Design and Analysis of Algorithms

Part II: Dynamic Programming

Lecture 15: Rod-Cutting

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

- 钢条切割
  - 现有一段长度为10的钢条，可以零成本将其切割为多段长度更小钢条

钢条长度	0	1	2	3	4	5	6	7	8	9	10
价格p	0	1	5	8	9	10	17	17	20	24	24

问题背景

- 钢条切割
  - 现有一段长度为10的钢条，可以零成本将其切割为多段长度更小钢条

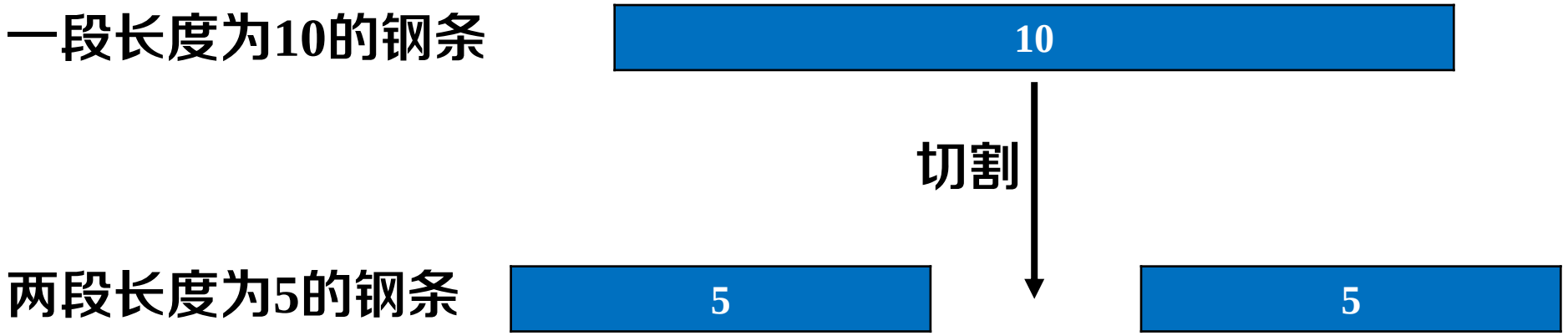
钢条长度	0	1	2	3	4	5	6	7	8	9	10
价格p	0	1	5	8	9	10	17	17	20	24	24

一段长度为10的钢条

问题背景

- 钢条切割
  - 现有一段长度为10的钢条，可以零成本将其切割为多段长度更小钢条

钢条长度	0	1	2	3	4	5	6	7	8	9	10
价格p	0	1	5	8	9	10	17	17	20	24	24

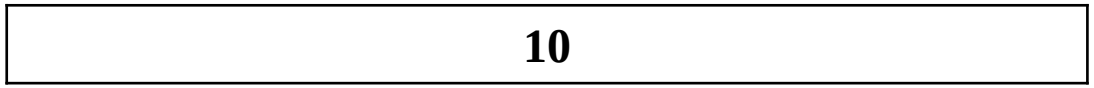


问题背景

- 钢条切割
  - 现有一段长度为10的钢条，可以零成本将其切割为多段长度更小钢条

钢条长度	0	1	2	3	4	5	6	7	8	9	10
价格p	0	1	5	8	9	10	17	17	20	24	24

切割方案		总收益
方案1	{10}	24

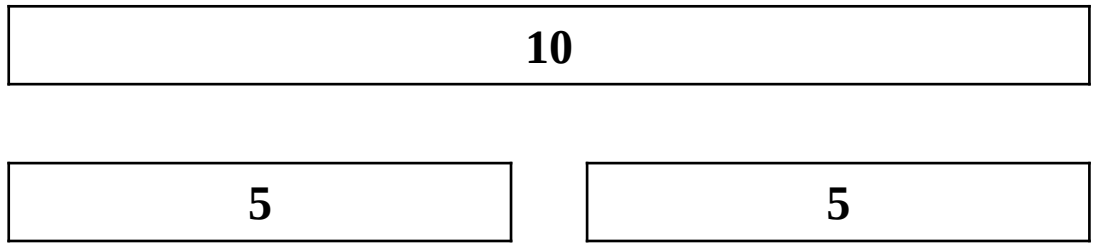


问题背景

- 钢条切割
  - 现有一段长度为10的钢条，可以零成本将其切割为多段长度更小钢条

钢条长度	0	1	2	3	4	5	6	7	8	9	10
价格p	0	1	5	8	9	10	17	17	20	24	24

切割方案		总收益
方案1	{10}	24
方案2	{5,5}	10+10=20

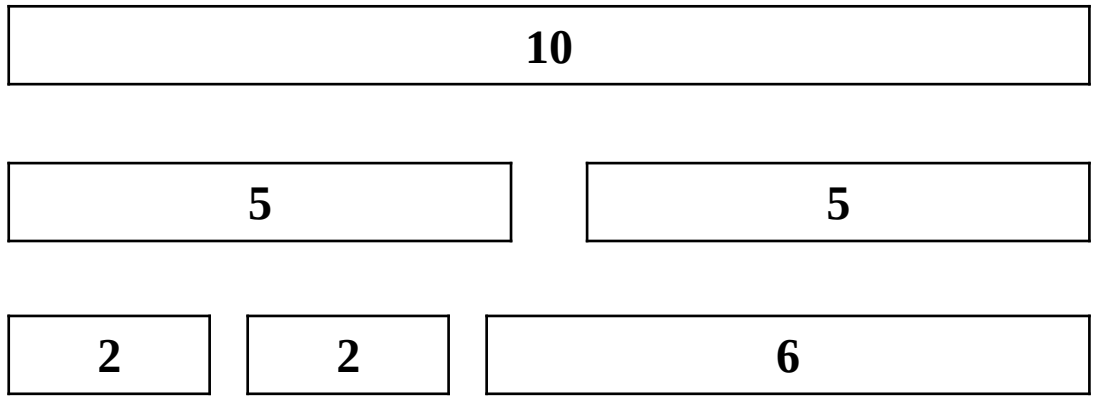


问题背景

- 钢条切割
  - 现有一段长度为10的钢条，可以零成本将其切割为多段长度更小钢条

钢条长度	0	1	2	3	4	5	6	7	8	9	10
价格p	0	1	5	8	9	10	17	17	20	24	24

切割方案		总收益
方案1	{10}	24
方案2	{5,5}	10+10=20
方案3	{2,2,6}	5+5+17=27

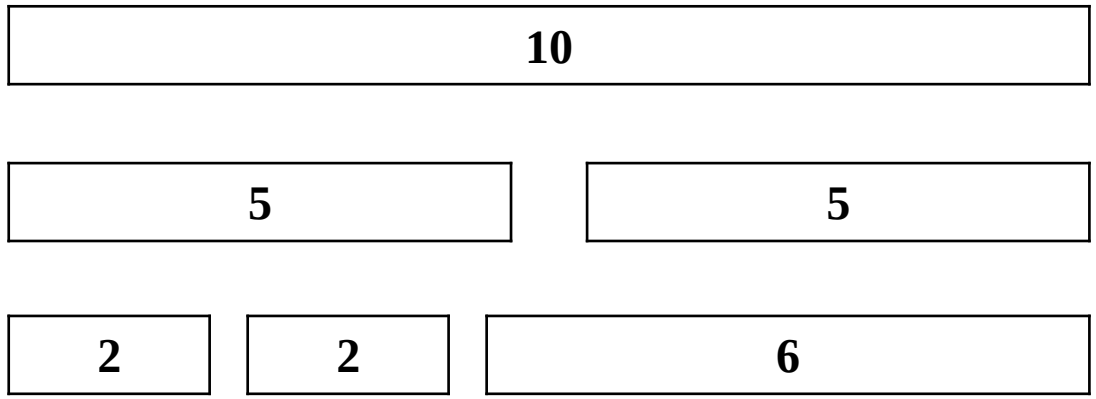


问题背景

- 钢条切割
  - 现有一段长度为10的钢条，可以零成本将其切割为多段长度更小钢条

钢条长度	0	1	2	3	4	5	6	7	8	9	10
价格p	0	1	5	8	9	10	17	17	20	24	24

切割方案		总收益
方案1	{10}	24
方案2	{5,5}	10+10=20
方案3	{2,2,6}	5+5+17=27



问题：怎样合理切割，使总收益最大?

问题定义

- 形式化定义

钢条切割问题

Rod Cutting Problem

输入

- 钢条长度n

问题定义

- 形式化定义

钢条切割问题

Rod Cutting Problem

输入

- 钢条长度n
- 价格表$p_l(1 \leq l \leq n)$：表示长度为l的钢条价格

问题定义

- 形式化定义

钢条切割问题

Rod Cutting Problem

输入

- 钢条长度n
- 价格表$p_l(1 \le l \le n)$：表示长度为l的钢条价格

输出

- 求解一组切割方案$T =< c_1, c_2, \dots, c_m >$，令

$$\max \sum_{l=1}^{m} p_{c_l}$$

$$s.t. \sum_{l=1}^{m} c_l = n$$

问题定义

- 形式化定义

钢条切割问题

Rod Cutting Problem

输入

- 钢条长度n
- 价格表$p_l(1 \le l \le n)$：表示长度为l的钢条价格

输出

- 求解一组切割方案$T =< c_1, c_2, \dots, c_m >$，令

$$\max \sum_{l=1}^{m} p_{c_l}$$

优化目标

$$s.t. \sum_{l=1}^{m} c_l = n$$

问题定义

- 形式化定义

钢条切割问题

Rod Cutting Problem

输入

- 钢条长度n
- 价格表$p_l(1 \le l \le n)$：表示长度为l的钢条价格

输出

- 求解一组切割方案$T =< c_1, c_2, \dots, c_m >$，令

$$\max \sum_{l=1}^{m} p_{c_l}$$

优化目标

$$s.t. \sum_{l=1}^{m} c_l = n$$

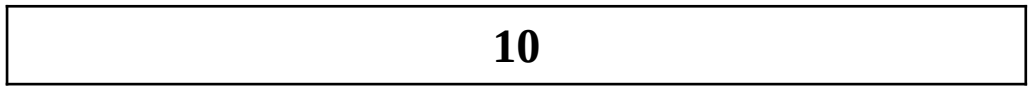
约束条件

问题简化

- 假设至多切割1次
  - 枚举所有可能的切割位置

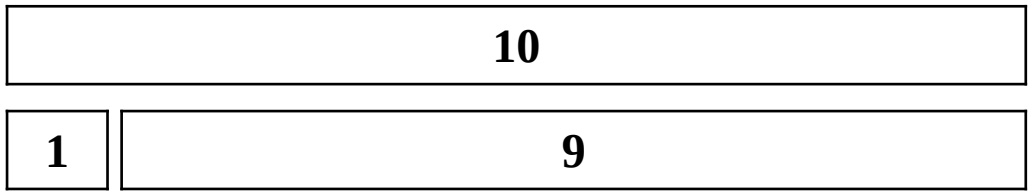
问题简化

- 假设至多切割1次
  - 枚举所有可能的切割位置
    - 不切：$p[10]$



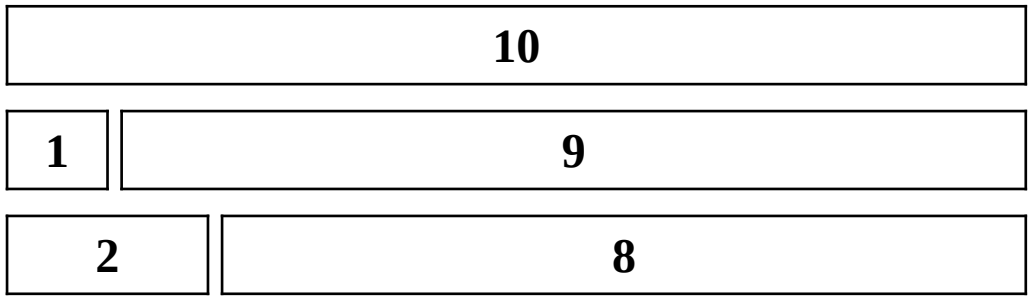
问题简化

- 假设至多切割1次
  - 枚举所有可能的切割位置
    - 不切：$p[10]$
    - 切割：$p[i]+p[10-i]$



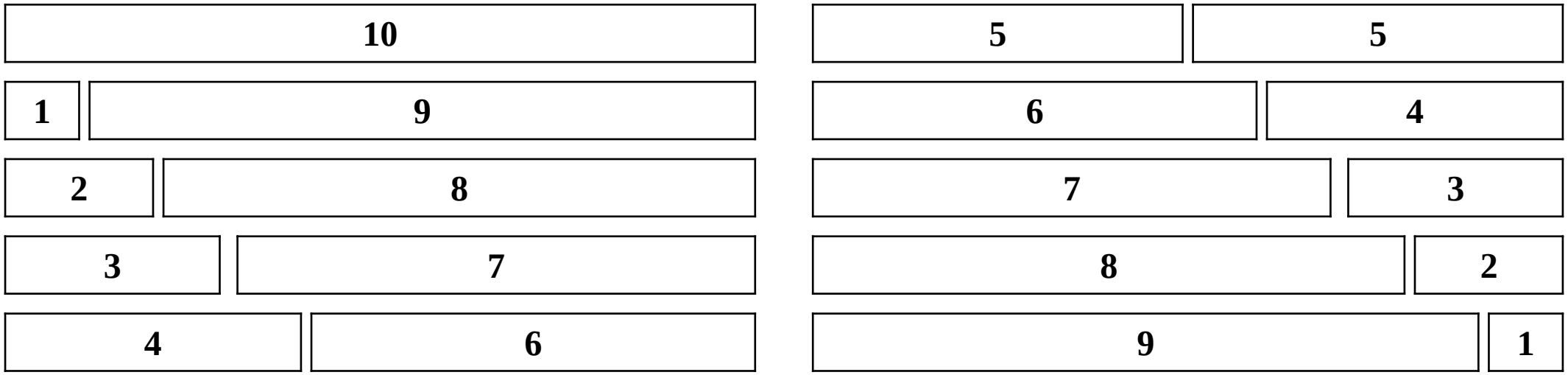
问题简化

- 假设至多切割1次
  - 枚举所有可能的切割位置
    - 不切：$p[10]$
    - 切割：$p[i]+p[10-i]$



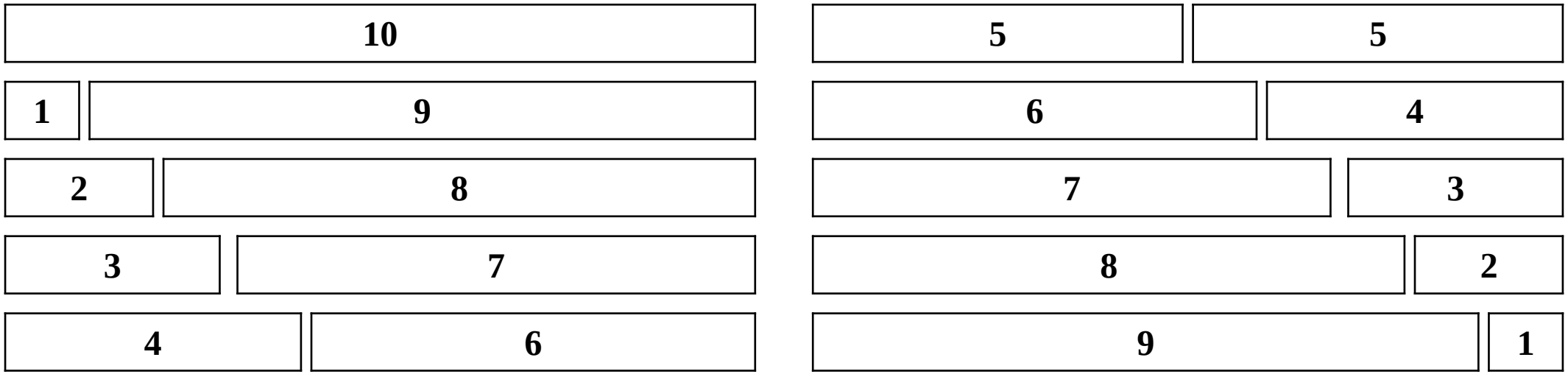
问题简化

- 假设至多切割1次
  - 枚举所有可能的切割位置
    - 不切：$p[10]$
    - 切割：$p[i]+p[10-i]$



问题简化

- 假设至多切割1次
  - 枚举所有可能的切割位置
    - 不切：$p[10]$
    - 切割：$p[i]+p[10-i]$
  - 最大收益$\max\limits_{1\le i\le 9}\{p[i]+p[10-i],p[10]\}$

