

贪心策略篇： 部分背包问题






童咏昕

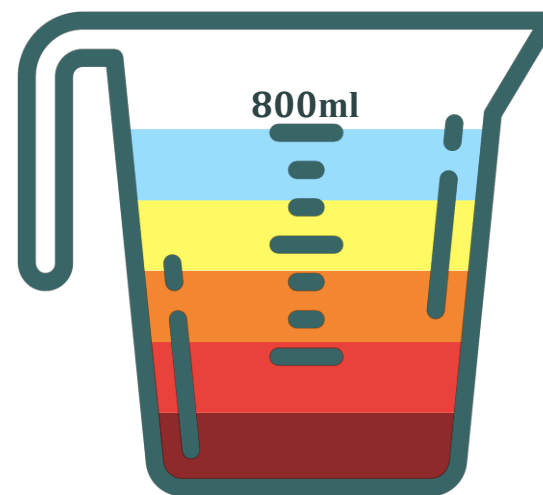
北京航空航天大学
计算机学院

中国大学MOOC北航《算法设计与分析》

● 调制饮品比赛

- 参赛者拥有容量为800ml的杯子，可任选不超过体积上限的饮料进行混合
- 调制饮品价格为各所使用饮料的价格之和，所得饮品价格之和最高者获胜

饮料	价格(元)	体积(ml)
 苏打水	60	600
 汽水	10	250
 橙汁	36	200
 苹果汁	16	100
 西瓜汁	45	300



问题：如何使调制的饮品价格最高？

部分背包问题

Fractional Knapsack Problem

输入

- n 个物品组成的集合 O ，每个物品有两个属性 v_i 和 p_i ，分别表示体积和价格
- 背包容量为 C

输出

选取物品的比例

- 求解一个解决方案 $S = \{x_i | 1 \leq i \leq n, 0 \leq x_i \leq 1\}$ ，使得：

$$\max \sum_{x_i \in S} x_i \cdot p_i$$

优化目标

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

约束条件

部分背包问题

Fractional Knapsack Problem

输入

- n 个物品组成的集合 O ，每个物品有两个属性 v_i 和 p_i ，分别表示体积和价格
- 背包容量为 C

输出

- 求解一个解决方案 $S = \{x_i | 1 \leq i \leq n, 0 \leq x_i \leq 1\}$ ，使得：

$$\max \sum_{x_i \in S} x_i \cdot p_i$$

优化目标

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

约束条件

选取物品的比例

x_i 只能取0或1时
变为0-1背包问题





- **最高性价比优先**

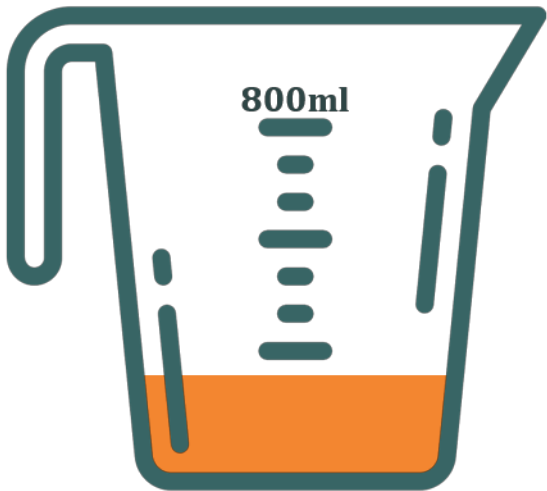
- 性价比 = 价格/体积
- 优先选择**高性价比**饮料全部装入，尽可能装满杯子

饮料	价格 (元)	体积 (ml)	性价比 (元/ml)
 苏打水	60	600	0.10
 汽水	10	250	0.04
 橙汁	36	200	0.18
 苹果汁	16	100	0.16
 西瓜汁	45	300	0.15



- 最高性价比优先
 - 解决方案





饮料	价格 (元)	体积 (ml)	总价格 (元)
 橙汁	36	200	

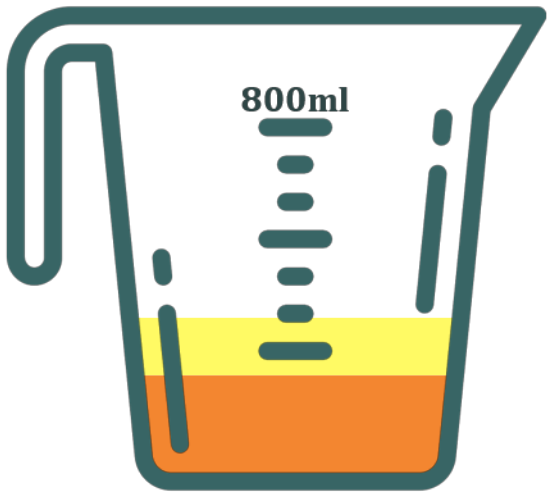
饮料	价格 (元)	体积 (ml)	性价比 (元/ml)
 橙汁	36	200	0.18
 苹果汁	16	100	0.16
 西瓜汁	45	300	0.15
 苏打水	60	600	0.10
 汽水	10	250	0.04






