商管程式設計（106-2）
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/judge/`）為第一到三題各上傳一份 Python 3 原始碼（以複製貼上原始碼的方式上傳）。第三題是加分題。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 ***2018 年 4 月 2 日凌晨一點***。為這份作業設計測試資料並且提供解答的助教是方心成。

第一題

（40 分）有一個生產排程的問題如下。有 n 個工作必須被分配給 m 個機臺，而由於機臺有新舊之分，因此也有效率上的差別，亦即機臺 i_1 完成工作 j 的時間，可能不同於機臺 i_2 完成工作 j 的時間。我們用 t_{ij} 來表示機臺 i 完成工作 j 的時間，並且已知 $t_{ij} = q_i p_j$，其中 q_i 是機臺的效率係數，愈高表示速度愈慢，而 p_j 表示機臺的工作量，愈大表示工作量愈高。一個工作只能被交給恰好一個機臺。我們的目標是分配完工作之後，可以最小化「最晚完成所有工作的機臺」完成最後一個工作的時間，也就是所謂的 makespan。更精確地說，如果 $x_{ij} = 1$ 表示工作 i 被分給機臺 j，反之則為 0，那麼分完所有工作之後的 makespan 即為

$$\max_{i=1,...,m} \left\{ \sum_{j=1}^{n} q_i p_j x_{ij} \right\}。$$

我們預計使用的演算法如下。首先，我們將工作依照其工作量 p_j 由大到小排序。接著我們將排序後的工作一個一個分配給機臺，每次要分配一個工作時，我們就逐一嘗試每個機臺，看看「如果分給此機臺，則到此工作為止的 makespan」是多少，然後選使此刻 makespan 最小的那個機臺。更精確地說，假設我們想分配工作 j 時，各機臺的累積總工時（是總工時，不是總工作量）是 T_1、T_2 直到 T_m，則我們所選的機臺 i^* 應該滿足

$$\max\{T_1, T_2, ..., T_{i^*} + q_{i^*} p_j, ..., T_m\} \leq \max\{T_1, T_2, ..., T_i + q_i p_j, T_m\} \quad \forall i = 1, ..., m。$$

如果有複數個機臺都滿足此條件，我們選分配前累積總工時 T_i 最小的；如果還有複數個機臺可選，則在其中我們選編號最小的。我們持續這樣分配直到工作都被分完為止。

舉例來說，假設有 $m = 3$ 個機臺要做 $n = 5$ 個工作，且 $q_1 = q_2 = 1$、$q_3 = 2$、$p_1 = 3$、$p_2 = 5$、$p_3 = 11$、$p_4 = 7$、$p_5 = 4$，則排程過程如下。首先，工作被依照工作量排序為 11、7、5、4、3，接著 11 被分配機臺 1、7 被分配機臺 2、5 被分配機臺 3、4 被分配機臺 2，最後 3 被分配機臺 1。請注意在最後一個步驟，雖然當下機臺 3 的總工時（10）最小，但若分給它則總 makespan 會變成 $\max\{11, 7, 16\} = 16$，分給機臺 1 則只有 $\max\{14, 11, 10\} = 14$，所以此工作量為 3 的工作應該被分給機臺 1。

我們想要找出分配完之後由機臺 1 到機臺 m 各自的總工時（即完成最後一個工作的完成時間）。以上例來說，就是 14、11、10。

重要：由於一些作業系統上的限制，在 PDOGS 上要用空白字元做字串切割，應該用 `split()` 而非 `split(' ')`。如果要用其他字元做分隔，則可以照正常作法進行，例如要用逗點隔開時，就寫 `split(',')`。之後的作業與考試我們將不再持續提醒此事。

輸入輸出格式

系統會提供一共 20 組測試資料，每組測試資料裝在一個檔案裡。每個檔案含有三行，第一行依序含有兩個正整數 m 和 n，第二行依序含有 m 個正整數 q_1、q_2 到 q_m，第三行依序含有 n 個正整數 p_1、p_2 到 p_n。已知 $1 \le n \le 1000$、$1 \le m \le 100$、$1 \le q_i \le 5$、$1 \le p_i \le 100$。一行內任兩個值被一個空白字元隔開。讀入這些值之後，請依照題目指定的規則做工作分配，並依序印出機臺 1 到機臺 m 的總工時，任兩個值中間用一個空白字元隔開。

