Data Mining－Project3

學生：吳汶峻

學號：Q36071229

系級：電通甲

程式與環境：Python3/Windows10

一、 演算法介紹

- **Hits：**

  與 Page-Rank 幾乎同時期被提出來的，在 HITS 算法中，每個葉面或網頁被給予兩種屬性：Hub 及 Authority。其中的 Hub，指的是那些包含很多指向 authority 頁面所連接的網頁；Authority 指有包含實質性內容的網頁。Hits 演算法的目的是當有人查詢時，希望能給此人高 Authority 的頁面。而我們的報告以節點與邊的方式代表網頁的連接方式。

- **PageRank：**

  是 Google「評價某種網頁重要性」的一種方式。概念為：若 B 網頁有連接 A 網頁的連結，表示 B 認為 A 有連結的價值，因此 A 是一個「重要」的網頁。PageRank 演算法表達了一個網頁的導入連結重要性，因此一般說來，PageRank 是由一個網站的導入連結的數量及這些導入連結的重要性所共同訂定的。

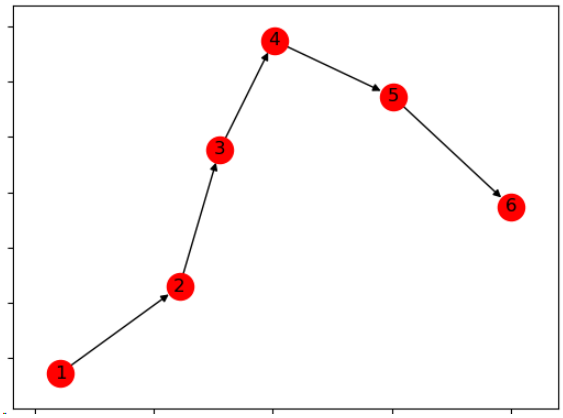
- **SimRank：**

  思想為若兩個用戶相似，則與這兩個用戶相關的物品也會相似；反之，若兩個物品相似，則與這兩個物品相關聯的用戶也會相似。可以利用此種演算法找出關聯性高的用戶中，他們所關心的網頁內容也會有很高的關聯性。

二、 實作結果與討論

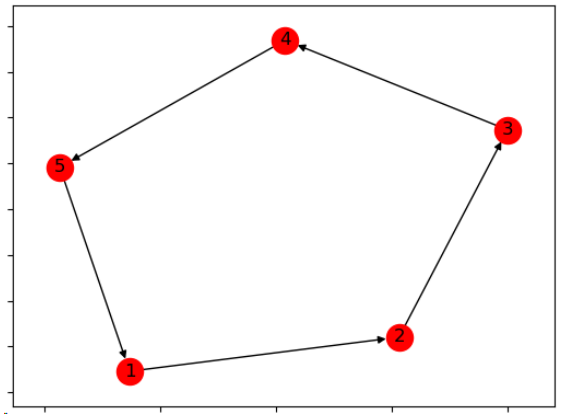
首先，我先利用程式讀檔去畫出 6 個文件檔(graph1~6)的圖形，接著對每一

