資料探勘 作業三

資訊所 P76074509 辜玉雯

- 目的：

此次作業是要實作出鏈結分析中重要的三個演算法 HITS、PageRank 以及 SimRank，這些演算法主要都是用在評估網頁重要性。HITS 與 PageRank 藉由網頁連結關係找出重要的網頁，而 SimRank 則計算網頁間相似度。除了實作出演算法，還會透過實驗分析三者的效能，並討論這些演算法的限制，以及應用在真實網站上的合適性。

- HITS 演算法：

在 HITS 演算法中，最主要需要計算的就是 authority 與 hub，authority 指的是別的網頁指向它的次數；而 hub 指的是它指向別的網頁的次數，藉此來評估網頁間的關係與重要程度。因此這兩個數值可以透過下面的公式來計算。

$$\mathrm{auth}(p) = \sum_{i=1}^{n} \mathrm{hub}(i) \qquad \mathrm{hub}(p) = \sum_{i=1}^{n} \mathrm{auth}(i)$$

在算出兩個數值後，必須要再做 normalization，因為這兩個數值都是經過迭代後不停相加的結果，如果不做正規化，那數值會無止盡變大，並沒辦法終止演算法，因此會再利用下面的公式去做正規化。以 authority 為例，先將每個 node 的 authority 值相加，將加總後的結果平方再開根號，所得到的值當分母，而每個 node 各自的 authority 當分子，最後求得正規化後的 authority。

$$\sum hub(x)^2 = \sum auth(x)^2 = 1$$

最後得出 authority 與 hub，根據每次迭代的變化量與 threshold 做比較，去判斷目前的計算是否趨於收斂，如果收斂到一定程度，那演算法就可以終止。

- PageRank 演算法：

在 PageRank 的演算法中考量的是，網頁之間連結的數量與品質，藉此來評斷網頁的重要性。如果某個網頁被越多網頁指向連結，那它的重要性也就越高，並且如果連結它的網頁僅有指向它這個網頁，則相較於胡亂指向多個其他網頁的網頁，可靠度會更高，因此它的 pagerank 在這種狀況下也會更高。

初始時，每個網頁的 pagerank 值會設為$\frac{1}{n}$，這是為了滿足最終機率值位於 0 到 1 之間的需要。接著同樣是迭代地去進行，根據投票的概念，若某個頁面同時連結到多個網頁，那它傳給每個網頁的 pagrank 值則會變成$\frac{pagerank}{n}$，透過下面的公式持續去計算 pagerank，直到數值處於收斂的狀態，最終得到結果。而最終所有節點的 pagerank 值的總和會等於 1。

$$PR(P_i) = \frac{(d)}{n} + (1-d) \times \sum_{l_{j,i} \in E} PR(P_j) / \text{Outdegree}(P_j)$$

公式之所以出現 d 這個參數，是由於「沒有向外連接的網頁」傳遞出去的 pagerank 會是 0，而這會遞迴地導致指向它的網頁的 pagerank 的計算結果同樣為零，所以這裡假設：這類網頁會連結到集合中所有的網頁 (不管它們是否相關)，使得這類網頁的 pagerank 將被所有網頁均分。

也因此在我的演算法一開始會先去檢查沒有向外連結網頁的頁面，並增加該頁面到其他所有網頁的邊，藉此達成這個假設。而 d 則依照作業要求採用 0.15 來表示使用者停止往下點選，而是隨機開啟新頁面的機率；而 1-d=0.85 則是使用者會繼續往下點選網頁的機率。

- SimRank 演算法：

SimRank 這個演算法是一個衡量任意兩個對象間相似程度的方法，它的想法是如果兩個網頁被同樣的網頁連結，也就是它們有相似的入鄰邊，那麼可以說這兩個網頁也可能是相似的，所以才會被同一個網頁連結到。

這個演算法的初始狀態是當 a=b 時，直接設定 simrank 值為 1，因為我們假設每個結點與它本身最相似。而當連結到 a 的網頁數為 0，或連結到 b 的網頁數為 0，表示我們無法比較這兩個網頁是否有同樣的入鄰邊，因此此時的 simrank 值則設為 0。其餘的狀況，我們則用下面這個公式去做計算。

$$s(a,b) = \frac{C}{|\mathcal{I}(a)|\,|\mathcal{I}(b)|} \sum_{i=1}^{|\mathcal{I}(a)|} \sum_{j=1}^{|\mathcal{I}(b)|} s(\mathcal{I}_i(a), \mathcal{I}_j(b))$$

