第13讲 从背包问题求解看算法
—数学建模与不同算法策略

战德臣

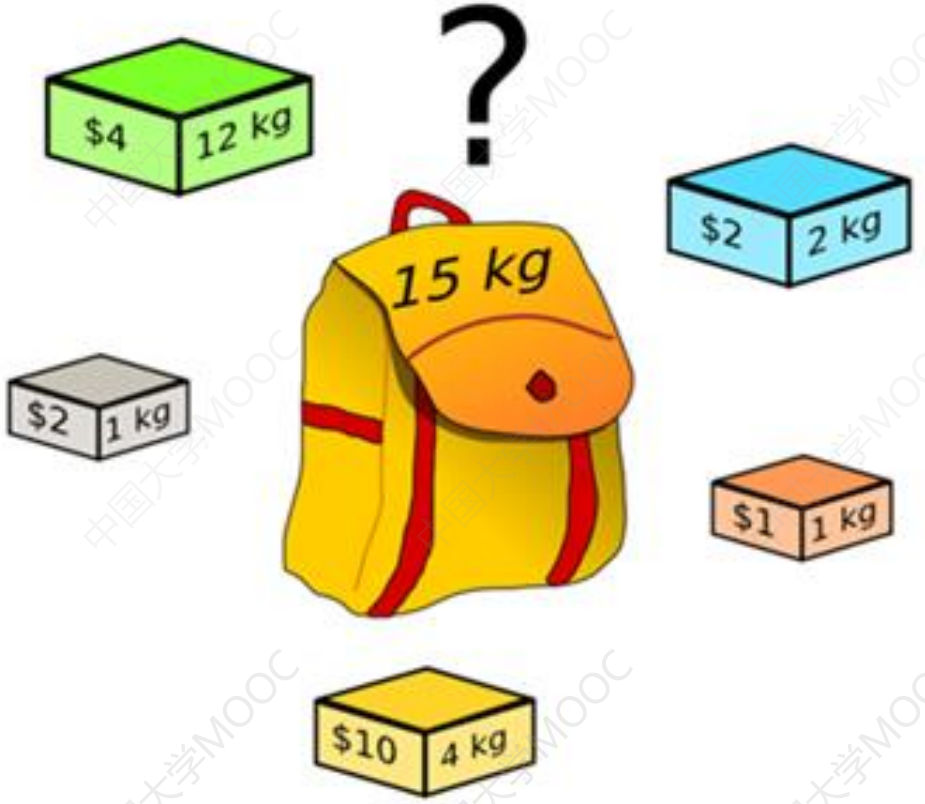
哈尔滨工业大学计算学部教学委员会主任
国家教学名师

18686783018, dechen@hit.edu.cn

一个示例

问题

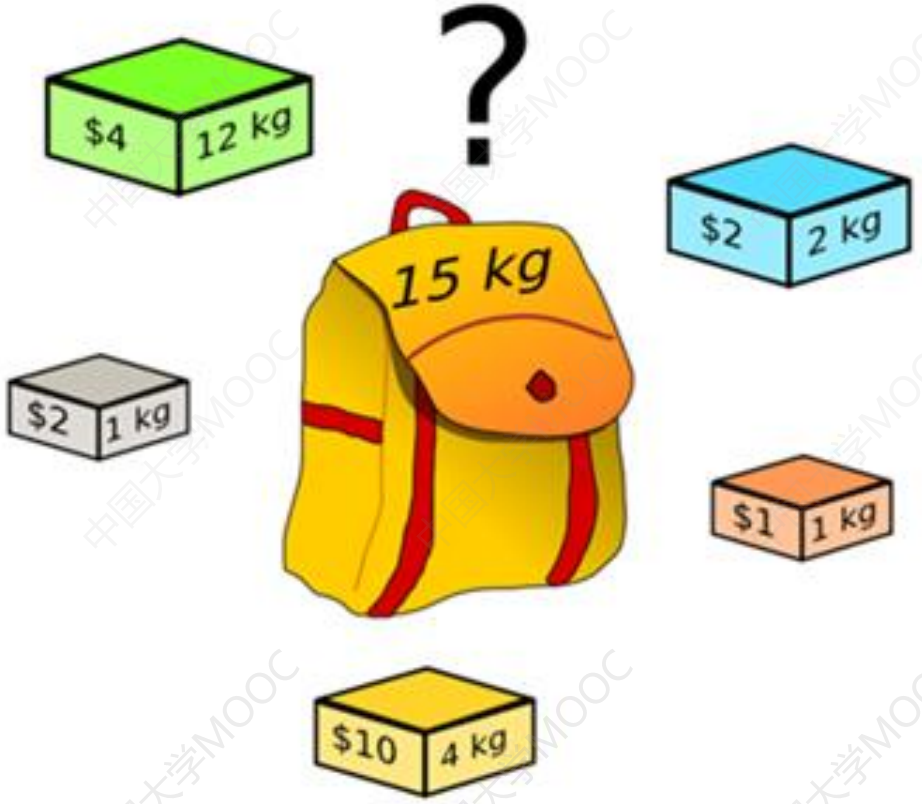
【示例】有1个背包，可装载重量最大15Kg；有5种物品，其重量和价值如图所示。问：如何装载物品，才能使该背包装载的物品总价值最大。



