國立臺南大學資訊工程學系

資工三「演算法」課程

第2次作業

**題目: Electirc Vehicle Moter**

|  |  |  |
| --- | --- | --- |
| 班級 | ： | 資工三 |
| 姓名 | ： | 周巧晴 |
| 學號 | ： | S10959042 |

老師：陳宗禧

中華民國 111年10月31日

# 目錄

1. **簡介及問題描述摘要….……………..…………………………………………3**
   1. **簡介…………….…………………………………………………………………………3**
   2. **問題………………………….……………………………………………………………3**
2. **理論分析….………………………..………………………………………4**
3. **演算法則….…………………………..……………………………………8**
4. **程式設計環境架構.………………………..…………………………………24**
5. **程式.…………………………………………..………………………………25**
6. **執行結果、討論與心得.………………………..……………………………54**

參考文獻……………………………………………………………………………65

**(一) 簡介及問題描述摘要**

設計與實作五個演算法，計算Fibonacci Number?….

1. 簡介

**Problem: Electric Vehicle Motor**

給定一地圖，設有*m*個u-motor充電站(還電動機車與充電站)，例：充電站設置在台南古蹟或車站旁等。假設目前有*n*個人騎u-motor(電動機車)，想找充電站，底下有幾個問題要解決：

2. 問題

(1) 哪兩個充電站靠最近? 距離多少? (brute-force algorithm)

(2) 哪兩台電動機車靠最近? 距離多少? (brute-force algorithm)

(3) m個充電站的範圍有多大(Convex-Hull，它的面積以及最遠的距離)? (brute-force algorithm)

(4) n台電動機車的範圍有多大(Convex-Hull，它的面積以及最遠的距離)? (brute-force algorithm)

(5) 假設*n=m*且每一個充電站只有一個空位，要以最省電的方式*n*個人騎u-motor到充電站，如何配對，以及求出最少的總耗能? (Hungarian method)

(6) 如(5)所述，假設*n≠m*，第*i*個充電站空位有𝑣𝑖個, 𝑣𝑖≥0, 1≤𝑖≤𝑚，如何解決這個問題? (Hungarian method or Exhaustive Search)

**Notes:**

(1) 輸入(Input) m個二維的座標充電站與*n*個電動機車的位置，以及充電站空位數𝑣𝑖, 𝑣𝑖≥0, 1≤𝑖≤𝑚。

(2) 請自行定義距離(兩點間)以及電動機車的耗能(假設電動機車每公里耗電量為*x kWh/km*，如0.024 kWh/km，盡可能列出參考文獻)。

**(二) 理論分析**

首先第一個遞迴演算法如下…

1. brute-force algorithm

根據題目要求直接解決問題的方法，通常仰賴於計算能力，嘗試所有可能性。

1. 哪兩個充電站靠最近? 距離多少?

Input：有m個充電站的n大小的地圖

判斷地圖某點是否是充電站，並計算某充電站點與其他充電站點間的距離，再從中找出最小的值，並記錄此兩點的位置。

Time Complexity?

掃一遍地圖判斷充電站在哪需O(n)，找任兩點需O(m2)，所以時間複雜度為O(n)+O(m2)。

1. 哪兩台電動機車靠最近? 距離多少?

Input：有m個電動機車的n大小的地圖

判斷地圖某點是否是電動機車，並計算此電動機車點與其他電動機車點間的距離，再從中找出最小的值，並記錄此兩點的位置。

Time Complexity?

掃一遍地圖判斷電動機車在哪需O(n)，找任兩點需O(m2)，所以時間複雜度為O(n)+O(m2)。

1. m個充電站的範圍有多大(Convex-Hull，它的面積以及最遠的距離)?

Input：有m個充電站的n大小的地圖

判斷其他所有點是否都在任兩點(x1,y1)、(x2,y2)形成的直線的左側或右側。

也就是所有m符合xm(y2-y1)+ym(x1-x2)=>x1y2-y1x2或xm(y2-y1)+ym(x1-x2)<=x1y2-y1x2

將這些點都Mark起來，並找出最長的距離和總面積。

總面積=

0~m是逆時針數的點。

Time Complexity?

掃一遍地圖把充電站找出來需耗時O(n)，找任兩點是O(m2)，判斷此兩點與其他所有點是否都在此兩點形成的直線的左側或右側會再找一遍，所以是O(m3)，排序為逆時針是O(m2)，計算面積是O(m2)，計算任兩點間最遠距離O(m2)，所以總共是O(n)+O(m3)。

1. n台電動機車的範圍有多大(Convex-Hull，它的面積以及最遠的距離)?

Input：有n個電動機車的m大小的地圖

判斷其他所有點是否都在任兩點(x1,y1)、(x2,y2)形成的直線的左側或右側。

也就是所有n符合xn(y2-y1)+yn(x1-x2)=>x1y2-y1x2或xn(y2-y1)+yn(x1-x2)<=x1y2-y1x2

將這些點都Mark起來，並找出最長的距離和總面積。

總面積=

0~n是逆時針數的點。

Time Complexity?

掃一遍地圖把電動機車找出來需耗時O(m)，找任兩點是O(n2)，判斷此兩點與其他所有點是否都在此兩點形成的直線的左側或右側會再找一遍，所以是O(n3)，排序為逆時針是O(n2)，計算面積是O(n2)，計算任兩點間最遠距離O(n2)，是所以總共是O(m)+O(n3)。

2. Hungarian algorithm

列：某電動機車到所有充電站的距離

行：某充電站到所有電動機車的距離

Step1：將每列減去該列最小值。

Step2：將每行減去該行最小值。

Step3：重複動作：從最少0的列開始，遇到第一個0標記起來，並把所有列的該行的0全數丟棄和該列的所有0丟棄。標記完所有0後，判斷是否分配完成。

Step4：若沒完成，則用最少的線將所有0劃掉

Step5：剩下沒被畫的值中找最小值，並將沒被畫的列減掉此最小值。將具有負數的行，加此最小值使負數回到0，回到Step3。

* 1. 假設*n=m*且每一個充電站只有一個空位，要以最省電的方式*n*個人騎u-motor到充電站，如何配對，以及求出最少的總耗能?

根據參考文獻：

Model S Long Range: 5.56 公里/度  
Model 3 Long Range: 5.73 公里/度

取平均值得5.645公里/度

所以假設電動機車每公里耗電量為0.177度

Time Complexity?

Input：一個有m個充電站的n個電動機車的x大小的地圖

先掃一遍地圖找出充電站和電動機車的位置，O(x)。

Step1要掃一遍找每列最小值，做完後再將每列減掉該列最小值，所以是O(m\*n)。

Step2要掃一遍找每行最小值，做完後再將每行減掉該行最小值，所以是O(m\*n)。

Step3最多做n次：要掃一遍找最少0的列，並將第一個0mark起來，做完後再將該行該列的0都丟棄，所以是O(m\*n2)。

Step4用最少的線將行或列劃掉。

1. 將沒有mark的0的列選起來
2. 將已選列的有0的行選起來
3. 將已選行的有mark的0的列選起來
4. 重複2、3直到沒新的選

2、3最多重複n次所以是O(m\*n2)。

Step5要掃一遍找到沒被畫的最小值，並將沒被畫的列減去此值，再將含負數的行加回此值。所以是O(m\*n)。

Step6重複3~5直到step3成功配對

所以時間複雜度是O(x)+O(m\*n2)=O(x)+O(n3)。

* 1. 如(5)所述，假設*n≠m*，第*i*個充電站空位有𝑣𝑖個, 𝑣𝑖≥0, 1≤𝑖≤𝑚，如何解決這個問題?

根據參考文獻：

Model S Long Range: 5.56 公里/度  
Model 3 Long Range: 5.73 公里/度

取平均值得5.645公里/度

所以假設電動機車每公里耗電量為0.177度

Time Complexity?

Input：一個有m個充電站空位的n個電動機車的x大小的地圖

先掃一遍地圖找出充電站空位和電動機車的位置，O(x)。

Step1要掃一遍找每列最小值，做完後再將每列減掉該列最小值，所以是O(m\*n)。

Step2要掃一遍找每行最小值，做完後再將每行減掉該行最小值，所以是O(m\*n)。

Step3最多做n次：要掃一遍找最少0的列，並將第一個0mark起來，做完後再將該行該列的0都丟棄，所以是O(m\*n2)。

Step4用最少的線將行或列劃掉。

1. 將沒有mark的0的列選起來
2. 將已選列的有0的行選起來
3. 將已選行的有mark的0的列選起來
4. 重複2、3直到沒新的選

2、3最多重複n次所以是O(m\*n2)。

Step5要掃一遍找到沒被畫的最小值，並將沒被畫的列減去此值，再將含負數的行加回此值。所以是O(m\*n)。

Step6重複3~5直到step3成功配對

所以時間複雜度是O(x)+O(m\*n2)。

**(三) 演算法則**

1. Brute-Force(Algorithm)

1. 哪兩個充電站靠最近? 距離多少?

