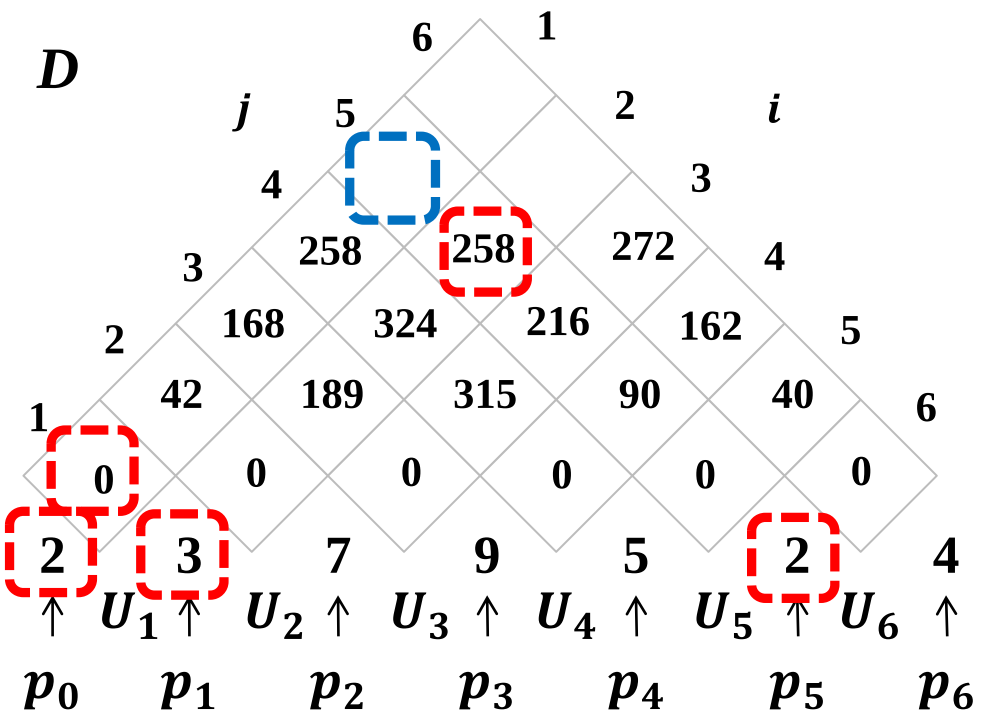
- $D[3,6] = \min \begin{cases} D[3,3] + D[4,6] + p_2 p_3 p_6 = 414 \\ D[3,4] + D[5,6] + p_2 p_4 p_6 = 495 \\ D[3,5] + D[6,6] + p_2 p_5 p_6 = 272 \end{cases}$

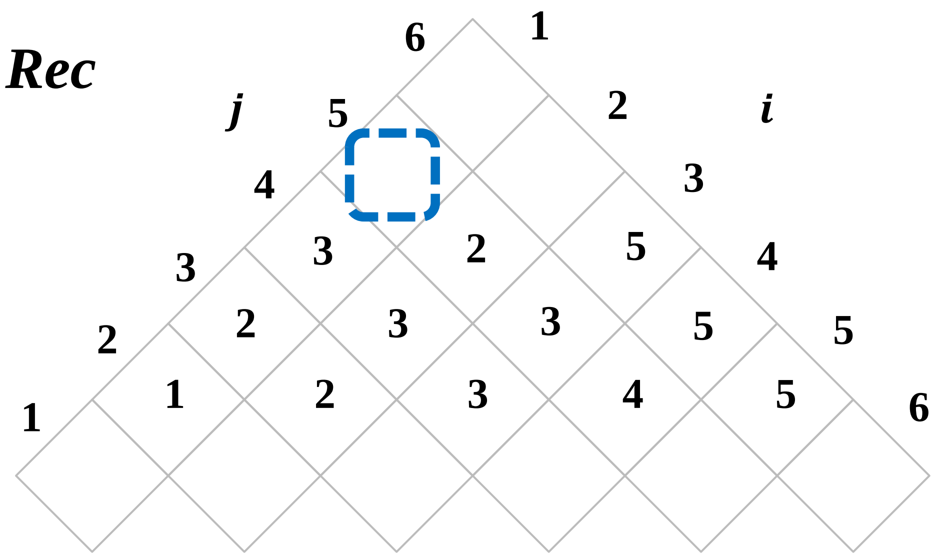




算法实例

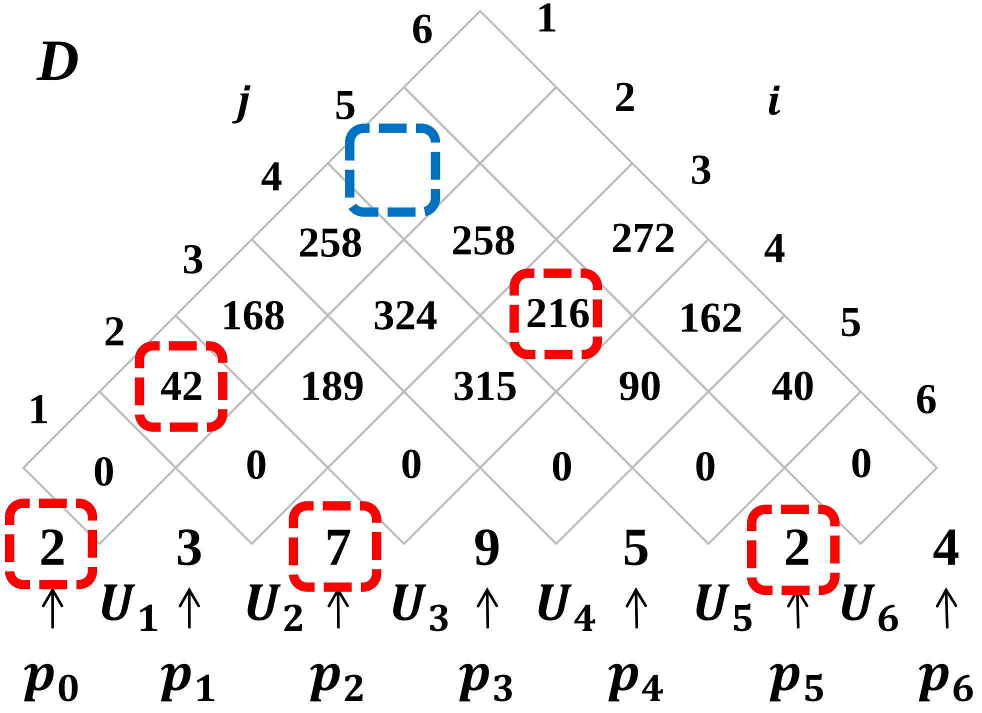
- $D[1,5]=\min\begin{cases}D[1,1]+D[2,5]+p_0p_1p_5=270\\ D[1,2]+D[3,5]+p_0p_2p_5=\\ D[1,3]+D[4,5]+p_0p_3p_5=\\ D[1,4]+D[5,5]+p_0p_4p_5=\end{cases}$

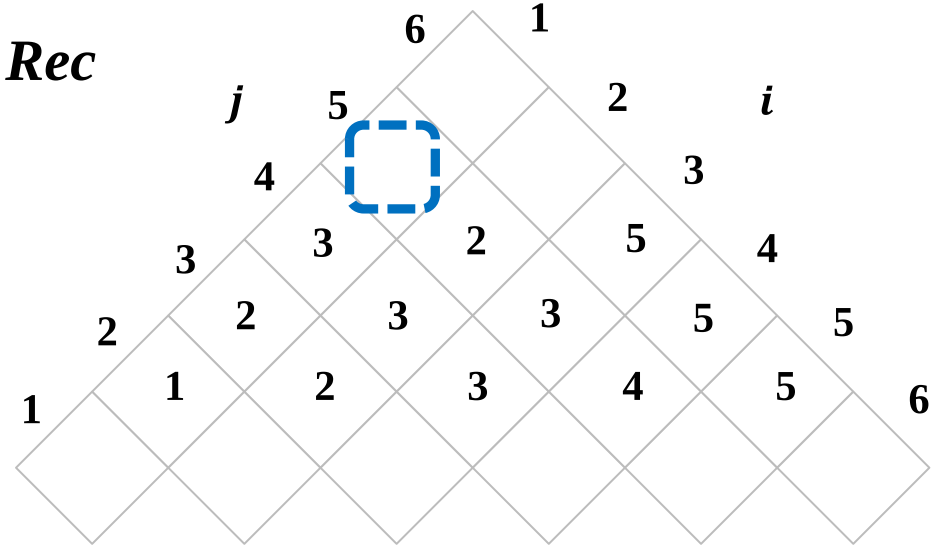




算法实例

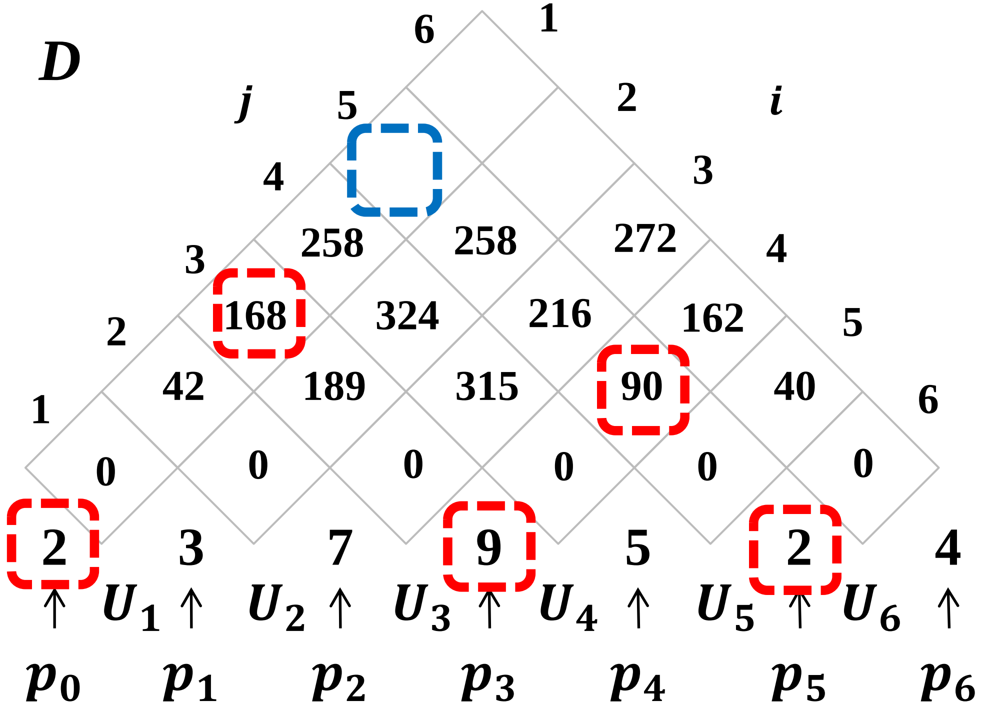
- $$D[1,5] = \min \begin{cases} D[1,1] + D[2,5] + p_0 p_1 p_5 = 270 \\ D[1,2] + D[3,5] + p_0 p_2 p_5 = 286 \\ D[1,3] + D[4,5] + p_0 p_3 p_5 = \\ D[1,4] + D[5,5] + p_0 p_4 p_5 = \end{cases}$$

