Knapsack Problem

(背包问题)

- 黄芝琪 林越

The Knapsack Problem (背包问题)

- Def: 所谓背包问题，是指有N个物品和一个背包，其中:
  - 物品具有重量 (w_1, w_2, …, w_n) 和利润 (p_1, p_2, …, p_n)
  - 背包的最大重量承受限制为W

  问如何取物可得最高价值?

- 此问题可以表示如下:

$$\sum_{item_i \in A} p_i \quad \text{is maximized subject to} \quad \sum_{item_i \in A} w_i \leq W.$$

背包问题形态

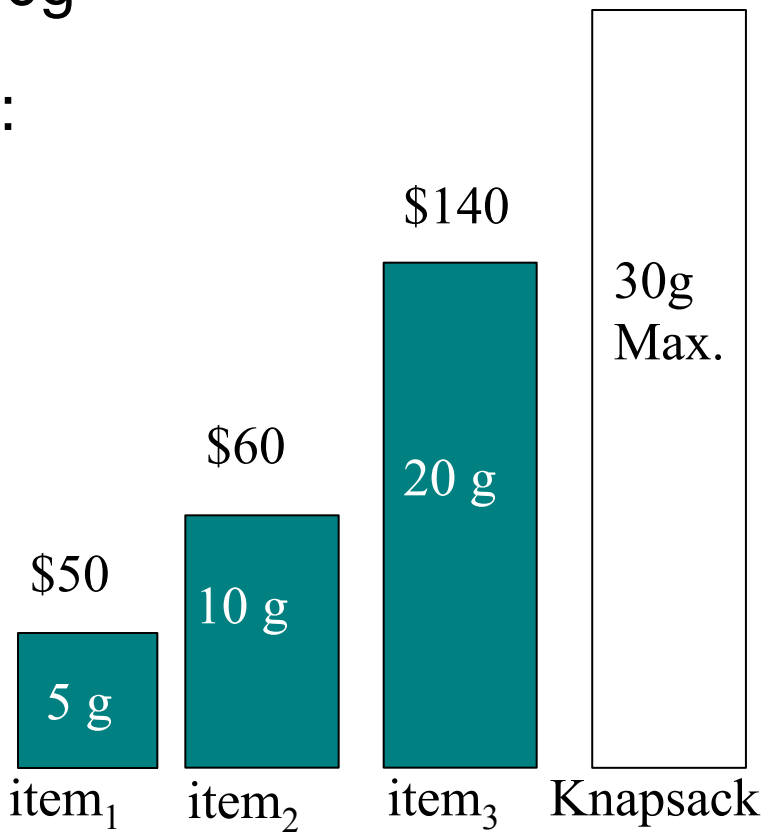
- 背包问题可分成两种问题形态:
  - 可切割背包问题:
    - 物品可被切割，亦即取物时可取部分
    - 采用贪心策略(Greedy Approach)
  - 0/1 背包问题:（取或不取）
    - 物品不可被切割，亦即取物时得取全部
    - 采用动态规划(Dynamic Programming)

Greedy Approach v.s. Dynamic Programming

- **Greedy Approach**
  - 是一种阶段性 **(Stage)** 的方法
  - 不考虑后续，只考虑每次取最优
- **Dynamic Programming**
  - 先把所有的情況都看一遍，才去挑出最佳的結果

范例

- 可切割的背包问题
  - 背包可承担的最大重量: 30g
  - 三个物品之重量及其利润:
    - Item 1: 5 g, $50
    - Item 2: 10 g, $60
    - Item 3: 20 g, $140