Input：有m個充電站的n=a\*b的地圖，x=-1代表電動機車，x>0代表充電站有x-1的空位，x=0代表是空的

Output：靠最近的兩個充電站的座標（單位公里）、距離（單位公里）

for i←0 to a do

for j←0 to b do

if map[i][j]>0

allx[index]←j

ally[index++]←i

for i←0 to indexdo

for j←i+1 to index do

if min>

x←allx[i]

y←ally[i]

x2←allx[j]

y2←ally[j]

min←

1. 演算法時間複雜度(time complexity)

掃一遍地圖判斷充電站在哪需O(n)，找任兩點需O(m2)，所以時間複雜度為O(n)+O(m2)。

1. 演算法空間複雜度(space complexity)

需兩個m大小的陣列存充電站的位置，所以空間複雜度是O(m)。

1. 哪兩台電動機車靠最近? 距離多少?

Input：有m個電動機車的n=a\*b的地圖，x=-1代表電動機車，x>0代表充電站有x-1的空位，x=0代表是空的

Output：靠最近的兩個電動機車的座標（單位公里）、距離（單位公里）

for i←0 to a do

for j←0 to b do

if map[i][j]==-1

allx[index]←j

ally[index++]←i

for i←0 to indexdo

for j←i+1 to index do

if min>

x←allx[i]

y←ally[i]

x2←allx[j]

y2←ally[j]

min←

1. 演算法時間複雜度(time complexity)

掃一遍地圖判斷電動機車在哪需O(n)，找任兩點需O(m2)，所以時間複雜度為O(n)+O(m2)。

1. 演算法空間複雜度(space complexity)

需兩個m大小的陣列存電動機車的位置，所以空間複雜度是O(m)。

1. m個充電站的範圍有多大(Convex-Hull，它的面積以及最遠的距離)?

Input：一個有m個充電站的n=a\*b的地圖，x=-1代表電動機車，x>0代表充電站有x-1的空位，x=0代表是空的

Output：Convex-Hull上的點座標（單位公里）、Convex-Hull的面積（單位平方公里）、任兩點的最遠距離（單位公里）

for i←0 to a do

for j←0 to b do

if map[i][j]>0

x[index]←j

y[index++]←i

for i←0 to index do

for j←i+1 to index do

for k←0 to indexdo

if (y[j] - y[i]) \* x[k] + (x[i] - x[j]) \* y[k] > x[i] \* y[j] - y[i] \* x[j]

ispoint\_small←false

if (y[j] - y[i]) \* x[k] + (x[i] - x[j]) \* y[k] < x[i] \* y[j] - y[i] \* x[j]

ispoint\_big←false

if ispoint\_small || ispoint\_big

for w←0 to index2 do

if pointy[w]==i && pointx[w]==j

exist←false

break

if exist

pointx[index2]←x[i]

pointy[index2++]←y[i]

else

exist←true

for w←0 to index2 do

if pointx[w]==x[j] && pointy[w]==y[j]

exist←false

break

if exist

pointx[index2]←x[j]

pointy[index2++]←y[j]

else

exist←true

ispoint\_small←true

ispoint\_big←true

else

ispoint\_small←true

ispoint\_big←true

for i←0 to index2 do

center[0]←center[0]+pointx[i]

center[1]←center[1]+pointy[i]

center←center/index2

for i←0 to index2 do

for j←0 to index2-i-1 do

if(pointx[j] - center[0]) \* (pointy[j + 1] - center[1]) - (pointx[j + 1] - center[0]) \* (pointy[j] - center[1]) < 0

temp[0]←pointx[j]

temp[1]←pointy[j]

pointx[j]←pointx[j+1]

pointy[j]←pointy[j+1]

pointx[j+1]←temp[0]

pointy[j+1]←temp[1]

for i←0 to index2 do

area←area+pointx[i%index]\*pointy[(i+1)%index]-pointx[(i+1)%index]\*pointy[i%index]

area←area/2

1. 演算法時間複雜度(time complexity)

掃一遍地圖把充電站找出來需耗時O(n)，找任兩點是O(m2)，判斷此兩點與其他所有點是否都在此兩點形成的直線的左側或右側會再找一遍，所以是O(m3)，排序為逆時針是O(m2)，計算面積是O(m2)，計算任兩點間最遠距離O(m2)，所以總共是O(n)+O(m3)。

1. 演算法空間複雜度(space complexity)

需兩個m大小的陣列存充電站的位置，所以空間複雜度是O(m)。

1. n台電動機車的範圍有多大(Convex-Hull，它的面積以及最遠的距離)?

Input：一個有n個電動機車的m=a\*b的地圖，x=-1代表電動機車，x>0代表充電站有x-1的空位，x=0代表是空的

Output：Convex-Hull上的點座標（單位公里）、Convex-Hull的面積（單位平方公里）、任兩點的最遠距離（單位公里）

for i←0 to a do

for j←0 to b do

if map[i][j]==-1

x[index]←j

y[index++]←i

for i←0 to index do

for j←i+1 to index do

for k←0 to indexdo

if (y[j] - y[i]) \* x[k] + (x[i] - x[j]) \* y[k] > x[i] \* y[j] - y[i] \* x[j]

ispoint\_small←false

if (y[j] - y[i]) \* x[k] + (x[i] - x[j]) \* y[k] < x[i] \* y[j] - y[i] \* x[j]

ispoint\_big←false

if ispoint\_small || ispoint\_big

for w←0 to index2 do

if pointx[w]==x[i] && pointy[w]==y[i]

exist←false

break

if exist

pointx[index2]←x[i]

pointy[index2++]←y[i]

else

exist←true

for w←0 to index2 do

if pointx[w]==x[j] && pointy[w]==y[j]

exist←false

break

if exist

pointx[index2]←x[j]

pointy[index2++]←y[j]

else

exist←true

ispoint\_small←true

ispoint\_big←true

else

ispoint\_small←true

ispoint\_big←true

for i←0 to index2 do

center[0]←center[0]+pointx[i]

center[1]←center[1]+pointy[i]

center←center/index2

for i←0 to index2 do

for j←0 to index2-i-1 do

if(pointx[j] - center[0]) \* (pointy[j + 1] - center[1]) - (pointx[j + 1] - center[0]) \* (pointy[j] - center[1]) < 0

temp[0]←pointx[j]

temp[1]←pointy[j]

pointx[j]←pointx[j+1]

pointy[j]←pointy[j+1]

pointx[j+1]←temp[0]

pointy[j+1]←temp[1]

for i←0 to index2 do

area←area+pointx[i%index]\*pointy[(i+1)%index]-pointx[(i+1)%index]\*pointy[i%index]

area←area/2

1. 演算法時間複雜度(time complexity)

掃一遍地圖把電動機車找出來需耗時O(m)，找任兩點是O(n2)，判斷此兩點與其他所有點是否都在此兩點形成的直線的左側或右側會再找一遍，所以是O(n3)，排序為逆時針是O(n2)，計算面積是O(n2)，計算任兩點間最遠距離O(n2)，所以總共是O(m)+O(n3)。

…

1. 演算法空間複雜度(space complexity)

需兩個m大小的陣列存電動機車的位置，所以空間複雜度是O(m)。

2. Hungarian(Algorithm)

1. 假設*n=m*且每一個充電站只有一個空位，要以最省電的方式*n*個人騎u-motor到充電站，如何配對，以及求出最少的總耗能?