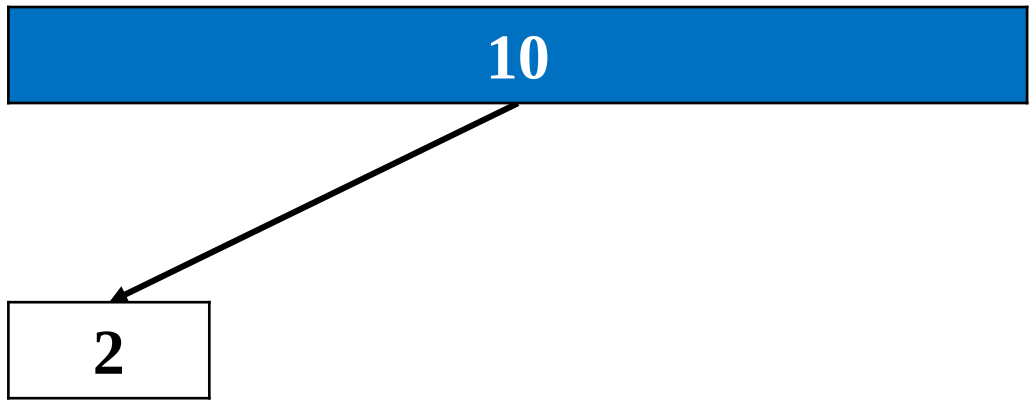


问题简化

- 假设至多切割2次

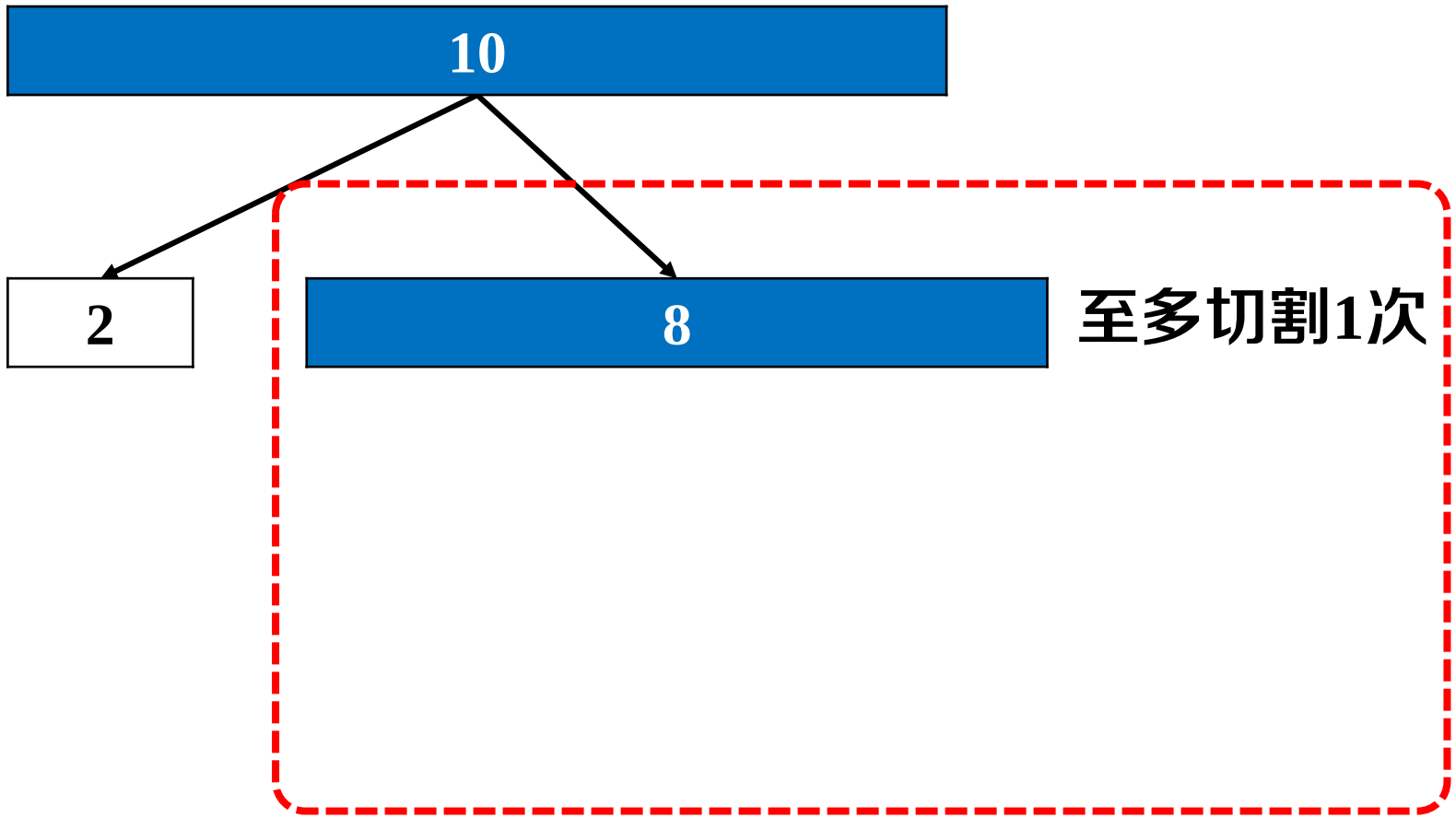
问题简化

- 假设至多切割2次
  - 先将钢条切割出一段



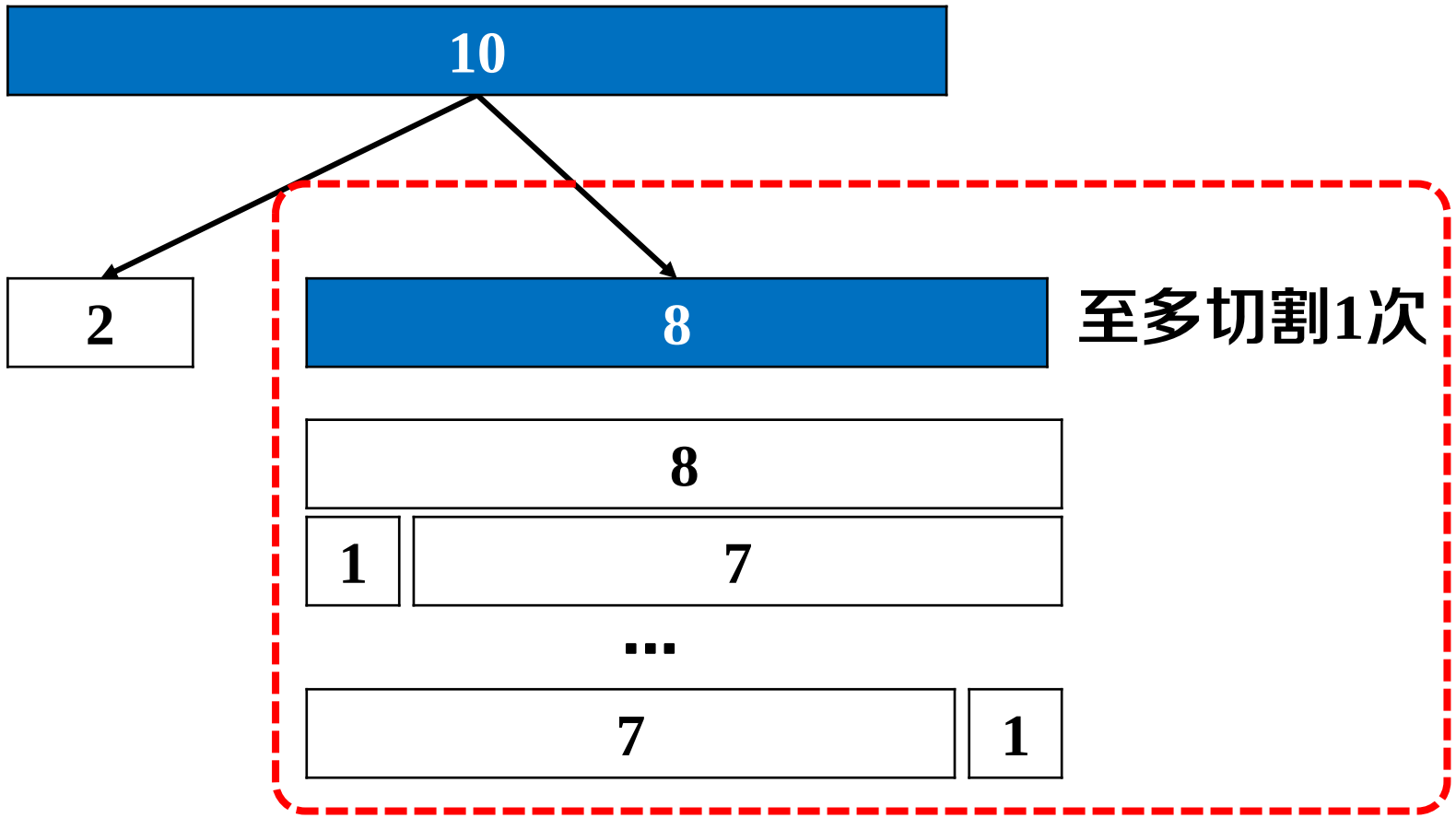
问题简化

- 假设至多切割2次
  - 先将钢条切割出一段
  - 在剩余钢条中继续切割



问题简化

- 假设至多切割2次
  - 先将钢条切割出一段
  - 在剩余钢条中继续切割



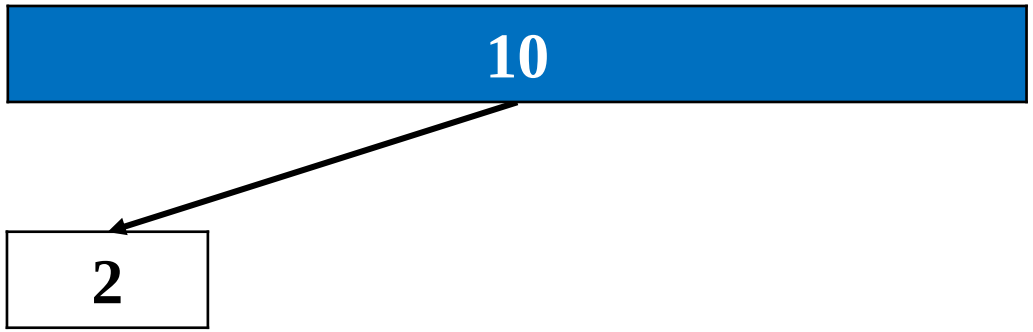
问题观察

- 原始问题不限制切割次数

枚举切割长度，或不切割

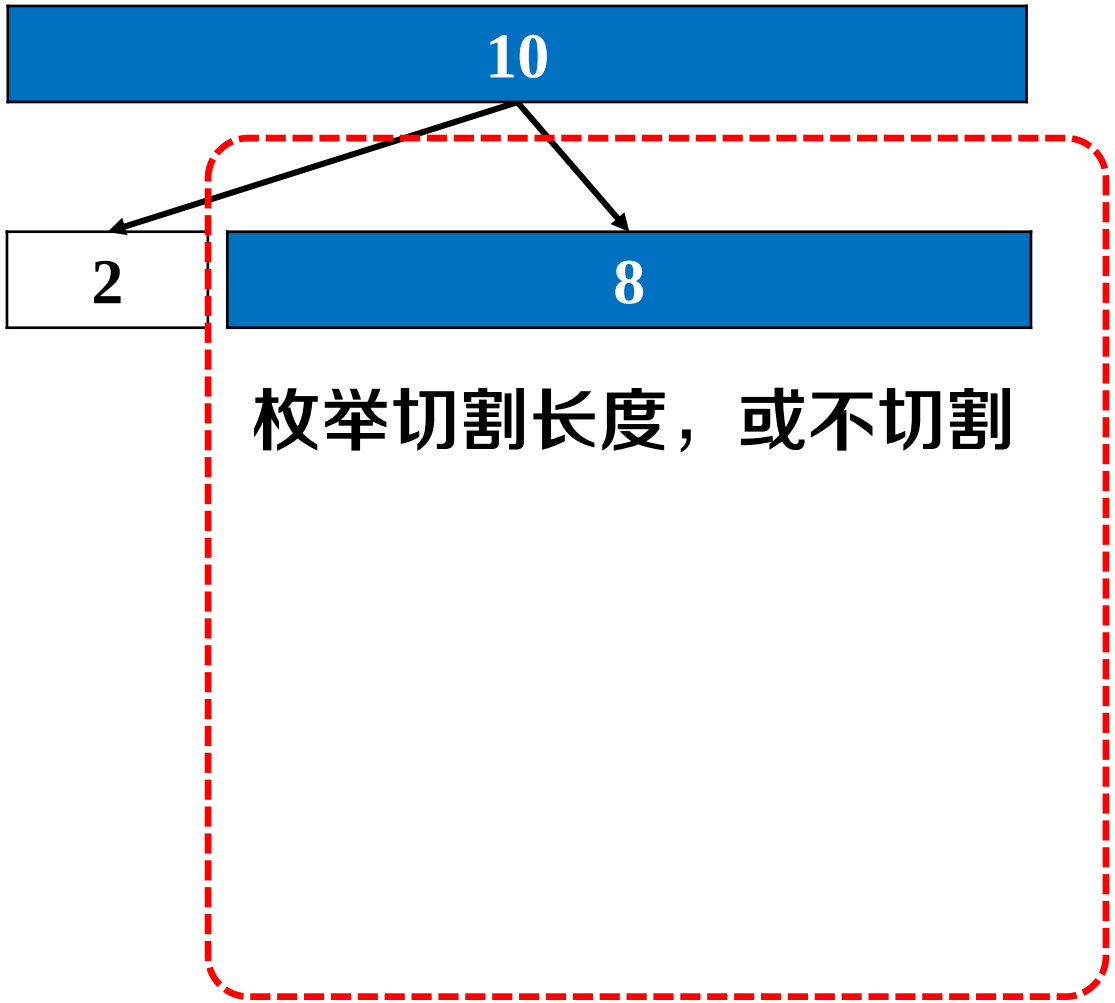
问题观察

- 原始问题不限制切割次数



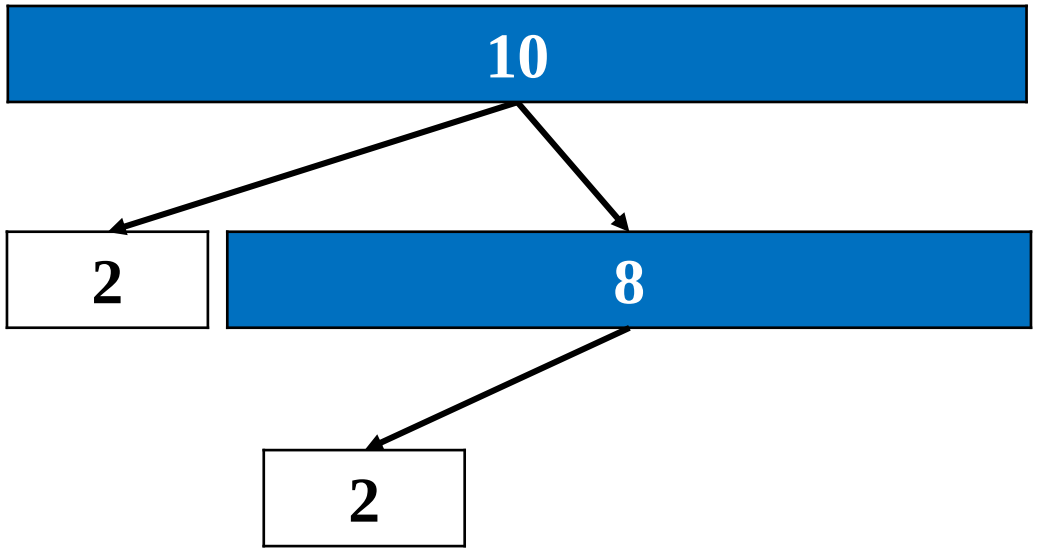
问题观察

- 原始问题不限制切割次数



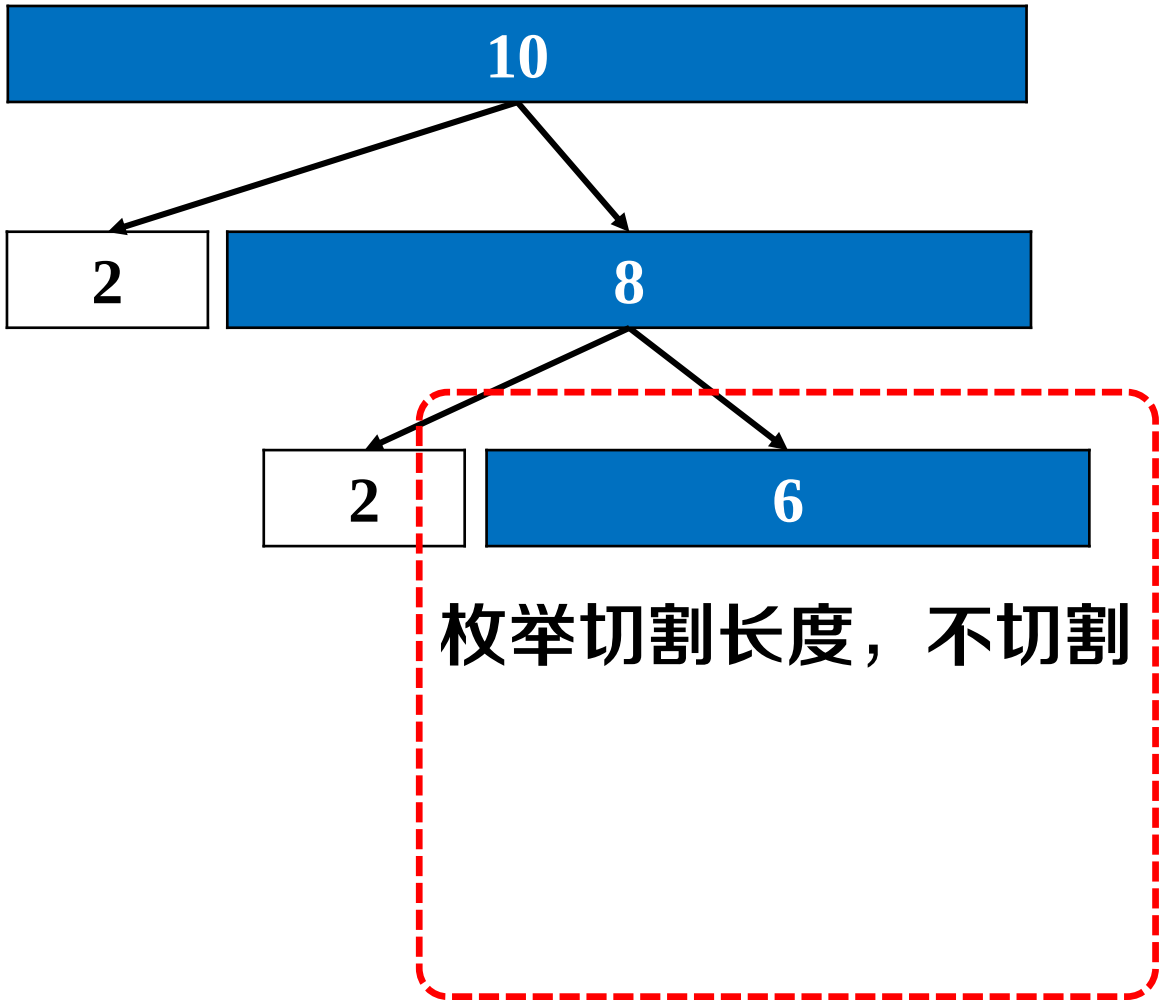
问题观察

- 原始问题不限制切割次数



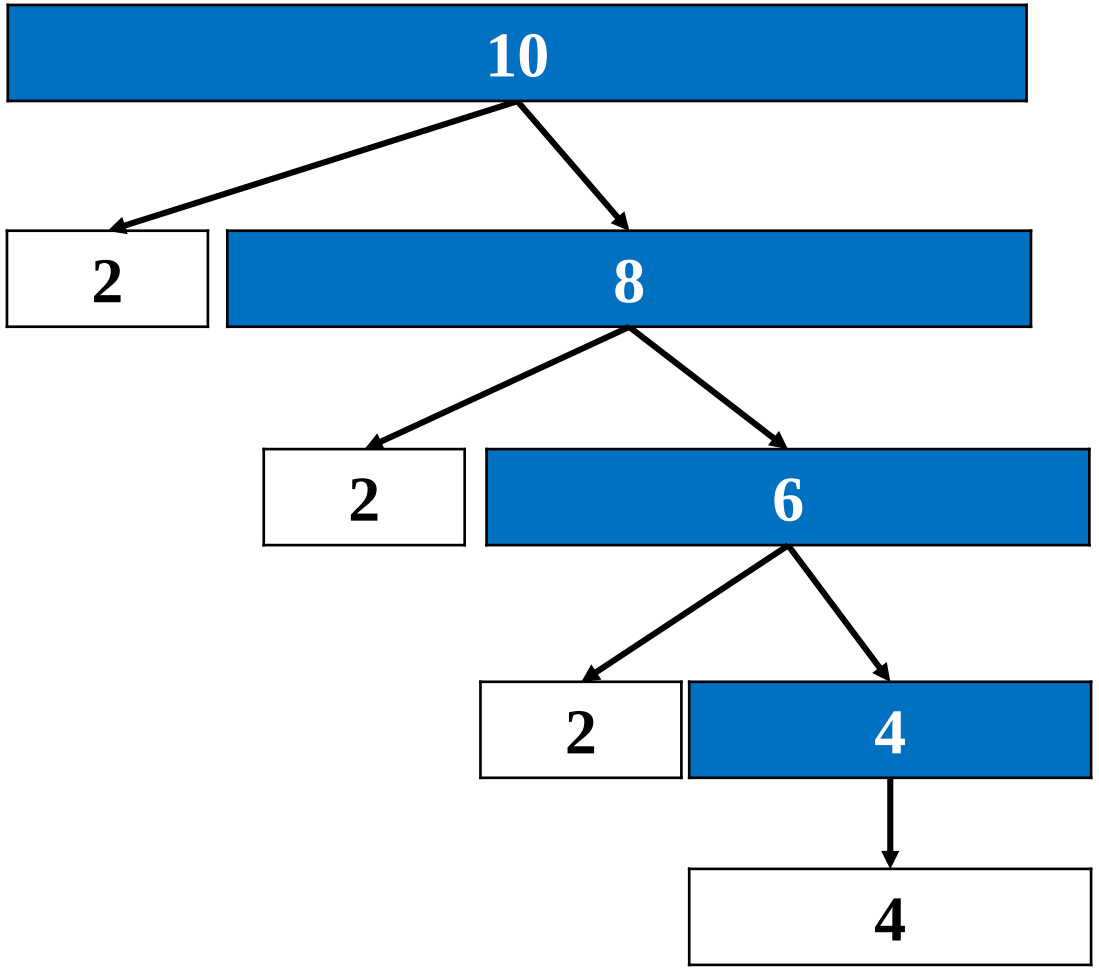
问题观察

- 原始问题不限制切割次数



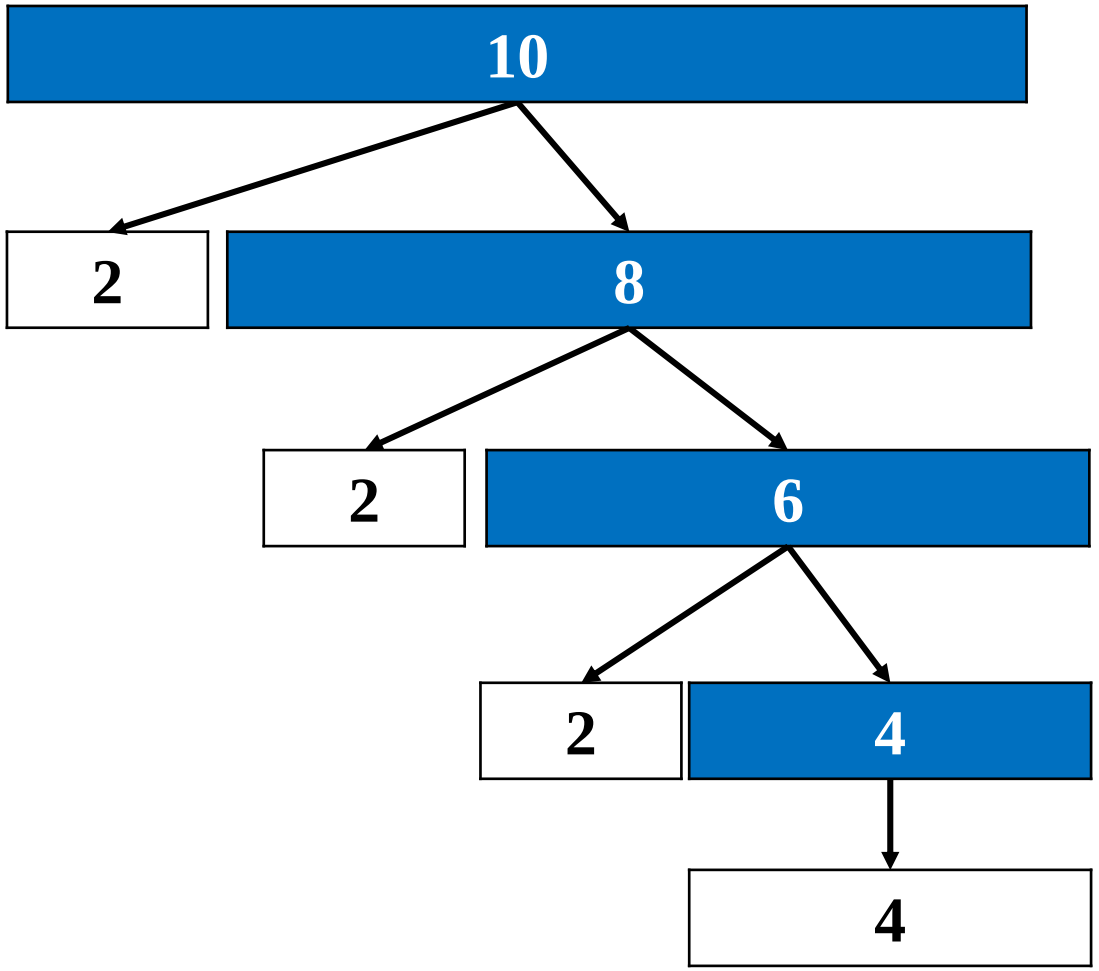
问题观察

- 原始问题不限制切割次数