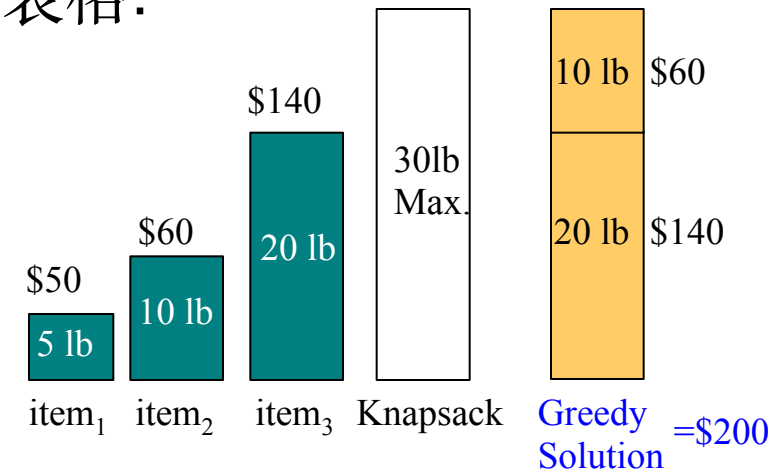


- 物品可被切割，亦即取物时可取部分
- 采用贪心策略
  - 1.利润
  - 2.重量:
  - 3.利润与重量比:

最大利润优先

- 根据题目定义，我們可以得到下列表格:

Item	重量 (g)	利润	利润/重量比
1	5	$50	10
2	10	$60	6
3	20	$140	7

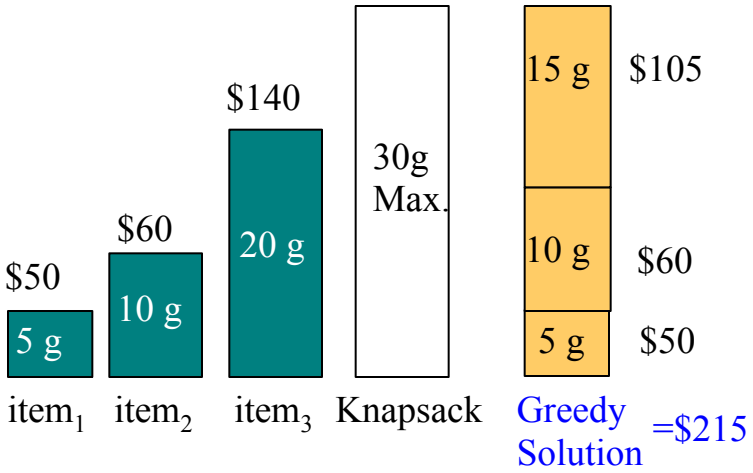


- 選擇程序採 “最大利潤優先”:
  - Step 1: 取 20 g的Item 1，可得利润為 $140，背包剩余重量: 10 g
  - Step 2: 取10 bl的Item 2，連同Step 1所取之20 g的Item 1，可得总利润为 $200，背包剩余重量: 0 g
  - Step 3: 因为背包满了，故完全无法取得Item 3
  - 所得总利润 = **$200**

最小重量優先

- 根据题目定义，可以得到下列表格:

Item	重量 (g)	利润	利润/重量比
1	5	$50	10
2	10	$60	6
3	20	$140	7

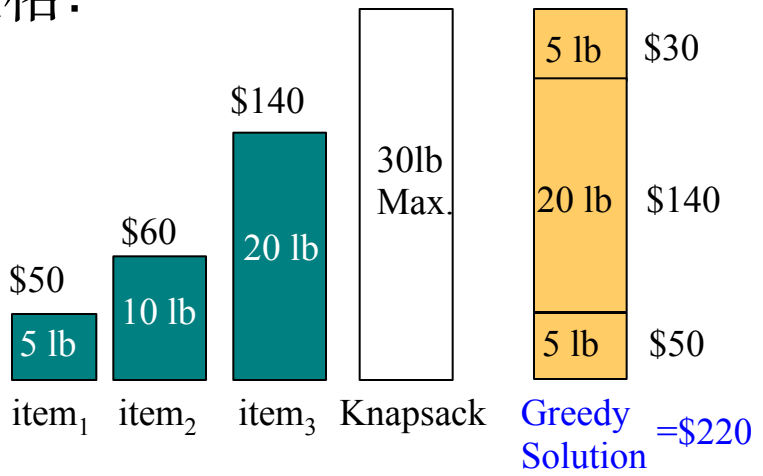


- 选择程序採 “最小重量优先”:
  - Step 1: 取 5 g的Item 1，可得利润为 $50，背包剩余重量: 25 g
  - Step 2: 取10 g的Item 2，連同Step 1所取之5 g的Item 1，可得总利润为 $110，背包剩余重量: 15 g
  - Step 3: 由于背包剩余重量为15 g，而Item 3的重量有20 g，因此只能取 ¾ 的Item 3，连同之前的操作，可得总利润 $215，背包剩剩余重量: 0 g
  - 所得总利润 = **$215**

