

# 组合优化理论

## 第8章 装箱问题

主讲人：陈安龙

# 第8章 装箱问题

## § 1 装箱问题的描述

## § 2 装箱问题的最优解值下界

## § 3 装箱问题的近似算法

# 第8章 装箱问题

装箱问题 (*Bin Packing*) 是一个经典的组合优化问题, 有着广泛的应用, 在日常生活中也屡见不鲜.

## § 1 装箱问题的描述

设有许多具有同样结构和负荷的箱子  $B_1, B_2, \dots$  其数量足够供所达到目的之用. 每个箱子的负荷 (可为长度、重量 *etc.*) 为  $C$ , 今有  $n$  个负荷为  $w_j$ ,  $0 < w_j < C$   $j = 1, 2, \dots, n$  的物品  $J_1, J_2, \dots, J_n$  需要装入箱内.

**装箱问题:**

是指寻找一种方法, 使得能以最小数量的箱子数将  $J_1, J_2, \dots, J_n$  全部装入箱内.

## § 1 装箱问题的描述

由于  $w_i < C$ ，所以  $BP$  的最优解的箱子数不超过  $n$ 。

设  $y_i = \begin{cases} 1 & \text{箱子 } B_i \text{ 被使用} \\ 0 & \text{否则} \end{cases} \quad i = 1 \sim n;$

$x_{ij} = \begin{cases} 1 & \text{物品 } J_j \text{ 放入箱子 } B_i \text{ 中} \\ 0 & \text{否则} \end{cases} \quad i, j = 1 \sim n.$

则装箱问题的整数线性规划模型为：

$$\min \quad z = \sum_{i=1}^n y_i$$

$$(BP) \quad s.t. \quad \sum_{j=1}^n w_j x_{ij} \leq C y_i \quad i = 1 \sim n \quad (1)$$

$$\sum_{i=1}^n x_{ij} = 1 \quad j = 1 \sim n \quad (2)$$

$$y_i = 0 \text{ or } 1, \quad x_{ij} = 0 \text{ or } 1 \quad i, j = 1 \sim n.$$

约束条件 (1) 表示：  
一旦箱子  $B_i$  被使用，  
放入  $B_i$  的物品总负  
荷不超过  $C$ ；

约束条件 (2) 表示：  
每个物品恰好放入一  
个箱子中。

上述装箱问题是这类问题最早被研究的，也是提法上最简单的问题，称为一维装箱问题，但  $BP \in NPC$ .

### 装箱问题的其他一些提法：

- 1、在装箱时，不仅考虑长度，同时考虑重量或面积、体积 *etc* . 即二维、三维、...装箱问题；
- 2、对每个箱子的负荷限制不是常数  $C$ ；而是  $C_i, i = 1 \sim n$ .
- 3、物品  $J_1, J_2, \dots, J_n$  的负荷事先并不知道，来货是随到随装；即在线 (*On-Line*) 装箱问题；
- 4、由于场地的限制，在同一时间只能允许一定数量的箱子停留现场可供使用, *etc* .

## **BP 的应用举例:**

- 1、下料问题** 轧钢厂生产的线材一般为**同一长度**，而用户所需的线材则可能具有各种不同的尺寸，如何根据用户提出的要求，用最少的线材**截出所需的定货**；
- 2、二维 BP** 玻璃厂生产出**长宽一定的大平板玻璃**，但用户所需玻璃的长宽可能有许多差异，如何根据用户提出的要求，用最少的平板玻璃**截出所需的定货**；
- 3、计算机的存贮问题** 如要把大小不同的共  $10\text{ MB}$  的文件拷贝到磁盘中去，而每张磁盘的容量为  $1.44\text{ MB}$ ，已知每个文件的字节数不超过  $1.44\text{ MB}$ ，而且**一个文件不能分成几部分存贮**，如何用最少的磁盘张数完成。
- 4、生产流水线的平衡问题** 给定流水节拍  $C$ ，如何设置最少的工作站，（按一定的紧前约束）沿着流水线将任务分配到各工作站上。称为**带附加优先约束的 BP**。  
**BP 是容量限制的工厂选址问题的特例之一。**

## § 2 装箱问题的最优解值下界

由于  $BP$  是  $NPC$  问题，所以求解考虑 一是尽可能改进简单的穷举搜索法，减少搜索工作量。如： 分支定界法；二是启发式（近似）算法。

$$\begin{aligned} \min \quad & z = \sum_{i=1}^n y_i \\ (BP) \quad s.t. \quad & \sum_{j=1}^n w_j x_{ij} \leq C y_i \quad i = 1 \sim n \end{aligned} \quad (1)$$

$$\sum_{i=1}^n x_{ij} = 1 \quad j = 1 \sim n \quad (2)$$

$$y_i = 0 \text{ or } 1, \quad x_{ij} = 0 \text{ or } 1 \quad i, j = 1 \sim n.$$

**显然**  $x_{ii} = 1, \quad x_{ij} = 0 \quad (i \neq j), \quad y_i = \frac{w_i}{C}, \quad i, j = 1 \sim n.$

是松弛问题的最优解，最优值为：
$$z_{opt} = \frac{1}{C} \sum_{i=1}^n w_i$$

## § 2 装箱问题的最优解值下界

**Theorem 8.1** *BP* 最优值的一个下界为  $L_1 = \left\lceil \frac{1}{C} \sum_{i=1}^n w_i \right\rceil$

$\lceil a \rceil$  表示不小于  $a$  的最小整数.

### **Theorem 8.2**

对于 *BP* 的任一实例  $I$ , 设  $a$  是任意满足  $w = \min \{ w_j \mid w_j \in I \}$

$0 \leq a \leq \frac{C}{2}, a \leq w$  的整数

记  $I_1 = \{ \text{物品 } j \mid w_j > C - a \}, I_2 = \{ \text{物品 } j \mid C - a \geq w_j > \frac{C}{2} \},$

$I_3 = \{ \text{物品 } j \mid \frac{C}{2} \geq w_j \geq a \},$

则  $L(a) = |I_1| + |I_2| + \max \left\{ 0, \left\lceil \frac{1}{C} \left( \sum_{j \in I_3} w_j - (|I_2|C - \sum_{j \in I_2} w_j) \right) \right\rceil \right\}$

是最优解的一个下界.

