Design and Analysis of Algorithms Part II: Dynamic Programming Lecture 10: 0-1 Knapsack

童咏昕

北京航空航天大学 计算机学院

问题背景



- 超市赢家
 - 超市允许顾客使用一个体积大小为13的背包,选择一件或多件商品带走

问题背景



• 超市赢家

商品	价格	体积
谭 啤酒	24	10
汽水	2	3
⊕ 饼干	9	4
1 面包	10	5
二 牛奶	9	4



问题背景



- 超市赢家
 - 超市允许顾客使用一个体积大小为13的背包,选择一件或多件商品带走

商品	价格	体积
谭 啤酒	24	10
汽水	2	3
⊕ 饼干	9	4
面包	10	5
牛奶	9	4



问题: 如何带走总价最多的商品?

问题定义



• 形式化定义

0-1背包问题

0-1 Knapsack Problem

输入

• n个商品组成集合o,每个商品有两个属性 v_i 和 p_i ,分别表示体积和价格

问题定义



• 形式化定义

0-1背包问题

0-1 Knapsack Problem

输入

- n个商品组成集合o,每个商品有两个属性 v_i 和 p_i ,分别表示体积和价格
- · 背包容量为C

问题定义



• 形式化定义

0-1背包问题

0-1 Knapsack Problem

输入

- n个商品组成集合o,每个商品有两个属性 v_i 和 p_i ,分别表示体积和价格
- 背包容量为C

输出

• 求解一个商品子集 $S \subseteq O$,令

$$\max \sum_{i \in S} p_i$$

$$s.\ t.\sum_{i\in S}v_i\leq C$$



• 形式化定义

0-1背包问题

0-1 Knapsack Problem

输入

- n个商品组成集合o,每个商品有两个属性 v_i 和 p_i ,分别表示体积和价格
- · 背包容量为C

输出

• 求解一个商品子集 $S \subseteq O$,令

$$\max \sum_{i \in S} p_i$$
 优化目标

$$s. t. \sum_{i \in S} v_i \leq C$$



• 形式化定义

0-1背包问题

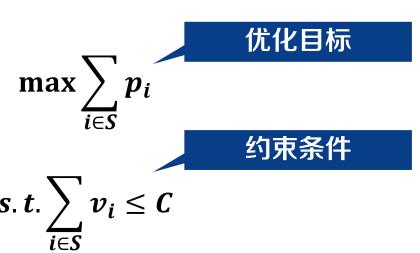
0-1 Knapsack Problem

输入

- n个商品组成集合o,每个商品有两个属性 v_i 和 p_i ,分别表示体积和价格
- 背包容量为C

输出

• 求解一个商品子集 $S \subseteq O$,令





• 超市赢家

商品	价格	体积
谭 啤酒	24	10
汽水	2	3
⊕ 饼干	9	4
面包	10	5
二 牛奶	9	4

	商品列表	总价格	总体积	说明
方案1		34	15	



• 超市赢家

商品	价格	体积
谭 啤酒	24	10
汽水	2	3
⊕ 饼干	9	4
面包	10	5
二 牛奶	9	4

	商品列表	总价格	总体积	说明
方案1		34	15	



• 超市赢家

商品	价格	体积
谭 啤酒	24	10
汽水	2	3
↔ 饼干	9	4
面包	10	5
二 牛奶	9	4

	商品列表	总价格	总体积	说明
方案1		34	15	错误方案



- 超市赢家
 - 超市允许顾客使用一个体积大小为13的背包,选择一件或多件商品带走

商品	价格	体积
谭 啤酒	24	10
汽水	2	3
↔ 饼干	9	4
面包	10	5
二 牛奶	9	4

	商品列表	总价格	总体积	说明
方案1		34	15	错误方案
方案2		26	13	



- 超市赢家
 - 超市允许顾客使用一个体积大小为13的背包,选择一件或多件商品带走

商品	价格	体积
谭 啤酒	24	10
汽水	2	3
↔ 饼干	9	4
面包	10	5
二 牛奶	9	4

	商品列表	总价格	总体积	说明
方案1		34	15	错误方案
方案2		26	13	可行方案



- 超市赢家
 - 超市允许顾客使用一个体积大小为13的背包,选择一件或多件商品带走

商品	价格	体积
谭 啤酒	24	10
汽水	2	3
∰ 饼干	9	4
面包	10	5
一片	9	4

	商品列表	总价格	总体积	说明
方案1		34	15	错误方案
方案2		26	13	可行方案
方案3		28	13	



• 超市赢家

商品	价格	体积
谭 啤酒	24	10
汽水	2	3
⊕ 饼干	9	4
一面包	10	5
二 牛奶	9	4

	商品列表	总价格	总体积	说明
方案1		34	15	错误方案
方案2		26	13	可行方案
方案3		28	13	最优方案



- 超市赢家
 - 超市允许顾客使用一个体积大小为13的背包,选择一件或多件商品带走

商品	价格	体积
谭 啤酒	24	10
汽水	2	3
∰ 饼干	9	4
面包	10	5
一 牛奶	9	4

	商品列表	总价格	总体积	说明
方案1		34	15	错误方案
方案2		26	13	可行方案
方案3		28	13	最优方案

问题: 如何求解该问题?



• 核心思想: 将商品排序,依次挑选

策略1: 按商品价格由高到低排序,优先挑选价格高的商品

商品		价格	体积
谭 啤油	酉	24	10
1 面包	<u> </u>	10	5
	F	9	4
4 4 1	乃	9	4
汽汽	J	2	3

商品列表	总体积	总价格	说明





• 核心思想: 将商品排序,依次挑选

策略1: 按商品价格由高到低排序,优先挑选价格高的商品

商品	价格	体积
啤酒	24	10
面包	10	5
饼干	9	4
牛奶	9	4
汽水	2	3

	商品列表	总体积	总价格	说明
策略1		10	24	





• 核心思想: 将商品排序,依次挑选

策略1: 按商品价格由高到低排序,优先挑选价格高的商品

	商品	价格	体积
	啤酒	24	10
	面包	10	5
	饼干	9	4
	牛奶	9	4
F	汽水	2	3

	商品列表	总体积	总价格	说明
策略1		10	24	





• 核心思想:将商品排序,依次挑选

	商品	价格	体积		商品列表	总体积	总价格	说明
	啤酒	24	10	策略1		10	24	
	面包	10	5					
	饼干	9	4					
	牛奶	9	4			+ IO. 2		
	汽水	2	3			本积: 3 介值: 24		



• 核心思想:将商品排序,依次挑选

	商品	价格	体积		商品列表	总体积	总价格	说明
ĺ	神 啤酒	24	10	策略1		10	24	
) 面包	10	5	•				
6	第 饼干	9	4	•				
	牛奶	9	4			# ID. 2		
8	汽水	2	3			本积: 3 介值: 24		



• 核心思想:将商品排序,依次挑选

	商品	价格	体积		商品列表	总体积	总价格	说明
	弹 啤酒	24	10	策略1		13	26	
	面包	10	5	•				
8	讲干	9	4	•				
	生 牛奶	9	4			# ∓Π• Λ		
5	汽水	2	3			体积: 0 介值: 26		



• 核心思想:将商品排序,依次挑选

	商品	价格	体积		商品列表	总体积	总价格	说明
	啤酒	24	10	策略1		13	26	非最优解
	面包	10	5					
3	饼干	9	4					
	牛奶	9	4		1			
	汽水	2	3) 💺				



• 核心思想: 将商品排序,依次挑选

策略2: 按商品体积由小到大排序,优先挑选体积小的商品

商品	价格	体积
汽水	2	3
饼干	9	4
牛奶	9	4
面包	10	5
啤酒	24	10

	商品列表	总体积	总价格	说明
策略1		13	26	非最优解
策略2				





• 核心思想: 将商品排序,依次挑选

策略2: 按商品体积由小到大排序,优先挑选体积小的商品

		商品	价格	体积
/ 		汽水	2	3
	③	饼干	9	4
		牛奶	9	4
		面包	10	5
		啤酒	24	10

	商品列表	总体积	总价格	说明
策略1		13	26	非最优解
策略2		3	2	



剩余体积: 10 商品价值: 2



• 核心思想: 将商品排序,依次挑选

策略2: 按商品体积由小到大排序,优先挑选体积小的商品

	商品	价格	体积
	汽水	2	3
③	饼干	9	4
	牛奶	9	4
	面包	10	5
	啤酒	24	10

	商品列表	总体积	总价格	说明
策略1		13	26	非最优解
策略2		7	11	





• 核心思想: 将商品排序,依次挑选

策略2: 按商品体积由小到大排序,优先挑选体积小的商品

	商品	价格	体积
	汽水	2	3
③	饼干	9	4
	牛奶	9	4
	面包	10	5
	啤酒	24	10

	商品列表	总体积	总价格	说明
策略1		13	26	非最优解
策略2		11	20	





• 核心思想: 将商品排序,依次挑选

策略2: 按商品体积由小到大排序,优先挑选体积小的商品

价格	体积	
2	3	į
9	4	
9	4	
10	5	X
24	10	
	2 9 9 10	2 3 9 4 9 4 10 5

	商品列表	总体积	总价格	说明
策略1		13	26	非最优解
策略2		11	20	





• 核心思想: 将商品排序,依次挑选

策略2: 按商品体积由小到大排序,优先挑选体积小的商品

商品	价格	体积		商品列表	总体积	总价格	说明
汽水	2	3	策略1		13	26	非最优解
❸ 饼干	9	4	策略2		11	20	
<u></u> 牛奶	9	4					
1 面包	10	5	, Trans	₽. / =1/2	>/# I□ .		
谭 啤酒	24	10			条体积: 2 3价值: 20		



• 核心思想:将商品排序,依次挑选

策略2: 按商品体积由小到大排序,优先挑选体积小的商品

商品	价格	体积
汽水	2	3
∰ 饼干	9	4
造 牛奶	9	4
1 面包	10	5
谭 啤酒	24	10

	商品列表	总体积	总价格	说明
策略1		13	26	非最优解
策略2		11	20	非最优解





• 核心思想: 将商品排序,依次挑选

策略3: 按商品价值与体积的比由高到低排序,优先挑选比值高的商品

商品	价格	体积	比值
谭 啤酒	24	10	2.4
∰饼干	9	4	2.25
二十奶	9	4	2.25
一面包	10	5	2
汽汽水	2	3	0.67

	商品列表	总体积	总价格	说明
策略1		13	26	非最优解
策略2		11	20	非最优解
策略3				





• 核心思想: 将商品排序,依次挑选

策略3: 按商品价值与体积的比由高到低排序,优先挑选比值高的商品

商品	价格	体积	比值
啤酒	24	10	2.4
饼干	9	4	2.25
牛奶	9	4	2.25
面包	10	5	2
汽水	2	3	0.67

	商品列表	总体积	总价格	说明
策略1		13	26	非最优解
策略2		11	20	非最优解
策略3		10	24	





• 核心思想: 将商品排序,依次挑选

策略3: 按商品价值与体积的比由高到低排序,优先挑选比值高的商品

商品	价格	体积	比值
一 啤酒	24	10	2.4
∰干	9	4	2.25
二 牛奶	9	4	2.25
面包	10	5	2
汽水	2	3	0.67

	商品列表	总体积	总价格	说明
策略1		13	26	非最优解
策略2		11	20	非最优解
策略3		10	24	





• 核心思想: 将商品排序,依次挑选

策略3: 按商品价值与体积的比由高到低排序,优先挑选比值高的商品

	商品	价格	体积	比值
111	땹 啤酒	24	10	2.4
	፡፡ 饼干	9	4	2.25
	二 牛奶	9	4	2.25
	一面包	10	5	2
	汽水	2	3	0.67