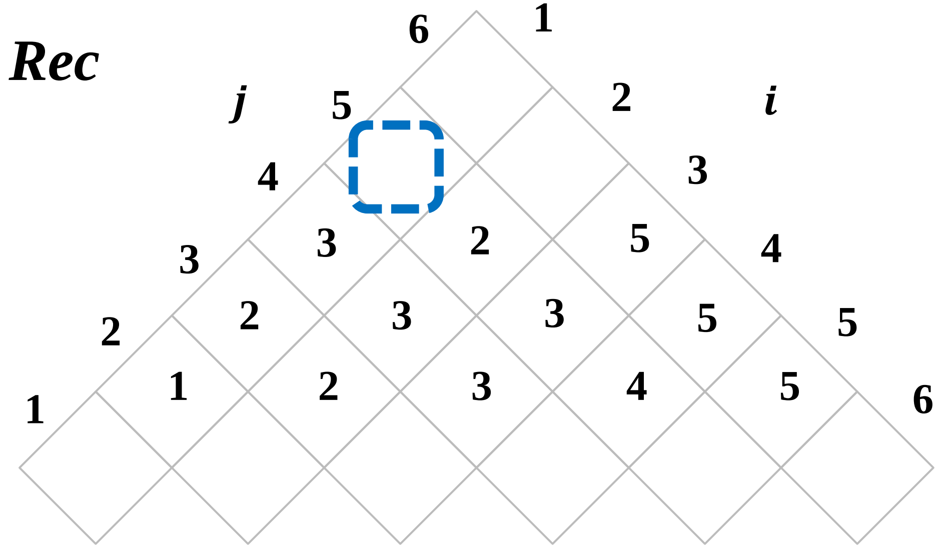




算法实例

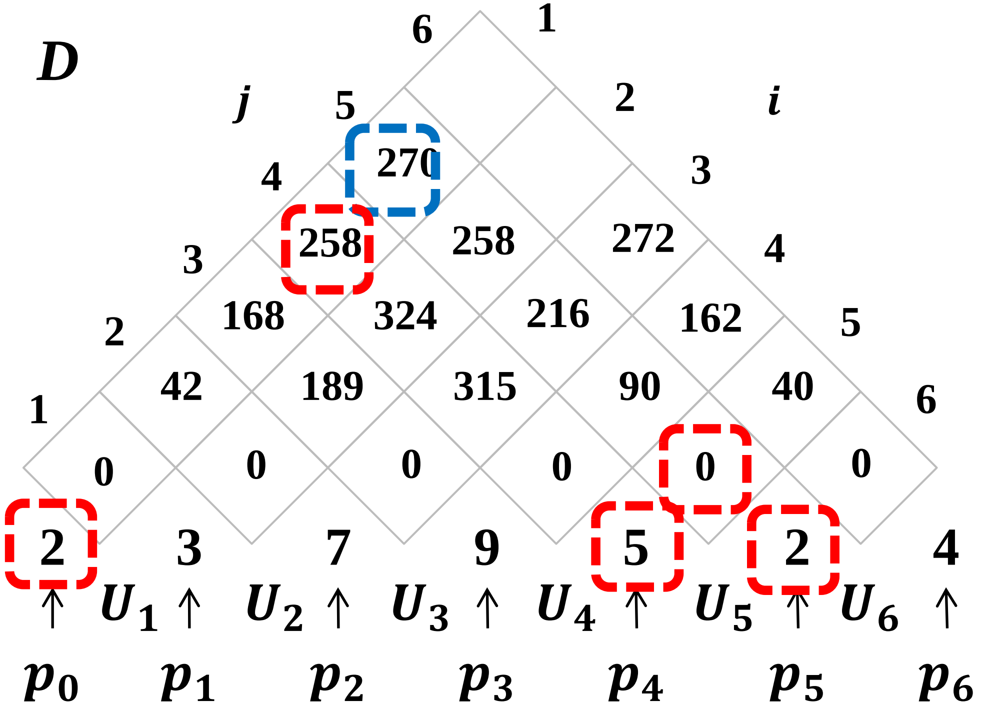
- $D[1,5] = \min \begin{cases} D[1,1] + D[2,5] + p_0p_1p_5 = 270 \\ D[1,2] + D[3,5] + p_0p_2p_5 = 286 \\ D[1,3] + D[4,5] + p_0p_3p_5 = 294 \\ D[1,4] + D[5,5] + p_0p_4p_5 = \end{cases}$

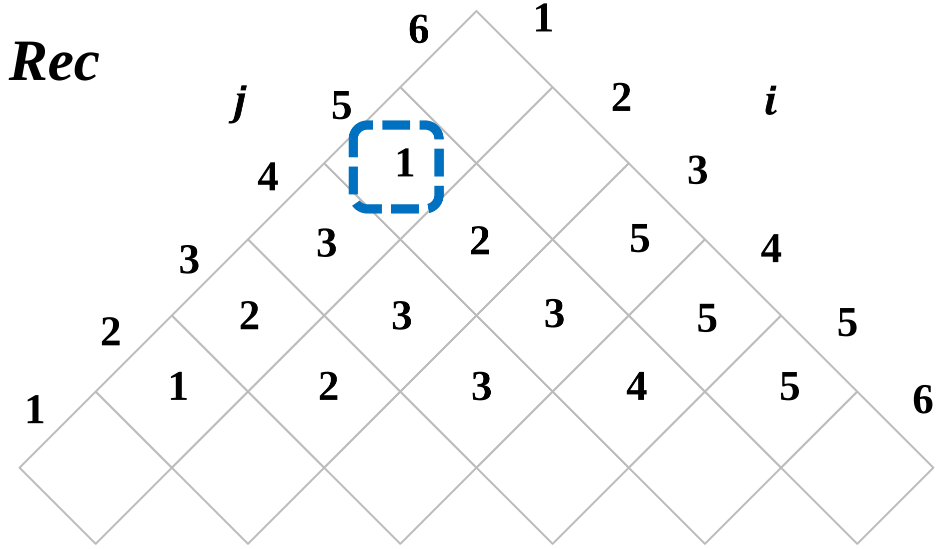




算法实例

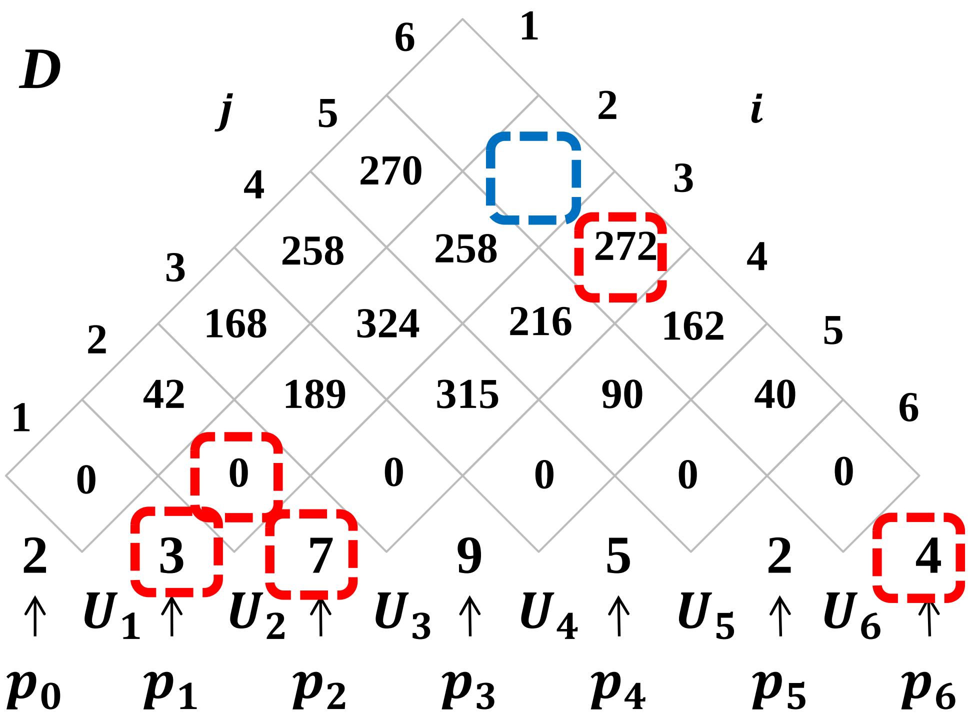
- $$D[1,5] = \min \begin{cases} D[1,1] + D[2,5] + p_0 p_1 p_5 = \color{red}{270} \\ D[1,2] + D[3,5] + p_0 p_2 p_5 = 286 \\ D[1,3] + D[4,5] + p_0 p_3 p_5 = 294 \\ D[1,4] + D[5,5] + p_0 p_4 p_5 = 278 \end{cases}$$

