程式設計（107-1）
作業十一

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/`）為第一題做同儕互評，再為第二至四題各上傳一份 C++ 原始碼（以複製貼上原始碼的方式上傳）。第四題是加分題。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。請以英文或中文作答。

這份作業的截止時間是 ***2017 年 12 月 29 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莊日陞。

第一題

（0 分）請在 PDOGS 上批改你被隨機分配到的作業十第四題的程式碼，根據它在正確性以外的部份給它 1 至 5 分的評分，並且說明你給分的依據。建議在評分時參考以下六個面向。在前五個面向上，一個面向上做得好就得一分，還不錯則半分，不好則零分；在第六個面向上則在有必要時扣分。六個面向的分數合計後無條件進入即為你最後給的總分。

- 可讀性：變數與函數名稱是否具有合適的資訊量？程式碼排版是否良好且具有前後一致性？是否有合適的註解？關於註解，當然不需要每一行都有註解，但若你發現在某一大段落裡都沒有註解，或某個你感覺很不易看懂的部份沒有註解，你可以指出來；不要直接說「註解太少」但沒有說是哪邊缺乏註解。
- 模組化程度：是否有宣告合適的函數與 class？是否有避免將非常類似的程式片段寫複數次而非寫成函數？是否有避免一個函數做非常多事情？函數間是否有合適的 decoupling？直接閱讀 main function 是否能很快地理解程式在大方向上的運算邏輯？
- 效率：程式運算是否有合理的運算效率？當然我們不要求每個同學都寫出超級有效率的精妙演算法，但至少一個程式不應該進行過多不必要的運算，也不應該耗用過多不必要的記憶體空間。如果你看不出這個程式的效率有明顯的問題，我們建議你直接給一分。
- 擴充性：當要解的問題變得更複雜的時候，我們能不能簡單地修改這個程式以解決新的問題，而不是寧可砍掉重練？這個議題當然也很主觀，所以如果你不能明確地指出在怎樣的新問題上，這個程式會有擴充性問題，我們建議你直接給一分；如果你不能指出很嚴重的問題，我們建議你至少給半分。但對批改者來說，這個關於擴充性的思考其實是很好的訓練。試試看吧！
- 其他：如果有任何其他令你想扣分的理由，請明確地寫出來並且在這個面向上扣分；沒有的話就給一分。
- 題目規範：你應該檢查那份程式碼有沒有違反題目的規範，如果有（例如題目說不可以用上課沒教過的東西，但他用了，或者題目說一定要用指標和動態記憶體配置，但他沒用），就給他零分。當然，請明確地指出他哪邊違反了題目的規範。

該題其中 10 分取決於檢視你的程式碼的同學給你的分數總和（必要時助教會出來主持公道，請不用緊張），另外 10 分取決於你對同學的程式碼的評語和評分的合理性和建設性（原則上除非被申訴，且助教檢視後發現你確實評得很不公允，否則只要有評就會得到 10 分）。

第二題

（40 分）針對以下十題是非題，我們會使用 PDOGS 自動批改，因此請寫一個 C++ 程式，內容就是先讀入一個整數，若讀入的數字為 i，則印出第 i 小題的答案，若為是則印出 1、若為否則印出 0。舉例來說，如果題目只有四題，且你認為答案依序是是、否、是、是，則你上傳的程式碼應該是

```
#include<iostream>
using namespace std;

int main()
{
  int problem = 0;
  cin >> problem;
  if(problem == 1)
    cout << 1;
  else if(problem == 2)
    cout << 0;
  else if(problem == 3)
    cout << 1;
  else
    cout << 1;

  return 0;
}
```

PDOGS 會餵給你的程式的，一定是 1、2 直到 10 這十個整數。有別於作業中一般的程式題，本題在你上傳程式碼時，測試資料是還沒有放上 PDOGS 的，助教會等作業截止後才上傳測試資料（和答案）到 PDOGS 並重新批改此題。換言之，你上傳程式碼時是不會顯示你得幾分的，更不會顯示你對或錯哪些筆測試資料。你會看到你得 0 分，但此數字在助教重新批改之後就會被更新成正確的分數了。

以下題目如果沒有特別指名，請用 C++ 為基準作答。若你看到一段程式碼，請假設他們是被寫在一個有良好且必備的 include 敘述、using namespace 敘述的程式的結構正確的 main function 裡面。