最大利潤與重量比

- 根據題目定義，我們可以得到下列表格:

Item	重量 (bl)	利潤	利潤/重量比
1	5	$50	10
2	10	$60	6
3	20	$140	7



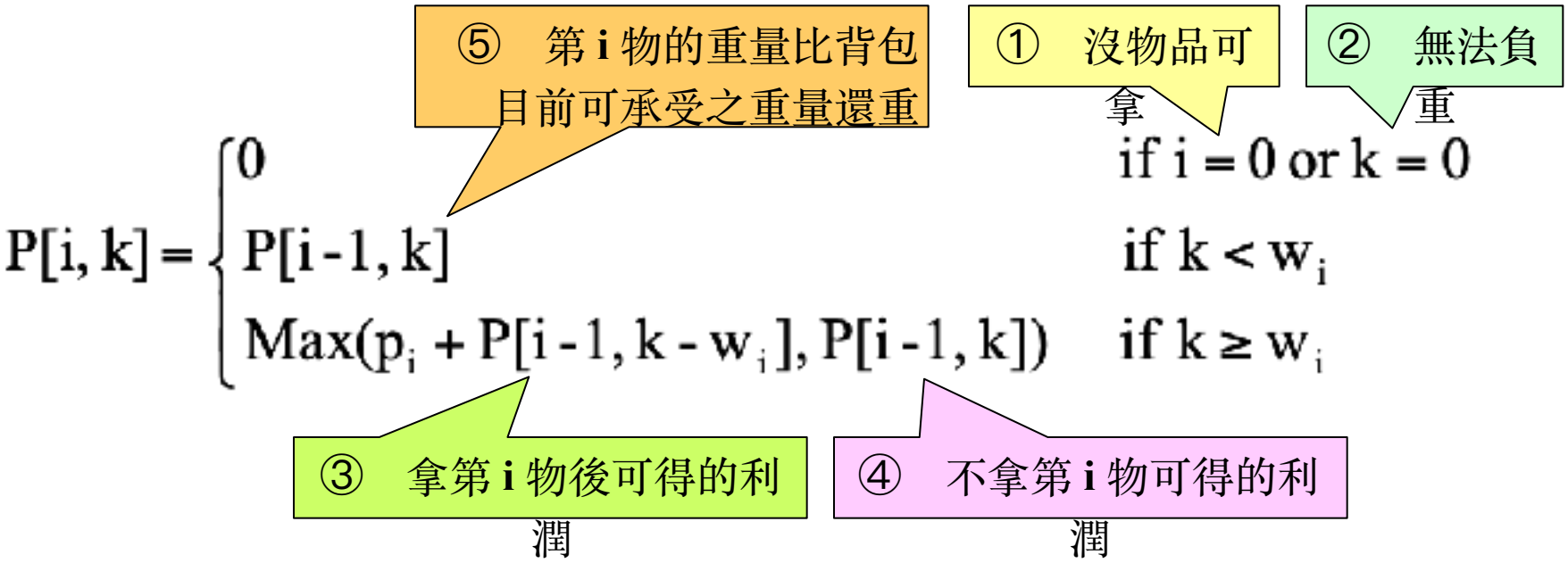
- 選擇程序採 "最大利潤與重量比":
  - Step 1: 取 5 bl的Item 1，可得利潤為 $50，背包剩餘重量: 25 bl
  - Step 2: 取20 bl的Item 3，連同Step 1的結果，可得總利潤為 $190，背包剩餘重量: 5 bl
  - Step 3: 由於背包剩餘重量為5 bl，而Item 2的重量有10 bl，因此僅能取 ½ 的 Item 2，連同前兩步的結果，可得總利潤為 $220，背包剩餘重量: 0 bl
  - 所得總利潤 = **$220**

