商管程式設計（112-2）
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/`）爲第一、二、三題各上傳一份 Python 3.9 原始碼（以複製貼上原始碼的方式上傳）。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 ***4 月 2 日中午十二點***。在你開始前，請閱讀課本的第十章[1]。爲這份作業設計測試資料並且提供解答的助教是林小喬。

第一題

特別說明：延續作業三的第四題，此處將完整呈現該題題目，情境與條件完全相同，大家可以理解爲期待大家再練習一下（如果當時沒寫出來），也可以理解爲我們幾乎就想要送大家 20 分（當然大家還是有一些工作要做）。此外，我們也要透過此題，讓大家稍稍地練習一下用模組化的精神來實作這一題。

（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 10 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。

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

爲了解決這個問題，小傑設計了以下的演算法。他會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。

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。

現在，在給定工作的數量、處理時間、延遲時間與一個初始的工作順序 $s^{(0)}$ 後，小傑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 pseudocode，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，並找出最佳的工作順序，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。資料以什麼方式被讀取你不需要知道；在本題中，爲了練習模組化，助教會先在 PDOGS 上放好「資料讀取模組」的程式碼，先把題目輸入存到 `job_cnt`、`processing_times`、`due_times`、`init_seq` 這一個整數和三個 Python 的 list（清單）中：

- `job_cnt` 存著一個正整數，資料型別爲 `int`，代表工作的個數。
- `processing_times` 存著裝著 n 個 `int` 的 `list`，資料型別爲 `list`，其中 `processing_times[i]` 代表工作 $i+1$ 的處理時間。

[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。

- due_times 存著裝著 n 個 int 的 list，資料型別爲 list，代表工作的完成期限，其中 due_times[i] 代表工作 $i+1$ 的完成期限。
- init_seq 存著裝著 n 個 int 的 list，資料型別爲 list，代表初始給定的工作順序，其中 init_seq[i] 代表第 $i+1$ 個要被執行的工作的編號。

再次提醒大家，助教已經在 PDOGS 上放好讀取輸入資料的程式碼了，所以大家只需要實做並且上傳後續的程式碼，PDOGS 會把助教寫的程式跟你寫的程式拼在一起直譯、執行。換個角度講，大家也只能上傳你該寫的部分，如果你的程式碼中還有 input() 等等不該出現的指令，PDOGS 就會因此拼出重複的程式碼，結果反而就會無法直譯跟執行了。