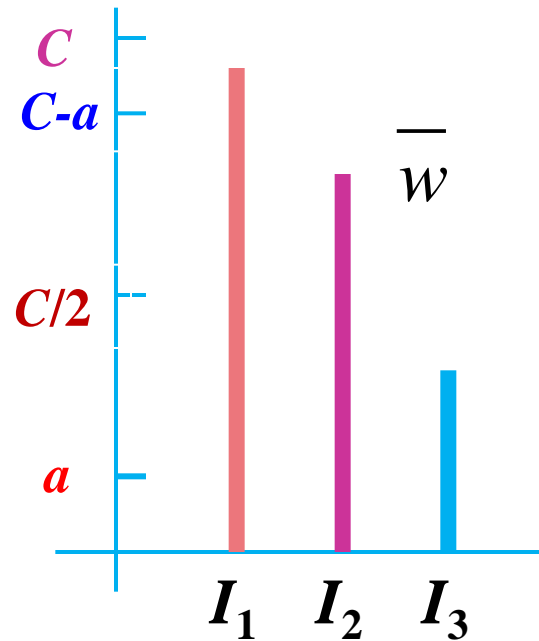


**Proof:** 仅考虑对  $I_1, I_2, I_3$  中物品的装箱.

$\because I_1 \cup I_2$  中物品的长度大于  $C/2$ ,  
每个物品需单独放入一个箱子,  
这就需要  $|I_1| + |I_2|$  个箱子.

又  $\because I_3$  中每个物品长度至少为  $a$ ,

$\therefore$  它不能与  $I_1$  中的物品共用箱子, 但可能与  $I_2$  中的物品共用箱子, 由于放  $I_2$  中物品的  $|I_2|$  个箱子的剩余总长度为  $\bar{C} = |I_2|C - \sum_{j \in I_2} w_j$



在最好的情形下,  $\bar{C}$  被  $I_3$  中的物品全部充满, 故剩

下总长度  $\bar{w} = \sum_{j \in I_3} w_j - \bar{C}$  将另外至少  $\left\lceil \frac{\bar{w}}{C} \right\rceil$  个附加的箱子.

$\bar{w}$  可能为负数

$$\therefore L(a) = |I_1| + |I_2| + \max \left\{ 0, \left\lceil \frac{1}{C} \left( \sum_{j \in I_3} w_j - (|I_2|C - \sum_{j \in I_2} w_j) \right) \right\rceil \right\}$$

是最优解的一个下界。

$$\text{问 } L(a) \geq L_1 = \left\lceil \frac{1}{C} \sum_{i=1}^n w_i \right\rceil ?$$

未必！ 如  $(w_j < a, j = 1 \sim n)$

## § 2 装箱问题的最优解值下界

**Corollary 8.1** 记  $L_2 = \max \left\{ L(a) \mid 0 \leq a \leq \frac{C}{2}, a \text{ 为整数} \right\}$

则  $L_2$  是装箱问题的最优解的一个下界, 且  $L_2 \geq L_1$ .

**Proof:**  $L_2$  为最优解的下界是显然的.

(若证明  $L(0) \geq L_1$ , 则可得  $L_2 \geq L_1$ )

$$L(a) = |I_1| + |I_2| + \max \left\{ 0, \left\lceil \frac{1}{C} \left( \sum_{j \in I_3} w_j - (|I_2|C - \sum_{j \in I_2} w_j) \right) \right\rceil \right\}$$

当  $a = 0$  时,  $I_1 = \emptyset$ ,  $I_2 \cup I_3$  是所有物品.

$$\begin{aligned} L(0) &= 0 + |I_2| + \max \left\{ 0, \left\lceil \frac{1}{C} \left( \sum_{j=1}^n w_j - |I_2|C \right) \right\rceil \right\} \\ &= |I_2| + \max \{ 0, L_1 - |I_2| \} = \max \{ |I_2|, L_1 \} \geq L_1 \end{aligned}$$

$$\therefore L_2 \geq L(0) \geq L_1$$

## § 3 装箱问题的近似算法

### 一、*NF (Next Fit)* 算法

设物品  $J_1, J_2, \dots, J_n$  的长度分别为  $w_1, w_2, \dots, w_n$   
箱子  $B_1, B_2, \dots$  的长均为  $C$ ，按物品给定的顺序装箱。