Input：一個有m個充電站的n個電動機車的y=a\*b的地圖，x=-1代表電動機車，x>0代表充電站有x-1的空位，x=0代表是空的

Output：兩兩配對的電動機車和充電站、最少的總耗能

for i←0 to a do

for j←0 to b do

if map[i][j]==-1

pointx[index] ←j

pointy[index++]←i

else if map[i][j]>0

pointx2[index2]←j

pointy2[index2++]←i

for i←0 to index do

for j←0 to index2 do

pointcost[i][j]←sqrt(pow(pointx[i]-pointx2[j],2)+pow(pointy[i]-pointy2[j],2))

tamppointcost←pointcost

for i←0 to index do

for j←0 to index2 do

if min>tamppointcost[i][j]

min←tamppointcost[i][j]

for j←0 to index2 do

tamppointcost[i][j] ←tamppointcost[i][j]-min

min←2000

for i←0 to index2 do

for j←0 to index do

if min>tamppointcost[j][i]

min←tamppointcost[j][i]

for j←0 to index do

tamppointcost[j][i] ←tamppointcost[j][i]-min

min←2000

do

ifrepeat←true

index3←0

index4←0

index5←0

for n←0 to index do

for i←0 to index do

for j←0 to index2 do

tamppointcost2[i][j]←tamppointcost[i][j]

for i←0 to index5 do

for j←0 to index do

tamppointcost2[j][countx[i]]←tamppointcost2[j][countx[i]]+1

for j←0 to index2 do

tamppointcost2[county[i]][j]←tamppointcost2[county[i]] [j] +1

min←index2+1

for i←0 to index do

for j←0 to index2 do

if tamppointcost2[i][j]==0

c++

if c>0&&min>c

min←c

minwhere←i

c←0

if min==index2+1

ifrepeat←false

break

for i←0 to index2 do

if tamppointcost2[minwhere][i]==0

county[index5]←minwhere

countx[index5++]←i

break

if ifrepeat

break

else

for i←0 to index do

for j←0 to index5 do

if i==county[j]

ifcontain←false

if ifcontain

rline[index4++]←i

else

ifcontain←true

do

ifcontain←true

for i←0 to index4 do

for j←0 to index2 do

if tamppointcost[rline[i]][j]==0

ifrepeat←true

for k←0 to index3 do

if cline[k]==j

ifrepeat←false

break

if ifrepeat

cline[index3++]←j

ifcontain←false

for i←0 to index3 do

for j←0 to index5 do

if countx[j]==cline[i]

ifrepeat←true

for k←0 to index4 do

if rline[k]==county[j]

ifrepeat←false

break

if ifrepeat

rline[index4++]←county[j]

ifcontain←false

break

while(!ifcontain)

min←2000

for i←0 to index do

for j←0 to index2 do

ifcontain←false

for row←0 to index4 do

if rline[row]==i

ifcontain←true

break

for col←0 to index3 do

if cline[col]==j

ifcontain←false

break

if ifcontain

if min>tamppointcost[i][j]

min←tamppointcost[i][j]

else

ifcontain←true

for i←0 to index do

for j←0 to index2 do

ifcontain←false

for row←0 to index4 do

if rline[row]==i

ifcontain←true

break

if ifcontain

tamppointcost[i][j] ←tamppointcost[i][j]-min

else

ifcontain←true

for i←0 to index2 do

for j←0 to index do

if tamppointcost[j][i]<0

ifcontain←false

if !ifcontain

for j←0 to index do

tamppointcost[j][i] ←tamppointcost[j][i]+min

ifcontain←true

while(true)

for i←0 to index5 do

e←e+pointcost[county[i]][countx[i]]\*0.177

1. 演算法時間複雜度(time complexity)

先掃一遍地圖找出充電站和電動機車的位置，O(y)。

Step1要掃一遍找每列最小值，做完後再將每列減掉該列最小值，所以是O(m\*n)。

Step2要掃一遍找每行最小值，做完後再將每行減掉該行最小值，所以是O(m\*n)。

Step3最多做n次：要掃一遍找最少0的列，並將第一個0mark起來，做完後再將該行該列的0都丟棄，所以是O(m\*n2)。

Step4用最少的線將行或列劃掉。

1. 將沒有mark的0的列選起來
2. 將已選列的有0的行選起來
3. 將已選行的有mark的0的列選起來
4. 重複2、3直到沒新的選

2、3最多重複n次所以是O(m\*n2)。

Step5要掃一遍找到沒被畫的最小值，並將沒被畫的列減去此值，再將含負數的行加回此值。所以是O(m\*n)。

Step6重複3~5直到step3成功配對

所以時間複雜度是O(y)+O(m\*n2)=O(y)+O(n3)。

1. 演算法空間複雜度(space complexity)

需陣列存電動機車與充電站間距離，所以是O(n\*m)=O(n2)。

1. 如(5)所述，假設*n≠m*，第*i*個充電站空位有𝑣𝑖個, 𝑣𝑖≥0, 1≤𝑖≤𝑚，如何解決這個問題?

Input：一個有m個充電站空位的n個電動機車的y=a\*b的地圖，x=-1代表電動機車，x>0代表充電站有x-1的空位，x=0代表是空的

Output：兩兩配對的電動機車和充電站、最少的總耗能

for i←0 to a do

for j←0 to b do

if map[i][j]==-1

pointx[index] ←j

pointy[index++]←i

else if map[i][j]>0

vacancy[index6]←map[i][j]-1

pointx2[index6]←j

pointy2[index6++]←i

if(map[i][j]-1>maxv

maxv←map[i][j]-1

for i←0 to index6 do

for j←0 to vacancy[i] do

pointx2[index2]←pointx3[i]

pointy2[index2++]←pointy3[i]

for i←0 to index do

for j←0 to index2 do

pointcost[i][j]←sqrt(pow(pointx[i]-pointx2[j],2)+pow(pointy[i]-pointy2[j],2))

tamppointcost←pointcost

for i←0 to index do

for j←0 to index2 do

if min>tamppointcost[i][j]

min←tamppointcost[i][j]

for j←0 to index2 do

tamppointcost[i][j] ←tamppointcost[i][j]-min

min←2000

for i←0 to index2 do

for j←0 to index do

if min>tamppointcost[j][i]

min←tamppointcost[j][i]

for j←0 to index do

tamppointcost[j][i] ←tamppointcost[j][i]-min

min←2000

do

ifrepeat←true

index3←0

index4←0

index5←0

for n←0 to index do

for i←0 to index do

for j←0 to index2 do

tamppointcost2[i][j]←tamppointcost[i][j]

for i←0 to index5 do

for j←0 to index do

tamppointcost2[j][countx[i]]←tamppointcost2[j][countx[i]]+1

for j←0 to index2 do

tamppointcost2[county[i]][j]←tamppointcost2[county[i]] [j] +1

min←index2+1

for i←0 to index do

for j←0 to index2 do

if tamppointcost2[i][j]==0

c++

if c>0&&min>c

min←c

minwhere←i

c←0

if min==index2+1

ifrepeat←false

break

for i←0 to index2 do

if tamppointcost2[minwhere][i]==0

county[index5]←minwhere

countx[index5++]←i

break

if ifrepeat

break

else

for i←0 to index do

for j←0 to index5 do

if i==county[j]

ifcontain←false

if ifcontain

rline[index4++]←i

else

ifcontain←true

do

ifcontain←true

for i←0 to index4 do

for j←0 to index2 do

if tamppointcost[rline[i]][j]==0

ifrepeat←true

for k←0 to index3 do

if cline[k]==j

ifrepeat←false

break

if ifrepeat

cline[index3++]←j

ifcontain←false

for i←0 to index3 do

for j←0 to index5 do

if countx[j]==cline[i]

ifrepeat←true

for k←0 to index4 do

if rline[k]==county[j]

ifrepeat←false

break

if ifrepeat

rline[index4++]←county[j]

ifcontain←false

break

while(!ifcontain)

min←2000

for i←0 to index do

for j←0 to index2 do

ifcontain←false

for row←0 to index4 do

if rline[row]==i

ifcontain←true

break

for col←0 to index3 do

if cline[col]==j

ifcontain←false

break

if ifcontain

if min>tamppointcost[i][j]

min←tamppointcost[i][j]

else

ifcontain←true

for i←0 to index do

for j←0 to index2 do

ifcontain←false

for row←0 to index4 do

if rline[row]==i

ifcontain←true

break

if ifcontain

tamppointcost[i][j] ←tamppointcost[i][j]-min

else

ifcontain←true

for i←0 to index2 do

for j←0 to index do

if tamppointcost[j][i]<0

ifcontain←false

if !ifcontain

for j←0 to index do

tamppointcost[j][i] ←tamppointcost[j][i]+min

ifcontain←true

while(true)

for i←0 to index5 do

e←e+pointcost[county[i]][countx[i]]\*0.177

1. 演算法時間複雜度(time complexity)

