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: Friends Graph**

|  |  |  |
| --- | --- | --- |
| 班級 | ： | 資工三 |
| 姓名 | ： | 黃詩豪 |
| 學號 | ： | S10259002 |

老師：陳宗禧

中華民國 104年11月04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**

1. 簡介

找出一個圖中的黨派問題，利用窮舉法設計演算法

2. 問題

給定 *n* 個朋友及其好友的關係，連結成一個朋友關聯圖*G=(V, E)*（Graph for Friends Linkage），朋友 *v∈V*, *e=(vi, vj) ∈ E*，vi, vj為朋友關係，*G*為無方向性圖。請設計與實作下列朋友關聯圖的功能：

1. 找出哪一個人有最多朋友
2. 找出該朋友關聯圖的直徑（Diameter of a Graph），並將其直徑（Path）列出來（Output）。
3. 利用Exhaustive Search（Brute-force）方法設計找*k*-Clique（朋友黨派*k*-ique）（input *k≧3*，找出所有的*k*-Clique）
4. 指定一點（人），利用Exhaustive Search（Brute-force）方法設計找最大朋友黨派（Clique），其Clique最大值（Size）以及列出包含哪一些朋友

Notes:

1. 實作程式時，請先說明你所用的圖（graph）表示方式，Adjacency Matrix or Adjacency List。
2. 必須分析n值變化（1~1000 以上…），以及所有功能執行時間變化的討論。

**(二) 理論分析**

1. 找出哪一個人有最多朋友

哪一個人有最多朋友等同於找哪一點有最多的邊（edge），*n*個點之中，每一個點最多有*(n-1)*個邊，因此最多需要找*n(n-1)*次，時間複雜度為***O(n2)***

2. 找出該朋友關連圖的直徑（Diameter of a Graph），並將其直徑（Path）列出來（Output）

利用廣路優先搜尋（Breadth-firs Search，BFS）對每一個頂點找出最短路徑表，然後從n個結果中找出最長的最短路徑，就得到無向圖的直徑，時間複雜度為O(n3)

3. 利用Exhaustive Search（Brute-force）方法設計找*k*-Clique（朋友黨派Clique）（input *k≧3*，找出所有的*k*-Clique）

n個點的無向圖，利用窮舉法找出所有的k-Clique，總共有

組合，故時間複雜度為***O(n!)***

4. 指定一點，利用Exhaustive Search（Brute-force）方法設計找最大*k*-Clique（朋友黨派Clique）

*n*個點的無向圖，利用窮舉法找出最大的Clique，因為選定其中一點，所以需要找出*(n-1)*個的所有可能，因此總共的組合有

故時間複雜度為***O(2n-1)***

**(三) 演算法則**

在以下的演算法中，對於圖的描述是使用adjacency matrix

1. 第一個演算法(Algorithm)

**ALGORITHM** *maxfriend*(A[0..n-1][0..n-1])

//**Input:** Two-dimentional Array A[0..n-1][0..n-1] of integer

//**Output:** The point has max edgs(max friends)

*tv* ← 0, *max* ← 0, *tmax* ← 0

**for** i ← 0 **to** *n - 1* **do**

**if** A[*0*][*i*] == 1

*max* ← *max* + 1

**for** *j* ← 1 **to** *n - 1* **do**

*tmax* ← 0

**for** *i* ← 0 **to** *n - 1* **do**

**if** A[*j*][*i*] == 1

*tmax* ← *tmax* + 1

**if** *max* < *tmax*

*max* ← *tmax*

*tv* ← *j*

**return** *tv*

1. 演算法時間複雜度(time complexity)

先找出第一次最多朋友的點，需要執行n次，然後從第二點開始找，需要執行n(n-1)=n2-n次，故時間複雜度為O(n2)

1. 演算法空間複雜度(space complexity)

程式執行過程未有動態配置的記憶體，所以空間複雜度為O(n2)，其中n為圖的頂點個數

2. 第二個演算法(Algorithm)

