國立臺南大學資訊工程學系

資工三「演算法」課程

第二次作業

**題目: Primality Testing**

|  |  |  |
| --- | --- | --- |
| 班級 | ： | 資工三 |
| 姓名 | ： | 陳博琳 |
| 學號 | ： | S10859007 |

老師：陳宗禧

中華民國 110年10月31日

# 目錄

1. **簡介及問題描述……….……………..…………………………………………1**
   1. **簡介…………….…………………………………………………………………………2**
   2. **問題………………………….……………………………………………………………4**
2. **理論分析….………………………..………………………………………8**
3. **演算法則….…………………………..……………………………………10**
4. **程式設計環境架構.………………………..…………………………………12**
5. **程式.…………………………………………..………………………………14**
6. **執行結果、討論與心得.………………………..……………………………18**

參考文獻………………………………………………………….…………………22

**(一) 簡介及問題描述**

給定一地圖，內有 n 個興趣點(POI, Point of Interest)，例：臺南古蹟、臺南小吃等。假設目 前有一個人或一個旅行團，對於臺南市的美食或古蹟有高度興趣，並且要規劃行程，但對於 該行程需要有資訊系統輔助，給定行前的建議，底下問題為規劃者需要知道的答案：

1. 簡介

使用Adjacency List，利用map上的點進行分析與旅遊規劃，尋找最短距離或是將convex-hall圈出尋找最短路線，或是讓地圖中的人可以最快速到達所有的點最短距離分配

2. 問題

**(1) 哪兩個 POIs 靠最近? 距離多少?**

求出圖中任意兩點之連線最短距離，用座標求出他們的距離。

**(2) 這些POIs的範圍有多大 (Convex-Hull，它的面積以及最遠的距離)?**

求出convex-hull需先求出任兩點的直線方程式，帶入線外的點判斷c為正或是負，全部同號者即為最大面積中的點

**(3)**

**(a) 假如有一個遊客(目前在某一個 POI a)要到所有 n 個 POIs，再回到原來的 POI a，則最短行程距離是多少(最佳解)? 時間複雜度?**

用階層手法將所有可能路徑累加獲得答案

**(b) 假如我們設計一個新演算法：(Convex-Hull-TSP Algorithm) i. 求出所有點的 Convex-Hull ii. 除 Convex-Hull 上的點外，其餘 POIs 找出離 Convex-Hull 邊最近的點投影 iii. 按照投影點依序由 Extreme Point 點開始旅遊，再繞回起始點 請問該演算法的時間複雜度(以 big O 表示)以及(行程)花多少距離?**

用上方求得的convex-hull將未在convex-hull內的點尋找最近的線，從邊上的點開始旅遊，依序走完離邊最近的點後走完convex-hull上的所有點

**(c) 比較 3(a)和 3(b)兩者的距離𝑑𝑎和𝑑𝑏、兩者比例 𝑑𝑏 /𝑑𝑎 ，以及比較兩者執行時間?**

**利用比值比較執行時間**

**(4) 假設有 n 個遊客，因時間有限，想要最短總距離的方式分別到 n 個不同 POIs 旅遊，要如何配對，以及求出最少的總距離和?**

利用Hungarian method先將所有的row減掉最小值，col在減掉最小值，用最少的線蓋所有的0如果覆蓋的線的總數等於人數完成，否則持續將覆蓋的線以外的數減掉這些數的最小值，直到線等於總數。從少的0開始分配工作

**(二) 理論分析**

首先第一個Basic Prime Test…

1. 哪兩個 POIs 靠最近brute-force algorithm理論

地圖上可以見到n個點此時需要將n點皆與n-1個點連接過全部總數為n\*n-1/2，a接到b與b接到a距離相同因此只需要計算一次。

2. Convex-Hull。brute-force algorithm理論

圖中最少的點連接的線可以包含所有的點，將途中所有的點兩兩連接帶入線外的點獲得c符號若相等則此兩點為convex-hull的邊點

3. 假如有一個遊客(目前在某一個 POI a)要到所有 n 個 POIs，再回到原來的 POI a，則 最短行程距離是多少(最佳解)exhaustive search + brute-force algorithm理論

利用點a到其餘所有點的距離，走過的點將不用再走世間複雜度為o(n!)