一个示例

基本的求解思维—【枚举-计算-验证-优化】

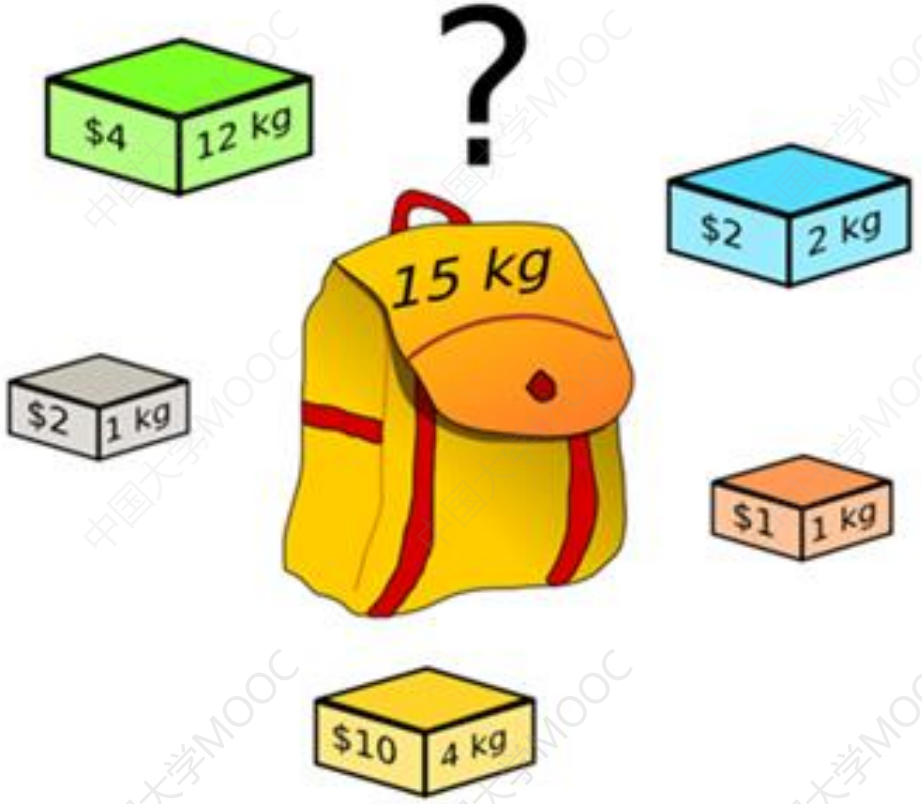
【示例】有1个背包，可装载重量最大15Kg；有5种物品，其重量和价值如图所示。问：如何装载物品，才能使该背包装载的物品总价值最大。



- **产生一种装载方案，并计算其装载物品价值**
  - ✓ 装载1种物品有5种方案
  - ✓ 装载2种物品：从5种中取出2种的所有组合=10
  - ✓ 装载3种物品：从5种中取出3种的所有组合=10
  - ✓ 装载4种物品：从5种中取出4种的所有组合=5
  - ✓ 装载5种物品有1种方案
- **产生所有装载方案，并从中选择出装载物品价值最大的方案，同时满足背包重量约束**

背包问题的数学建模

【一般性问题】有1个背包，可装载重量有限制。有n种物品，每种物品最多1件，且其重量和价值已知。怎样选择物品放入背包，才能使背包中物品价值最大?



【输入/已知】

- n种物品可用编号1,...,n表示---**（整型变量）**
- 每种物品j的重量为w_j---**（w[]数组变量）**
- 每种物品j的价值为p_j ---**（p[]数组变量）**
- 背包的总重量限制为W---**（整型变量）**

【输出/解】

- 求一组合x，$x=(x_1,x_2,...x_j,...,x_n)$。其中$x_j$=0不选物品j，=1选择物品j。 **（一个【可能解】就是一个二进制编码）**

【约束】

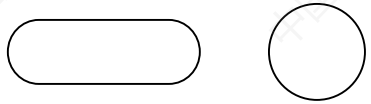
- $\sum_{j=1}^{n} w_j x_j \le W$,其中$x_j$=0或1 **（【可行解】要满足约束）**