**ALGORITHM** *fdiameter*(A[0..n-1][0..n-1])

//**Input:** Two-dimentional Array A[0..n-1][0..n-1] of integer

//**Output:** The point has max edgs(max friends)

1. find the shortest path from vertex 1 to others by **Breadth-first Search**

2. find the shortest path from vertex *i* to others by **Breadth-first Search** (*i*=1, 2, 3,...,n)

3. if the result of the graph of the diameter from **Step 1.** is smaller than **Step 2.**,

store this result to **max**

4.if *i* <= n, go to **Step 2**

5. return the **max** **diameter** and **the path in graph**

1. 演算法時間複雜度(time complexity)

使用BFS找出特定點到其他點的最點路徑需要O(n2)，然後對n點都做一次BFS，因此時間複雜度為O(n3)

1. 演算法空間複雜度(space complexity)

靜態空間為輸入的adjacency matrix，空間複雜度為O(n2)，建立最短路徑表需要n\*2的動態分配空間，以及BFS執行中需要使用到的佇列大小n，因此空間複雜度為O(n2)

3. 第三個演算法(Algorithm)

**ALGORITHM** *FindkClique*(A[0..n-1][0..n-1], k)

//**Input:** Two-dimentional Array A[0..n-1][0..n-1] of integer

// Integer number k of the k-Clique

//**Output:** All of the k-Clique result

1. Initialize **com[0..k-1]** array, which is the combinations of ***k*** in ***n***

2. for each combination in **com[0..k-1]**, check the vertex in **com** whether a clique,

if true, then print the vertices

if false, then do next combination in **com[0..k-1]**

1. 演算法時間複雜度(time complexity)

指定特定的k(>=3)，找出n個頂點的無向圖中的k-Clique，利用窮舉法的方式，需要檢查n!/k!(n-k)!，因此時間複雜度為O(n!)

1. 演算法空間複雜度(space complexity)

靜態空間為adjacency matrix的大小n\*n，動態空間為紀錄c(n, k)組合的陣列k，因為n>=k，因此空間複雜度為O(n2)

4. 第四個演算法(Algorithm)

**ALGORITHM** *FindMaxClique*(A[0..n-1][0..n-1], vertex)

//**Input:** Two-dimentional Array A[0..n-1][0..n-1] of integer

// Specific **vertex** number of integer

//**Output:** The max clique with the **vertex**

1. Initialize **nums[0..n-1]** value from 0 to *n-1*

2. Generate combinations of {0, 1, 2,..., n-2} to **idx** array with length **k**=1, 2, 3,.., n-1

3. for each **idx** finding the **k**-clique

4. if **k**-clique is the maximum, then print the maximum clique and the member of the clique

1. 演算法時間複雜度(time complexity)

從特定點的點找出最大的黨派，對除了該點以外的點，利用窮舉法的方式，直到找到最大的黨派，因此需要驗證C(n-1, 1) + C(n-1, 2) + C(n-1, 3) + ... + C(n-1, n-1) = 2n-1次，故時間複雜度為O(2n-1)

1. 演算法空間複雜度(space complexity)

靜態空間主要是adjacency matrix的大小n\*n，動態空間為紀錄有n個點的陣列大小n，加上紀錄組合的陣列大小1 + 2 + 3 + ... + (n-1)= n(n-1)/2，還有紀錄最大黨派資訊的陣列大小2 + 3 + 4 + ... + n = (n+2)(n-1)/2，故空間複雜度為O(n2)

**(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 Linux GCC

OS: Linux Mint 17.2 "Rafaela" – Cinnamon (64-bit)

2. 程式開發工具

Eclipse，Version: Mars Release (4.5.0)

