背包問題(Knapsack Problem)

2009.10.27 AikoSenoo

Q

你要準備出遠門去賣水果，卻發現自己的包包太小，裝不下太多你想賣的水果，於是，問題就來了！

給你一堆你想賣的水果，每種東西有「重量」和「價值」，並告訴你包包的負重量，請找出每種水果可以帶多少個，讓整個背包裝完後最有價值？

舉個例子來說……

4Kg $800

3Kg $450

5Kg $300

Some Ideas!?

How about Greedy!?

用Greedy的話.......

策略是什麼？

- 價值高的先？
- 重量小的先？
- 單位價值高的先？

三種擺放順序

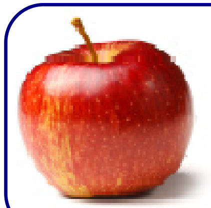
- 最比較輕的先放

3Kg $450

X 4

12Kg $2300

- 比較有價值的先放
  單位價值較高的先放



4Kg $800

X3

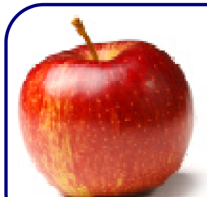
12Kg $2400

最佳擺法

3Kg $450

X2

+



4Kg $800

X2

14Kg **$2500**

關於Greedy的作法...

在很多時候可能是對的
但有時會有因為沒放滿而
造成價值較低的情況

另外一種作法...

背包問題之

Dynamic Programming
（動態歸化法）

遞迴式

C(x, n): 負重量為x且最多放n種東西進去時，
背包裡所裝的物品的最大價值

令w[i]為第i樣物品的重量，v[i]為第i樣物品的價值

遞迴關係為：

$$C(x,\ n) = \max(\underbrace{C(x,\ n-1)}_{\text{不把這樣東西放進去}},\ \underbrace{C(x-w[i],\ n) + v[i]}_{\text{把這樣東西放進去}})$$

令 W = 背包的最大負重量，且總共有N種物品

➔最佳解為C(W, N)

邊界情況

關於C(x, n) = max(C(x, n-1), C(x-w[i], n) + v[i])這個式子

需做以下特殊處理：

C(x, 0) = 0

若x == w[i]

→C(x, n) = max(C(x, n-1), v[i])

若C(x-w[i], n) == 0 或 x-w[i] < 0

→ C(x, n) = C(x, n-1)

化為表格

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1															
2															
3															

化為表格

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	800	0	0	0	1600	0	0	0	2400	0	0
2															
3															

4Kg $800

化為表格

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	800	0	0	0	1600	0	0	0	2400	0	0
2	0	0	0	450	800	0	900	1250	1600	1350	1700	2050	2400	2150	2500
3															

3Kg $450

化為表格

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	800	0	0	0	1600	0	0	0	2400	0	0
2	0	0	0	450	800	0	900	1250	1600	1350	1700	2050	2400	2150	2500
3	0	0	0	450	800	300	900	1250	1600	1350	1700	2050	2400	2150	2500

5Kg $300

其實可以化為一維表格

（因為每次只會用到當前狀態和上一個狀態）

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	800	0	0	0	1600	0	0	0	2400	0	0

其實可以化為一維表格

（因為每次只會用到當前狀態和上一個狀態）

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	450	800	0	900	1250	1600	1350	1700	2050	2400	2150	2500

其實可以化為一維表格

（因為每次只會用到當前狀態和上一個狀態）

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	450	800	300	900	1250	1600	1350	1700	2050	2400	2150	2500

Something else...

- 物品填表的順序不影響正確性
- 時間複雜度為O(NW)，但只有在W夠小時(陣列開得起來)，背包問題才有解

0/1 背包問題
（0/1 Knapsack problem）

Q

如果現在問題變成：
每樣東西只有一個

那我們該如何做呢？

修改遞迴式

C(x, n)： 負重量為x且最多放n樣東西進去時，
背包裡所裝的物品的最大價值

令w[i]為第i樣物品的重量，v[i]為第i物品的價值

遞迴關係為：

C(x, n) = max(C(x, n-1), C(x-w[i], n-1) + v[i])

不把這樣東西放進去　把這樣東西放進去

令 W = 背包的最大負重量，且總共有N樣物品

➔最佳解為C(W, N)

邊界情況

關於C(x, n) = max(C(x, n-1), C(x-w[i], n-1) + v[i])這個式子

需做以下特殊處理：

C(x, 0) = 0

若x == w[i]

→C(x, n) = max(C(x, n-1), v[i])

若C(x-w[i], n-1) == 0 或 x-w[i] < 0

→ C(x, n) = C(x, n-1)

再舉個例子來說.......