張圖來做分析：



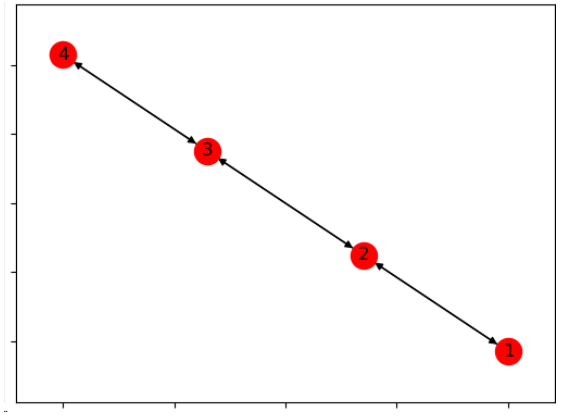
<Graph1>：

為一個單指向的有向圖，Node1 沒有被指向，Node6 沒有指向其他 Node



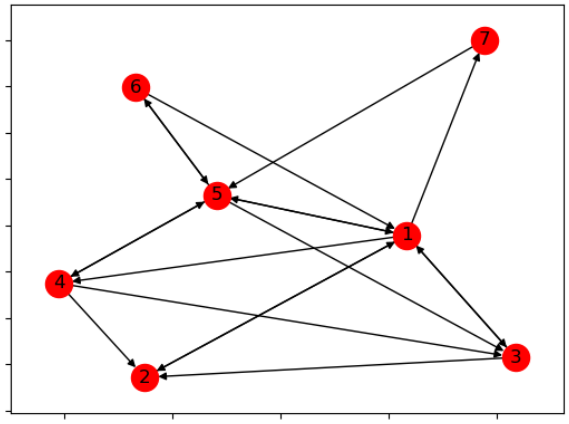
<Graph2>：

為一個封閉圓圈，形成一個以逆時針方向指向的圖



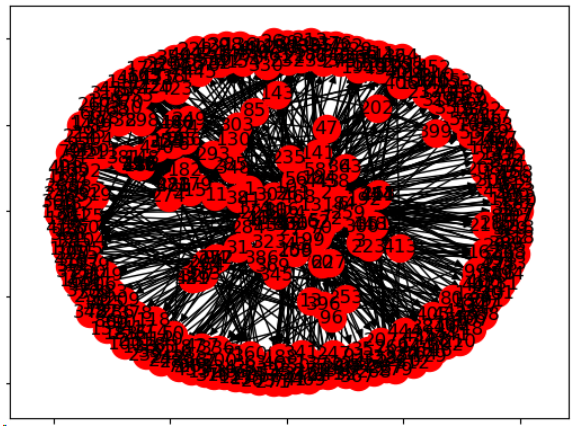
<Graph3>：

為一個互相指向的圖，但其中 Node1 與 Node4 互相沒有指向對方



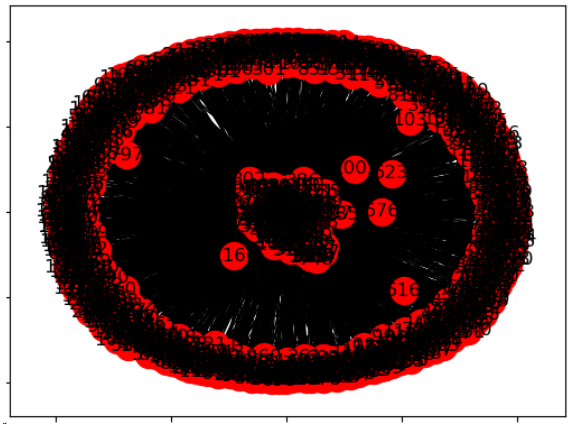
<Graph4>：

圖形相較前三張更為複雜，截點數也變多了



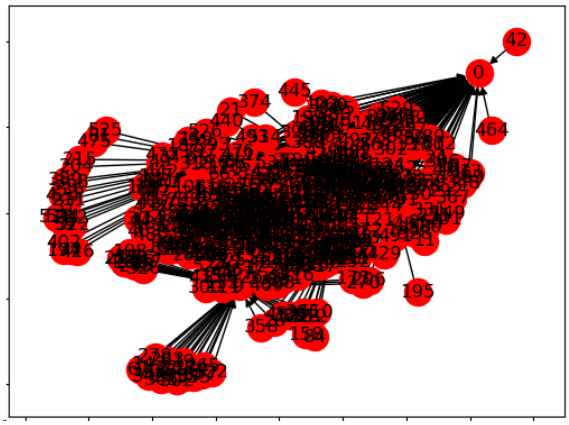
<Graph5>：

相當複雜的一張圖，但似乎也有特定的規律



<Graph6>：

節數量相當的多



<Graph_trdata>：

為利用 IBM-data-generator 產生的 data，節點數也是相當多

由這幾張圖形可以知道，我們接下來的實驗都是以有向圖來做處理。

✧ Hits：

指向同一個節點的 hub 值加總計算為其節點的 authority 值，將其節點指向 authority 值加總後就是其節點的 hub 值。而在每次迭代中都會將他們正規化，讓它們保持在 0~1 之間，重複這些步驟直到其值收斂為止。以下為 Graph1~Graph6 圖形算出來的 authority 與 hub 值：

Graph1：

```
Hub: {1: 0.2, 2: 0.2, 3: 0.2, 4: 0.2, 5: 0.2, 6: 0.0}
Authority {1: 0.0, 2: 0.2, 3: 0.2, 4: 0.2, 5: 0.2, 6: 0.2}
```

