HITS、PageRank 和 SimRank 三者為鏈結分析中重要的演算法，常應用在網頁的評估，並能達到很好的效果。HITS，PageRank 利用網頁間的連結性質排名出適合的名次推薦給使用者。SimRank 相較於前兩者只能衡量每個節點的重要性，還可以比較任意兩個節點的相似度，提供使用者更多能用來判斷是否符合需求的依據。

本 Project 主要實作上述三個演算法，利用不同的測試資料觀察三者的優缺點以及差別。此外也會比較三者間的效能，看何者可以快速的提供使用者不錯的網頁排名。最後會探討應用在實際網路上的情況，並找出可能的缺失，希望能進一步找到有效的解決辦法。

一、Implementation detail

➢ HITS

HITS 演算法是 Link Analysis 中非常基礎且重要的算法，Graph 資料輸入後，會經由 HITS 演算法利用迭代的方式不斷計算每個 node 的 Hub 和 Authority 分數，直到兩者達到收斂，才輸出結果，去找到高質量的 Authority node。

Algorithm 流程：

1. 初始化：讀入資料後，會將每個 node 的 Authority 和 Hub 初始化成 1。
2. 計算 Authority：此 iteration 的 Authority 為指向該 node 的所有 node 的 Hub 值總和除以最高 Authority。

$$a_t(\mathrm{v}) = \sum_{(w,v)\in E} h_{t-1}(w)$$

$$a_t = a_t / \|a_t\|$$

3. 計算 Hub：此 iteration 的 Hub 為該 node 指向的 node 的 Authority 值總和除以最高 Hub。

$$\mathrm{h}(\mathrm{v}) = \sum_{(v,w)\in E} a_{t-1}(w)$$

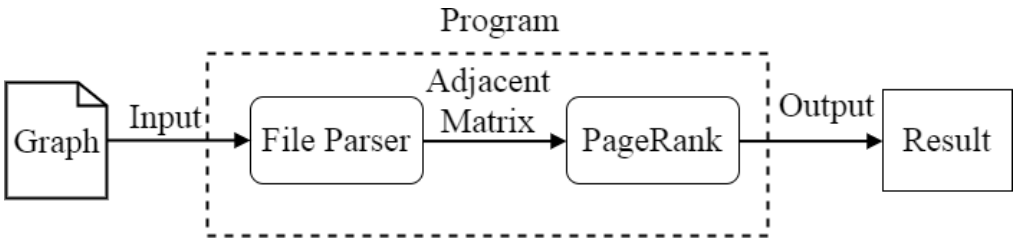
$$h_t = h_t / \|h_t\|$$

4. 判斷是否收斂：判斷前一 iteration 的 Authority、Hub 與當下的 Authority、Hub 差值和是否小於收斂門檻值，如果是則終止迭代輸出結果，否則繼 續下一 iteration 的計算直到收斂。

➢ PageRank

系統架構如圖一所示，首先，File Parser 會讀取 Graph 檔案，並計算出 Graph 的 Adjacent Matrix，然後 PageRank 會根據事先設定的 Damping Factor、收斂標準門檻 threshold 與先前計算出來的

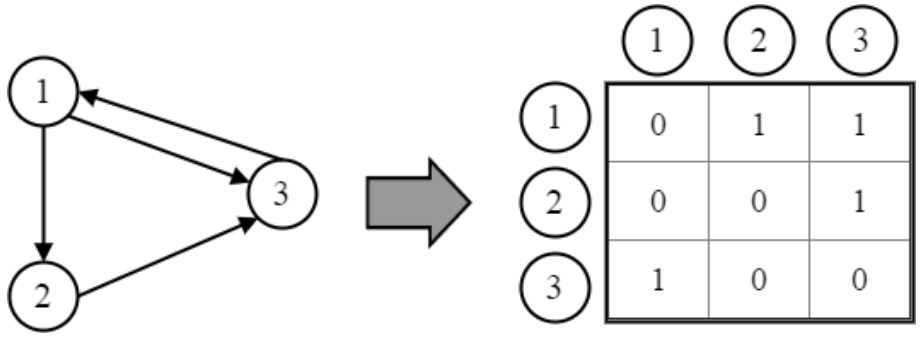
Adjacent Matrix 計算出每個 node 的 Rank 值，最後將計算結果顯示出來。