先掃一遍地圖找出充電站空位和電動機車的位置，O(y)。

Step1要掃一遍找每列最小值，做完後再將每列減掉該列最小值，所以是O(m\*n)。

Step2要掃一遍找每行最小值，做完後再將每行減掉該行最小值，所以是O(m\*n)。

Step3最多做m次：要掃一遍找最少0的列，並將第一個0mark起來，做完後再將該行該列的0都丟棄，所以是O(m\*n2)。

Step4用最少的線將行或列劃掉。

1. 將沒有mark的0的列選起來
2. 將已選列的有0的行選起來
3. 將已選行的有mark的0的列選起來
4. 重複2、3直到沒新的選

2、3最多重複n次所以是O(m\*n2)。

Step5要掃一遍找到沒被畫的最小值，並將沒被畫的列減去此值，再將含負數的行加回此值。所以是O(m\*n)。

Step6重複3~5直到step3成功配對

所以時間複雜度是O(y)+O(m\*n2)。

1. 演算法空間複雜度(space complexity)

需陣列存電動機車與充電站間距離，所以是O(n\*m)。

**(四) 程式設計環境架構**

1. 程式語言

C in Windows 10

2. 程式開發工具

Visual Studio 2022

3. 電腦硬體

處理器 Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz 2.40 GHz

已安裝記憶體(RAM) 8.00 GB (7.81 GB 可用)

系統類型 64 位元作業系統，x64 型處理器

手寫筆與觸控 手寫筆支援

**(五) 程式 (含source code, input code, and output code)**

程式含source code, input code, and output code等…

**(1) 哪兩個充電站靠最近? 距離多少?**

1. 主程式

C program

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "stdio.h"

#include "stdlib.h"

#include "time.h"

#include "math.h"

int main(void) {

double START, END;

FILE\* inputmap;

char s[10];

printf("請輸入檔案名稱(9位元內)>>");

scanf("%s", s);

inputmap = fopen(s, "r");

double mind = 2000, d;

int x, y, x2, y2;

int\* allx=new int[1000000];

int\* ally = new int[1000000];

int index = 0;

int p;//-1:電動機車 1:沒空位的充電站 2:一個空位的充電站 3:兩個空位的充電站 以此類推

START = clock();

for (int i = 0; i < 1000; i++) {//把檔案的地圖存入map中

for (int j = 0; j < 1000; j++) {

fscanf(inputmap, "%d ", &p);

if (p > 0) {

allx[index] = j;

ally[index++] = i;

}

}

}

for (int i = 0; i < index; i++) {//計算兩充電站最小距離

for (int j = i+1; j < index; j++) {

d = sqrt(pow((allx[j] - allx[i]), 2) + pow((ally[j] - ally[i]), 2));

if (d < mind) {

y = ally[i];

x = allx[i];

y2 = ally[j];

x2 = allx[j];

mind = d;

}

}

}

printf("座標(%d,%d)和座標(%d,%d)的充電站距離最近，距離為%f公里\n", x, y, x2, y2, mind);

END = clock();

printf("Time:%.0fms", (END - START));

delete[]allx;

delete[]ally;

fclose(inputmap);//關掉地圖檔案

system("pause");

return 0;

}

2. Input Code Format

Three of examples for input use are in below….

(1)map1.txt

(2)map2.txt

(3)map3.txt

3. Output Code Format

Three of examples for output use are in below….

1. 座標(401,121)和座標(425,146)的充電站距離最近，距離為34.655447公里

Time:757ms

1. 座標(173,888)和座標(204,889)的充電站距離最近，距離為31.016125公里

Time: 985ms

1. 座標(132,713)和座標(151,714)的充電站距離最近，距離為19.026298公里

Time:1017ms

**(2) 哪兩台電動機車靠最近? 距離多少?**

1. 主程式

C program

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "stdio.h"

#include "stdlib.h"

#include "time.h"

#include "math.h"

int main(void) {

double START, END;

FILE\* inputmap;

char s[10];

printf("請輸入檔案名稱(9位元內)>>");

scanf("%s", s);

inputmap = fopen(s, "r");

double mind = 2000, d;

int x, y, x2, y2;

int\* allx = new int[1000000];

int\* ally = new int[1000000];

int index = 0;

int p;//-1:電動機車 1:沒空位的充電站 2:一個空位的充電站 3:兩個空位的充電站 以此類推

START = clock();

for (int i = 0; i < 1000; i++) {//把檔案的地圖存入map中

for (int j = 0; j < 1000; j++) {

fscanf(inputmap, "%d ", &p);

if (p == -1) {

allx[index] = j;

ally[index++] = i;

}

}

}

for (int i = 0; i < index; i++) {//計算兩電動機車最小距離

for (int j = i + 1; j < index; j++) {

d = sqrt(pow((allx[j] - allx[i]), 2) + pow((ally[j] - ally[i]), 2));

if (d < mind) {

y = ally[i];

x = allx[i];

y2 = ally[j];

x2 = allx[j];

mind = d;

}

}

}

printf("座標(%d,%d)和座標(%d,%d)的電動機車距離最近，距離為%f公里\n", x, y, x2, y2, mind);

END = clock();

printf("Time:%.0fms", (END - START));

delete[]allx;

delete[]ally;

fclose(inputmap);//關掉地圖檔案

system("pause");

return 0;

}

2. Input Code Format

Three of examples for input use are in below….

(1)map1.txt

(2)map2.txt

(3)map3.txt

3. Output Code Format

Three of examples for output use are in below….

1. 座標(963,893)和座標(985,894)的電動機車距離最近，距離為22.022716公里

Time:834ms

1. 座標(226,13)和座標(186,28)的電動機車距離最近，距離為42.720019公里

Time:1015ms

1. 座標(755,917)和座標(801,925)的電動機車距離最近，距離為46.690470公里

Time:970ms

**(3) m個充電站的範圍有多大(Convex-Hull，它的面積以及最遠的距離)?**

1. 主程式

C program

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "stdio.h"

#include "stdlib.h"

#include "time.h"

#include "math.h"

int main(void) {

double START, END;

FILE\* inputmap;

char s[10];

printf("請輸入檔案名稱(9位元內)>>");

scanf("%s", s);

inputmap = fopen(s, "r");

int p;//-1:電動機車 1:沒空位的充電站 2:一個空位的充電站 3:兩個空位的充電站 以此類推

int\* x = new int[1000000];

int\* y = new int[1000000];

int index = 0;

START = clock();

for (int i = 0; i < 1000; i++) {//把檔案的地圖挑取充電站存入

for (int j = 0; j < 1000; j++) {

fscanf(inputmap, "%d ", &p);

if (p > 0) {

x[index] = j;

y[index++] = i;

}

}

}

int\* pointx = new int[index];

int\* pointy = new int[index];

int index2 = 0;

bool ispoint\_big = true;

bool ispoint\_small = true;

bool exist = true;

for (int i = 0; i < index; i++) {

for (int j = i+1; j < index; j++) {

for (int k = 0; k < index && (ispoint\_small || ispoint\_big); k++) {

if ((y[j] - y[i]) \* x[k] + (x[i] - x[j]) \* y[k] > x[i] \* y[j] - y[i] \* x[j]) {

ispoint\_small = false;

}

if ((y[j] - y[i]) \* x[k] + (x[i] - x[j]) \* y[k] < x[i] \* y[j] - y[i] \* x[j]) {

ispoint\_big = false;

}

}

if (ispoint\_small || ispoint\_big) {

for (int a = 0; a < index2; a++) {

if (pointx[a] == x[i]) {

if (pointy[a] == y[i]) {

exist = false;

break;

}

}

}

if (exist) {

pointx[index2] = x[i];

pointy[index2++] = y[i];

}

else {

exist = true;

}

for (int a = 0; a < index2; a++) {

if (pointx[a] == x[j]) {

if (pointy[a] == y[j]) {

exist = false;

break;

}

}

}

if (exist) {

pointx[index2] = x[j];

pointy[index2++] = y[j];

}

else {

exist = true;

}

ispoint\_big = true;

ispoint\_small = true;

}

else {

ispoint\_big = true;

ispoint\_small = true;

}

}

}

printf("充電站的集合的凸包的極值點為：\n");

for (int i = 0; i < index2; i++) {

printf("(%d,%d)\n", pointx[i], pointy[i]);

}

double center[2] = { 0,0 };

int temp[2];

for (int i = 0; i < index2; i++) {

center[0] += pointx[i];

center[1] += pointy[i];

}

center[0] /= index2;

center[1] /= index2;

for (int i = 0; i < index2 - 1; i++) {

for (int j = 0; j < index2 - i - 1; j++) {

if ((pointx[j] - center[0]) \* (pointy[j + 1] - center[1]) - (pointx[j + 1] - center[0]) \* (pointy[j] - center[1]) < 0) {

temp[0] = pointx[j];

temp[1] = pointy[j];

pointx[j] = pointx[j + 1];

pointy[j] = pointy[j + 1];

pointx[j + 1] = temp[0];

pointy[j + 1] = temp[1];

}

}

}

double area = 0;

for (int i = 0; i < index2; i++) {

area += pointx[i % index2] \* pointy[(i + 1) % index2] - pointx[(i + 1) % index2] \* pointy[i % index2];

}

area /= 2;

printf("凸包的面積：");

printf("%f平方公里\n", area);

double max = 0;

for (int i = 0; i < index2; i++) {

for (int j = i + 1; j < index2; j++) {

if (pow(pointx[j] - pointx[i], 2) + pow(pointy[j] - pointy[i], 2) > max) {

max = pow(pointx[j] - pointx[i], 2) + pow(pointy[j] - pointy[i], 2);

}

}

}

max = sqrt(max);

printf("任兩點最遠的距離為%f公里\n", max);

END = clock();

printf("Time:%.0fms", (END - START));

delete[]x;

delete[]y;

delete[]pointx;

delete[]pointy;

fclose(inputmap);//關掉地圖檔案

system("pause");

return 0;

}

2. Input Code Format

Three of examples for input use are in below….