先将  $J_1$  放入  $B_1$ ，如果  $w_1 + w_2 \leq C$  则将  $J_2$  放入  $B_1 \dots$

如果  $w_1 + w_2 + \dots + w_j \leq C$  而  $w_1 + w_2 + \dots + w_j + w_{j+1} > C$

则  $B_1$  已放入  $J_1, J_2, \dots, J_j$ ，将其关闭，将  $J_{j+1}$  放入  $B_2$ 。

同法进行，直到所有物品装完为止。计算复杂性为  $O(n)$ 。

- 特点：
- 1、按物品给定的顺序装箱；
  - 2、关闭原则。

对当前要装的物品  $J_i$  只关心具有最大下标的已使用过的箱子  $B_j$  能否装得下？

能. 则  $J_i$  放入  $B_j$ ； 否. 关闭  $B_j$ ， $J_i$  放入新箱子  $B_{j+1}$ 。

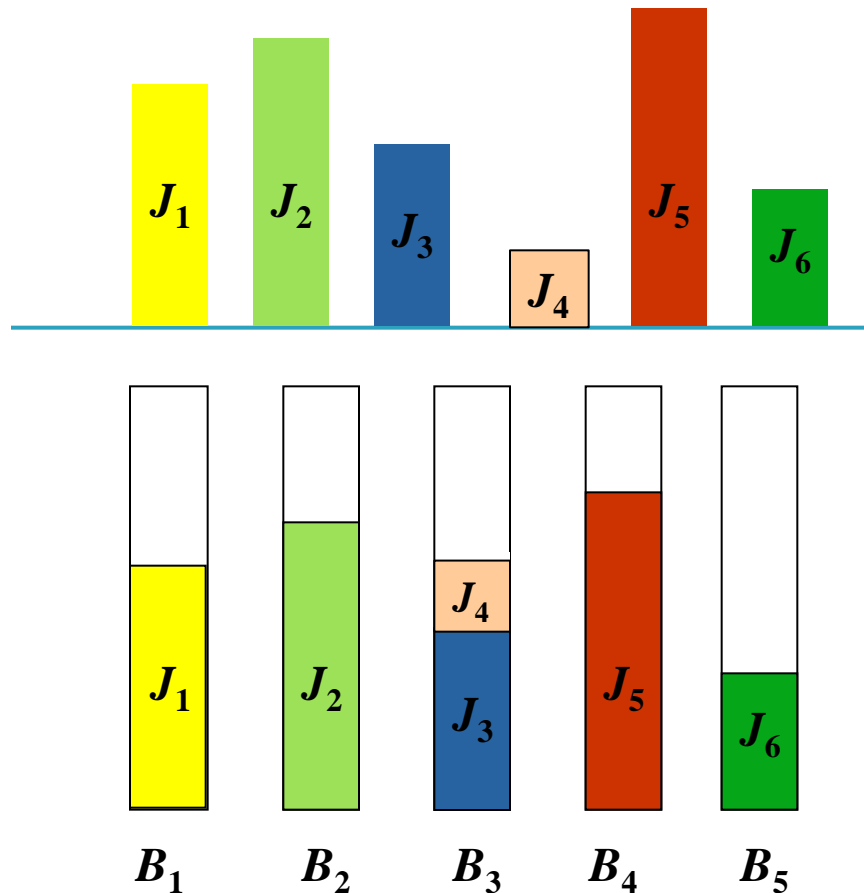
## § 3 装箱问题的近似算法

**Example 1**  $I : C = 10$

**Solution :**

首先, 将  $J_1$  放入  $B_1$ ;  
由于  $J_2$  在  $B_1$  中放不下, 所以关闭  $B_1$ , 将  $J_2$  放入  $B_2$ ,  
 $J_3$  在  $B_2$  中放不下(不考虑  $B_1$  是否能装), 所以关闭  $B_2$   
将  $J_3$  放入  $B_3$ , ...

物品	$J_1$	$J_2$	$J_3$	$J_4$	$J_5$	$J_6$
$w_j$	6	7	4	2	8	3



$\therefore$  解为:  $x_{11} = x_{22} = x_{33} = x_{34} = x_{45} = x_{56} = 1$  其余为零,

**Theorem 8.3** 
$$\frac{z_{NF}(I)}{z_{opt}(I)} \leq 2$$

**Proof:** 设  $I$  为任一实例,  $z_{opt}(I) = k$ . (要证  $z_{NF}(I) \leq 2k$ )

显然, 由  $k = z_{opt}(I) \geq \frac{1}{C} \sum_{i=1}^n w_i$  得  $\sum_{i=1}^n w_i \leq Ck$

**反证** 如果  $z_{NF}(I) > 2k$ , 则对任意  $i = 1, 2, \dots, k$   
 由于起用第  $2i$  个箱子是因为第  $2i-1$  个箱子放不下第  $2i$   
 个箱子中第一个物品, 因此这两个箱子中物品的总长度  
 大于  $C$ , 所以前  $2k$  个箱子中物品的总长度大于  $Ck$ .

这与  $\sum_{i=1}^n w_i \leq Ck$  矛盾.  $\therefore \frac{z_{NF}(I)}{z_{opt}(I)} \leq 2$ .

## § 3 装箱问题的近似算法

$$I : C = 10$$

### 二、*FF (First Fit)* 算法

用 *NF* 算法装箱, 当放入

物品	$J_1$	$J_2$	$J_3$	$J_4$	$J_5$	$J_6$
$w_j$	6	7	4	2	8	3

$J_3$  时, 仅看  $B_2$ , 是否能放入, 因  $B_1$  已关闭, 但事实上,  $B_1$  此时是能放得下  $J_3$  的.

设物品  $J_1, J_2, \dots, J_n$  的长度分别为  $w_1, w_2, \dots, w_n$ , 箱子  $B_1, B_2, \dots$  的长均为  $C$ , 按物品给定的顺序装箱.

先将  $J_1$  放入  $B_1$ , 若  $w_1 + w_2 \leq C$ , 则  $J_2$  放入  $B_1$ , 否则,  $J_2$  放入  $B_2$ ; 若  $J_2$  已放入  $B_2$ , 对于  $J_3$  则依次检查  $B_1, B_2$ , 若  $B_1$  能放得下, 则  $J_3$  放入  $B_1$ , 否则查看  $B_2$ , 若  $B_2$  能放得下, 则  $J_3$  放入  $B_2$ , 否则启用  $B_3$ ,  $J_3$  放入  $B_3$ .



一般地,  $J_1, \dots, J_j$  已放入  $B_1, \dots, B_i$  箱子, 对于  $J_{j+1}$ , 则依次检查  $B_1, B_2, \dots, B_i$ , 将  $J_{j+1}$  放入首先找到的能放得下的箱子, 如果都放不下, 则启用箱子  $B_{i+1}$ , 将  $J_{j+1}$  放入  $B_{i+1}$ , 如此继续, 直到所有物品装完为止.

- 特点:
- 1、按物品给定的顺序装箱;
  - 2、对于每个物品  $J_j$  总是放在能容纳它的具有最小标号的箱子.

计算复杂性为  $O(n \log n)$ .

## § 3 装箱问题的近似算法

**Theorem 8.4**  $\frac{z_{FF}(I)}{z_{opt}(I)} \leq \frac{7}{4} .$

**Theorem 8.5**

对任意实例  $I$  ,  $z_{FF}(I) \leq \frac{17}{10} z_{opt}(I) + 1$

而且存在  $z_{opt}(I)$  任意大的实例  $I$  , 使  $z_{FF}(I) \geq \frac{17}{10} (z_{opt}(I) - 1)$

