(1) 0-1 背包問題

前 i 個物品，在重量限制為 j 的情況下，若選了第 i 個物品，則問題就變成「前 i-1 個物品，在重量限制為 $j-w_i$ 的情況下，能拿取的最大價值」；若不拿，則重量限制維持不變，因此可以寫出以下的狀態轉移式：

$dp(i, j) = max(dp(i-1, j), dp(i-1, j-w_i) + v_i)$

(2) 0-1 背包問題之二

這題物品沒有數量限制，很容易想出下面的狀態轉移式：

$dp(i, j) = max(\quad dp(i-1, j-k*w_i) + k*v_i), \quad j - k*w_i >= 0$

但這樣做時間複雜度太大。可以想到，在 k>= 1，也就是拿超過一個的情況下，相當於在 $j - w_i$ 的情況下，拿取 k-1 個。也就是說 k >= 1 的情況，已經儲存在 $dp(i, j-w_i)$裡了，因此可以改寫成下面這樣：

$dp(i, j) = max(dp(i-1, j), \quad dp(i, j-w_i) + v_i)$

(3) 這題重量限制變得很大，但價值大小不變。因此可以改寫成「前 i 個物品，選擇價值為 j 的物品組合的最小重量」：

$dp(i, j) = min(dp(i-1, j), dp(i-1, j-v_i) + w_i)$

最後只要在 dp(n, j)中選取滿足 dp(n, j) <= W 的最大 j 就行了。

(4) 這題跟(2)很像，差別是有數量限制。很快地可以想出如下的狀態轉移式：

$dp(i, j) = sum(dp(i-1, j-k)) \quad j-k >=0 \&\& k <= a_i$

但這樣會超時。根據(2)的思路，我們可以知道拿 k >= 1 個的情況，相當於在 j-1 情況下拿 k-1 個。但在 j-1 拿剛好 a_i 個的情況，在 j 的情況下就會變成拿 a_i+1 個，因此必須將其扣掉：

$dp(i, j) = dp(i-1, j) + dp(i, j-1) - dp(i-1, j-1-a_i)$

(5) 這題可以想成「將 n 顆相同的球，放進 m 個相同的箱子裡」。可以分成兩種情況：(1)每個箱子至少都有一顆球 (2) 至少有一個箱子沒球。

假設 dp(i, j)表 i 顆球放進 j 個箱子的方法數，根據上面的兩種情況，可以寫出如下的狀態轉移式：

$dp(i, j) = dp(i - j, j) + dp(i, j-1)$

(6) 假設 dp(i, j)代表前 i 個數值是否能組成 j，可以想如下狀態轉移式：

$dp(i, j) \mid= dp(i-1, j-k*a_i) \quad j-k*a_i >= 0 \&\& k <= m_i$

反正還是超時啦。因此需要將 k 給提出來。假設 dp(i, j)表示前 i 個數值組成 j 後，還剩多少個 a_i。可以分三種情況

(a) 若使用前 i-1 個數值就可以完成 j 的話，那第 i 個數值就會剩下全部

(b) 若利用前 i 個數值組成 $j-a_i$，還剩下 k >= 1 個 a_i 的話，前 i 個組成 j 就會剩下 k-1 個

(c) 若都不行，則 dp(i, j) = -1

三者取較大值，最後檢查 dp(n, K)是否大於等於 0 即可。