(a) C++ 是編譯語言、Python 是直譯語言。

(b) C++ 是 strong typing 語言、Python 是 weak typing 語言。

(c) C++ 中宣告變數通常都有預設的初始值，Python 則通常沒有。

(d) 與 Python 相比，C++ 較為「低階」。

(e) 與 Python 相比，在小型程式方面 C++ 的開發時間通常較長。

(f) Python 的發明人是因為聖誕節和女友分手，傷心之餘發明了 Python。

(g) Bubble sort 的執行時間，預期是 insertion sort 的兩倍。

(h) 一個 binary tree 中非為 leaf 的 node 一定恰好有兩個 child node。

(i) 只要 $i < j$，一個 min heap 中第 i 層的 node 的值，一定比第 j 層的 node 的值小。

(j) 一個有 k 個 node 的 min heap 最多只有 $\lceil \log k \rceil$ 層。

小提醒：在 PDOGS 上面讓大家繳交此題的地方，會有兩組「與上面正式要計分的題目完全無關的」範例輸入輸出，純粹是用來讓大家確認自己那個被批改的 `if-else` 程式是可以被正確執行的。請確認你的程式在針對範例輸入輸出做撰寫後，能讓你在這一題得到「Accepted」，接著再去針對要計分的題目把你的正確答案寫上去然後繳交。當然，即使你曾經看到「Accepted」，也不代表你繳交的題目在這題已經得到滿分了。

第三題

（60 分）之前的課程中，我們寫了 `MyVector` 這樣的 class，裡面包含有多維的向量，也可以包含其他資訊。`MyVector` 有什麼商業應用呢？在本題中，我們將用 `MyVector` 來代表客戶、多維的向量代表客戶的一些屬性，來試著實做一個客戶分群演算法。

各種資料探勘（data mining）的工作中，有一種叫做分群（clustering），目標是把一堆資料點分成數群（cluster），讓群內的點盡量「相似」或「靠近」，而群與群之間盡量「不同」或「遠離」。舉例來說，行銷人員經常希望將所有客戶分群，然後針對不同客群提供不同的產品或廣告。假設我們有 10 個客戶，年齡和年收入如表 1 和圖 1 所示。如果要將這些客戶分成兩群，應該要依照年齡分嗎？還是依照年收入分？還是同時使用兩個屬性來分呢？

客戶編號	1	2	3	4	5	6	7	8	9	10
年齡（歲）	21	26	45	31	39	42	21	28	31	27
月收入（千元新臺幣）	38	28	62	51	44	39	27	29	60	54

表 1: 分群的例子：原始資料

要決定怎麼分，就要先定義什麼叫做「好的分群」。很自然地，我們會希望被分成同一群的客戶彼此之間是很像的，而不同群的客戶之間則有一定的差異，不然跟通通分成同一群就沒什麼兩樣了。關於如何定義兩點間「距離」與「相似度」，也有很多方法，其中一個就是使用歐幾里得距離（Euclidean distance），就是大家熟悉的平方和開根號。精確地說，給定兩個點 $x = (x_1, x_2, ..., x_n)$ 和 $y = (y_1, y_2, ..., y_n)$，我們定義他們的距離為

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}。$$

現在我們定義我們的分群問題如下：給定 n 個點，我們要找 k 個「群中心」（cluster center），決定之後，每個點就會被分給離它最近的群中心，所有的點就這麼被分成 k 群。我們的目標是最小化所謂的「群半徑」（cluster radius），定義為「離自己的群中心最遠的那個點到它的群中心的距離」。在資料探勘領域，這個問題就叫做「k-center」問題。

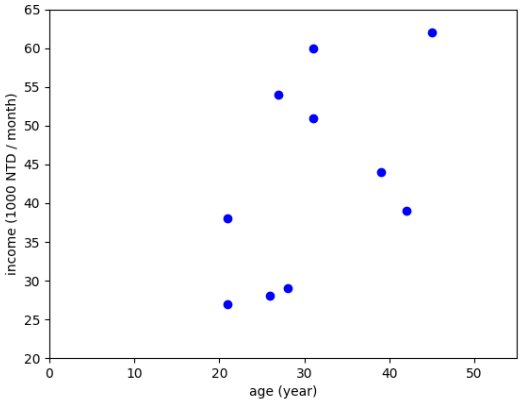

圖 1: 分群的例子：示意圖

聽起來有點像繞口令，讓我們用剛剛的例子說明吧。假設我要分成 $k = 2$ 群，而且我指定客戶 1 跟 9 當群中心，就會得到如圖 2 所示的分群結果，而「離自己的群中心最遠的那個點」是客戶 6，它距離它的群中心 1 的距離是 $\sqrt{(42-21)^2+(39-38)^2} = 21.02$。如果我指定客戶 2 跟 4 當群中心，就會得到如圖 3 所示的分群結果，而「離自己的群中心最遠的那個點」是客戶 3，它距離它的群中心 4 的距離是 $\sqrt{(45-31)^2+(62-51)^2} = 17.8$。根據我們的定義，我們認為以客戶 2 跟 4 為群中心的分群是比較好的。