4Kg $800

3Kg $450

2Kg $300

5Kg $900

1Kg $100

填表格囉～

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1											
2											
3											
4											
5											

填表格囉～

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	800	0	0	0	0	0	0
2											
3											
4											
5											

4Kg $800

填表格囉～

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	800	0	0	0	0	0	0
2	0	0	0	450	800	0	0	1250	0	0	0
3											
4											
5											

3Kg $450

填表格囉～

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	800	0	0	0	0	0	0
2	0	0	0	450	800	0	0	1250	0	0	0
3	0	0	300	450	800	750	1100	1250	0	1550	0
4											
5											

2Kg $300

填表格囉～

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	800	0	0	0	0	0	0
2	0	0	0	450	800	0	0	1250	0	0	0
3	0	0	300	450	800	750	1100	1250	0	1550	0
4	0	0	300	450	800	900	1100	1250	1350	1700	1650
5											

5Kg $900

填表格囉～

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	800	0	0	0	0	0	0
2	0	0	0	450	800	0	0	1250	0	0	0
3	0	0	300	450	800	750	1100	1250	0	1550	0
4	0	0	300	450	800	900	1100	1250	1350	1700	1650
5	0	100	300	450	800	900	1100	1250	1350	1700	1800

1Kg $100

化為一維的注意事項

我們所需要的狀態變成只有上一個狀態，然而如果我們由前往後更新的話，上一個狀態就會被覆蓋掉

➔ 由後往前更新

填表格囉～

0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	800	0	0	0	0	0	0

4Kg $800

填表格囉～

0	1	2	3	4	5	6	7	8	9	10
0	0	0	450	800	0	0	1250	0	0	0

3Kg $450

填表格囉～

0	1	2	3	4	5	6	7	8	9	10
0	0	300	450	800	750	1100	1250	0	1550	0

2Kg $300

填表格囉～

0	1	2	3	4	5	6	7	8	9	10
0	0	300	450	800	900	1100	1250	1350	1700	1650

5Kg $900

填表格囉～

0	1	2	3	4	5	6	7	8	9	10
0	100	300	450	800	900	1100	1250	1350	1700	1800

1Kg　　$100

關於回溯

- 另外開一個陣列，紀錄該重量最後一個放入的是哪樣東西
- 然後再看看(該重量-最後一樣放入東西的重量)這個重量最後放入哪樣東西
- 形成一個遞迴關係，所以我們可以用遞迴一路找回去

0/1 背包問題的變化

常見的幾種變化

- 給你一些物品的重量，不論物品價值，問你能不能裝到某個重量
  Ex: ACM Q624
- 給你一些數字，問你能不能夠把這些數字分成相等兩半(或盡量相等的兩半)
  Ex: ACM Q10664, TIOJ 1508(2008TOI初選)
  **** 注意奇偶數問題
  ACM Q562, Q10032
- 給你一些物品的重量，問你能達到最大負重量的方式有幾種(含排列方式)
  Ex: TOIJ 1475
  (96北市賽 with 大數 注意overflow)

一些雜談

剛剛提到W要夠小，背包問題才有解，實際上背包問題是一種NP-Complete Problem，NP的意思是「Nondeterministic Polynomial time」也就是說，「目前還不知道有沒有可以在多項式時間解掉他的方法」，有些NP-Complete有近似解法，像背包在W不大時有DP解法，如果真的很大時，我們就會用Greedy去求近似解

NP-Complete是資訊科學界很多人都再尋找解法的一群問題，最奇妙的是，只要能夠有人找到任何一個問題的多項式時間解法，那麼這些問題就通通被解決了！

一些雜談(2)

關於動態規劃法，其實是一門尋找適當「狀態」與「轉移方法」的學問。轉移方法通常都是遞迴式，然而轉移方法常常很容易找到錯的(而且不一定能很快地找到問題點)！

DP是所有演算法裡變化最多的方法，一開始可能會不習慣，但隨著經驗的累積，會慢慢知道什麼樣的狀態是比較適合的:)

so, 練習很重要唷！