透過迭代的方式去計算連結到 a、b 的網頁，他們的相似度是多少，來加總計算最後得到 a 與 b 本身的相似度。

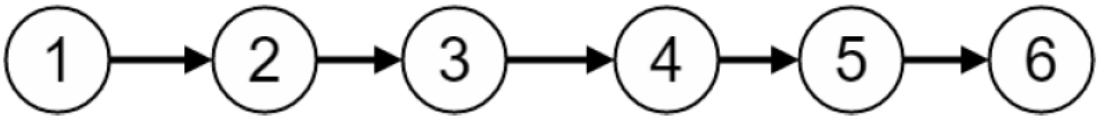
在 SimRank 演算法中，因為在節點與邊數量較大的情況下，SimRank 的時間複雜度非常高，所以我同樣設定一個 threshold 來判斷迭代是否結束。

- 環境：

實作語言	Python
CPU	intel i5-6400 2.7GHz
Memory	16G
OS	Windows 10 x64

- 實驗：

Graph_1：



◆ HITS

	Node1	Node2	Node3	Node4	Node5	Node6
Authority	0	0.4472	0.4472	0.4472	0.4472	0.4472
Hub	0.4472	0.4472	0.4472	0.4472	0.4472	0

由於 Graph_1 是一條單向的路徑，因此可以看到 node1 因為完全沒有被其他節點指向，所以 authority 值為 0，其他節點都被一個節點指向，所以經過迭代後，剩餘節點的值都為 0.4472。

同樣的，node6 因為完全沒有指向其他節點，所以 hub 值為 0，其他節點都有指向一個其他節點，所以經過迭代後，剩餘節點的值都為 0.4472。

◆ PageRank

	Node1	Node2	Node3	Node4	Node5	Node6
Pagerank	0.0607	0.1123	0.1562	0.1935	0.2252	0.2521

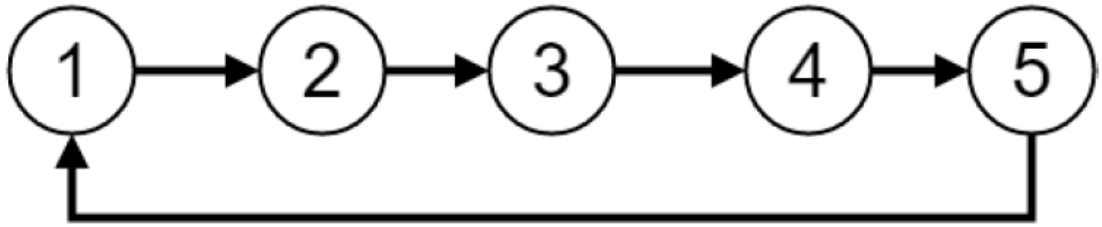
根據 PageRank 計算方式，前面的 node 會將本身的 PageRank 傳遞給後面的 Node，因此會產生分數遞增的現象。所以從結果就可以看到由 node1 到 node6 他們的 pagerank 確實逐漸上升。

◆ SimRank

	Node1	Node2	Node3	Node4	Node5	Node6
Node1	1	0	0	0	0	0
Node2	0	1	0	0	0	0
Node3	0	0	1	0	0	0
Node4	0	0	0	1	0	0
Node5	0	0	0	0	1	0
Node6	0	0	0	0	0	1

由於圖一當中相異的兩個點間都沒有被同一個節點指向，因此只有在 a=b 時，s(a,b)=1，則其他任兩點間的 SimRank 值皆為 0。

Graph_2：



◆ HITS

	Node1	Node2	Node3	Node4	Node5
Authority	0.4472	0.4472	0.4472	0.4472	0.4472
Hub	0.4472	0.4472	0.4472	0.4472	0.4472

由於 Graph_2 是一條 cycle 的路徑，因此所有 node 都被相異的一個節點指向，所以 authority 值都等於 0.4472。同樣的，所有 node 指向相異的一個節點，所以 hub 值都等於 0.4472。

◆ PageRank

	Node1	Node2	Node3	Node4	Node5
PageRank	0.2	0.2	0.2	0.2	0.2

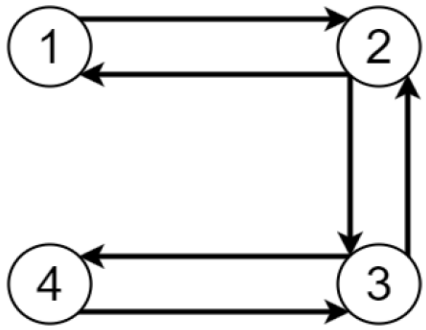
這張圖根據 PageRank 計算方式，所有的 node 會接收上個節點所有的 pagerank 值，因此經過 cycle，所有節點的值都為 0.2。

