商管程式設計（112-2）
作業三

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/`）爲第一、二、三、四題各上傳一份 Python 3.9 原始碼（以複製貼上原始碼的方式上傳；第四題爲額外加分題，本次作業滿分 120 分，得幾分算幾分）。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 ***3 月 23 日中午十二點***。在你開始前，請閱讀課本的第五、七章[1]。爲這份作業設計測試資料並且提供解答的助教是陳鵬仁。

第一題

（20 分）在送了精心設計的玫瑰花束之後，小傑迎來了與心儀的女生的第一場約會。不過小傑的平日的工作相當繁重，他必須盡力在期限前做完手邊的所有工作才能夠去赴約。具體來說，小傑手邊總共有 n 件工作，每一個工作都有一個固定的處理時間 p_i，以及工作必須完成的的期限 d_i。對於這些工作，小傑會安排好一個「工作順序」（簡稱「順序」）s 去逐一完成每一項工作，例如 $s = (4, 2, 1, 3)$ 表示小傑要依序做工作 4、工作 2、工作 1，最後做工作 3。當然，任何一項順序都無法保證所有工作都可以在期限前完成。每項工作的「延遲時間」的計算方式如下。首先，把該工作的完成時間 x_i 減去該工作的期限 d_i，如果算出來是正的，該數字就是這項工作的延遲時間；反之如果是負的，表示該工作在期限前就已經完成了，則該工作的延遲時間爲 0。如果所有工作的總延遲時間太多，那位女生就會認定小傑工作太繁忙了，進而取消約會，因此小傑必須有辦法評估一個給定順序會導致的總延遲時間。

舉例來說，假設小傑手邊有 4 個工作，依序分別是 1、2、3、4 號工作，這些工作的處理時間依序是 5 小時、4 小時、3 小時、2 小時，完成期限分別是開始工作後的第 6 小時、第 7 小時、第 8 小時、第 10 小時。若小傑安排的工作順序爲 $(1, 4, 2, 3)$，則總延遲時間的計算過程如下：

- 第一步做 1 號工作並開始計時，5 小時後工作完成，完成的時間爲開始後第 5 小時，不超過給定的期限第 6 小時，總延遲時間目前爲 0。
- 第二步做 4 號工作並開始計時，2 小時後工作完成，完成的時間爲開始後第 $5 + 2 = 7$ 小時，不超過給定的期限第 7 小時，總延遲時間目前爲 0。
- 第三步做 2 號工作並開始計時，4 小時後工作完成，完成的時間爲開始後第 $7 + 4 = 11$ 小時，超過給定的期限 $11 - 7 = 4$ 小時，總延遲時間目前爲 4 小時。
- 第四步做 3 號工作並開始計時，3 小時後工作完成，完成的時間爲開始後第 $11 + 3 = 14$ 小時，超過給定的期限 $14 - 8 = 6$ 小時，總延遲時間最終爲 10 小時。

因此，小傑在這個工作順序下總延遲時間爲 10 小時。

請參考以上例子與依照題目指示，根據給定的工作資訊，輸出給定工作順序下的總延遲時間。

[1] 課本是 A. Downey 所著的 *Think Python 2*，在 `http://greenteapress.com/wp/think-python-2e/` 可以下載。

輸入輸出格式

系統會提供數組測試資料，每組測試資料裝在一個檔案裡。在每個檔案中會有四列資料，第一列含有一個正整數 n；第二列含有 n 個正整數 p_1、p_2 直到 p_n；第三列含有 n 個正整數 d_1、d_2 直到 d_n；第四列含有 n 個正整數 s_1、s_2 直到 s_n。每一列的任兩個數字之間用一個逗點隔開。已知 $1 \leq n \leq 20$、$1 \leq p_i \leq 50$、$1 \leq d_i \leq 1000$、$1 \leq s_i \leq n$，且 s_i 彼此之間不重複。

讀入這些資訊以後，請依題目指示印出一個整數，代表總延遲時間。舉例來說，如果輸入是

```
4
5,4,3,2
6,7,8,10
1,4,2,3
```

則輸出應該是

```
10
```

如果輸入是

```
4
5,8,6,3
5,12,13,10
3,2,4,1
```

則輸出應該是

```
26
```

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 3.9 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。一筆測試資料佔 2 分。

第二題

（20 分）承第一題，這次小傑總共安排了 m 種工作順序，分別爲 $s^{(1)}$、$s^{(2)}$ 直到 $s^{(m)}$，請幫小傑計算哪一個工作順序的總延遲時間最小，讓小傑更有機會實現他心心念念的約會。

舉例來說，假設小傑手邊有 4 個工作，依序分別是 1、2、3、4 號工作，這些工作的處理時間依序是 5 小時、4 小時、3 小時、2 小時，完成期限分別是開始工作後的 6 小時、7 小時、8 小時、10 小時。小傑安排的工