- 最高性价比优先
 - 解决方案

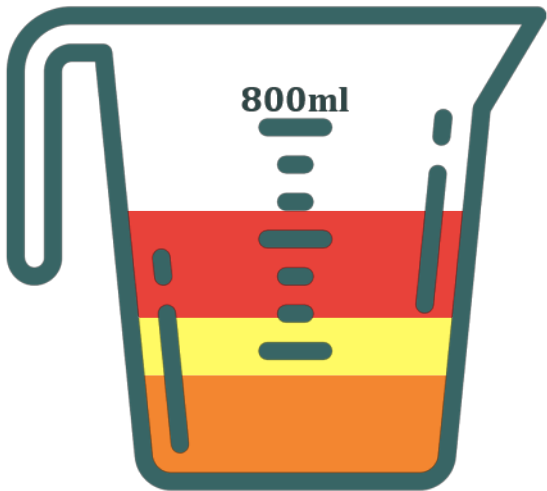
饮料	价格 (元)	体积 (ml)	总价格 (元)
 橙汁	36	200	
 苹果汁	16	100	






饮料	价格 (元)	体积 (ml)	性价比 (元/ml)
 橙汁	36	200	0.18
 苹果汁	16	100	0.16
 西瓜汁	45	300	0.15
 苏打水	60	600	0.10
 汽水	10	250	0.04





- 最高性价比优先
 - 解决方案

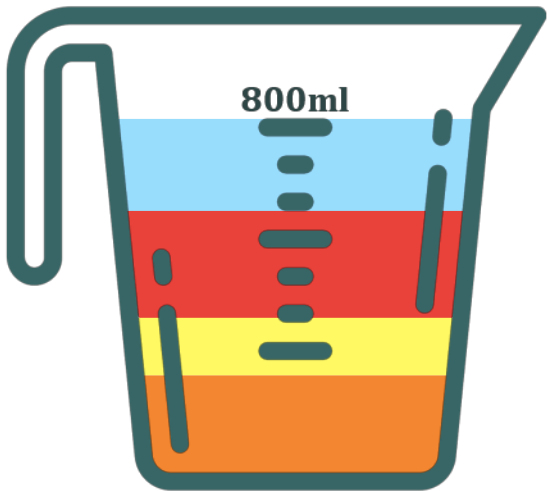
饮料	价格 (元)	体积 (ml)	总价格 (元)
 橙汁	36	200	
 苹果汁	16	100	
 西瓜汁	45	300	








饮料	价格 (元)	体积 (ml)	性价比 (元/ml)
 橙汁	36	200	0.18
 苹果汁	16	100	0.16
 西瓜汁	45	300	0.15
 苏打水	60	600	0.10
 汽水	10	250	0.04


- 最高性价比优先
 - 解决方案

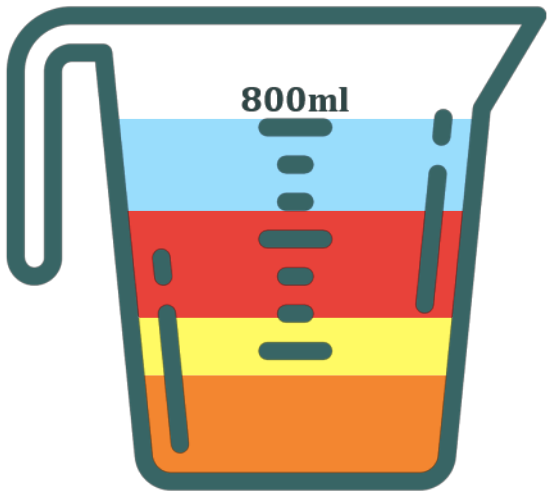
饮料	价格 (元)	体积 (ml)	总价格 (元)
 橙汁	36	200	
 苹果汁	16	100	
 西瓜汁	45	300	
 苏打水	20	200	








饮料	价格 (元)	体积 (ml)	性价比 (元/ml)
 橙汁	36	200	0.18
 苹果汁	16	100	0.16
 西瓜汁	45	300	0.15
 苏打水	60	600	0.10
 汽水	10	250	0.04