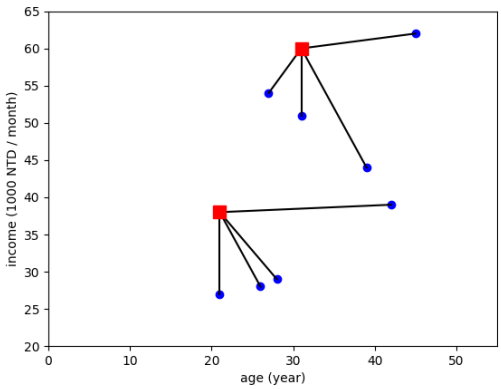

圖 2: 以客戶 1 跟 9 當群中心

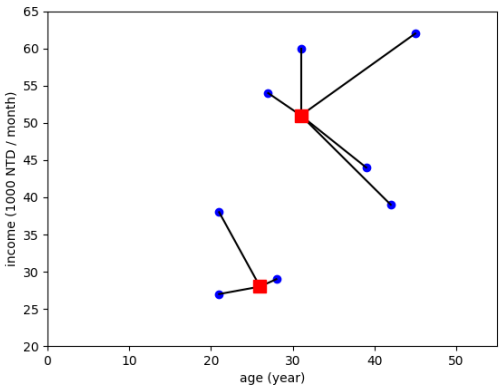

圖 3: 以客戶 2 跟 4 當群中心

給定 n 個點和 k 個群中心，要計算這個分群方式的群半徑顯然不是太困難，而且就算這些點都不是二維的（例如我們可以同時用客戶的身高、體重、年齡、年收入、在本公司的歷史消費總金額等等分群），也依然不困難。不過如果是要找出能最小化群半徑的一組 k 個群中心，那就不容易了。在本題中，我們要請你實作一個很經典的演算法。

這個演算法（的基礎版）很簡單。首先，給定一個點 x 和一個點的集合 S，我們定義這個點到這個集合的距離為

$$d(x,S) = \min_{y \in S} d(x,y)\,,$$

也就是從這個點到這個集合中離它最近的點的距離。現在我們開始執行這個演算法。首先，我們隨便挑一個點當第一個群中心。接下來的每一個回合，就在所有還不是群中心的點之中，根據上述的公式挑

「距離現在的群中心集合最遠」的點當群中心，如果平手就挑編號比較小的。反覆這樣做直到挑滿 k 個點，這個演算法就結束了。

以上面的例子來說，如果我現在要挑 $k = 3$ 個群中心，而且我指定客戶 1 當我的初始群中心，那麼我會依序得到客戶 1、3、6 當我的群中心，分群結果如圖 4 所示，群半徑為 17.08；如果我以客戶 10 為我的初始群中心，我會依序得到客戶 10、7、6 當我的群中心，分群結果如圖 4 所示，群半徑為 19.69。

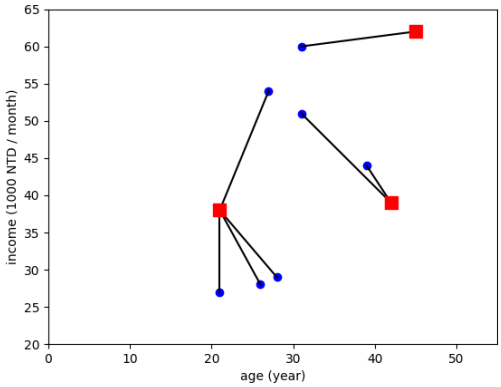

圖 4: 以客戶 1 為初始群中心

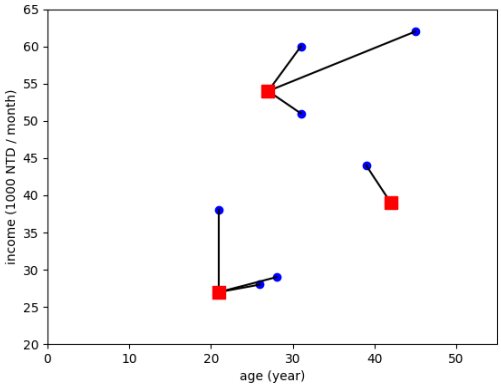

圖 5: 以客戶 10 為初始群中心

當然，這個演算法不保證能得到最佳解，不過他符合直覺，大部分時候得到的解確實也都還不錯，我們甚至可以嚴謹地證明（雖然我們沒有要這麼做）這個演算法得到的群半徑（不論用哪個點當初始點），比起最佳解的群半徑，不會長超過兩倍，也就是再爛也有個極限的意思。像這樣的一個不保證能得到最佳解，但是可以很快地得到一個還不錯的可行解的演算法，被稱為一個啟發性演算法（heuristic algorithm）。對困難的問題（例如資料探勘、機器學習領域的許多問題）設計好的啟發性演算法，是電腦科學家（computer science）和作業研究學家（operations researchers）的工作；我們的工作則是理解這許許多多演算法，並且為我們的應用挑選合適的演算法。而（有些人認為）徹底了解一個演算法最好的方法，就是實作一遍。這或許也是學習程式設計的其中一個好處吧。