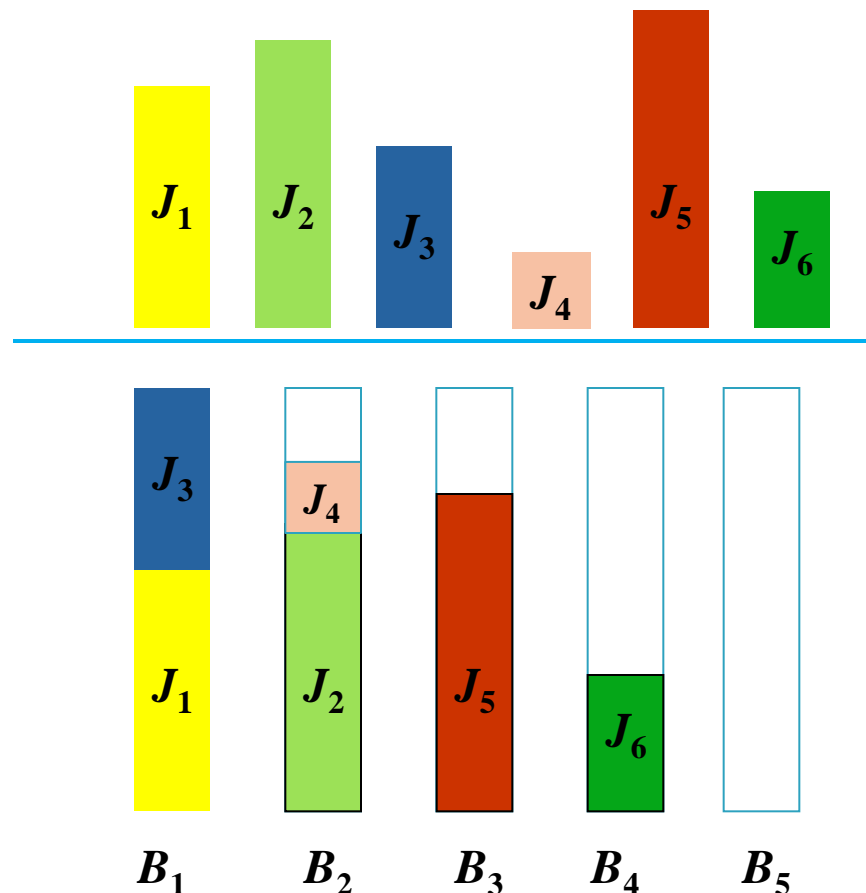
因而  $R_{FF}^{\infty} = \frac{17}{10} .$

## Example 2 $I : C = 10$

物品	$J_1$	$J_2$	$J_3$	$J_4$	$J_5$	$J_6$
$w_j$	6	7	4	2	8	3

### Solution :

首先，将  $J_1$  放入  $B_1$ ；  
 由于  $J_2$  在  $B_1$  中放不下，所以  
 将  $J_2$  放入  $B_2$ ，  
 对于  $J_3$ ，先检查  $B_1$  是否能  
 容纳下，能。所以将  $J_3$  放  
 入  $B_1$ ，...



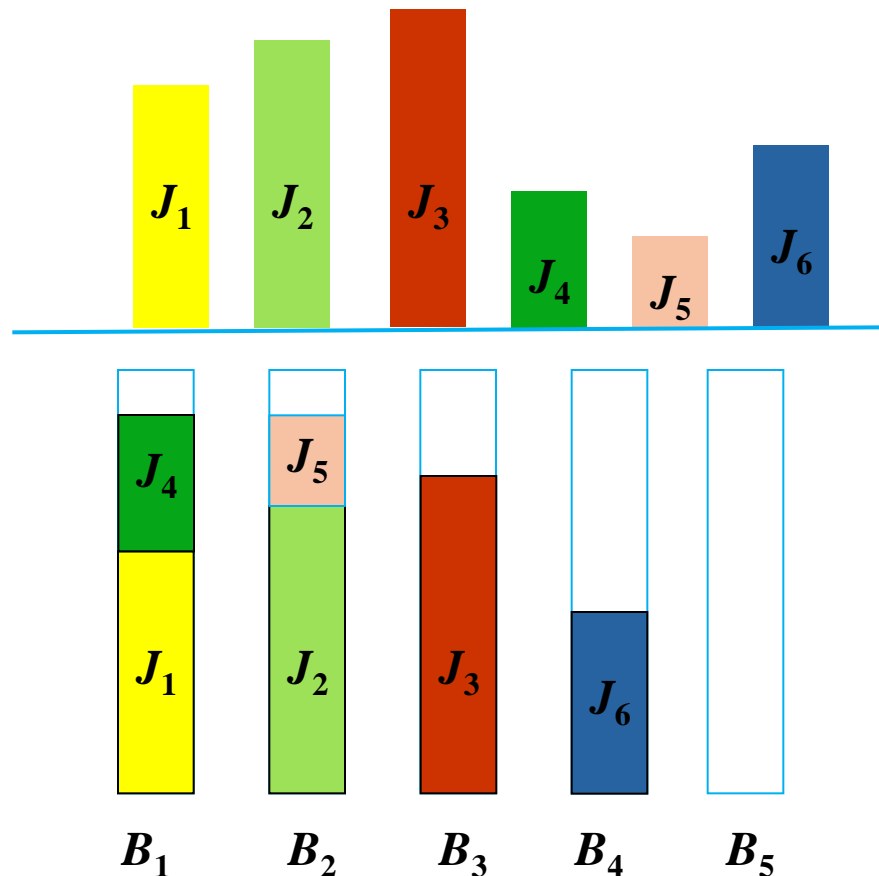
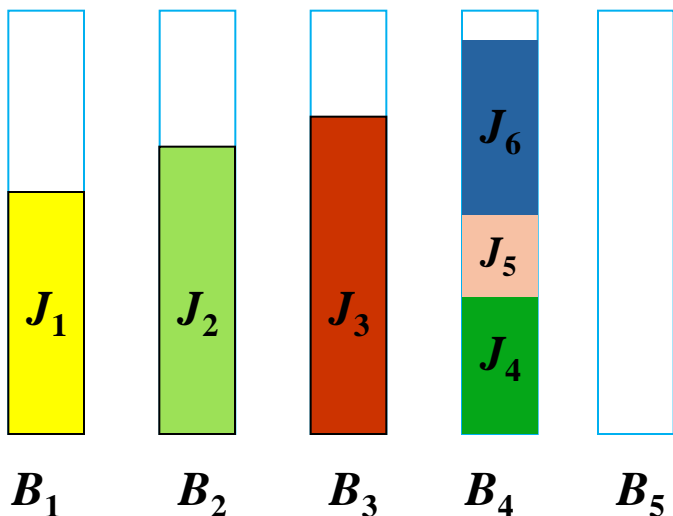
∴ 解为：

$x_{11} = x_{22} = x_{13} = x_{24} = x_{35} = x_{46} = 1$  其余为零， $z_{FF}(I) = 4$ .

物品	$J_1$	$J_2$	$J_3$	$J_4$	$J_5$	$J_6$
$w_j$	6	7	8	3	2	4

### Example 3 $I : C = 10$

**Solution :**



用  $NF$  算法  $z_{NF}(I) = 4$

用  $FF$  算法  $z_{FF}(I) = 4$

用  $FF$  算法装箱, 当放入  $J_4$  时,  $B_1$  能容纳  $J_4$  就放入  $B_1$ , 而事实上, **放入  $B_2$  更好**.