- 最高性价比优先
 - 解决方案

饮料	价格 (元)	体积 (ml)	总价格 (元)
 橙汁	36	200	117
 苹果汁	16	100	
 西瓜汁	45	300	
 苏打水	20	200	

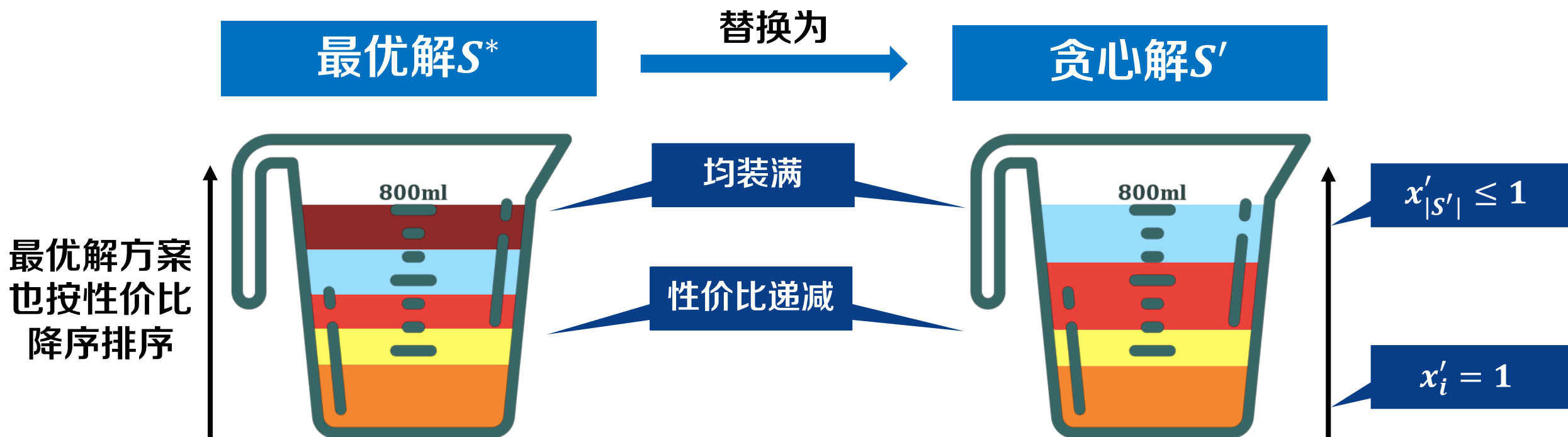


饮料	价格 (元)	体积 (ml)	性价比 (元/ml)
 橙汁	36	200	0.18
 苹果汁	16	100	0.16
 西瓜汁	45	300	0.15
 苏打水	60	600	0.10
 汽水	10	250	0.04