(1)map1.txt

(2)map2.txt

(3)map3.txt

3. Output Code Format

Three of examples for output use are in below….

因為太長，占版面，所以丟到檔案裡。

(1) o1.txt

(2) o2.txt

(3) o3.txt

**(4) n台電動機車的範圍有多大(Convex-Hull，它的面積以及最遠的距離)?**

1. 主程式

C program

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "stdio.h"

#include "stdlib.h"

#include "time.h"

#include "math.h"

int main(void) {

double START, END;

FILE\* inputmap;

char s[10];

printf("請輸入檔案名稱(9位元內)>>");

scanf("%s", s);

inputmap = fopen(s, "r");

int p;//-1:電動機車 1:沒空位的充電站 2:一個空位的充電站 3:兩個空位的充電站 以此類推

int\* x = new int[1000000];

int\* y = new int[1000000];

int index = 0;

START = clock();

for (int i = 0; i < 1000; i++) {//把檔案的地圖挑取充電站存入

for (int j = 0; j < 1000; j++) {

fscanf(inputmap, "%d ", &p);

if (p == -1) {

x[index] = j;

y[index++] = i;

}

}

}

int\* pointx = new int[index];

int\* pointy = new int[index];

int index2 = 0;

bool ispoint\_big = true;

bool ispoint\_small = true;

bool exist = true;

for (int i = 0; i < index; i++) {

for (int j = i + 1; j < index; j++) {

for (int k = 0; k < index && (ispoint\_small || ispoint\_big); k++) {

if ((y[j] - y[i]) \* x[k] + (x[i] - x[j]) \* y[k] > x[i] \* y[j] - y[i] \* x[j]) {

ispoint\_small = false;

}

if ((y[j] - y[i]) \* x[k] + (x[i] - x[j]) \* y[k] < x[i] \* y[j] - y[i] \* x[j]) {

ispoint\_big = false;

}

}

if (ispoint\_small || ispoint\_big) {

for (int a = 0; a < index2; a++) {

if (pointx[a] == x[i]) {

if (pointy[a] == y[i]) {

exist = false;

break;

}

}

}

if (exist) {

pointx[index2] = x[i];

pointy[index2++] = y[i];

}

else {

exist = true;

}

for (int a = 0; a < index2; a++) {

if (pointx[a] == x[j]) {

if (pointy[a] == y[j]) {

exist = false;

break;

}

}

}

if (exist) {

pointx[index2] = x[j];

pointy[index2++] = y[j];

}

else {

exist = true;

}

ispoint\_big = true;

ispoint\_small = true;

}

else {

ispoint\_big = true;

ispoint\_small = true;

}

}

}

printf("電動機車的集合的凸包的極值點為：\n");

for (int i = 0; i < index2; i++) {

printf("(%d,%d)\n", pointx[i], pointy[i]);

}

double center[2] = { 0,0 };

int temp[2];

for (int i = 0; i < index2; i++) {

center[0] += pointx[i];

center[1] += pointy[i];

}

center[0] /= index2;

center[1] /= index2;

for (int i = 0; i < index2 - 1; i++) {

for (int j = 0; j < index2 - i - 1; j++) {

if ((pointx[j] - center[0]) \* (pointy[j + 1] - center[1]) - (pointx[j + 1] - center[0]) \* (pointy[j] - center[1]) < 0) {

temp[0] = pointx[j];

temp[1] = pointy[j];

pointx[j] = pointx[j + 1];

pointy[j] = pointy[j + 1];

pointx[j + 1] = temp[0];

pointy[j + 1] = temp[1];

}

}

}

double area = 0;

for (int i = 0; i < index2; i++) {

area += pointx[i % index2] \* pointy[(i + 1) % index2] - pointx[(i + 1) % index2] \* pointy[i % index2];

}

area /= 2;

printf("凸包的面積：");

printf("%f平方公里\n", area);

double max = 0;

for (int i = 0; i < index2; i++) {

for (int j = i + 1; j < index2; j++) {

if (pow(pointx[j] - pointx[i], 2) + pow(pointy[j] - pointy[i], 2) > max) {

max = pow(pointx[j] - pointx[i], 2) + pow(pointy[j] - pointy[i], 2);

}

}

}

max = sqrt(max);

printf("任兩點最遠的距離為%f公里\n", max);

END = clock();

printf("Time:%.0fms", (END - START));

delete[]x;

delete[]y;

delete[]pointx;

delete[]pointy;

fclose(inputmap);//關掉地圖檔案

system("pause");

return 0;

}

2. Input Code Format

Three of examples for input use are in below….

(1)map1.txt

(2)map2.txt

(3)map3.txt

3. Output Code Format

Three of examples for output use are in below….

因為太長，占版面，所以丟到檔案裡。

(1) o1.txt

(2) o2.txt

(3) o3.txt

**(5) 假設*n=m*且每一個充電站只有一個空位，要以最省電的方式*n*個人騎u-motor到充電站，如何配對，以及求出最少的總耗能?**

1. 主程式

C program

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "stdio.h"

#include "stdlib.h"

#include "time.h"

#include "math.h"

int main(void) {

double START, END;

FILE\* inputmap;

char s[10];

printf("請輸入檔案名稱(9位元內)>>");

scanf("%s", s);

inputmap = fopen(s, "r");

int p;//-1:電動機車 1:沒空位的充電站 2:一個空位的充電站 3:兩個空位的充電站 以此類推

int\* pointx = new int[1000000];//電動車

int\* pointy = new int[1000000];

int\* pointx2 = new int[1000000];//充電站

int\* pointy2 = new int[1000000];

int index = 0;//電動車

int index2 = 0;//充電站

START = clock();

for (int i = 0; i < 1000; i++) {//把檔案的地圖存入map中

for (int j = 0; j < 1000; j++) {

fscanf(inputmap, "%d ", &p);

if (p == -1) {

pointx[index] = j;

pointy[index++] = i;

}

else if (p > 0) {

pointx2[index2] = j;

pointy2[index2++] = i;

}

}

}

double\*\* pointcost = new double\*[index];

double\*\* tamppointcost = new double\* [index];

double\*\* tamppointcost2 = new double\* [index];

for (int i = 0; i < index; i++) {

pointcost[i] = new double[index2];

tamppointcost[i] = new double[index2];

tamppointcost2[i] = new double[index2];

}

for (int i = 0; i < index; i++) {//計算每個電動車和每個充電站的距離

for (int j = 0; j < index2; j++) {

pointcost[i][j] = sqrt(pow(pointx[i] - pointx2[j], 2) + pow(pointy[i] - pointy2[j], 2));

}

}

for (int i = 0; i < index; i++) {

for (int j = 0; j < index2; j++) {

tamppointcost[i][j] = pointcost[i][j];

}

}

double min = 2000;

for (int i = 0; i < index; i++) {//step1：將每列減掉該列最小的值

for (int j = 0; j < index2; j++) {

if (min > tamppointcost[i][j]) {

min = tamppointcost[i][j];

}

}

for (int j = 0; j < index2; j++) {

tamppointcost[i][j] -= min;

}

min = 2000;

}

for (int i = 0; i < index2; i++) {//step2：將每行減掉該行最小的值

for (int j = 0; j < index; j++) {

if (min > tamppointcost[j][i]) {

min = tamppointcost[j][i];

}

}

for (int j = 0; j < index; j++) {

tamppointcost[j][i] -= min;

}

min = 2000;