### 三、 $BF$ ( $Best\ Fit$ ) 算法

与  $FF$  算法相似，按物品给定的顺序装箱，区别在于对于每个物品  $J_j$  是放在一个使得  $J_j$  放入之后， $B_i$  所剩余长度为最小者。

即在处理  $J_j$  时，若  $B_1, B_2, \dots, B_i$  非空，而  $B_{i+1}$  尚未启用，设  $B_1, B_2, \dots, B_i$  所余的长度为

$\bar{w}_1, \bar{w}_2, \dots, \bar{w}_i$ ，若  $w_j > \max_{1 \leq k \leq i} \bar{w}_k$  则将  $J_j$  放入  $B_{i+1}$  内；

否则，从  $w_j \leq \bar{w}_k$  的  $B_k$  中，选取一个  $B_l$  ( $1 \leq l \leq i$ )

使得  $\bar{w}_l - w_j = \min_{\bar{w}_k \geq w_j} (\bar{w}_k - w_j)$  为最小者。

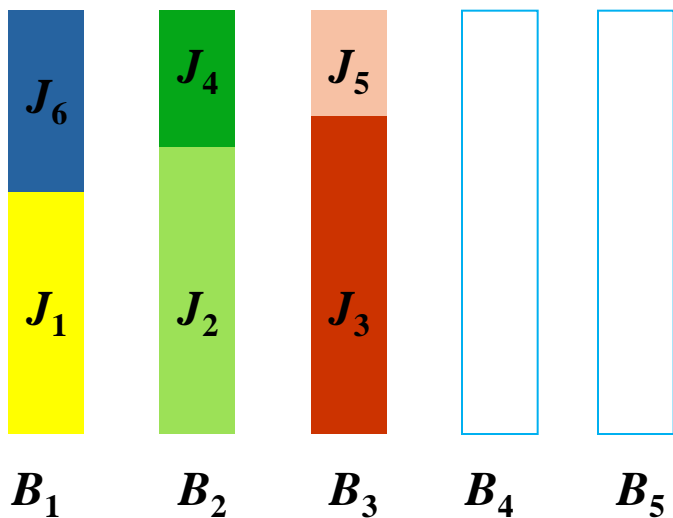
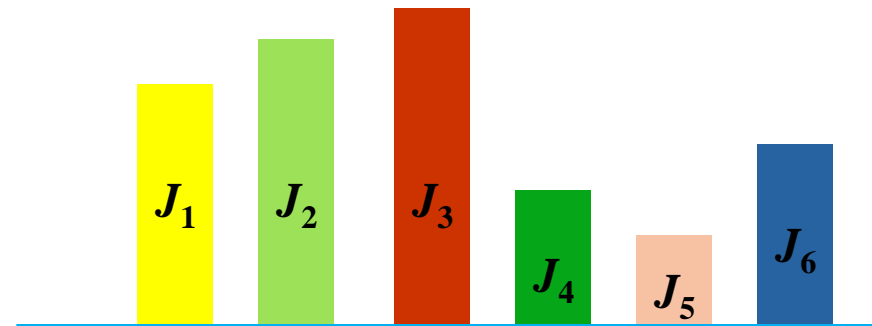
$BF$  算法的绝对性能比、计算复杂性与  $FF$  算法相同。

# Example 4 $I : C = 10$

Solution :

物品	$J_1$	$J_2$	$J_3$	$J_4$	$J_5$	$J_6$
$w_j$	6	7	8	3	2	4

用  $BF$  算法



$$\therefore L_1 = \left\lceil \frac{1}{10} \sum_{j=1}^6 w_j \right\rceil = 3$$

解为:

而  $z_{BF}(I) = L_1$ ,  $\therefore$  此为最优解.

$x_{11} = x_{22} = x_{33} = x_{24} = x_{35} = x_{16} = 1$  其余为零,  $z_{BF}(I) = 3$ .

## 四、*FFD* (*First Fit Decreasing*) 算法

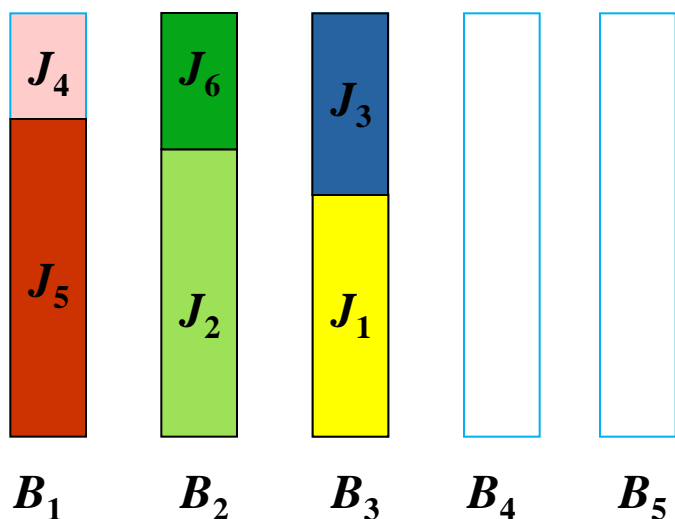
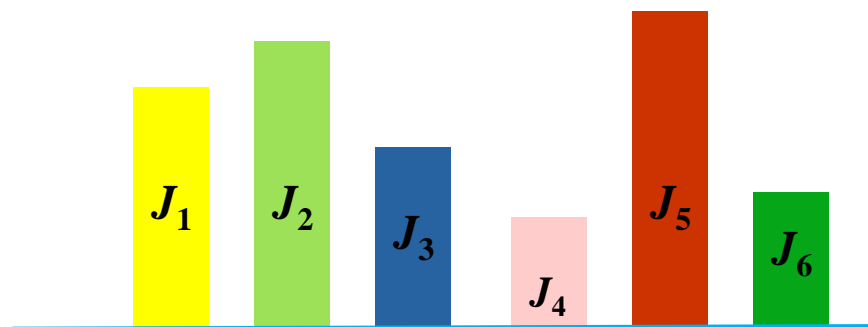
*FFD* 算法是先将物品按长度从大到小排序，然后用 *FF* 算法对物品装箱。该算法的计算复杂性为  $O(n \log n)$ 。

**Example 5**  $I : C = 10$

**Solution** : 已知:  $z_{FF}(I) = 4$

物品	$J_1$	$J_2$	$J_3$	$J_4$	$J_5$	$J_6$
$w_j$	6	7	4	2	8	3

物品	$J_5$	$J_2$	$J_1$	$J_3$	$J_6$	$J_4$
$w_j$	8	7	6	4	3	2



$z_{FFD}(I) = 3$  是最优的。

*NFD* 算法? *BFD* 算法?