0/1 Knapsack Problem

- 物品不可被切割，亦即取物时得取全部
- 若仍采用贪心算法，使得程序为 “最大利润与重量比”:
  - Step 1: 取 5 g的Item 1，可得利润为 $50，背包剩余重量: 25 g
  - Step 2: 取20 g的Item 3，連同Step 1的結果，可得总利润 $190，背包剩余重量: 5 g
  - Step 3: 不可分割，所以取不了了
  - 所得总利润 = **$190**，但是真正的最佳解是 **200**

    ∴ 0/1 背包不可用贪心算法求解!!

- [状态转移方程]:


⑤ 第 i 物的重量比背包目前可承受之重量還重
① 沒物品可拿
② 無法負重
$P[i,k]=\begin{cases}0 & \text{if } i=0 \text{ or } k=0\\ P[i-1,k] & \text{if } k<w_i\\ \text{Max}(p_i+P[i-1,k-w_i],P[i-1,k]) & \text{if } k\geq w_i\end{cases}$
③ 拿第 i 物後可得的利潤
④ 不拿第 i 物可得的利潤

- 范例: 假设有一背包W = 5，考虑以下的Items，求0/1 背包最佳解:

Item	重量	利潤
O_1	1	$6
O_2	2	$10
O_3	3	$12

Sol:

P	0	1	2	3	4	5
0						
1						
2						
3						

- Step 1: 当 i = 0，表示沒有任何物品可以拿 (即: 狀況 ①)。
  ∴ P[0, k] = 0

P	0	1	2	3	4	5
0	0	0	0	0	0	0
1						
2						
3						

- Step 2: 当 i = 1，表示有 1 个物品可以拿 (即: O_1)。
  - ∵k = 0 ，表示無法負重 (狀況 ②); ∴P[1, 0] = 0
  - k=1，表示能負重1; 此時:

$$P[1,1] = \max\begin{cases} P[0,1] \\ p_1 + P[0,0] \end{cases} = \max\begin{cases} 0 \\ 6+0 \end{cases} = 6$$

  - K = 2~5，表示能負重2 ~ 5; 但由于只有1个物品 (即: O_1) 可拿，因此P[1, k]。

P	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	6	6	6	6	6
2						
3						

- Step 3: 当 i = 2，表示有 2 个物品可以拿 (即: O_1 和O_2)。
  - ∵k = 2 ，表示负重2; 此時:

$$P[2,2]=\max\begin{cases}P[1,2]\\ p_2+P[1,0]\end{cases}=\max\begin{cases}6\\ 10+0\end{cases}=10$$

  - ∵k = 3 ，表示负重3; 此時:

$$P[2,3]=\max\begin{cases}P[1,3]\\ p_2+P[1,1]\end{cases}=\max\begin{cases}6\\ 10+6\end{cases}=16$$

P	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	6	6	6	6	6
2	0	6	10	16	16	16
3						

- Step 4: 当 i = 3，表示有 3 个物品可以拿 (即: O_1、 O_2 與O_3)。
  - ∵k = 3 ， 表示负重3; 此時:

$$P[3,3]=\max\begin{cases}P[2,3]\\ p_3+P[2,0]\end{cases}=\max\begin{cases}16\\ 12+0\end{cases}=16$$

P	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	6	6	6	6	6
2	0	6	10	16	16	16
3	0	6	10	16		

- ∵k = 4 ， 表示负重4; 此时:

$$P[3,4]=\max\begin{cases}P[2,4]\\ p_3+P[2,1]\end{cases}=\max\begin{cases}16\\ 12+6\end{cases}=18$$

- ∵k = 5 ， 表示负重5; 此時:

$$P[3,5]=\max\begin{cases}P[2,5]\\ p_3+P[2,2]\end{cases}=\max\begin{cases}16\\ 12+10\end{cases}=22$$

P	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	6	6	6	6	6
2	0	6	10	16	16	16
3	0	6	10	16	18	22

Thanks.