

國立成功大學

Data Mining 資料探勘

Project 3
Link Analysis

姓名：金雅倫

學號：P96074105

目錄

一、目標說明

Goal Description.....3

二、實作細節

Implementation detail.....4

三、結果分析與討論

Result analysis and discussion.....7

四、運算效能分析

Computation performance analysis.....26

五、心得與討論

Thoughts and discussion.....27

一、目標說明< Goal Description >

概述

- Please implement
 - **HITS** and **PageRank** (Lecture 7, P37, random jumping probability, i.e., damping factor=0.15) and calculate authority, hub and PageRank values for the following **8** graphs
 - 6 graphs in project3dataset
 - 1 graphs from project1 transaction data (connect items in each row, **bi-directed** or **directed**)
 - **SimRank** to calculate pair-wise similarity of nodes (choice any parameter C you like) , using
 - first **5** graphs of project3dataset.
- Find a way (e.g., add/delete some links) to increase hub, authority, and PageRank of Node 1 in first 3 graphs respectively.

本專案旨在實作鏈結分析(Link Analysis)的三個演算法，包括：HITS(Hyperlink-Induced Topic Search)、PageRank 以及 SimRank，搭配 Dataset 分析節點與節點之間的關係。HITS 和 PageRank 主要為分析網頁之間超連結、點閱率排名的方法，而 SimRank 主要為分析網頁之間的相似程度。Dataset 包含 8 個 graph data，其中 6 個 graph data 為此 project3 的 dataset，而 graph 7 為實作 project 1 所產生的交易資料集，graph 8 為 project 1 交易資料中所產生的關聯法則推導出的 graph，每個皆以(.txt)檔的格式儲存。其中 graph 7 及 graph 8 自行根據 project 1 所產生出的 dataset 放在此 project 中的 hw3dataset 中，與前 6 個 graph 相互呼應。

專案最後須針對前 3 張 graph 找到一方法提升節點 1 的目錄值(Hub)、權威值(Authority)及 PageRank 值。

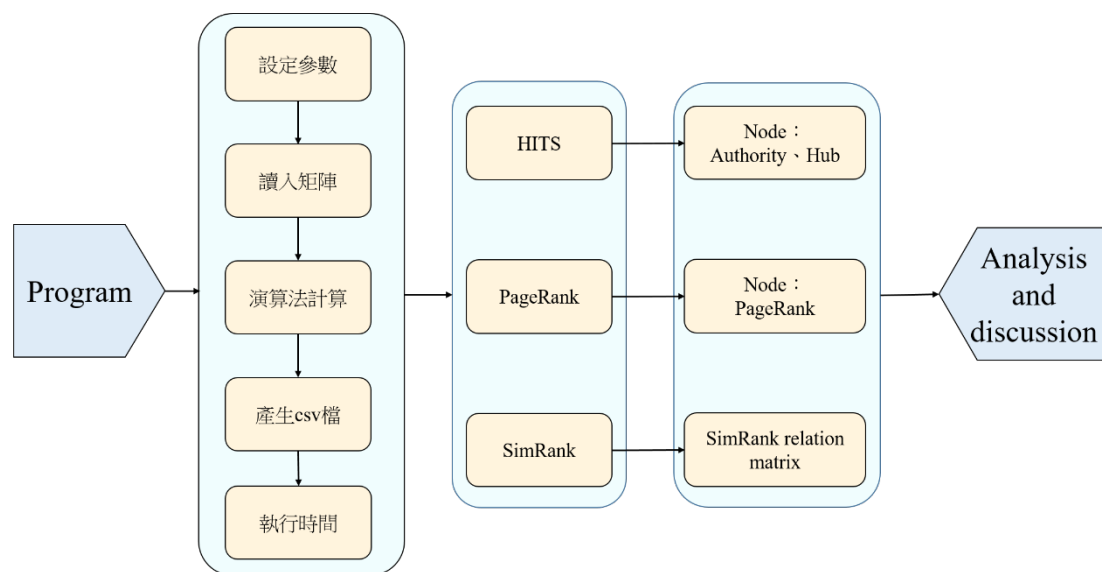
二、實作細節< Implementation detail >

概述

✧ 作業環境：Windows 10

✧ 程式語言：Python

本專案主要流程為先輸入要分析的 dataset 資料，經過三個鏈結分析的演算法，產生結果進行討論與分析。其中三個演算法的程式碼運作過程皆可以分為三個步驟，如下圖一所示，首先先進行演算法中設定參數的動作，再將每個 dataset 之資料讀入矩陣，第二步將每個資料帶入演算法進行計算，第三步為輸出每個 dataset 之結果，並把資料結果輸出產生 csv 檔案，並記錄每個 dataset 在三個演算法中產生的執行時間。



圖一 實作流程圖

實作三個演算法程式中，HITS 及 PageRank，皆使用設定容錯常數 (ERROR_TOLERANCE) 判定迴圈何時結束，此常數在 SimRank 分析較大量資料時分會大幅降低分析速度，由於輸入資料皆為小量，所以單純以設定其迴圈的迭代次數來判斷收斂。

✧ HITS :

```

P96074105_HITS.py - F:\Project3_1215\P96074105_HITS.py (3.6.4)
File Edit Format Run Options Window Help

inp = open(FILE_INPUT_NAME)
for row in inp.readlines():
    source, target = row.split(',')
    file_list.append([int(source), int(target)])
    if int(source) not in pages_arr:
        pages_arr.append(int(source))
    if int(target) not in pages_arr:
        pages_arr.append(int(target))
inp.close()

page_num = max(pages_arr)
graph = np.zeros((page_num, page_num))
for row in file_list:
    graph[row[0]-1][row[1]-1] = 1

hub = np.full((page_num), 1.0)
aut = np.full((page_num), 1.0)
error = sys.maxsize
start_time = time.time()
while error > ERROR_TOLERANCE:
    new_hub = np.full((page_num), 0.0)
    new_aut = np.full((page_num), 0.0)
    max_hub = 0.0
    max_aut = 0.0
    error = 0.0

    for x in range(0, page_num):
        for y in range(0, page_num):
            if graph[x][y] == 1:
                new_hub[x] += aut[y]
                new_aut[y] += hub[x]

    for z in range(0, page_num):
        if new_hub[z] > max_hub:
            max_hub = new_hub[z]
        if new_aut[z] > max_aut:
            max_aut = new_aut[z]

    for s in range(0, page_num):
        new_hub[s] = new_hub[s]/max_hub
        new_aut[s] = new_aut[s]/max_aut
        error += (abs(new_hub[s] - hub[s]) + abs(new_aut[s] - aut[s]))
    hub[s] = new_hub[s]
    aut[s] = new_aut[s]

```

開始讀檔

將所有不重複的頁面讀入

頁面最大數量

圖形矩陣初始化

連接資料讀進圖形

初始化 Hub Authority

計算數值、更新值

找到最大的 Hub Authority

正規化

✧ PageRank :

```

inp = open(FILE_INPUT_NAME)
for row in inp.readlines():
    source, target = row.split(',')
    file_list.append([int(source), int(target)])
    if int(source) not in pages_arr:
        pages_arr.append(int(source))
    if int(target) not in pages_arr:
        pages_arr.append(int(target))
inp.close()

page_num = max(pages_arr)
graph = np.zeros((page_num, page_num))
for row in file_list:
    graph[row[0]-1][row[1]-1] = 1

pr_vector = np.full((page_num), 1.0)

temp1, temp2 = np.unique(graph, return_counts=True)
link_num = temp2[1]

for i in range(0, page_num):
    for j in range(0, page_num):
        if graph[i][j] == 1.0:
            graph[i][j] = graph[i][j] / link_num

graph = graph.T

page_ranks_i = np.full((page_num, page_num), 0.0)
page_ranks_j = np.full((page_num, page_num), 0.0)
start_time = time.time()
for i in range(0, page_num):
    for j in range(0, page_num):
        DAMP_FACTOR = 0.15
        page_ranks_j[i][j] = (1.0 - DAMP_FACTOR) / page_num
        page_ranks_i[i][j] = (DAMP_FACTOR * graph[i][j]) + page_ranks_j[i][j]

error = sys.maxsize
new_pr_vector = np.full((page_num), 1.0)

while error > ERROR_TOLERANCE:
    error = 0.0
    for i in range(0, page_num):
        temp = 0.0
        for j in range(0, page_num):
            temp += page_ranks_i[i][j] * pr_vector[j]
        new_pr_vector[i] = temp

    for i in range(0, page_num):
        error += abs(new_pr_vector[i] - pr_vector[i])
        pr_vector[i] = new_pr_vector[i]

print("--- Excution time %s seconds ---" % (time.time() - start_time))