【目标】

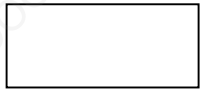
- 使$\sum_{j=1}^{n} p_j x_j$最大的x **（【最优解】要满足目标公式）**

算法表达

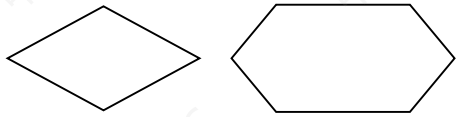
程序流程图的基本符号



圆形框表示程序的起始和结束



矩形框表示一组顺序执行的语句

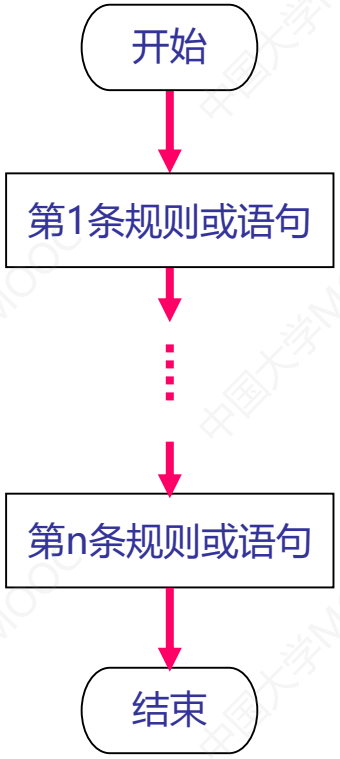


菱形框表示判断语句，
决定下一步程序的走向

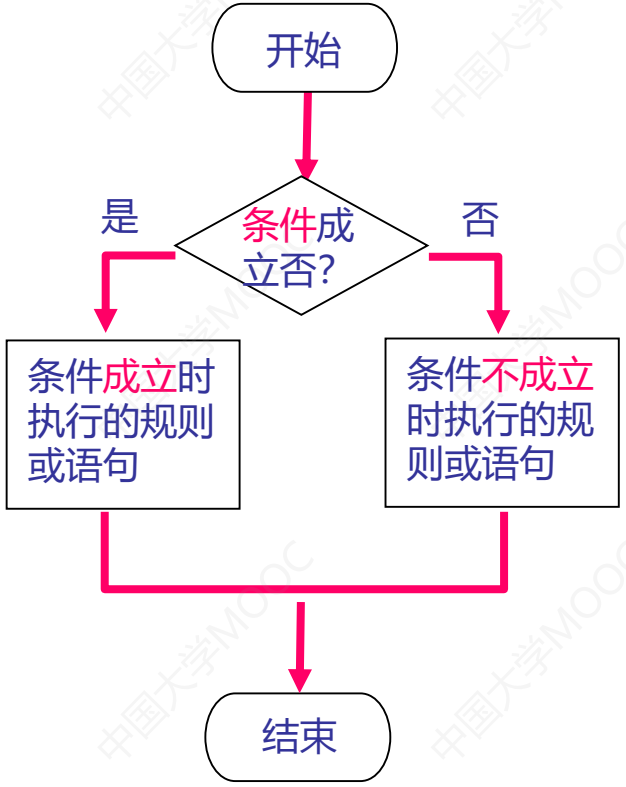