**Theorem 8.6**  $R_{FFD}(I) = \frac{3}{2}.$

**Proof:** 显然对任意实例  $I$ , 有  $z_{FFD}(I) \geq z_{opt}(I)$

记  $z_{FFD}(I) = l$   $z_{opt}(I) = l^*$

首先证明两个结论:

(1)  $FFD$  算法所用的第  $l^* + 1, l^* + 2, \dots, l$  个箱子中, 每个物品长度不超过  $\frac{C}{3}$ ;

记  $w_i$  是放入第  $l^* + 1$  个箱子中的第一个物品, 只需证  $w_i \leq \frac{C}{3}$

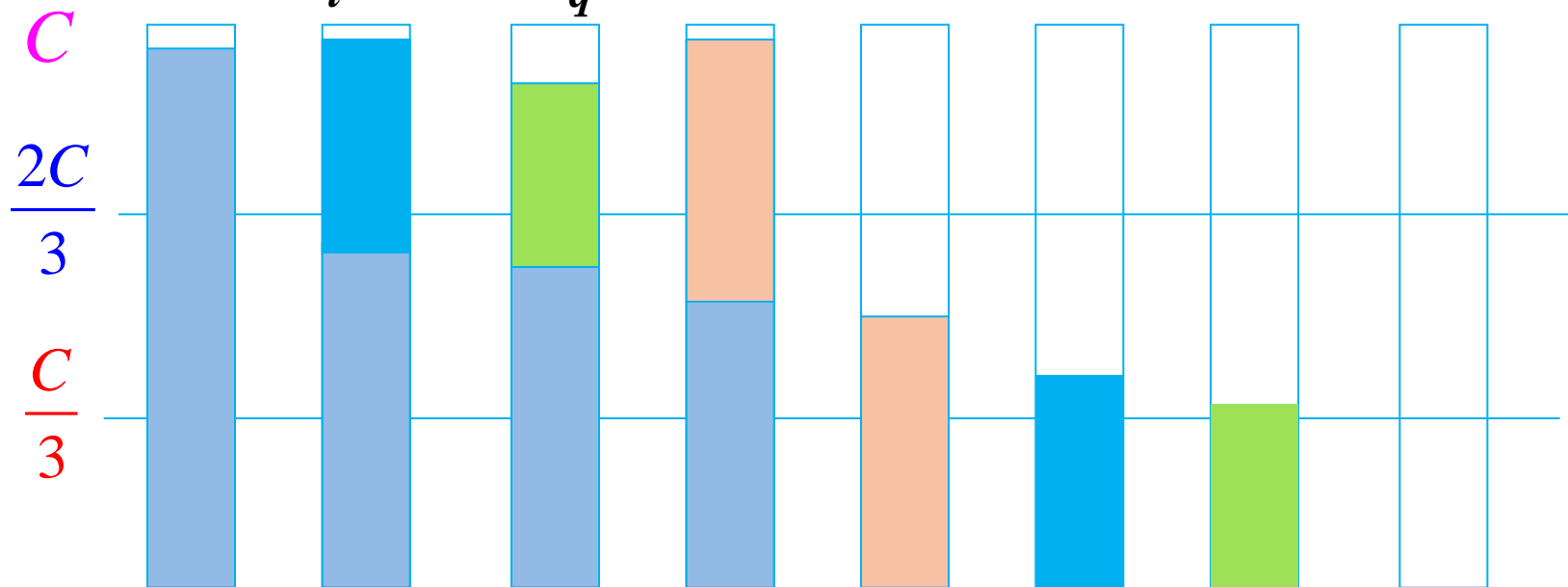
用反证法, 若不然, 则有  $w_1, \dots, w_{i-1} > \frac{C}{3}$ , 因此  $FFD$

算法中前  $l^*$  个箱子中, 每个箱子至多有两个物品.



可证明存在  $k \geq 0$  使前  $k$  个恰各含一个物品，后  $l^* - k$  个箱子各含两个物品。

因为若不然，则存在两个箱子  $B_p, B_q, p < q$ ，使  $B_p$  有两个物品  $w_{t_1}, w_{t_2} (t_2 > t_1)$ ， $B_q$  有一个物品  $w_{t_3}$  因物品已从大到小排列，故  $w_{t_1} \geq w_{t_3}, w_{t_2} \geq w_i$ ，因此  $C \geq w_{t_1} + w_{t_2} \geq w_{t_3} + w_i$ 。从而可以将  $w_i$  放入  $B_q$  中，矛盾。



因为  $FFD$  未将  $w_{k+1}, \dots, w_i$  放入前  $k$  个箱子, 说明其中任一个箱子已放不下, 故在最优解中也至少有  $k$  个箱子不含  $w_{k+1}, \dots, w_i$  中任一个物品. 假设就是前  $k$  个箱子, 因此在最优解中,  $w_{k+1}, \dots, w_{i-1}$  也会两两放入第  $k+1, \dots, l^*$  个箱子中, 且因为这些物品长度大于  $\frac{C}{3}$ , 所以每个箱子中只有两个物品, 且  $w_i > \frac{C}{3}$  已放不下. 但最优解中  $w_i$  必须放入前  $l^*$  个箱子中, 矛盾. 故  $w_i \leq \frac{C}{3}$

(2)  $FFD$  算法放入第  $l^*+1, \dots, l$  个箱子中物品数不超过  $l^*-1$

$$\because \sum_{i=1}^n w_i \leq l^* C, \text{ 而如果至少有 } l^* \text{ 个物品放入第 } l^*+1, \dots, l$$

个箱子中, 记前  $l^*$  个物品的长度为  $a_1, \dots, a_{l^*}$ .

记  $FFD$  算法中前  $l^*$  个箱子中每个箱子物品总长为  $b_j$ ,  $j = 1 \sim l^*$  显然, 对任意  $j = 1 \sim l^*$ ,  $b_j + a_j > C$  否则长为  $a_j$  的物品可放入第  $j$  个箱子中, 因此

$$\sum_{j=1}^n w_j \geq \sum_{j=1}^{l^*} b_j + \sum_{j=1}^{l^*} a_j = \sum_{j=1}^{l^*} (b_j + a_j) > l^* C \quad \text{矛盾}.$$

所以 (2) 结论成立.

