商管程式設計（109-1）
作業四

作業設計：孔令傑
國立臺灣大學資訊管理學系

繳交作業時，請至 PDOGS（`http://pdogs.ntu.im/judge/`）為第一、二、三、四題各上傳一份 Python 3.5 原始碼（以複製貼上原始碼的方式上傳）。每位學生都要上傳自己寫的解答。不接受紙本繳交；不接受遲交。

這份作業的截止時間是 ***2020 年 10 月 26 日晚上九點***。在你開始前，請閱讀課本的第十章[1]。為這份作業設計測試資料並且提供解答的助教是林聖典。

第一題

（10 分）你正要去參加俗稱戀愛巴士的森多概野外課程，在思考要帶哪些東西。你有 n 個東西可以選，每個東西都有其重量跟效用（價值），並且不可分割（要就不帶，不然就整個帶）。令第 i 個東西的重量與效用各為 w_i 公斤與 v_i 單位。你的背包（或你）最多只能承受 B 公斤的負重，你想要在此限制下最大化你帶的東西的總效用。

下面是一個例子。你最多只能背 $B = 5$ 公斤，而你的七個東西如表 **??** 所示。顯然你不能全帶，因為這樣總負重 6.9 公斤就超過負重限制了。你可以帶指南針、雨傘、環保餐具、照相機、雨衣，這樣的總效用是 22；你也可以把雨衣換成筆電，你還是背得動，且這樣總效用就提高到 23 了。事實上，在這個例子中 23 就是你所能得到的最大效用了。

編號 i	1	2	3	4	5	6	7
名稱	指南針	雨傘	環保餐具	照相機	雨衣	筆電	拉拉熊
重量（公斤）w_i	0.5	1.5	0.4	1	1.1	1.6	0.8
效用 v_i	6	5	4	4	3	4	1

表 1: 範例一

這個問題就是資訊科學領域中有名的「背包問題」（knapsack problem）。如果要更精確地描述這個問題，我們可以寫一個數學模型[2]。令 $x_i \in \{0, 1\}$ 表示我們是否帶物品 i，$x_i = 1$ 表示要帶，為 0 則否。則背包問題的模型即為

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} v_i x_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} w_i x_i \le B \\ & x_i \in \{0, 1\} \quad \forall i = 1, ..., n. \end{aligned}$$

任意一個「解」（solution）都可以被一個維度為 n 的向量 $x = (x_1, ..., x_n)$ 表示，若 $\sum_{i=1}^{n} w_i x_i \le B$ 則 x 為一個「可行解」（feasible solution），若 $\sum_{i=1}^{n} v_i x_i$ 還不小於任何一個可行解的總效用，則 x 為一個

[1]課本是 A. Downey 所著的 *Think Python 2*，在 `http://greenteapress.com/wp/think-python-2e/` 可以下載。
[2]這個模型是所謂的整數規劃模型（integer program）。

「最佳解」(optimal solution)。

背包問題有非常多的應用。舉例來說，當零售商要決定在店裡展示什麼商品時，他有有限的貨架空間、各商品要佔去的空間與預期營收不一，零售商要最大化其總預期營收；公司的有限預算要用來從數個專案中挑一些出來執行，每個專案所需的預算與報酬不一，公司要最大化總報酬。當然在大多數的應用中，它都伴隨著其他因子一起以更複雜的形式出現，但精神是不變的。

在資訊科學（computer science）與作業研究（operations research）領域中，學者們普遍認為背包問題是「困難的」，也就是說只要可選的物品數夠多、背包夠大，那任何演算法都無法在合理的時間內求得最佳解。因此，本題並不要求你針對給定的背包問題求得最佳解。我們將給你一組解，要求你判斷這組解是否為可行解，若為可行解則計算出總效用。