问题观察

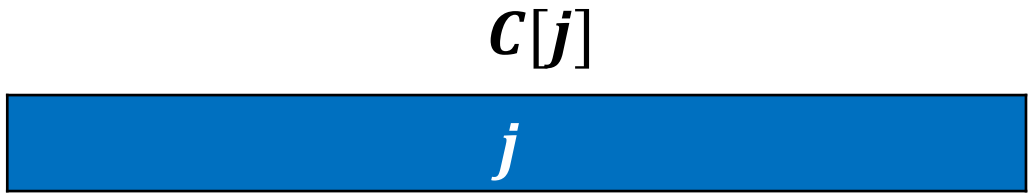
- 原始问题不限制切割次数

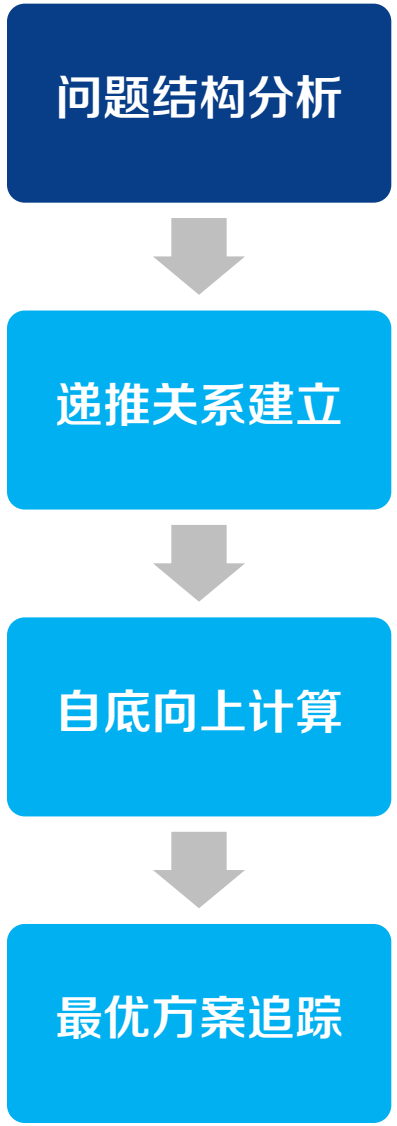


- 可能存在最优子结构和重叠子问题

问题结构分析

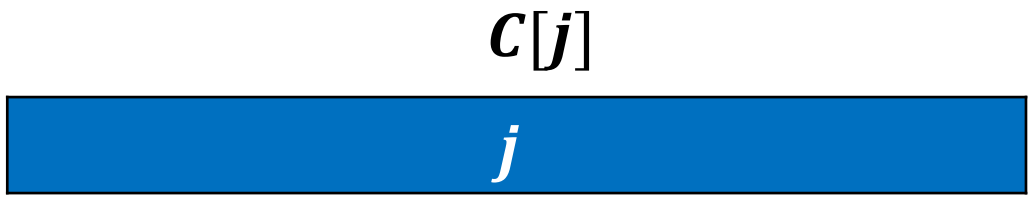
- 给出问题表示
  - $c[j]$：切割长度为j的钢条可得最大总收益



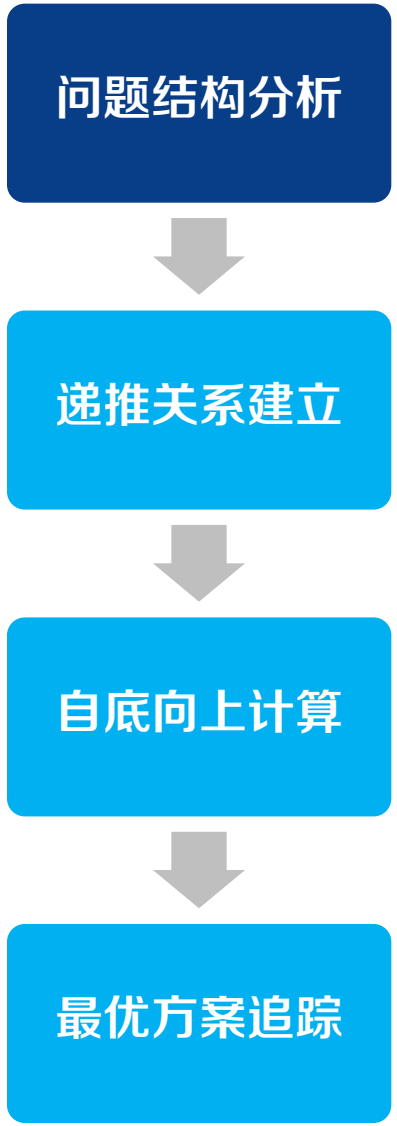


问题结构分析

- 给出问题表示
  - $C[j]$：切割长度为j的钢条可得最大总收益



- 明确原始问题
  - $C[n]$：切割长度为n的钢条可得最大总收益



递推关系建立：分析最优（子）结构

$C[10]$

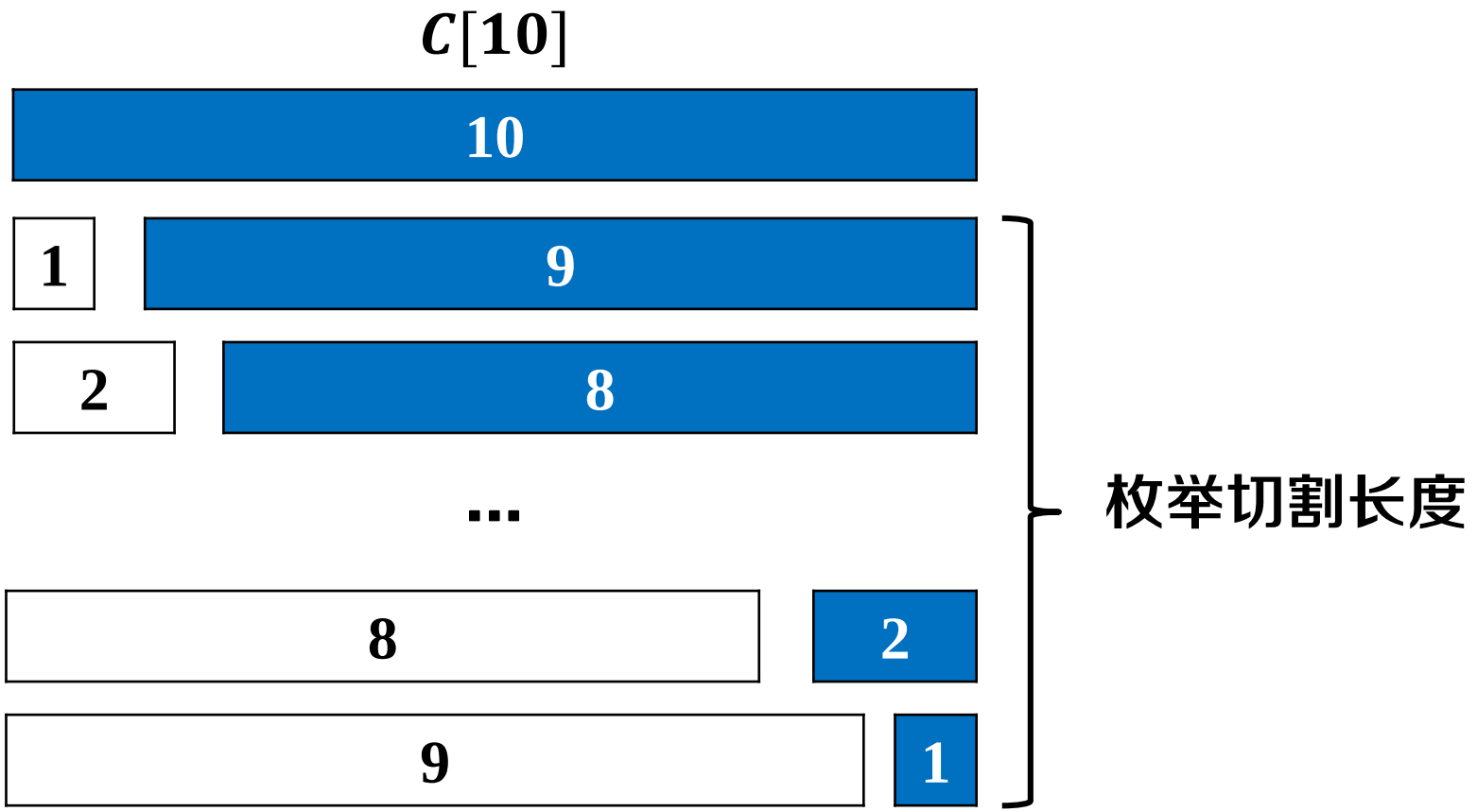
问题结构分析

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



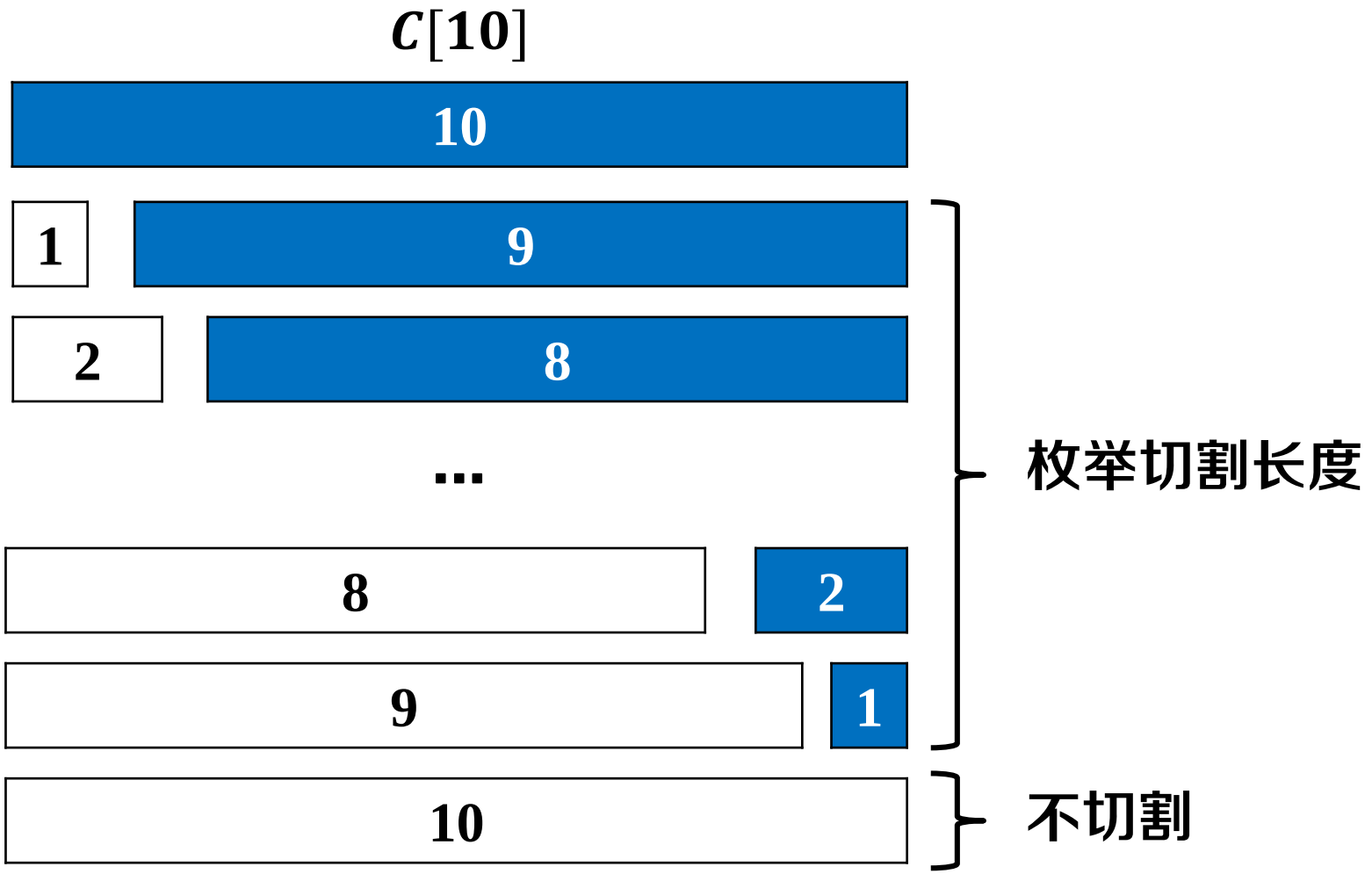
问题结构分析

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



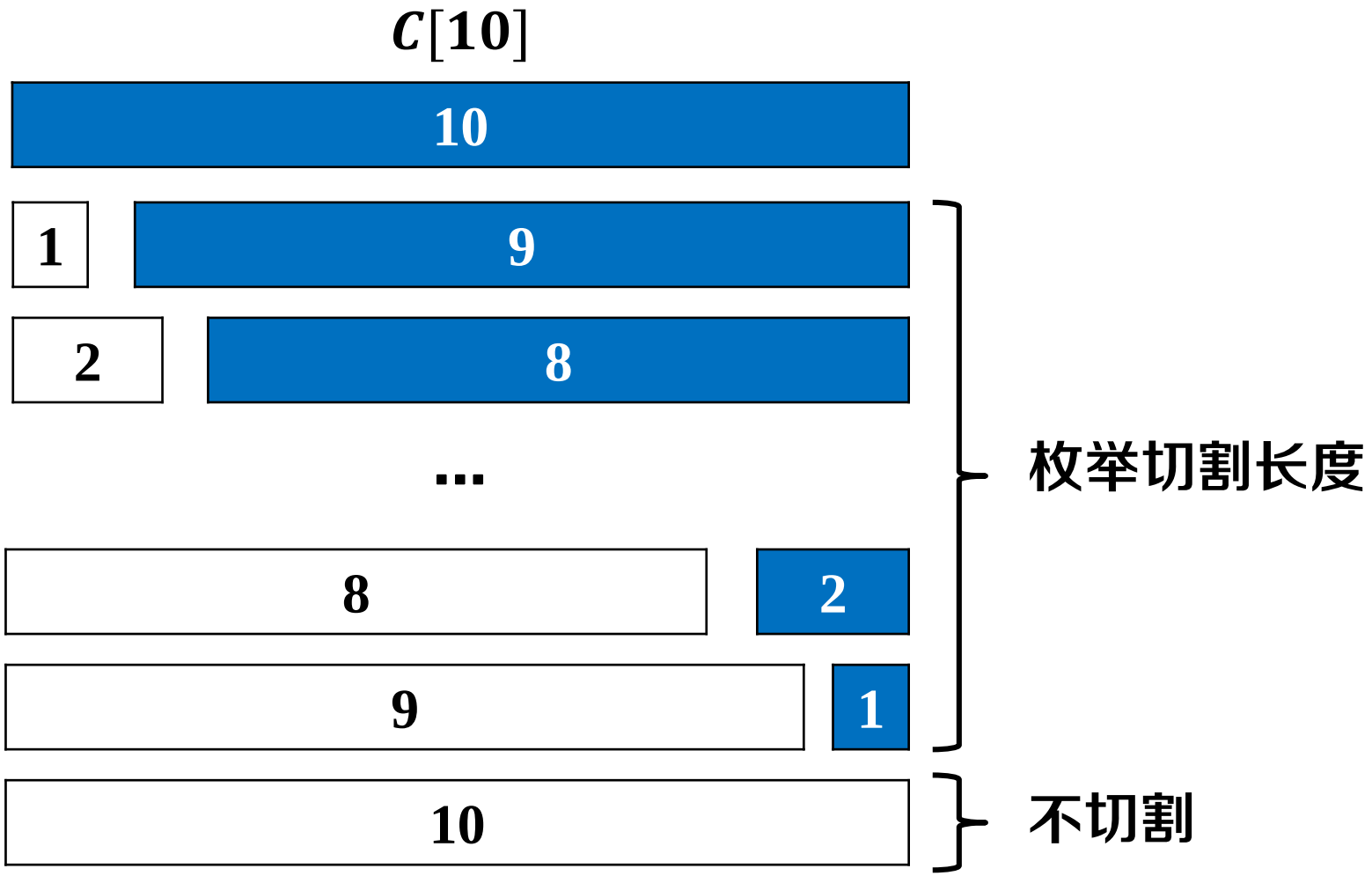
问题结构分析