Eclipse C/C++ Plugin: CDT 8.8.0 for Eclipse Mars

Compiler: Linux GCC

3. 電腦硬體

CPU: Intel Xeon E3-1231 v3，Main Memory: 32GB

**(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

見附件Source\_code.zip

2. Input Code Format

使用產生n個點的無向圖，並且用亂數決定哪兩點有相鄰，然後以Adjacency matrix的資料結構儲存圖，儲存於檔案TestVertex\_$i.txt檔案（$i為5, 10, 15）

(1) TestVertex\_5.txt

(2) TestVertex\_10.txt

(3) TextVertex\_15.txt

3. Output Code Format

T程式輸出結果直接顯示於Terminal上，分別印出題目所要輸出的資訊，以下輸出範例以TestVertex\_5.txt輸入檔案呈現

(1)

1. Vertex "0" has max friends.

2. Diameter: 2

Path: [ 1] -> [ 0] -> [ 4]

3. 3-Clique: 1, 2, 4

3-Clique: 1, 3, 4

3-Clique: 1, 3, 5

3-Clique: 1, 4, 5

3-Clique: 3, 4, 5

4. Maximum clique: 4-Clique (1, 3, 4, 5)

**(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:

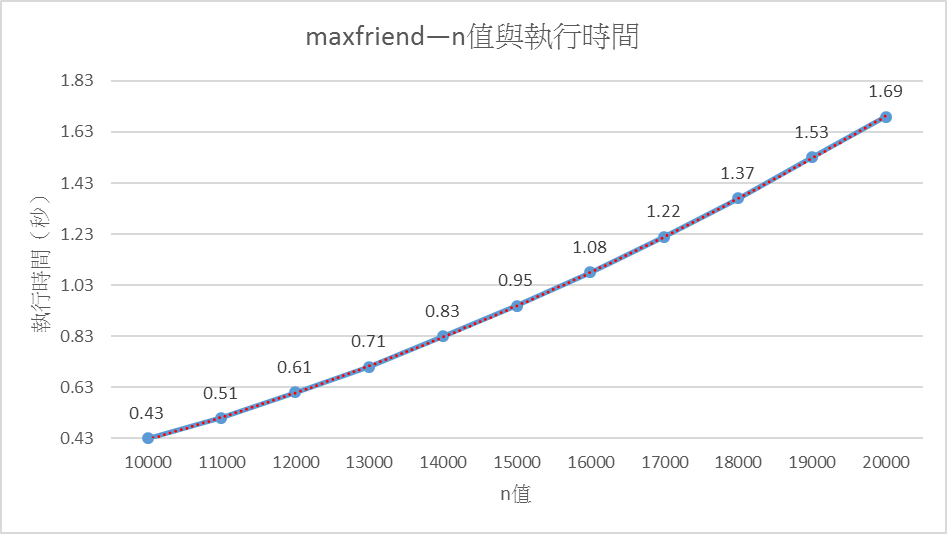
正確輸出最多朋友的點，該圖的直徑與所經過的點，所有的指定的k-Clique，以及指定的頂點中最大的Clique