带箭头的线段表示程序的走向

算法表达

算法的三种控制结构：顺序、分支、循环（有界循环和条件循环）

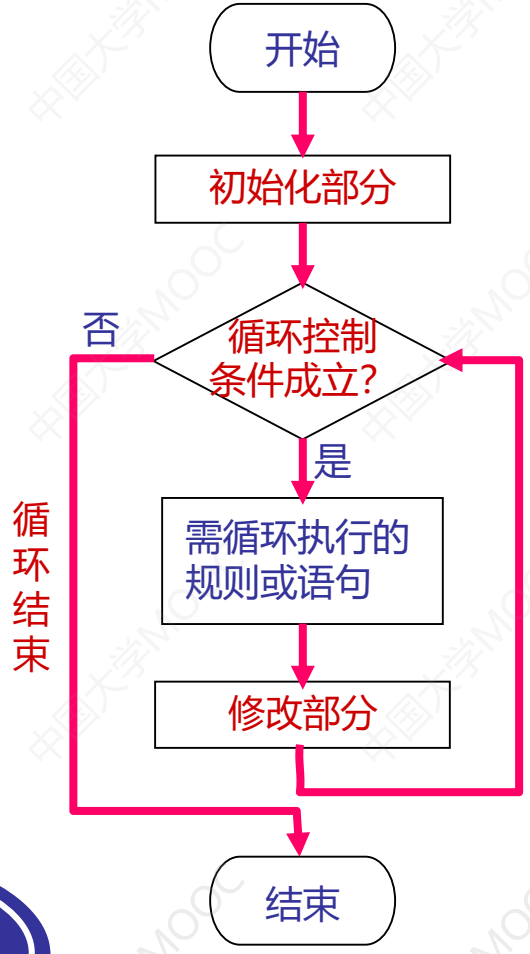


顺序结构的流程图

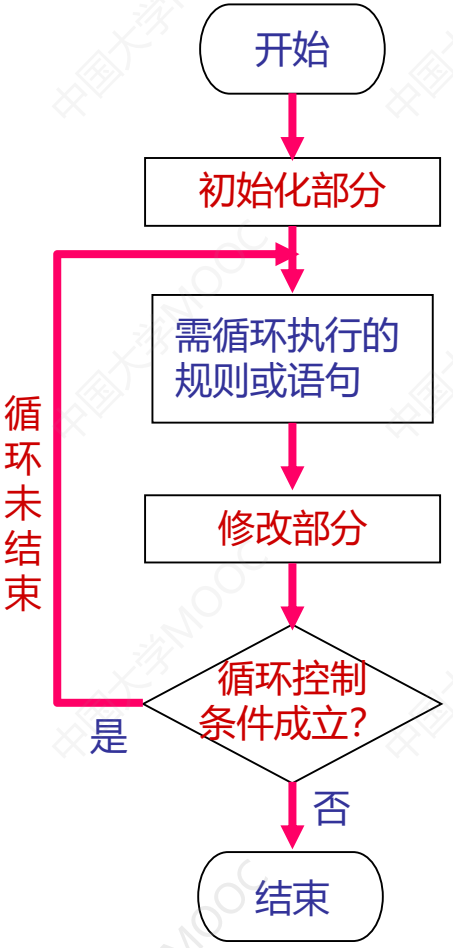


分支结构的流程图

要注意：循环结构不能无限循环，所以要有修改改变条件



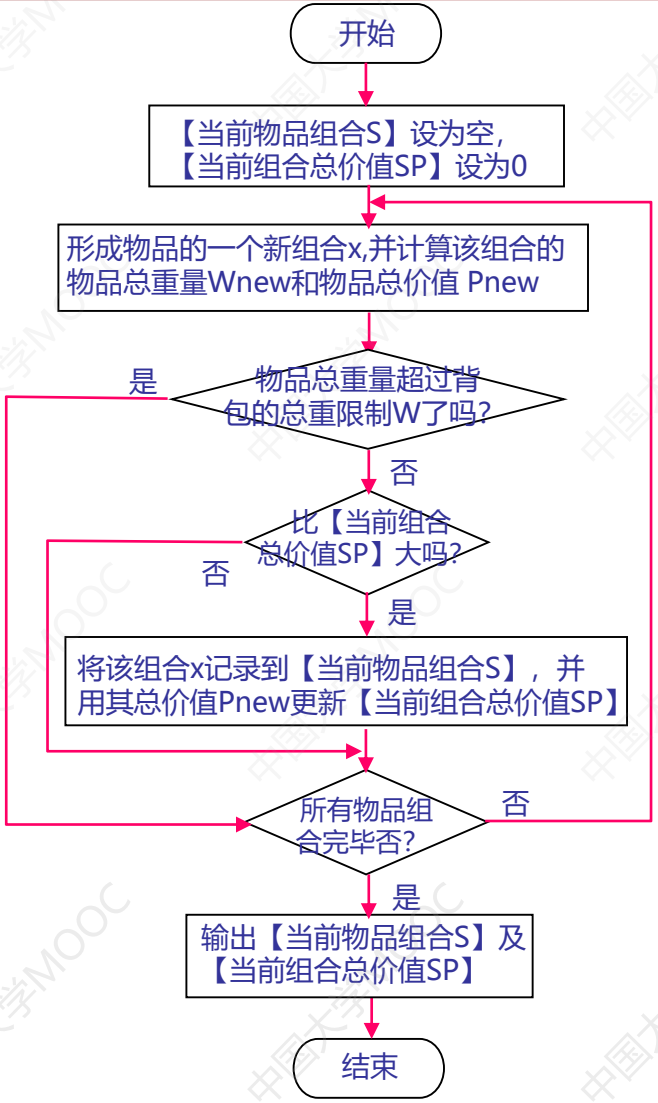
有界循环结构的流程图
(循环次数确定)



条件循环结构的流程图
(循环次数不确定)