4. Convex-Hull-TSP Algorithm理論

先做完convex-hull將此圖形以外的點尋找距離圖形上最近的邊，邊上的起始點為頭，走完頭再從離頭最近的圖形外的點開始走，接續走到下一個起始點將路線走完

5. Hungarian method理論

將n個物品最有效分配給n個人，先將所有的row減掉最小值，col在減掉最小值，用最少的線蓋所有的0如果覆蓋的線的總數等於人數完成，否則持續將覆蓋的線以外的數減掉這些數的最小值，直到線等於總數。從少的0開始分配工作

**(三) 演算法則**

1. 第一個演算法(Algorithm)

哪兩個 POIs 靠最近brute-force algorithm

Int min minnodea minnodeb

For(int i=0;i<n;i++)

for(int j=0;j<i;j++)

dis=(map[x][i]-map[x][j])^2+(map[y][i]-map[j][y])^2

if(dis<min)

minnodea=i

minnodeb=j

1. 演算法時間複雜度(time complexity)

O(N^2)

因為n\*(n-1)/2

1. 演算法空間複雜度(space complexity)

O(n^2)

用來裝map的二維陣列

2. 第二個演算法(Algorithm)

Convex-Hull。brute-force algorithm理論

Int positive negative

For(int i=0;i<n;i++)

for(int j=0;j<i;j++)

for(int k=0;k<n;k++)

if(map[x][i]==map[y][i]&&(k!=i||k!=j))

//此時斜率分母維0額外處理

c = (double)map[x][i]-(double)map[x][k]; //帶入點

If(c>0)

Positive++

Else

Negative++

If(positive>0||negative>0)

Convex[x][count]=map[x][i]

Convex[y][count]=map[y][i]

Else //斜率不為0

a = (double)(map[y][i] - map[y][j]) / deltax;

b = map[y][i] - map[x][i] \* a;

c = a \* (double)map[x][k] + b - (double)map[y][k];

if(c>0)

positive++

else

negative++

If(positive>0||negative>0)

Convex[x][count]=map[x][i]

Convex[y][count]=map[y][i]

1. 演算法時間複雜度(time complexity)

O(n^3), …

將N\*(n-1)個點連線再將線外的所有(n-2)個點帶入，n\*(n-1)/2\*(n-2)

1. 演算法空間複雜度(space complexity)

O(n^2), 存放map的空間此處將會用到另一個n^2的空間存放convex-hull的點。

3. 第三個演算法(Algorithm)

假如有一個遊客(目前在某一個 POI a)要到所有 n 個 POIs，再回到原來的 POI a，則 最短行程距離是多少(最佳解)exhaustive search + brute-force algorithm理論

if (i < n) {

for (j = i; j <= vectornum; j++) {

tmp = num[j];

for (k = j; k > i; k--)

{

num[k] = num[k - 1];

}

num[i] = tmp;

perm(num, i + 1);

for (k = i; k < j; k++)

{

num[k] = num[k + 1];

}

num[j] = tmp;

}

}

else {

if (CountRoute(num) < min)

{

min = CountRoute(num);

}

}

1. 演算法時間複雜度(time complexity)

O(n!), …

利用遞迴計算所有n以外的點的距離算過的不用重複計算

1. 演算法空間複雜度(space complexity)

O(n^2), 存放map的空間

第四個演算法(Algorithm)

先做完convex-hull將此圖形以外的點尋找距離圖形上最近的邊，邊上的起始點為頭，走完頭再從離頭最近的圖形外的點開始走，接續走到下一個起始點將路線走完

//不再線上的node找最近的node

for (int i = 0; i < vectornum; i++)

{

if (edgenode[i] == 1)

{

continue;

}

else

{

min = 100000;

minedge = 10;

for (int j = 0; j < convexcount; j++)