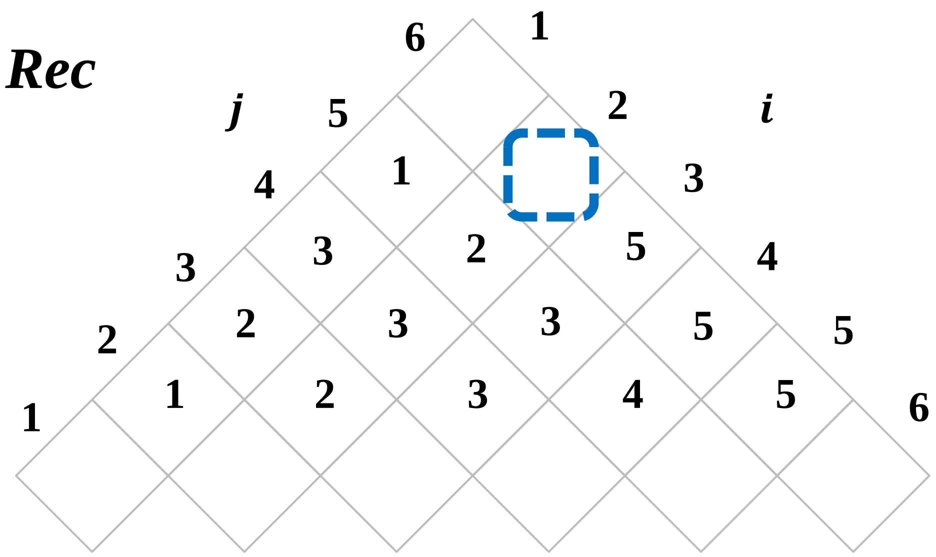




算法实例

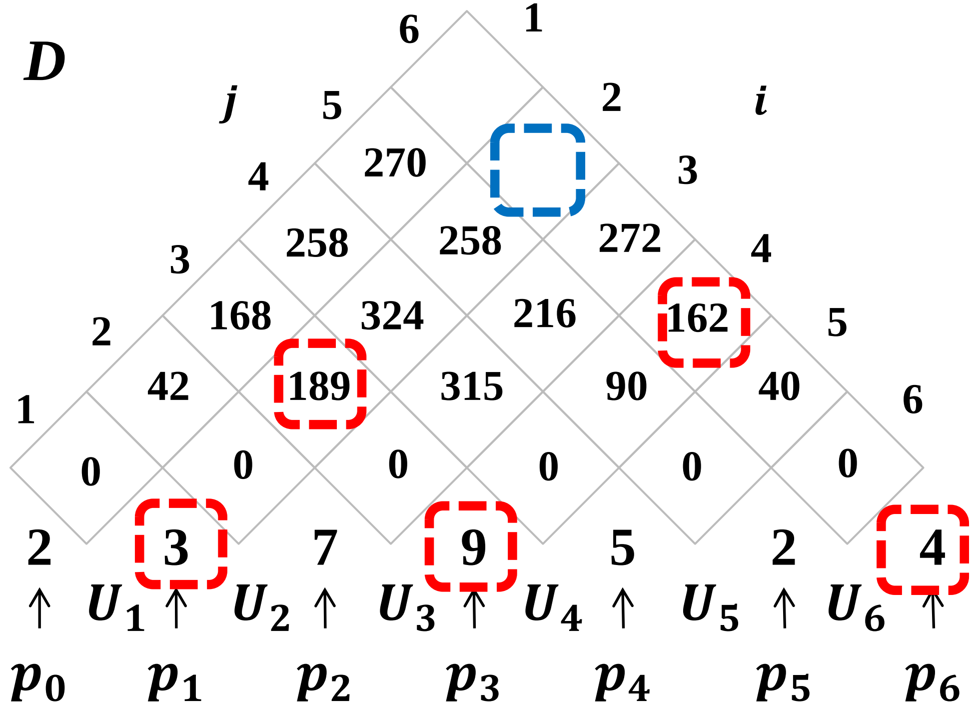
- $D[2,6] = \min \begin{cases} D[2,2] + D[3,6] + p_1 p_2 p_6 = 356 \\ D[2,3] + D[4,6] + p_1 p_3 p_6 = \\ D[2,4] + D[5,6] + p_1 p_4 p_6 = \\ D[2,5] + D[6,6] + p_1 p_5 p_6 = \end{cases}$

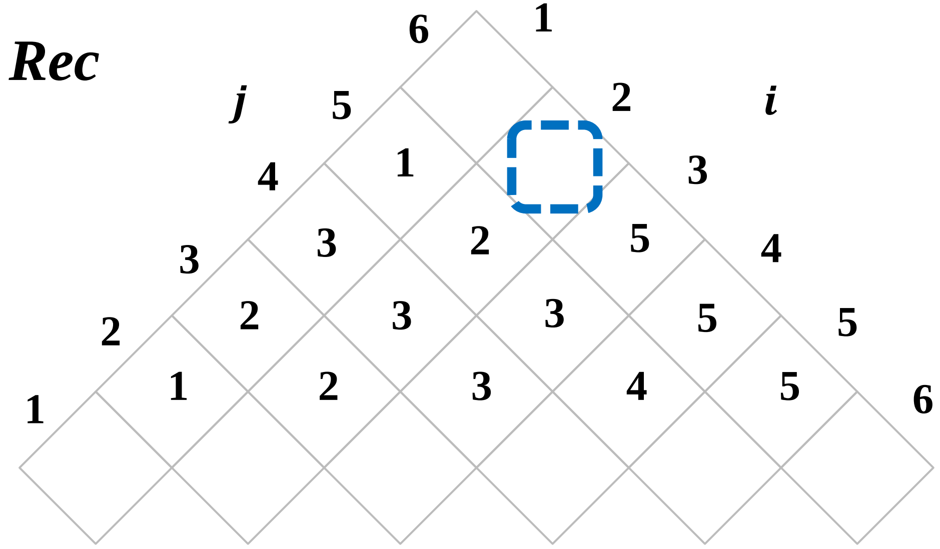




算法实例

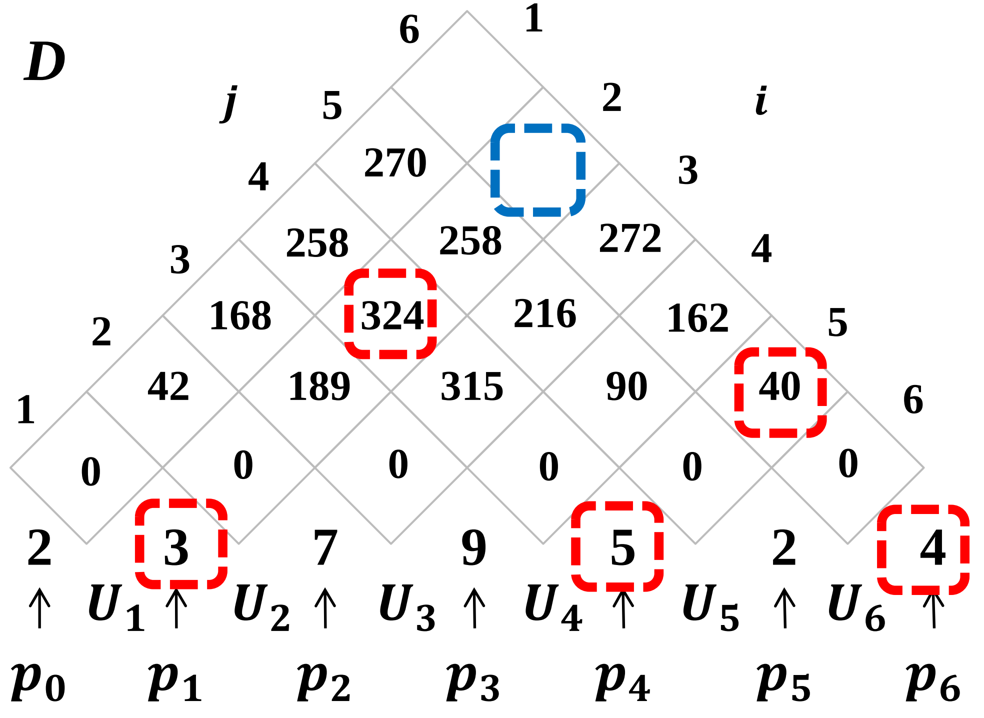
- $D[2,6] = \min \begin{cases} D[2,2] + D[3,6] + p_1 p_2 p_6 = 356 \\ D[2,3] + D[4,6] + p_1 p_3 p_6 = 459 \\ D[2,4] + D[5,6] + p_1 p_4 p_6 = \\ D[2,5] + D[6,6] + p_1 p_5 p_6 = \end{cases}$