算法表达

背包问题的求解算法【遍历算法】



- 用数组的下标i表示物品i
- 物品的一个组合表示为数组x，即：x[1]，…，x[n]。

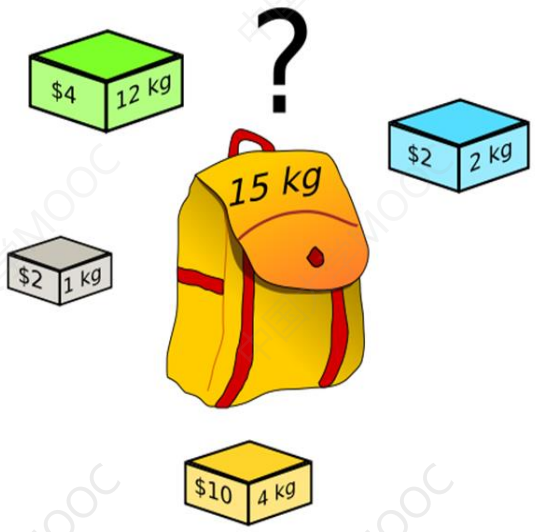
其中x[i]为0或1：为0表示不选择物品i，为1表示选择物品i

- 物品i的重量存储在w[i]中，价值存储在p[i]中
- 【当前物品组合S】与x一样的数组，S[1]，…，S[n]
- 【当前组合总价值SP】为一数值型变量
- 新组合，即产生一个不重复的x, j从1到n, 依次给x[j]赋0和1。

  如 {0,0,0,0,0} {0,0,0,0,1} {0,0,0,1,0} {0,0,0,1,1} {0,0,1,0,0} …
- 物品总重量的计算：$Wnew = \Sigma\ w[i]*x[i]$ for i=1,…,n。
- 物品总价值的计算：$Pnew = \Sigma\ p[i]*x[i]$ for i=1,…,n。
- If Wnew > W　　W为背包的总重限制
- If Pnew > SP
- SP ← Pnew
- S ← x

算法模拟

背包问题求解-遍历策略【练习】



X_4 X_3 X_2 X_1

可能解空间 $X_4X_3X_2X_1$	$\sum_{k=1}^{k=n} w_j x_j \le W$ 被选中物品的总重量	可行解空间 $X_4X_3X_2X_1$	$\sum_{k=1}^{k=n} p_j x_j$ 被选中物品的总价值	问题的最优解（精确解） $X_4X_3X_2X_1$
0000	0，满足约束	**0000**	**0**	
0001	4，满足约束	**0001**	**10**	
0010	1，满足约束	**0010**	**2**	
0011	5，满足约束	**0011**	**12**	
0100	2，满足约束	**0100**	**2**	
0101	2，满足约束	**0101**	**12**	
0110	3，满足约束	**0110**	**4**	
0111	7，满足约束	**0111**	**14**	**0111**
1000	12，满足约束	**1000**	**4**	
1001	16，不满足约束			
1010	13，满足约束	**1010**	**6**	
1011	17，不满足约束			
1100	14，满足约束	**1100**	**6**	
1101	18，不满足约束			
1110	15，满足约束	**1110**	**8**	
1111	19，不满足约束			

算法复杂性

算法的时间复杂性与空间复杂性

算法获得结果的时间有多长?

◆时间复杂性: 如果一个问题的规模是n，解这一问题的某一算法所需要的时间为T(n)，通常评估算法基本步骤的执行次数来获得T(n)，它是n的某一函数，则T(n)称为这一算法的“时间复杂性”或“时间复杂度”。