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



- $C[10] = \max_{1 \le i \le 9}\{p[i] + C[10-i], p[10]\}$

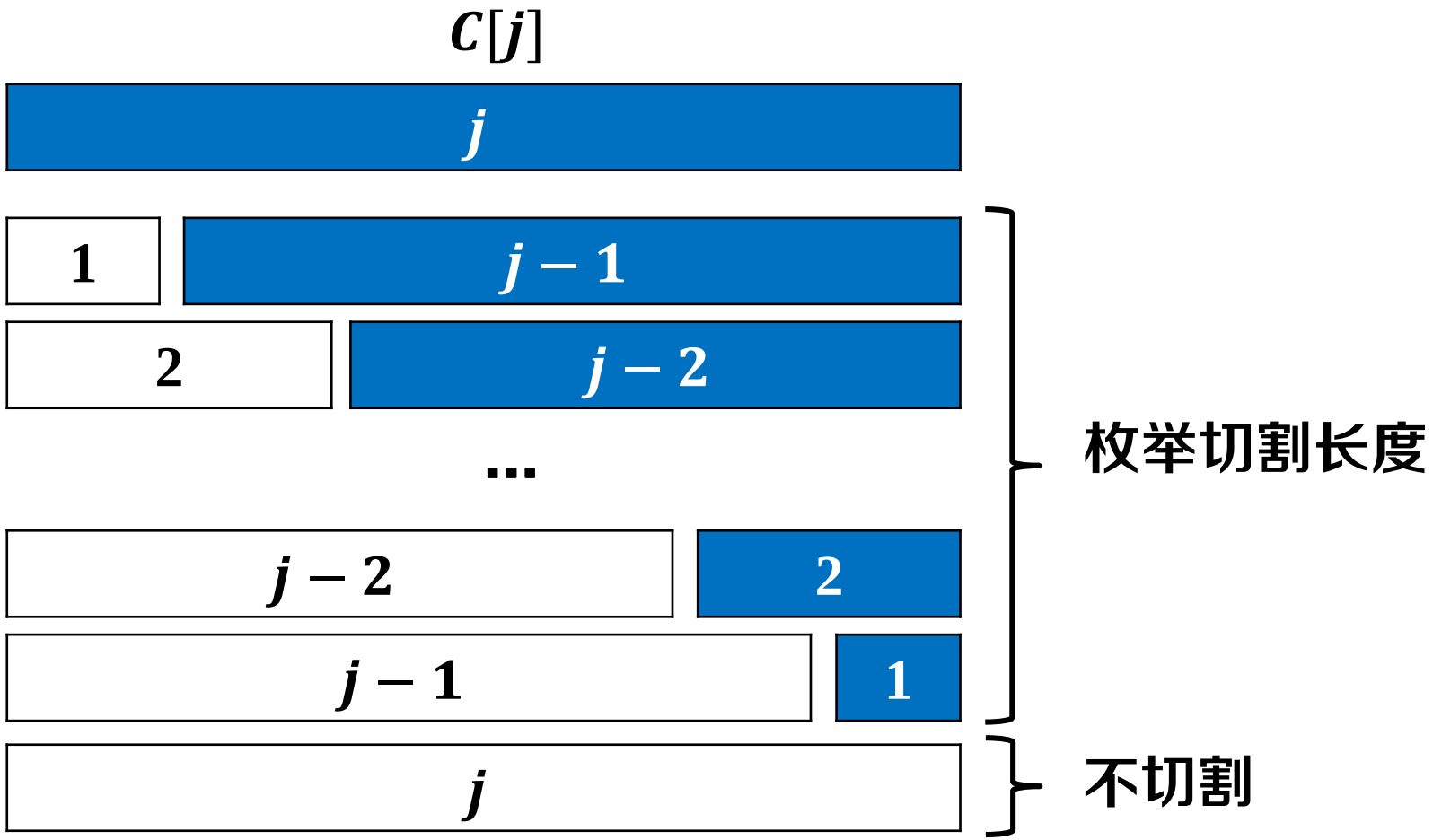
问题结构分析

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



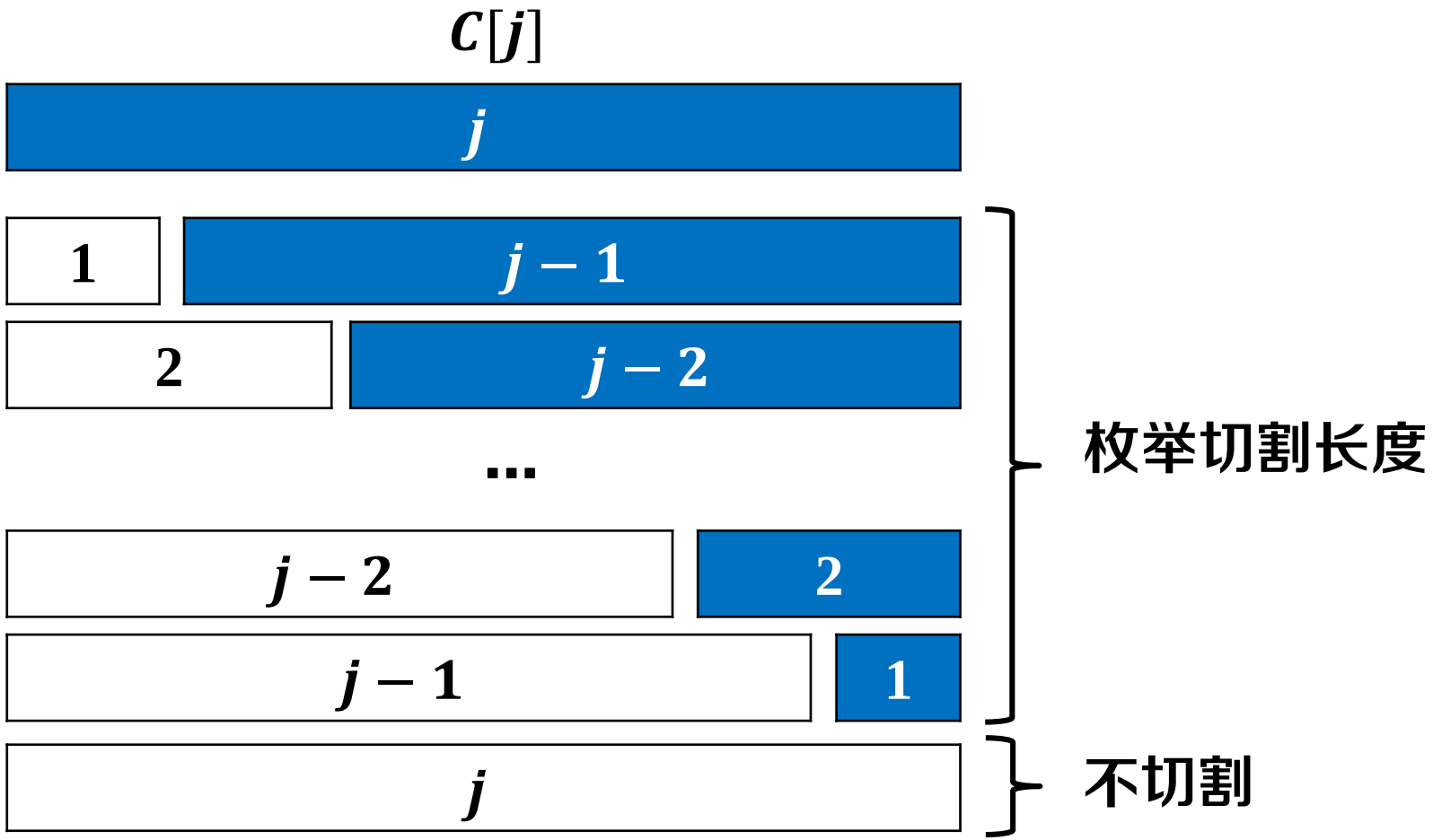
问题结构分析

递推关系建立

自底向上计算

最优方案追踪

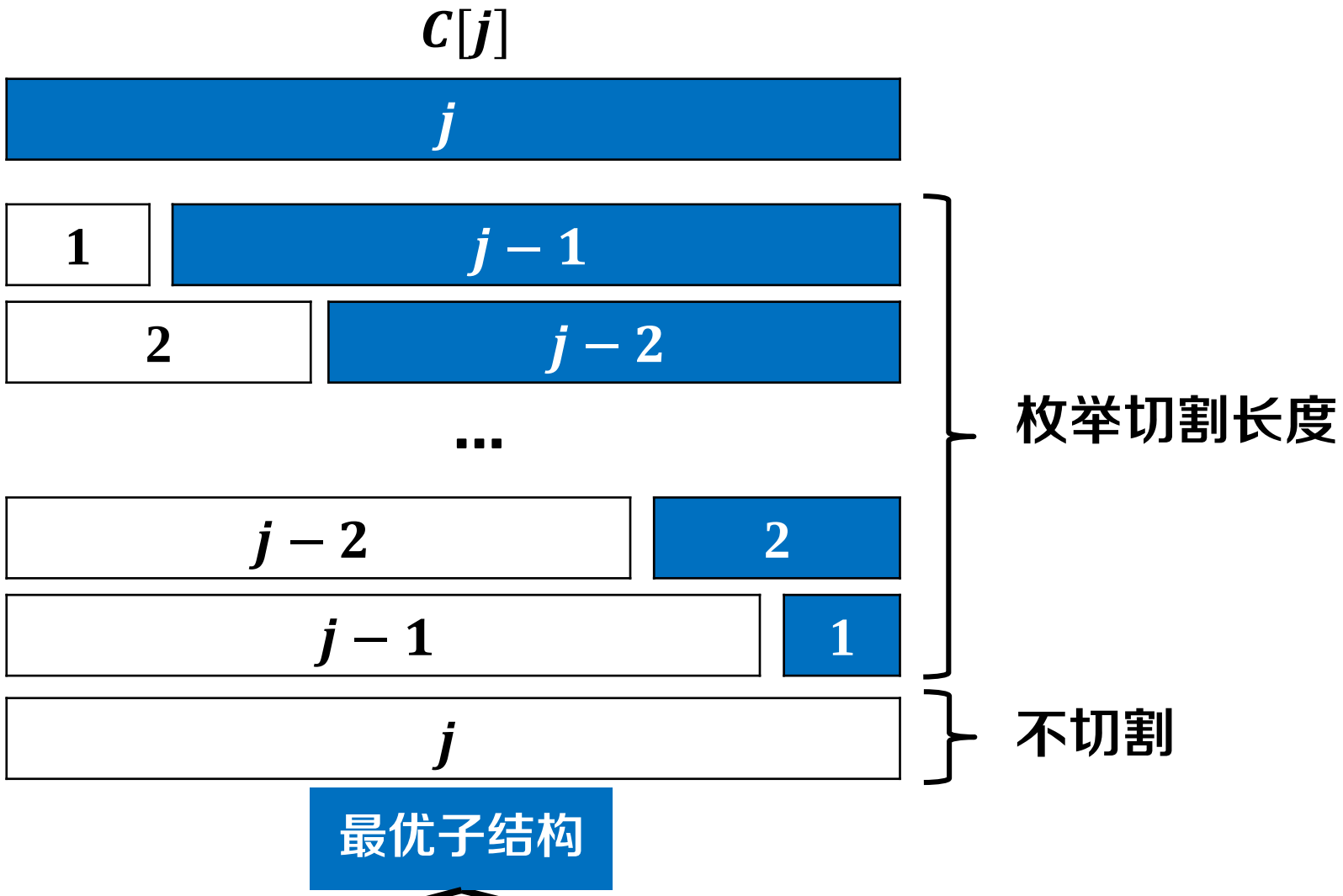
递推关系建立：分析最优（子）结构



- 问题结构分析
- 递推关系建立
- 自底向上计算
- 最优方案追踪

- $C[j] = \max_{1 \le i \le j-1} \{p[i] + C[j-i], p[j]\}$

递推关系建立：分析最优（子）结构



- $C[j] = \max_{1 \le i \le j-1} \{p[i] + C[j-i], p[j]\}$

问题结构分析

递推关系建立

自底向上计算

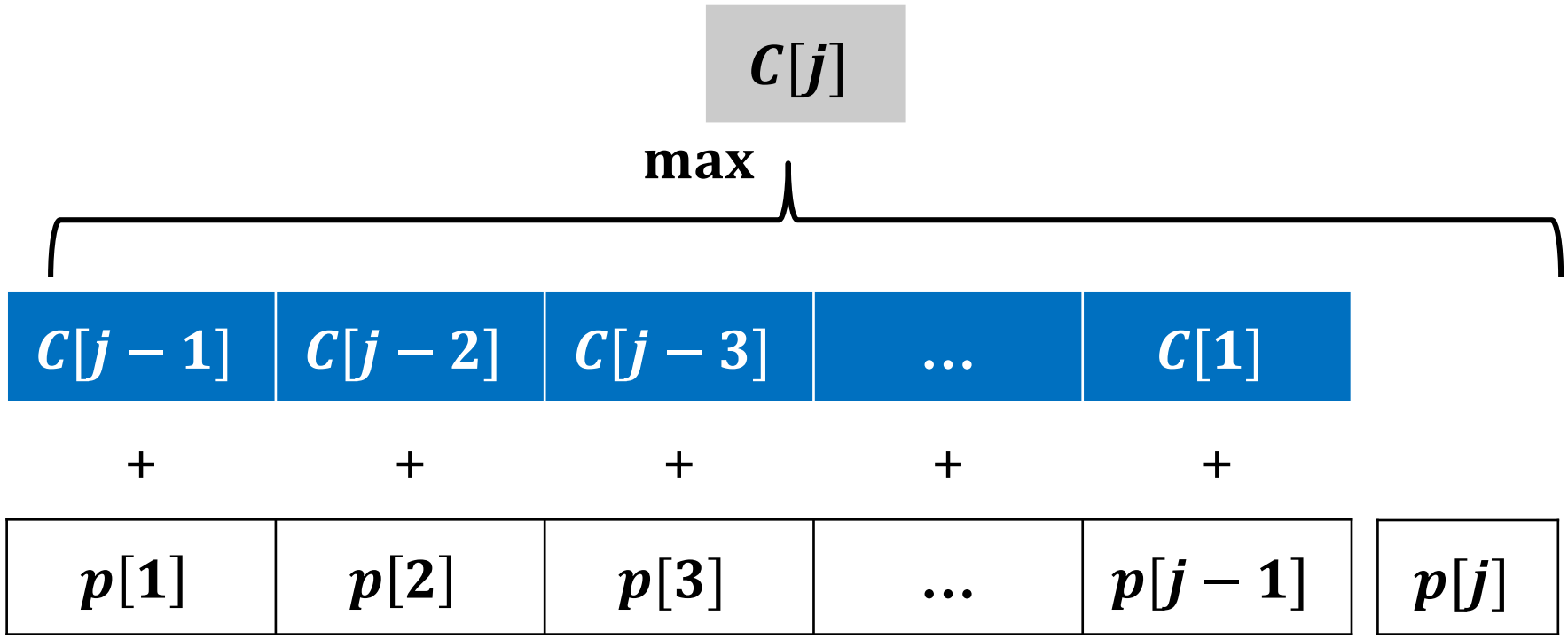
最优方案追踪

递推关系建立：构造递推公式

- 对于每个钢条长度j
  - $C[j] = \max_{1 \le i \le j-1} \{p[i] + C[j-i], p[j]\}$