作順序有三種，第一種爲 $s^{(1)} = (3, 2, 4, 1)$，第二種爲 $s^{(2)} = (2, 4, 3, 1)$，最後一種爲 $s^{(3)} = (3, 1, 4, 2)$。經過計算後，可以得知以上三種工作順序的總延遲時間分別爲 26 小時、22 小時、20 小時，因此應該要選擇擁有最小延遲時間 20 小時的最後一種工作順序。

請依照題目指示，找出擁有最小總延遲時間的工作順序，並輸出其順序編號及總延遲時間。請注意，如果同時有兩個工作順序都擁有最小的延遲時間，請選擇編號比較小的工作順序。

輸入輸出格式

系統會提供數組測試資料，每組測試資料裝在一個檔案裡。在每個檔案中會有四列資料，第一列含兩個正整數，依序是 n、m；第二列含有 n 個正整數 p_1、p_2 直到 p_n；第三列含有 n 個正整數 d_1、d_2 直到 d_n；第四列到第 $m+3$ 列各含有 n 個正整數，第 $k+3$ 列含有 $s_1^{(k)}$、$s_2^{(k)}$ 直到 $s_n^{(k)}$，其中 $s_i^{(k)}$ 表示第 k 個工作順序當中第 i 個執行的工作。每一列的任兩個數字之間用一個逗點隔開。已知 $1 \le n \le 20$、$1 \le m \le 10$、$1 \le p_i \le 50$、$1 \le d_i \le 1000$、$1 \le s_i^{(k)} \le n$，且當 k 固定後 $s_i^{(k)}$ 彼此之間不重複。

讀入這些資訊後，請依題目指示印出兩個整數，依序代表擁有最小總延遲時間的工作順序編號以及其總延遲時間，兩兩之間以一個逗點隔開。舉例來說，如果輸入是

```
4,3
5,8,6,3
5,12,13,10
3,2,4,1
2,4,3,1
3,1,4,2
```

則因爲每一個順序依序會造成的總延遲時間是 26、22、20，所以輸出應該是

```
3,20
```

如果輸入是

```
4,5
5,4,3,2
6,7,8,10
4,2,3,1
1,4,2,3
4,3,2,1
3,4,2,1
1,3,2,4
```

則因爲每一個順序依序會造成的總延遲時間是 9、10、10、10 和 9，雖然第 1 號工作順序與第 5 號工作順序都同時擁有最小總延遲時間，但我們優先輸出編號較小的，所以輸出應該是

```
1,9
```

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 3.9 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。一筆測試資料佔 2 分。

第三題

（60 分）承第一題和第二題，現在小傑有辦法評估多個工作順序中哪個最好了，但他要如何獲得多個順序呢？小傑決定要從一個給定的順序出發，逐步搜尋這個順序「相鄰」的順序，每一輪都移動到這些順序中最好的順序，然後再執行新一輪的搜尋與移動，直到無法移動爲止。這樣的演算法有個名稱，叫做「局部搜尋」（local search）演算法。在這一題，讓我們來完成這個演算法的其中一個步驟吧！

局部搜尋演算法的核心在於「相鄰」的概念。小傑首先定義所謂「相鄰的工作」如下：在一個工作順序中，序位剛好差一的兩個工作以及頭尾的兩個工作，會被視爲「相鄰」。舉例來說，給定一個工作順序 $s = (1, 2, 4, 3)$，則工作 4 的相鄰工作有工作 2 和工作 3（序位差一），工作 1 的相鄰工作則有工作 2（序位差一）和工作 3（分別爲頭尾）。顯然在一個工作順序中，每個工作都有恰好兩個相鄰的工作。接著小傑定義所謂「相鄰的工作順序」，若兩個順序恰好只差在一對相鄰的工作被對調，那就說這兩個順序是相鄰的。舉例來說，給定一個順序 $s = (1, 4, 3, 2)$，那麼這個順序 s 的相鄰順序有以下四種：

- 調換序位一與序位二的工作，此相鄰順序爲 $(4, 1, 3, 2)$。
- 調換序位二與序位三的工作，此相鄰順序爲 $(1, 3, 4, 2)$。
- 調換序位三與序位四的工作，此相鄰順序爲 $(1, 4, 2, 3)$。
- 調換序位四與序位一的工作，此相鄰順序爲 $(2, 4, 3, 1)$。

顯然給定一個有 n 個工作的工作順序後（$n > 2$），與之相鄰的順序一定是恰好 n 個。最後，請注意「工作間的相鄰」和「順序間的相鄰」雖然都叫相鄰，但是是兩個概念。