Graph2：

```
Hub: {1: 0.2, 2: 0.2, 3: 0.2, 4: 0.2, 5: 0.2}
Authority {1: 0.2, 2: 0.2, 3: 0.2, 4: 0.2, 5: 0.2}
```

Graph3：

```
Hub: {1: 0.1909830056647784, 2: 0.3090169943352216, 3: 0.3090169943352216, 4: 0.1909830056647784}
Authority {1: 0.190983005521049, 2: 0.309016994478951, 3: 0.309016994478951, 4: 0.190983005521049}
```

Graph4：

```
Hub: {1: 0.2754531765654345, 2: 0.047762306376935765, 3: 0.10868323971440082, 4: 0.1986595564890898, 5: 0.1837345993425369, 7: 0.06897240756733328, 6: 0.11673471394426906}
Authority {1: 0.1394838930854497, 2: 0.1779120314091907, 3: 0.20082320536937326, 4: 0.1401777533243232, 5: 0.20142536348733986, 7: 0.08408849143863972, 6: 0.05608926188568348}
```

Graph5(部分截圖)：

```
 186: 0.002619092772817122, 90: 3.837498662112 68e-24, 18
 9: 2.6447015505976335e-20, 158: 2.6447015505976335e-20,
 5e-20, 322: 2.6447015505976335e-20, 324: 2.6447015505976
 759635507e-13, 301: 1.7465227541707573e-44, 221: 1.8764
 0300873350794e-23, 420: 7.0380445681838 36e-24, 222: 2.99
 09886207e-21, 146: 9.05920309886207e-21, 161: 9.0592030
 207e-21, 378: 9.05920309886207e-21, 406: 1.7052978224635
```

Graph6(部分截圖)：

```
 230: 9.05920309886207e-21, 289: 9.05920309886207e-21,
21: 3.794916767801804e-23, 224: 0.004889734627465237,
 : 1.5117331927128425e-20, 259: 1.5117331927128425e-20,
1: 0.0014700050393062268, 49: 0.0014700050393062268, 51
2: 0.0014700050393062268, 175: 0.0014700050393062268,
10: 0.0014700050393062268, 313: 0.0014700050393062268
```

Graph_trdata(部分截圖)：

```
1194: 0.002113775670797266, 195: 0.0003117317450066 6746,
 201: 0.0002915641015730325, 202: 0.0008894742267064926,
208: 0.0023603201983950182, 209: 0.0021303968684865376,
 215: 0.0008894742267064926, 216: 0.0013772004804134723,
 222: 0.0016623066769246662, 223: 0.0025137671856327467,
29: 0.0012006743904246337, 230: 0.0026696810864112828, 2
236: 0.0008894742267064926, 237: 0.0020491200346768774,
243: 0.0021107608063414287, 244: 0.002551780903631159, 2
0: 0.0025137671856327467, 251: 0.0024445347304276 28, 252
```

《討論》：在 Graph1 中可以發現，Node1 只有指向其他節點，並沒有其他節點指向他，authority 值為 0。相對的，Node6 只有被指向，並沒有去指向其他節點，因此 hub 值為 0；在 Graph2 中可以發現，因為它是個封閉迴圈，因此每個 Node 的 authority 及 hub 皆相同；Graph3 及 Graph4 可以發現，authority 高的 Node 所指向別人的節點多，hub 高的

Node 被指向的 Node 很多。

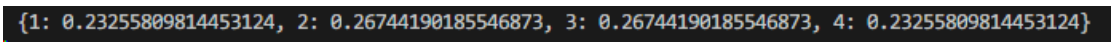
✧ PageRank：

將 damp factor 設為 0.15，在每次進行迭代時，都將每個節點的 PageRank 值重新更新為連接進此結點的結點之 PageRank / vertex，接著一直重覆此步驟直到收斂為止。以下為 Graph1~Graph6 圖形算出來的 PageRank 之值：