	商品列表	总体积	总价格	说明
策略1		13	26	非最优解
策略2		11	20	非最优解
策略3		10	24	





• 核心思想:将商品排序,依次挑选

策略3: 按商品价值与体积的比由高到低排序,优先挑选比值高的商品

商品	价格	体积	比值
땹 啤酒	24	10	2.4
∰饼干	9	4	2.25
二牛奶	9	4	2.25
面包	10	5	2
汽水	2	3	0.67

	商品列表	总体积	总价格	说明
策略1		13	26	非最优解
策略2		11	20	非最优解
策略3		10	24	



直观策略



• 核心思想: 将商品排序,依次挑选

策略3: 按商品价值与体积的比由高到低排序,优先挑选比值高的商品

	商品	价格	体积	比值
	啤酒	24	10	2.4
3	饼干	9	4	2.25
	牛奶	9	4	2.25
	面包	10	5	2
	汽水	2	3	0.67

	商品列表	总体积	总价格	说明
策略1		13	26	非最优解
策略2		11	20	非最优解
策略3		13	26	



剩余体积: 0 商品价值: 26

直观策略



• 核心思想: 将商品排序,依次挑选

● 策略3: 按商品价值与体积的比由高到低排序,优先挑选比值高的商品

	商品	价格	体积	比值
111	啤酒	24	10	2.4
•	҈ 饼干	9	4	2.25
	造 牛奶	9	4	2.25
	一面包	10	5	2
/ 	∄ 汽水	2	3	0.67

	商品列表	总体积	总价格	说明
策略1		13	26	非最优解
策略2		11	20	非最优解
策略3		13	26	非最优解

问题: 如何保证获得最优解?





商品	价格	体积
🎬 啤酒	24	10
∄ 汽水	2	3
ῷ 饼干	9	4
1 面包	10	5
造 牛奶	9	4









商品	价格	体积
🎬 啤酒	24	10
汽水	2	3
ῷ 饼干	9	4
1 面包	10	5
上 牛奶	9	4







商品	价格	体积
🎬 啤酒	24	10
汽水	2	3
ῷ 饼干	9	4
1 面包	10	5
造 牛奶	9	4







	• •	
商品	价格	体积
问 啤酒	24	10
汽水	2	3
ῷ 饼干	9	4
🔐 面包	10	5
造 牛奶	9	4



• 枚举所有商品组合



背包体积13

商品	价格	体积
彈 啤酒	24	10
 汽水 饼干	2	3
ῷ 饼干	9	4
一面包	10	5
4	9	4



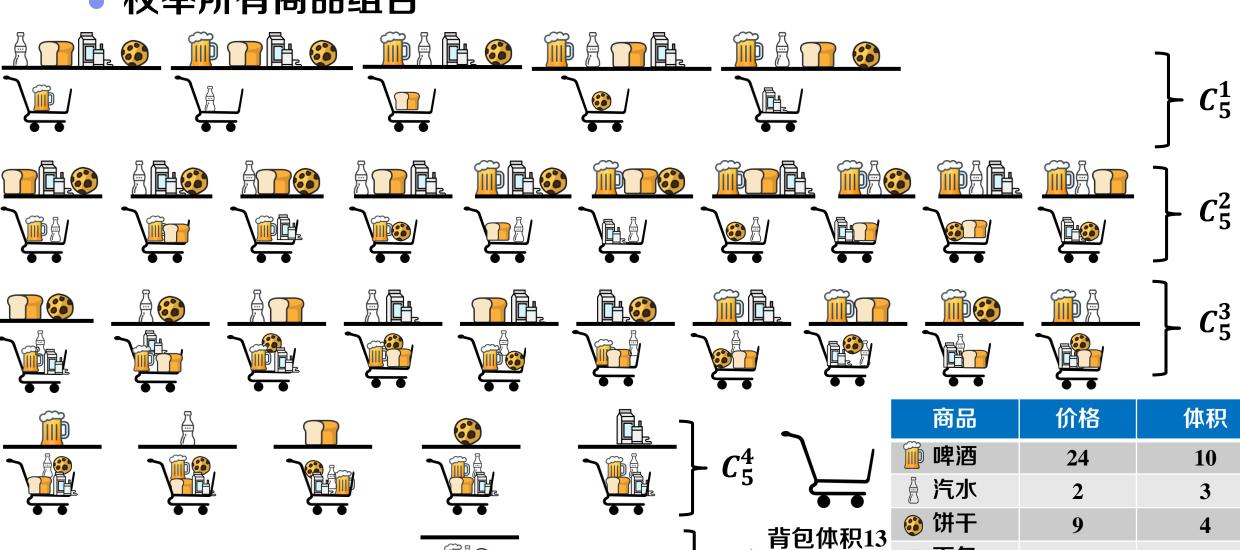
5

4

面包

10

9





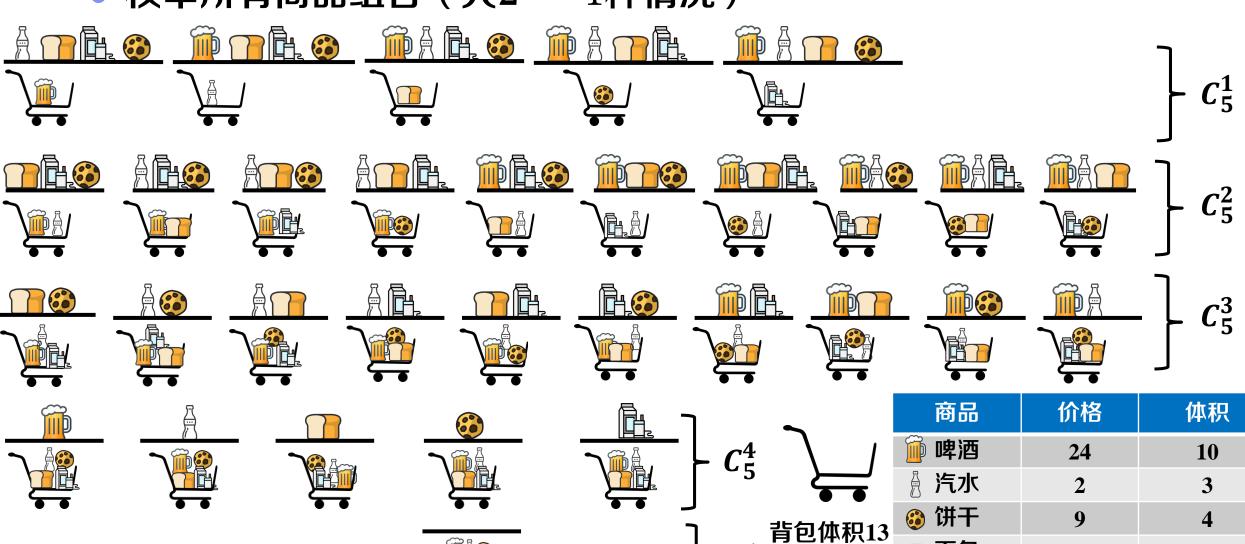
面包

10

9

4

• 枚举所有商品组合(共 2^n-1 种情况)

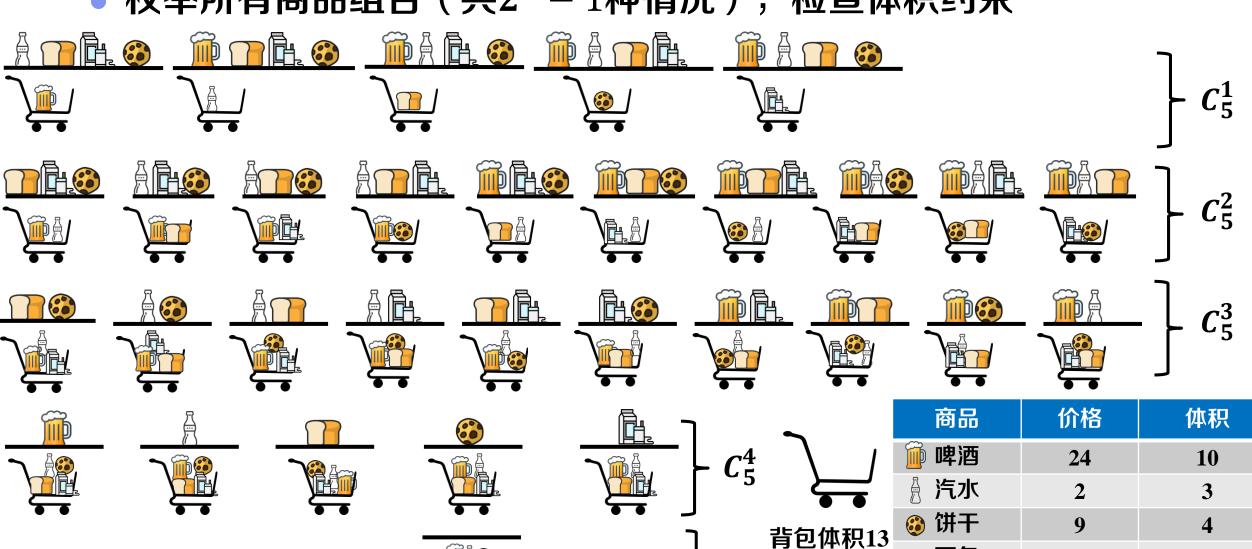




5

4

• 枚举所有商品组合(共 2^n-1 种情况),检查体积约束



面包

10

9



• 枚举所有商品组合(共 2^n-1 种情况),检查体积约束



25	22	22
	26	





• 枚举所有商品组合(共 2^n-1 种情况),检查体积约束



9

	7			
16	23	25	22	
			26	



商品	价格	体积
🏺 啤酒	24	10
△ 汽水	2	3
ῷ 饼干	9	4
1 面包	10	5
造 牛奶	9	4



• 枚举所有商品组合(共 2^n-1 种情况),检查体积约束











商品	价格	体积
🎬 啤酒	24	10
点 汽水	2	3
ῷ 饼干	9	4
一面包	10	5
二 牛奶	9	4



4

枚举所有商品组合(共 2^n-1 种情况), 检查体积约束



_	冏而	加恰	14
\ /	🎬 啤酒	24	-
—	∄ 汽水	2	
背包体积13	ῷ 饼干	9	
	1 面包	10	



● 递归函数: KnapsackSR(h, i, c)

● 在第*h*个到第*i*个商品中,容量为*c*时最优解



序号	商品		价格	体积
1		饼干	9	4
2		面包	10	5
3		牛奶	9	4
4		汽水	2	3
5		啤酒	24	10



递归函数: KnapsackSR(h, i, c)