舉例來說，如果輸入是

```
3 5
1 1 2
3 5 11 7 4
```

則輸出應該是

```
14 11 10
```

如果輸入是

```
4 10
1 1 2 3
9 5 7 12 8 4 3 9 5 1
```

則輸出應該是

```
23 22 20 24
```

其中機臺 1 被分配到的工作的工作量依序是 12、7 和 4、機臺 2 被分配到的工作的工作量依序是 9、8 和 5、機臺 3 被分配到的工作的工作量依序是 9 和 1、機臺 4 被分配到的工作的工作量依序是 5 和 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 3.6 程式碼。當然，你應該寫適當的註解。針對這個題目，你***可以***使用任何方法。

評分原則

這一題的全部 40 分都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。

第二題

（60 分）有一家快遞公司，要從其物流中心送泡麵到 n 個顧客家中。我們把物流中心的位置標號為 0，n 個顧客的家則依序編號為 1、2 到 n。在地點 i 和 j 之間移動要花的最短時間為 d_{ij} 分鐘。我們現在要幫此公司決定送貨路線，以最小化快遞員要花的總時間。由於是送貨，因此車上要裝著要送的貨，而貨車空間又有限，所以可能需要跑複數趟，則總時間即為這複數趟的車行總時間。我們假設所有泡麵都採用一樣的包裝、q_i 是顧客 i 要收到的泡麵包數，而車子最多只能裝 K 包泡麵。因此我們若規劃車子一趟要送貨給若干個顧客，則這些顧客的箱子數總和不能超過 K。

我們使用如下演算法來規劃送貨路線。一個比較容易思考的方式是，我們的車上裝著 K 包泡麵，而面前有 n 個地點有人想吃泡麵，因此我們的車子一抵達地點 i，車上的泡麵就會減少 q_i 包，我們就一直跑一直跑直到車上的泡麵不足以發給任何還有需求之地點的人時，就回我們的據點補充到 K 包，然後再出發。我們從物流中心（位置 0）出發，每當要決定下一個要送貨的對象時，就從所有還沒被送貨且我們車上的剩餘泡麵數能滿足的顧客中，挑選距離我們最近的顧客去送，平手則挑編號小的；如果所有還沒被送貨的顧客的需求，我們車上的剩餘泡麵數都無法滿足，那就回物流中心補貨到 K 包泡麵，然後再出發直到全部送完為止。

舉例來說，假設我們共有 4 個顧客，則所有的距離 d_{ij} 可以被表示成一個矩陣（編號從 0 開始）

$$D = \begin{bmatrix} 0 & 4 & 5 & 7 & 9 \\ 4 & 0 & 6 & 2 & 3 \\ 5 & 6 & 0 & 4 & 2 \\ 7 & 2 & 4 & 0 & 5 \\ 9 & 3 & 2 & 5 & 0 \end{bmatrix},$$

表示 $d_{01} = 4$、$d_{13} = 6$，依此類推。假設 q_1 到 q_4 依序是 20、30、40、10，而我們的車子一次最多裝 $K = 55$ 包泡麵，則根據我們的演算法，我們會這樣規劃：

- 第一趟出車：考慮未被滿足的顧客 1、2、3、4，我們能滿足的是顧客 1、2、3、4，我們去最近的顧客 1，送出 20 包泡麵，車上剩 35 包；接著考慮未被滿足的顧客 2、3、4，我們能滿足的是顧客 2、4，我們去最近的顧客 4，送出 10 包泡麵，車上剩 25 包；接著考慮未被滿足的顧客 2、3，我們不能滿足任何一位，所以回公司補貨。這趟出車總共花了 $d_{01} + d_{14} + d_{40} = 4 + 3 + 9 = 16$ 分鐘。
- 第二趟出車：考慮未被滿足的顧客 2、3，我們能滿足的是顧客 2、3，我們去最近的顧客 2，送出 30 包泡麵，車上剩 25 包；接著考慮未被滿足的顧客 3，我們不能滿足它，所以回公司補貨。這趟出車總共花了 $d_{02} + d_{20} = 5 + 5 = 10$ 分鐘。
- 第三趟出車：我們往返於顧客 3 和公司，這趟出車總共花了 $d_{03} + d_{30} = 7 + 7 = 14$ 分鐘。