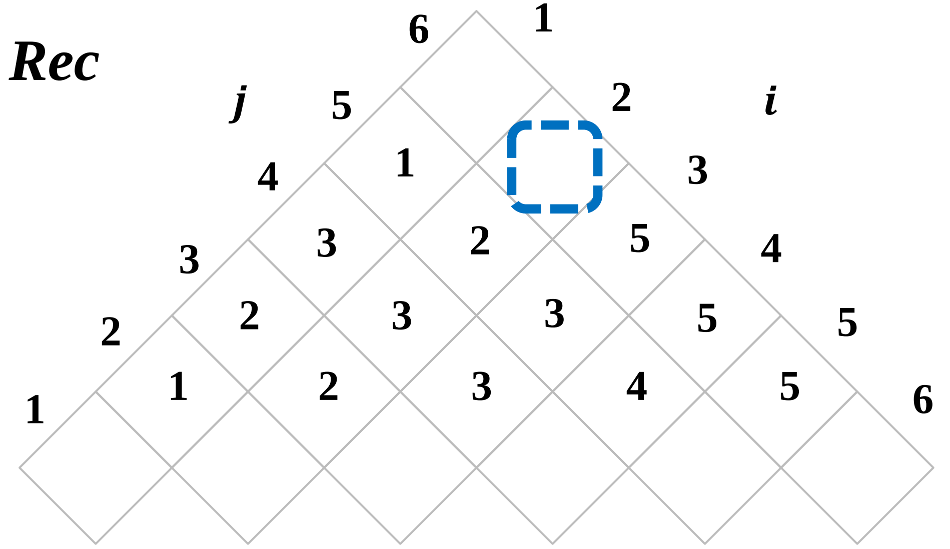




算法实例

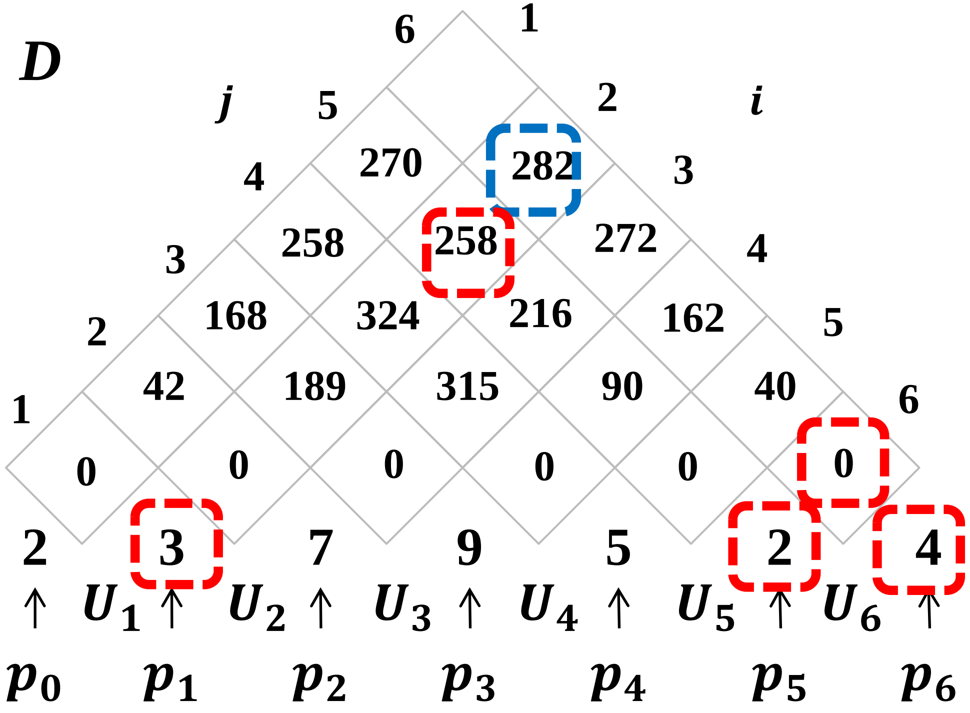
- $D[2,6] = \min \begin{cases} D[2,2] + D[3,6] + p_1p_2p_6 = 356 \\ D[2,3] + D[4,6] + p_1p_3p_6 = 459 \\ D[2,4] + D[5,6] + p_1p_4p_6 = 424 \\ D[2,5] + D[6,6] + p_1p_5p_6 = \end{cases}$

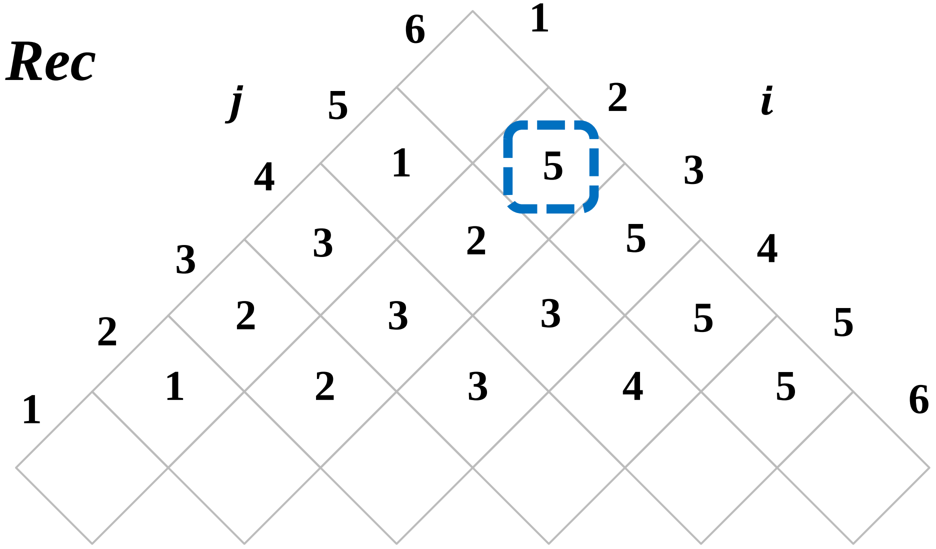




算法实例

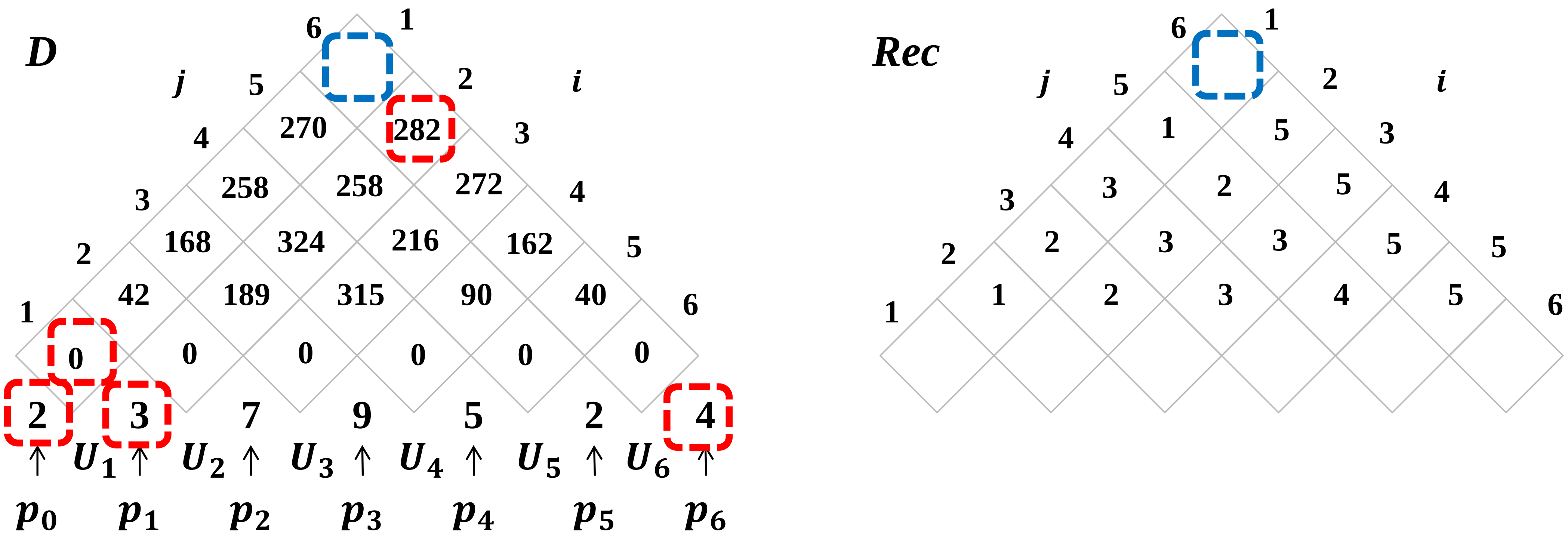
- $D[2,6] = \min \begin{cases} D[2,2] + D[3,6] + p_1p_2p_6 = 356 \\ D[2,3] + D[4,6] + p_1p_3p_6 = 459 \\ D[2,4] + D[5,6] + p_1p_4p_6 = 424 \\ D[2,5] + D[6,6] + p_1p_5p_6 = 282 \end{cases}$





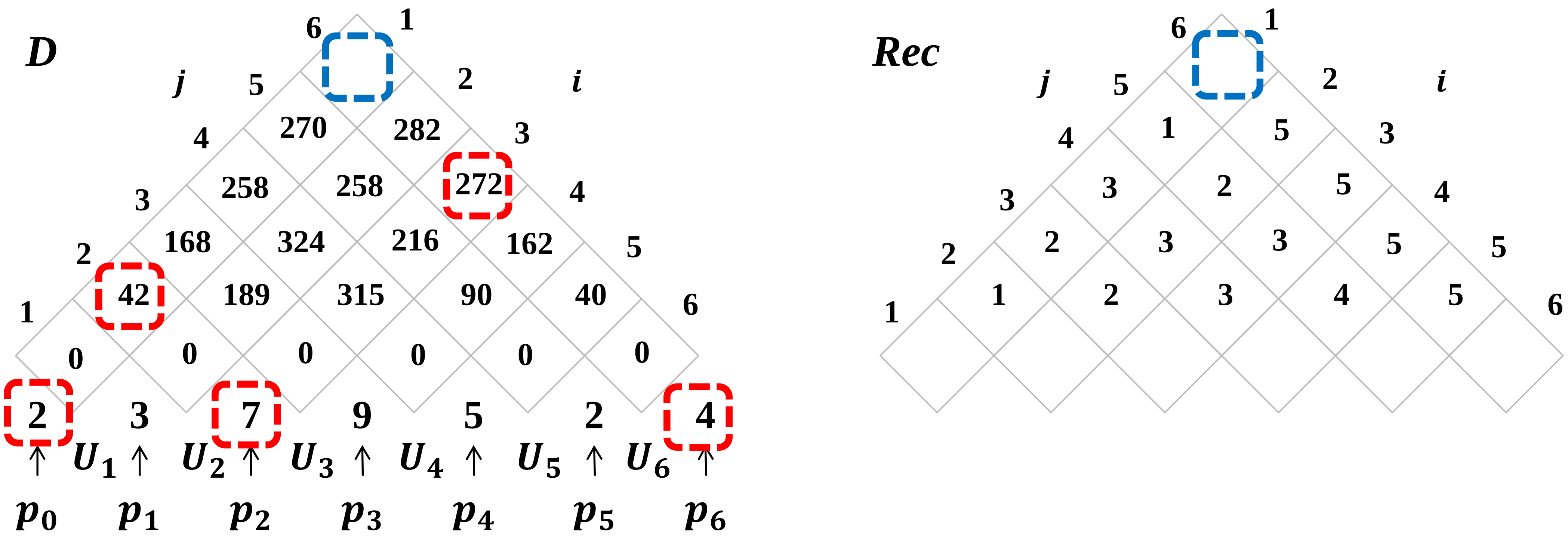
算法实例

- $$D[1,6] = \min \begin{cases} D[1,1] + D[2,6] + p_0p_1p_6 = 306 \\ D[1,2] + D[3,6] + p_0p_2p_6 = \\ D[1,3] + D[4,6] + p_0p_3p_6 = \\ D[1,4] + D[5,6] + p_0p_4p_6 = \\ D[1,5] + D[6,6] + p_0p_5p_6 = \end{cases}$$