● 在第h个到第i个商品中,容量为c时最优解

选择啤酒: KnapsackSR(1, 4, 3) + 24

序号	商品		价格	体积
1		饼干	9	4
2		面包	10	5
3		牛奶	9	4
4		汽水	2	3_
5		啤酒	24	10



」在第1到5个商品中选择 ■ 剩余体积:13

选择啤酒

KnapsackSR(1, 4, 3) + 24





● 递归函数: KnapsackSR(h, i, c)

在第h个到第i个商品中,容量为c时最优解

选择啤酒

选择啤酒: KnapsackSR(1, 4, 3) + 24

不选啤酒: KnapsackSR(1, 4, 13)

序号	商品		价格	体积
1		饼干	9	4
2		面包	10	5
3		牛奶	9	4
4		汽水	2	3
5		啤酒	24	10



KnapsackSR(1, 5, 13)



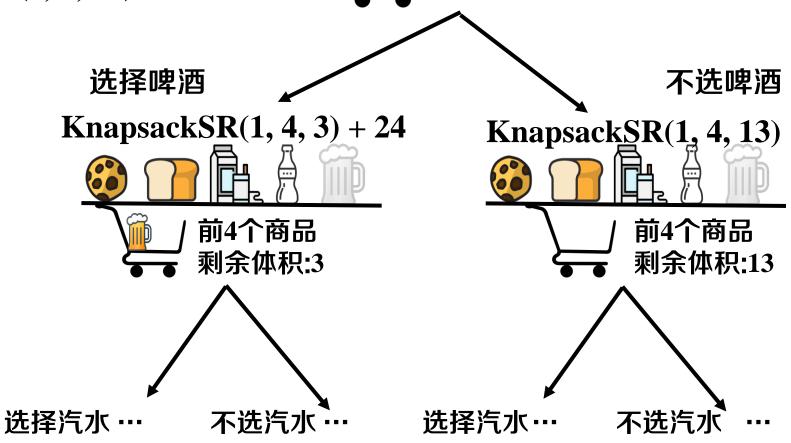
● 递归函数: KnapsackSR(h, i, c)

• 在第h个到第i个商品中,容量为c时最优解

选择啤酒: KnapsackSR(1, 4, 3) + 24

不选啤酒: KnapsackSR(1, 4, 13)

序号	商品		价格	体积
1		饼干	9	4
2		面包	10	5
3		牛奶	9	4
4		汽水	2	3
5		啤酒	24	10



KnapsackSR(1, 5, 13)

剩余体积:13

在第1到5个商品中选择



● 递归函数: KnapsackSR(h, i, c)

在第h个到第i个商品中,容量为c时最优解

选择啤酒: KnapsackSR(1, 4, 3) + 24

不选啤酒: KnapsackSR(1, 4, 13)

				选择啤酒	不选啤酒
				KnapsackSR(1, 4, 3) + 24	Knapsack S R(1, 4, 13)
序号	商品	价格	体积		
1	₩ 饼干	9	4	/ 前4个商品	/ 前4个商品
2	0 面包	10	5	剩余体积:3	剩余体积:13

在第1到5个商品中选择 剩余体积:13 不选啤酒

KnapsackSR(1, 5, 13)

KnapsackSR(1, 5, 13) =

 $\max\{\text{KnapsackSR}(1, 4, 3) + 24, \text{KnapsackSR}(1, 4, 13)\}$



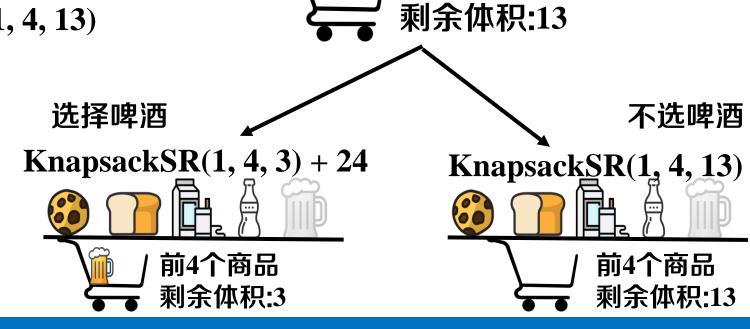
● 递归函数: KnapsackSR(h, i, c)

● 在第h个到第i个商品中,容量为c时最优解

选择啤酒: KnapsackSR(1, 4, 3) + 24

• 不选啤酒: KnapsackSR(1, 4, 13)

序号	商品	价格	体积
1	❸ 饼干	9	4
2	1 面包	10	5



KnapsackSR(1, 5, 13)

在第1到5个商品中选择

KnapsackSR(h, i, c) =

 $\max\{\operatorname{KnapsackSR}(h, i-1, c-v_i) + p_i, \operatorname{KnapsackSR}(h, i-1, c)\}$



```
输入: 商品集合\{h,...,i\}, 背包容量c
 输出: 最大总价格P
I if c < 0 then
                                  超出背包容量限制
    \operatorname{return} -\infty
 end
if i \le h-1 then
    return 0
 end
 P_1 \leftarrow \text{KnapsackSR}(h, i-1, c-v_i)
 P_2 \leftarrow \text{KnapsackSR}(h, i-1, c)
 P \leftarrow \max\{P_1 + p_i, P_2\}
 return P
```



```
输入: 商品集合\{h,...,i\}, 背包容量c
 输出: 最大总价格P
 if c < 0 then
    return -\infty
 end
if i \le h-1 then
                              所有商品已决策完成
  return 0
end
 P_1 \leftarrow \text{KnapsackSR}(h, i-1, c-v_i)
 P_2 \leftarrow \text{KnapsackSR}(h, i-1, c)
 P \leftarrow \max\{P_1 + p_i, P_2\}
 return P
```



```
输入: 商品集合\{h,...,i\}, 背包容量c
 输出: 最大总价格P
 if c < 0 then
    return -\infty
 end
 if i \leq h-1 then
  return 0
 end
P_1 \leftarrow \text{KnapsackSR}(h, i-1, c-v_i)
                                                 选则商品i
P_2 \leftarrow \text{KnapsackSR}(h, i-1, c)
 P \leftarrow \max\{P_1 + p_i, P_2\}
 return P
```



```
输入: 商品集合\{h,...,i\}, 背包容量c
 输出: 最大总价格P
 if c < 0 then
    \operatorname{return} -\infty
 end
 if i \leq h-1 then
  | return 0
 end
 P_1 \leftarrow \text{KnapsackSR}(h, i-1, c-v_i)
P_2 \leftarrow \text{KnapsackSR}(h, i-1, c)
                                                    不选商品i
P \leftarrow \max\{P_1 + p_i, P_2\}
 return P
```



```
输入: 商品集合\{h,...,i\}, 背包容量c
 输出: 最大总价格P
 if c < 0 then
    return -\infty
 end
if i \leq h-1 then
return 0
 end
 P_1 \leftarrow \text{KnapsackSR}(h, i-1, c-v_i)
                                                 简化描述
 P_2 \leftarrow \text{KnapsackSR}(h, i-1, c)
 P \leftarrow \max\{P_1 + p_i, P_2\}
 return P
```



• KnapsackSR(i,c): 前i个商品中,容量为c时最优解

```
输入: 前i个商品, 背包容量c
 输出: 最大总价格P
 if c < 0 then
    return -\infty
 end
if i \leq 0 then
return 0
end
P_1 \leftarrow \text{KnapsackSR}(i-1,c-v_i)
                                                 简化描述
P_2 \leftarrow \text{KnapsackSR}(i-1,c)
P \leftarrow \overline{\max}\{P_1 + p_i, P_2\}
 return P
```



• KnapsackSR(i,c): 前i个商品中,容量为c时最优解

```
输入: 前i个商品, 背包容量c
输出: 最大总价格P
if c < 0 then
    return -\infty
end
if i \le 0 then
 return 0
end
P_1 \leftarrow \text{KnapsackSR}(i-1, c-v_i)
P_2 \leftarrow \text{KnapsackSR}(i-1,c)
P \leftarrow \max\{P_1 + p_i, P_2\}
return P
```