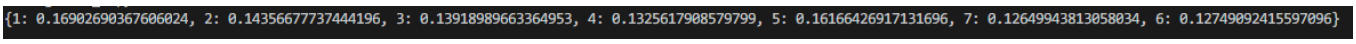
Graph1：

Graph2：

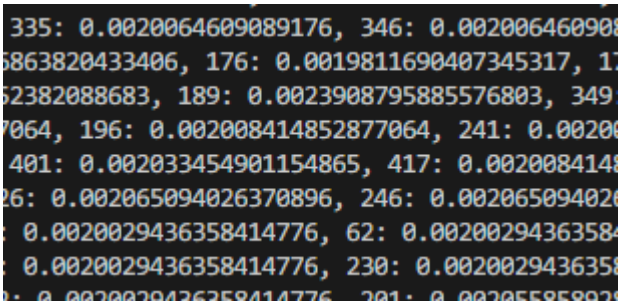
Graph3：



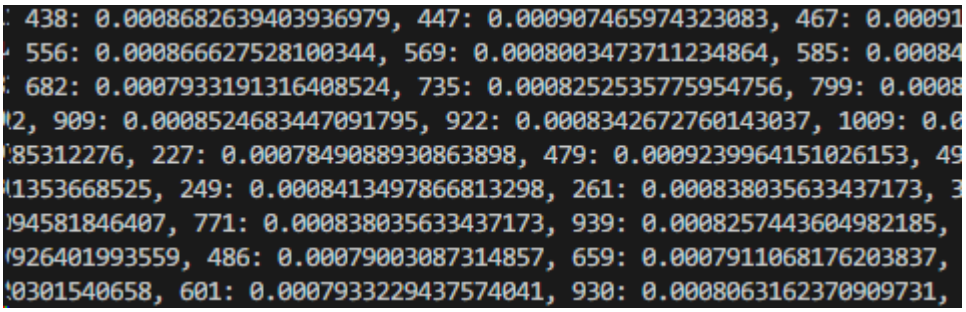
Graph4：



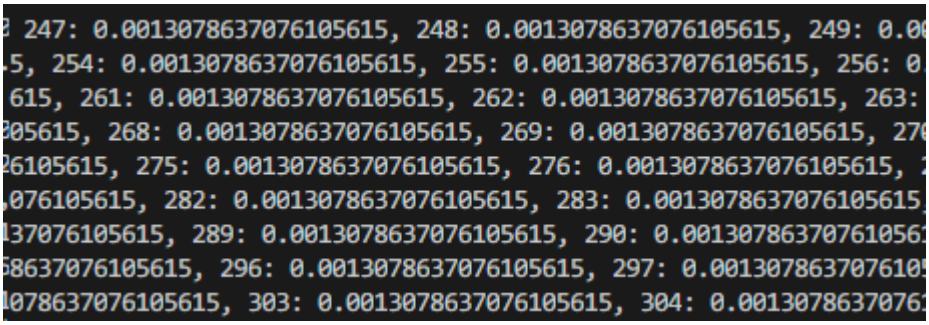
Graph5(部分截圖)：



Graph6(部分截圖)：

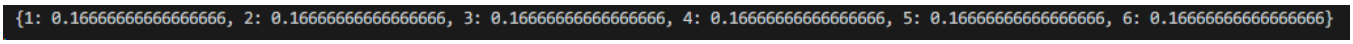


Graph_trdata(部分截圖)：



《討論》：我發現到，經過多次執行程式，在時間上來看，PageRank 比 Hits 更為有效率。此外，若是將 damp factor 值進行調整的話，所算出來的 damp factor 會不一樣。而以 Graph1 為例，在 damp factor 值為 0 時，會發現每個 Node 的 value 皆為 0.1666，在此可以猜測，當 damp factor 的值越小時，會讓其計算更為簡單。以下為 Graph1 的 damp factor=0 及 0.3 時的圖：

Damp factor = 0：

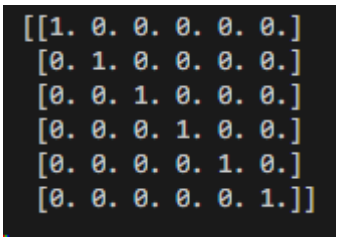


Damp factor = 0.3：

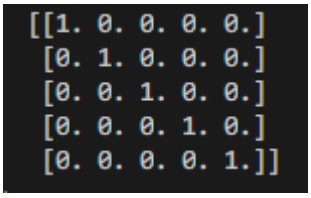
✧ SimRank：

先計算出每個節點的 In-link 數量，因為要將每輪的正規化到 0~1 之間，根據每個節點相關的 In-link 來進行每次的更新。其中 C 為一阻尼系數。接著一直重複此步驟直到收斂為止。以下為 Graph1~Graph5 圖形算出來的 SimRank 之值：