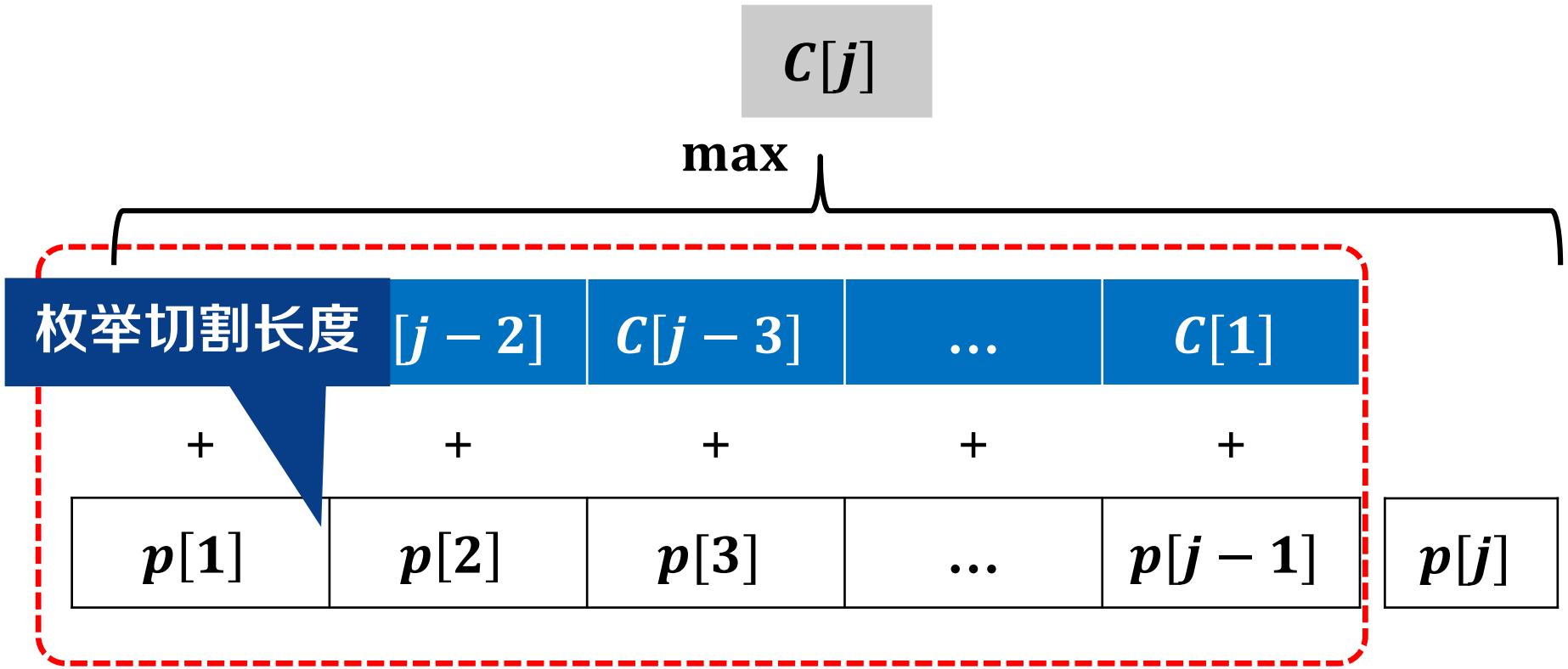
递推关系建立：构造递推公式

- 对于每个钢条长度j
  - $C[j] = \max_{1 \le i \le j-1} \{p[i] + C[j-i], p[j]\}$



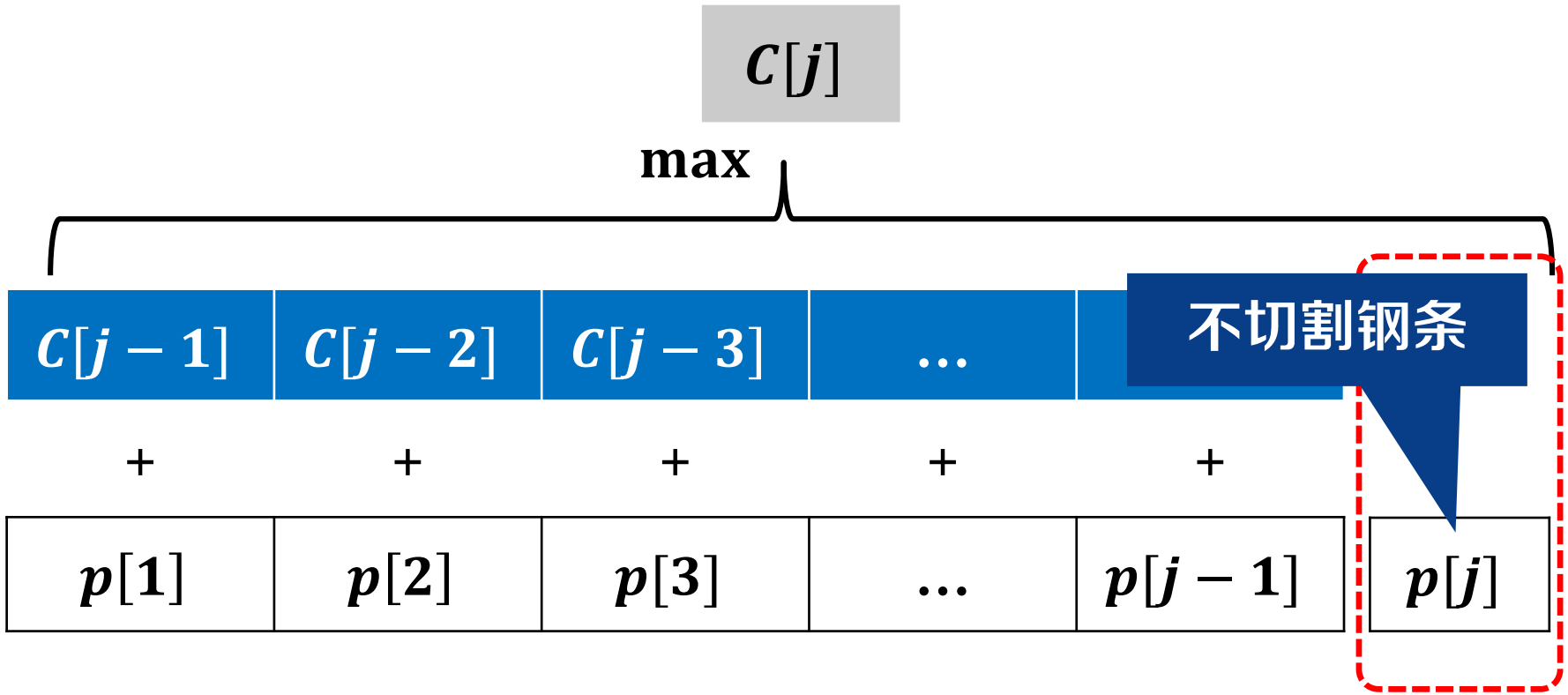
递推关系建立：构造递推公式

- 对于每个钢条长度j
  - $C[j] = \max_{1 \le i \le j-1} \{p[i] + C[j-i], p[j]\}$



递推关系建立：构造递推公式

- 对于每个钢条长度j
  - $C[j] = \max_{1 \le i \le j-1} \{p[i] + C[j-i], p[j]\}$



自底向上计算：确定计算顺序

已知钢条价格

$p[\,]$	$p[1]$	$p[2]$	…	$p[n]$

- 初始化
  - $C[0]=0$ 切割长度为0的钢条，总收益为0

初始化

	0	1	2	…	n
$C[\,]$	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

已知钢条价格

$p[\,]$	$p[1]$	$p[2]$	…	$p[n]$

- 初始化
  - $C[0]=0$ 切割长度为0的钢条，总收益为0
- 递推公式
  - $C[j]=\max\limits_{1\le i\le j-1}\{p[i]+C[j-i],p[j]\}$