确定最优子问题



• 递归树

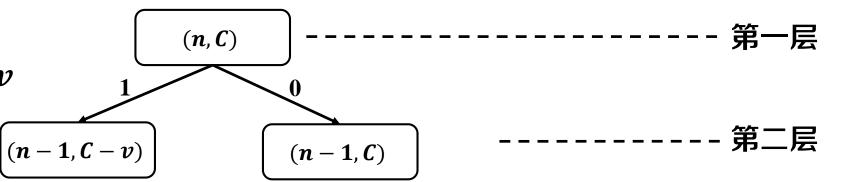
• 例: 商品体积均为v

(n, C) = | ----- 第一层

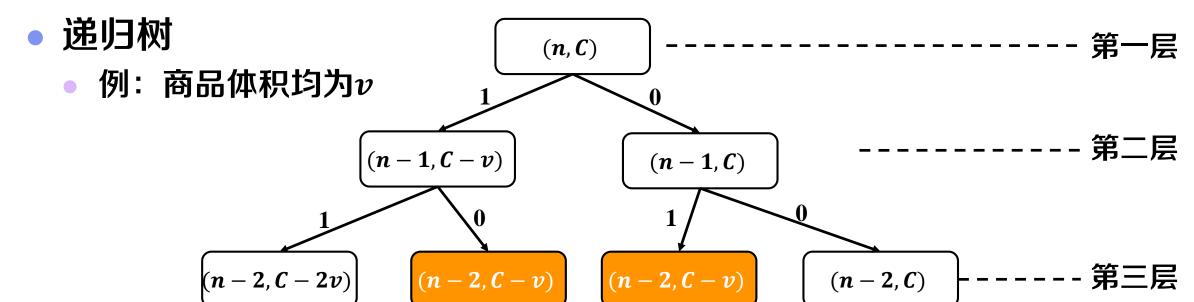


• 递归树

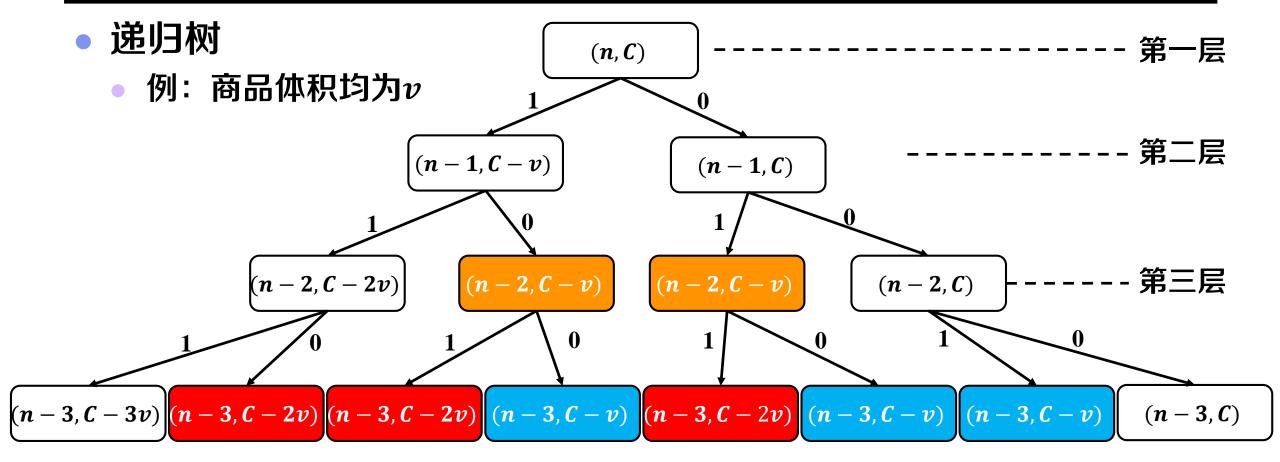
• 例: 商品体积均为v



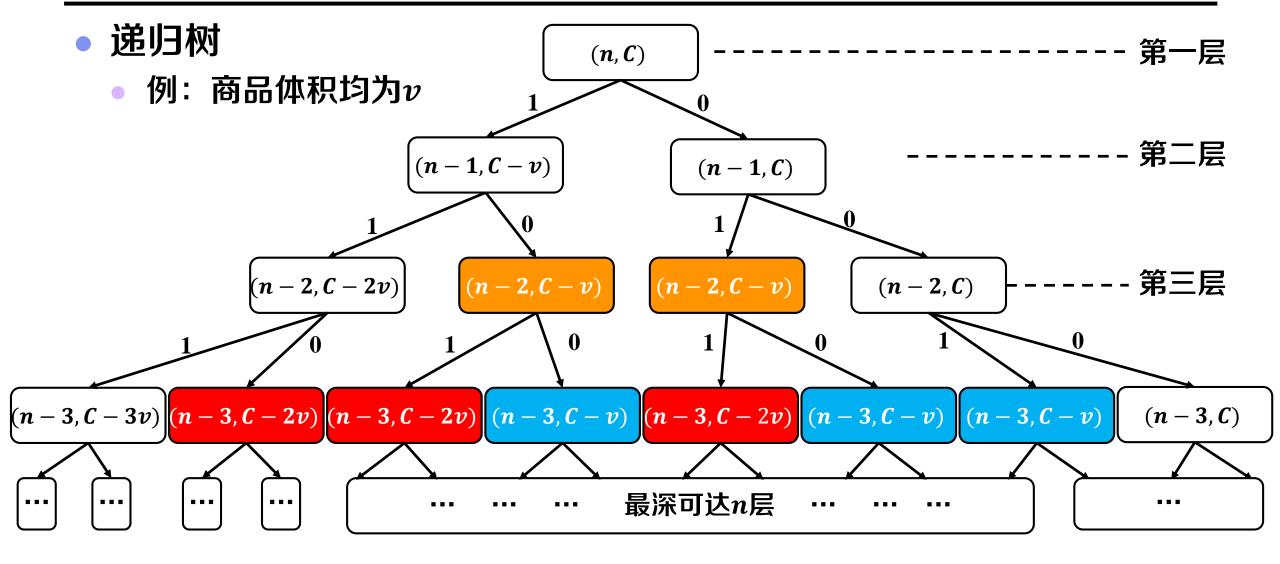




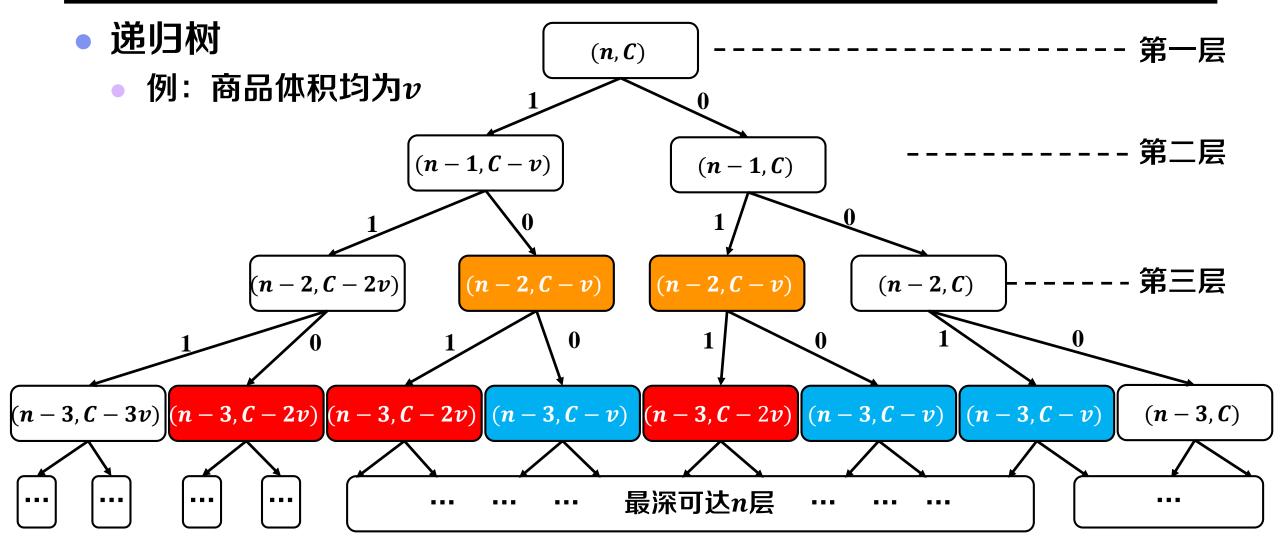






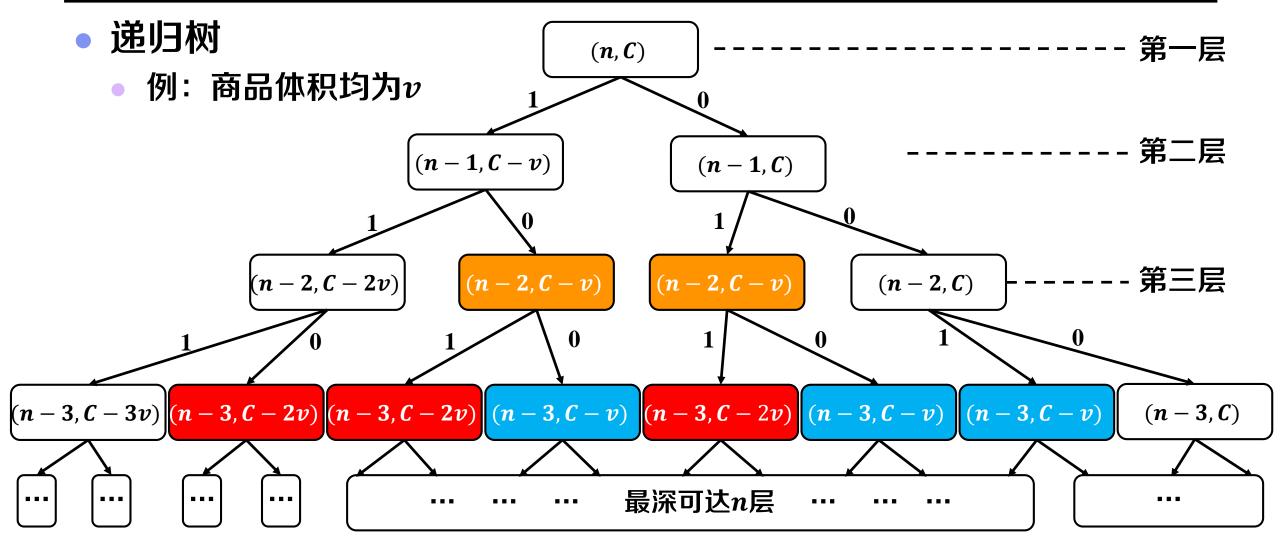






• 重复求解大量子问题 $O(2^n)$



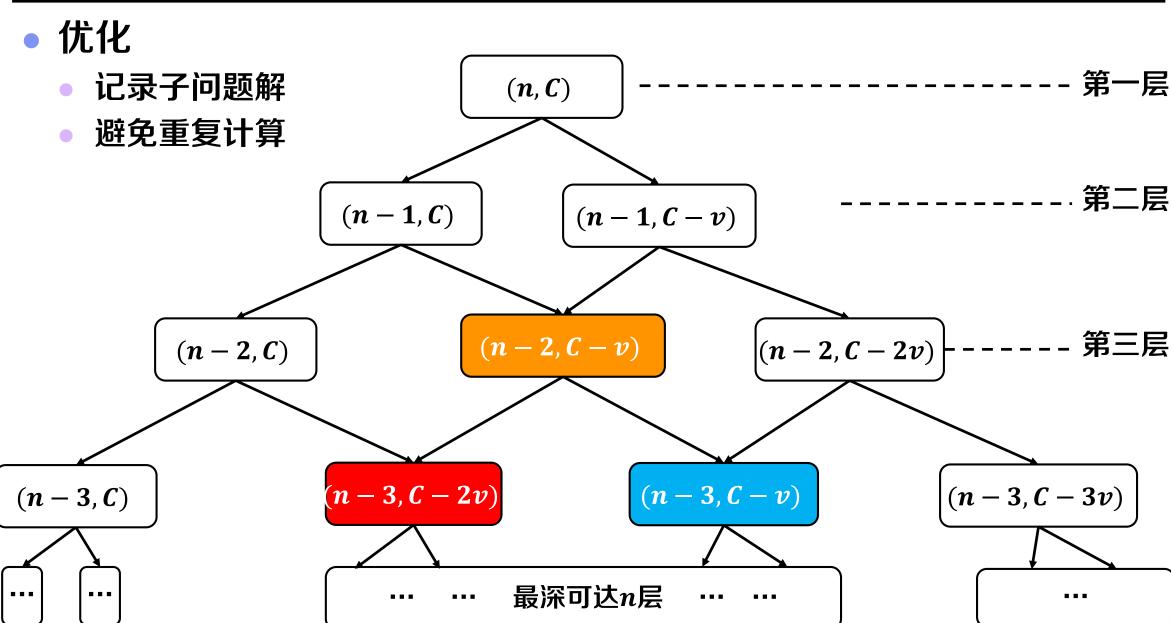


• 重复求解大量子问题 $O(2^n)$

问题: 如何优化?