◆ SimRank

	Node1	Node2	Node3	Node4	Node5
Node1	1	0	0	0	0
Node2	0	1	0	0	0
Node3	0	0	1	0	0
Node4	0	0	0	1	0
Node5	0	0	0	0	1

圖二與圖一類似，當中相異的兩個點間都沒有被同一個節點指向，因此只有在 a=b 時，s(a,b)=1，則其他任兩點間的 SimRank 值皆為 0。

Graph_3：



◆ HITS

	Node1	Node2	Node3	Node4
Authority	0.3717	0.6015	0.6015	0.3717
Hub	0.3717	0.6015	0.6015	0.3717

從圖三可以看出 node2 與 node3 被兩個節點指向，同時也指向兩個節點；而 node1 與 node4 只被一個節點指向，也只指向一個節點，所以 node1、node4 的 authority 與 hub 都比 node2、node3 低，且兩者數值相同。

◆ PageRank

	Node1	Node2	Node3	Node4
PageRank	0.1754	0.3246	0.3246	0.1754

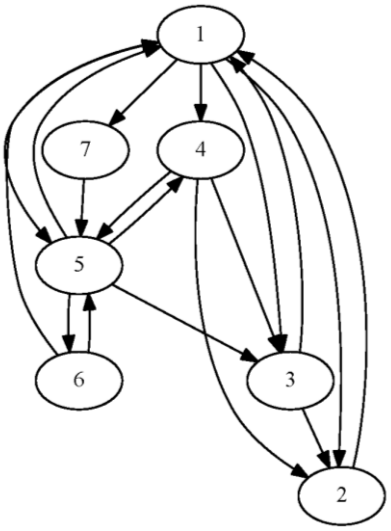
由於指向 node1 與 node4 的節點，都是同時指向兩個節點的情況，因此他們的 pagerank 值會被分散，因此 node1 與 node4 都為 0.1754 並且低於 node2 與 node3。

◆ SimRank

	Node1	Node2	Node3	Node4
Node1	1	0	0.7391	0
Node2	0	1	0	0.7391
Node3	0.7391	0	1	0
Node4	0	0.7391	0	1

由於 node1 與 node3 同時被 node2 指向，因此兩者是相似的，並且計算出相似度為 0.7391；node2 與 node4 同時被 node3 指向，因此兩者是相似的，並且計算出相似度為 0.7391。其他 node 之前則沒有被同時指向的關係，因此 simrank 皆為 0。

Graph_4：



◆ HITS

	Node1	Node2	Node3	Node4	Node5	Node6	Node7
Authority	0.3467	0.4422	0.4991	0.3484	0.5006	0.1394	0.2090
Hub	0.6464	0.1121	0.2551	0.4662	0.4312	0.2740	0.1619

從結果可以發現雖然 node1 與 node5 都有 4 個節點指向，但 node5 的 authority 值較高，這是由於指向 node5 的節點 hub 值相對較高。而 node1 hub 值較高的原因可能是它指向較多的節點，且當中有 node4、node5 這些 authority 較高的節點。

◆ PageRank

	Node1	Node2	Node3	Node4	Node5	Node6	Node7
PageRank	0.2803	0.1588	0.1389	0.1082	0.1842	0.0606	0.0691

針對 node1 會發現指向它的節點，都恰好只有一個向外的邊，因此他們的 pagerank 不會分散，因此 node1 獲得的 pagerank 相對較高。而 node5 雖然也有很多指向它的節點，但他們同時都有很多向外邊，因此 node5 並無法獲得完整的 pagerank，因此數值相對較低。

◆ SimRank[程式結果是按照 txt 節點出現的順序排序，並非由大到小]

	Node1	Node2	Node3	Node4	Node5	Node7	Node6
Node1	1	0.4461	0.4351	0.4388	0.4243	0.3826	0.4951
Node2	0.4461	1	0.4877	0.4531	0.4944	0.5309	0.3752
Node3	0.4351	0.4877	1	0.5263	0.4727	0.5279	0.5247
Node4	0.4388	0.4531	0.5263	1	0.4275	0.6053	0.6053
Node5	0.4243	0.4944	0.4727	0.4275	1	0.4923	0.3627
Node7	0.3826	0.5309	0.5279	0.6053	0.4923	1	0.3607
Node6	0.4951	0.3752	0.5247	0.6053	0.3627	0.3607	1