由(1)、(2) 知  $FFD$  算法比最优算法多用的箱子是用来放至多  $l^* - 1$  个物品, 而每个物品长不超过  $\frac{C}{3}$ , 因此

$$\frac{z_{FFD}(I)}{z_{opt}(I)} \leq \frac{z_{opt}(I) + \left\lceil \frac{z_{opt}(I) - 1}{3} \right\rceil}{z_{opt}(I)} \leq 1 + \frac{z_{opt}(I) + 1}{3z_{opt}(I)} = \frac{4}{3} + \frac{1}{3z_{opt}(I)}$$

因为 如果  $z_{opt}(I) = 1$ ，则  $z_{FFD}(I) = 1$ ，故不妨设  $z_{opt}(I) \geq 2$

因此 
$$\frac{z_{FFD}(I)}{z_{opt}(I)} \leq \frac{4}{3} + \frac{1}{6} = \frac{3}{2}$$

考虑实例 I：物品集长度为  $\left\{\frac{C}{2}, \frac{C}{3}, \frac{C}{3}, \frac{C}{3}, \frac{C}{4}, \frac{C}{4}\right\}$ ，C 为箱长。

$z_{opt}(I) = 2$ ， $z_{FFD}(I) = 3$  说明  $\frac{3}{2}$  是不可改进的。□

比较  $NF$  算法、 $FF(BF)$  算法、 $FFD$  算法，它们的近似程度一个比一个好，但这并不是说  $NF$ 、 $FF(BF)$  就失去了使用价值。

1、 $FF(BF)$ 、 $FFD$  算法都要将所有物品全部装好后，所有箱子才能一起运走，而  $NF$  算法无此限制，很适合装箱场地小的情形；

2、 $FFD$  算法要求所有物品全部到达后才开始装箱，而  $NF$ 、 $FF(BF)$  算法在给某一物品装箱时，可以不知道下一个物品的长度如何，适合在线装箱。

## 存储罐注液问题

某化工厂有 9 个不同大小的存储罐，有一些已经装某液体。现新到一批液体化工原料需要存储，这些液体不能混合存储，它们分别是  $1200\text{ m}^3$  苯， $700\text{ m}^3$  丁醇， $1000\text{ m}^3$  丙醇， $450\text{ m}^3$  苯乙醇和  $1200\text{ m}^3$  四氢呋喃。下表列出每个存储罐的属性(单位:  $\text{m}^3$ )，问应如何将新到的液体原料装罐，才能使保留未用的存储罐个数最多？

存储罐编号	1	2	3	4	5	6	7	8	9
容 量	500	400	400	600	600	900	800	800	800
当前内容	-	苯	-	-	-	-	四氢呋喃	-	-
体 积		100					300		

**Solution :** 分别记苯、丁醇、丙醇、苯乙醇、四氢呋喃为第1, 2, 3, 4, 5种液体. 显然, 新到液体应尽可能装入已存有此种液体的罐中.

所以余下液体为:  $900 m^3$  苯,  $700 m^3$  丁醇,  $1000 m^3$  丙醇,  $450 m^3$  乙醇和  $700 m^3$  四氢呋喃. 剩余空罐为1, 3, 4, 5, 6, 8, 9. 由于不允许混合, 每种液体至少需要1个空罐.

令  $x_{ij} = \begin{cases} 1 & \text{第 } i \text{ 种液体装入第 } j \text{ 个存储罐} \\ 0 & \text{否则} \end{cases}$

记第  $j$  个空罐的容量为  $c_j$ ,  $j = 1, 3, 4, 5, 6, 8, 9$ ,

第  $i$  种剩余液体的体积为  $l_i$ ,  $i = 1, 2, 3, 4, 5$ .

存储罐编号	1	2	3	4	5	6	7	8	9
容量	500	400	400	600	600	900	800	800	800
当前内容	-	苯	-	-	-	-	四氢呋喃	-	-
体积		100					300		

## 整数规划模型:

$$\min z = \sum_{\substack{j=1 \\ j \neq 2,7}}^9 \sum_{i=1}^5 x_{ij}$$

$$s.t. \sum_{\substack{j=1 \\ j \neq 2,7}}^9 c_j x_{ij} \geq l_i \quad i = 1 \sim 5$$