```

開始讀檔

將所有不重複的頁面讀入

頁面最大數量

圖形矩陣初始化

連接資料讀進圖形

將 node PageRank 初始化

平分每個連結

轉置

套用公式

調整係數

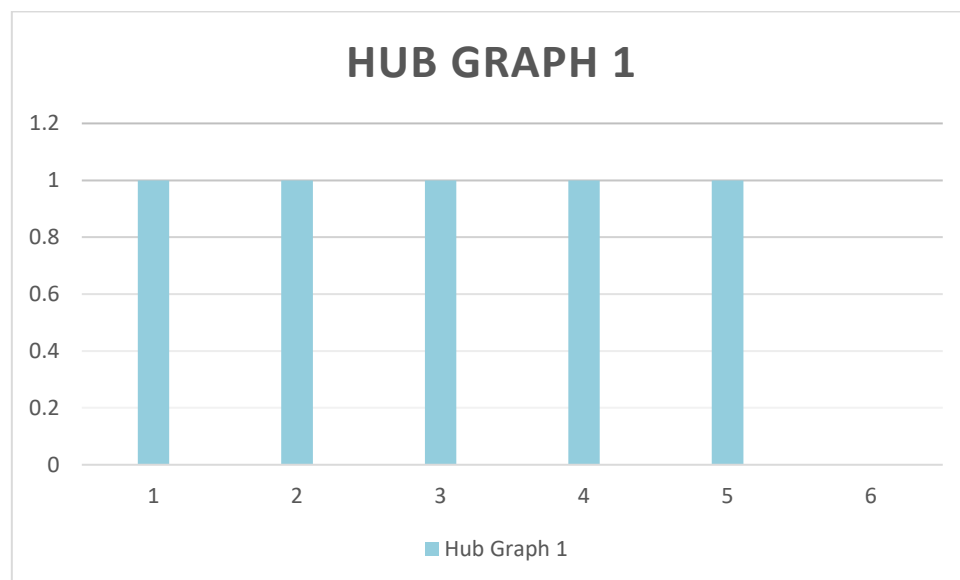
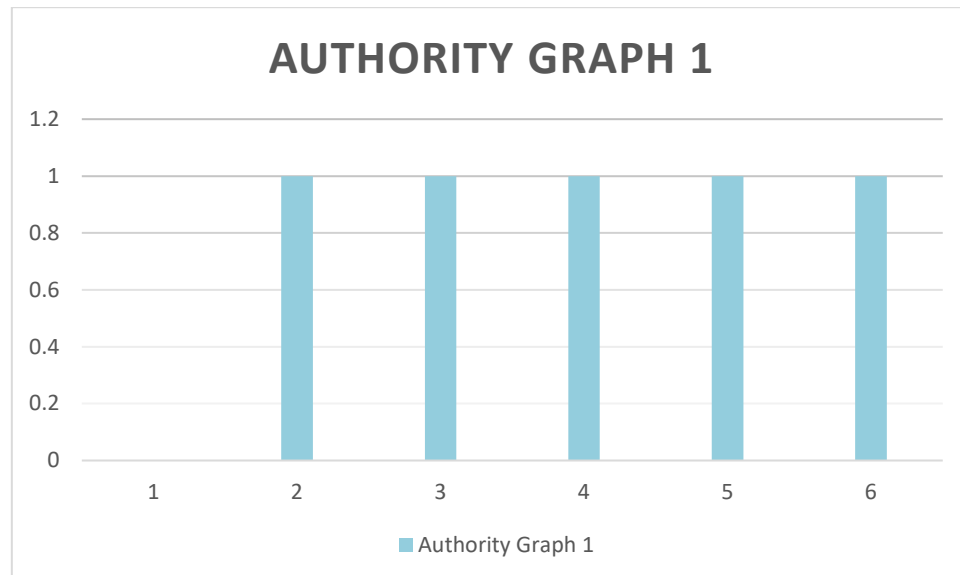
疊代

紀錄時間

三、結果分析與討論< Result analysis and discussion >

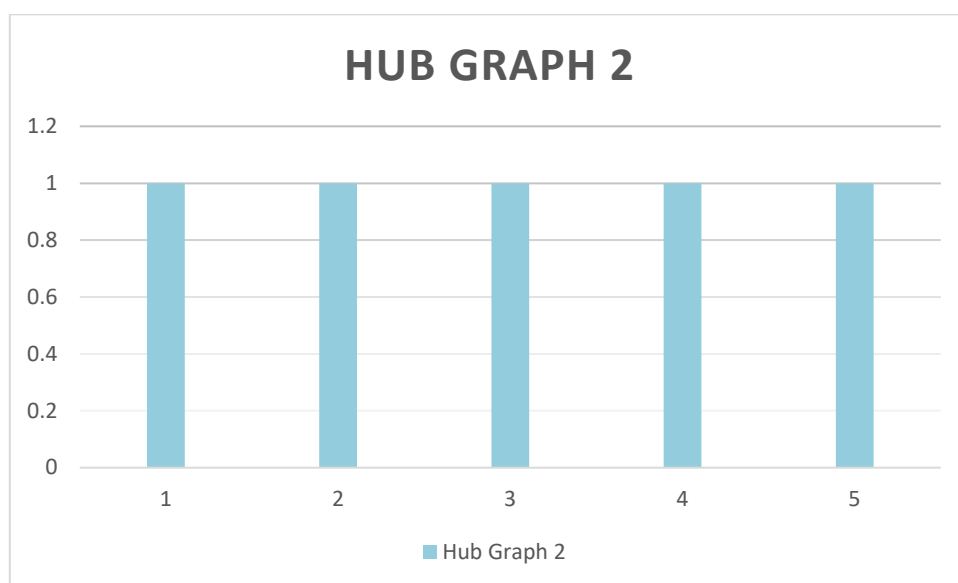
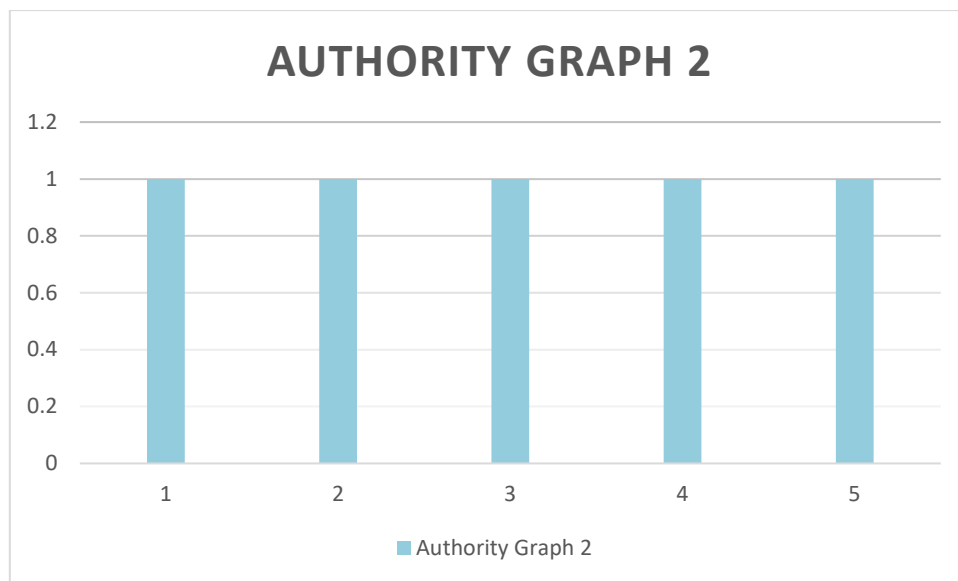
A. HITS 演算法

✧ Graph 1 :



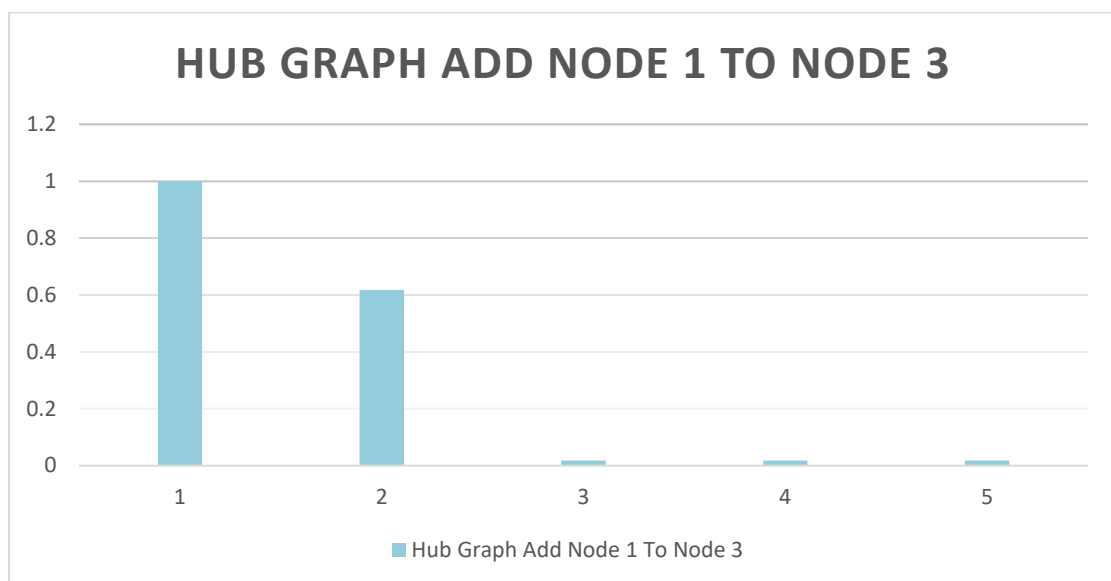
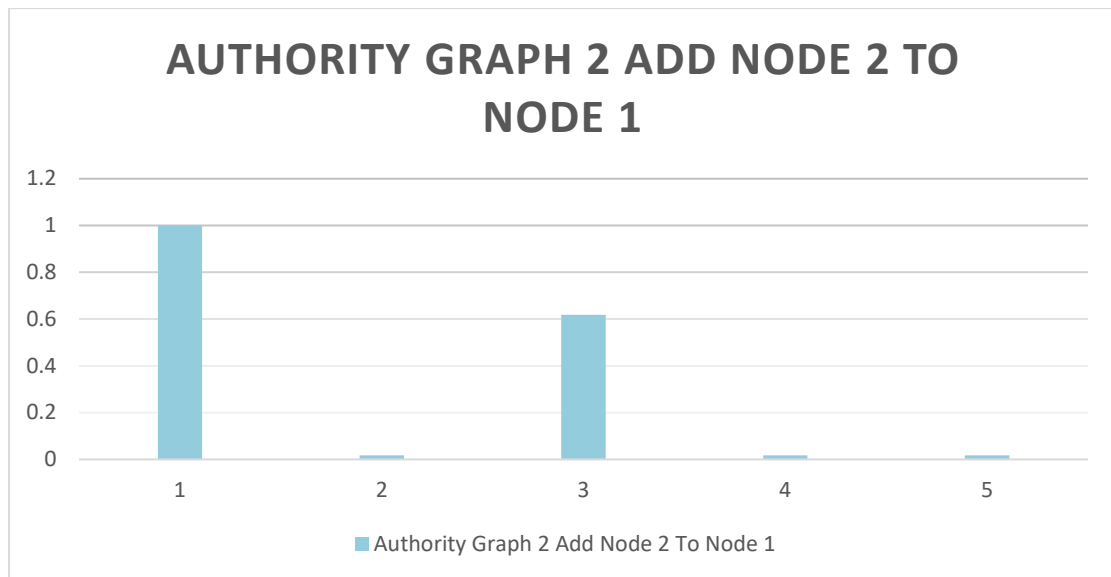
Graph 1 為一個串接式連接，1—2—3—4—5—6 的連接狀況，目前節點 1 的權威值為 0 以及節點 6 的目錄值為 0，如果想增加節點 1 的權威值，可以將其他點連接至節點 1;想增加節點 6 的目錄值可以把節點 6 連至其他節點。

