

# NCKU Programming Contest Training Course Course 8 2015/03/18

#### Mong-Ting Wu(aj211y)

aj211y@gmail.com

http://myweb.ncku.edu.tw/~f74006323/course\_8\_dp2.pptx

Department of Computer Science and Information Engineering National Cheng Kung University Tainan, Taiwan





#### **Outline**



- Coin change problem
- Knapsack problem







- 類型:
  - 硬幣限制各1個
  - 硬幣無限多個
  - 硬幣有限

#### 求

- 是否湊得某個價位
- 湊得某價位的方法數
- 湊得某價位的最少硬幣用量





- 硬幣限制各一個,是否湊得某價位
- dp[i]:能否湊得價位i(0-不可/1-可以) (dp[0]=1→0元必可湊得)
- v[k]:第k種幣值
- If (dp[i-v[k]] == true) dp[i] = 1;

```
如果價位 i 可以由v[ k ]湊得,
表示價位 i – v [ k ]可以被湊
得( dp [ i - v[ k ])=1 )
```





- v[k] = 2, 5 (有2跟5這兩種幣值)
- If (dp[i-v[k]] == true) dp[i] = 1;

注意:每個硬幣只有一個, 所以可以湊得<mark>最大的價位</mark>就 是每個硬幣都用過一次 **Σν[k]** 

初始化:零元必可湊得

	0	1	2	3	4	5	6	7	8
dp	1	0	1	0	0	0	0	0	0

$$v[k] = 2$$

dp[2] 更新值為1 因為價位 2可以被湊出來





- v[k] = 2, 5 (有2跟5這兩種幣值)
- If (dp[i-v[k]] == true) dp[i] = 1;

注意:每個硬幣只有一個, 所以可以湊得<mark>最大的價位</mark>就 是每個硬幣都用過一次 Σν[k]

	0	1	2	3	4	5	6	7	8
dp	1	0	1	0	0	1	0	1	0

$$v[k] = 5$$

dp[7] 更新值為1 因為價位 7可以被湊出來

• dp[i] = 1 → 可以湊出價位 i





- 硬幣無限,是否湊得某價位
- dp[0]=1
- v[k] = 2, 5
- If (dp[i-v[k]] == true) dp[i] = 1; (i 從v[k]開始跑)

	0	1	2	3	4	5	6	7	8
dp	1	0	1	0	1	0	1	0	1

$$v[k] = 2$$

dp[2] 更新值為1 因為價位 2可以被湊出來

要注意 i 的邊界值,只需要從 v[k] 跑到 Max





- 硬幣無限,是否湊得某價位
- dp[0]=1
- v[k] = 2, 5
- If (dp[i-v[k]] == true) dp[i] = 1; (i 從v[k]開始跑)

	0	1	2	3	4	5	6	7	8
dp	1	0	1	0	1	1	1	1	1

$$v[k] = 5$$

dp[5] 更新值為1 因為價位 5可以被湊出來





- 硬幣無限,湊得某價位有幾種可能
- dp[0]=1
- v[k] = 2, 3
- If (dp[i-v[k]] == true) dp[i] += dp[i-v[k]];

	0	1	2	3	4	5	6	7	8
dp	1	0	0	0	0	0	0	0	0



	0	1	2	3	4	5	6	7	8
dp	1	0	1	0	1	0	1	0	1





- 硬幣無限,湊得某價位有幾種可能
- dp[0]=1
- v[k] = 2, 3
- If (dp[i-v[k]] == true) dp[i] += dp[i-v[k]];

	0	1	2	3	4	5	6	7	8
dp	1	0	1	1	1	1	2	1	2

$$v[k] = 3$$

價位 3 的可能組合有 dp[ 3 ] 種 每一種加上硬幣 3 之後皆可組成價位 6 所以組成價位 6 的可能組合 多了dp [ 3 ] 種

dp[6] = dp[6] + dp[3];