輸入輸出格式

系統會提供一共數組測試資料，每組測試資料裝在一個檔案裡。在每個檔案中，會有四行。第一行有兩個正整數 n 與 B，分別是物品個數與負重上限；第二行含有 n 個正整數 w_1、w_2 直到 w_n，其中 w_i 是物品 i 的重量；第三行含有 n 個正整數 v_1、v_2 直到 v_n，其中 v_i 是物品 i 的效用；第四行含有 n 個正整數 x_1、x_2 直到 x_n，其中 $x_i \in \{0,1\}$ 代表是否有帶第 i 個物品，若有則為 1，反之則為 0。已知 $1 \le n \le 100$、$1 \le B \le 10000$、$1 \le w_i \le 1000$、$1 \le v_i \le 1000$，且 $\max_{i=1,\ldots,n}\{w_i\} \le B$，亦即每個物品都可以被單獨地裝進背包。每一行的任兩個值之間被一個逗號隔開。請依題目所述，判斷這組解是否為可行解，若為可行解則印出總負重與總效用，中間以一個逗號隔開，不然的話就輸出 -1。

舉例來說，如果輸入是

```
4,10
3,4,3,1
7,6,3,2
1,1,0,1
```

則輸出應該是

```
8,15
```

如果輸入是

```
5,9
2,3,4,3,8
2,4,5,3,10
1,1,1,0,1
```

則輸出應該是

```
-1
```

你上傳的原始碼裡應該包含什麼

你的.py 原始碼檔案裡面應該包含讀取測試資料、做運算，以及輸出答案的 Python 3.5 程式碼。當然，你應該寫適當的註解。針對這個題目，你***可以***使用上課沒有教過的方法。

評分原則

這一題的所有分數都根據程式運算的正確性給分。PDOGS 會直譯並執行你的程式、輸入測試資料，並檢查輸出的答案的正確性。本題共有 5 組測試資料，一筆測試資料佔 2 分。

第二題

（20 分）接續上題，這次我們將給你一個簡單的演算法，要求你按照這個演算法去執行，以求得一個可行解。這個演算法執行的方式如下。我們計算出每一個物品的 CP 值 r_i，在這裡我們定義 $r_i = \frac{v_i}{w_i}$。將物品依 CP 值由大到小排序以後，我們依序檢視每一個物品，看該物品能不能被放進背包裡，如果能放的話就放，不然就繼續看下一個物品，直到所有物品都已經檢查過了。本題中不會有任兩個物品 CP 值相同。

下面我們將用一個例子說明。假設 $n = 4$、$B = 10$，題目給定 w_i 和 v_i 後，我們計算出 CP 值，如表 **??**。CP 值由大排到小的物品編號為 4、2、3、1。我們依這個順序檢查能不能選擇該物品：

- 順位一：背包目前負重為 0，物品 4 重量為 4，選擇物品 4。
- 順位二：背包目前負重為 4，物品 2 重量為 5，選擇物品 2。
- 順位三：背包目前負重為 9，物品 3 重量為 3，跳過物品 3。
- 順位四：背包目前負重為 9，物品 1 重量為 1，選擇物品 1。

編號 i	1	2	3	4
重量（公斤）w_i	1	5	3	4
效用 v_i	1	100	10	1000
CP 值 r_i	1	20	$\frac{10}{3}$	250

表 2: 範例二

在本題中，要請你使用本演算法去找出要被放進背包的物品編號，並將此解按照物品編號由小到大印出。以上面的例子來說，印出來的解應依序為 1、2、4。在本題中若你需要做排序，可以使用 `list` 函式庫裡面的函數。

輸入輸出格式

系統會提供一共數組測試資料，每組測試資料裝在一個檔案裡。在每個檔案中，會有三行。第一行有兩個正整數 n 與 B，分別是物品個數與負重上限；第二行含有 n 個正整數 w_1、w_2 直到 w_n，其中 w_i 是物品 i 的重量；第三行含有 n 個正整數 v_1、v_2 直到 v_n，其中 v_i 是物品 i 的效用；已知 $1 \le n \le 100$、$1 \le B \le 10000$、$1 \le w_i \le 1000$、$1 \le v_i \le 1000$，且 $\max_{i=1,\ldots,n}\{w_i\} \le B$，亦即每個物品都可以被單獨地裝進背包。每一行的任兩個值之間被一個逗號隔開。請利用指定的演算法求得一個可行解，並將此解依物品編號由小到大依序印出，兩兩以一個逗號隔開。