✧ **Graph 2 :**



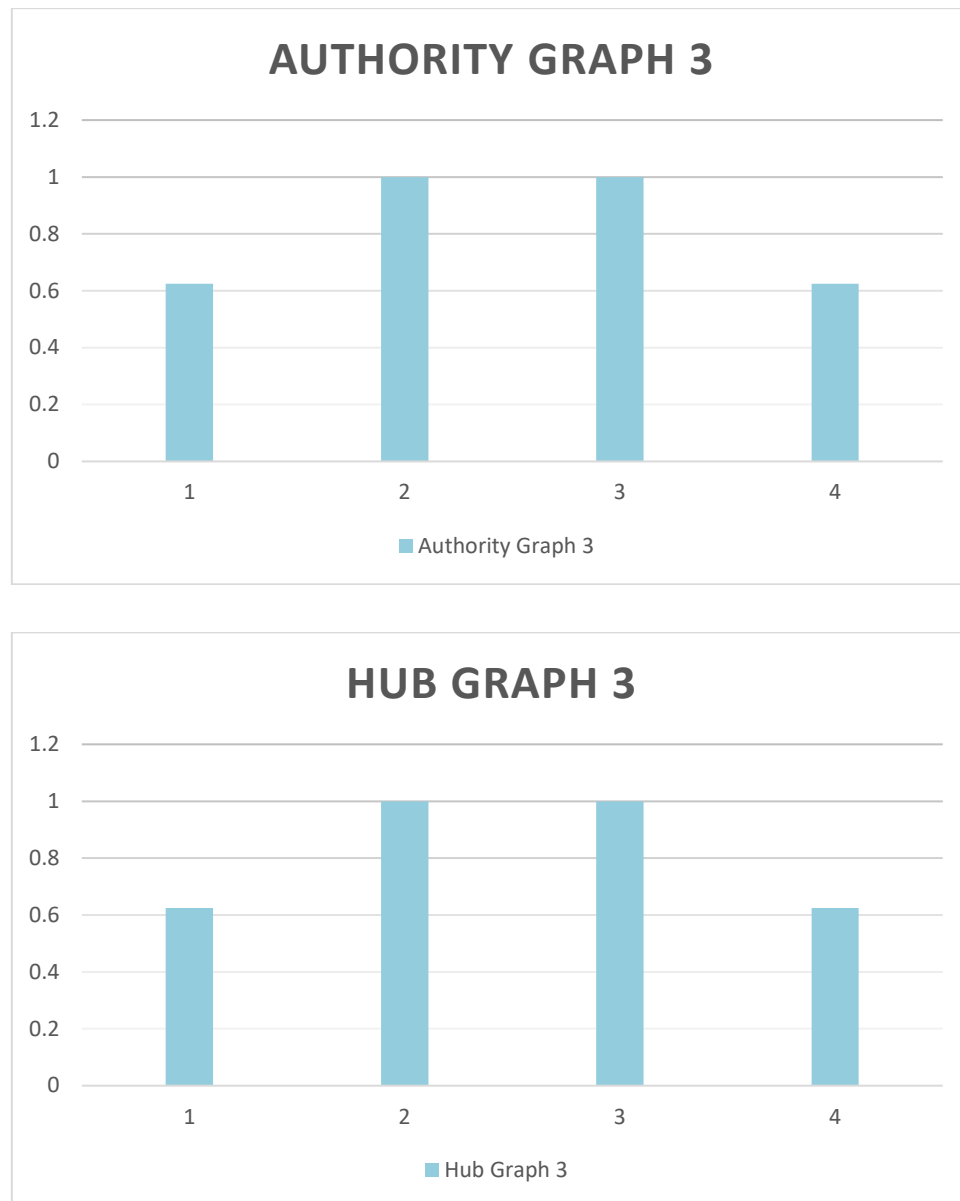
Graph 2 的連接情況呈現一個 Circle 的圖形，所以每個節點的權威值以及目錄值都恰好是最大值 1 的情況。

✧ **Graph 2 提高 Node 1 值：**



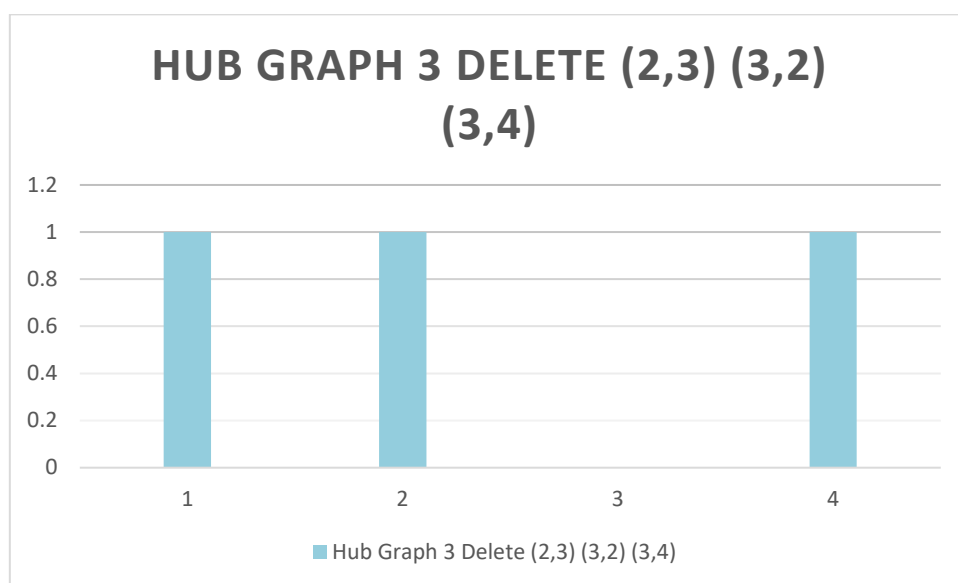
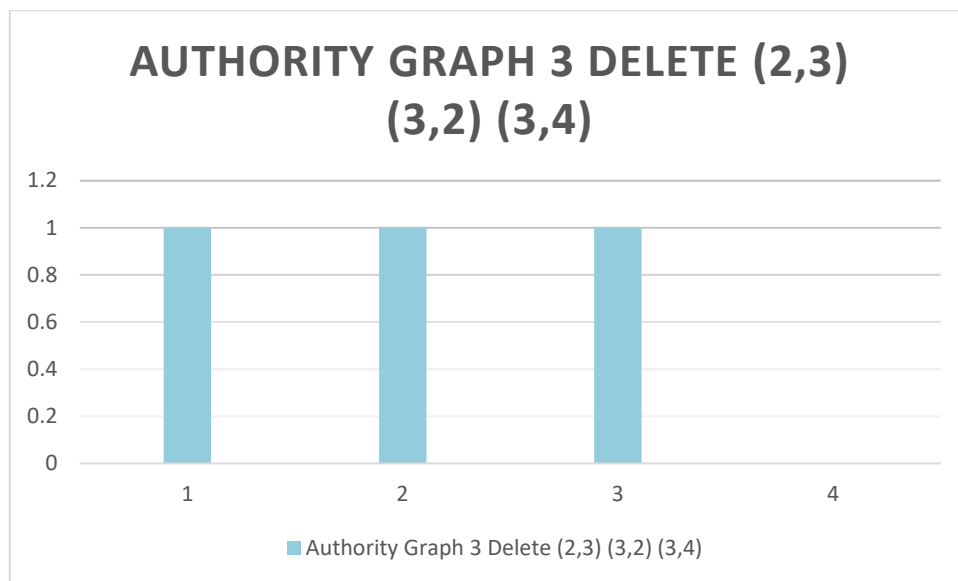
如果要把節點 1 的權威值變成最高，可以加入節點 2 連結節點 1;想把節點 1 的目錄值變成最高，可以加入節點 1 連結節點 3，如上兩張圖所示。