◆ “大O记法”：

✓基本参数 n——问题实例的规模

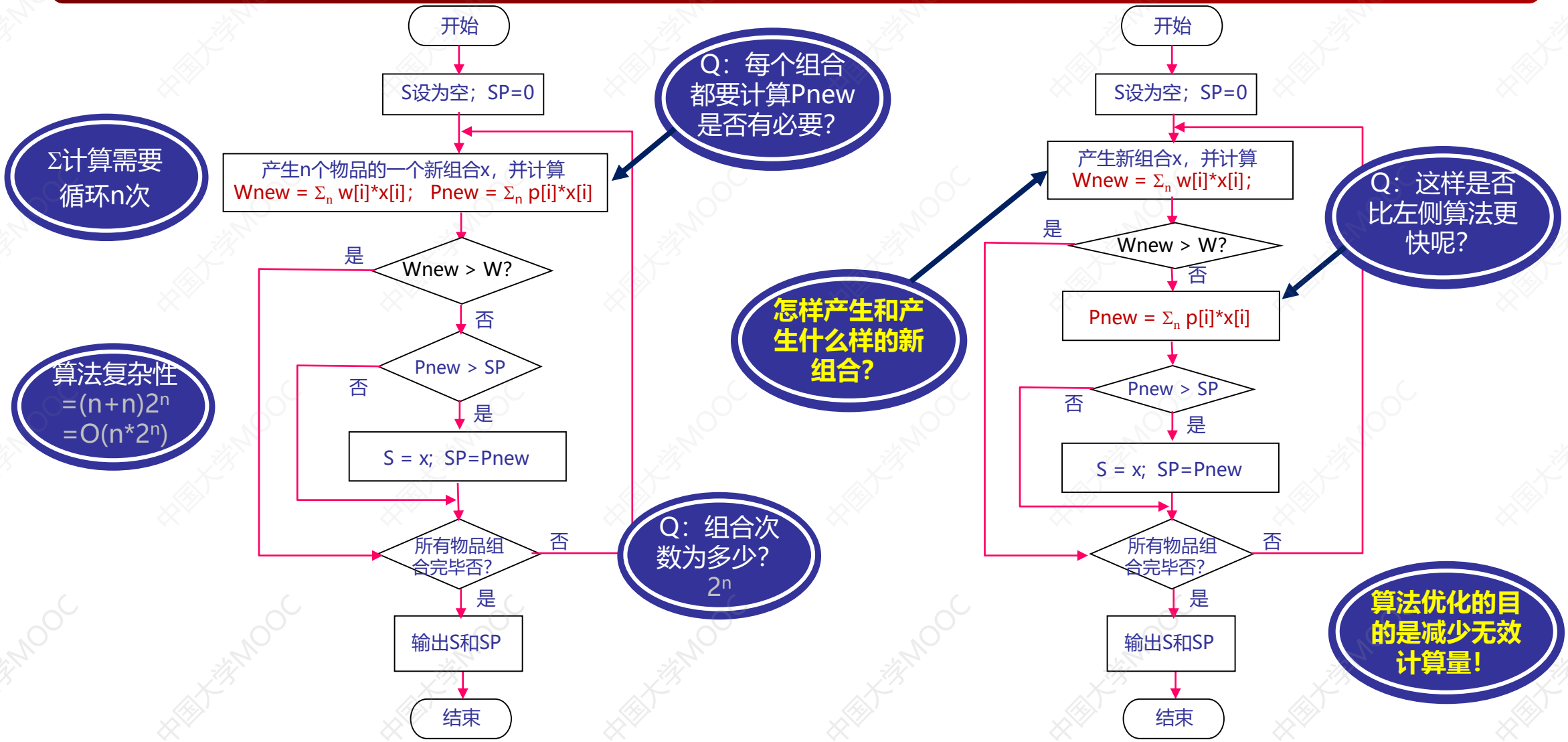
✓把复杂性或运行时间表达为n的函数。

✓ “O”表示量级 (order)，允许使用“=”代替“≈”，如$n^2+n+1 = O(n^2)$。

◆空间复杂性： 算法在执行过程中所占存储空间的大小。

算法优化

背包问题的求解算法【遍历算法—优化】



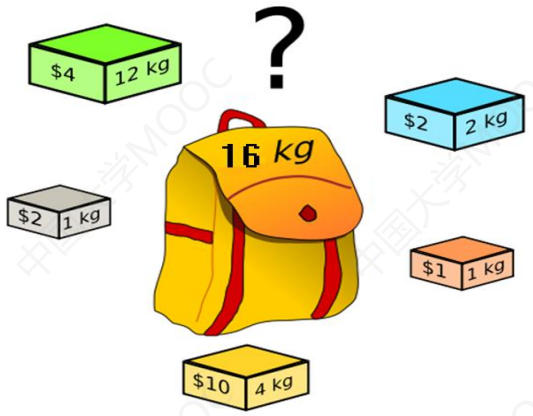
算法的不同策略

背包问题求解-贪心策略

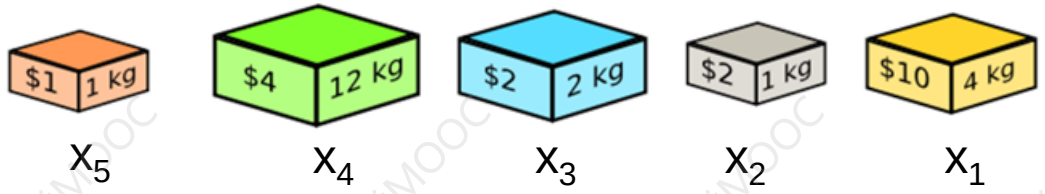
贪心算法是一种算法策略。基本思想“今朝有酒今朝醉”，**一定要做当前情况下的最好选择，否则将来可能会后悔，故名“贪心”。**