綜合所有出車，我們拜訪顧客的順序是 1、4、2、3，而總出車時間是 $16 + 10 + 14 = 40$ 分鐘。

輸入輸出格式

系統會提供一共 20 組測試資料，每組測試資料裝在一個檔案裡。每個檔案含有 $n + 3$ 行，第一行含有兩個正整數 n 和 K，第二行到第 $n + 2$ 行中的第 i 行依序含有 $d_{i-2,0}$、$d_{i-2,1}$ 到 $d_{i-2,n}$，第 $n + 3$ 行則含有 n 個正整數，依序為 q_1、q_2 到 q_n。已知 $1 \le n \le 100$、$1 \le d_{ij} \le 30$、$1 \le K \le 100000$，以及 $1 \le q_i \le K$。一行中任兩個數字用一個空白字元隔開。讀入這些值之後，請依照題目指定的規則規劃出

車與送貨給顧客的順序，先依序印出被拜訪的顧客的編號，最後印出總車行時間。任意兩個值之間用一個空白字元隔開。

舉例來說，如果輸入是

```
4 55
0 4 5 7 9
4 0 6 2 3
5 6 0 4 2
7 2 4 0 5
9 3 2 5 0
20 30 40 10
```

則輸出應該是

```
1 4 2 3 40
```

如果輸入是

```
4 1000
0 4 5 7 9
4 0 6 2 3
5 6 0 4 2
7 2 4 0 5
9 3 2 5 0
20 30 40 10
```

則輸出應該是

```
1 3 2 4 21
```

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 3.6 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運算邏輯、可讀性，以及可擴充性。請寫一個「好」的程式吧！

第三題（bonus）

（20 分）承上題，現在我們想要改一下演算法以及輸出的規定。當我們在考慮還沒被送貨的顧客時，第二題的演算法是要求「如果不能一次滿足，就不要送貨給該顧客」，現在我們假設部份送貨是可以的[1]，因此在考慮所有還未被滿足的顧客時，我們就直接挑最近的顧客去送貨，如果車上的泡麵不足以滿足這個顧客，那這個顧客就繼續維持在未被滿足的狀態，只是他的需求量要相對應地變少。

以第二題的例子來說，若使用新演算法，我們規劃如下：

- 第一趟出車：考慮未被滿足的顧客 1、2、3、4，我們去最近的顧客 1，送出 20 包泡麵，車上剩 35 包；接著考慮未被滿足的顧客 2、3、4，我們去最近的顧客 3，送出 35 包泡麵，顧客 3 還有 5 包待送，我們回公司補貨。這趟出車總共花了 $d_{01} + d_{13} + d_{30} = 4 + 2 + 7 = 13$ 分鐘。
- 第二趟出車：考慮未被滿足的顧客 2、3、4，我們去最近的顧客 2，送出 30 包泡麵，車上剩 25 包；接著考慮未被滿足的顧客 3、4，我們去最近的顧客 4，送出 10 包泡麵，車上剩 15 包；接著考慮未被滿足的顧客 3，我們去送 5 包給他，任務完成回公司。這趟出車總共花了 $d_{02} + d_{24} + d_{43} + d_{30} = 5 + 2 + 5 + 7 = 19$ 分鐘。

綜合所有出車，我們拜訪顧客的順序是 1、3（第一次）、2、4、3（第二次），而總出車時間是 $13 + 19 = 32$ 分鐘。

本題的輸入格式和第三題一模一樣，輸出方面則稍微複雜些。我們一樣先依照送貨順序印出被拜訪的顧客，兩個顧客間用一個空白字元隔開，但若兩個顧客之間我們有回公司補貨，則用一個逗點取代一個空白字元，另外在最後一個拜訪的顧客與總出車時間之間，請印出一個分號而非一個空白字元。

舉例來說，如果輸入是

```
4 55
0 4 5 7 9
4 0 6 2 3
5 6 0 4 2
7 2 4 0 5
9 3 2 5 0
20 30 40 10
```

則輸出應該是

```
1 3,2 4 3;32
```

針對這個題目，你***可以***使用任何方法。這一題的 20 分都根據程式運算的正確性給分，一筆測試資料佔 2 分。

[1] 這樣會打擾顧客，可能要付出額外的賠償成本，所以實務上我們未必會這麼做。