✧ **Graph 3 :**



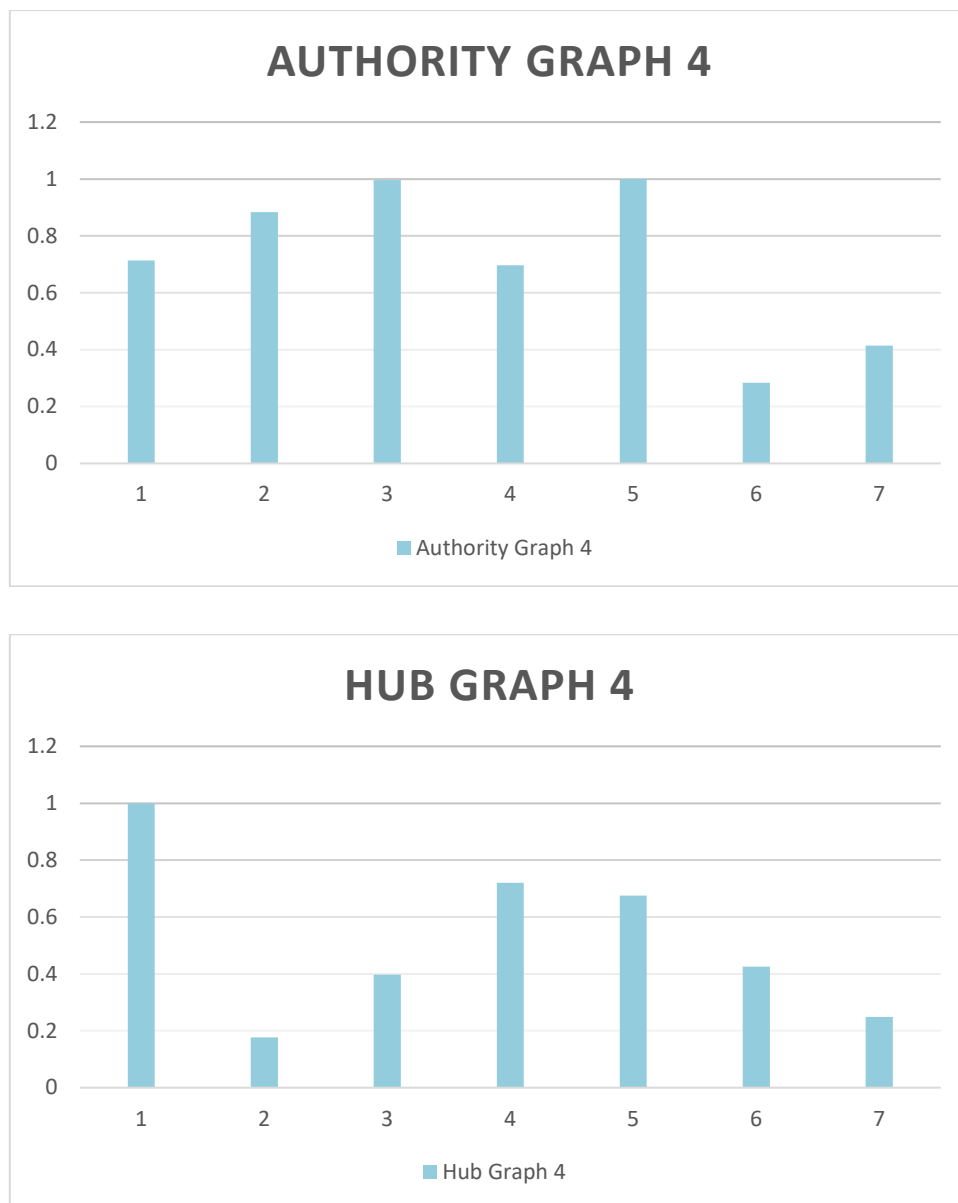
Graph 3 中可以觀察到節點 2 及節點 3 的權威值以及目錄值目前都是最高的，節點 2、3 的權威值能最高是因為這兩個節點被指向的次數都是相同並且最多次的;而目錄值最高是因為這兩個節點指向的數量相同，因此都是最高的兩次。

☆ **Graph 3 提高 Node 1 值：**



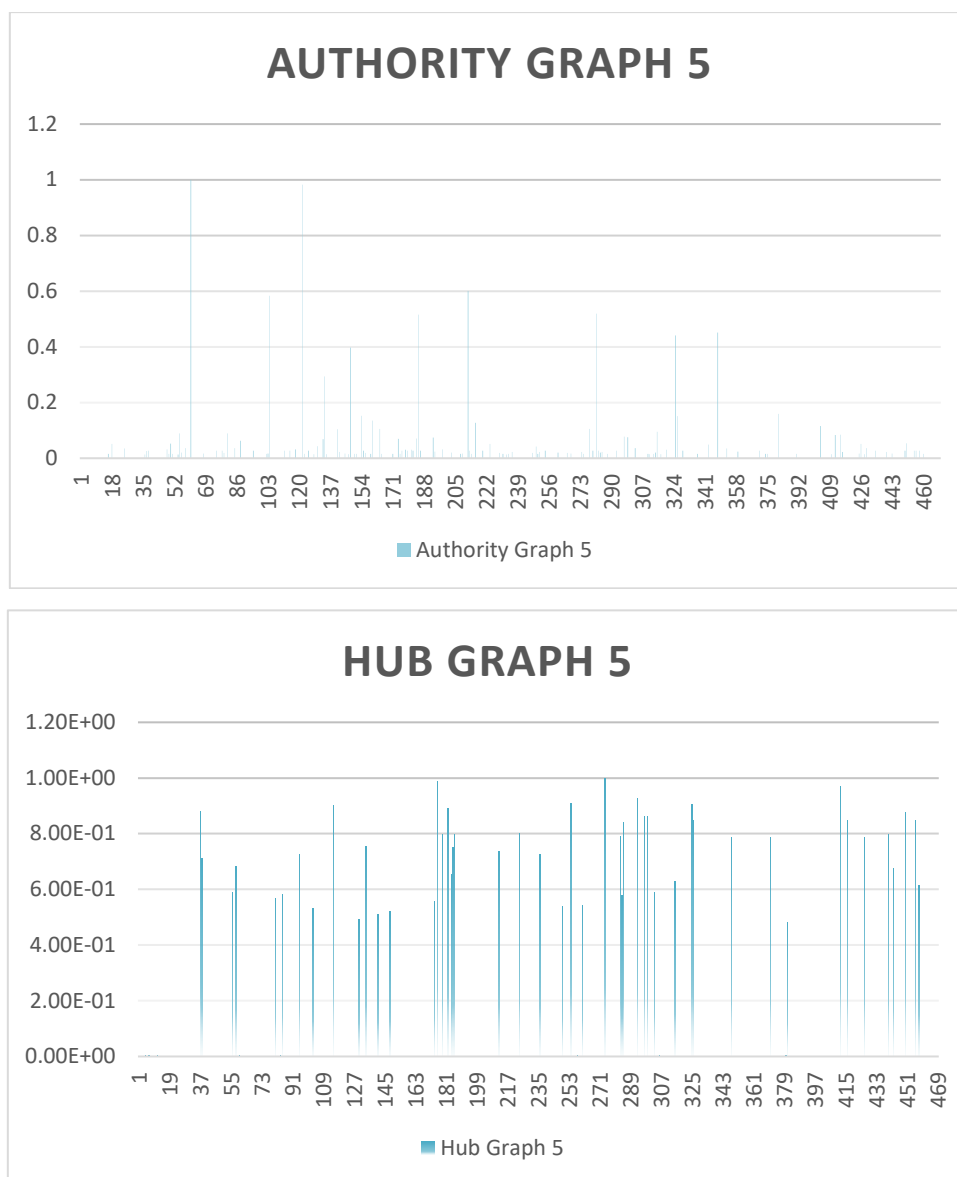
若想提高節點 1 的權威值以及目錄值可以透過刪除(2,3) (3,2) (3,4)這三個連結，同時也可以觀察到節點 4 的權威值以及節點 3 的目錄值會降為 0。

✧ **Graph 4 :**



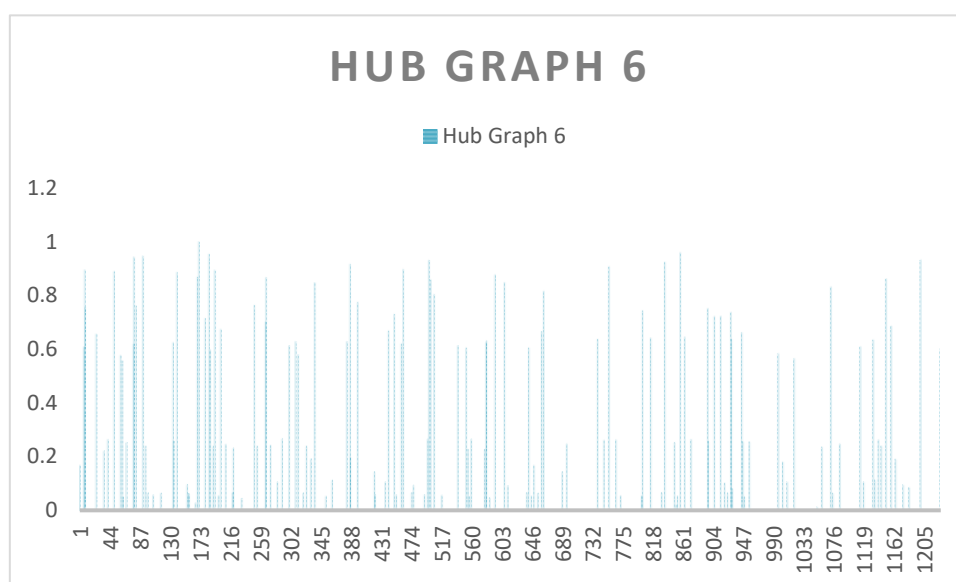
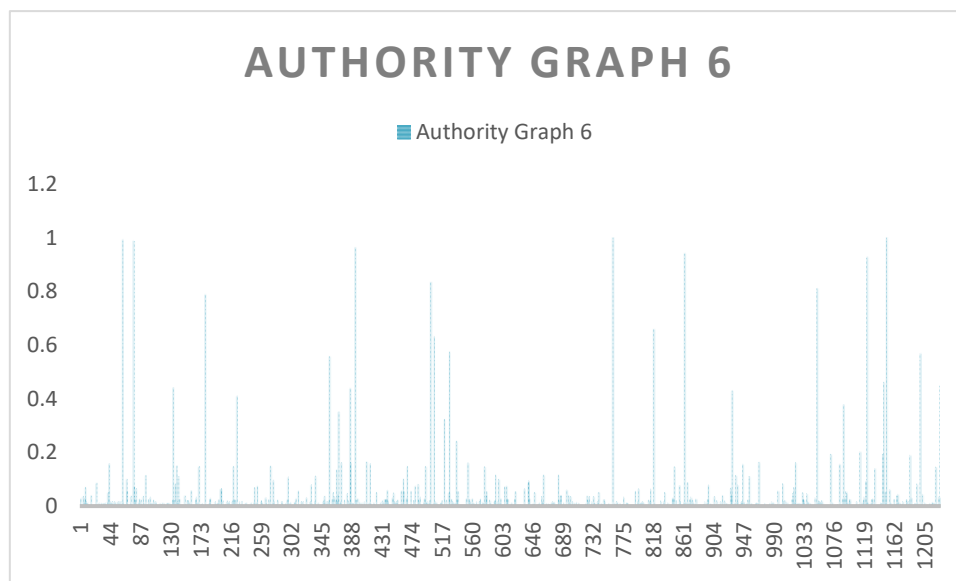
先以權威值分析來看，節點 2、3 以及 5 的值是相對來說較高的，可以推測出這三個頁面可能是最常從其他頁面連結至的頁面;以目錄值來看的話，節點 1、4、和 5 為前三名，可以推測出可能各為某平台的主要頁面。