自底向上计算

	0	1	2	…	n
$C[\,]$	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：依次求解问题

已知钢条价格

$p[\,]$	$p[1]$	$p[2]$	…	$p[n]$

初始化

	0	1	2	…	n
$C[\,]$	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：依次求解问题

已知钢条价格

$p[\,]$	$p[1]$	$p[2]$	…	$p[n]$

$C[1]$

max

$p[1]$

$$C[1] = \max\{p[1]\}$$

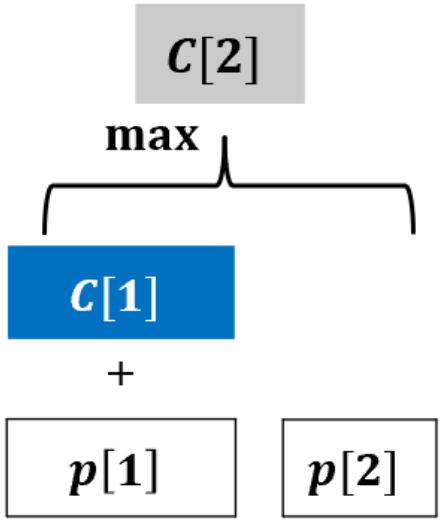
	0	1	2	…	n
$C[\,]$	0 →				

- 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：依次求解问题

已知钢条价格

$p[\,]$	$p[1]$	$p[2]$	…	$p[n]$



$$C[2] = \max\{C[1] + p[1], p[2]\}$$

	0	1	2	…	n
$C[\,]$	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：依次求解问题

已知钢条价格

$p[\,]$	$p[1]$	$p[2]$	…	$p[n]$

$C[j]$

max

$C[j-1]$	…	$C[1]$

\+ \+ \+

$p[1]$	…	$p[j-1]$	$p[j]$

$$C[j] = \max_{1 \le i \le j-1} \{p[i] + C[j-i], p[j]\}$$

	0	1	2	…	n
$C[\,]$	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：依次求解问题

已知钢条价格

$p[\,]$	$p[1]$	$p[2]$	…	$p[n]$

$C[n]$

max

$C[n-1]$	$C[n-2]$	…	$C[1]$
+	+	+	+
$p[1]$	$p[2]$	…	$p[n-1]$	$p[n]$

$$C[n] = \max_{1 \le i \le n-1} \{p[i] + C[n-i], p[n]\}$$

	0	1	2	…	n
$C[\,]$	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：依次求解问题

已知钢条价格

$p[\,]$	$p[1]$	$p[2]$	…	$p[n]$

$C[n]$

max

$C[n-1]$	$C[n-2]$	…	$C[1]$
+	+	+	+
$p[1]$	$p[2]$	…	$p[n-1]$	$p[n]$

$$C[n] = \max_{1 \le i \le n-1} \{p[i] + C[n-i], p[n]\}$$

	0	1	2	…	n
$C[\,]$	0 →				★

问题结构分析

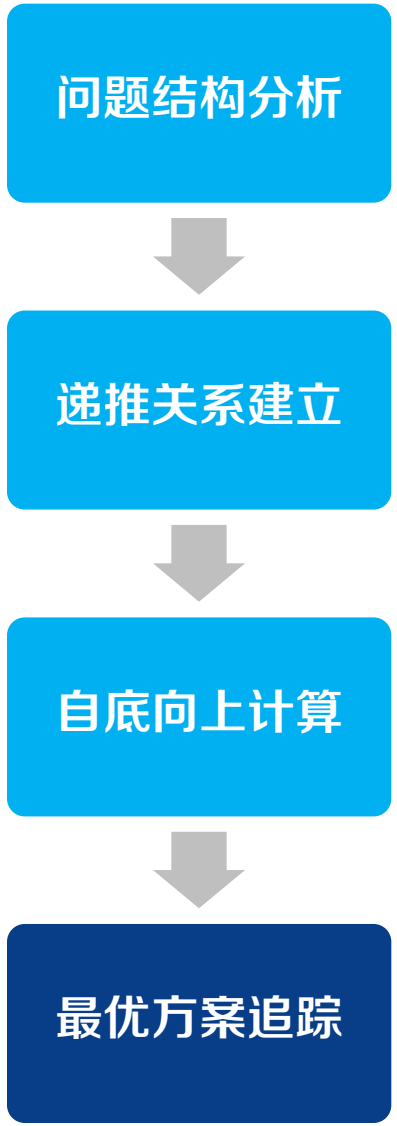
递推关系建立

自底向上计算

最优方案追踪

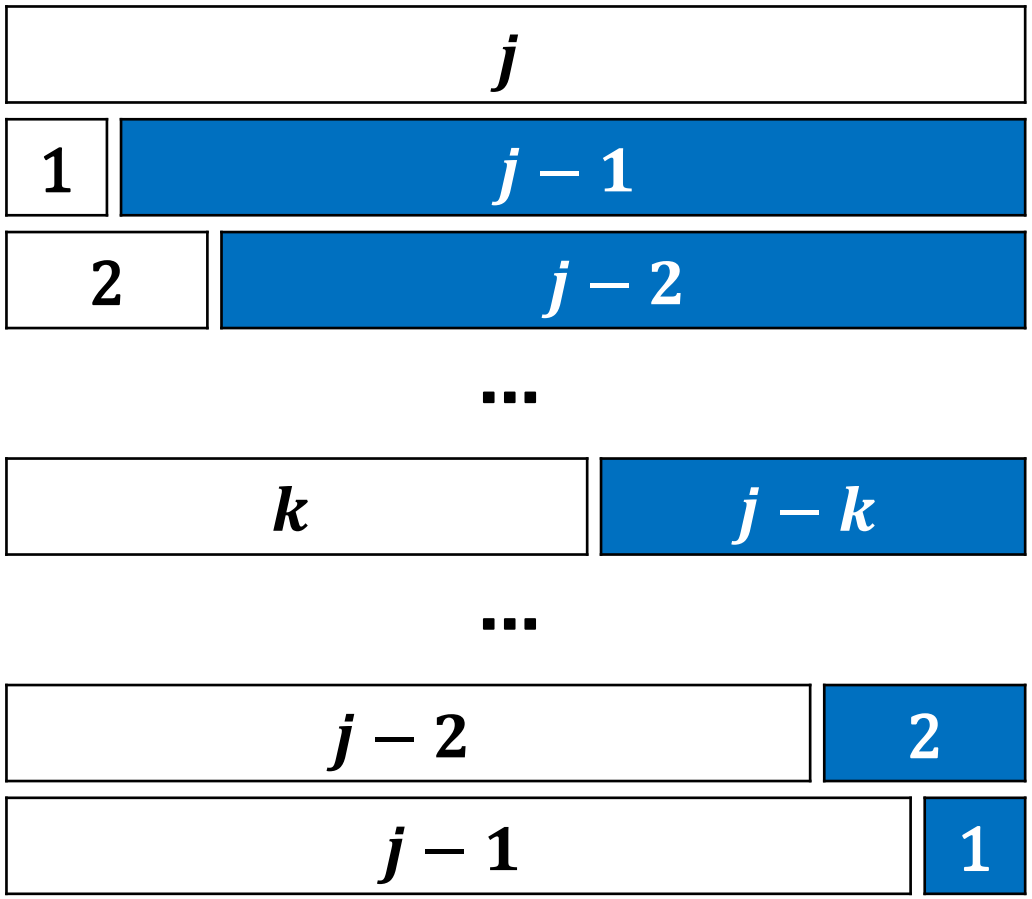
最优方案追踪：记录决策过程

- 构造追踪数组$rec[1..n]$



最优方案追踪：记录决策过程

- 构造追踪数组$rec[1..n]$



所有方案

问题结构分析

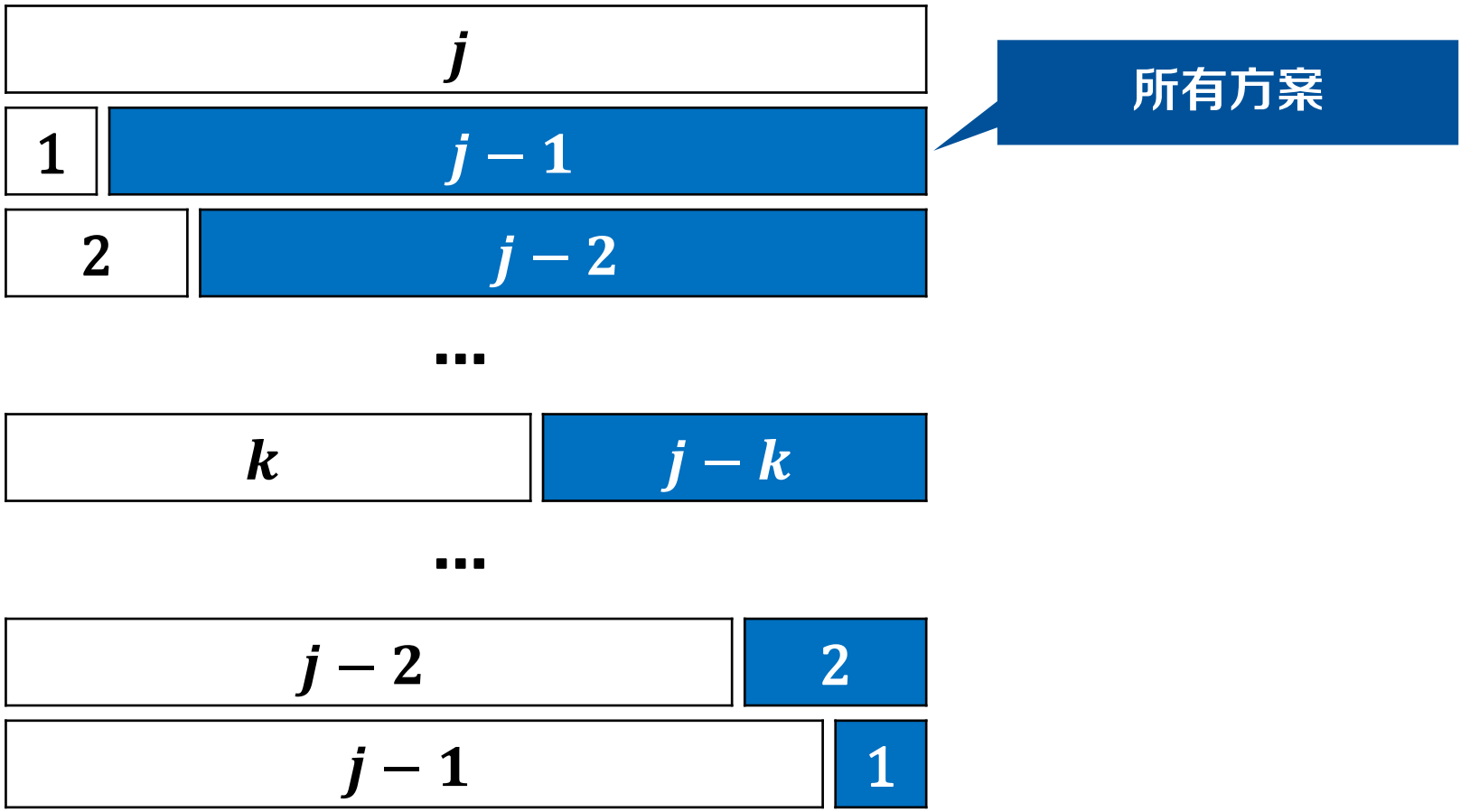
递推关系建立

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$rec[1..n]$
- $rec[j]$：记录长度为j钢条的最优切割方案



问题结构分析

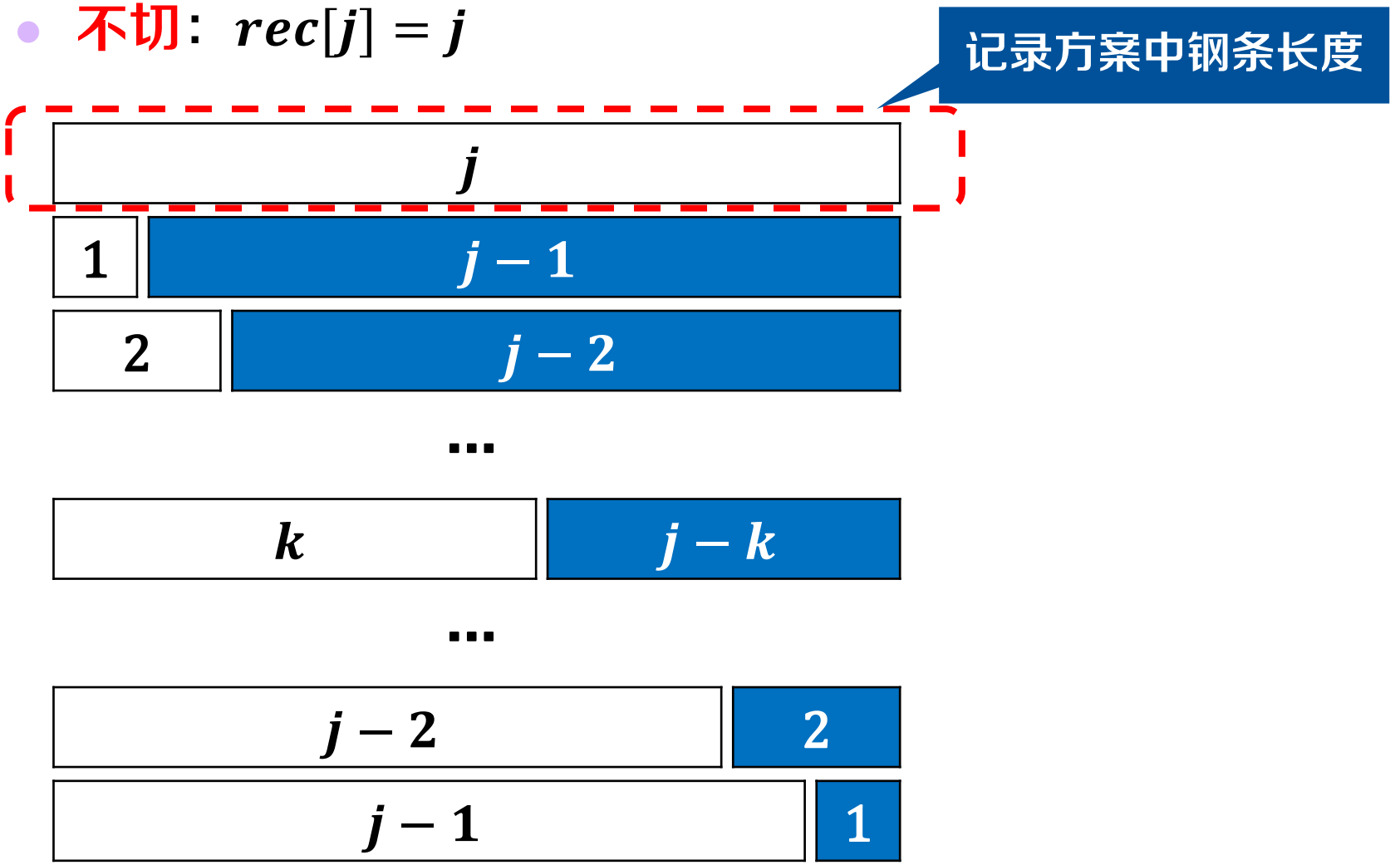
递推关系建立

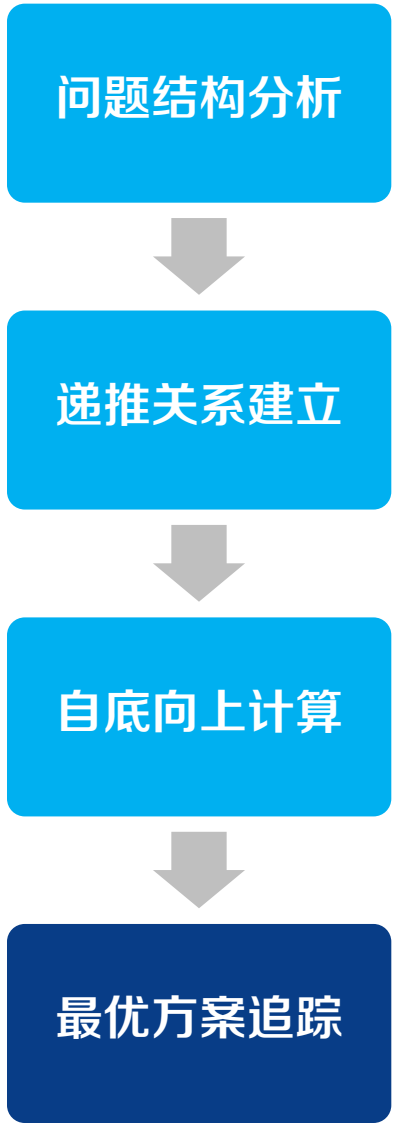
自底向上计算

最优方案追踪

最优方案追踪：记录决策过程

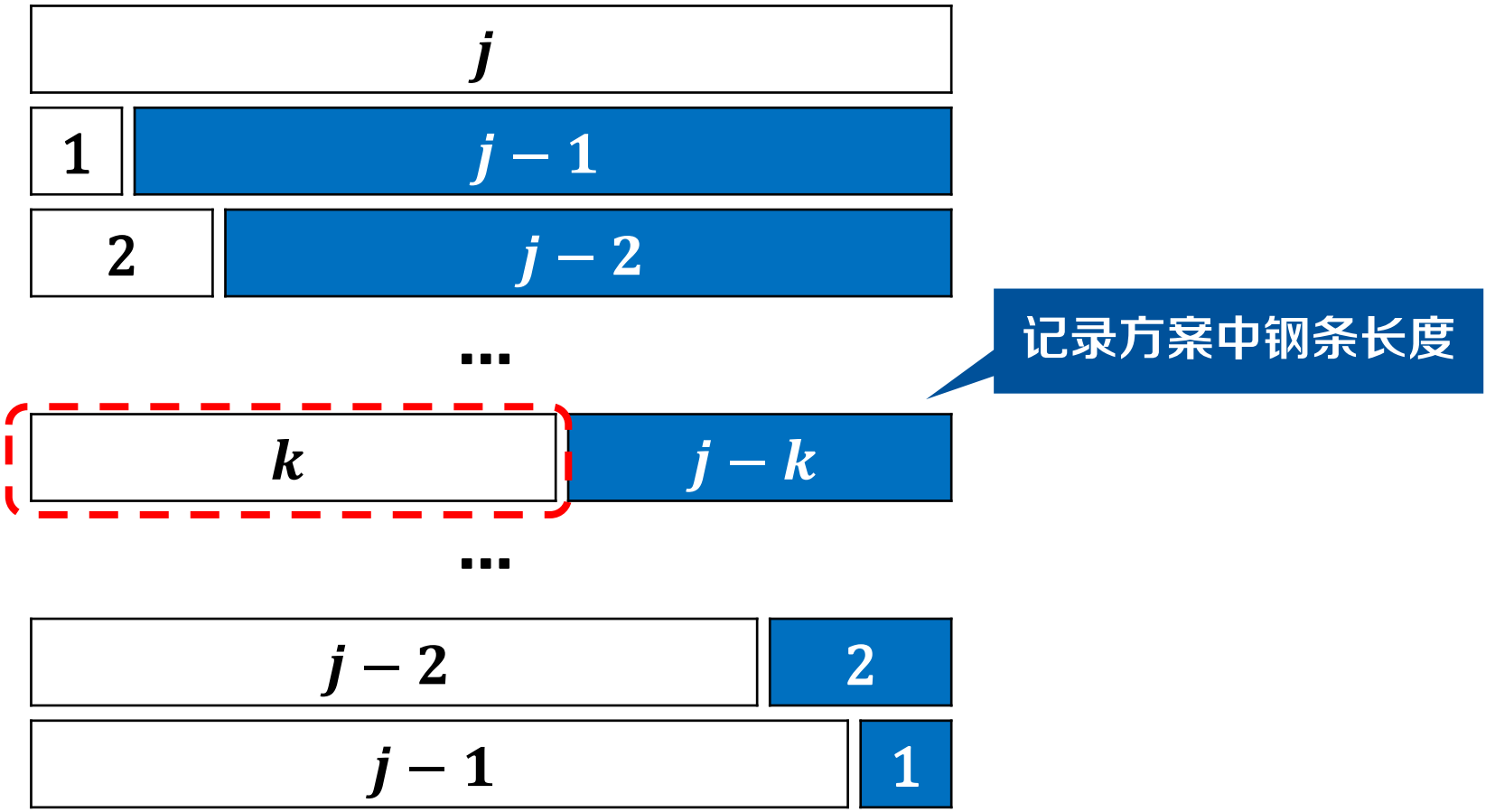
- 构造追踪数组$rec[1..n]$
- $rec[j]$：记录长度为j钢条的最优切割方案
  - 不切：$rec[j] = j$