✓一个物品一个物品地选择，直到背包装满为止。

✓**每次在选择下一个物品的时候，只考虑当前情况，保证迄今为止所做出的选择是最好的。**

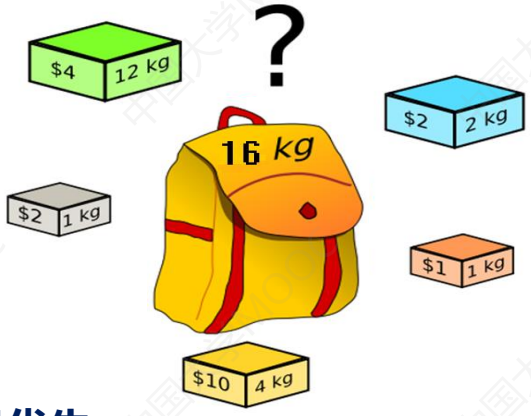


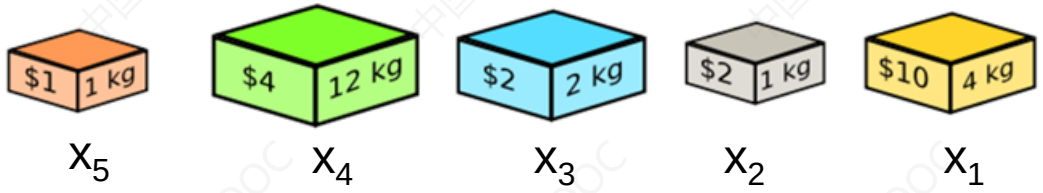
Q：贪心算法求得的解是最优解吗？不一定



算法的不同策略

背包问题求解-贪心策略【练习】





X_5 X_4 X_3 X_2 X_1

重量小的物品优先

物品选择次序	策略 1：重量小的物品优先	策略 1：被选中的重量和价值
第 1 次	X_5	1,1
第 2 次	X_2	2,3
第 3 次	X_3	4,5
第 4 次	X_1	8,15
最终结果	$X_5X_4X_3X_2X_1$ =10111	8, 15

价值高的物品优先

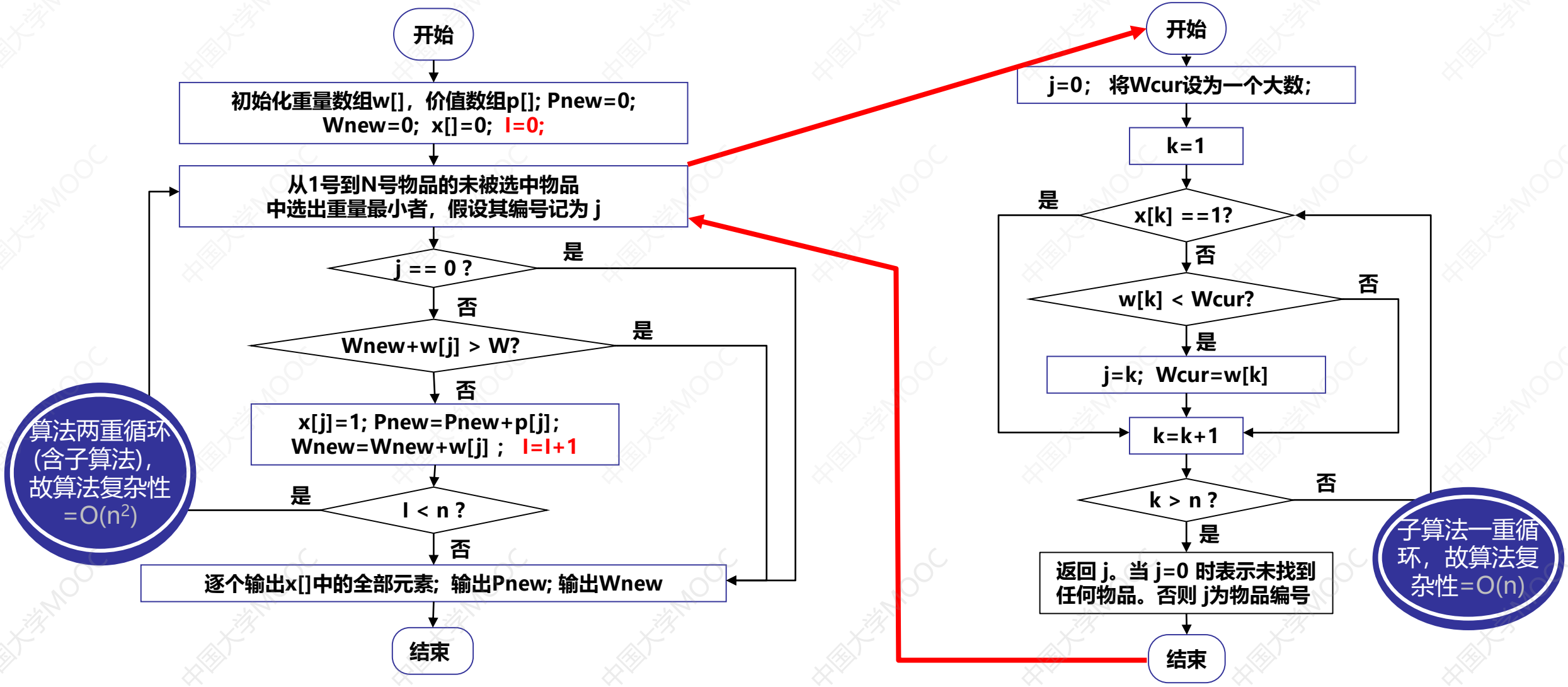
物品选择次序	策略 2：价值高的物品优先	策略 2：被选中的重量和价值
第 1 次	X_1	4,10
第 2 次	X_4	16,14
第 3 次		
第 4 次		
最终结果	$X_5X_4X_3X_2X_1$ =01001	16, 14