正确性证明



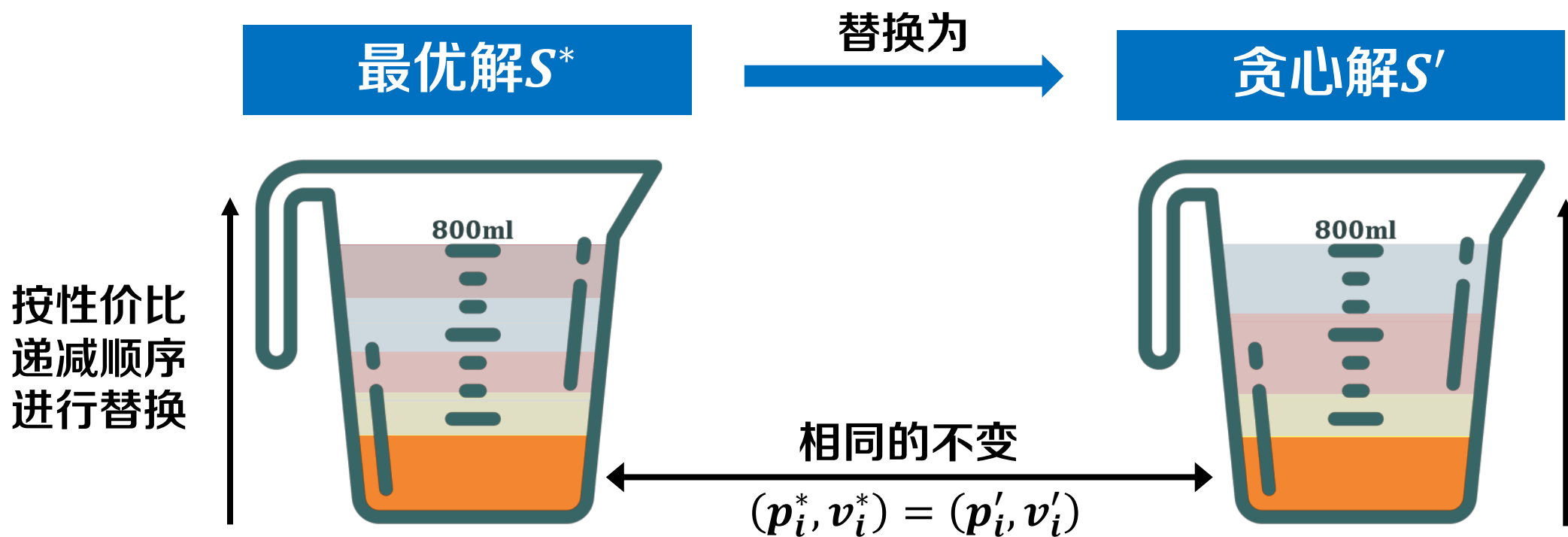
- 贪心策略：最高性价比优先
- 证明：贪心解不劣于最优解



正确性证明



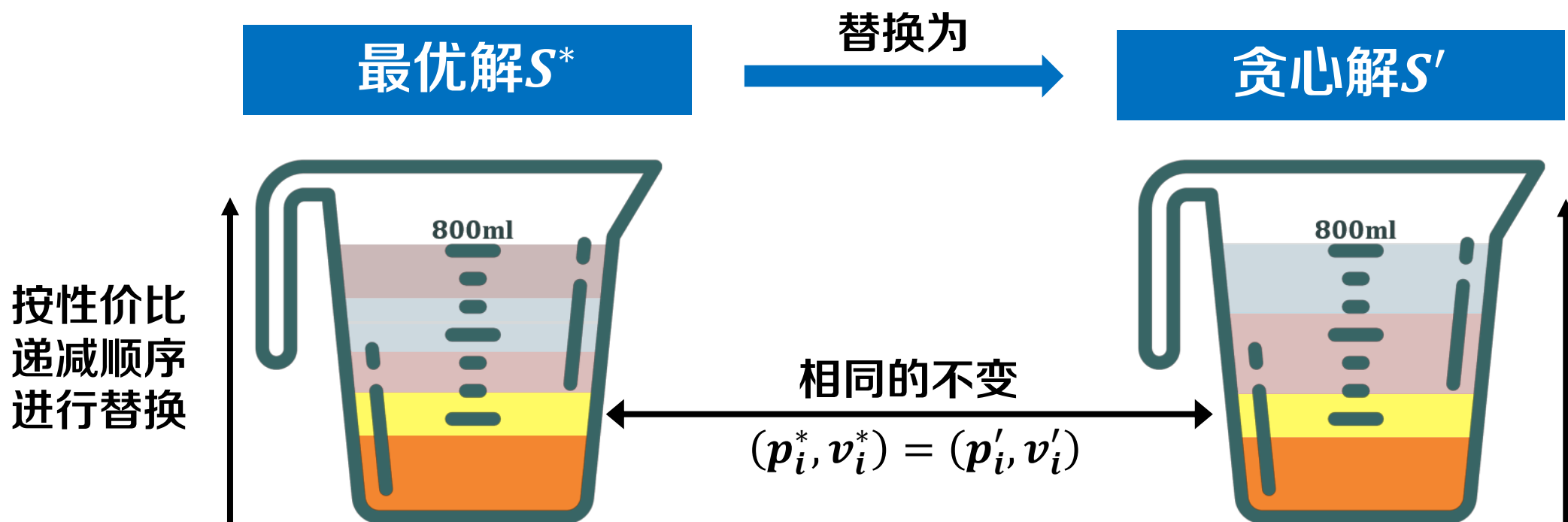
- 贪心策略：最高性价比优先
- 证明：贪心解不劣于最优解



正确性证明



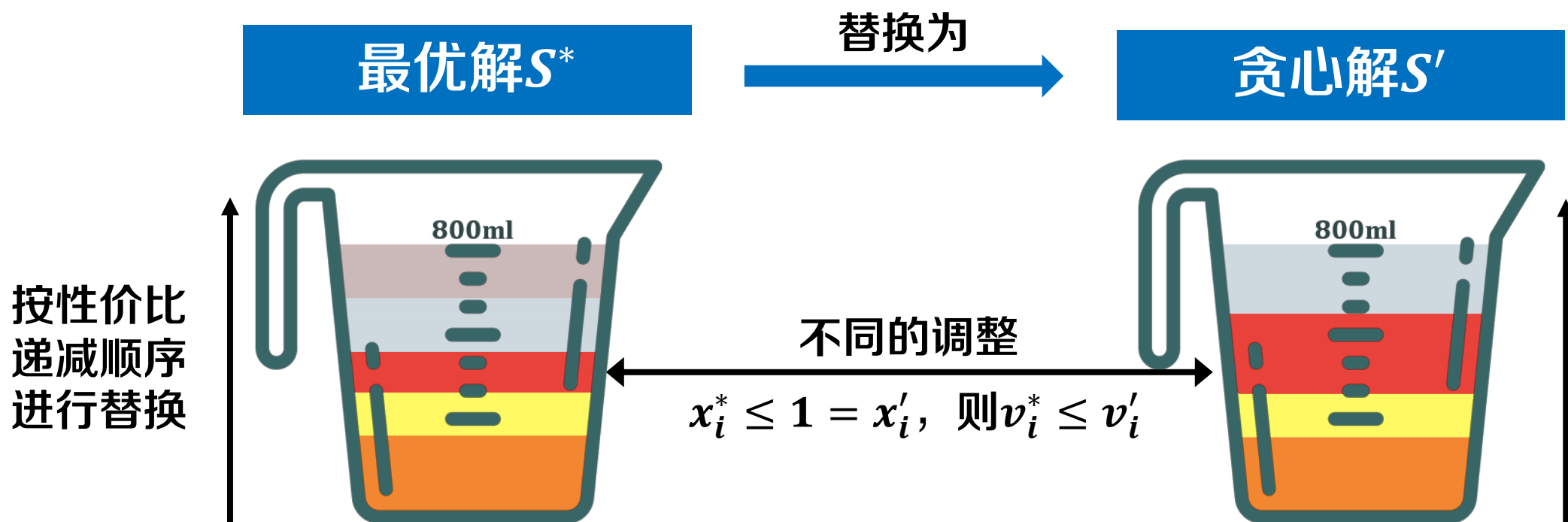
- 贪心策略：最高性价比优先
- 证明：贪心解不劣于最优解



正确性证明



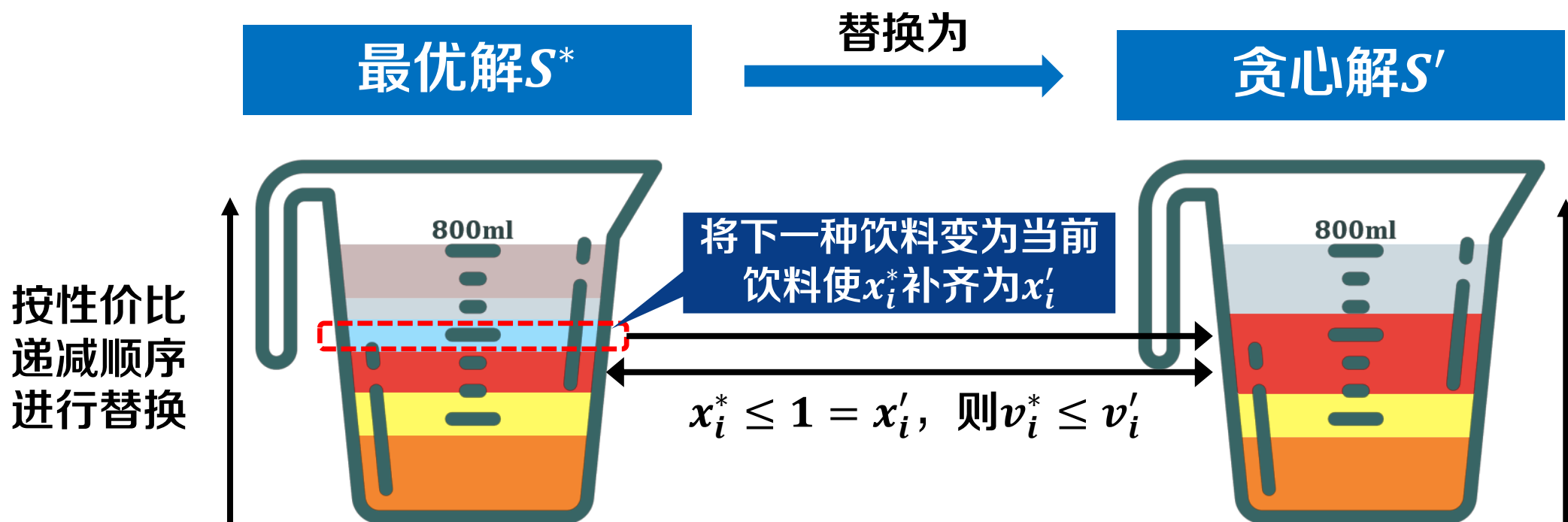
- 贪心策略：最高性价比优先
- 证明：贪心解不劣于最优解



正确性证明



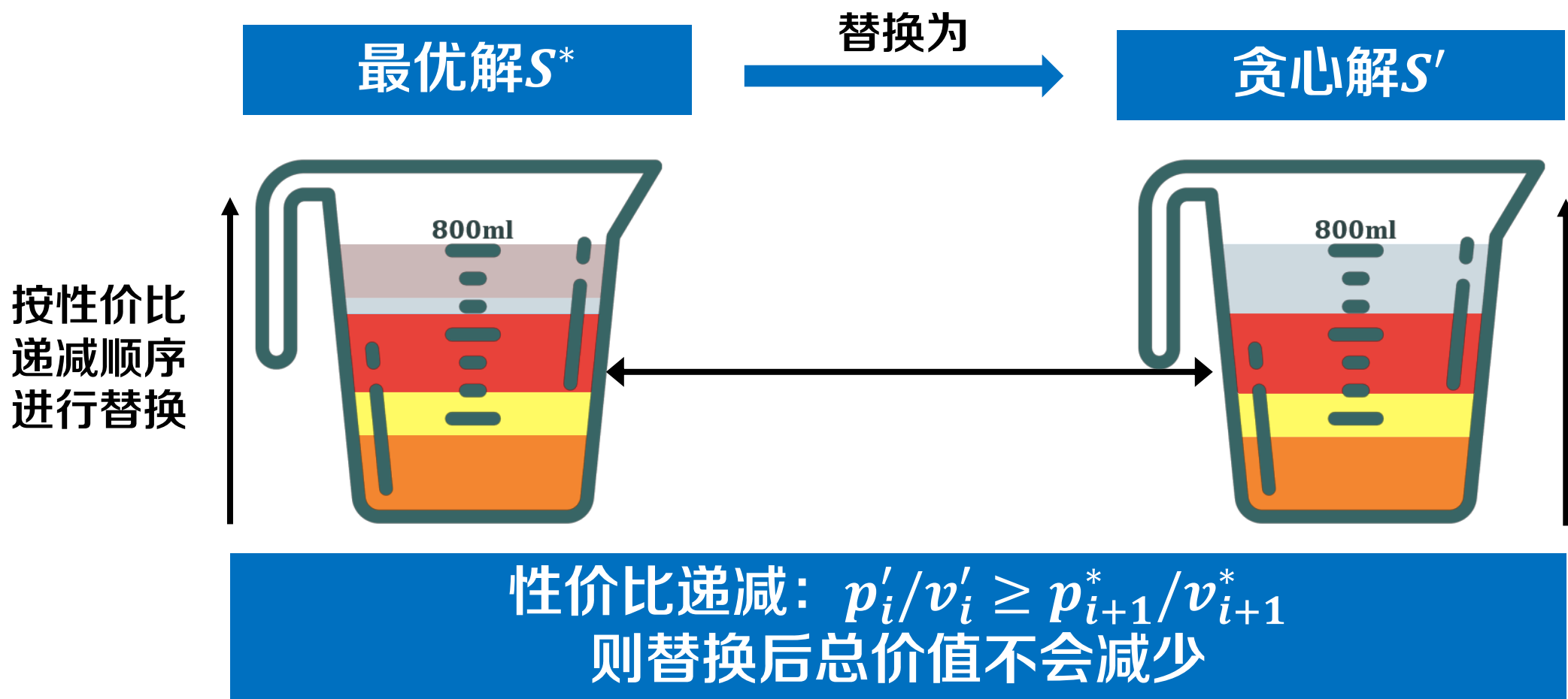
- 贪心策略：最高性价比优先
- 证明：贪心解不劣于最优解



正确性证明



- 贪心策略：最高性价比优先
- 证明：贪心解不劣于最优解



正确性证明



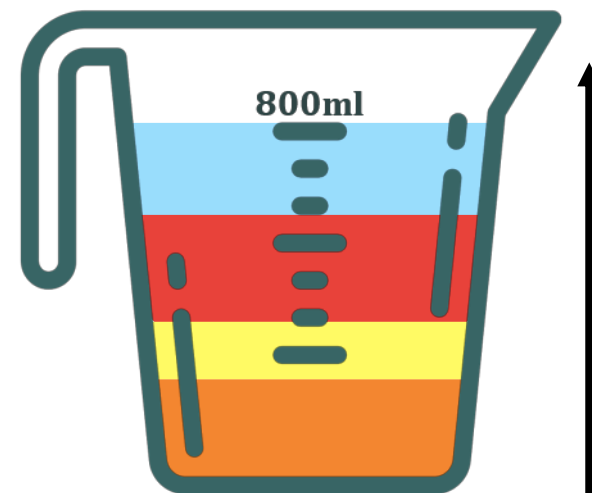
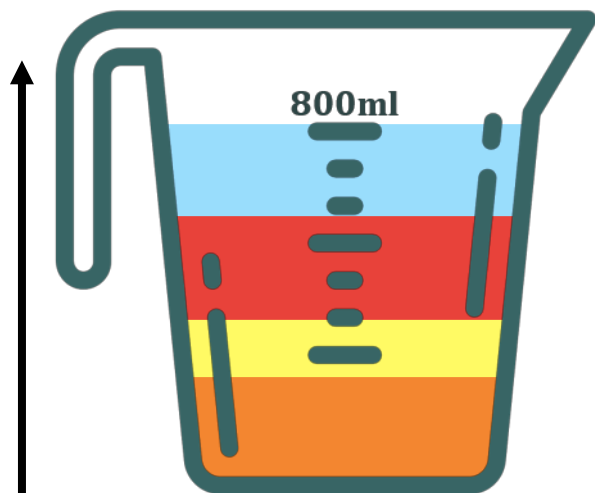
- 贪心策略：最高性价比优先
- 证明：贪心解不劣于最优解

最优解 S^*

替换为

贪心解 S'

按性价比
递减顺序
进行替换



替换后单位体积价值均不减少
故贪心解不劣于最优解

● FractionalKnapsack(n, p, v, C)