圖四連結情況比較複雜，所以所有節點之間相似度都不太一樣，但這當中我們可以發現 node5、node6 以及 node6、node7 之間的相似度是最低的，這是因為他們完全沒有共同的連接來源，因此我們會說這兩兩節點之間是較不相似的。

Graph_5：

◆ HITS

Authority	
Node 61	0.4914
Node 122	0.4826
Node 212	0.2951
Node 104	0.2867
Node 282	0.2548

Hub	
Node 274	0.1919
Node 176	0.1898
Node 412	0.1857
Node 293	0.1776
Node 254	0.1747

由於圖五的節點過多，因此這裡只列出數值最高的前五個。可以看出 authority 與 hub 值前五高的節點完全沒有重複，由此可知他們之間並不存在正比的關係。

◆ PageRank

PageRank	
Node 61	0.0144
Node 122	0.0141
Node 104	0.0103
Node 212	0.0078
Node 282	0.0074

可以發現 pagerank 排名前五高的，與 authority 數值前五高的是同樣的節點，由此可知被指向的次數越多，pagerank 就越高。

◆ SimRank

```
[1.   0.   0.   ... 0.   0.   0.  ]
[0.   1.   0.85 ... 0.   0.   0.  ]
[0.   0.85 1.   ... 0.   0.   0.  ]
...
[0.   0.   0.   ... 1.   0.85 0.85]
[0.   0.   0.   ... 0.85 1.   0.85]
[0.   0.   0.   ... 0.85 0.85 1.  ]
```

由於圖五節點過多，所以這裡無法列出。因此我計算出整個 matrix 的平均值是 0.0747，可以知道這張圖上大多數點與點之間的相似度皆為 0，只有少數有大於 0 的數值。

Graph_6：

◆ HITS

Authority	
Node 761	0.2751
Node 1151	0.2751
Node 62	0.2730
Node 78	0.2717
Node 394	0.2653

Hub	
Node 171	0.1563
Node 857	0.1501
Node 185	0.1492
Node 91	0.1479
Node 79	0.1475

圖六同樣節點與邊相當多，在分數排名前五的都是指向與被指向關係最

複雜的節點，node171 有四十多條指向邊，node761 也被指向四十多次，這也是他們分數最高的原因。但他們分數普遍都不高可能是因為指向關係太分散，導致沒有一個最重要的節點出現。

- PageRank

PageRank	
Node 1052	0.0039
Node 761	0.0031
Node 1151	0.0031
Node 62	0.0031
Node 394	0.0030

可以發現 pagerank 排名前五高的，與 authority 數值前五高的是節點有很高的重複性，可以再次發現被指向的次數越多，pagerank 就越高。

Graph_7：

- HITS

Authority	
Node 85	0.2974
Node 38	0.2888
Node 63	0.2882
Node 7	0.2529
Node 73	0.2338

Hub	
Node C4	0.5861
Node C10	0.3977
Node C5	0.3414
Node C1	0.3343
Node C7	0.3307

圖七是利用 project1 的 dataset 來進行演算法，我取 dataset 當中的一部份來測試。(Cx 代表顧客，數字代表商品)可以發現 authority 較高的都是商品，因為在商品買賣的關係上，只有商品有可能會被指向。而 hub 值較高的都是顧客，因為同樣建立在商品買賣的關係上，只有顧客有可能會指向商品。

◆ PageRank

PageRank	
Node 8	0.0212
Node 38	0.0191
Node 63	0.0187
Node 85	0.0168
Node 36	0.0166

由於 pagerank 與被指向的邊有很大關聯，因此可以看到 pagerank 排名高的都是商品，顧客的 pagerank 值相對都較低。

● 練習：

✓ 增減連結邊，使得前三張圖的 node1 的 authority、hub 以及 pagerank 值上升？

Graph_1：

改變前：

	Authority	Hub	Pagerank
Node1	0	0.4472	0.0607

改變後：

	Authority	Hub	Pagerank
Node1	0.4082	0.7071	0.1035

<sol>加入 2->1 & 1->3

任意加入一條 hub 值非 0 的邊，並指向 node1 就可以使 authority 升高；使得 authority 升高後，再任意加入 node1 指向別的節點的邊，就可以使它 hub 值跟著上升。而 pagerank 跟指向它得邊有很大關聯，所以前面已經加入 2->1，也會使得 pagerank 上升。