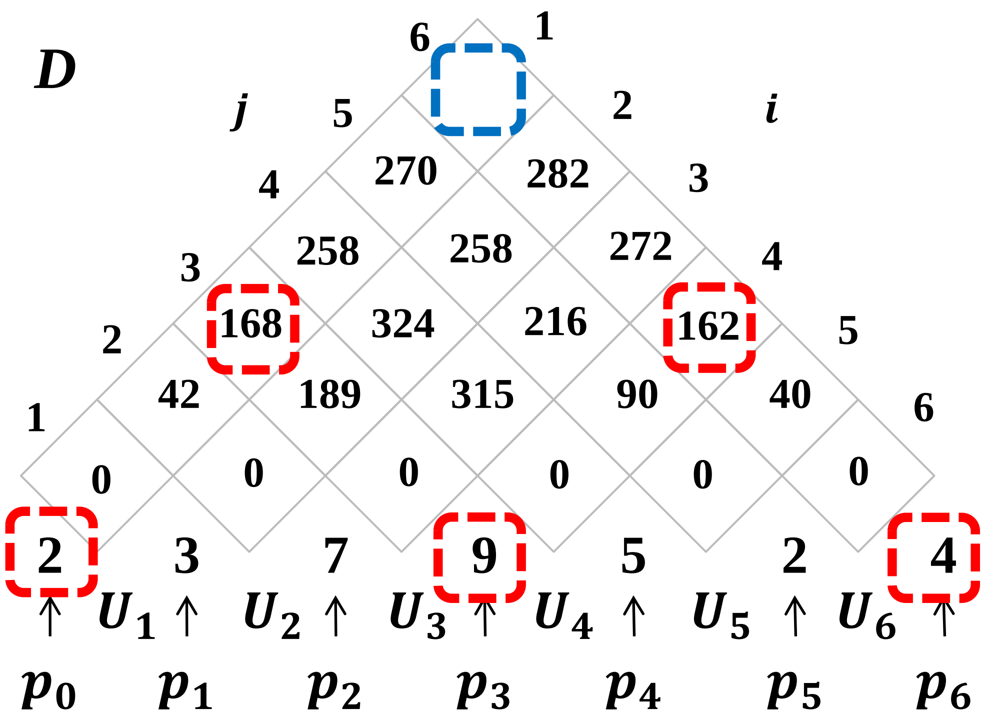
算法实例

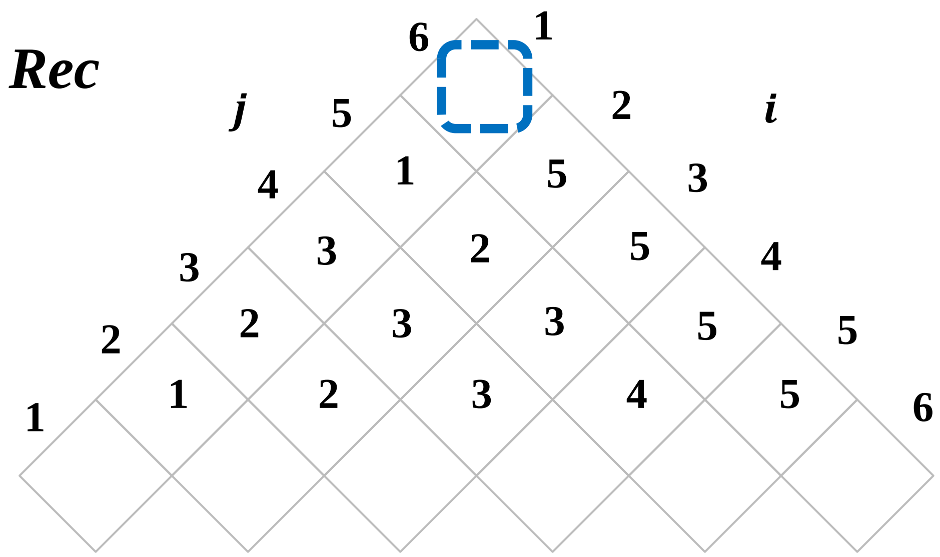
- $$D[1,6]=\min\begin{cases} D[1,1]+D[2,6]+p_0p_1p_6=306 \\ D[1,2]+D[3,6]+p_0p_2p_6=370 \\ D[1,3]+D[4,6]+p_0p_3p_6= \\ D[1,4]+D[5,6]+p_0p_4p_6= \\ D[1,5]+D[6,6]+p_0p_5p_6= \end{cases}$$



算法实例

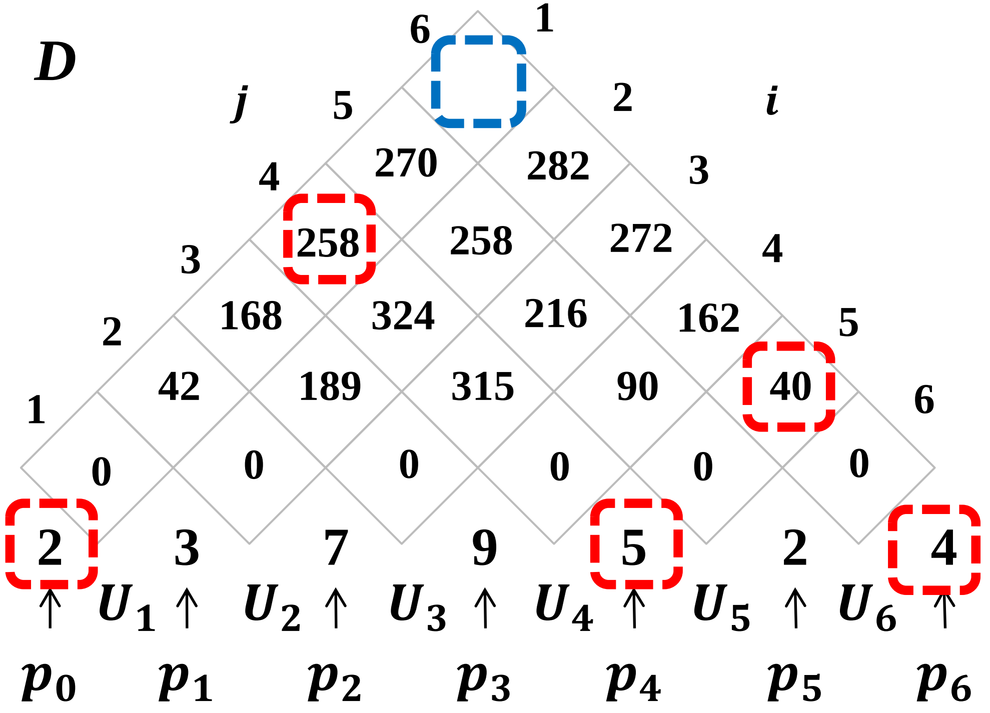
- $D[1,6] = \min \begin{cases} D[1,1] + D[2,6] + p_0p_1p_6 = 306 \\ D[1,2] + D[3,6] + p_0p_2p_6 = 370 \\ D[1,3] + D[4,6] + p_0p_3p_6 = 402 \\ D[1,4] + D[5,6] + p_0p_4p_6 = \\ D[1,5] + D[6,6] + p_0p_5p_6 = \end{cases}$

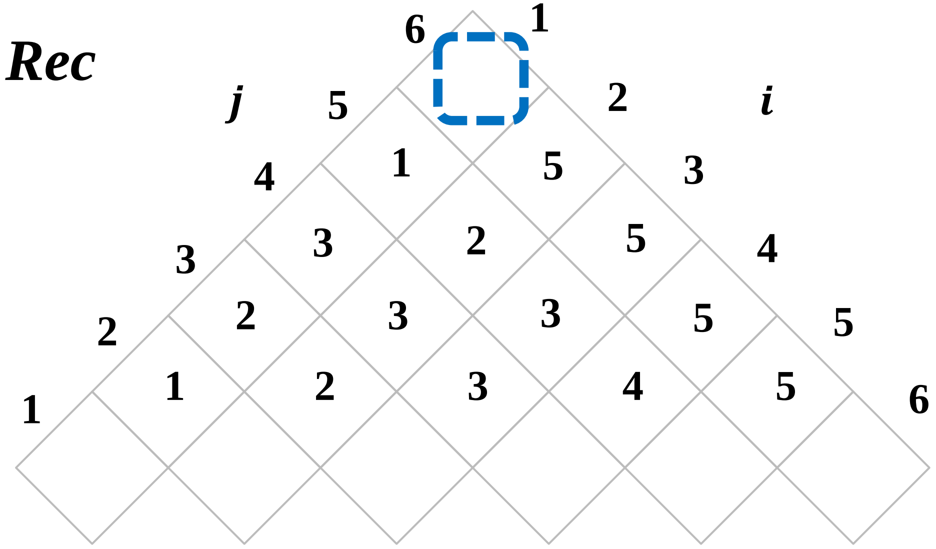




算法实例

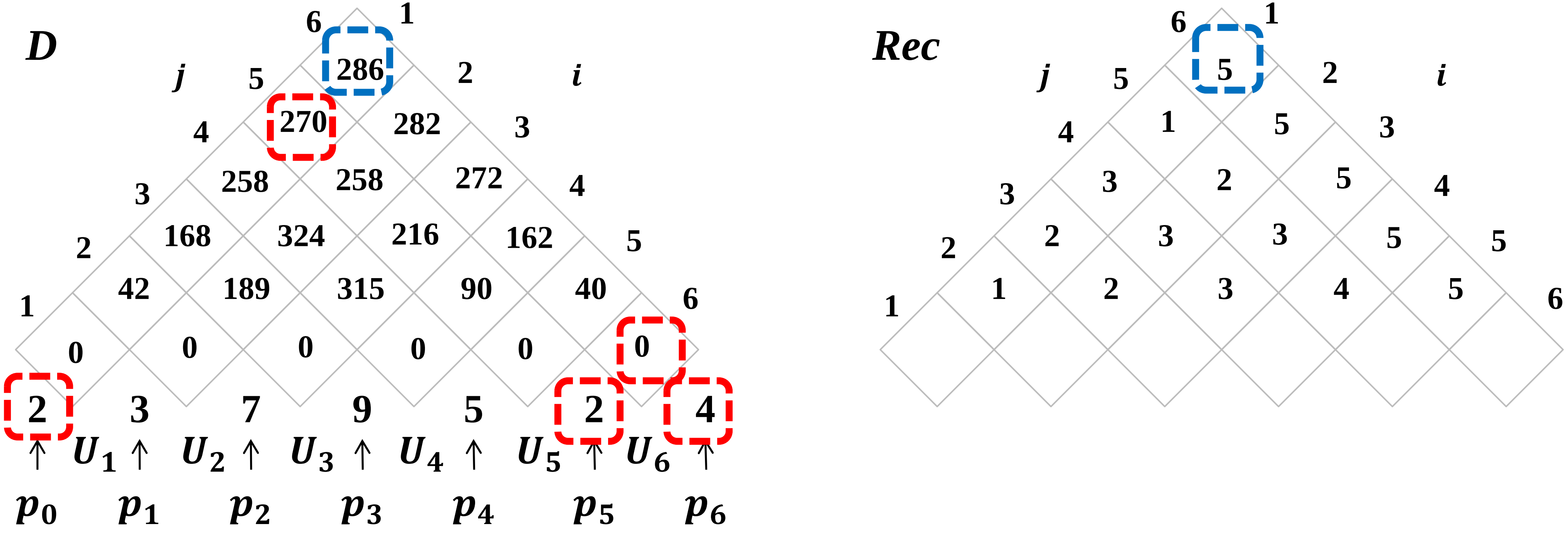
- $$D[1,6]=\min\begin{cases} D[1,1]+D[2,6]+p_0p_1p_6 = 306 \\ D[1,2]+D[3,6]+p_0p_2p_6 = 370 \\ D[1,3]+D[4,6]+p_0p_3p_6 = 402 \\ D[1,4]+D[5,6]+p_0p_4p_6 = 338 \\ D[1,5]+D[6,6]+p_0p_5p_6 = \end{cases}$$