2. 討論

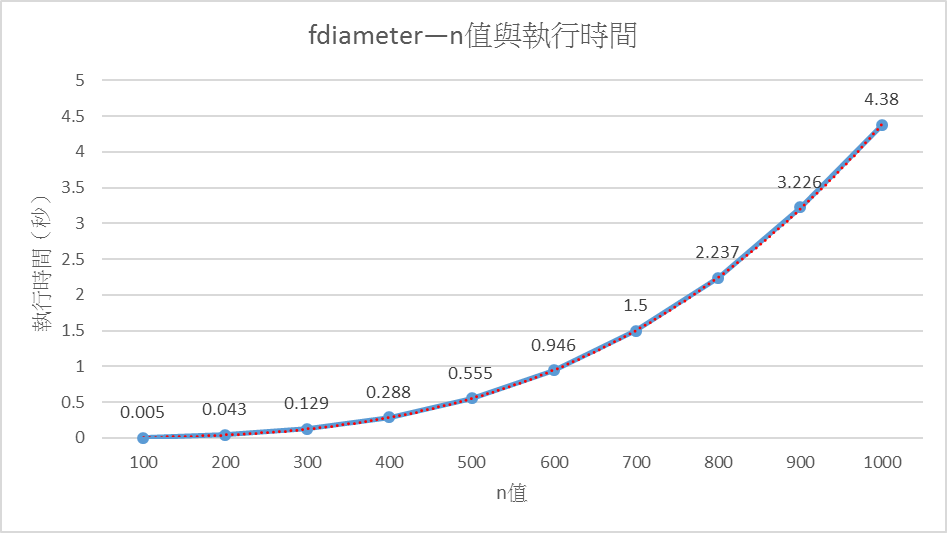
執行時間、問題大小等問題討論

1. Running Time

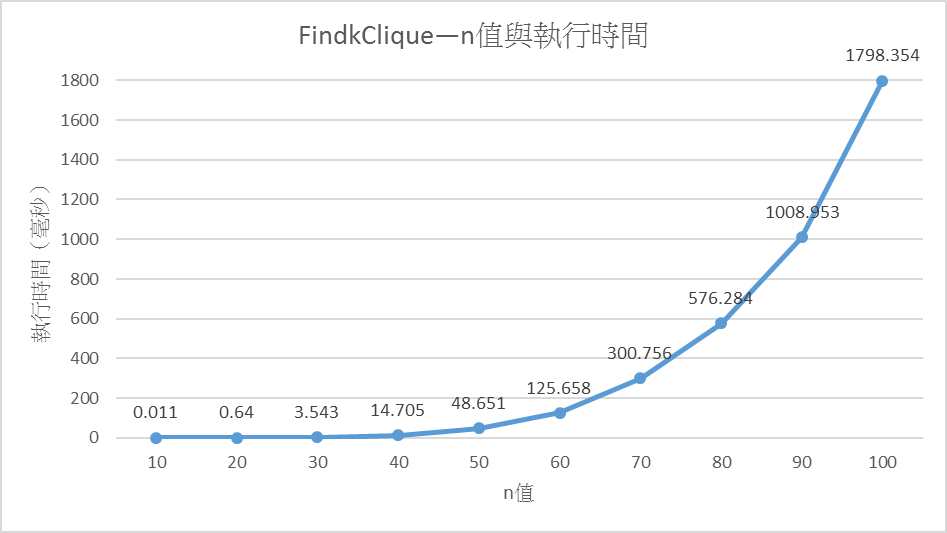
i.在maxfriend方法中，時間複雜度為O(n2)，下圖為此方法實際執行時間，實驗結果與演算法分析的結果近似



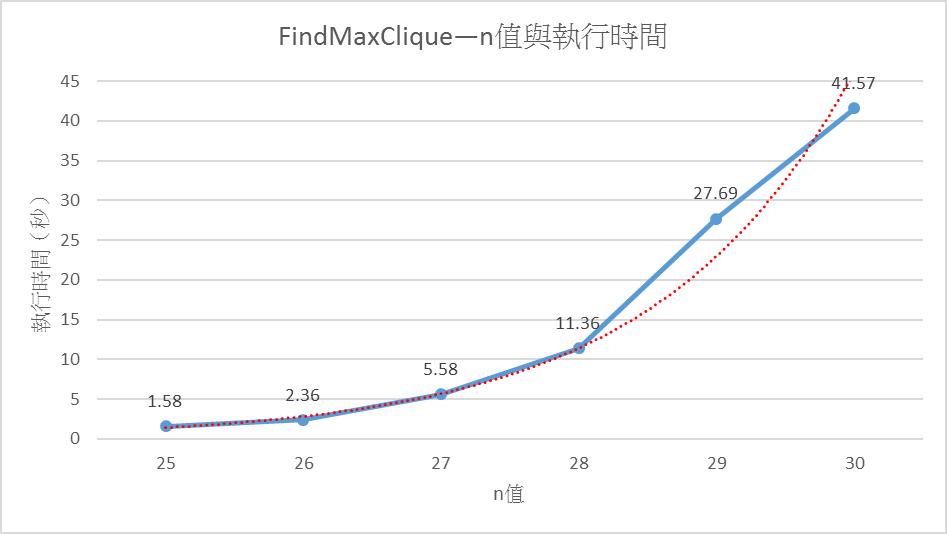
ii.在fdiameter方法中，使用BFS找尋每一對頂點的最短路徑，然後再從這些最短路徑中找出最長的路徑，即成為圖的直徑，而實驗結果與預期相近