从蛮力枚举到带备忘递归





带备忘递归: 伪代码



• KnapsackMR(i, c)

return P

```
输入: 商品集合\{1,...,i\}, 背包容量c
输出: 最大总价格P
if c < 0 then
 \perp return -\infty
end
if i \leq 0 then
 \perp return 0
end
P_1 \leftarrow \text{KnapsackMR}(i-1, c-v_i)
P_2 \leftarrow \text{KnapsackMR}(i-1,c)
P \leftarrow \max\{P_1 + p_i, P_2\}
```



• KnapsackMR(i, c)

```
输入: 商品集合\{1,...,i\}, 背包容量c
输出: 最大总价格P
if c < 0 then
 \perp return -\infty
end
if i \leq 0 then
 return 0
end
P_1 \leftarrow \text{KnapsackMR}(i-1, c-v_i)
P_2 \leftarrow \text{KnapsackMR}(i-1,c)
P \leftarrow \max\{P_1 + p_i, P_2\}
P[i,c] \leftarrow P
return P
```



• KnapsackMR(i, c)

```
输入: 商品集合\{1,...,i\}, 背包容量c输出: 最大总价格P if c < 0 then -\infty end if i \le 0 then -1 return 0 end end
```

```
P_1 \leftarrow \text{KnapsackMR}(i-1, c-v_i)
P_2 \leftarrow \text{KnapsackMR}(i-1, c)
P \leftarrow \max\{P_1 + p_i, P_2\}
P[i, c] \leftarrow P
\text{return } P
```

构造备忘录P[i,c]

P[i,c]表示在前i个商品中选择,背包容量为c时的最优解



• KnapsackMR(i, c)

```
输入: 商品集合\{1,...,i\}, 背包容量c
输出: 最大总价格P
if c < 0 then
 \perp return -\infty
end
if i \leq 0 then
 return 0
end
if P[i,c] \neq NULL then
| return P[i,c]
end
P_1 \leftarrow \text{KnapsackMR}(i-1,c-v_i)
P_2 \leftarrow \text{KnapsackMR}(i-1,c)
P \leftarrow \max\{P_1 + p_i, P_2\}
return P
```

构造备忘录P[i,c]

P[i,c]表示在前i个商品中选择,背包容量为c时的最优解



• KnapsackMR(i, c)

```
输入: 商品集合\{1,...,i\}, 背包容量c
 输出: 最大总价格P
 if c < 0 then
  \perp return -\infty
 end
 if i \leq 0 then
  return 0
end
(if P[i, c] \neq NULL then
lacksquare \mathbf{I} \mid \mathbf{return} \ P[i, c]
P_1 \leftarrow \text{KnapsackMR}(i-1, c-v_i)
 P_2 \leftarrow \text{KnapsackMR}(i-1,c)
 P \leftarrow \max\{P_1 + p_i, P_2\}
 return P
```

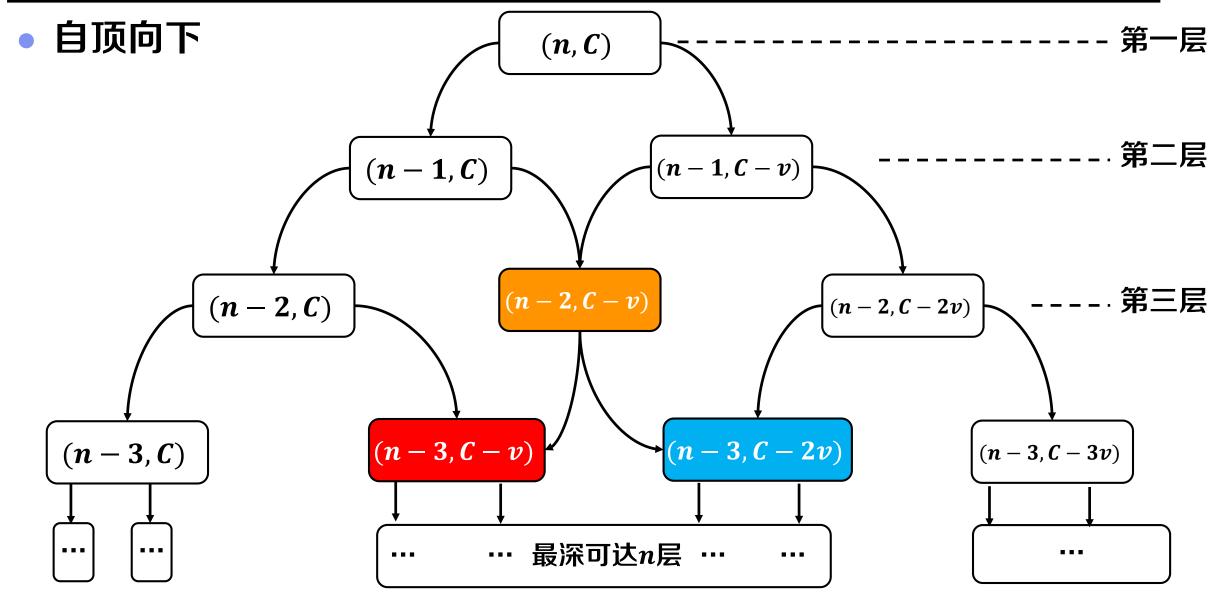
重复子问题

构造备忘录P[i,c]

P[i,c]表示在前i个商品中选择,背包容量为c时的最优解

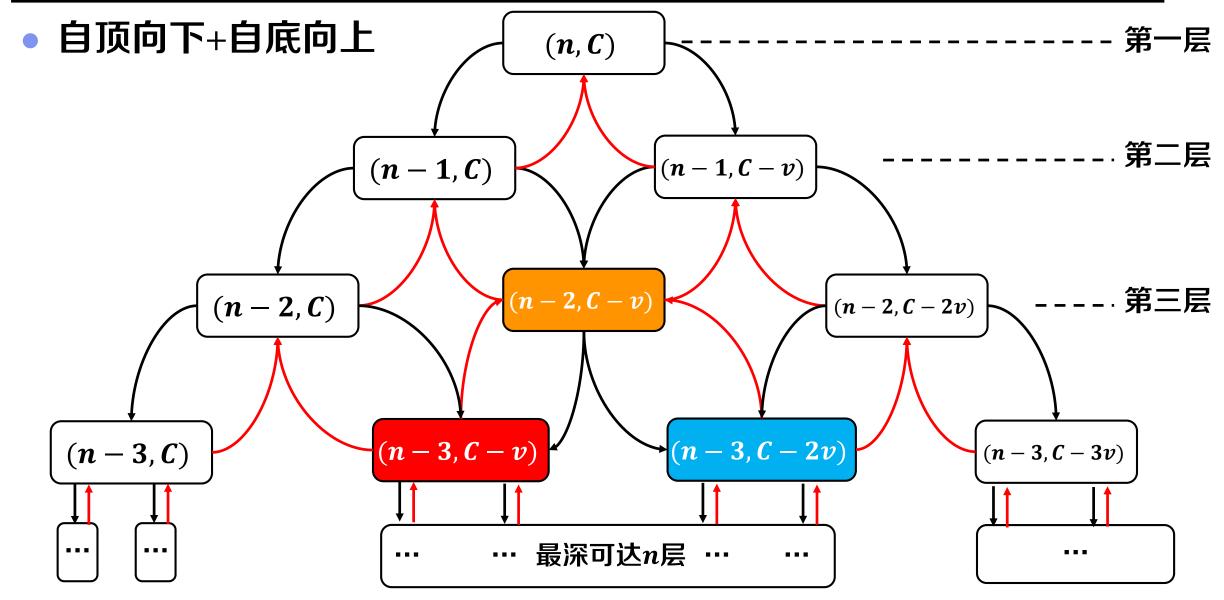
带备忘递归: 计算顺序





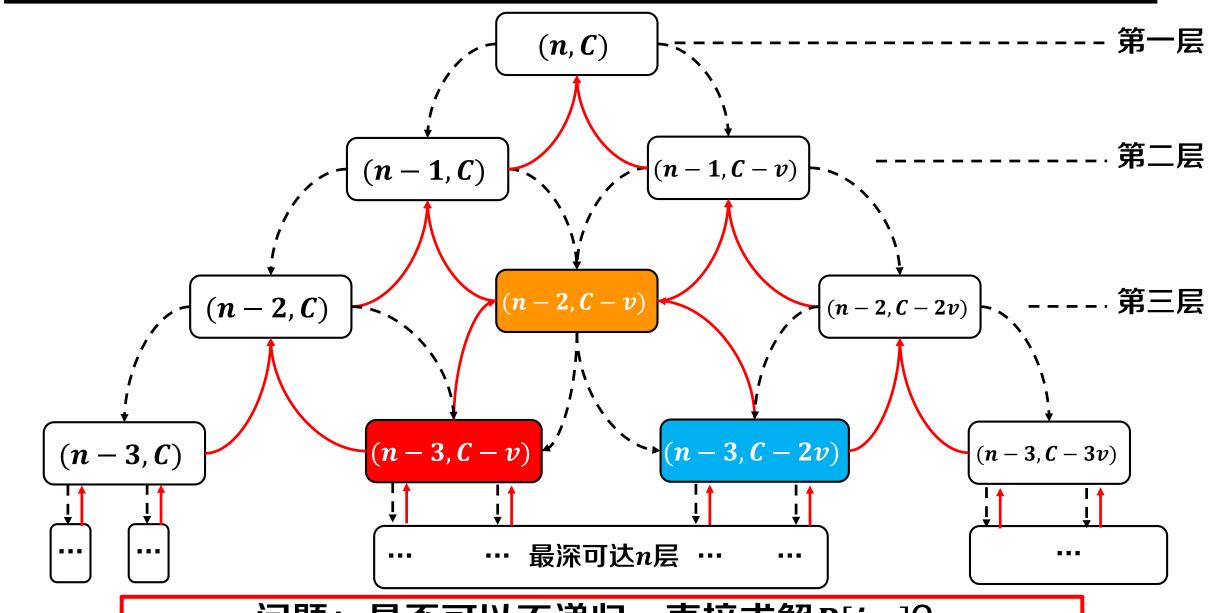
带备忘递归: 计算顺序





带备忘递归: 计算顺序





问题:是否可以不递归,直接求解P[i,c]?

递推计算



• 初始化

容量为0时: P[i,0] = 0

• 没有商品时: P[0,c] = 0

P[i,c]	c = 0	1	2	3	···	10	11	12	13
i = 0	0	0	0	0	•••	0	0	0	0
1	0								
•••	0								
n	0								
	\								

递推计算



- 确定计算顺序
 - $P[i-1,c-v_i]$ 和P[i-1,c]位于P[i,c]的左上方

P[i,c]	c = 0	1	2	3	•••	10	11	12	13	_
i = 0	0	0	0	0	•••	0	0	0	0	
1	0		1	Pli		P[i]	-1,c]			
•••	0			Pli-1,c	$v_{i]+p_{i}$	P[i,c]				
n	0									
n	0									

问题: 观察子问题依赖关系, 如何确定计算顺序?

递推计算



- 确定计算顺序
 - 按从左到右、从上到下的顺序计算

P[i,c]	c = 0	1	2	3	•••	10	11	12	13
i = 0	0	0	0	0	•••	0	0	0	0
1	0								→
					. — —				
•••	0	<u></u>							

问题: 观察子问题依赖关系, 如何确定计算顺序?