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

计算商品性价比 $Ratio[1..n]$ 并按降序排序

// $Ratio[i], p[i], v[i]$ 分别表示性价比第 i 大的商品的性价比、价格和体积

$i \leftarrow 1$

$ans \leftarrow 0$

//根据贪心策略求解

while $C > 0$ and $i \leq n$ do

 if $v[i] \leq C$ then

 选择商品 i

$ans \leftarrow ans + p[i]$

$C \leftarrow C - v[i]$

 end

 else

 选择 C 体积的商品 i

$ans \leftarrow ans + p[i] \cdot \frac{C}{v[i]}$

$C \leftarrow 0$

 end

$i \leftarrow i + 1$

end

return ans

按性价比排序, 并初始化

● FractionalKnapsack(n, p, v, C)

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

计算商品性价比 $Ratio[1..n]$ 并按降序排序

// $Ratio[i], p[i], v[i]$ 分别表示性价比第 i 大的商品的性价比、价格和体积

$i \leftarrow 1$

$ans \leftarrow 0$

// 根据贪心策略求解

while $C > 0$ and $i \leq n$ do

 if $v[i] \leq C$ then

 选择商品 i

$ans \leftarrow ans + p[i]$

$C \leftarrow C - v[i]$

 end

 else

 选择 C 体积的商品 i

$ans \leftarrow ans + p[i] \cdot \frac{C}{v[i]}$

$C \leftarrow 0$

 end

$i \leftarrow i + 1$

end

return ans

当背包未装满且商品未装完时

● FractionalKnapsack(n, p, v, C)

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

计算商品性价比 $Ratio[1..n]$ 并按降序排序

// $Ratio[i], p[i], v[i]$ 分别表示性价比第 i 大的商品的性价比、价格和体积

$i \leftarrow 1$

$ans \leftarrow 0$

// 根据贪心策略求解

while $C > 0$ and $i \leq n$ do

 if $v[i] \leq C$ then

 选择商品 i

$ans \leftarrow ans + p[i]$

$C \leftarrow C - v[i]$

 end

 else

 选择 C 体积的商品 i

$ans \leftarrow ans + p[i] \cdot \frac{C}{v[i]}$

$C \leftarrow 0$

 end

$i \leftarrow i + 1$

end

return ans

商品体积不大于容量则全部装入

- **FractionalKnapsack(n, p, v, C)**

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

计算商品性价比 $Ratio[1..n]$ 并按降序排序

// $Ratio[i], p[i], v[i]$ 分别表示性价比第 i 大的商品的性价比、价格和体积

$i \leftarrow 1$

$ans \leftarrow 0$

// 根据贪心策略求解

while $C > 0$ and $i \leq n$ do

 if $v[i] \leq C$ then

 选择商品 i

$ans \leftarrow ans + p[i]$

$C \leftarrow C - v[i]$

 end

 else

 选择 C 体积的商品 i

$ans \leftarrow ans + p[i] \cdot \frac{C}{v[i]}$

$C \leftarrow 0$

 end

$i \leftarrow i + 1$

end

return ans

否则装入部分商品填满背包

● FractionalKnapsack(n, p, v, C)