請你在寫這題的時候，相信 PDOGS 上已經有人幫你寫好正確無誤的「資料讀取模組」，並直接從這些變數中讀取資料、進行演算、依照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

```
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。

第二題

(40 分) 有一家銀行要邀請自家的存款戶都來辦一張新的信用卡，並且要針對每個客戶給予不同優惠 (但也有可能沒有優惠)。題目會給定一共有 n 個客戶以及 m 種優惠 (例如辦卡就送 10 杯便利商店的咖啡是一種優惠、辦卡且首月刷滿 999 就送一個行李箱是另一種優惠等等)；請注意，m 種優惠加上沒有優惠 (編號爲 0)，就代表著總共有 $m+1$ 種「優惠」(只是其中一種「優惠」跟沒有一樣)。

資料分析師已經透過過往資料，分析出每位客戶對於每一種優惠的喜好，並把這些資訊量化成一個機率 $p_{ij}\%$，代表客戶 i 看到優惠 j 後會來辦卡的機率。此外，每種優惠是有人數限制的，第 j 種優惠只能發給至多 K_j 位不同的客戶 (不論他們是否來申請)，而第 0 種優惠的上限 K_0 相當於是 n (因爲我們也可以選擇全部的客戶都不發送優惠)。爲了最大化客戶來辦卡的機率，資料分析師們也另外提供了一份清單，列出哪些客戶不該被發送優惠。我們用 $x_i = 1$ 表示第 i 個客戶不該被發送邀請，而 $x_i = 0$ 表示第 i 個客戶可以被發送邀請。已知至少有一個 x_i 的值爲 0，表示至少有一個客戶還可以被發送邀請。

根據以上的條件，現在請你寫一個程式，從還有剩餘名額的優惠 ($K_j \geq 1$) 和該被發送優惠 ($x_i = 0$) 的客戶中，去找出一個優惠去發給一個客戶，目標爲最大化這一個優惠發出去後，那一個客戶會來辦卡的機率。如果有數種配對都有最大的成功機率，那就選擇這些配對之中優惠剩餘名額最多的；如果仍平手，那就選擇優惠編號最小的；仍再次平手，那就給客戶編號最小的。

舉例來說，我們要從 3 位客戶、$2+1$ 種優惠 (包含第 0 種優惠，即不發送優惠) 中選擇最大成功機率的配對。已知第一位客戶看到 3 種優惠會來辦卡的機率依序是 44%、31%、41%；第二位客戶的機率依序是 17%、45%、23%；第三位客戶的機率依序是 23%、65%、44%。三種優惠的剩餘數量依序是 3、2、2。最後，不該被發送優惠的清單中依序紀錄著 0、1、0，代表第二位客戶不應被考慮是否要發送優惠。因此，我們只需要考慮第一與第三位客戶要發送哪一種優惠，經檢查與計算我們可以發現，三種優惠都剩餘足夠的數量，而其中可以最大化辦卡機率的組合爲發送給第三位客戶第一種優惠 (優惠編號是由 0 開始計算的)，其辦卡機率爲 65%。

在這個例子中，如果三種優惠的剩餘數量依序是 3、0、2，表示第一種優惠已經沒有餘額了，那可行的配對中就有兩個配對同爲最佳：把第零種優惠給第一位客戶，以及把第二種優惠給第三位客戶，辦卡機率都是 44%。此時按照規定，我們應該選擇優惠剩餘名額最多的第零種優惠，因此把第零種優惠發送給第一位客戶。如果三種優惠的剩餘數量依序是 3、0、3，表示優惠零跟優惠二的剩餘名額一樣，那按照題目的規定，也應該選擇優惠編號較小的優惠零給第一位用戶。請注意雖然在本段的兩個例子中，結果都是選擇優惠零，但判斷的過程其實是不同的。

請依照題目指示，輸出一項要發送優惠的客戶編號以及優惠編號。

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 $n+3$ 列資料，第一列含有兩個正整數，依序爲 n、m；第二列到第 $n+1$ 列，每列有 $m+1$ 個正整數 $p_{i,0}$、$p_{i,1}$ 直到 p_{im}；第 $n+2$ 列含有 m 個正整數 K_1、K_2 直到 K_m；第 $n+3$ 列含有 n 個正整數 x_1、x_2 直到 x_n。每一列的任兩個數字之間用一個逗點隔開。已知 $1 \leq n \leq 20$、$1 \leq m \leq 5$、$0 \leq p_{ij} \leq 100$、$0 \leq K_j \leq n$、$x_i \in \{0,1\}$，且至少有一個 x_i 的值爲 0。

讀入這些資訊以後，請依題目指示印出兩個整數，依序爲要發送優惠的客戶編號 i，以及優惠編號 j，兩兩之間以一個逗點隔開。舉例來說，如果輸入是

```
3,2
44,31,41
```

```
17,45,23
23,65,44
2,2
0,1,0
```

則輸出應該是

```
3,1
```

如果輸入是

```
3,2
44,31,41
17,45,23
23,65,44
0,2
0,1,0
```

則輸出應該是

```
1,0
```

如果輸入是

```
5,3
10,38,21,98
7,89,65,34
12,56,88,76
11,24,87,98
15,99,35,44
2,3,1
0,0,0,0,1
```

則有兩個配對（客戶一、優惠三；客戶四、優惠三）都能達到最大的辦卡機率 98%，此時按照規則應該選擇客戶編號最小的，因此輸出應該是