{

if (edge[j].pointax == edge[j].pointbx)

{

a = 1;

b = 0;

c = edge[j].pointax;

distance = myabs((a \* map[x][i] + b \* map[y][i] + c)) / sqrt((double)(a \* a + b \* b));

if (distance < min)

{

min = distance;

minedge = j;

}

}

else

{

a = (double)(edge[j].pointay - edge[j].pointby) / (double)(edge[j].pointax - edge[j].pointbx);

c = edge[j].pointay - edge[j].pointax \* a;

b = -1;

distance = myabs((a \* map[x][i] + b \* map[y][i] + c)) / sqrt((double)(a \* a + b \* b));

if (distance < min)

{

min = distance;

minedge = j;

}

}

}

edge[minedge].closenode[edge[minedge].closenodecount][0] = i;

edge[minedge].closenode[edge[minedge].closenodecount][1] = min;

edge[minedge].closenodecount++;

}

}

//排序所有點中的距離

int temp1,temp2;

for (int k = 0; k < convexcount; k++)

{

for (int i = 0; i < edge[k].closenodecount; i++)

{

for (int j = 0; j < i; j++)

{

if (edge[k].closenode[i][1] > edge[k].closenode[j][1])

{

temp1 = edge[k].closenode[i][1];

edge[k].closenode[i][1] = edge[k].closenode[j][1];

edge[k].closenode[j][1] = temp1;

temp2= edge[k].closenode[i][0];

edge[k].closenode[i][0] = edge[k].closenode[j][0];

edge[k].closenode[j][0] = temp2;

}

}

}

1. 演算法時間複雜度(time complexity)

O(n^2)convex-hall以外的點n-a尋找最近的邊a-1,a為常數

1. 演算法空間複雜度(space complexity)

O(n^3)convex-hall一維陣列中用class儲存二維陣列的最近點

第五個演算法(Algorithm)

while (checknode())

{

min = 1000;

count++;

for (int i = 0; i < vectornum; i++)

{

for (int j = 0; j < vectornum; j++)

{

if (checkuse[i][j] < min && checkuse[i][j] != -1 && checkuse[i][j] != -2)

{

min = checkuse[i][j];

}

}

}

for (int i = 0; i < vectornum; i++)

{

for (int j = 0; j < vectornum; j++)

{

if (checkuse[i][j] == -2)

{

noderange[i][j] = noderange[i][j] + min;

}

else if (checkuse[i][j] == -1)

{

continue;

}

else

{

noderange[i][j] = noderange[i][j] - min;

}

}

}

}

int temp = 0;

int job;

while (temp < vectornum)

{

temp++;

for (int i = 0; i < vectornum; i++)

{

count = 0;

for (int j = 0; j < vectornum; j++)

{

if (noderange[i][j] == 0)

{

count++;

job = j;

}

}

if (count == 1)

{

finaljob[i] = job;

noderange[i][job] = -2;

for (int m = 0; m < vectornum; m++)

{

for (int n = 0; n < vectornum; n++)

{

if (noderange[m][n] == 0 && n == job && noderange[m][n] != -2)

{

noderange[m][n] = -1;

}

}

}

}

}

}

1. 演算法時間複雜度(time complexity)

O(n^3)尋找一人一景點時跑過n次尋找每個node對應，從0少的開始選，再2D ARRAY中尋找，需要n^2時間

1. 演算法空間複雜度(space complexity)

O(n^2)放置每個人到點的距離

**(四) 程式設計環境架構**

程式設計語言、工具、環境與電腦硬體等規格說明…

1. 程式語言

C in MS Windows

2. 程式開發工具

Visual C++ 2019

3. 電腦硬體

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

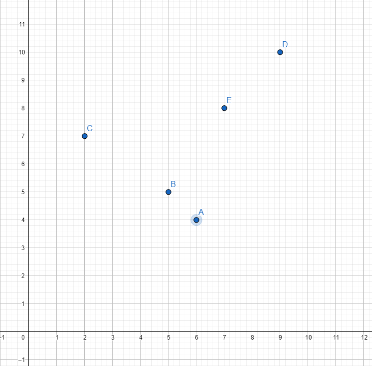
程式含source code, input code, and output code等…