在本題中，你就要實作上述的演算法。在仔細描述本題的要求之前，我們再小小地跟你說兩件事。首先，上述的分群問題應用顯然不僅止於行銷。比如說現在政府可能想要找一些地方蓋公園，讓民眾可以運動，而民眾都會去離自己家最近的公園。如何決定把公園蓋在哪裡，以最小化要走最遠才能到公園的民眾的移動距離呢？這顯然就是我們這一題介紹的分群問題了[1]。你可能已經發現，這種公園位置問題特別重視「最『不幸』的那個人的不幸程度」，並試圖最小化這個不幸程度，所以可以說是重視公平性的一個決策問題。有的問題重視公平性，有的重視社會總效用，有個希望兼顧，但共通點在於都是某種最佳化問題，而演算法設計與實作自然是解決這類問題的核心。

最後，回到最初的客戶分群這個行銷問題。雖然我們的演算法可以在給定資料之後做出良好的分群，但要使用客戶的哪些資訊（身高？體重？年齡？月收入？其他？），就是決策者要決定的了。而分出來的群到底有沒有意義，事實上也是個必須回答的問題。只有在「同一群的客戶會傾向於喜歡同類型商品」或「同一群的客戶會傾向於對同一種廣告有反應」，這樣的分群才有意義。這樣的問題，不太容易被電腦程式或演算法回答。這就是大家發揮自己的產業知識與個人專業的時候了[2]。

[1] 如果今天不是要蓋公園，而是要蓋無線網路基地台，那也一樣。

[2] 當然，如果你有這些消費者過往的行為資訊，比如說消費記錄、點擊記錄之類的，那又有別的事情可以做了！

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。在每個檔案中，會有 $n+1$ 列資料，其中第一列是四個整數 n、m、k 和 j，表示要把 n 個 m 維的點分成 k 群，以點 j 當成初始群中心。第 $i+1$ 列是 m 個精確到小數點後第二位的小數 x_1^i、x_2^i 直到 x_m^i，表示編號 i 的點是 $x^i = (x_1^i, x_2^i, ..., x_m^i)$。每一列中的兩個數字都以一個逗點隔開。已知 $1 \leq n \leq 100$、$m = 2$、$1 \leq k \leq n$、$1 \leq j \leq n$、$-1000 \leq x_i \leq 1000$，以及 $-1000 \leq y_i \leq 1000$。讀入資料後，你的程式要用本題指定的演算法，依序找出剩下的群中心，並將所有群中心（包含第一個）依照找到的順序輸出在同一列，兩兩之間以一個逗點隔開。如果遇到平手，就挑其中編號最小的點當下一個群中心。舉例來說，如果輸入

```
10,2,3,1
21.00,38.00
26.00,28.00
45.00,62.00
31.00,51.00
39.00,44.00
42.00,39.00
21.00,27.00
28.00,29.00
31.00,60.00
27.00,54.00
```

則輸出應該是

```
1,3,6
```

如果輸入是

```
10,2,3,10
21.00,38.00
26.00,28.00
45.00,62.00
31.00,51.00
39.00,44.00
42.00,39.00
21.00,27.00
28.00,29.00
31.00,60.00
27.00,54.00
```

則輸出應該是

```
10,7,6
```

你上傳的原始碼裡應該包含什麼

你的.cpp 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 C++ 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，但上課提過的 library 裡面的所有功能都可以用。

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 會編譯並執行你的程式、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 分會在作業十二中被評定。屆時我們會讓同學們互相檢視彼此的本題程式碼，並且就可讀性、易維護性、模組化程度、排版等面向寫評語和給評分（當然一切都是匿名的）。該任務在本題中會佔 20 分，其中 10 分取決於檢視你的程式碼的同學給你的分數（必要時助教會出來主持公道，請不用緊張），另外 10 分取決於你對同學的程式碼的評語和評分的合理性和建設性。若你在本次作業中完全沒有寫這一題，那屆時自然沒有人能檢視你的程式碼，你也就得要損失這 10 分了。

第四題（bonus）

（40 分）當資料量小的時候，好或壞的程式實作可能沒有差別，但資料量大的時候，差別就會顯現出來了。為了給大家多一點挑戰，本題延續第三題，但測試資料的規模會變大許多，其中 $1 \le n \le 50000$、$2 \le m \le 10$、$1 \le k \le \min\{100, n\}$，其餘則都和第三題一樣。如果大家用不夠有效率的方式實作演算法（例如多做了很多沒有必要的運算），即使演算法實作正確無誤，在本題也未必能得到分數。大家挑戰看看吧！

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。