最优方案追踪：记录决策过程

- 构造追踪数组$rec[1..n]$
- $rec[j]$：记录长度为j钢条的最优切割方案
  - 不切：$rec[j]=j$ 切割：$rec[j]=k$



问题结构分析

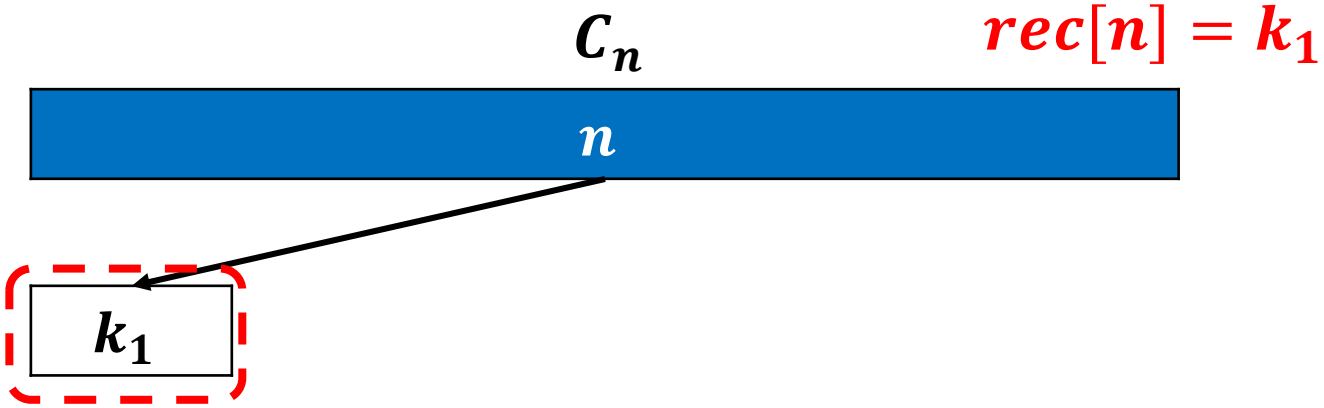
递推关系建立

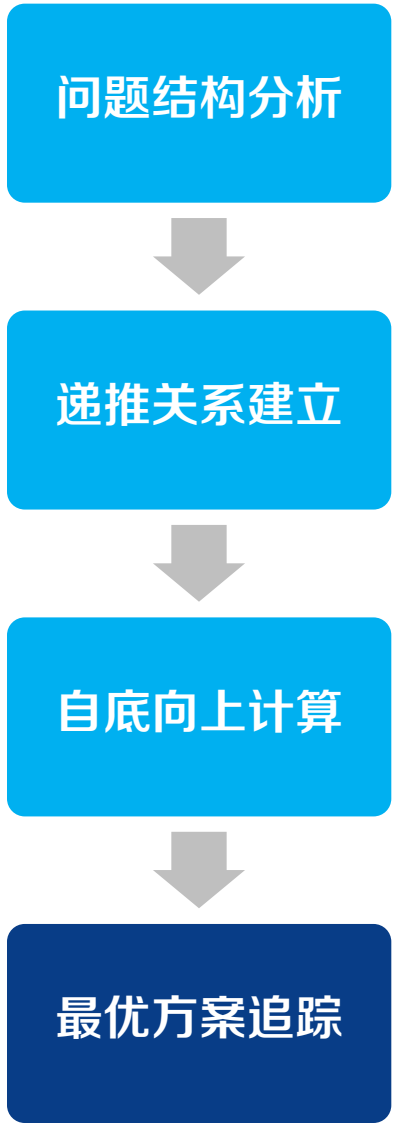
自底向上计算

最优方案追踪

最优方案追踪：输出最优方案

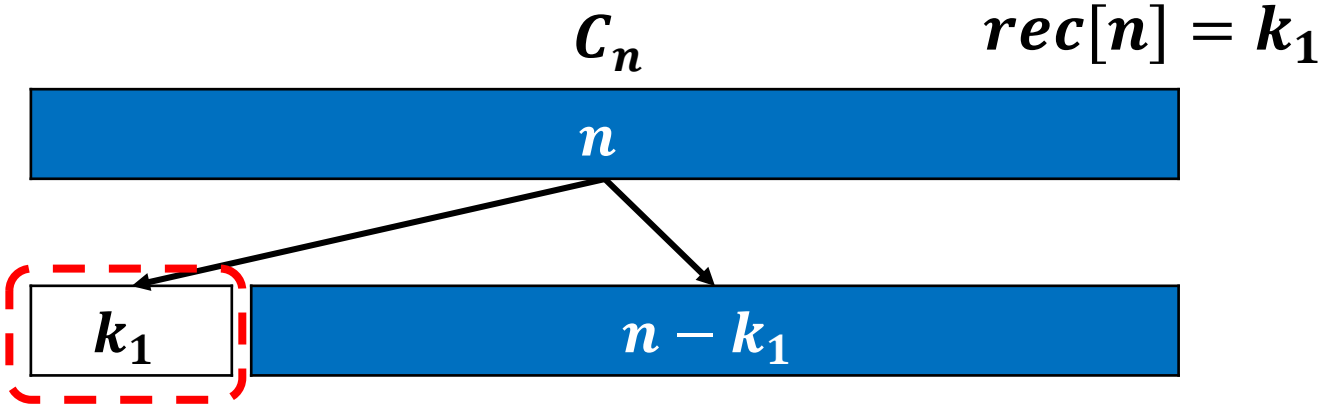
- 根据追踪数组，递归输出方案
  - 输出长度为k_1的钢条





最优方案追踪：输出最优方案

- 根据追踪数组，递归输出方案
  - 输出长度为k_1的钢条



问题结构分析

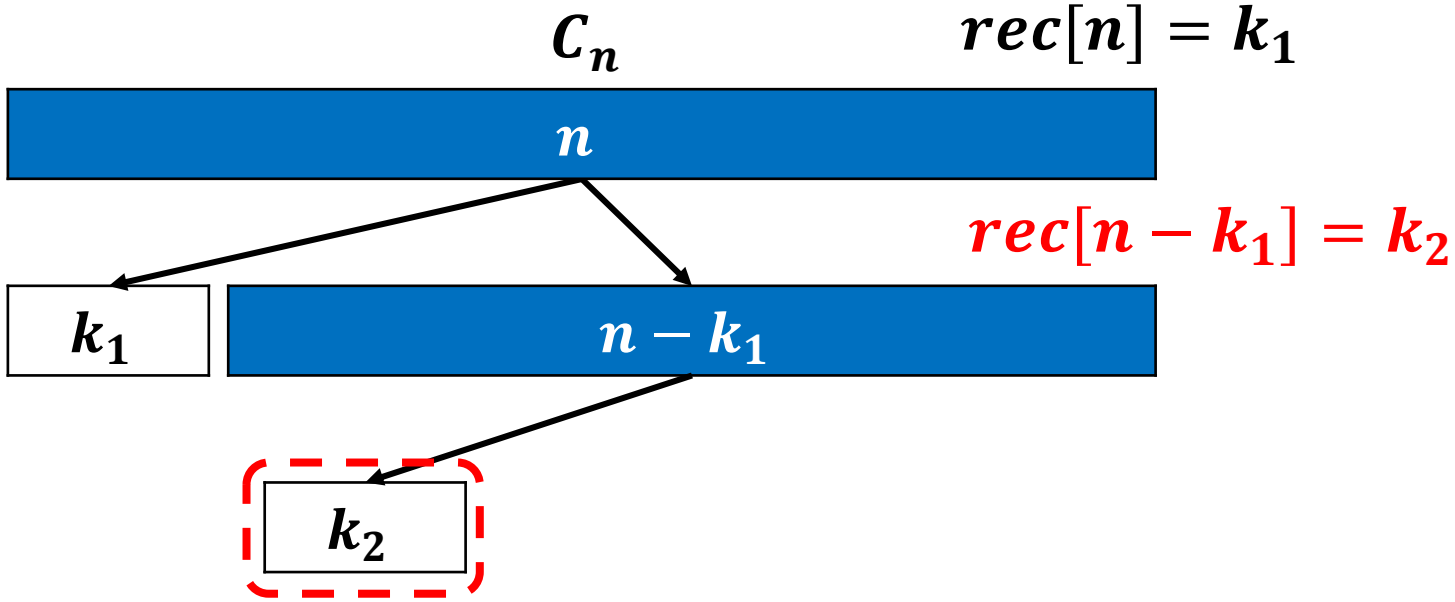
递推关系建立

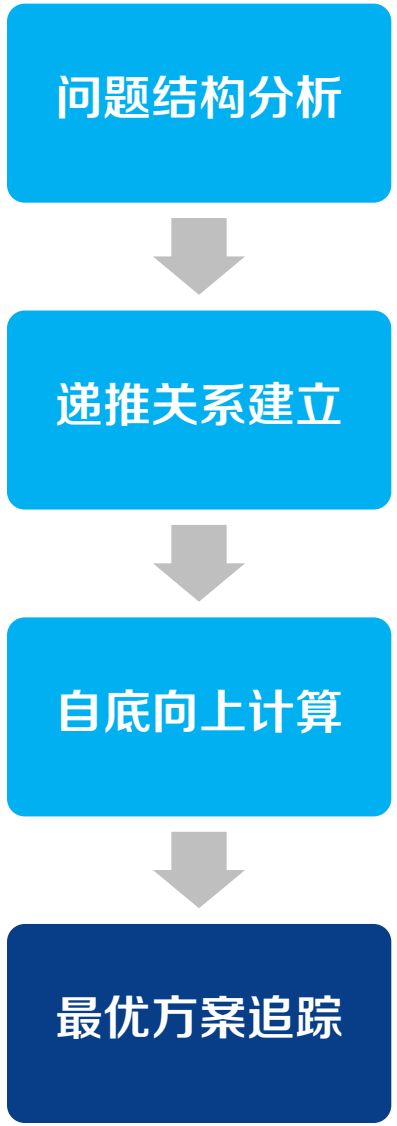
自底向上计算

最优方案追踪

最优方案追踪：输出最优方案

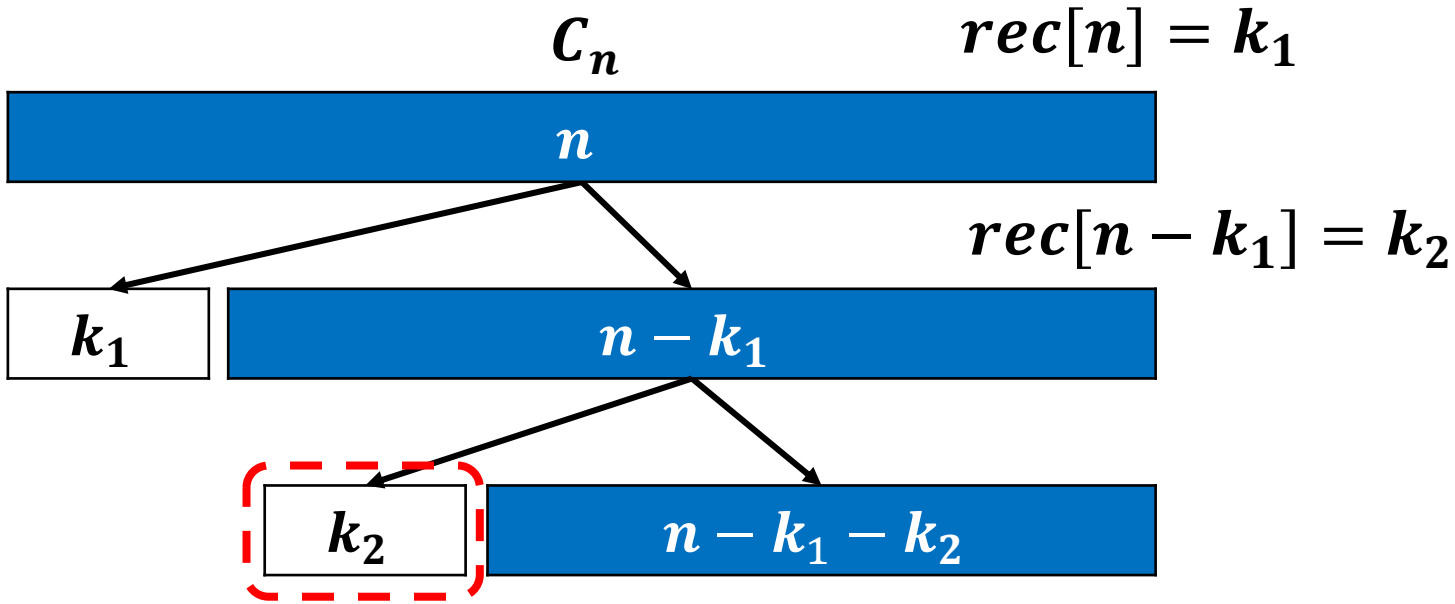
- 根据追踪数组，递归输出方案
  - 输出长度为k_2的钢条





最优方案追踪：输出最优方案

- 根据追踪数组，递归输出方案
  - 输出长度为k_2的钢条



问题结构分析

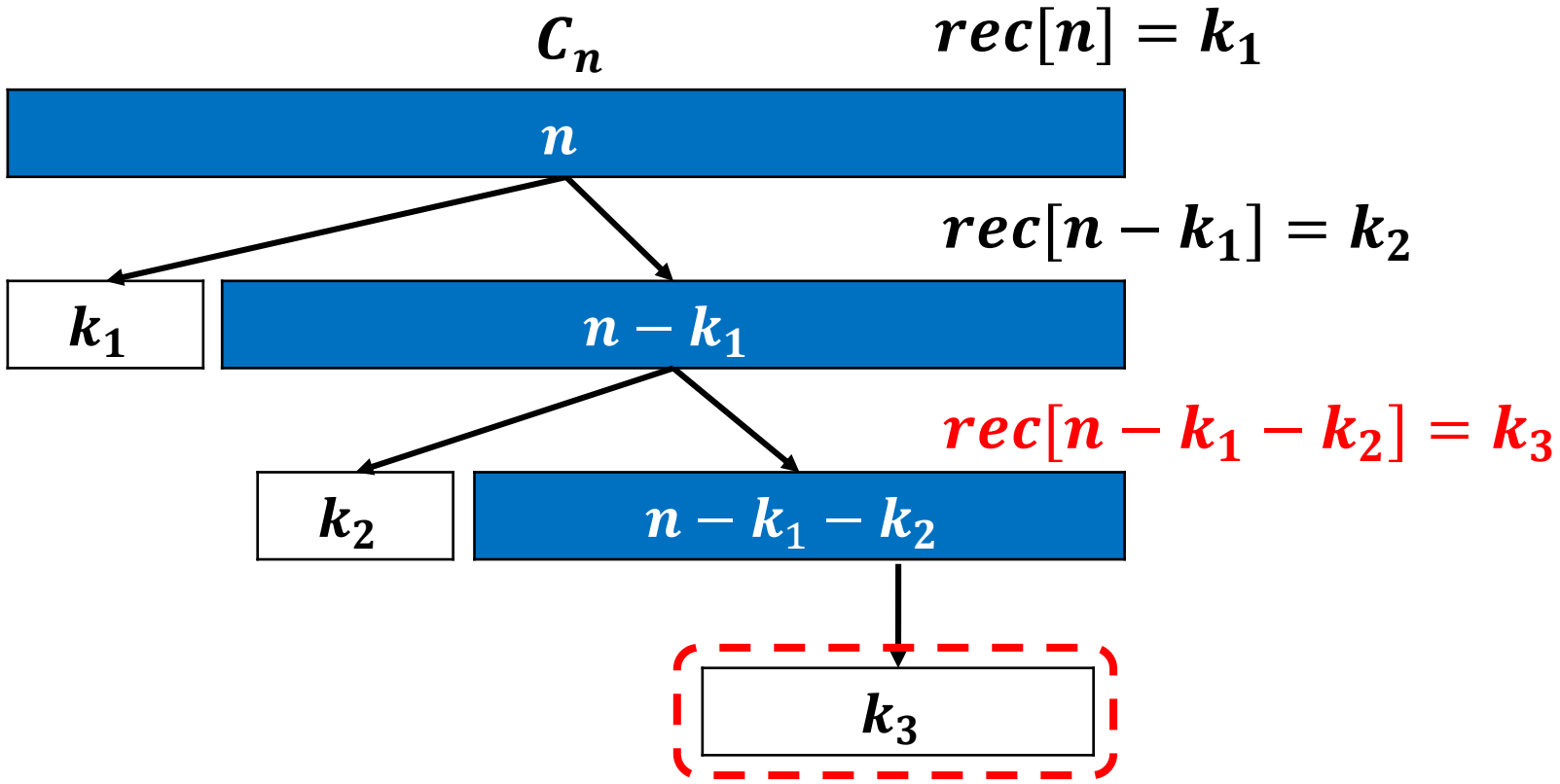
递推关系建立

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
  - 输出长度为k_3的钢条



问题结构分析

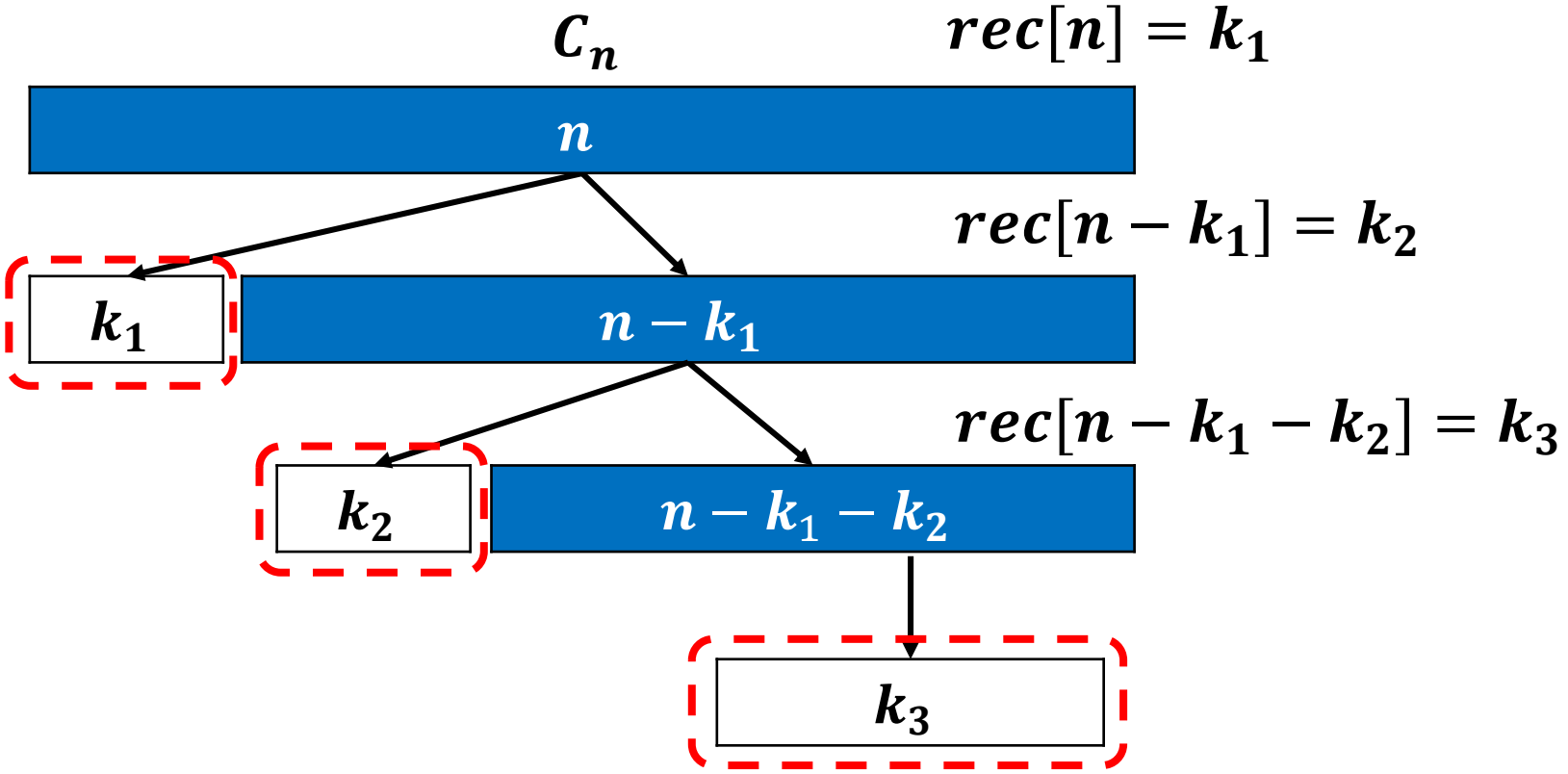
递推关系建立

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：输出的钢条总长度已达n



$$k_1 + k_2 + k_3 = n$$

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

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0										
rec	0	1	2	3	4	5	6	7	8	9	10

初始化

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 1$

i	1
	1

$p[j] = p[1]$

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0										
rec	0	1	2	3	4	5	6	7	8	9	10

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 1$

i	1
	1

$\max\{p[i] + C[j-i], p[j]\}$

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0										
rec	0	1	2	3	4	5	6	7	8	9	10

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 1$

i	1
	1

$\max\{p[i] + C[j-i], p[j]\}$

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1									
rec	0	1	2	3	4	5	6	7	8	9	10
		1									

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 2$

i	1
	2

$p[1] + C[1]$

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1									
rec	0	1	2	3	4	5	6	7	8	9	10
		1									

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 2$

i	1	2
	2	5

$p[j] = p[2]$

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1									
rec	0	1	2	3	4	5	6	7	8	9	10
		1									

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 2$

i	1	2
	2	5

$\max\{p[i] + C[j-i], p[j]\}$

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1									
rec	0	1	2	3	4	5	6	7	8	9	10
		1									

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 2$

i	1	2
	2	5

$\max\{p[i] + C[j-i], p[j]\}$

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5								
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2								

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 3$

i	1
	6

$p[1] + C[2]$

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5								
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2								

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 3$

i	1	2
	6	6

$p[2] + C[1]$

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5								
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2								

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 3$

i	1	2	3
	6	6	8

$p[j] = p[3]$

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5								
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2								

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 3$

i	1	2	3
	6	6	8

$\max\{p[i] + C[j-i], p[j]\}$

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5								
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2								

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 3$

i	1	2	3
	6	6	8

$\max\{p[i] + C[j-i], p[j]\}$

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8							
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3							

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 4$

i	1
	9

$p[1] + C[3]$

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8							
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3							

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 4$

i	1	2
	9	10

$p[2] + C[2]$

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8							
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3							

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 4$

i	1	2	3
	9	10	9

$p[3] + C[1]$

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8							
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3							

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 4$

i	1	2	3	4
	9	10	9	9

$p[j] = p[4]$

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8							
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3							

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 4$

$\max\{p[i] + C[j-i], p[j]\}$

i	1	2	3	4
	9	10	9	9

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8							
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3							

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 4$

$\max\{p[i] + C[j-i], p[j]\}$

i	1	2	3	4
	9	10	9	9

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8	10						
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3	2						

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 5$

i	1	2	3	4	5
	11	13	13	10	10

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8	10						
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3	2						

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 5$

$\max\{p[i] + C[j-i], p[j]\}$

i	1	2	3	4	5
	11	13	13	10	10

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8	10						
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3	2						

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 5$

$\max\{p[i] + C[j-i], p[j]\}$

i	1	2	3	4	5
	11	13	13	10	10

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8	10	13					
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3	2	2					

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 6$

i	1	2	3	4	5	6
	14	15	16	14	11	17

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8	10	13					
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3	2	2					

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 6$

i	1	2	3	4	5	6
	14	15	16	14	11	17

$\max\{p[i] + C[j-i], p[j]\}$

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8	10	13					
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3	2	2					

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 6$

i	1	2	3	4	5	6
	14	15	16	14	11	17

$\max\{p[i] + C[j-i], p[j]\}$

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8	10	13	17				
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3	2	2	6				

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 7$

i	1	2	3	4	5	6	7
	18	18	18	17	15	18	17

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8	10	13	17				
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3	2	2	6				

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 7$

$\max\{p[i] + C[j-i], p[j]\}$

i	1	2	3	4	5	6	7
	18	18	18	17	15	18	17

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8	10	13	17				
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3	2	2	6				

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 7$

$\max\{p[i] + C[j-i], p[j]\}$

i	1	2	3	4	5	6	7
	18	18	18	17	15	18	17