圖一、PageRank 系統架構圖

File Parser：

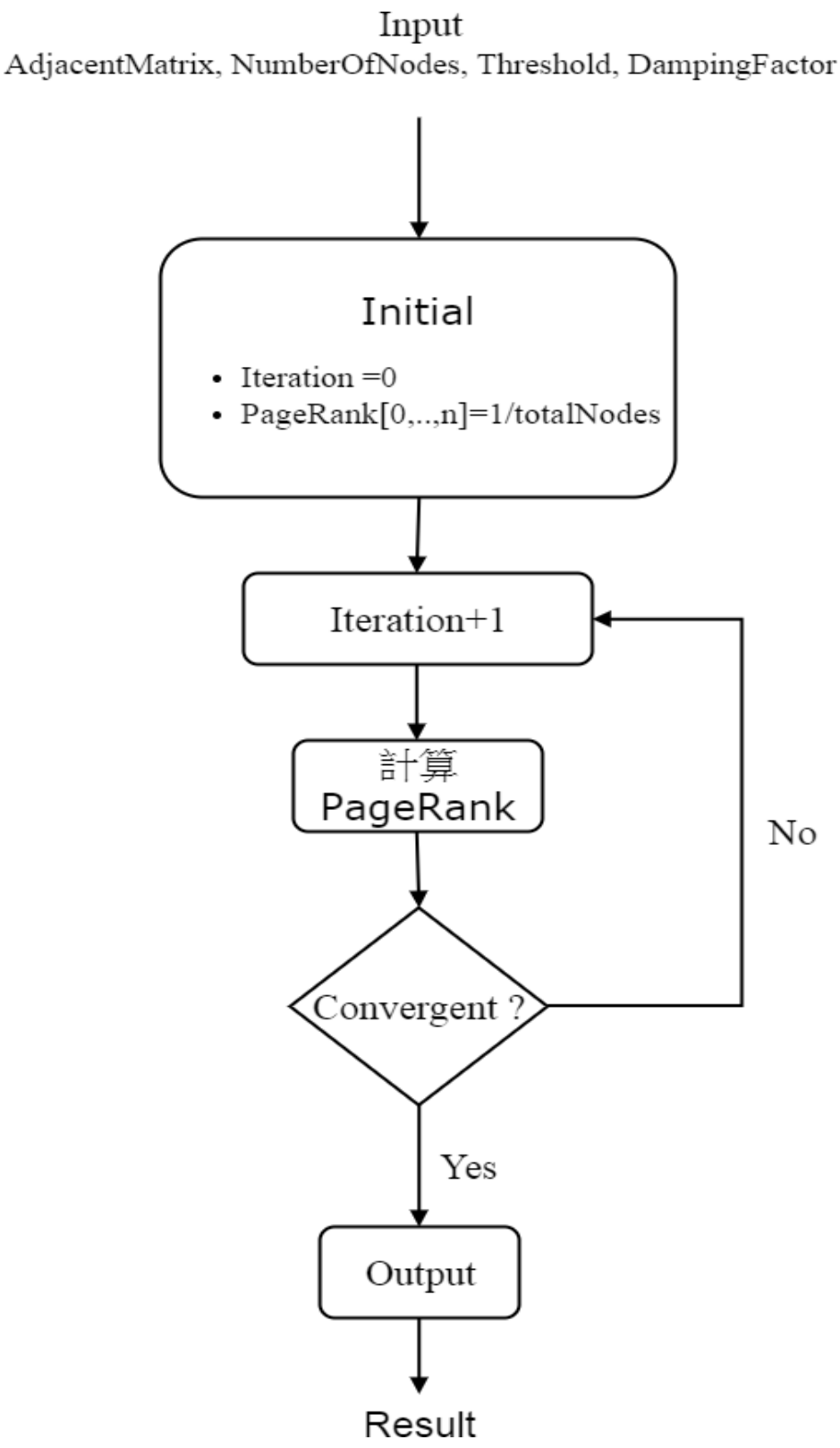
File Parser 負責將讀取的 Graph 轉換成 Adjacent Matrix，假設 Graph 共有 n 個 node，則 File Parser 會產生一個n × n的 Adjacent Matrix，若 node a 指向 node b，則 Adjacent Matrix[a][b]=1，否則為零，如圖二所示。