}

int max = 0;

int max2 = 0;

int compare = 0;

int compare2 = 0;

int c = 0;

int maxwhere[2] = { 0,0 };

int\* cline = new int[index];//step4畫的縱線

int\* rline = new int[index];//step4沒畫的橫線

int index3 = 0, index4 = 0;//cline和rline的索引值

int\* countx = new int[index];//mark的0的x軸值

int\* county = new int[index];//mark的0的y軸值

int index5 = 0;//count的索引值

int minwhere = 0;

bool ifrepeat = true;//判斷是否結束迴圈

bool ifcontain = true;//判斷某點是否被畫掉

do {

ifrepeat = true;

index3 = 0;

index4 = 0;

index5 = 0;

for (int n = 0; n < index; n++) {//step3：判斷是否已完成配對

for (int i = 0; i < index; i++) {

for (int j = 0; j < index2; j++) {

tamppointcost2[i][j] = tamppointcost[i][j];

}

}

for (int i = 0; i < index5; i++) {

for (int j = 0; j < index; j++) {

tamppointcost2[j][countx[i]] += 1;

}

for (int j = 0; j < index2; j++) {

tamppointcost2[county[i]][j] += 1;

}

}

min = index2 + 1;

for (int i = 0; i < index; i++) {

for (int j = 0; j < index2; j++) {

if (tamppointcost2[i][j] == 0) {

c++;

}

}

if (c > 0 && min > c) {

min = c;

minwhere = i;

}

c = 0;

}

if (min == index2 + 1) {

ifrepeat = false;

break;

}

for (int i = 0; i < index2; i++) {

if (tamppointcost2[minwhere][i] == 0) {

county[index5] = minwhere;

countx[index5++] = i;

break;

}

}

}

if (ifrepeat) {//如果完成配對就跳出

break;

}

else {//step4：如果沒完成配對，用最小線的數量畫掉所有的0

for (int i = 0; i < index; i++) {

for (int j = 0; j < index5; j++) {

if (i == county[j]) {

ifcontain = false;

}

}

if (ifcontain) {

rline[index4++] = i;

}

else {

ifcontain = true;

}

}

do {

ifcontain = true;

for (int i = 0; i < index4; i++) {

for (int j = 0; j < index2; j++) {

if (tamppointcost[rline[i]][j] == 0) {

ifrepeat = true;

for (int k = 0; k < index3; k++) {

if (cline[k] == j) {

ifrepeat = false;

break;

}

}

if (ifrepeat) {

cline[index3++] = j;

ifcontain = false;

}

}

}

}

for (int i = 0; i < index3; i++) {

for (int j = 0; j < index5; j++) {

if (countx[j] == cline[i]) {

ifrepeat = true;

for (int k = 0; k < index4; k++) {

if (rline[k] == county[j]) {

ifrepeat = false;

break;

}

}

if (ifrepeat) {

rline[index4++] = county[j];

ifcontain = false;

}

break;

}

}

}

} while (!ifcontain);

//step5

min = 2000;

for (int i = 0; i < index; i++) {//找到沒被劃掉的值中最小的值

for (int j = 0; j < index2; j++) {

ifcontain = false;

for (int row = 0; row < index4; row++) {

if (rline[row] == i) {

ifcontain = true;

break;

}

}

for (int col = 0; col < index3; col++) {

if (cline[col] == j) {

ifcontain = false;

break;

}

}

if (ifcontain) {

if (min > tamppointcost[i][j]) {

min = tamppointcost[i][j];

}

}

else {

ifcontain = true;

}

}

}

for (int i = 0; i < index; i++) {//將沒被劃掉的值的那列減掉沒被劃掉的值中最小的值

for (int j = 0; j < index2; j++) {

ifcontain = false;

for (int row = 0; row < index4; row++) {

if (rline[row] == i) {

ifcontain = true;

break;

}

}

if (ifcontain) {

tamppointcost[i][j] -= min;

}

else {

ifcontain = true;

}

}

}

for (int i = 0; i < index2; i++) {//將全部負的行加回0

for (int j = 0; j < index; j++) {

if (tamppointcost[j][i] < 0) {

ifcontain = false;

}

}

if (!ifcontain) {

for (int j = 0; j < index; j++) {

tamppointcost[j][i] += min;

}

ifcontain = true;

}

}

}

} while (true);

printf("\t\t\t\t");

for (int i = 1; i <= index2; i++) {

printf(" 充電站%4d號(%4d,%4d) ", i, pointx2[i - 1], pointy2[i - 1]);

}

printf("\n");

for (int i = 0; i < index; i++) {

printf("\n電動車%4d號(%4d,%4d) ", i + 1, pointx[i], pointy[i]);

for (int j = 0; j < index2; j++) {

printf("%25.3f", pointcost[i][j]);

}

}

printf("\n電動車配對的充電站：\n");

for (int i = 0; i < index5; i++) {

printf("電動車%4d號配對充電站%4d號距離%.3f公里\n", county[i]+1, countx[i]+1, pointcost[county[i]][countx[i]]);

}

double e = 0;

for (int i = 0; i < index5; i++) {

e += pointcost[county[i]][countx[i]]\*0.177;

}

printf("每公里耗電0.1777度，總耗電量為%f度\n", e);

END = clock();

printf("Time:%.0fms", (END - START));

delete[]countx;

delete[]county;

delete[]cline;

delete[]rline;

delete[]pointx;

delete[]pointy;

delete[]pointx2;

delete[]pointy2;

for (int i = 0; i < index; i++) {//將new的delete掉

delete[]pointcost[i];

}

delete[]pointcost;

for (int i = 0; i <index; i++) {//將new的地圖delete掉

delete[]tamppointcost[i];

}

delete[]tamppointcost;

for (int i = 0; i < index; i++) {//將new的地圖delete掉

delete[]tamppointcost2[i];

}

delete[]tamppointcost2;

fclose(inputmap);

system("pause");

return 0;

}

2. Input Code Format

Three of examples for input use are in below….

(1)map1.txt

(2)map2.txt

(3)map3.txt

3. Output Code Format

Three of examples for output use are in below….

因為太長，占版面，所以丟到檔案裡。

(1)o1.txt

(2)o2.txt

(3)o3.txt

**(6) 如(5)所述，假設*n≠m*，第*i*個充電站空位有𝑣𝑖個, 𝑣𝑖≥0, 1≤𝑖≤𝑚，如何解決這個問題?**

1. 主程式

C program

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "stdio.h"

#include "stdlib.h"

#include "time.h"

#include "math.h"

int main(void) {

double START, END;

FILE\* inputmap;

char s[10];

printf("請輸入檔案名稱(9位元內)>>");

scanf("%s", s);

inputmap = fopen(s, "r");

int p;//-1:電動機車 1:沒空位的充電站 2:一個空位的充電站 3:兩個空位的充電站 以此類推

int maxv = 0;

int\* pointx = new int[1000000];//電動車

int\* pointy = new int[1000000];

int\* pointx3 = new int[1000000];//充電站

int\* pointy3 = new int[1000000];

int\* vacancy = new int[1000000];

int index = 0;//電動車

int index2 = 0;//充電站空位

int index6 = 0;//充電站

START = clock();

for (int i = 0; i < 1000; i++) {//把檔案的地圖存入map中

for (int j = 0; j < 1000; j++) {

fscanf(inputmap, "%d ", &p);

if (p == -1) {

pointx[index] = j;

pointy[index++] = i;

}

else if (p > 0) {

vacancy[index6] = p - 1;

pointx3[index6] = j;

pointy3[index6++] = i;

if (p - 1 > maxv) {

maxv = p - 1;

}

}

}

}

int\* pointx2 = new int[maxv \* 1000000];//充電站空位

int\* pointy2 = new int[maxv \* 1000000];

for (int i = 0; i < index6; i++) {

for (int j = 0; j < vacancy[i]; j++) {

pointx2[index2] = pointx3[i];

pointy2[index2++] = pointy3[i];

}

}

double\*\* pointcost = new double\* [index];

for (int i = 0; i < index; i++) {

pointcost[i] = new double[index2];

}

for (int i = 0; i < index; i++) {//計算每個電動車和每個充電站的距離

for (int j = 0; j < index2; j++) {

pointcost[i][j] = sqrt(pow(pointx[i] - pointx2[j], 2) + pow(pointy[i] - pointy2[j], 2));

}

}

double\*\* tamppointcost = new double\* [index];

for (int i = 0; i < index; i++) {

tamppointcost[i] = new double[index2];

}

double\*\* tamppointcost2 = new double\* [index];

for (int i = 0; i < index; i++) {

tamppointcost2[i] = new double[index2];

}

for (int i = 0; i < index; i++) {

for (int j = 0; j < index2; j++) {

tamppointcost[i][j] = pointcost[i][j];