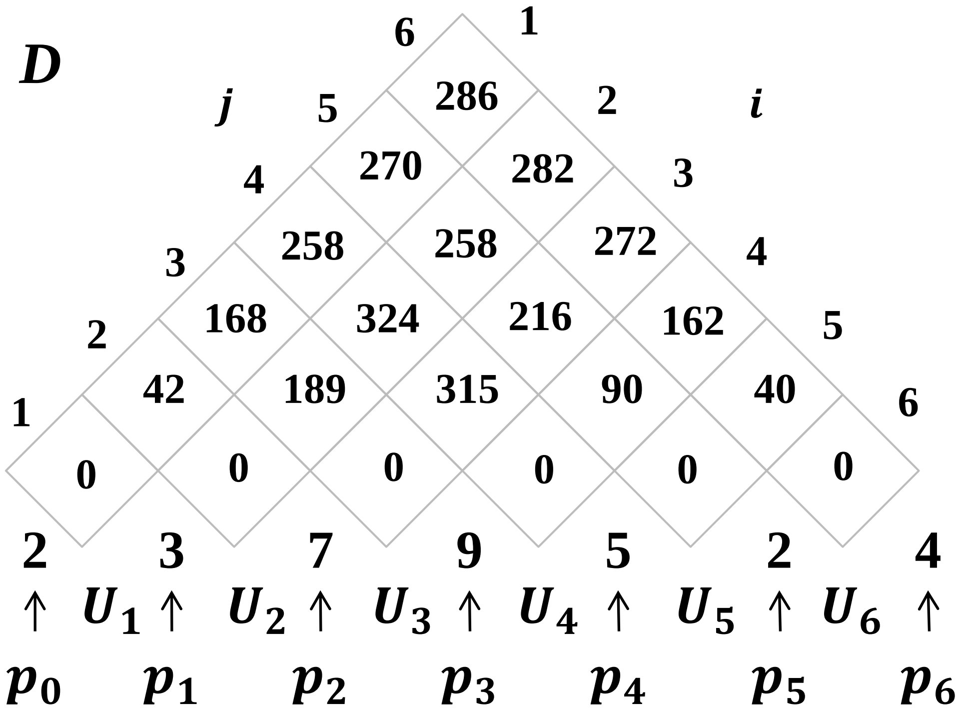
算法实例

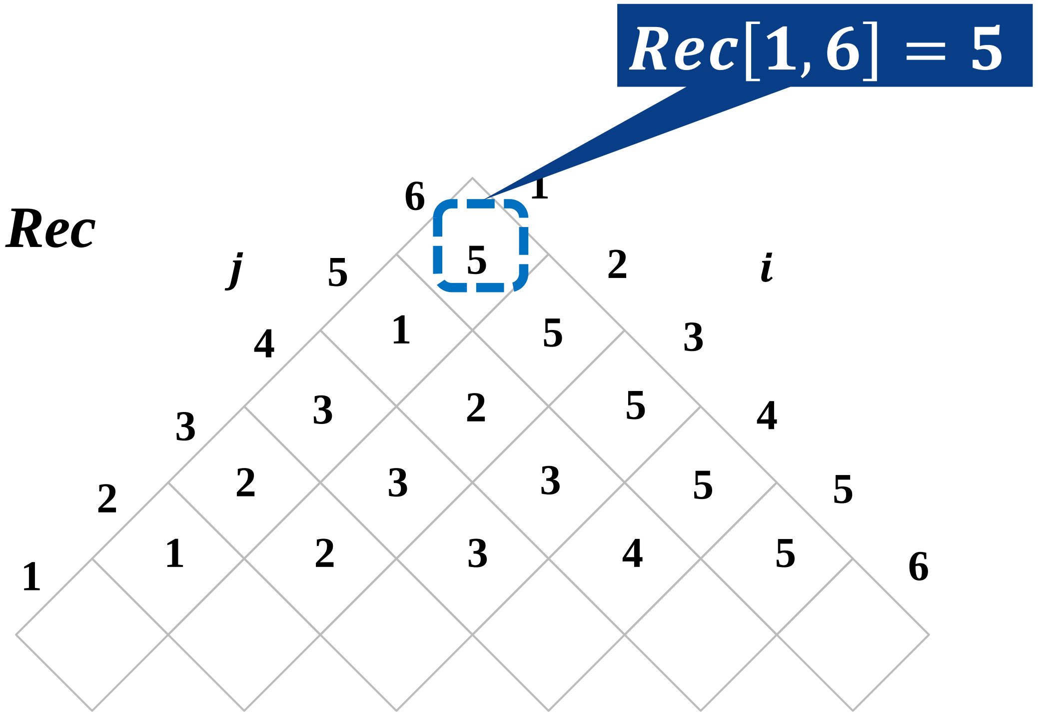
- $D[1,6] = \min \begin{cases} D[1,1] + D[2,6] + p_0p_1p_6 = 306 \\ D[1,2] + D[3,6] + p_0p_2p_6 = 370 \\ D[1,3] + D[4,6] + p_0p_3p_6 = 402 \\ D[1,4] + D[5,6] + p_0p_4p_6 = 338 \\ D[1,5] + D[6,6] + p_0p_5p_6 = 286 \end{cases}$