#### **Practice**



• 基礎:Uva 674

• 進階:Uva 10898





- Knapsack Problem: 背包問題 將一堆物品塞進背包,要使背包裡的<mark>物品總價值最高</mark>, 但背包有**耐重限制**,所以塞的太重的話,背包就會撐 破。
- **0/1**: 物品只會放進背包**0個或1個**,物品不可切割,所以只有不放或者全放兩種可能。





- 窮舉法 每個物品有放或不放2種選擇,有N個物品,所有的情 況列舉需O(2^N)
- DP

對某個物品可以選擇放或不放,然後移去這個物品來縮小問題範圍,時間複雜度 O(NW)。(N種物品,W是限重)





- dp[n][m]: 前n項物品,在背包載重m的情況下,所能擁有的最大價值
- v[n]: 物品n的價值
- w[n]:物品n的重量
- dp[n][m] = 以dp[n-1][m']來模擬dp[n][m]的情況 max (dp[n-1][m], dp[n-1][m-w[n]]+v[n])

不放物品 n 的情況下 背包載重m'=m 不變

放了物品 n 的情況下 背包載重m' = m - w[n]最大價值要加上物品n 的value





- dp[n][m] =
   max (dp[n-1][m], dp[n-1][m-w[n]]+v[n])

有物品且背包還有空間可以放



#### Top-down DP

```
int dp[N+1][W+1], v[N], w[N];
                                       // top-down, N items with
                                       maximum total weight W
bool isfind[N][W];
int knapsack(int n, int m){
    if (m < 0) return -INF;
                                               // basic constrain
    if (n == 0) return 0;
                                               // isfind before
   if (isfind[n][m]) return dp[n][m];
   dp[n][m] = max(knapsack(n-1, m),
               knapsack(n-1, m-w[n]) + v[n]); // recursively call
                                               // record isfind
   isfind[n][m] = true;
    return dp[n][m];
```

# acm international Collegiate Programming Contest

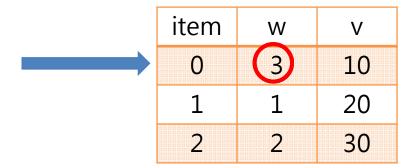
# 0/1 Knapsack Problem

- Bottom-up DP
- c[m]: 背包載重m的情況下,所能擁有最大的價值





• c[m] = max(c[m], c[m - w[i]] + v[i]);



	,	١		١		•		
			Н	-	H			
			1	щ				
			l	9	)			

$$max(c[5], c[5-3] + v[0]) = 10$$



c[0]	c[1]	c[2]	c[3]	c[4]	c[5]	c[6]
0	0	0	10	10	10	10

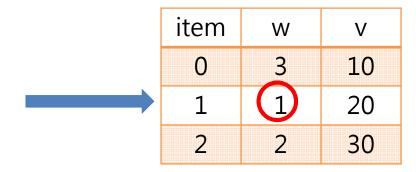
因為m=0表示 背包能載重0 故背包最大價值0







• c[m] = max(c[m], c[m - w[i]] + v[i]);



	١	/	\	/			
 							7
		×	J				
		١	J				
		١	J				
		١	J				

$$max(c[5], c[5-1] + v[1]) = 30$$

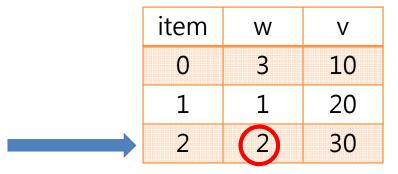
					V	
c[0]	c[1]	c[2]	c[3]	c[4]	c[5]	c[6]
0	20	20	20	30	30	30







• c[m] = max(c[m], c[m - w[i]] + v[i]);





$$max(c[5], c[5-2] + v[2]) = 50$$

							_
c[0]	c[1]	c[2]	c[3]	c[4]	c[5]	c[6]	ans
0	20	30	50	50	50	60	<b>—</b>
					1	1	



#### **Practice**



• POJ-3624





# Thank for Your Attention