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i,c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0													
2	0		初期	怡化										
3	0													
4	0													
5	0													

0

0



v_i	10	3	4	5	4									
p_i	24	2	9	10	9								C	= 13
													U	– 1 3
P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0													
2	0	·												
3	0											v[i] >	· C	



v_i	10	3	4	5	4
p_i	24	2	9	10	9

C = 13

P[i,c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0													
2	0	`												
3	0													
4	0													
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

C = 13

P[i,c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0				
2	0	`												
3	0													
4	0													
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

C = 13

P[i,c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0				
2	0													
3	0													
4	0													
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

C = 13

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	$\begin{bmatrix} 0 \end{bmatrix}$	0	0	0	0	0	0	0	0	0	0_	0	0	0
1	0	0	0	0	0	0	0	0	0	0				
2	0													
3	0													
4	0													
5	0													

递推公式: $P[i,c] = \max\{P[0,0] + p[1], P[0,10]\}$



v_i	10	3	4	5	4
p_i	24	2	9	10	9

C = 13

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	$\begin{bmatrix} 0 \end{bmatrix}$	0	0	0	0	0	0	0	0	0	0_	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24			
2	0													
3	0													
4	0													
5	0													

递推公式: $P[i, c] = \max\{0 + 24, 0\}$



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24		
2	0													
3	0													
4	0													
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	
2	0													
3	0													
4	0													
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	$\begin{bmatrix} 0 \end{bmatrix}$	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0													
3	0													
4	0													
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0											
3	0													
4	0				v[i]	> <i>c</i>								
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

C = 13

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2										
3	0													
4	0													
5	0													

递推公式: $P[i,c] = \max\{0 + 2, 0\}$



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2									
3	0													
4	0													
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2								
3	0													
4	0													
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2							
3	0													
4	0													
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ĩ	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2						
3	0													
4	0													
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2					
3	0													
4	0													
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2				
3	0													
4	0													
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2				
3	0													
4	0													
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

C = 13

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24			
3	0													
4	0													
5	0													

递推公式: $P[i, c] = \max\{0 + 2, 24\}$



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24		
3	0													
4	0													
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	
3	0													
4	0													
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

C = 13

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0													
4	0													
5	0													

递推公式: $P[i,c] = \max\{24+2, 24\}$



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i,c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2										
4	0													
5	0					v[i]	>c							



v_i	10	3	4	5	4
p_i	24	2	9	10	9

C = 13

P[i,c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9									
4	0													
5	0													

递推公式: $P[i,c] = \max\{0 + 9, 2\}$



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9								
4	0													
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9							
4	0													
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11						
4	0													
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11					
4	0													
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11				
4	0													
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24			
4	0													
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24		
4	0													
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	
4	0													
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ĩ	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0													
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

C = 13

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9									
5	0													

v[i] > c



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9	10								
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9	10	10							
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9	10	10	11						
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9	10	10	11	12					
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11		24	24	26
4	0	0	0	2	9	10	10	11	12	19				
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9	10	10	11	12	19	24			
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9	10	10	11	12	19	24	24		
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9	10	10	11	12	19	24	24	24	
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9	10	10	11	12	19	24	24	24	26
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

C = 13

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9	10	10	11	12	19	24	24	24	26
5	0	0	0	2										

v[i] > c



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i,c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9	10	10	11	12	19	24	24	24	26
5	0	0	0	2	9									



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9	10	10	11	12	19	24	24	24	26
5	0	0	0	2	9	10								



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9	10	10	11	12	19	24	24	24	26
5	0	0	0	2	9	10	10							

W. T. S. A.

4 // 0 UNIVE

v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9	10	10	11	12	19	24	24	24	26
5	0	0	0	2	9	10	10	11						



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9	10	10	11	12	19	24	24	24	26
5	0	0	0	2	9	10	10	11	18					



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9	10	10	11	12	19	24	24	24	26
5	0	0	0	2	9	10	10	11	18	19				



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9	10	10	11	12	19	24	24	24	26
5	0	0	0	2	9	10	10	11	18	19	24			



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i,c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9	10	10	11	12	19	24	24	24	26
5	0	0	0	2	9	10	10	11	18	19	24	24		



v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9	10	10	11	12	19	24	24	24	26
5	0	0	0	2	9	10	10	11	18	19	24	24	24	

/	张宝龙中
(*(*
1	
/	No UNIVE

v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9	10	10	11	12	19	24	24	24	26
5	0	0	0	2	9	10	10	11	18	19	24	24	24	28

W. T. S. #
(* * * * * * * * * * * * * * * * * * *
4 // 0 UNIVE

v_i	10	3	4	5	4
p_i	24	2	9	10	9

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2		つん 三人士 名刀	24	26
3	0	0	0	2	9	9	9	11	11	11	24	最优解 	24	26
4	0	0	0	2	9	10	10	11	12	19	24	24	24	26
5	0	0	0	2	9	10	10	11	18	19	24	24	24	28



v_i	10	3	4	5	4
p_i	24	2	9	10	9

C = 13

P[i,c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	○/	24	26
3	0	0	0	2	9	9	9	11	11	11	24	最优解	24	26
4	0	0	0	2	9	10	10	11	12	19	24	24	24	26
5	0	0	0	2	9	10	10	11	18	19	24	24	24	28

问题: 如何确定选取了哪些商品?



• 递推公式

•
$$P[i, c] = \max\{P[i-1, c-v_i] + p_i, P[i-1, c]\}$$

• 记录决策过程

•
$$Rec[i,c] = \begin{cases} 1, &$$
 选择商品 $0, &$ 不选商品

P[i,c]	c = 0	1	2	3	•••	10	11	12	13
i = 0	0	0	0	0	•••	0	0	0	0
1	0			Rest		Re	ec[i,c] =	= 0	
•••	0			Rec[i, c	/=1	P[i,c]			
\boldsymbol{n}	0								



• 递推公式

•
$$P[i, c] = \max\{P[i-1, c-v_i] + p_i, P[i-1, c]\}$$

• 记录决策过程

•
$$Rec[i,c] = \begin{cases} 1, &$$
 选择商品 $0, &$ 不选商品

P[i,c]	c = 0	1	2	3	•••	10	11	12	13
i = 0	0	0	0	0	•••	0	0	0	0
1	0			Rest		Re	ec[i,c] =	= 0	
•••	0			Celi,c	=1	P[i,c]			
n	0								



• 递推公式

•
$$P[i, c] = \max\{P[i-1, c-v_i] + p_i, P[i-1, c]\}$$

• 记录决策过程

•
$$Rec[i,c] = \begin{cases} 1, &$$
 选择商品 $0, &$ 不选商品

P[i,c]	c = 0	1	2	3	•••	10	11	12	13
i = 0	0	0	0	0	•••	0	0	0	0
1	0			Rest		Re	ec[i,c] =	= 0	
•••	0			Celi, c	=1	P[i,c]			
n	0								



• 递推公式

•
$$P[i, c] = \max\{P[i-1, c-v_i] + p_i, P[i-1, c]\}$$

• 回溯解决方案

- 倒序判断是否选择商品
- 根据选择结果,确定最优子问题

P[i,c]	c = 0	1	2	3	•••	10	11	12	13
i = 0	0	0	0	0	•••	0	0	0	0
1	0			Real		↑ Re	ec[i,c] =	= 0	
•••	0			Rec[i, c	\ = 1	P[i,c]			
n	0								



v_i	10	3	4	5	4									
p_i	24	2	9	10	9								C	= 13
P[i,c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0				
2	0								_					
3	0									v	e[i] > c			
4	0													
5	0													
Rec	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 1	0	0	0	0	0	0	0	0	0	0				
2	0													
3	0										当前状	态最优	解不包	含商品i
4	0													
5	0													



v_i	10	3	4	5	4
p_i	24	2	9	10	9

C = 13

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24			
2	0										(
3	0													
4	0													
5	0													

Rec	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 1	0	0	0	0	0	0	0	0	0	0	1			
2	0													
3	0										当前壮	大态最优	尤解包包	含商品i
4	0													
5	0													

0



v_i	10	3	4	5	4
p_i	24	2	9	10	9

C = 13

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24			
3	0													
4	0													
5	0													

Rec	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 1	0	0	0	0	0	0	0	0	0	0	1	1	1	1
2	0	0	0	1	1	1	1	1	1	1	0			
3	0													
4	0										当前	犬态最低	尤解不	包含商品



v_i	10	3	4	5	4									
p_i	24	2	9	10	9								C	= 13
P[i,c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9	10	10	11	12	19	24	24	24	26
5	0	0	0	2	9	10	10	11	18	19	24	24	24	
Rec	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
	0	^	•	0	•	^	0	0	0	0	4	_	4	4

	rec	$\iota - 0$	1	4	3	4	5	U	/	O	9	10	11	12	13
i	= 1	0	0	0	0	0	0	0	0	0	0	1	1	1	1
	2	0	0	0	1	1	1	1	1	1	1	0	0	0	1
	3	0	0	0	0	1	1	1	1	1	1	0	0	0	0
	4	0	0	0	0	0	1	1	0	1	1	0	0	0	0
	5	0	0	0	0	0	0	0	0	1	0	0	0	0	



v_i	10	3	4	5	4									
p_i	24	2	9	10	9								C	=
P[i,c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	24	24	24	
2	0	0	0	2	2	2	2	2	2	2	24	24	24	
3	0	0	0	2	9	9	9	11	11	11	24	24	24	
4	0	0	0	2	9	10	10	11	12	19	24	24	24	
5	0	0	0	2	9	10	10	11	18	19	24	24	24	
Dog	a - 0	1	2	2	1	5		7	O	0	10	11	10	
Rec	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	
i = 1	0	0	0	0	0	0	0	0	0	0	1	1	1	

											_			
2	0	0	0	1	1	1	1	1	1	1	0	0	0	1
3	0	0	0	0	1	1	1	1	1	1	0	0	0	0
4	0	0	0	0	0	1	1	0	1	1	0	0	0	0
5	0	0	0	0	0	0	0	0	1	0	0	0	0	1

424
74 T
1 10 UNIVE

v_i	10	3	4	5	4
p_i	24	2	9	10	9

选取的商品: C = 13

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9	10	10	11	12	19	24	24	24	26
5	0	0	0	2	9	10	10	11	18	19	24	24	24	28

Rec	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 1	0	0	0	0	0	0	0	0	0	0	1	1	1	1
2	0	0	0	1	1	1	1	1	1	1	0	0	0	1
3	0	0	0	0	1	1	1	1	1	最	优解包	含商品	15	0
4	0	0	0	0	0	1	1	0	1	1	0	0		0
5	0	0	0	0	0	0	0	0	1	0	0	0	0	1



v_i	10	3	4	5	4
p_i	24	2	9	10	9

选取的商品: 5,4 C=13

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9	10	10	11	12	19	24	24	24	26
5	0	0	0	2	9	10	10	11	18	19	24	24	24	28

Rec	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 1	0	0	0	0	0		化 级石	今商品	1	0	1	1	1	1
2	0	0	0	1	1	最	优解包	.含商品 	յ4 —	1	0	0	0	1
3	0	0	0	0	1	1	1	1	1	1	0	0	0	0
4	0	0	0	0	0	1	1	0	1	1	0	0	0	0
5	0	0	0	0	0	0	0	0	1	0	0	0	0	1



v_i	10	3	4	5	4
p_i	24	2	9	10	9

选取的商品: 5, 4, 3 C = 13

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9	10	10	11	12	19	24	24	24	26
5	0	0	0	2	9	10	10	11	18	19	24	24	24	28

Rec	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 1	0	0	0	0	0	0	0	0	0	0	1	1	1	1
2	0	0	0	1	1	1	1	1	1	1	0	0	0	1
3	0	0	0	0	1	1	1	1	1	1	0	0	0	0
4	0	0	0	0	0		1	0	1	1	0	0	0	0
5	0	0	0	0	当前状态最优解包含商品3					0	0	0	0	1



v_i	10	3	4	5	4
p_i	24	2	9	10	9

选取的商品: 5, 4, 3 C = 13

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9	10	10	11	12	19	24	24	24	26
5	0	0	0	2	9	10	10	11	18	19	24	24	24	28