性价比高的物品优先

物品选择次序	策略 3：性价比高的物品优先	策略 3：被选中的重量和价值
第 1 次	X_1	4,10
第 2 次	X_2	5,12
第 3 次	X_3	7,14
第 4 次	X_5	8,15
最终结果	$X_5X_4X_3X_2X_1$ =10111	8, 15

算法表达

背包问题的贪心算法求解（重量小的物品优先）



不同的问题

背包问题分类

有n种物品，物品j的重量为w_j，价格为p_j, 假定所有物品的重量和价格都是非负的。

背包所能承受的最大重量为W。

- 如果限定每种物品只能选择0个或1个，则为**0-1背包问题**
- 如果限定物品j最多只能选择b_j个，则为**有界背包问题**或**多重背包问题**
- 如果不限定每种物品的数量，则为**无界背包问题**或**完全背包问题**

背包问题是一类重要的问题，研究者提出了很多算法，如贪心算法、动态规划算法等，同学可自主研究之。

完成数学建模才能精准抓住应用问题的本质

不同的应用问题，可能是相同的算法问题

建模要素 \ 应用问题	股票投资组合问题	产品品种的组合问题	背包问题（有界背包）
问题表述	有n种股票可供选择，每种股票j最多买b_j股，怎样一个投资组合，才能使股票收益最大	有n种产品可供销售，每种产品j最多可销售b_j个，怎样一个产品组合，才能使产品销售价值最大	有n种物品，每种物品j最多选择b_j个，怎样选择物品放入背包，才能使背包中物品价值最大
输入/已知	n种股票可用编号1,...,n表示 每种股票j的价格为w_j。 每种股票j的收益为p_j。 可用于购买股票的资本为W	n种产品。可用编号1,...,n表示 每种产品j的可销售数量为w_j。 每种产品j的价格为p_j。 市场可接受产品的总数量W	n种物品。可用编号1,...,n表示 每种物品j的重量为w_j。 每种物品j的价值为p_j。 背包的总重量为W
输出/结果	求一组合$x=(x_1,x_2,..x_j,..,x_n)$。 其中股票j选择x_j股。$x_j=0,...,b_j$为购买股数	求一组合$x=(x_1,x_2,..x_j,..,x_n)$。 其中产品j选择x_j个。$x_j=0,...,b_j$为销售数量	求一组合$x=(x_1,x_2,..x_j,..,x_n)$。 其中物品j选择x_j个。$x_j=0,...,b_j$为选择个数
约束	$\sum_{k=1}^{k=n} w_j x_j \le W$,其中$x_j=0,...,b_j$	$\sum_{k=1}^{k=n} w_j x_j \le W$,其中$x_j=0,...,b_j$	$\sum_{k=1}^{k=n} w_j x_j \le W$,其中$x_j=0,...,b_j$
目标	x要使得 $\sum_{k=1}^{k=n} p_j x_j$ 最大	x要使得 $\sum_{k=1}^{k=n} p_j x_j$ 最大	x要使得 $\sum_{k=1}^{k=n} p_j x_j$ 最大

从背包问题求解看算法

小结

- **问题建模**：输入【已知】，输出【可能解的形式】，约束【可行解的条件】，目标【最优解的条件】
- 用自然数1...n表达现实的具体对象，具体对象的相关特征数据就可用带下标的变量表达，如第i个对象的特征就可用p_i,t_i等来表达（对应到算法/程序就是p[i],t[i]等）。**通过自然数及其下标将不同类型的数据关联起来，是算法类问题抽象的重要方法。**
- 输出可用一个自然数序列来表示**$x=(x_1,x_2,...,x_n)$**，此时的i表示为对象i，而x_i则为对象i的某一特征值（典型的是0选择和1不选），在算法/程序中可用一维数组x[]来表达。这是表达**可能解空间**的典型形式。
- 衡量算法的执行时间长短，可用**算法复杂度**来衡量。
- 求解问题可有不同的**算法策略**，不同算法策略得到的解可能是不同的，算法复杂度也可能有很大的差异。
- 算法研究的基础就是**枚举-计算-验证**，然后对算法做**优化**，主要目标是**减少无效计算量**。