$$\sum_{i=1}^5 x_{ij} \leq 1 \quad j = 1, 3, 4, 5, 6, 8, 9$$

$$x_{ij} \in \{0, 1\} \quad i = 1 \sim 5;$$

$$j = 1, 3, 4, 5, 6, 8, 9$$

$$\sum_{i=1}^5 x_{ij} = 1 \quad \text{表示第 } j \text{ 个空罐被使用}$$

每种液体的体积不能超过装这些液体的罐子的总容量

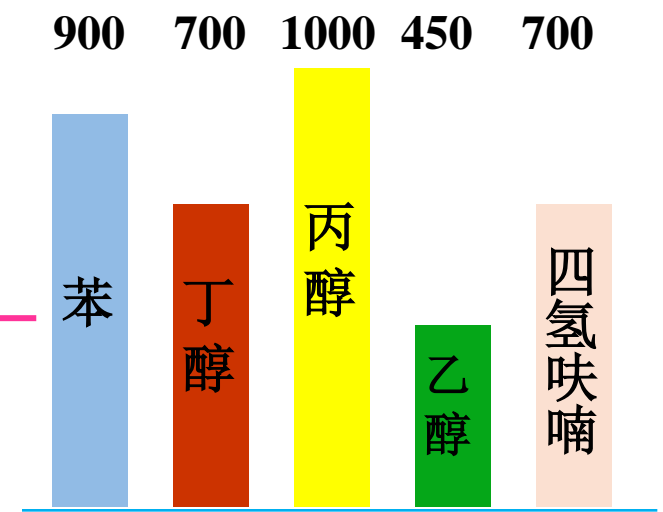
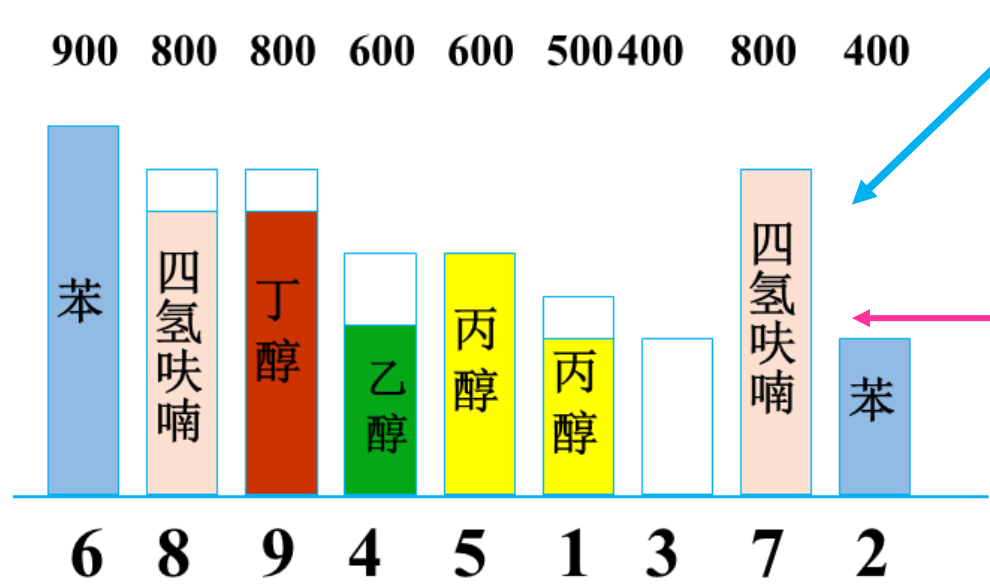
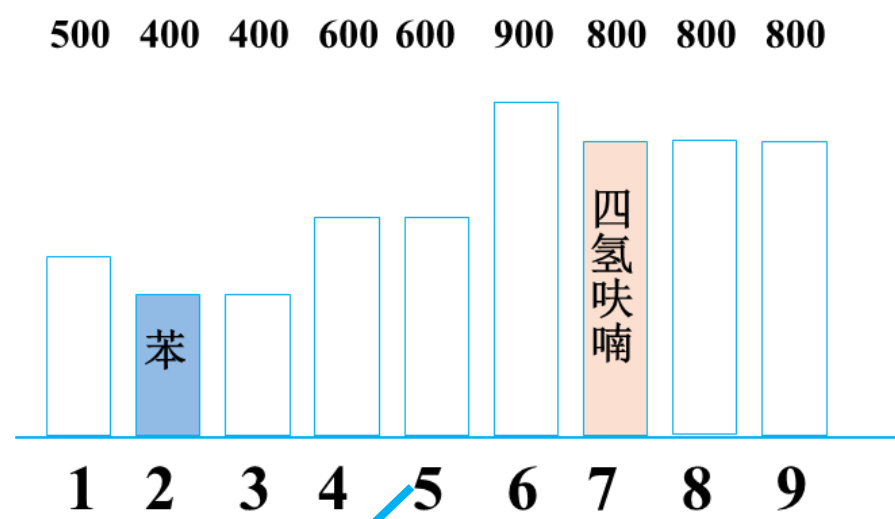
每个罐子至多装一种液体

将  $l_i$ 、 $c_j$  代入, 可用 *Lindo*、*Lingo* 等软件求解.



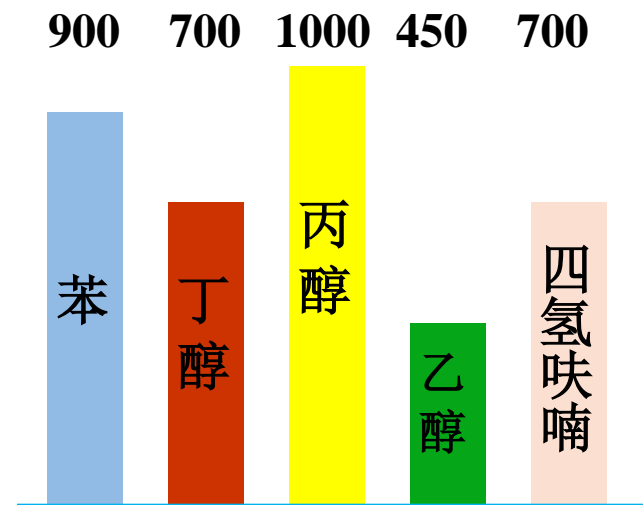
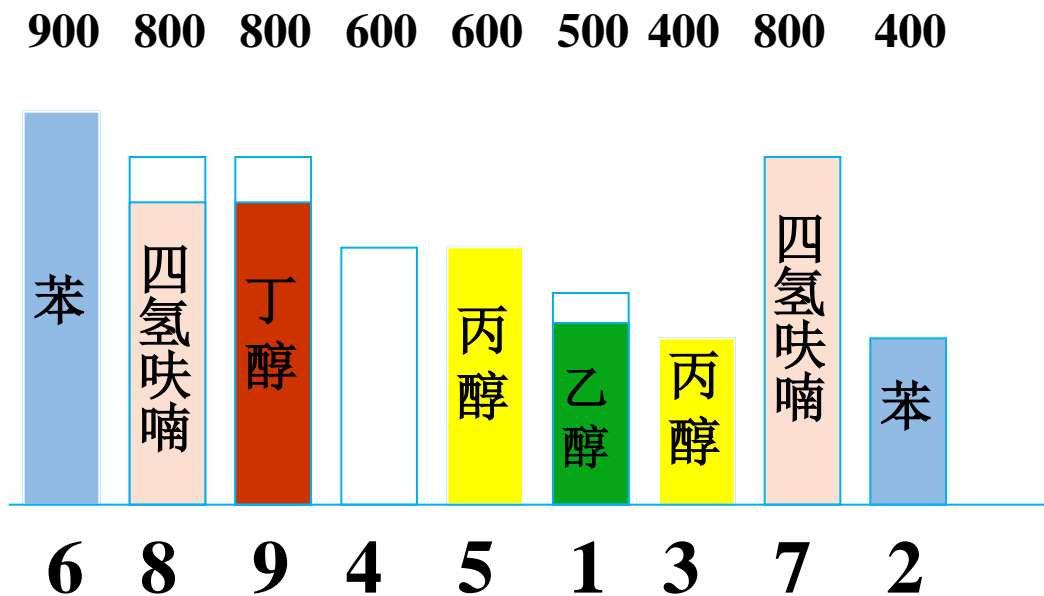
当问题的数据很大时，*IP* 的计算复杂性很高，可考虑采用近似算法或启发式算法求解。

如利用 *FFD* 算法思想：对每一种液体，将空罐按容积排成非增序；若需  $k$  个罐子，则装入具有连贯序号的罐子，使这  $k$  个空罐中最大容积空罐序号最大。



如果算法再细一点，只需要一个空罐的液体先装，效果会好一些。

后一种装法的空罐容积大，这是因为只需一个空罐的液体先装时，这部分装罐实现了最优。需要两个及以上空罐时，寻找最优算法的计算量就会变大。



# 本章结束