Rec	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 1	0	0	0	0	0	0	0	0	0	0	1	1	1	1
2	0	0	0	1	1	1	1	1	1	1	0	0	0	1
3	0	0	0	0	1	1	1	1	1	1	0	0	0	0
4		最优解	不包含	含商品2		1	1	0	1	1	0	0	0	0
5	0	0	0	0	0	0	0	0	1	0	0	0	0	1



v_i	10	3	4	5	4
p_i	24	2	9	10	9

选取的商品: 5, 4, 3 C = 13

P[i, c]	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	24	24	24	24
2	0	0	0	2	2	2	2	2	2	2	24	24	24	26
3	0	0	0	2	9	9	9	11	11	11	24	24	24	26
4	0	0	0	2	9	10	10	11	12	19	24	24	24	26
5	0	0	0	2	9	10	10	11	18	19	24	24	24	28

Rec	c = 0	1	2	3	4	5	6	7	8	9	10	11	12	13
i = 1	0	0	0	0	0	0	0	0	0	0	1	1	1	1
2	0	0	0	1	1	1	1	1	1	1	0	0	0	1
3	0	0	0	0	1	1	1	1	1	1	0	0	0	0
4	0	0	0	0	0	1	1	0	1	1	0	0	0	0
5	0	0	0	0	0	0	0	0	1	0	0	0	0	1



```
输入: 商品数量n,各商品的价值p, 各商品的体积v, 背包容量C
输出: 商品价格的最大值,最优解方案
//初始化
创建二维数组 P[0..n,0..C] 和 Rec[0..n,0..C]
for i \leftarrow 0 to C do
                              初始化
   P[0,i] \leftarrow 0
\mathbf{I} end
for i \leftarrow 0 to n do
   P[i,0] \leftarrow 0
end
```



```
//求解表格
for i \leftarrow 1 to n do
                                                  依次计算子问题
     for c \leftarrow 1 to C do
        \neg \mathbf{if} \ (v[i] \leq c) \ and
         (p[i] + P[i-1, c-v[i]] > P[i-1, c]) then
             P[i,c] \leftarrow p[i] + P[i-1,c-v[i]]
             Rec[i,c] \leftarrow 1
         end
         else
             P[i,c] \leftarrow P[i-1,c]
             Rec[i,c] \leftarrow 0
         end
     \mathbf{end}
end
```



```
//求解表格
for i \leftarrow 1 to n do
    for c \leftarrow 1 to C do
      \int \mathbf{if} \ \overline{(v[i] \leq c)} \ and
                                                                      选择商品i
      (p[i] + P[i-1, c-v[i]] > P[i-1, c]) then
        P[i, c] \leftarrow p[i] + P[i-1, c-v[i]]
            Rec[i,c] \leftarrow 1
        end
        else
             P[i,c] \leftarrow P[i-1,c]
             Rec[i,c] \leftarrow 0
         end
    end
end
```



```
//求解表格
for i \leftarrow 1 to n do
     for c \leftarrow 1 to C do
          if (v[i] \leq c) and
           \begin{array}{l} (p[i] + P[i-1, c-v[i]] > P[i-1, c]) \text{ then} \\ | P[i, c] \leftarrow p[i] + P[i-1, c-v[i]] \end{array} 
                                                                                       记录价格和决策
            Rec[i,c] \leftarrow 1
           end
           else
                P[i,c] \leftarrow P[i-1,c]
                Rec[i,c] \leftarrow 0
           end
     end
end
```



```
//求解表格
for i \leftarrow 1 to n do
    for c \leftarrow 1 to C do
        if (v[i] \leq c) and
        (p[i] + P[i-1, c-v[i]] > P[i-1, c]) then
            P[i, c] \leftarrow p[i] + P[i - 1, c - v[i]]
            Rec[i,c] \leftarrow 1
        end
       felse
                                                                   不选商品i
       P[i,c] \leftarrow P[i-1,c]
       Rec[i,c] \leftarrow 0
       \mathbf{end}
    end
end
```



```
//输出最优解方案
\int_{\mathbf{for}} \frac{K \leftarrow C}{\mathbf{for}} \ i \leftarrow n \ to \ 1 \ \mathbf{do}
                                                                   倒序判断是否选择该商品
     if Rec[i,K] \equiv 1 then
          print 选择商品i K \leftarrow K - v[i]
       end
       else
        │ print 不选商品i
       end
  end
  return P[n, C]
```



```
//输出最优解方案
K \leftarrow C
for i \leftarrow n \text{ to } 1 \text{ do}
   if Rec[i, K] = 1 then
                                                     选择商品i
   F print 选择商品i — K \leftarrow K - v[i]
    end
    else
     │ print 不选商品i
    end
end
return P[n, C]
```



```
//输出最优解方案
K \leftarrow C
for i \leftarrow n to 1 do
    if Rec[i, K] = 1 then
       \operatorname{print} 选择商品i K \leftarrow K - v[i]
                                                         回溯子问题
    \mathbf{end}
    else
     print 不选商品i
    end
end
return P[n, C]
```



```
//输出最优解方案
K \leftarrow C
for i \leftarrow n to 1 do
   if Rec[i, K] = 1 then
       print 选择商品i
      K \leftarrow K - v[i]
   end
   (else
                                              不选商品i
    │ print 不选商品i
   \mathbf{end}
end
return P[n, C]
```

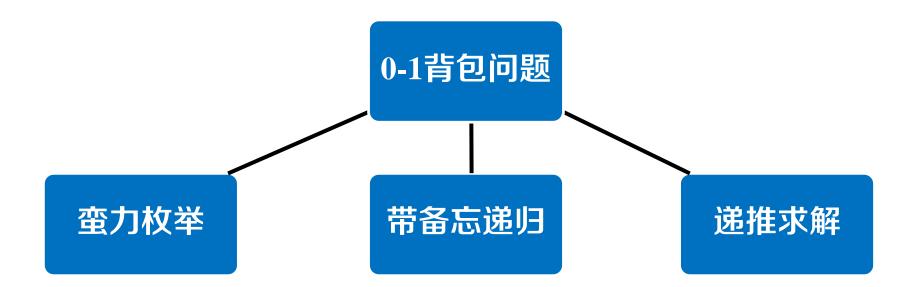
时间复杂度分析



```
//求解表格
for i \leftarrow 1 to n do
   for c \leftarrow 1 to C do
       if (v[i] \leq c) and
       (p[i] + P[i-1, c-v[i]] > P[i-1, c]) then
          P[i,c] \leftarrow p[i] + P[i-1,c-v[i]]
           Rec[i,c] \leftarrow 1
       end
       else
          P[i,c] \leftarrow P[i-1,c]
          Rec[i,c] \leftarrow 0
       end
    end
                                               时间复杂度: O(n \cdot C)
end
```