```
1,3
```

請注意在這個例子中，雖然把優惠一發送給客戶五可以達到更高的辦卡機率（99%），但因爲客戶五不應該被發送優惠，所以答案還是把優惠三發送給客戶一。

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、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 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）以及是否使用上課沒學過的方法。請寫一個「好」的程式吧！

第三題

（40 分）承上題，現在我們要幫所有客戶都提供一種優惠，亦即題目不會再給定「列出哪些客戶不該被發送優惠」的清單了。在最大化總辦卡機率的目標下，我們可以採用一個簡單的貪婪演算法來產出一個優惠發送方案[3]，步驟如下：

- 初始化一個清單，記錄所有該被發送優惠之客戶的編號（一開始裡面會存所有客戶的編號，表示所有客戶都該被發送優惠）。
- 每輪都使用第二題的程式碼，挑選一個擁有最大辦卡機率的客戶與優惠種類配對。
- 挑好之後，將這個已經發送優惠的客戶從清單中移除，並把該優惠的剩餘數量減一，接著進行下一輪挑選。
- 重複執行，直到所有用戶都被發放一項優惠爲止。

對於那個要記錄「還需要被發送優惠的客戶編號」的清單，大家可以自己選擇自己喜歡的實作方式。舉例來說，可以如上所述裝許多客戶編號，每當過了一輪我們發送了一個優惠給某個客戶，就從清單中把該客戶的編號刪掉，如此直到該清單變成空的，就可以結束這個演算法。又或者也可以讓這個清單在一開始時是 n 個 0，之後每發送一個優惠給某個客戶，就把相對應的 0 改成 1，當整個清單都是 1 的時候演算法就可以結束了。

讓我們舉個例子說明。如果我們要從 3 位客戶、$2+1$ 種優惠（包含第 0 種優惠，即不發送優惠）中選擇最大成功機率的配對。已知第一位客戶看到 3 種優惠會來辦卡的機率依序是 10%、31%、51%；第二位客戶的機率依序是 17%、17%、51%；第三位客戶的機率依序是 23%、65%、44%。三種優惠的剩餘數量依序是 3、2、1。

[3]這套貪婪演算法，其實並不一定能保證眞的可以最大化客戶們的辦卡機率，大多時候都是獲得一個近似最佳解的答案而已，但在本課程中，我們期望地是大家可以正確地實作指定的演算法。如果你很想知道該怎麼才能獲得這個問題的最佳解，推薦修習「作業研究」或「管理科學模式」、「最適化方法」等課程。

- 在第一輪我們可以發現，三種優惠都剩餘足夠的數量，而其中可以最大化辦卡機率的組合爲發送給第三位客戶第一種優惠（優惠編號是由 0 開始計算的），其辦卡機率爲 65%；
- 在第二輪，三種優惠也都剩餘足夠的數量，而其中可以最大化辦卡機率的組合爲發送給第一位客戶第二種優惠，其辦卡機率爲 51%。請注意雖然發送給第二位客戶第二種優惠也能最大化辦卡機率，但按照規定我們選編號最小的客戶。
- 在第三輪，第二種優惠已經沒有餘額了，且第零和第一種優惠中都可以最大化僅剩的客戶二的辦卡機率，此時按照規定我們應該選擇優惠剩餘名額較多的優惠零，其辦卡機率爲 17%。

綜合以上，最終我們依序會發送給第一、二、三位客戶第二種優惠、第零種優惠與第一種優惠。

請根據題目指示實作指定的演算法，最後按照客戶編號由小到大，依序輸出要發送給每一位客戶的優惠種類編號。

特別說明：在實作這題時，一個很合適的作法是將第二題的程式碼拿來包成一個函數，並在這題中反覆呼叫，以達到模組化的效果。如果你還不知道什麼叫函數，就請不用這麼做，用你能完成的方式完成就好（當然，你第二題的程式碼還是應該要被利用）；我們也鼓勵你先預習 Coursera 上第二門課的第一週「函數」的影片，然後試著使用（當然這是非必要的）。總之，本題只看正確性並且允許使用任何語法，請自行決定怎麼實作吧！

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 $n+2$ 列資料，第一列含有兩個正整數，依序爲 n、m；第二列到第 $n+1$ 列，每列有 $m+1$ 個正整數 $p_{i,0}$、$p_{i,1}$...$p_{i,j}$ 直到 $p_{i,m}$；第 $n+2$ 列含有 m 個正整數 K_1、K_2 直到 K_m 。每一列的任兩個數字之間用一個逗點隔開。已知 $1 \leq n \leq 20$、$1 \leq m \leq 5$、$0 \leq p_{i,j} \leq 100$、$1 \leq K_j \leq n$。

讀入這些資訊以後，請依題目指示印出 n 個整數，第 i 個數字爲要發送給編號 i 的客戶的優惠編號 j，兩兩之間以一個逗點隔開。舉例來說，如果輸入是

```
3,2
10,31,51
17,17,51
23,65,44
2,1
```

則輸出應該是

```
2,0,1
```

如果輸入是

```
5,3
10,38,21,98
7,89,65,34
12,56,88,76
11,24,87,90
15,70,35,44
```

```
2,3,1
```

則輸出應該是

```
3,1,2,2,1
```

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 3.9 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。