✧ **Graph 5 :**



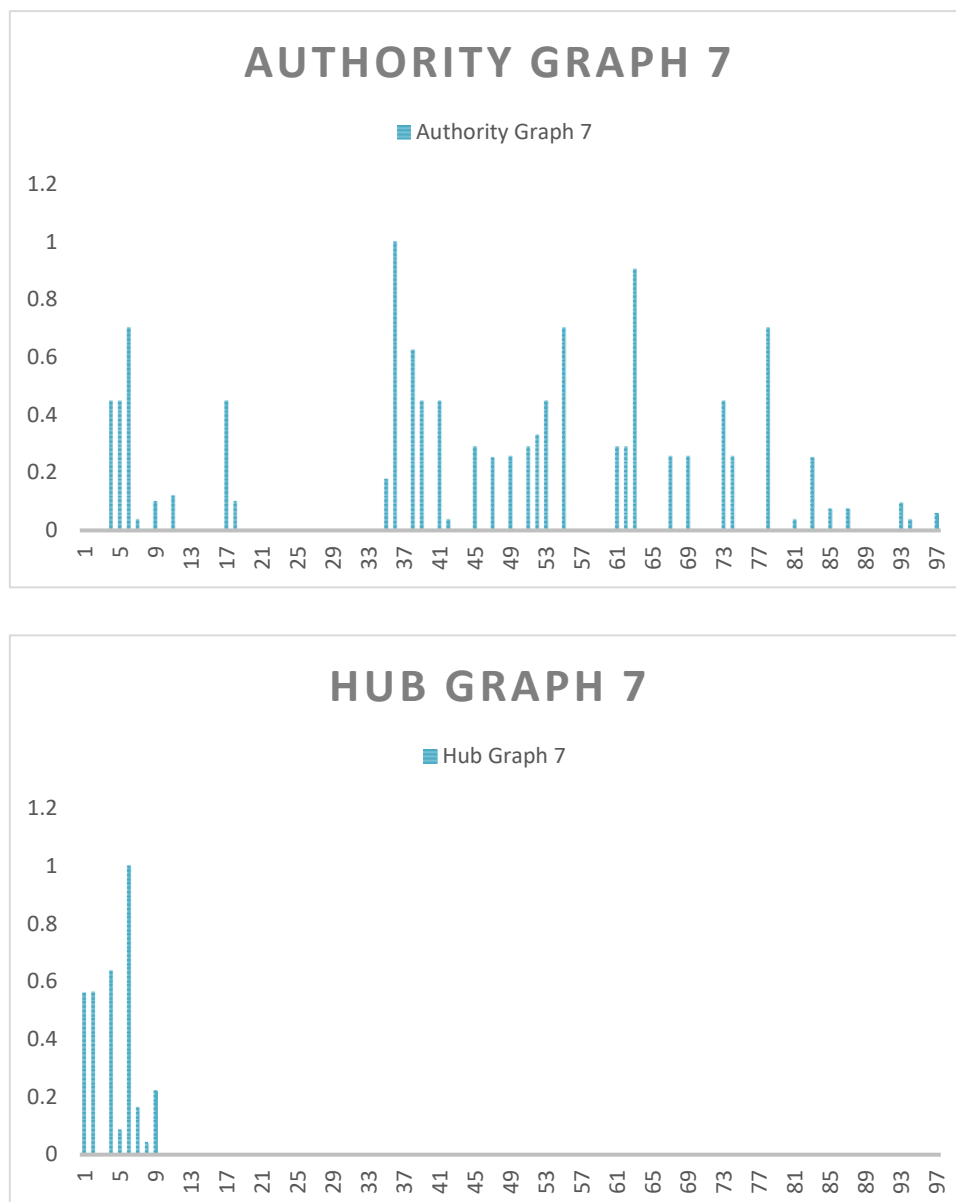
Graph 5 的資料量與前幾張 Graph 相對來說高出許多，Graph 5 中的節點超過百個以上，但分析圖中可以看出高權威以及高目錄的值都為特定某個節點，高目錄節點比高權威節點多出許多，以實際例子來描述的話，可以用網路購物來當作一個代表，當我們想買日用品、服飾、化妝品等的時候，會在搜尋引擎鍵入想要購買的商品，然後會跑出各個購物網站對於此商品的相關頁面，所以多數的高目錄值對應到少數的高權威值。

✧ **Graph 6 :**



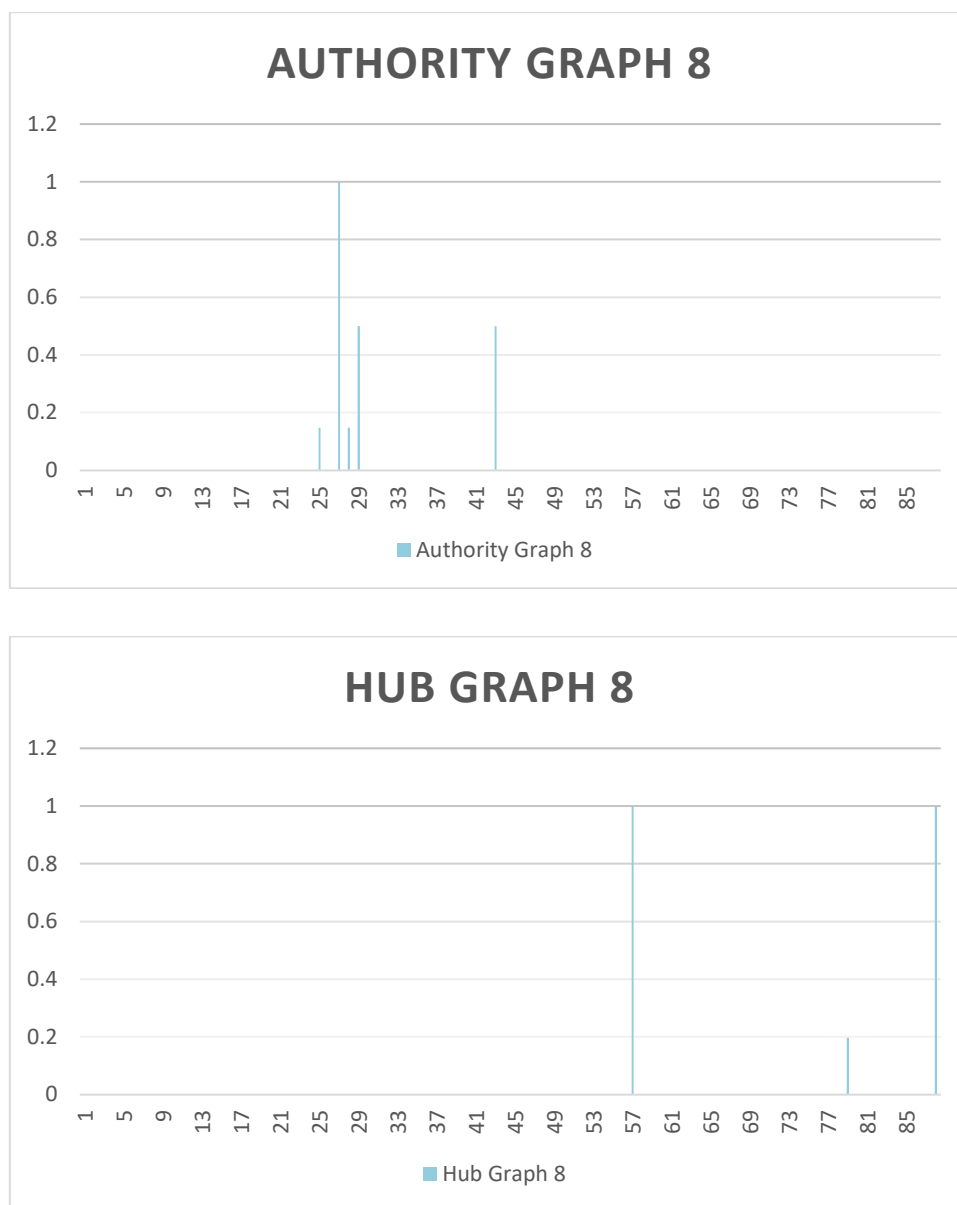
Graph 6 可以說是 Graph 5 的進階版，擁有高目錄值、高權威值節點和高密度的比例，並且數量也比 Graph 5 高出許多，因此判斷可能是系統更加複雜的一個網路平台。

☆ **Graph 7 :**



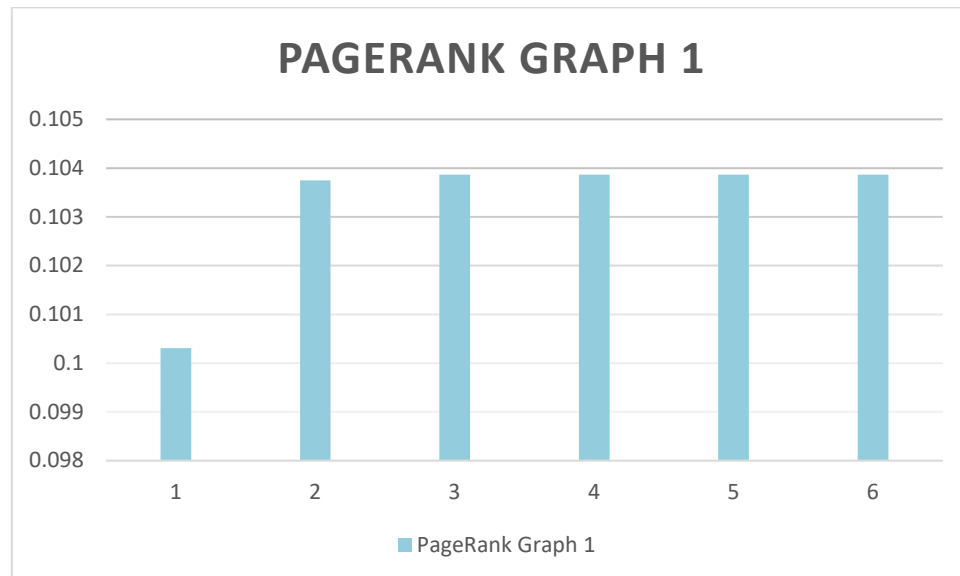
Graph 7 則是利用了 Project 1 的交易資料，總共產生了 9 筆交易紀錄，每筆交易紀錄都列出所購買的物品，以權威值來看的話，如果這個節點是高權威值的話，就代表著這個節點所對應到的商品為較多人購買的，能夠做為市場行銷的重要資料;若以目錄值來看的話，可以看出 9 節點過後的值都是 0，這是因為 9 後面的數字都是代表著商品，商品是不可能指向訂單的，所以通通為 0，目錄值高的訂單代表這筆訂單都是購買到高人氣商品的，可能是賣場的促銷計劃生效或者是特定活動造成某類型物品銷售量一起提升。