算法实例

- $U_1U_2U_3U_4U_5U_6$
- $\rightarrow (U_1U_2U_3U_4U_5)(U_6)$

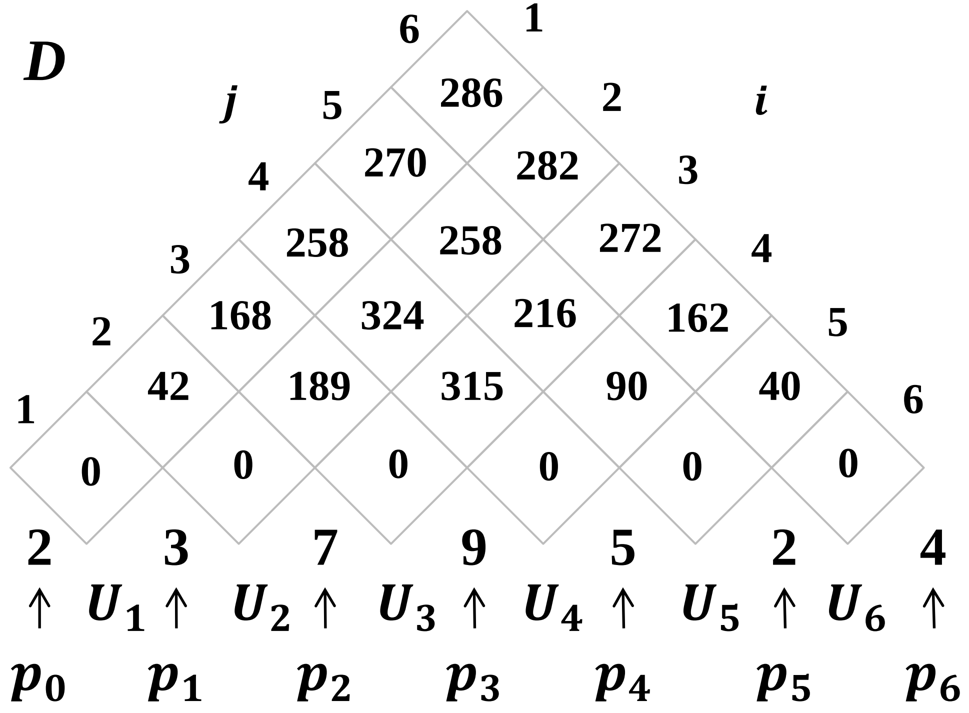


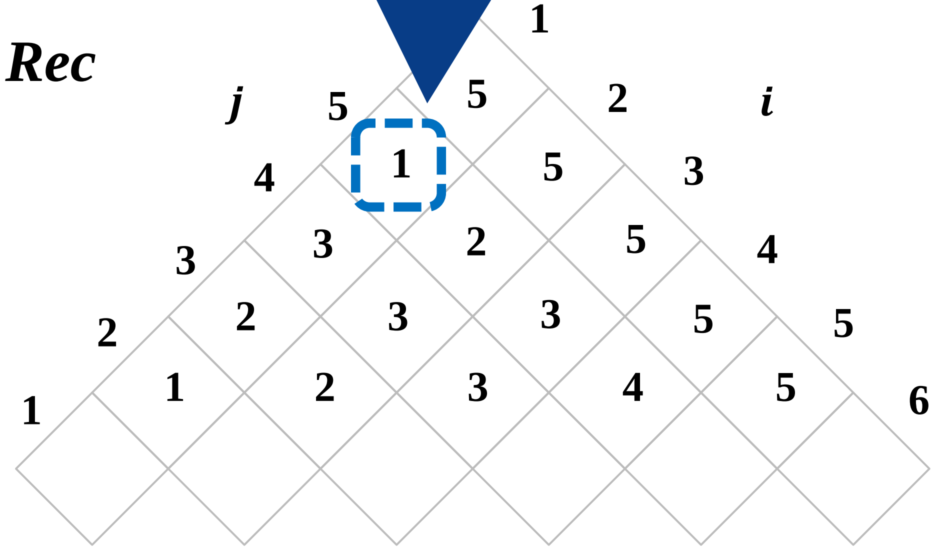


算法实例

- $U_1U_2U_3U_4U_5U_6$
- $\rightarrow (U_1U_2U_3U_4U_5)(U_6)$

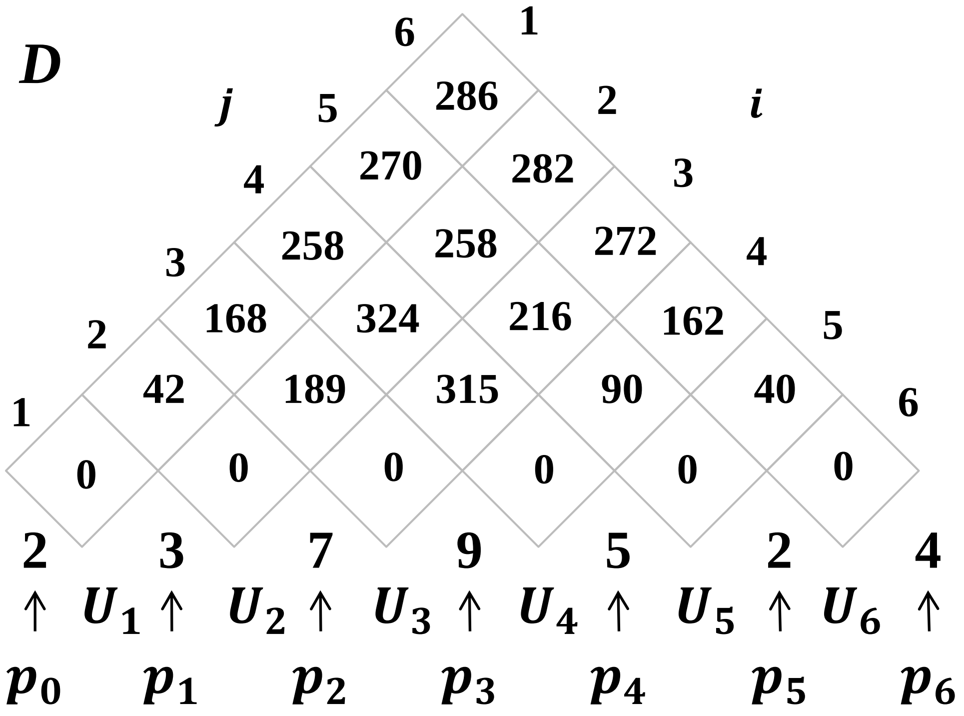
$Rec[1, Rec[1,6]] = Rec[1,5]$

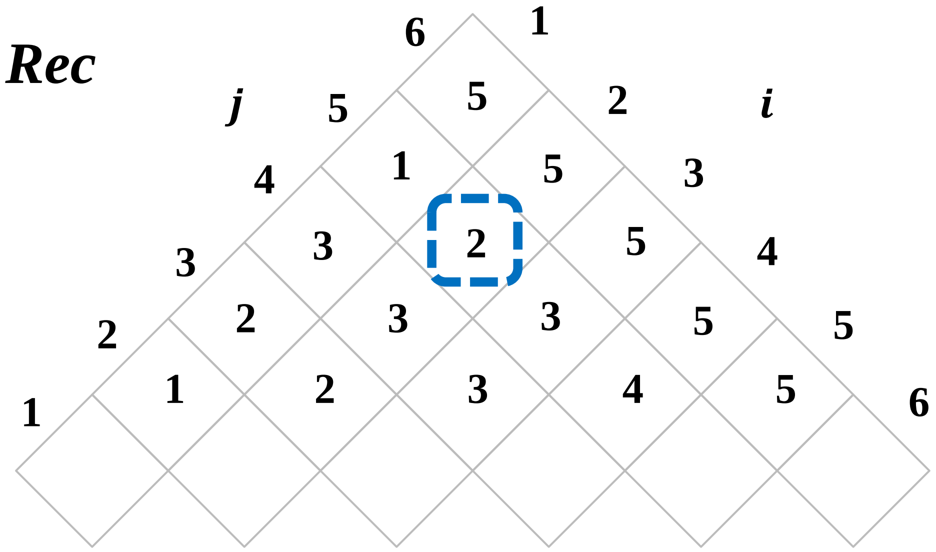




算法实例

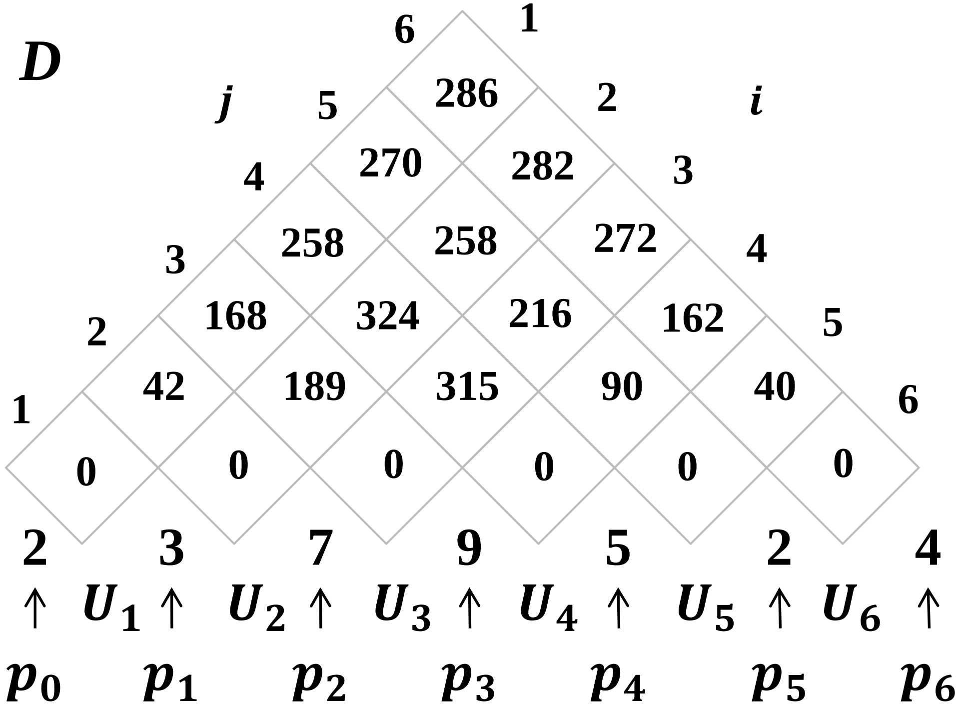
- $U_1U_2U_3U_4U_5U_6$
- $\rightarrow (U_1U_2U_3U_4U_5)(U_6)$
- $\rightarrow (U_1(U_2U_3U_4U_5))(U_6)$

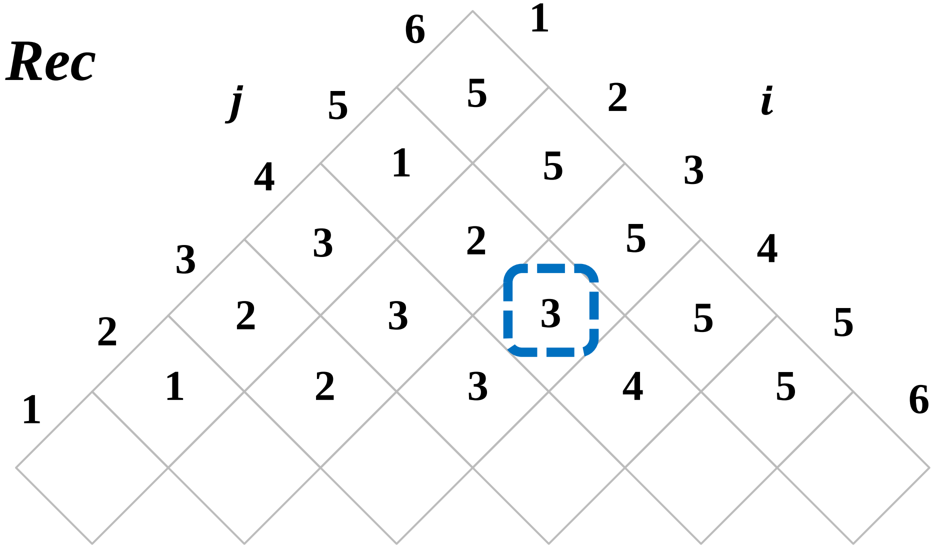




算法实例

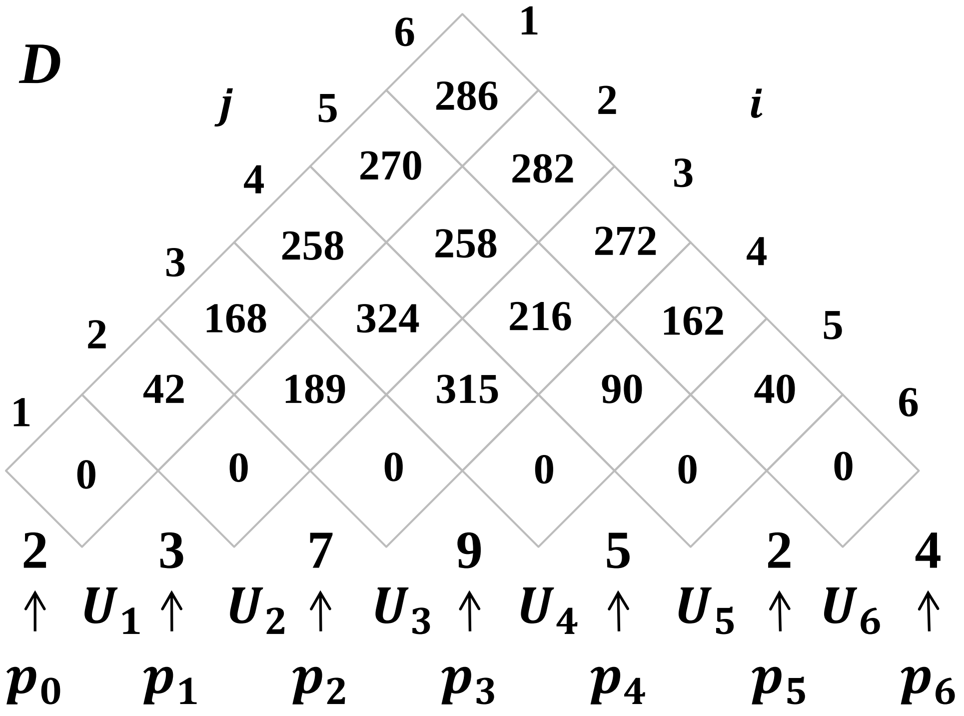
- $U_1U_2U_3U_4U_5U_6$
- $\rightarrow (U_1U_2U_3U_4U_5)(U_6)$
- $\rightarrow (U_1(U_2U_3U_4U_5))(U_6)$
- $\rightarrow (U_1(U_2(U_3U_4U_5)))(U_6)$