舉例來說，如果輸入是

```
4,10
3,4,3,1
7,6,3,2
```

則輸出應該是

```
1,2,4
```

如果輸入是

```
5,9
2,3,4,3,8
3,4,5,3,11
```

則輸出應該是

```
1,2,3
```

你上傳的原始碼裡應該包含什麼

你的.py 原始碼檔案裡面應該包含讀取測試資料、做運算，以及輸出答案的 Python 3.5 程式碼。當然，你應該寫適當的註解。針對這個題目，你***可以***使用上課沒有教過的方法。

評分原則

這一題的所有分數都根據程式運算的正確性給分。PDOGS 會直譯並執行你的程式、輸入測試資料，並檢查輸出的答案的正確性。本題共有 10 組測試資料，一筆測試資料佔 2 分。

第三題

（30 分）接續上題，現在我們告訴你還有另外一種演算法，就是在每一次挑選物品時，選出當下能讓總效用最大、且不會讓背包過重的物品裝進背包裡，直到背包再也裝不下了。如果有兩樣物品效用相同的話，選比較輕的那個；如果重量還是一樣的話，選編號小的那個。本題中可能會有兩個物品 CP 值相同，執行第一種演算法時，如果有兩樣物品 CP 值相同的話，選比較輕的那個；如果重量還是一樣的話，選編號小的那個。請你執行在第二題和第三題介紹的這兩個演算法，選出結果比較好（總效用比較大）的那一個，並將解依照要挑選的物品的編號由小到大印出來。如果兩個演算法結果一樣，請選第二題的演算法的執行結果。

下面我們將用一個例子說明。假設 $n = 4$、$B = 10$，題目給定 w_i 和 v_i 如表 **??**。執行第一種演算法的話，CP 值由大排到小的物品編號為 1、4、2、3。我們依這個順序檢查能不能選擇該物品，發現應該選擇物品 1、2、4，此時總效用為 15。執行第二種演算法的話，過程如下：

- 順位一：背包目前負重為 0，目前效用最大的物品 1 重量為 3，選擇物品 1。

- 順位二：背包目前負重為 3，目前效用最大的物品 2 重量為 4，選擇物品 2。
- 順位三：背包目前負重為 7，目前效用最大的物品 3 重量為 3，選擇物品 3。
- 順位四：背包目前負重為 10，背包已經滿了。

執行第二種演算法的話，應該選擇物品 1、2、3，此時總效用為 16。比較兩種解，發現第二種演算法的總效用比較大，因此印出來的解應為 1、2、3。

編號 i	1	2	3	4
重量（公斤）w_i	3	4	3	1
效用 v_i	7	6	3	2

表 3: 範例三

輸入輸出格式

系統會提供一共數組測試資料，每組測試資料裝在一個檔案裡。輸入格式和第二題一模一樣。請你利用指定的演算法求得比較好的可行解，並將此解依物品編號順序印出，兩兩以一個逗號隔開。

舉例來說，如果輸入是

```
4,10
3,4,3,1
7,6,3,2
```

則輸出應該是

```
1,2,3
```

如果輸入是

```
5,9
2,3,4,3,8
2,4,5,3,10
```

則輸出應該是

```
1,2,3
```

你上傳的原始碼裡應該包含什麼

你的.py 原始碼檔案裡面應該包含讀取測試資料、做運算，以及輸出答案的 Python 3.5 程式碼。當然，你應該寫適當的註解。針對這個題目，你**可以**使用上課沒有教過的方法。

評分原則

這一題的所有分數都根據程式運算的正確性給分。PDOGS 會直譯並執行你的程式、輸入測試資料，並檢查輸出的答案的正確性。本題共有 15 組測試資料，一筆測試資料佔 2 分。

第四題

（40 分）在投資理財時，有時我們會面對類似前述的背包問題：我們有有限的預算 B 元，以及數個投資標的，投資標的 i 需要 w_i 元、預期報酬為 v_i 元，而我們想要最大化總預期報酬。當然，實務上面對投資標的，我們通常可以選擇要投多少錢（例如一支股票要買幾股），但在本題為了簡單起見，讓我們假設面對每個投資標的，我們的投資依然是全有全無的：要就花 w_i 元做完整的投資，不然就完全不可以投資。