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

计算商品性价比 $Ratio[1..n]$ 并按降序排序

// $Ratio[i], p[i], v[i]$ 分别表示性价比第 i 大的商品的性价比、价格和体积

$i \leftarrow 1$

$ans \leftarrow 0$

// 根据贪心策略求解

while $C > 0$ and $i \leq n$ do

 if $v[i] \leq C$ then

 选择商品 i

$ans \leftarrow ans + p[i]$

$C \leftarrow C - v[i]$

 end

 else

 选择 C 体积的商品 i

$ans \leftarrow ans + p[i] \cdot \frac{C}{v[i]}$

$C \leftarrow 0$

 end

$i \leftarrow i + 1$

end

return ans

$O(n \log n)$

● FractionalKnapsack(n, p, v, C)

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

计算商品性价比 $Ratio[1..n]$ 并按降序排序

// $Ratio[i], p[i], v[i]$ 分别表示性价比第 i 大的商品的性价比、价格和体积

$i \leftarrow 1$

$ans \leftarrow 0$

// 根据贪心策略求解

while $C > 0$ and $i \leq n$ do

 if $v[i] \leq C$ then

 选择商品 i

$ans \leftarrow ans + p[i]$

$C \leftarrow C - v[i]$

 end

 else

 选择 C 体积的商品 i

$ans \leftarrow ans + p[i] \cdot \frac{C}{v[i]}$

$C \leftarrow 0$

 end

$i \leftarrow i + 1$

end

return ans

$O(n \log n)$

$O(n)$

● FractionalKnapsack(n, p, v, C)

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

计算商品性价比 $Ratio[1..n]$ 并按降序排序

// $Ratio[i], p[i], v[i]$ 分别表示性价比第 i 大的商品的性价比、价格和体积

$i \leftarrow 1$

$ans \leftarrow 0$

// 根据贪心策略求解

while $C > 0$ and $i \leq n$ do

 if $v[i] \leq C$ then

 选择商品 i

$ans \leftarrow ans + p[i]$

$C \leftarrow C - v[i]$

 end

 else

 选择 C 体积的商品 i

$ans \leftarrow ans + p[i] \cdot \frac{C}{v[i]}$

$C \leftarrow 0$

 end

$i \leftarrow i + 1$

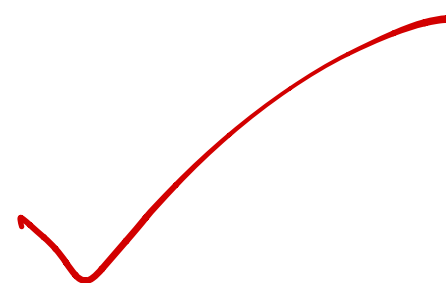
end

return ans

$O(n \log n)$

$O(n)$

时间复杂度: $O(n \log n)$

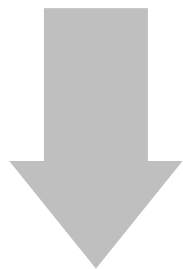


贪心策略：一般步骤



提出贪心策略

观察问题特征，构造贪心选择



证明策略正确

假设最优方案，通过替换证明

对比0-1背包问题

- 0-1背包问题

- 贪心算法结果:



啤酒
 $v = 10$

剩余
容量
 $v = 3$

总价值24

- 动态规划算法结果:



饼干
 $v = 4$




面包
 $v = 4$



牛奶
 $v = 5$

总价值28

商品	价格	体积	性价比
 啤酒	24	10	2.40
 牛奶	9	4	2.25
 饼干	9	4	2.25
 面包	10	5	2.00

背包体积为13

0-1背包问题不能使用贪心算法

对比0-1背包问题



- 问题定义:

物品不可分割



啤酒



饼干



面包



牛奶

- 解决方法:

动态规划

$P[i, c]$	$c = 0$	1	2	3	...	10	11	12	13
$i = 0$	0	0	0	0	...	0	0	0	0
1	0								
2	0								
3	0								
4	0								
5	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

物品可分割



苏打水



汽水



橙汁



苹果汁



西瓜汁

贪心策略

饮料	价格(元)	体积(ml)	性价比(元/ml)
橙汁	36	200	0.18
苹果汁	16	100	0.16
西瓜汁	45	300	0.15
苏打水	60	600	0.10
汽水	10	250	0.04