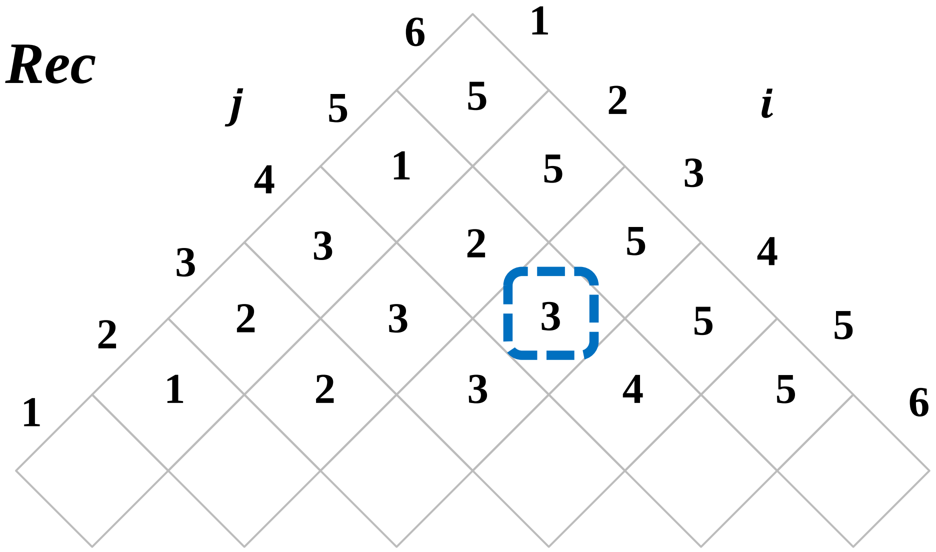




算法实例

- $U_1U_2U_3U_4U_5U_6$
- $\rightarrow (U_1U_2U_3U_4U_5)(U_6)$
- $\rightarrow (U_1(U_2U_3U_4U_5))(U_6)$
- $\rightarrow (U_1(U_2(U_3U_4U_5)))(U_6) \rightarrow (U_1(U_2((U_3)(U_4U_5))))(U_6)$





动态规划：伪代码

- **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

动态规划：伪代码

- **Matrix-Chain-Multiply(p, n)**

```
//动态规划
for l ← 2 to n do                                  区间长度从小到大
    for i ← 1 to n − l + 1 do
        j ← i + l − 1
        for k ← i to j − 1 do
            q ← D[i, k] + D[k + 1, j] + p[i − 1] * p[k] * p[j]
            if q < D[i, j] then
                D[i, j] ← q
                Rec[i, j] ← k
            end
        end
    end
end
return D[1, n], Rec
```

动态规划：伪代码

- **Matrix-Chain-Multiply(p, n)**

```
//动态规划
for l ← 2 to n do
    for i ← 1 to n − l + 1 do
        j ← i + l − 1
        for k ← i to j − 1 do
            q ← D[i, k] + D[k + 1, j] + p[i − 1] ∗ p[k] ∗ p[j]
            if q < D[i, j] then
                D[i, j] ← q
                Rec[i, j] ← k
            end
        end
    end
end
return D[1, n], Rec
```

依次计算子问题

动态规划：伪代码

- **Matrix-Chain-Multiply(p, n)**

```
//动态规划
for l ← 2 to n do
    for i ← 1 to n − l + 1 do
        j ← i + l − 1
        for k ← i to j − 1 do
            q ← D[i, k] + D[k + 1, j] + p[i − 1] ∗ p[k] ∗ p[j]
            if q < D[i, j] then
                D[i, j] ← q
                Rec[i, j] ← k
            end
        end
    end
end
return D[1, n], Rec
```

枚举所有分割位置

动态规划：伪代码

- **Matrix-Chain-Multiply(p, n)**

```
//动态规划
for l ← 2 to n do
    for i ← 1 to n − l + 1 do
        j ← i + l − 1
        for k ← i to j − 1 do
            q ← D[i, k] + D[k + 1, j] + p[i − 1] ∗ p[k] ∗ p[j]
            if q < D[i, j] then
                D[i, j] ← q
                Rec[i, j] ← k
            end
        end
    end
end
return D[1, n], Rec
```

计算最少乘法次数

动态规划：伪代码

- **Matrix-Chain-Multiply(p, n)**

```
//动态规划
for l ← 2 to n do
    for i ← 1 to n − l + 1 do
        j ← i + l − 1
        for k ← i to j − 1 do
            q ← D[i, k] + D[k + 1, j] + p[i − 1] * p[k] * p[j]
            if q < D[i, j] then
                D[i, j] ← q
                Rec[i, j] ← 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
print "("
Print-Matrix-Chain(U, Rec, i, Rec[i, j])
print ")("
Print-Matrix-Chain(U, Rec, Rec[i, j] + 1, j)
print ")"
return
```

链长为1直接输出

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

时间复杂度分析

- **Matrix-Chain-Multiply(p, n)**

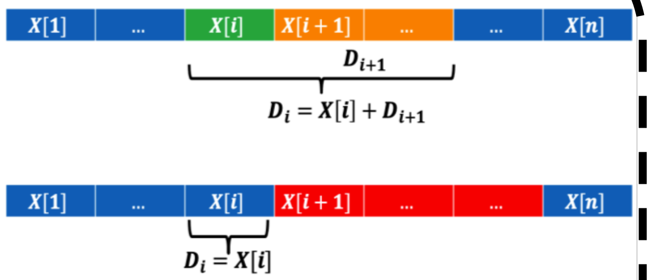
```
//动态规划
for l ← 2 to n do
    for i ← 1 to n − l + 1 do
        j ← i + l − 1
        D[i, j] ← ∞
        for k ← i to j − 1 do
            q ← D[i, k] + D[k + 1, j] + p[i − 1] ∗ p[k] ∗ p[j]
            if q < D[i, j] then
                D[i, j] ← q
                Rec[i, j] ← k
            end
        end
    end
end
return D[1, n], Rec
```

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

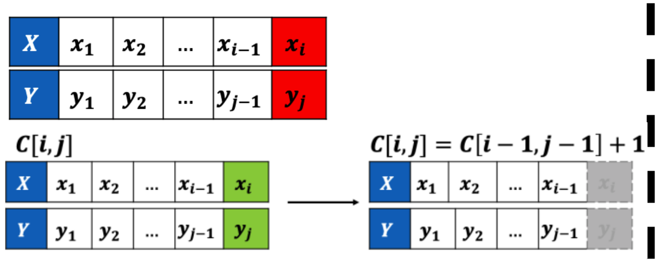
时间复杂度：$O(n^3)$

总结

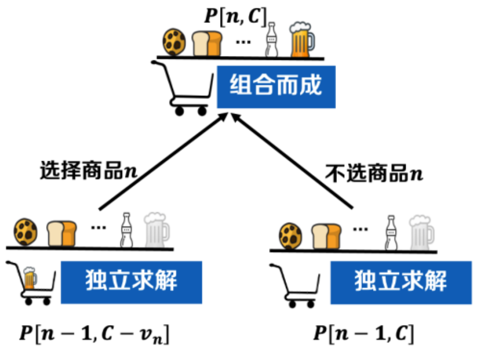
- 问题最优解依赖的子问题数量各不相同



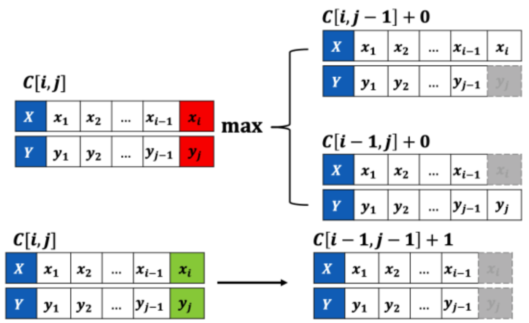
最大字数组问题



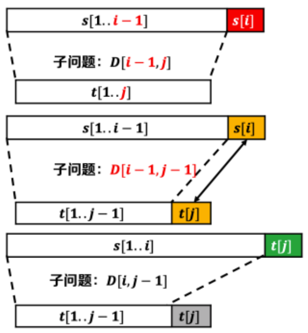
最长公共子串问题



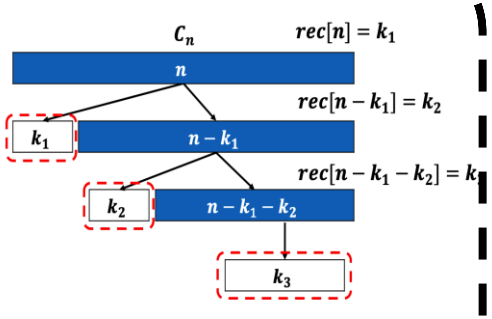
0-1背包问题



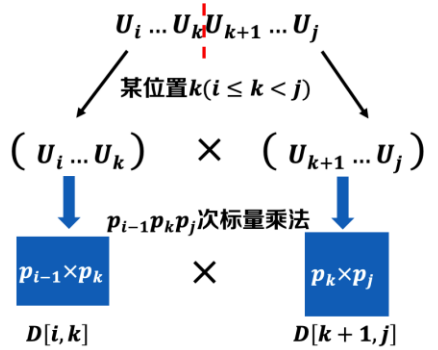
最长公共子序列问题



最小编辑距离问题



钢条切割问题



矩阵链乘法问题

依赖一个子问题　　依赖常数个子问题　　依赖枚举子问题

谢谢

BDA

Beihang University