Graph1：



Graph2：



Graph3：

```
[[1.         0.         0.33333302 0.        ]
 [0.         1.         0.         0.33333302]
 [0.33333302 0.         1.         0.        ]
 [0.         0.33333302 0.         1.        ]]
```

Graph4：

```
[[1.         0.11292512 0.1065411  0.1149133  0.09817619 0.16901194
  0.06081467]
 [0.11292512 1.         0.16057946 0.13114497 0.15502011 0.058726
  0.20356393]
 [0.1065411  0.16057946 1.         0.20184115 0.14331203 0.20151273
  0.20216958]
 [0.1149133  0.13114497 0.20184115 1.         0.11097399 0.27453489
  0.27453489]
 [0.09817619 0.15502011 0.14331203 0.11097399 1.         0.05386876
  0.16807922]
 [0.16901194 0.058726   0.20151273 0.27453489 0.05386876 1.
  0.04906978]
 [0.06081467 0.20356393 0.20216958 0.27453489 0.16807922 0.04906978
  1.        ]]
```

Graph5：

```
[[1. 0. 0. ... 0. 0. 0.]
 [0. 1. 0. ... 0. 0. 0.]
 [0. 0. 1. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 1. 0. 0.]
 [0. 0. 0. ... 0. 1. 0.]
 [0. 0. 0. ... 0. 0. 1.]]
```

《討論》：由於 SimRank 為計算兩點之間的相似度，所以若和本身去做 SimRank 其值為 1，它的比較方式和 Hits 及 PageRank 計算出來的值較為不同。在 Graph1 和 Graph2 中，可以看到 Node 自己跟自己為 1 之外其他都為 0，這是因為每個點都是單向連接的，並沒有相似節點。而在效能方面，我認為它不能與 Hits 與 PageRank 相比較，因為算出來的東西不太一樣。

三、 相關討論與說明

✧ **Discussion (what you learned from this project and your comments about this project)：**

在這次的報告中，讓我了解到原來網頁裡還有所謂的權威性網頁，然而這些權威性網頁其實背後就只是被其次重要的網頁或多個比較不重要的網頁所關聯而成的。在這三個演算法中，我覺得 PageRank 是一個

傑出有效率的方法，因為 PageRank 是很直接地把每個網頁的價值依據用戶搜索或瀏覽到的下一個網頁的機率加起來，變成下一個網頁的價值分數。因為我對網頁怎麼流通與背後的意義蠻有興趣的，希望在未來熟悉這三種的演算法後，可以學以致用，然後再自己上網搜尋更多更進階的演算法及技巧！

✧ **More limitations about link analysis algorithms：**

我覺得 Time cost 問題很嚴重，若是相當複雜的網路，所花費的時間一定不可小覷，因此在計算時要確保網路是不會突然受到干擾或入侵的，而且在計算大型網路的 Value 時，若已經經過計算的網頁此時又受到很多次點擊，就會發生失真、不準確的現象。

✧ **Can link analysis algorithms really find the “important” pages from Web?**

根據 link analysis 我覺得只能找到部分較重要的網頁，有些網頁重要的網頁會因為其他網頁並沒有想要連結它而使它的重要度下降，到最後用 link analysis 會找不到。

✧ **What are practical issues when implement these algorithms in a real Web?**

先前執行的演算法時間只有單單幾個節點就要花很久的時間，而由於今日的網頁數目相當的多，若要依照前面所說的演算法去跑的話 Time Cost 其實是相當巨大的，而且現今網頁一個連接一個，連結網相當複雜，若是執著在重要度低的網站而疏忽相對重要的網站，其實是十分不切實際的。

✧ **Any new idea about the link analysis algorithm?**

我覺得可以在每一個邊設置權重，Node 之間比較重要的連接可以設較高的權重，這樣在算 Value 時準確率會高不少。

✧ **What is the effect of “C” parameter in SimRank?**

SimRank 中的 C 為阻尼係數，若是越大，最後每個節點的 SimRank 值就會越大，這相當於是在計算時的 Value 權重，而且在計算中它位於分子，因此當 C 值越大時，每個節點之間的分數就會被計算的越大。