投資有賺有賠，除了預期報酬，我們也關心風險，而很不幸地（很自然地?），高報酬通常總是伴隨著高風險，這會讓我們的選擇變得更加不易。更不幸的是，投資標的的損益之間經常是相關的（例如同類型的股票可能會一起漲跌），也就是說若投資在兩個標的上，風險可能會加成（例如買可口可樂和百事可樂的股票）或減少（例如買可口可樂和有機健康食品公司的股票）。這些都應該納入考慮，才能找到兼顧預期報酬和風險的最佳投資組合。

針對一個投資標的，人們經常用損益的變異數（variance）來衡量風險，變異數愈大表示風險愈大；針對兩個投資標的，人們則經常用兩者的損益的共變數（covariance）來衡量兩者間的風險連動性，共變數為正表示兩者的損益變動為正相關，為負則表示為負相關，絕對值愈大表示連動性愈高。在本題中，我們將使用 σ_i^2 來表示投資標的 i 之損益的變異數，用 σ_{ij} 來表示投資標的 i 與 j 之損益的共變數。如果把所有的變異數和共變數都收集到一個矩陣 Σ，這個矩陣就可以在 Python 裡面被一個二維清單（two-dimensional list）儲存起來。與之類似地，如果把 w_i 和 v_i 也各自收集到一個向量 w 和 v，則這兩個向量也各可以被一個一維清單儲存起來。

舉例來說，假設我們有四個投資標的（編號依序為 1 到 4），並且已知

$$w = \begin{bmatrix} 8 \\ 7 \\ 5 \\ 4 \end{bmatrix}、v = \begin{bmatrix} 20 \\ 30 \\ 40 \\ 35 \end{bmatrix}、\Sigma = \begin{bmatrix} 10 & 4 & -3 & -9 \\ 4 & 8 & -2 & -3 \\ -3 & -2 & 11 & 4 \\ -9 & -3 & 4 & 3 \end{bmatrix},$$

則表示投資標的 2 的資金需求為 7、預期報酬為 30、變異數為 8，依此類推。此外，也可以看到投資標的 1 和 2 之間為正相關、3 和 4 之間也正相關，而 1 和 3、1 和 4、2 和 3、2 和 4 之間則為負相關。想當然爾，共變數矩陣 Σ 會是個對稱矩陣（$\sigma_{ij} = \sigma_{ji}$），且對角線上的數字會是正的（$\sigma_i^2 > 0$）。

有一種權衡預期報酬與風險的方法是求解以下的最佳化問題。令 $x_i \in \{0, 1\}$ 為是否投資第 i 個投資標的，$x_i = 1$ 表示要投資，$x_i = 0$ 表示不要，則要被求解的最佳化問題為

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} v_i x_i - \eta \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_{ij} x_i x_j \\ \text{s.t.} \quad & \sum_{i=1}^{n} w_i x_i \le B \\ & x_i \in \{0, 1\} \quad \forall i = 1, ..., n， \end{aligned} \tag{1}$$

其中 $\eta \geq 0$ 是該投資人心中的風險趨避係數，愈大表示他愈在意風險。換言之，每多選一個投資標的，就可以賺到一些預期報酬，但通常也相對應地要承擔多一些風險（也有的時候會降低風險）。

由於這顯然比前述的背包問題還難，在本題中我們將給你一個簡單的演算法，請照著實做並且求出一個可行的投資組合。演算法會執行數輪挑選，每一輪會挑一個投資標的進入投資組合，挑選的方式很簡單：在「當下還沒被挑中，且挑了不會超過預算」的投資標的中，挑選「能最大化目標式 (**??**)」的那個投資標的，如果有數個符合條件的選項，就挑其中資金需求最低的，再平手就挑編號小的。演算法會持續這個挑選規則，直到在某一輪沒有挑選任何投資標的（可能是因為沒有足夠的剩餘資金了，也可能是因為挑選任何新的投資標的都不會使目標式值上升了）為止。請注意也有可能在第一輪就沒有挑選任何投資標的。