在本題中，小傑打算在被給定一個工作順序後，找出這個順序的所有相鄰順序，接著從這些順序中找出哪個最好（會帶來最小的總延遲時間），供自己做選擇。舉例來說，假設小傑手邊有 4 個工作，依序分別是 1、2、3、4 號工作，這些工作的處理時間依序是 5 小時、4 小時、3 小時、2 小時，完成期限分別是開始工作後的 6 小時、7 小時、8 小時、10 小時。若小傑一開始被給定的工作順序爲 $(1, 4, 2, 3)$，則經過計算後，可以得知以上五種工作順序（一開始被給定的順序，以及與之相鄰的四個順序）的總延遲時間，分別爲 9 小時、10 小時、7 小時、10 小時、9 小時，因此最小延遲時間爲 7 小時。

請依照題目指示，找出給定的工作順序及其相鄰順序中，最小的總延遲時間並輸出。

輸入輸出格式

系統會提供數組測試資料，每組測試資料裝在一個檔案裡。在每個檔案中會有四列資料，第一列含有一個正整數 n；第二列含有 n 個正整數 p_1、p_2 直到 p_n；第三列含有 n 個正整數 d_1、d_2 直到 d_n；第四列含有 n 個正整數 s_1、s_2 直到 s_n。每一列的任兩個數字之間用一個逗點隔開。已知 $1 \le n \le 20$、$1 \le p_i \le 50$、$1 \le d_i \le 1000$、$1 \le s_i \le n$，且 s_i 彼此之間不重複。

讀入這些資訊以後，請依題目指示印出一個整數，代表給定的工作順序及其相鄰順序中，最小的總延遲時間。舉例來說，如果輸入是

```
4
5,4,3,2
6,7,8,10
1,4,3,2
```

則因爲給定的順序會造成的延遲時間是 9，而相鄰的順序 $(4,1,3,2)$、$(1,3,4,2)$、$(1,4,2,3)$ 和 $(2,4,3,1)$ 分別會造成的延遲時間分別是 10、7、10 和 9，所以輸出應該是

```
7
```

如果輸入是

```
4
5,8,6,3
5,12,13,10
3,2,4,1
```

則因爲給定的順序會造成的延遲時間是 26，而相鄰的順序 $(2,3,4,1)$、$(3,4,2,1)$、$(3,2,1,4)$ 和 $(1,2,4,3)$，分別會造成的延遲時間分別是 25、22、28 和 16，所以輸出應該是

```
16
```

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 3.9 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：

- 確定可以使用的語法包含 `for`、`while`、各種維度的清單、Python 內建的所有操作清單的函數（包含參數只有一個清單的函數，以及由清單後面加一個「點」去呼叫的函數），以及之前作業說過可以使用的語法。
- 確定不可以使用的語法包含自定義函數、tuple、dictionary、`print` 中沒教過的格式化輸出法（例如百分比、f-string、str.format()）、類別等等。

請注意正面表列的固然是都確定可以用，但沒有被負面表列的不表示可以用喔！

評分原則

- 這一題的其中 40 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）以及是否使用上課沒學過的方法。請寫一個「好」的程式吧！

第四題（額外加分題）

特別說明：這一題顯然是本學期目前爲止最難的一題。在作業三的此刻，同學們已經擁有完成他所需的知識、技能了，但善用這些知識、技能解決這種規模的問題絕非易事。爲了不要造成大家過多的負擔，首先我們把這一題拆成兩題（第三題、第四題），並且把第四題設成加分題，讓現在只能寫出第三題的同學也能拿到基本上全部的分數，也讓有興趣挑戰自己的同學可以寫第四題拿到額外分數（並且使用任何你覺得合適的方法）。其次，這一題在期中考後的作業四還會再次出現，換言之就是如果有位同學到時候才有辦法寫出來，也是符合課程期待的。最後，先讓大家知道，小考和第一次期中考不會有像第四題這麼難的題目，請大家安心；但不排除有像第三題這麼難的，請大家還是要好好練習、做足準備。

（20 分）承前三題，結合以上問題，小傑設計了一套演算法，來選出可以擁有最小總延遲時間的工作順序。本題與第一題、第三題相同，會給定工作的數量、處理時間、延遲時間與一個初始的工作順序 $s^{(0)}$。指定的演算法的步驟如下：