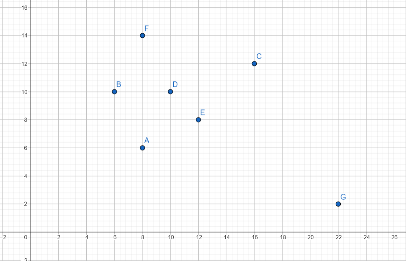
1. 主程式

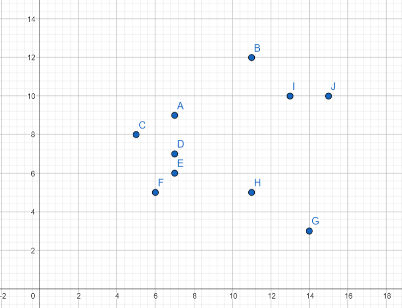
有提供cpp檔，請參閱，使用Adjacency List

2. Input Code Format

Three of examples for input use are in below….

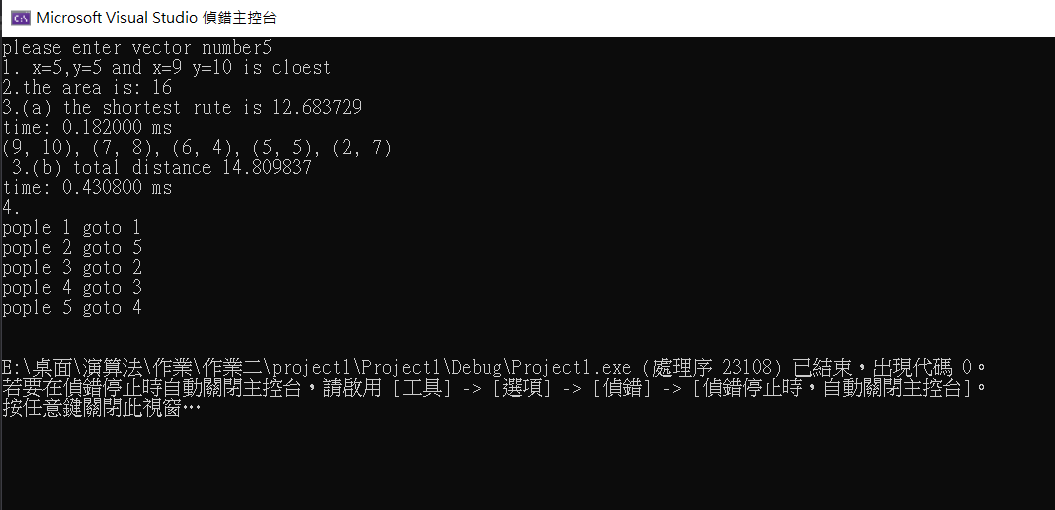
(1)5個點

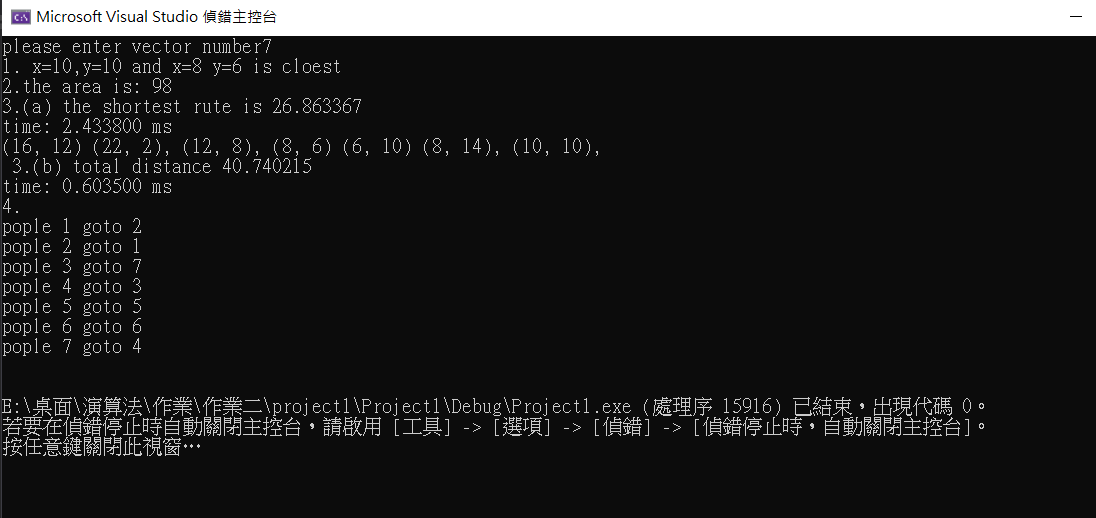
(2)7個點

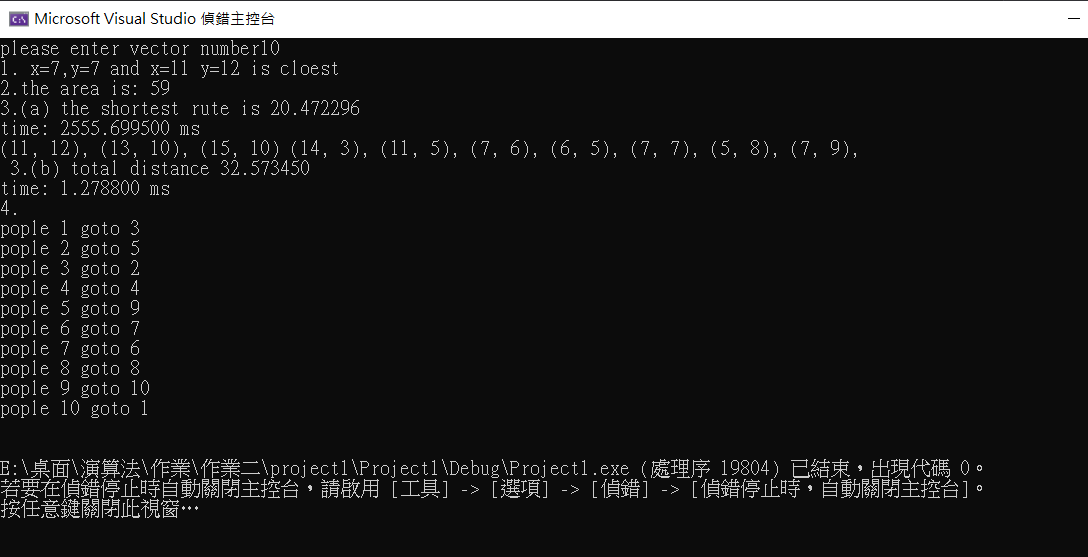
(3)10個點

3. Output Code Format

Three of examples for output use are in below….

(1) 

(2) 

(3) 

**(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. 執行結果

Output of program:

da/db距離比例。ta/tb時間比例

1. 12.68/14.80。0.182/0.43
2. 26.86/40.74。2.43/0.603
3. 20.47/32.57。2555.69/1.27

2. 討論

執行時間、問題大小等問題討論! 利用 MS Excel 畫出問題大小與執行時間的關係!

1. Running Time

可以發現5個點時時間沒有明顯的差異暴力解和7,10個點沒有太大的差異，但是n!到10的時候變得相當大，2555完全超出比例尺外

(2) Problem size *n*

i.5

ii.7

iii.10

3. 心得

最後一題分配景點再如何用最少的線包含所有的0花了不少時間，本來想要用15個點但是15!實在跑太久因此作罷，看來演算法真的很重要，雖然解果不是最短距離但是省下很多時間

**參考文獻**

1. https://www.1ju.org/cplusplus/factorial-program-in-cpp