✧ **Graph 8 :**

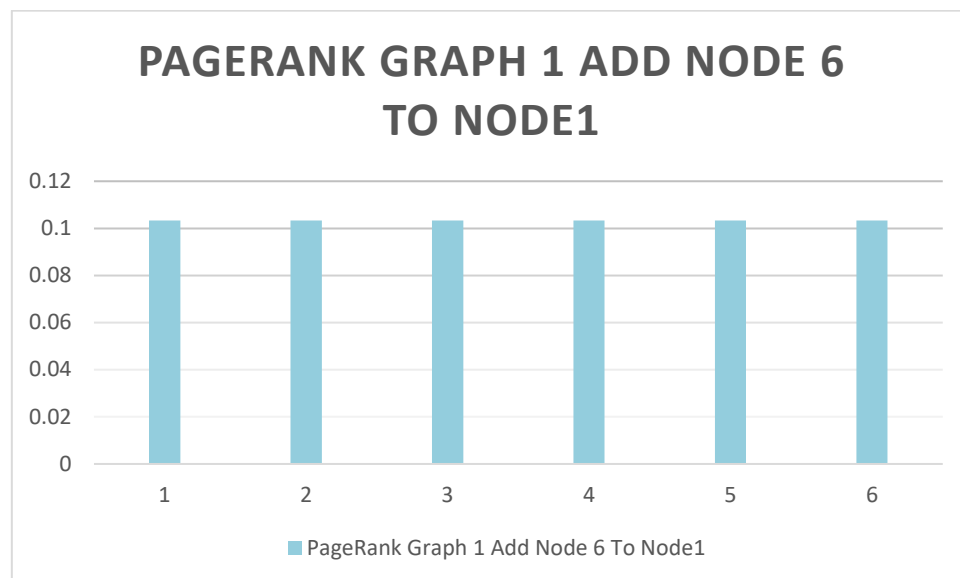


Graph 8 則是將 Project 1 的交易紀錄透過 FP-Growth 產生關聯法則，產生出每個商品同時出現的關聯性，並將它做成與其他 Graph 相同的 txt 文字檔以觀察權威值以及目錄值。以權威值來看，節點 27 是最高是因為它被兩個最好的目錄值節點所指向；以目錄值來看節點 57 以及節點 88 是最高的，因為它們兩個同時指向最高權威的 27 節點。代表著如果商品 27 被購買，商品 57 跟商品 88 被購買的機率相對高出許多。

✧ Graph 1 :

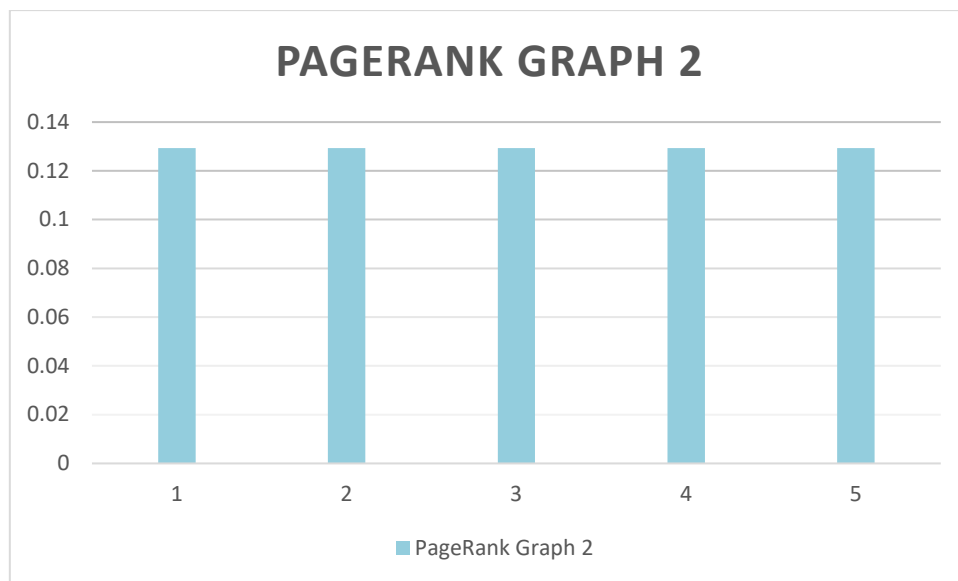


✧ Graph 1 提高 Node 1 值 :

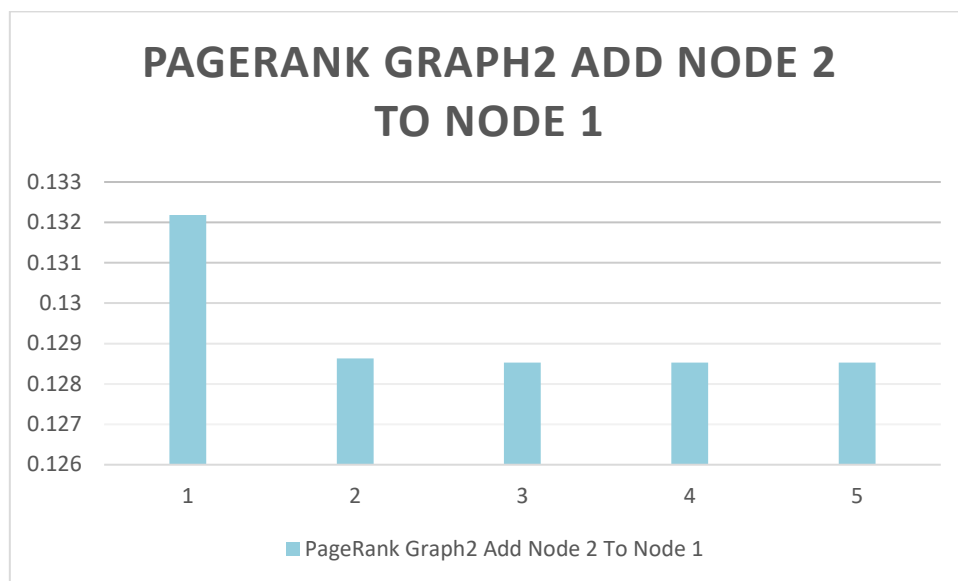


由於 PageRank 的值與連入的節點數量有關，在一開始節點 1 的 PageRank 值為最低，如果想平均所有節點 PageRank 值的話，可以增加一個節點 6 至節點 1 的連結，所有節點的數值就會趨於平緩且一致。

✧ **Graph 2 :**

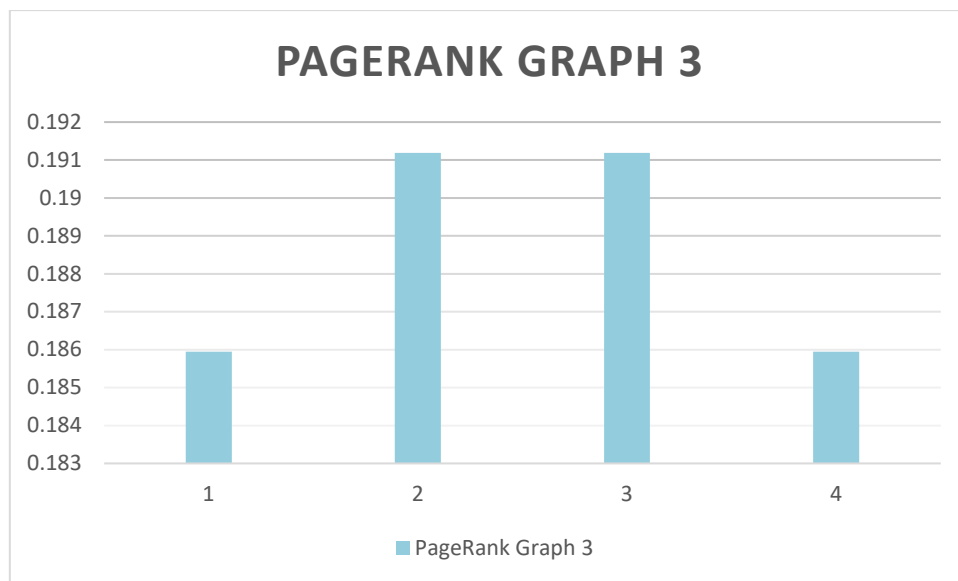


✧ **Graph 2 提高 Node 1 值：**

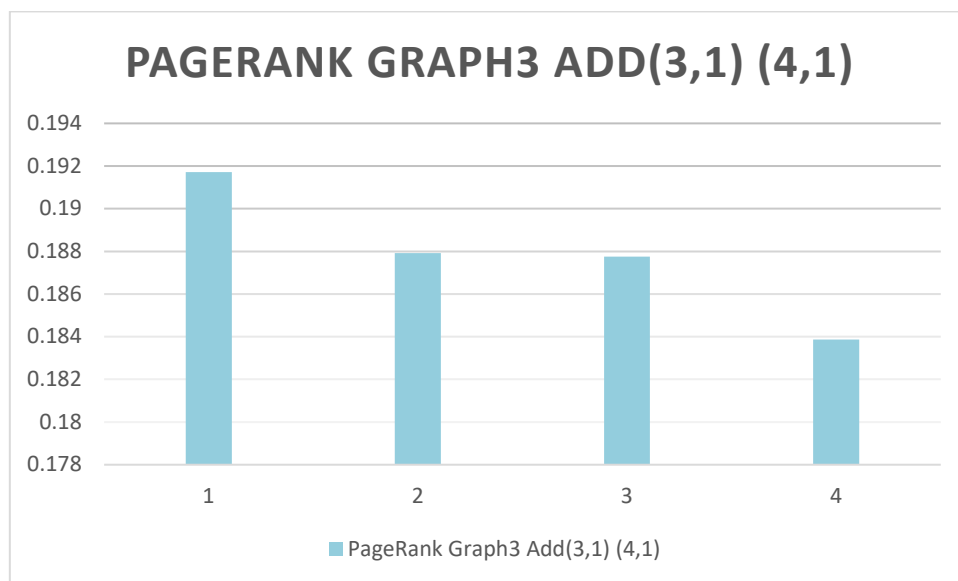


由於 Graph2 連結方式呈現一個 Circle，所以每個節點都各被連接到一次，因此造成圖表上所有的 PageRank 值都為相同，如果想單獨增加節點 1 的 PageRank 值，所使用的方法為增加節點 2 連結到節點 1 的連結關係，這樣就代表節點 1 被連結的次數高於所有其他節點一次。