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8	10	13	17	18			
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3	2	2	6	1			

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 8$

i	1	2	3	4	5	6	7	8
	19	22	21	19	18	22	18	20

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8	10	13	17	18			
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3	2	2	6	1			

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 8$

$\max\{p[i] + C[j-i], p[j]\}$

i	1	2	3	4	5	6	7	8
	19	22	21	19	18	22	18	20

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8	10	13	17	18			
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3	2	2	6	1			

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 8$

$\max\{p[i] + C[j-i], p[j]\}$

i	1	2	3	4	5	6	7	8
	19	22	21	19	18	22	18	20

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8	10	13	17	18	22		
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3	2	2	6	1	2		

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 9$

i	1	2	3	4	5	6	7	8	9
	23	23	25	22	20	25	22	21	24

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8	10	13	17	18	22		
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3	2	2	6	1	2		

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 9$

$\max\{p[i] + C[j-i], p[j]\}$

i	1	2	3	4	5	6	7	8	9
	23	23	25	22	20	25	22	21	24

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8	10	13	17	18	22		
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3	2	2	6	1	2		

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 9$

$\max\{p[i] + C[j-i], p[j]\}$

i	1	2	3	4	5	6	7	8	9
	23	23	25	22	20	25	22	21	24

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8	10	13	17	18	22	25	
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3	2	2	6	1	2	3	

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 10$

i	1	2	3	4	5	6	7	8	9	10
	26	27	26	26	23	27	25	25	25	24

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8	10	13	17	18	22	25	
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3	2	2	6	1	2	3	

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 10$

$\max\{p[i] + C[j-i], p[j]\}$

i	1	2	3	4	5	6	7	8	9	10
	26	27	26	26	23	27	25	25	25	24

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8	10	13	17	18	22	25	
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3	2	2	6	1	2	3	

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 10$

$\max\{p[i] + C[j-i], p[j]\}$

i	1	2	3	4	5	6	7	8	9	10
	26	27	26	26	23	27	25	25	25	24

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8	10	13	17	18	22	25	27
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3	2	2	6	1	2	3	2

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 10$

i	1	2	3	4	5	6	7	8	9	10
	26	27	26	26	23	27	25	25	25	24

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8	10	13	17	18	22	25	27
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3	2	2	6	1	2	3	2

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 10$

i	1	2	3	4	5	6	7	8	9	10
	26	27	26	26	23	27	25	25	25	24

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8	10	13	17	18	22	25	27
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3	2	2	6	1	2	3	2

最大收益$= C[10] = 27$

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 10$

i	1	2	3	4	5	6	7	8	9	10
	26	27	26	26	23	27	25	25	25	24

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8	10	13	17	18	22	25	27
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3	2	2	6	1	2	3	2

最大收益$= C[10] = 27$

切割方案$= \{2,$

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 10$

i	1	2	3	4	5	6	7	8	9	10
	26	27	26	26	23	27	25	25	25	24

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8	10	13	17	18	22	25	27
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3	2	2	6	1	2	3	2

最大收益$= C[10] = 27$

切割方案$= \{2, 2$

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 10$

i	1	2	3	4	5	6	7	8	9	10
	26	27	26	26	23	27	25	25	25	24

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8	10	13	17	18	22	25	27
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3	2	2	6	1	2	3	2

最大收益$= C[10] = 27$

切割方案$= \{2, 2, 6$

算法实例

$n = 10$

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	24

$j = 10$

i	1	2	3	4	5	6	7	8	9	10
	26	27	26	26	23	27	25	25	25	24

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	0	1	5	8	10	13	17	18	22	25	27
rec	0	1	2	3	4	5	6	7	8	9	10
		1	2	3	2	2	6	1	2	3	2

最大收益$= C[10] = 27$

切割方案$= \{2, 2, 6\}$

伪代码

- **RodCutting(p, n)**

```
输入: 钢条价格表p[1..n],钢条长度n
输出: 最大收益C[n],钢条切割方案
//初始化
新建一维数组C[0..n], rec[0..n]
C[0] ← 0
//动态规划
for j ← 1 to n do
    q ← p[j]
    rec[j] ← j
    for i ← 1 to j − 1 do
        if q < p[i] + C[j − i] then
            q ← p[i] + C[j − i]
            rec[j] ← i
        end
    end
    C[j] ← q
end
```

初始化

伪代码

- **RodCutting(p, n)**

```
输入: 钢条价格表p[1..n],钢条长度n
输出: 最大收益C[n],钢条切割方案
//初始化
新建一维数组C[0..n], rec[0..n]
C[0] ← 0
//动态规划
for j ← 1 to n do
    q ← p[j]
    rec[j] ← j
    for i ← 1 to j − 1 do
        if q < p[i] + C[j − i] then
            q ← p[i] + C[j − i]
            rec[j] ← i
        end
    end
    C[j] ← q
end
```

依次计算子问题

伪代码

- **RodCutting(p, n)**

```
输入: 钢条价格表p[1..n],钢条长度n
输出: 最大收益C[n],钢条切割方案
//初始化
新建一维数组C[0..n], rec[0..n]
C[0] ← 0
//动态规划
for j ← 1 to n do
    q ← p[j]
    rec[j] ← j
    for i ← 1 to j − 1 do
        if q < p[i] + C[j − i] then
            q ← p[i] + C[j − i]
            rec[j] ← i
        end
    end
    C[j] ← q
end
```

不切割钢条

伪代码

- **RodCutting(p, n)**

```
输入: 钢条价格表p[1..n],钢条长度n
输出: 最大收益C[n],钢条切割方案
//初始化
新建一维数组C[0..n], rec[0..n]
C[0] ← 0
//动态规划
for j ← 1 to n do
    q ← p[j]
    rec[j] ← j
    for i ← 1 to j − 1 do
        if q < p[i] + C[j − i] then
            q ← p[i] + C[j − i]
            rec[j] ← i
        end
    end
    C[j] ← q
end
```

枚举切割长度

伪代码

- **RodCutting(p, n)**

输入: 钢条价格表$p[1..n]$,钢条长度n
输出: 最大收益$C[n]$,钢条切割方案
//初始化
新建一维数组$C[0..n], rec[0..n]$
$C[0] \leftarrow 0$
//动态规划
for $j \leftarrow 1$ *to* n **do**
  $q \leftarrow p[j]$
  $rec[j] \leftarrow j$
  for $i \leftarrow 1$ *to* $j-1$ **do**
    if $q < p[i] + C[j-i]$ **then**
      $q \leftarrow p[i] + C[j-i]$
      $rec[j] \leftarrow i$
    end
  end
  $C[j] \leftarrow q$
end

记录价格和决策

伪代码

- **RodCutting(p, n)**

输入: 钢条价格表$p[1..n]$,钢条长度n
输出: 最大收益$C[n]$,钢条切割方案
//初始化
新建一维数组$C[0..n], rec[0..n]$
$C[0] \leftarrow 0$
//动态规划
for $j \leftarrow 1$ *to* n **do**
  $q \leftarrow p[j]$
  $rec[j] \leftarrow j$
  for $i \leftarrow 1$ *to* $j-1$ **do**
    if $q < p[i] + C[j-i]$ **then**
      $q \leftarrow p[i] + C[j-i]$
      $rec[j] \leftarrow i$
    end
  end
  $C[j] \leftarrow q$
end

保存子问题的解

伪代码

- **RodCutting(p, n)**

```
//输出最优方案
while n > 0 do
    print rec[n]
    n ← n − rec[n]
end
```

追踪切割方案

时间复杂度分析

- **RodCutting(p, n)**

```
输入: 钢条价格表p[1..n],钢条长度n
输出: 最大收益C[n],钢条切割方案
//初始化
新建一维数组C[0..n], rec[0..n]
C[0] ← 0
//动态规划
for j ← 1 to n do
    q ← p[j]
    rec[j] ← j
    for i ← 1 to j − 1 do
        if q < p[i] + C[j − i] then
            q ← p[i] + C[j − i]
            rec[j] ← i
        end
    end
    C[j] ← q
end
```

时间复杂度：$O(n^2)$

谢谢

BDA
Beihang University