iii.在FindkClique方法中，找出輸入的k的所有黨派，在每一實驗中，皆是尋找k=5的黨派，因此執行時間會偏向n!/5!(n-5)!的曲線



iv.在FindMaxClique方法中，給定某一個頂點，找出有這個頂點所在的最大的黨派，因為有n的頂點且已經選定某一頂點，所以在窮舉法中需要枚舉C(n-1, k){k=1, 2, 3,..., (n-1)}的所有可能，利用二項式定理，可以得知曲線會較偏向於2n的曲線



(2) Problem size *n*

i.maxfriend方法只隨著輸入的的adjacency matrix的大小變動

ii.fdiameter方法除了輸入的adjacency matrix大小外，還有BFS執行所需要的佇列大小n，以及儲存計算結果的distance table n\*2的大小，但因為只記錄最大最長的最短路徑的distance table，因此程式只隨著輸入的二為陣列大小成長，不因程式執行過程中的distance table增加

iii.FindkClique方法中，輸入的相鄰矩陣大小n\*n，程式執行過程中要找出指定的k-Clique，因為要枚舉C(n, k)的組合，需要另外的k大小的陣列存枚舉的結果

iv.FindMaxClique方法中，輸入的相鄰矩陣大小n\*n，加上枚舉k=1, 2, 3,..., (n-1)所有組合的結果，需要動態地配置記憶體外，還需要2, 3, 4,..., n的陣列空間大小儲存最大Clique的資訊，故程式會除了會隨著輸入的相鄰矩陣大小增加外，還會隨著枚舉數k的增加而增加，直到k的中間值為止開始下降

3. 心得

這次的作業當中，覺得最困難的部分在於第二、三小題的尋找圖的直徑與枚舉所有組合的演算法設計，在查找圖的直徑資料過程中，發現找圖得的直徑就有分有、無向圖，圖是否有權重的方法，這次的作業中使用無向又無權重的圖，所以利用廣度優先搜尋的方法找出某一點到其他點的最短路徑，然後重複此步驟從1到n個點都做一次，最後在找出最長的最短路徑。然後是第三小題如何用程式枚舉C(n, k)的所有組合情況，嘗試過backtrack方法，發現此方法枚舉的過程中，並不是按照字典順序枚舉的，所以另外找到了以字典順序枚舉的方法，使得枚舉的方法可以獨立出來，以C(n, k)的形式枚舉，如此一來，對於第三小題的執行速度更有效率，同時也使得在設計第四小題演算法的過程中，更加的方便。

最後，在這次的作業中，體會到窮舉法在演算法執行中，需要非常大量的計算，導致找出問題解的時間過久，單從第三小題中可以發現，如果圖的頂點數為30，要找出15點的黨派就需要執行1.5億次，且還不包含副程式內部的執行次數，更可以想到第四小題找出最大黨派需要2n-1的次數，當頂點數為30就需要執行10億次，因此以後在設計演算法的時候，會盡量避免次方與階層的時間複雜度方法。

**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. http://faculty.simpson.edu/lydia.sinapova/www/cmsc250/LN250\_Weiss/L21-MinPath.htm
6. https://j101044.wordpress.com/2014/08/19/algorithmc-c-%E4%BD%87%E5%88%97queue/, 2014/08/19