✧ **Graph 3 :**

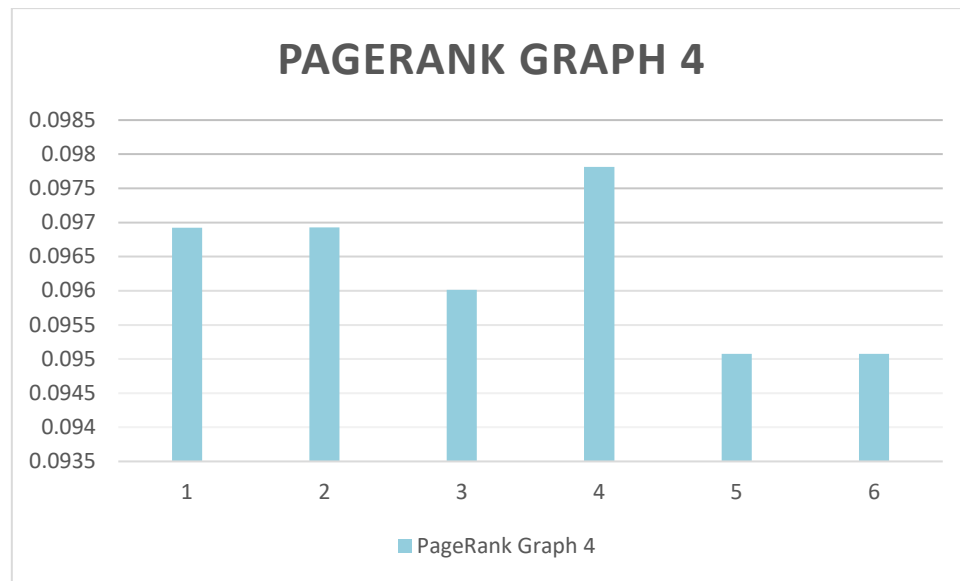


✧ **Graph 3 提高 Node 1 值 :**



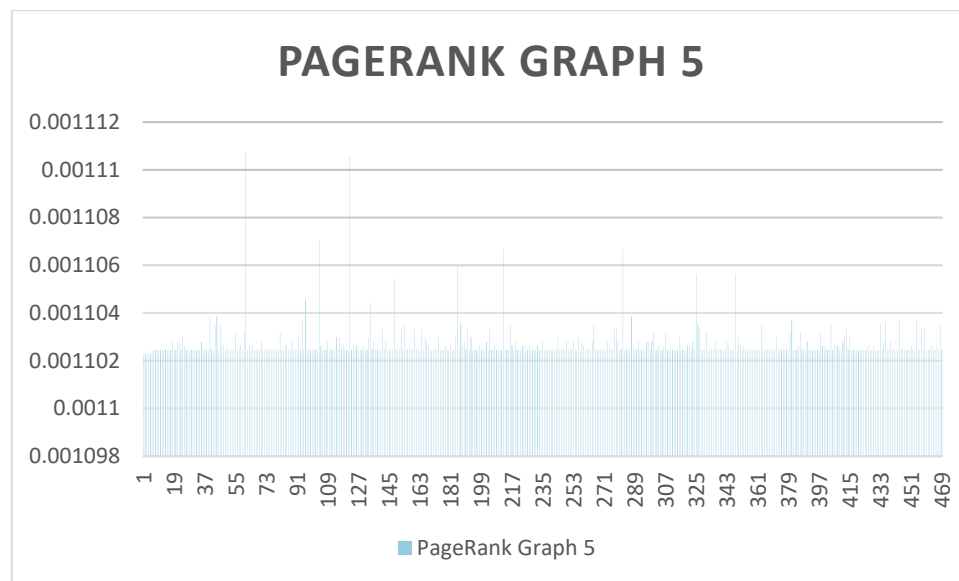
從圖表可以看出節點 2 以及節點 3 的 PageRank 值都是相同並最高，這是由於這兩個節點被連結次數都為兩次，高於節點 1 和 4。若想單獨增加節點 1 的 PageRank 值，透過增加節點 3 連結至節點 1 以及節點 4 連結至節點 1 這兩個連結關係，就能產生節點 1 有最高被連結次數，PageRank 值就能相對提高。

✧ **Graph 4 :**



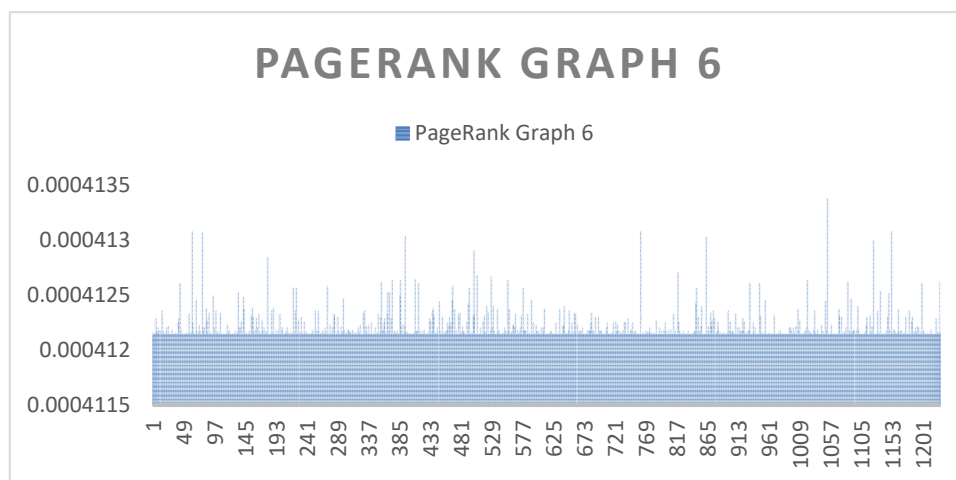
圖表中顯示節點 5 以及節點 6 的值為最低，可以推測出這兩個頁面可能是平常較少人點擊到的，例如平台的註冊或是忘記密碼，屬於使用者不會經常造訪的網頁，但這兩個頁面同時也是非常重要的會員功能，因此相較起來數值並不會低太多。

☆ **Graph 5 :**



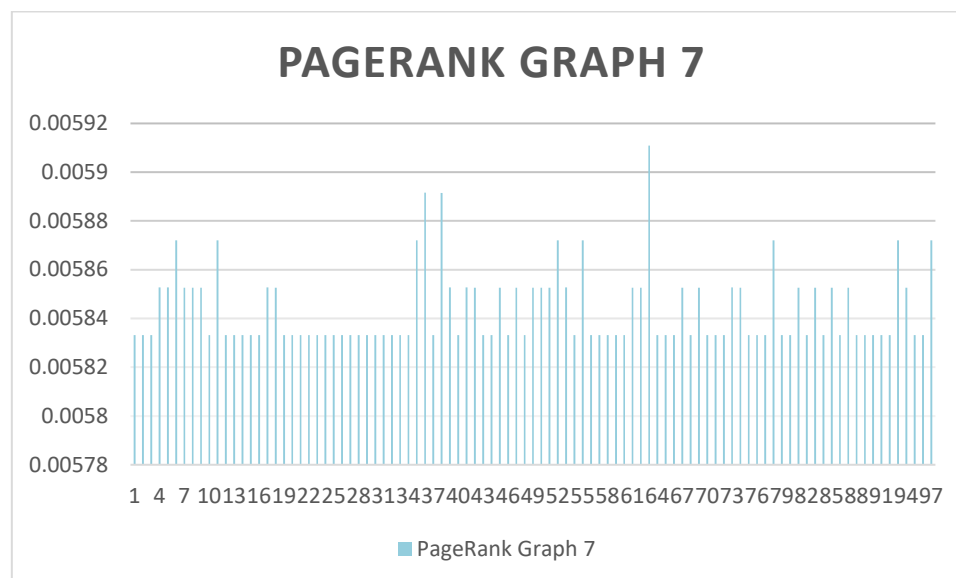
圖表中看出高巔峰 PageRank 值只落在特定幾個節點上，最高值與最低值差異極大，若以網路購物做為推測，當我們要買一個物品，人氣高的品牌或是有透過電商廣告的品牌就是固定那幾個大廠牌，而其他較默默無聞或不常在市場上看到的牌子則較少會被使用者所觸及到。

✧ **Graph 6 :**



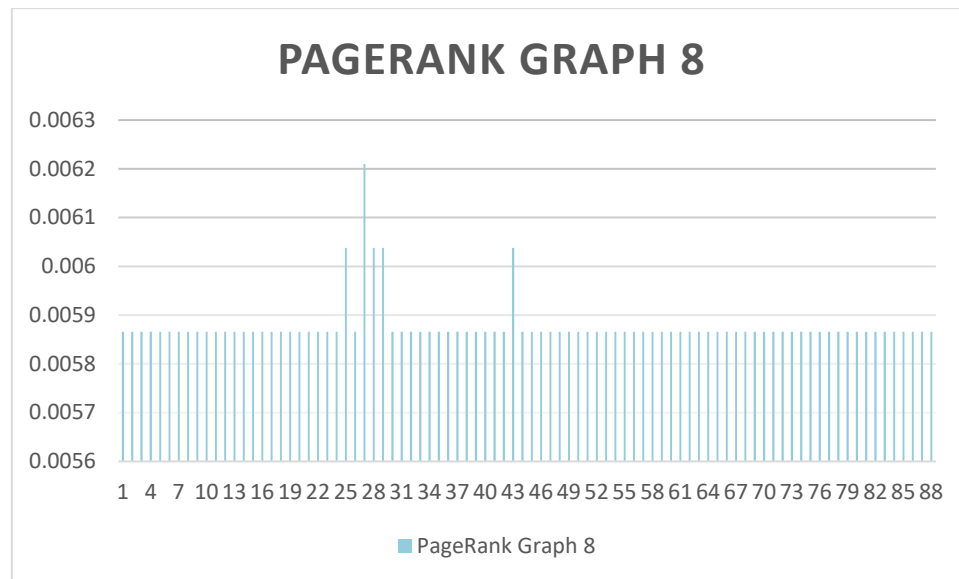
Graph 6 與 Graph 5 的差異並不大，可以從圖中看出 PageRank 高的節點有更密集且數量更多的趨勢，因此推測可能為一個較複雜的網際網絡。