}

}

double min = 2000;

for (int i = 0; i < index; i++) {//step1：將每列減掉該列最小的值

for (int j = 0; j < index2; j++) {

if (min > tamppointcost[i][j]) {

min = tamppointcost[i][j];

}

}

for (int j = 0; j < index2; j++) {

tamppointcost[i][j] -= min;

}

min = 2000;

}

for (int i = 0; i < index2; i++) {//step2：將每行減掉該行最小的值

for (int j = 0; j < index; j++) {

if (min > tamppointcost[j][i]) {

min = tamppointcost[j][i];

}

}

for (int j = 0; j < index; j++) {

tamppointcost[j][i] -= min;

}

min = 2000;

}

int c = 0;

int maxwhere[2] = { 0,0 };

int\* cline = new int[index];//step4畫的縱線

int\* rline = new int[index];//step4沒畫的橫線

int index3 = 0, index4 = 0;//cline和rline的索引值

int\* countx = new int[index];//mark的0的x軸值

int\* county = new int[index];//mark的0的y軸值

int index5 = 0;//count的索引值

int minwhere = 0;

bool ifrepeat = true;

bool ifcontain = true;

do {

ifrepeat = true;

index3 = 0;

index4 = 0;

index5 = 0;

for (int n = 0; n < index; n++) {//step3：判斷是否已完成配對

for (int i = 0; i < index; i++) {

for (int j = 0; j < index2; j++) {

tamppointcost2[i][j] = tamppointcost[i][j];

}

}

for (int i = 0; i < index5; i++) {

for (int j = 0; j < index; j++) {

tamppointcost2[j][countx[i]] += 1;

}

for (int j = 0; j < index2; j++) {

tamppointcost2[county[i]][j] += 1;

}

}

min = index2 + 1;

for (int i = 0; i < index; i++) {

for (int j = 0; j < index2; j++) {

if (tamppointcost2[i][j] == 0) {

c++;

}

}

if (c > 0 && min > c) {

min = c;

minwhere = i;

}

c = 0;

}

if (min == index2 + 1) {

ifrepeat = false;

break;

}

for (int i = 0; i < index2; i++) {

if (tamppointcost2[minwhere][i] == 0) {

county[index5] = minwhere;

countx[index5++] = i;

break;

}

}

}

if (ifrepeat) {//如果完成配對就跳出

break;

}

else {//step4：如果沒完成配對，用最小線的數量畫掉所有的0

for (int i = 0; i < index; i++) {

for (int j = 0; j < index5; j++) {

if (i == county[j]) {

ifcontain = false;

}

}

if (ifcontain) {

rline[index4++] = i;

}

else {

ifcontain = true;

}

}

do {

ifcontain = true;

for (int i = 0; i < index4; i++) {

for (int j = 0; j < index2; j++) {

if (tamppointcost[rline[i]][j] == 0) {

ifrepeat = true;

for (int k = 0; k < index3; k++) {

if (cline[k] == j) {

ifrepeat = false;

break;

}

}

if (ifrepeat) {

cline[index3++] = j;

ifcontain = false;

}

}

}

}

for (int i = 0; i < index3; i++) {

for (int j = 0; j < index5; j++) {

if (countx[j] == cline[i]) {

ifrepeat = true;

for (int k = 0; k < index4; k++) {

if (rline[k] == county[j]) {

ifrepeat = false;

break;

}

}

if (ifrepeat) {

rline[index4++] = county[j];

ifcontain = false;

}

break;

}

}

}

} while (!ifcontain);

//step5

min = 2000;

for (int i = 0; i < index; i++) {//找到沒被劃掉的值中最小的值

for (int j = 0; j < index2; j++) {

ifcontain = false;

for (int row = 0; row < index4; row++) {

if (rline[row] == i) {

ifcontain = true;

break;

}

}

for (int col = 0; col < index3; col++) {

if (cline[col] == j) {

ifcontain = false;

break;

}

}

if (ifcontain) {

if (min > tamppointcost[i][j]) {

min = tamppointcost[i][j];

}

}

else {

ifcontain = true;

}

}

}

for (int i = 0; i < index; i++) {//將沒被劃掉的值的那列減掉沒被劃掉的值中最小的值

for (int j = 0; j < index2; j++) {

ifcontain = false;

for (int row = 0; row < index4; row++) {

if (rline[row] == i) {

ifcontain = true;

break;

}

}

if (ifcontain) {

tamppointcost[i][j] -= min;

}

else {

ifcontain = true;

}

}

}

for (int i = 0; i < index2; i++) {//將全部負的行加回0

for (int j = 0; j < index; j++) {

if (tamppointcost[j][i] < 0) {

ifcontain = false;

}

}

if (!ifcontain) {

for (int j = 0; j < index; j++) {

tamppointcost[j][i] += min;

}

ifcontain = true;

}

}

}

} while (true);

printf("\t\t\t\t");

for (int i = 1; i <= index6; i++) {

printf(" 充電站%4d號(%4d,%4d)%d個空位 ", i, pointx3[i - 1], pointy3[i - 1], vacancy[i - 1]);

}

printf("\n");

for (int i = 0; i < index; i++) {

printf("\n電動車%4d號(%4d,%4d) ", i + 1, pointx[i], pointy[i]);

for (int j = 0, k = 0; j < index2; j += vacancy[k++]) {

printf("%32.3f", pointcost[i][j]);

}

}

printf("\n電動車配對的充電站：\n");

for (int i = 0, c = countx[i], c2 = 0; i < index5; i++, c = countx[i], c2 = 0) {

for (int j = 0; j < index6; j++) {

c -= vacancy[c2++];

if (c < 0) {

break;

}

}

printf("電動車%4d號配對充電站%4d號距離%.3f公里\n", county[i] + 1, c2, pointcost[county[i]][countx[i]]);

}

double e = 0;

for (int i = 0; i < index5; i++) {

e += pointcost[county[i]][countx[i]] \* 0.177;

}

printf("每公里耗電0.1777度，總耗電量為%f度\n", e);

END = clock();

printf("Time:%.0fms", (END - START));

delete[]countx;

delete[]county;

delete[]cline;

delete[]rline;

delete[]pointx;

delete[]pointy;

delete[]pointx2;

delete[]pointy2;

delete[]pointx3;

delete[]pointy3;

delete[]vacancy;

for (int i = 0; i < index; i++) {//將new的delete掉

delete[]pointcost[i];

}

delete[]pointcost;

for (int i = 0; i < index; i++) {//將new的地圖delete掉

delete[]tamppointcost[i];

}

delete[]tamppointcost;

for (int i = 0; i < index; i++) {//將new的地圖delete掉

delete[]tamppointcost2[i];

}

delete[]tamppointcost2;

fclose(inputmap);

system("pause");

return 0;

}

2. Input Code Format

Three of examples for input use are in below….

(1)map1.txt

(2)map2.txt

(3)map3.txt

3. Output Code Format

Three of examples for output use are in below….

因為太長，占版面，所以丟到檔案裡。

(1)o1.txt

(2)o2.txt

(3)o3.txt**(五) 執行結果、討論與心得**

執行結果與討論 (執行時間、problem *n*的大小等問題討論)等…

**(1) 哪兩個充電站靠最近? 距離多少?**

1. 執行結果

Output of program:

* + 1. 座標(668,24)和座標(655,42)的充電站距離最近，距離為22.203603公里

Time:735ms

* + 1. 座標(849,925)和座標(844,928)的充電站距離最近，距離為5.830952公里

Time:1075ms

* + 1. 座標(458,859)和座標(456,864)的充電站距離最近，距離為5.385165公里

Time:976ms

* + 1. 座標(802,835)和座標(800,839)的充電站距離最近，距離為4.472136公里

Time:1268ms

* + 1. 座標(125,759)和座標(124,762)的充電站距離最近，距離為3.162278公里

Time:1023ms

* + 1. 座標(124,755)和座標(123,757)的充電站距離最近，距離為2.236068公里

Time:1309ms

2. 討論

執行時間、問題大小等問題討論

1. Running Time

i.735ms

ii. 1075ms

iii. 976ms

iv. 1268ms

v. 1023ms

vi.1309ms

縱軸：時間(ms)

橫軸：充電站數量

可能因為地圖大小遠超充電站數量的平方，所以上升幅度不明顯。

(2) Problem size *n*

i. 包含50個充電站、50個電動機車的1000\*1000的地圖

ii. 包含100個充電站、100個電動機車的1000\*1000的地圖

iii. 包含150個充電站、150個電動機車的1000\*1000的地圖

iv. 包含200個充電站、200個電動機車的1000\*1000的地圖

v. 包含250個充電站、250個電動機車的1000\*1000的地圖

vi. 包含300個充電站、300個電動機車的1000\*1000的地圖

1. **哪兩台電動機車靠最近? 距離多少?**

1. 執行結果

Output of program:

* + 1. 座標(282,21)和座標(285,25)的電動機車距離最近，距離為5.000000公里

Time:738ms

* + 1. 座標(579,988)和座標(583,988)的電動機車距離最近，距離為4.000000公里

Time:1012ms

* + 1. 座標(7,889)和座標(5,891)的電動機車距離最近，距離為2.828427公里

Time:1017ms

* + 1. 座標(892,369)和座標(896,370)的電動機車距離最近，距離為4.123106公里

Time:927ms

* + 1. 座標(256,641)和座標(257,643)的電動機車距離最近，距離為2.236068公里

Time:1099ms

* + 1. 座標(948,342)和座標(950,342)的電動機車距離最近，距離為2.000000公里

Time:944ms

2. 討論

執行時間、問題大小等問題討論

1. Running Time

i. 738ms

ii. 1012ms

iii. 1017ms

iv. 927ms

v. 1099ms

vi. 944ms

縱軸：時間(ms)

橫軸：電動機車數量

可能因為地圖大小遠超電動機車數量的平方，所以上升幅度不明顯。

1. Problem size *n*

i. 包含50個充電站、50個電動機車的1000\*1000的地圖

ii. 包含100個充電站、100個電動機車的1000\*1000的地圖

iii. 包含150個充電站、150個電動機車的1000\*1000的地圖

iv. 包含200個充電站、200個電動機車的1000\*1000的地圖

v. 包含250個充電站、250個電動機車的1000\*1000的地圖

vi. 包含300個充電站、300個電動機車的1000\*1000的地圖

1. **m個充電站的範圍有多大(Convex-Hull，它的面積以及最遠的距離)?**

1. 執行結果

Output of program:

* + 1. o4.txt
    2. o5.txt
    3. o6.txt
    4. o7.txt
    5. o8.txt
    6. o9.txt

2. 討論

執行時間、問題大小等問題討論

1. Running Time

i. 1129ms

ii. 1305ms

iii. 1077ms

iv. 1253ms

v. 1007ms

vi. 1249ms

縱軸：時間(ms)

橫軸：充電站數量

可能因為充電站數量不夠大，所以上升幅度不明顯。

1. Problem size *n*

i. 包含50個充電站、50個電動機車的1000\*1000的地圖

ii. 包含100個充電站、100個電動機車的1000\*1000的地圖

iii. 包含150個充電站、150個電動機車的1000\*1000的地圖

iv. 包含200個充電站、200個電動機車的1000\*1000的地圖

v. 包含250個充電站、250個電動機車的1000\*1000的地圖

vi. 包含300個充電站、300個電動機車的1000\*1000的地圖

1. **n台電動機車的範圍有多大(Convex-Hull，它的面積以及最遠的距離)?**

1. 執行結果

Output of program:

* + 1. o4.txt
    2. o5.txt
    3. o6.txt
    4. o7.txt
    5. o8.txt
    6. o9.txt

2. 討論

執行時間、問題大小等問題討論

1. Running Time

i. 1265ms

ii. 1313ms

iii. 1189ms

iv. 1085ms

v. 1072ms

vi. 1386ms

縱軸：時間(ms)

橫軸：電動機車數量

可能因為電動機車數量不夠大，所以上升幅度不明顯。

1. Problem size *n*

i. 包含50個充電站、50個電動機車的1000\*1000的地圖

ii. 包含100個充電站、100個電動機車的1000\*1000的地圖

iii. 包含150個充電站、150個電動機車的1000\*1000的地圖

iv. 包含200個充電站、200個電動機車的1000\*1000的地圖

v. 包含250個充電站、250個電動機車的1000\*1000的地圖

vi. 包含300個充電站、300個電動機車的1000\*1000的地圖

1. **假設*n=m*且每一個充電站只有一個空位，要以最省電的方式*n*個人騎u-motor到充電站，如何配對，以及求出最少的總耗能?**

1. 執行結果

Output of program:

* + 1. o4.txt
    2. o5.txt
    3. o6.txt
    4. o7.txt
    5. o8.txt
    6. o9.txt

2. 討論

執行時間、問題大小等問題討論

1. Running Time

i. 1358ms

ii. 6060ms

iii. 15358ms

iv. 49474ms

v. 74907ms

vi. 137781ms

縱軸：時間(ms)

橫軸：充電站、電動機車數量

因充電站、電動機車數量的三次方大於地圖大小，所以呈指數圖形，符合結果。

1. Problem size *n*

i. 包含50個充電站、50個電動機車的1000\*1000的地圖

ii. 包含100個充電站、100個電動機車的1000\*1000的地圖

iii. 包含150個充電站、150個電動機車的1000\*1000的地圖

iv. 包含200個充電站、200個電動機車的1000\*1000的地圖

v. 包含250個充電站、250個電動機車的1000\*1000的地圖

vi. 包含300個充電站、300個電動機車的1000\*1000的地圖

1. **如(5)所述，假設*n≠m*，第*i*個充電站空位有𝑣𝑖個, 𝑣𝑖≥0, 1≤𝑖≤𝑚，如何解決這個問題?**

1. 執行結果

Output of program:

* + 1. o4.txt
    2. o5.txt
    3. o6.txt
    4. o7.txt
    5. o8.txt
    6. o9.txt

2. 討論

執行時間、問題大小等問題討論

1. Running Time

i. 2108ms

ii. 3340ms

iii. 5327ms

iv. 6142ms

v. 12558ms

vi.24024ms

縱軸：時間(ms)

橫軸：電動機車數量

因電動機車數量的三次方大於地圖大小，所以呈指數圖形，符合結果。

縱軸：時間(ms)

橫軸：充電站數量

因充電站空位數量(約為2.5\*充電站數量)的三次方大於地圖大小，所以呈指數圖形，符合結果。

1. Problem size *n*

i. 包含55個充電站（空位０～５個亂數產生）、50個電動機車的1000\*1000的地圖

ii. 包含105個充電站（空位０～５個亂數產生）、100個電動機車的1000\*1000的地圖

iii. 包含155個充電站（空位０～５個亂數產生）、150個電動機車的1000\*1000的地圖

iv. 包含205個充電站（空位０～５個亂數產生）、200個電動機車的1000\*1000的地圖

v. 包含255個充電站（空位０～５個亂數產生）、250個電動機車的1000\*1000的地圖

vi. 包含305個充電站（空位０～５個亂數產生）、300個電動機車的1000\*1000的地圖

3. 心得

1~2題我寫了兩遍，第一遍用地圖直接倆倆比較，第二遍我就先將充電站和電動機車的位置提出來，再倆倆比較，當地圖很大但充電站和電動機車數量少時，時間能快上不少，3~4題計算面積方面我花了很多功夫，本來想說只要計算角度，就能知道是逆時針還是順時針，但仔細想想卻發現不對，後來找到了正確的方法(參考文獻中)，更熟悉凸多邊形的面積計算。第5題在step4卡了很久，實在想不出方法，最後只好求助網路，找到了看起來可行的方法(參考文獻中)，也因為一開始step4的方法不對，debug很久，所以已經非常熟悉匈牙利演算法的Step了，第6題本來是判斷是否還有空間，來允許電動機車重複配對充電站，後來發現這種方法可能會不匹配原來匈牙利演算法step4的畫線方式，所以改為電動機車直接與充電站空位匹配，這樣直觀上就與第5題相同了。3、4和5、6雖然都是三次方，但是時間複雜度仍然有差距，因為5、6的3次方步驟有執行多次，所以其實實際操作已經有點接近四次方了。**參考文獻**

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, "Introduction to Algorithms," Third Edition, The MIT Press, 2009.
2. R.C.T. Lee, S.S. Tseng, R.C. Chang, and Y.T.Tsai, "Introduction to the Design and Analysis of Algorithms," McGraw-Hill, 2005.
3. Anany V. Levitin, "Introduction to the Design and Analysis of Algorithms," 2nd Edition, Addison Wesley, 2007.
4. Richard Neapolitan and Kumarss Naimipour, "Foundations of Algorithms," Fourth Edition, Jones and Bartlett Publishers, 2010.
5. <https://buy.line.me/u/article/171337>
6. <https://www.cnblogs.com/xiexiaokui/p/3989053.html>
7. <https://blog.csdn.net/weixin_38442390/article/details/109059002>