Graph_2：

改變前：

	Authority	Hub	Pagerank
Node1	0.4472	0.4472	0.2

改變後：

	Authority	Hub	Pagerank
Node1	0.6015	0.6015	0.2737

<sol>加入 1->4 & 4->1

Node1 本來就有指向與被指向的邊，為了使數值上升，所以再各多加入一條指向與被指向的邊就可以使 authority 與 hub 上升。而 pagerank 也因為被指向數量增加，進而跟著上升。

Graph_3：

改變前：

	Authority	Hub	Pagerank
Node1	0.3717	0.3717	0.1754

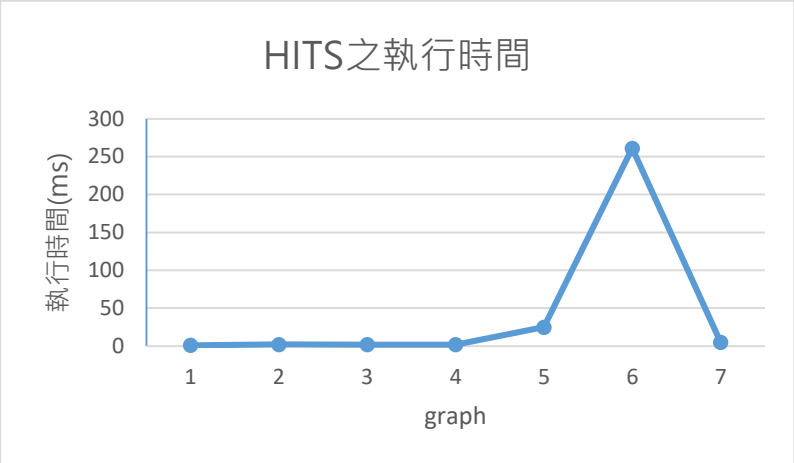
改變後：

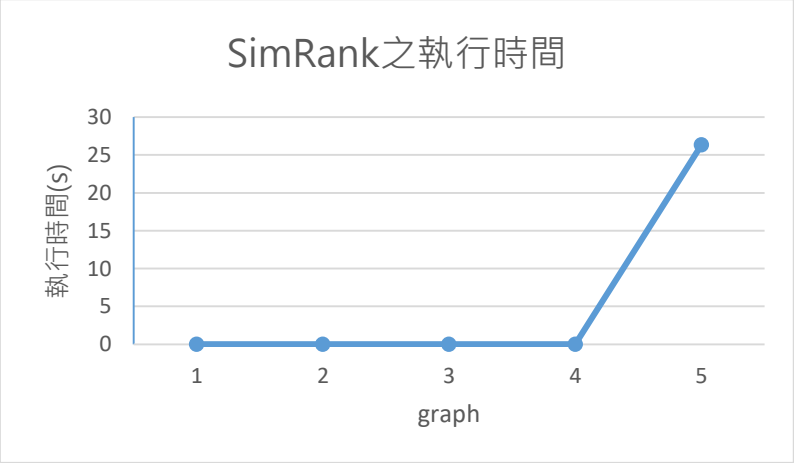
	Authority	Hub	Pagerank
Node1	0.5227	0.5227	0.2459

<sol>加入 1->3 & 3->1

Node1 因為只有一條指向與被指向的邊，所以數值會比其他節點來的低，為了使數值上升，所以跟前一張圖做法相同，再各多加入一條指向與被指向的邊就可以使 authority 與 hub 上升。而 pagerank 也因為被指向數量增加，進而跟著上升。