✧ **Graph 7 :**



Graph 7 是 Project 1 的交易資料，透過 PageRank 演算法實作後，可以看出 PageRank 值高代表此商品的熱銷度，從圖表中可以很明顯發現某幾個商品是較多消費者會購買的。

✧ **Graph 8 :**



Graph 8 則是把交易紀錄經過 FP-Growth 後產生關聯法則，數值高的節點代表著關聯法則的 Confidence 可信度較高，可做為市場銷售的重要指標。

C. SimRank 演算法



✧ Graph 1 SimRank :

1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

✧ Graph 2 SimRank :

1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Graph 1 跟 Graph 2 的連結方式並沒有同時兩個節點進入同一個節點，而 Graph 1 為一連串式的串接、Graph 2 為 Circle 型的串接，所以經過 SimRank 演算法過後，無法分析各個節點的關聯性，皆呈現對稱的單位矩陣。

✧ Graph 3 SimRank :

1	0	0.4	0
0	1	0	0.4
0.4	0	1	0
0	0.4	0	1

Graph 3 中出現了兩個節點進入同一節點，剛好滿足 SimRank 分析的特性，以 Graph 3 中的(1,2)(3,2)來說明的話，可以很明顯的在矩陣中看到節點 1 和節點 3 是有關連性的，因為它們兩個同時連接到節點 2，所以相關度會相同。

✧ Graph 4 SimRank：

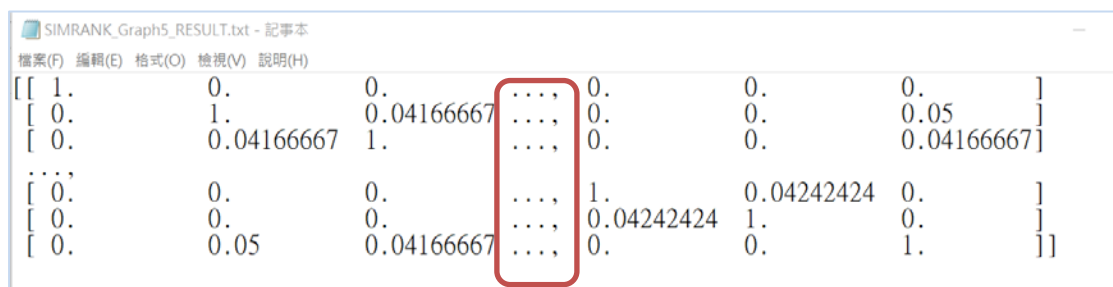
1	0.066667	0.133333	0.266667	0.16	0.133333	0
0.066667	1	0.1	0	0.08	0.1	0.2
0.133333	0.1	1	0	0.08	0.2	0.4
0.266667	0	0	1	0.16	0.4	0
0.16	0.08	0.08	0.535063	1	0.08	0
0.133333	0.1	0.2	0.4	0.08	1	0.4
0	0.2	0.4	0	0	0.4	1

由矩陣中可以看到節點4以及節點6的相關度是在所有數值裡面最高的(0.4)，
可以推定 Graph 4 中節點4跟節點6具有高相關度。

✧ Graph 5 SimRank：

Graph 5 由於資料量過於大，所以產生的矩陣也相對龐大，程式碼無法正確
將結果產生 csv.檔，因此由 txt 的格式儲存。

如下圖所示，中間...的部分由於資料量過於龐大，以此符號省略代表，因此
無圖片顯示。

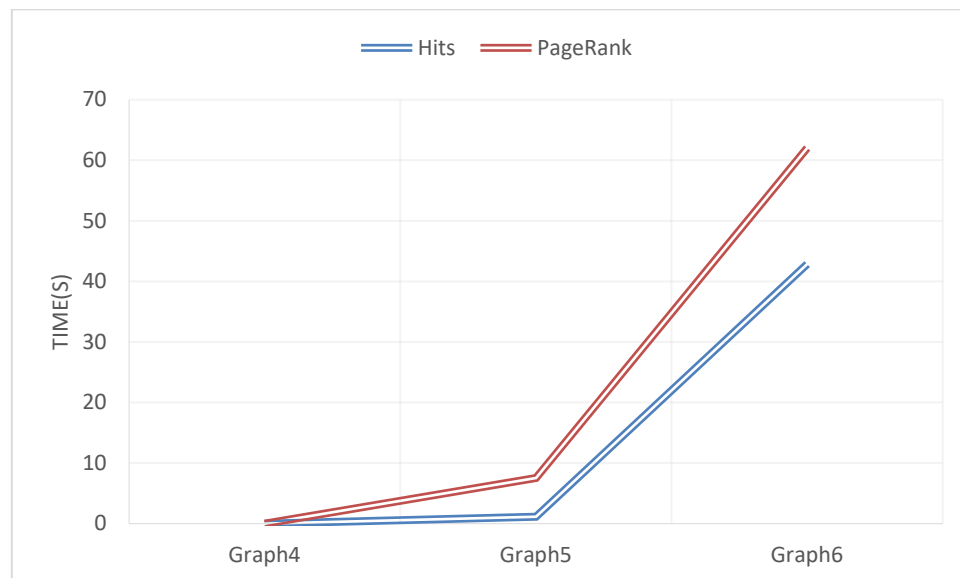


```

SIMRANK_Graph5_RESULT.txt - 記事本
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)
[[ 1.      0.      0.      ..., 0.      0.      0.      ]
 [ 0.      1.      0.04166667 ..., 0.      0.      0.05    ]
 [ 0.      0.04166667 1.      ..., 0.      0.      0.04166667]
 ...,
 [ 0.      0.      0.      ..., 1.      0.04242424 0.      ]
 [ 0.      0.      0.      ..., 0.04242424 1.      0.      ]
 [ 0.      0.05    0.04166667 ..., 0.      0.      1.      ]]
```

四、運算效能分析< Computation performance analysis >

運算效能分析上，由於前 3 個 graph 資料量太少，較難以分析出東西，因此選用其資料量夠多的 Graph4、Graph5 以及 Graph6 來做為測試集，才可以觀測出 Hits 和 PageRank 在運算效能上的差異。由圖表中很明顯地看到，當資料量到達一定數量，Hits 演算的速度會比 PageRank 高上許多，以 Graph 6 來看的話更是差了 20 秒之多，推測是因為 HITS 算法只是對網路中一個子集進行分析，所需要的迭代次數較少，收斂速度會較快，相對減少了時間複雜度。因此若以此 project 的 dataset 為計算，Hits 相對花費較少的執行時間，整體運算效能較好。



五、心得與討論< Thoughts and discussion >

實作這三個演算法之後，發覺所有演算法在實際生活的應用中都是結合傳統的內容分析技術再進行優化，其實沒有一種演算法是完美的，配合應用程式的需求不同，所要分析的東西就會不同，所需要的演算法也會不同，在某些查詢下，有些演算法的結果可能很好，在另外的查詢下，結果也可能很差。

其中比起 PageRank，HITS 對我來說相對有參考價值，不僅是因為在執行此 project 時，HITS 的執行時間較快、運算效能較佳外，利用 HITS 能夠得到較多對於節點之間的資訊，如果再結合用於分析網頁的之間的存在的關係的 SimRank，對於鏈結分析將會更加完整。

PageRank 的分析方式有時會遺漏掉重要的資訊。因被連結較多的網頁如同被引用較多，代表該網頁具有較高的重要性，網頁排名值亦較高，並且若是被重要性較高的網頁所連結，則被連結網頁的重要性與排名值也相對較高。但是這因素在某些網頁之間的關聯性很高，卻不會互相連結時，往往會遺漏當中可以分析的重要指標和資訊，像是 7-11 的網頁就不會有全家的連結，因為他們彼此是競爭對手，但是使用者下的搜尋關鍵字若是便利商店，這些網頁也都會是使用者需要的，因此不能夠明顯的反映出使用者的需求，找出來的結果也會因為某一個商店有比較多的網頁或其他網頁連結到，而將所有與這家商店相關的網頁排在前面。因此探討其他在此領域專業的研究後，發現有新提出一個方法：Virtual Link。

此方式首先必須先找出熱門的搜尋結果，建構為 Hot Set，再紀錄使用者對這些搜尋結果上的操作動作，如點選的次數或瀏覽的時間，用來建構 Virtual Link，再利用類似 HITS 的評分方式，來更新相關網頁的分數。因為這些連結是搜尋結果的連結，並不是真正的網頁超連結，所以稱為 Virtual，此方法比起只用 HITS 的系統來說，整體的搜尋結果及衡量標準都有較好的表現，由另一個層面來看，得到的搜尋結果也比較符合使用者的需求。

近年來大家似乎愈來愈依賴搜尋引擎做為進入各大網站的橋樑，各網站在搜尋引擎上的搜尋排名高低，似乎也決定了此網站的人氣，因此，我認為，如何提高自己在搜尋引擎上的排名，也成了各大網站業者與搜尋引擎業者間互相鬥智的一點。然而，在現今日益豐富的網路資源當中，除了搜尋結果的排名這個課題之外，還有很多值得利用的資源，平時留意日常生活中的細微線索，若拿來當作研究主題細細鑽研及品嚐，都將會有意想不到的收穫