1. 我們會使用迴圈進行很多回合的工作順序篩選，第 k 回合的起始工作順序叫 $s^{(k)}$。最一開始的回合被稱爲第零回合，指定的工作順序將從 $s^{(0)}$ 開始。
2. 在第 k 回合，首先找出 $s^{(k)}$ 的 n 個相鄰順序，並且計算以上 $n+1$ 個工作順序（$s^{(k)}$ 以及它的相鄰順序）各自的總延遲時間。接著找出以上工作順序中擁有最小總延遲時間的工作順序，稱之爲 s^*，及其總延遲時間。
3. 如果 s^* 的總延遲時間跟 $s^{(k)}$ 的總延遲時間一樣，代表這一輪沒有找到更好的工作順序，那就中斷迴圈，把 $s^{(k)}$ 當作最終得到的工作順序。如果 s^* 的總延遲時間小於 $f(s^{(k)})$，代表找到更好的工作順序了，那就將 s^* 做爲下一輪的的起始順序 $s^{(k+1)}$，並回到步驟二進行下一回合的搜尋。

舉例來說，假設小傑手邊有 4 個工作，依序分別是 1、2、3、4 號工作，這些工作的處理時間依序是 5 小時、4 小時、3 小時、2 小時，完成期限分別是開始工作後的 6 小時、7 小時、8 小時、10 小時。若小傑一開始被給定的工作順序爲 $s^{(0)} = (1, 4, 3, 2)$，總延遲時間爲 9。則演算法實作的過程如下。

首先，找出 $s^{(0)}$ 的相鄰工作順序，並且計算各自的總延遲時間：

- 第一個相鄰順序是 $(4, 1, 3, 2)$，總延遲時間爲 10 小時。
- 第二個相鄰順序是 $(1, 3, 4, 2)$，總延遲時間爲 7 小時。
- 第三個相鄰順序是 $(1, 4, 2, 3)$，總延遲時間爲 10 小時。
- 第四個相鄰順序是 $(2, 4, 3, 1)$，總延遲時間爲 9 小時。

在這些相鄰順序中，$(1,3,4,2)$ 擁有最小的總延遲時間 7 小時，因此我們將這個順序選爲新的工作順序，並繼續尋找其相鄰順序進行搜尋。

在第二輪中，對於新的順序 $s^{(1)} = (1,3,4,2)$，我們再次尋找其相鄰順序並計算總延遲時間：

- 第一個相鄰順序是 $(3,1,4,2)$，總延遲時間爲 9 小時。
- 第二個相鄰順序是 $(1,4,3,2)$，總延遲時間爲 9 小時。
- 第三個相鄰順序是 $(1,3,2,4)$，總延遲時間爲 9 小時。
- 第四個相鄰順序是 $(2,3,4,1)$，總延遲時間爲 8 小時。

在這一輪中，所有相鄰順序都沒有比目前最佳的順序 $s^{(1)} = (1,3,4,2)$ 好，因此跳出迴圈，小傑會選擇 $(1,3,4,2)$ 爲他最終要執行的的工作順序。

以上演算法如果寫成 puesdocode，將會長得像下面這樣，其中爲了方便說明，「計算單一工作順序 s 的總延遲時間」被描述成一個函數 f：[2]

```
Let s_0 be the initial job sequence.
Let k be 0.
while true:
    Find all neighbors of s_k. Put them in N_k.
    For all s in N_k:
        Calculate f(s).
    Find the best sequence s* such that f(s*) <= f(s) for all s in N_k.

    if f(s*) == f(s_k):
        Break the loop. Report s*.
    else:
        Let k become k + 1.
        Let s_k become s*.
```

請幫助小傑實作這套演算法，並完整輸出最佳的工作順序，以及其延遲時間。請注意，如果同時有兩個工作順序都擁有最小的延遲時間，請選擇編號比較小的工作順序。

輸入輸出格式

系統會提供數組測試資料，每組測試資料裝在一個檔案裡。在每個檔案中會有四列資料，第一列含有一個正整數 n；第二列含有 n 個正整數 p_1、p_2 直到 p_n；第三列含有 n 個正整數 d_1、d_2 直到 d_n；第四列含有 n 個正整數 $s_1^{(0)}$、$s_2^{(0)}$ 直到 $s_n^{(0)}$。每一列的任兩個數字之間用一個逗點隔開。已知 $1 \le n \le 20$、$1 \le p_i \le 50$、$1 \le d_i \le 1000$、$1 \le s_i^{(0)} \le n$，且 $s_i^{(0)}$ 彼此之間不重複。

讀入這些資訊以後，請依題目指示，印出最佳的工作順序，兩兩之間以一個逗點隔開，再輸出一個分號，最後印出最佳工作順序的延遲時間。舉例來說，如果輸入是

[2]在 Python 寫「函數」是期中考後的內容，現在如果不會寫，就用沒有函數的方式完成就好。

```
4
5,4,3,2
6,7,8,10
1,4,3,2
```

則輸出應該是

```
1,3,4,2;7
```

如果輸入是

```
4
5,8,6,3
5,12,13,10
3,2,4,1
```

則輸出應該是

```
1,4,3,2;11
```

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 3.9 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。一筆測試資料佔 2 分。