- 討論：
  - 資料量大小對三種演算法執行時間的影響



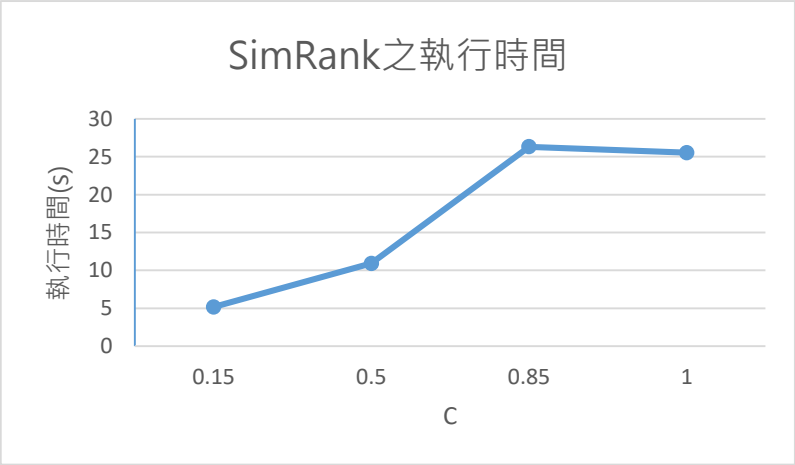


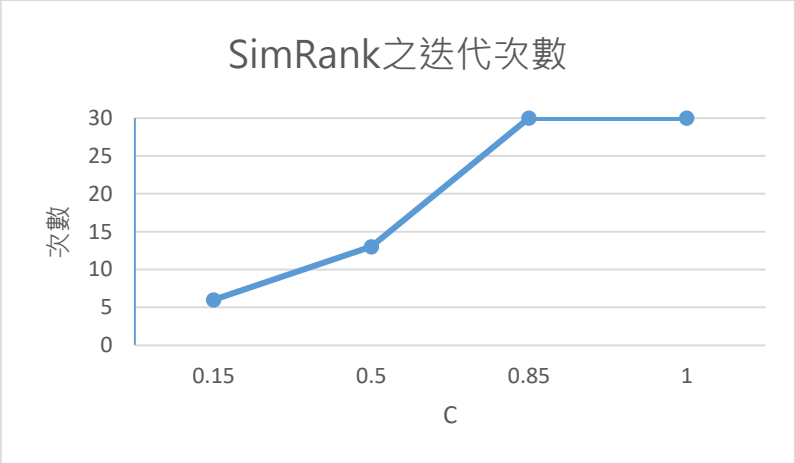
實驗結果：

從三張圖可以發現 graph_5 和 graph_6(只有前兩個演算法)在三個演算法所花費的時間都是最多的，這是因為這裡兩張圖節點

數量與邊的數量相當多，因此迭代計算花費相當長的時間。此外 simrank 由於是計算兩點之間相似度，因此需要同時查找兩個點的入鄰邊，所以在執行上所需時間更多，所以圖五可以看出再 pagerank 花費不到 5 秒，而 simrank 則需要花費超過 25 秒的時間。

- 不同的" C" 對於 SimRank 演算法的影響





實驗結果：

透過不同的 C 值去比較 SimRank 演算法在評估 graph_5 時的執行時間。結果發現當 C 值越小，所花費的時間就越短。這是因為當 C 值越小，每次迭代所產生的差異也就越小，所以使得 SimRank 值能更快收斂，也因此隨著 C 值下降，迭代次數跟著下降，執行時間則會加快。

◆ Link Analysis 演算法是否真的能夠找出「重要」的網頁？

透過這些演算法去計算具有重要性的網頁，雖然是一個很有邏輯性的想法，但是由於這些演算法簡單易懂，所以很可能可以被人為操縱。舉例來說，某個網站只要大量的在一些知名網站買廣告來產生連結，就可以提升本身的分數，但事實上對使用者來說，該網站並不沒有任何參考價值。而且過去一段時間，有人努力鑽研如何讓 pagerank 分數上升，靠的是鑽研演算法的特性，而非實際上提升網站內容品質，因此這些演算法反而可能出現本末倒置的亂象。因此未來想透過演算法找出「重要」的網頁，勢必需要想辦法解決這些買廣告提升分數的問題。

- 結論：

在這次的作業中實作了三種演算法，關於圖形節點與邊的建立，我都是利用 graph-core 這個套件去建立。而通過各個圖形跑出來的結果，都符合演算法的原理，像是 authority 與 pagerank 都跟被指向的邊數量有關，所以這兩個數值會有正向關係，而 authority 與 hub 一個在乎的是被指向，一個在乎的是指向關係，所以他們並不會出現正比或反比的趨勢。

然而在執行效能上，出現的問題是，一旦資料量很大時，執行時間會大幅上升。這應該也是 Google 將 pagerank 查詢移除掉的原因之一，它所需耗費的效能實在太大。

此外，關於網頁排名這方面，我上網搜尋發現是有一部份的人專門在做 SEO(搜尋引擎優化)，它能夠透過自然排序（無付費）的方式增加網頁能見度，這對於提升新興公司的網站曝光度，或許是最合適的管道。關於網頁排名，未來應該還有不少可能性。