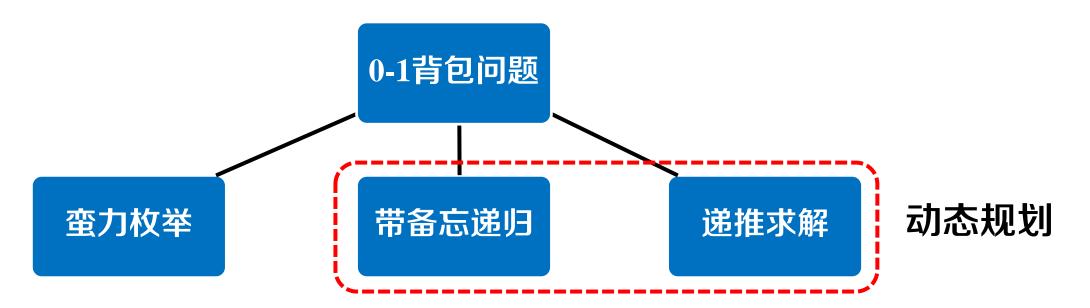
动态规划



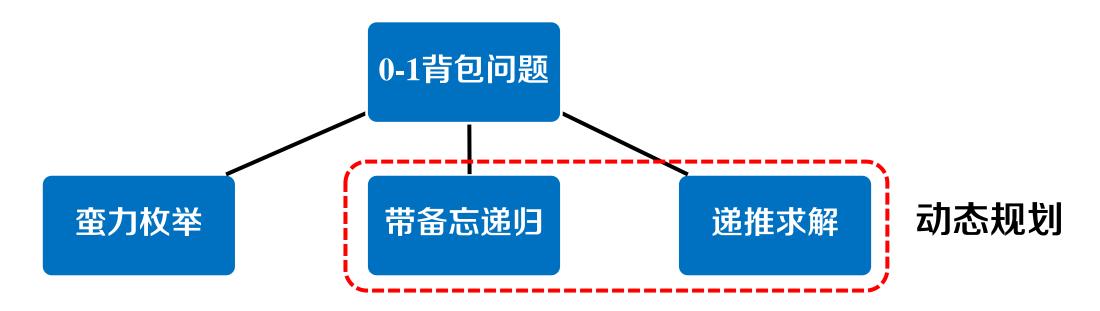


动态规划



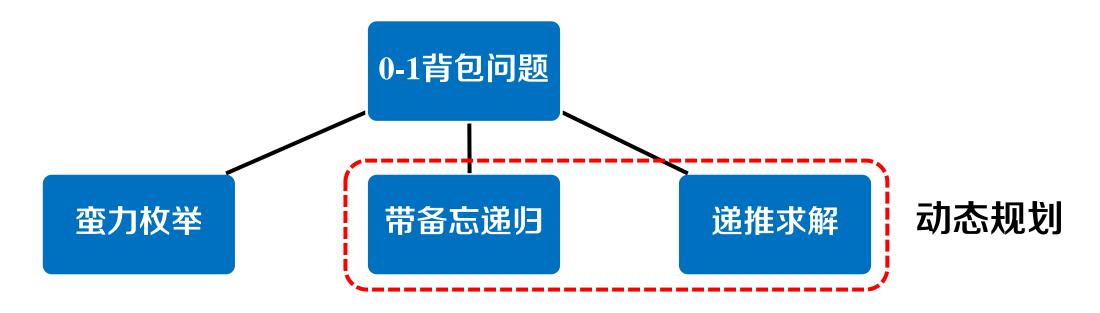






	带备忘递归	递推求解
相同	分解问题 ,	寻找关系
不同	自顶向下	自底向上





	带备忘递归	递推求解	
相同	分解问题 ,	寻找关系	
不同	自顶向下	自底向上	更高效



• 给出问题表示

• P[i,c]: 前i个商品可选、背包容量为c时的最大总价格

问题结构分析



• 明确原始问题

• P[n,C]: 前n个商品可选、背包容量为C时的最大总价格

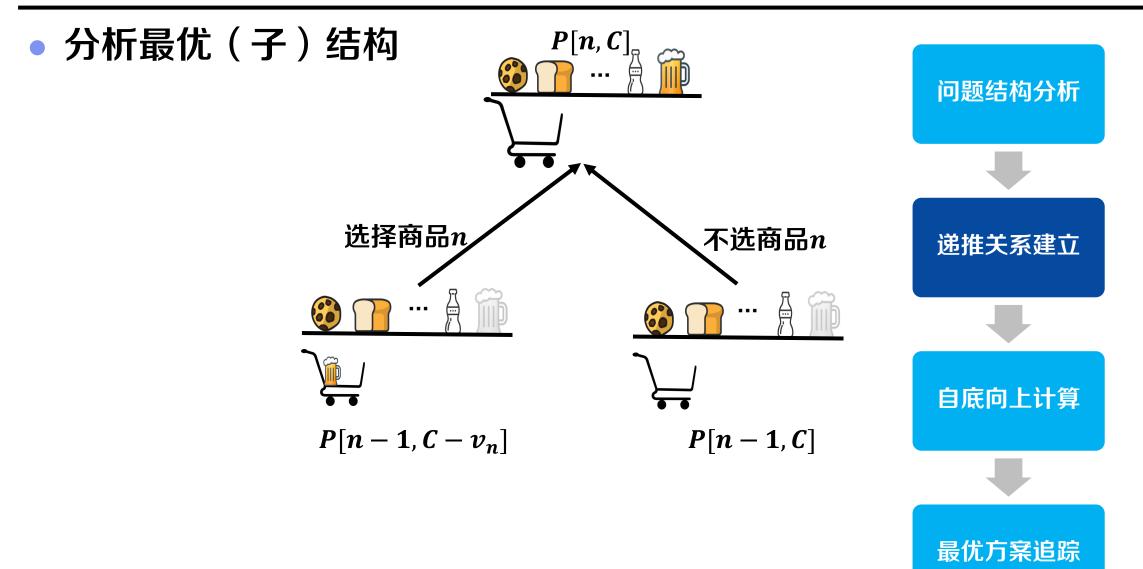
递推关系建立



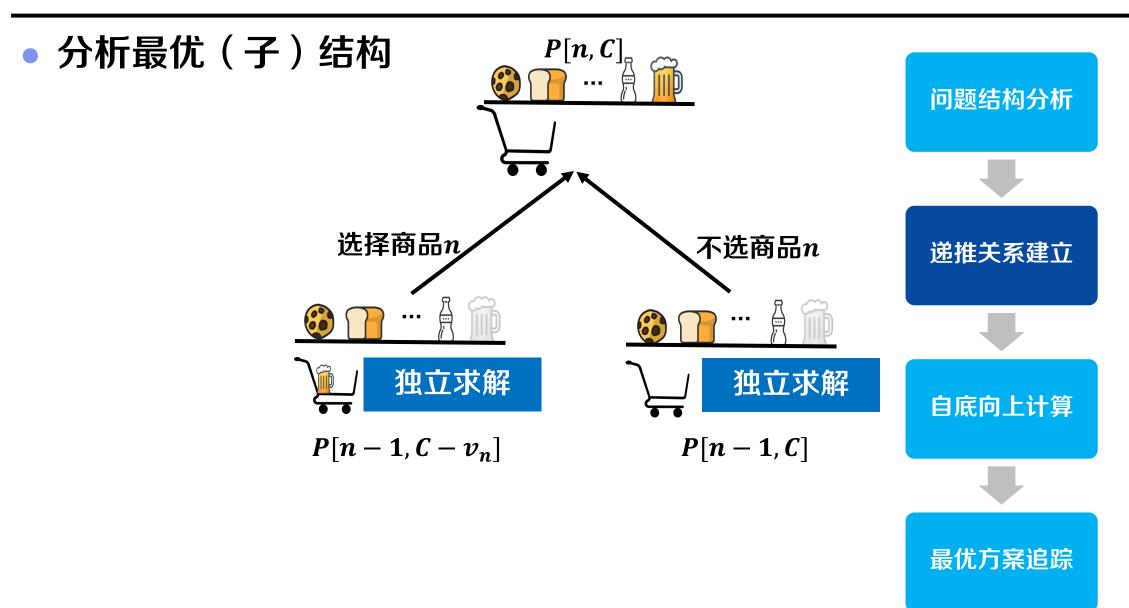
自底向上计算



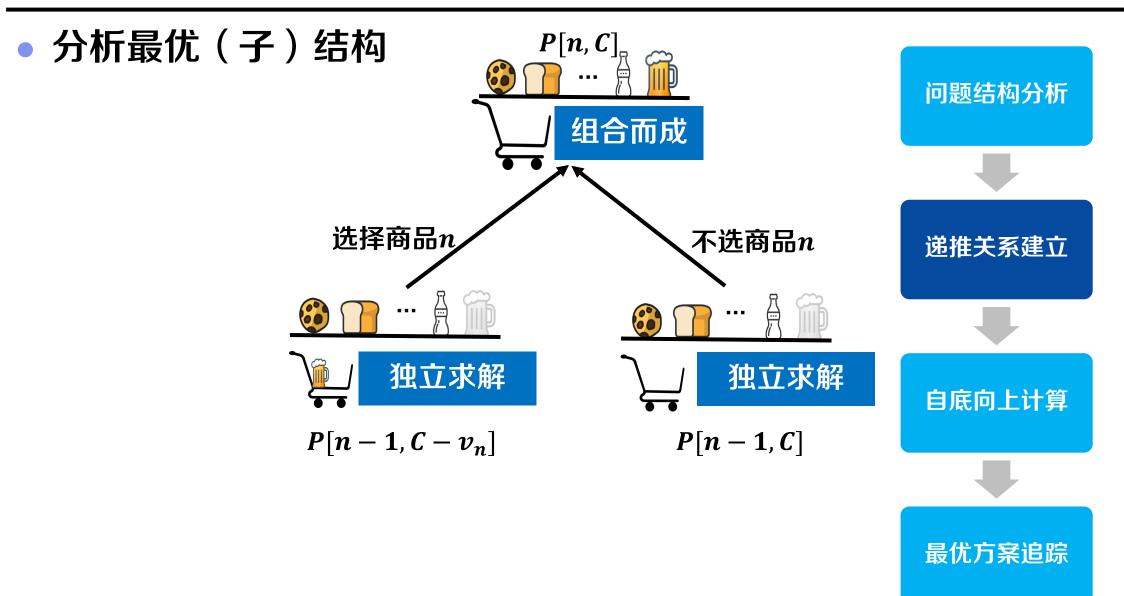














• 分析最优(子)结构

最优子结构性质(Optimal Substructure)

问题的最优解由相关子问题最优解组合而成

子问题可以独立求解

问题结构分析



递推关系建立



自底向上计算





• 分析最优(子)结构

最优子结构性质(Optimal Substructure)

问题的最优解由相关子问题最优解组合而成

子问题可以独立求解

问题结构分析



递推关系建立



自底向上计算



- 构造递推公式
 - $P[i,c] = \max\{P[i-1,c], P[i-1,c-v_i] + p_i\}$



• 确定计算顺序

- 初始化
 - 容量为0时: P[i,0] = 0 没有商品时: P[0,c] = 0
- P[i,c]依赖于子问题 $P[i-1,c-v_i]$ 和P[i-1,c]

• 依次求解问题

P[i,c]	c = 0	1	2	3	•••	10	11	12	13
i = 0	0	0	0	0	•••	0	0	0	0
1	0	-							
2	0	<u>4-</u>							 >
3	0	4-							
4	0	<u>4-</u> -							 >
5	0								

问题结构分析



递推关系建立



自底向上计算





• 记录决策过程

•
$$Rec[i,c] = \begin{cases} 1, &$$
 选择商品 $0, &$ 不选商品

P	c = 0	1	2	•••	9	10	11	12	13
i = 1									
2			R	Poli			Rec[i, c] = 0	
3						$ \downarrow R $ $ P[i, c] $			
4									
5									

问题结构分析



递推关系建立



自底向上计算





• 输出最优方案

- 倒序判断是否选择商品
- Rec[i,c] = 1时,选择商品i,考察子问题 $P[i-1,c-v_i]$
- Rec[i,c] = 0时,不选商品i,考察子问题P[i-1,c]

P	c = 0	1	2		9	10	11	12	13
i = 1		Rec[i	, c/ = -						
2			1	\;					
3				<u>Re</u>	c[i, c] =	= 0	Rec[i c] = 1	
4							[3] (j=1	
5									

问题结构分析



递推关系建立

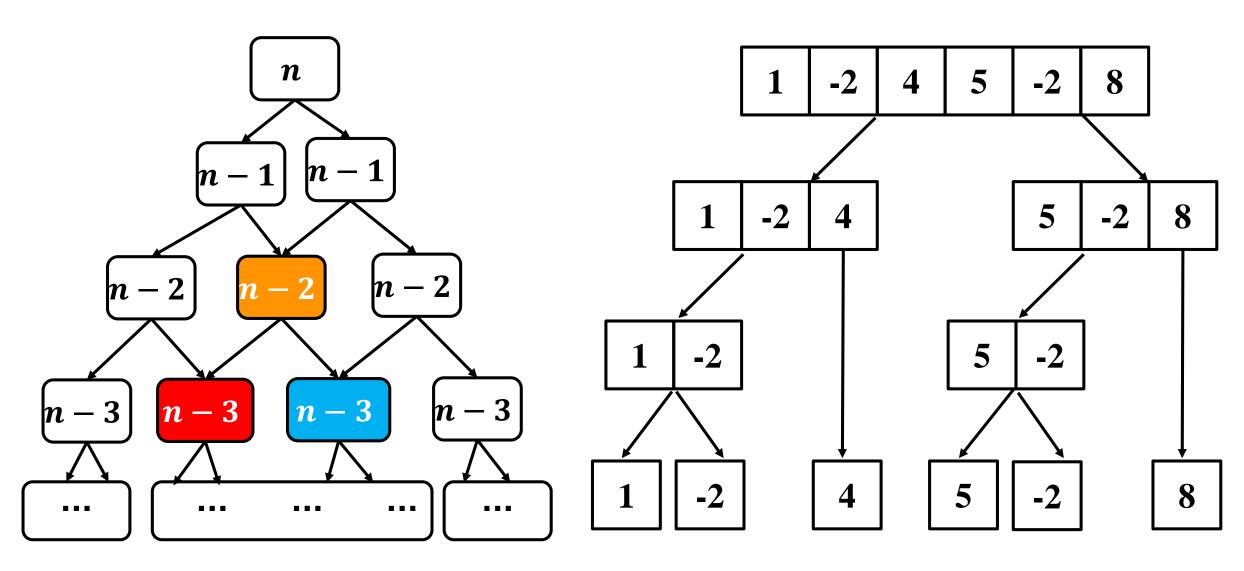


自底向上计算



方法比较





动态规划: 重叠子问题

分而治之: 独立子问题

动态规划小结



- 如何设计一个动态规划算法? 四个步骤
 - 问题结构分析
 - 。 给出问题表示,明确原始问题
 - 递推关系建立
 - 。分析最优结构,构造递推公式
 - 自底向上计算
 - 。 确定计算顺序,依次求解问题
 - 最优方案追踪
 - 。 记录决策过程,输出最优方案

问题结构分析



递推关系建立



自底向上计算