以上面的例子來說，假設預算 $B = 15$，$\eta = 1$，則：

- 第一輪：挑投資標的 4，因為 $35 - 1 \times 3$ 是我們能得到的最大目標式值，請注意風險需要再乘上 η，變異數與共變異數都要，η 此處為 1，下面亦同。另外請注意風險不是 0，而是 $\sigma_4\sigma_4 = \sigma_4^2 = 3$，是目標式值中的減項，所以目標式值是 $35 - 1 \times 3 = 32$ 而非 35。
- 第二輪：讓我們依序考慮剩下的三個投資標的：
  - 投資標的 1：預期報酬加 20，風險部分則是減 10（σ_1^2）然後加 18（$\sigma_{14} + \sigma_{41}$），目標式值總計加 28。
  - 投資標的 2：預期報酬加 30，風險部分則是減 8（σ_2^2）然後加 6（$\sigma_{24} + \sigma_{42}$），目標式值總計也是加 28。
  - 投資標的 3：預期報酬加 40，風險部分則是減 11（σ_3^2）然後減 8（$\sigma_{34} + \sigma_{43}$），目標式值總計加 21。

  由於前兩個投資標的一樣好，因此我們選資金需求較低的投資標的 2。
- 第三輪：此時由於剩餘的資金 $15 - 11 = 4$ 已經不足以投資任何還未被選中的投資標的，因此演算法結束。

演算法結束後，請依然依照前幾題的輸出規則，將被選中的投資標的依照編號由小到大印出。在這個例子中，請依序輸出 2、4。如果演算法沒有挑選任何投資標的，請印出 0。

輸入輸出格式

系統會提供一共數組測試資料，每組測試資料裝在一個檔案裡。在每個檔案中，會有 $n + 3$ 行。第一行有三個正整數 n、B、η，分別是投資標的個數、預算與風險趨避係數；第二行含有 n 個正整數 w_1、w_2 直到 w_n，其中 w_i 是投資標的 i 的資金需求；第三行含有 n 個正整數 v_1、v_2 直到 v_n，其中 v_i 是投資標的 i 的預期報酬；第四行到第 $n + 3$ 行的每一行都含有 n 個整數，第四行含有 σ_1^2、σ_{12} 直到 $\sigma_{1,n}$、第五行含有 σ_{21}、σ_2^2 直到 $\sigma_{2,n}$，依此類推，最後第 $n + 3$ 行含有 $\sigma_{n,1}$、$\sigma_{n,2}$ 直到 σ_n^2、已知 $1 \leq n \leq 100$、$1 \leq B \leq 10000$、$1 \leq w_i \leq 1000$、$1 \leq v_i \leq 1000$、$-1000 \leq \sigma_{ij} \leq 1000$（若 $i \neq j$）、$1 \leq \sigma_i^2 \leq 1000$、$0 \leq \eta \leq 10$。每一行的任兩個值之間被一個逗號隔開。請利用指定的演算法求得一個可行解，並將此解依物品編號由小到大依序印出，兩兩以一個逗號隔開。

舉例來說，如果輸入是

```
4,15,1
8,7,5,4
20,30,40,35
10,4,-3,-9
4,8,-2,-3
-3,-2,11,4
-9,-3,4,3
```

則輸出應該是

```
2,4
```

如果輸入是

```
4,8,10
8,7,5,4
2,3,4,3
10,4,-3,-9
4,8,-2,-3
-3,-2,11,4
-9,-3,4,3
```

則輸出應該是

```
0
```

你上傳的原始碼裡應該包含什麼

你的.py 原始碼檔案裡面應該包含讀取測試資料、做運算，以及輸出答案的 Python 3.5 程式碼。當然，你應該寫適當的註解。針對這個題目，你***不可以***使用上課沒有教過的方法。

評分原則

- 這一題的 20 分會根據程式運算的正確性給分。PDOGS 會直譯並執行你的程式、輸入測試資料，並檢查輸出的答案的正確性。一筆測試資料佔 2 分。
- 這一題的其中 20 分會根據你所寫的程式的品質來給分。助教會打開你的程式碼並檢閱你的程式的可讀性（包含排版、變數命名、註解等等）。請寫一個「好」的程式吧！