圖二、File Parser 範例

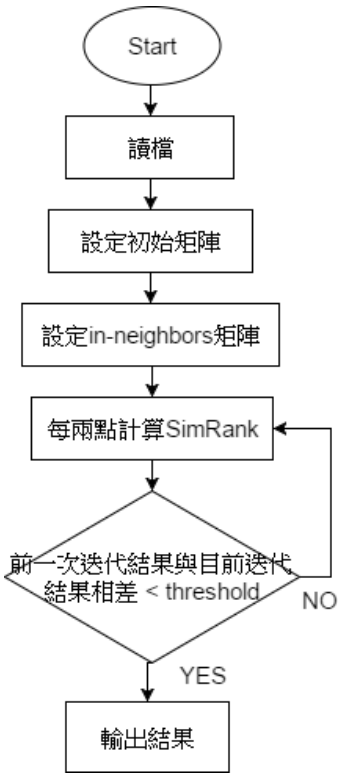
PageRank：

PageRank 流程圖如圖三所示。



圖三、PageRank 流程圖

➢ SimRank



圖四、SimRank 流程圖

SimRank 流程如圖四，讀檔後會先初始化矩陣，如圖五左，[x,y]當 x、y 相等為 1，其餘為 0，並另用一矩陣紀錄指向此節點者(in-neighbors)，以 3->1 為例，在[1,3]的位置設為 1，即代表節點 1 有來自節點 3 的 in-link 指向它，我們稱節點 3 為節點 1 的 in-neighbor，如圖五右，在獲得此兩陣列後，便可進入 SimRank 演算法。

	1	2	3	4	5
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	0	1	0
5	0	0	0	0	1

	1	2	3	4	5
1	0	0	1	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

圖五、SimRank 矩陣範例

SimRank 計算：

原理：若節點 a 和節點 b 依賴於相同的節點，也就是有一節點同時指向 a 與 b，那麼我們認為 a 和 b 是相似的，用 s(a,b)表示兩個節點間的相似度， 用記號 I(a)表示所有指向節點 a 的節點集合（即 in-neighbors 集合）。

二、Result analysis and discussion

➢ 開發環境與 Dataset

實驗環境：

CPU	Intel-i5-3570 3.4GHz
Memory	8GB
OS	Windows 10 x64

實作語言：

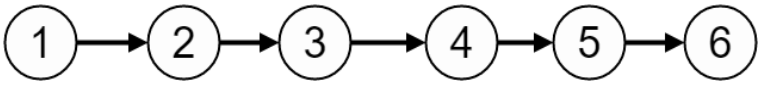
HITS	C++
PageRank	Java
SimRank	Java

下表為我們測試時所用的圖形資訊與測試方法，將分別針對每張圖形的結果及三個方法的效能做分析與討論，最後，為我們額外探討的結果。其中 Graph 7, Graph 8 為前兩次作業生成圖。

Graph	Node	Method
1	6	HITS, PageRank, SimRank
2	5	HITS, PageRank, SimRank
3	4	HITS, PageRank, SimRank
4	7	HITS, PageRank, SimRank
5	469	HITS, PageRank, SimRank
6	1228	HITS, PageRank
7		HITS, PageRank
8		HITS, PageRank

➢ Analysis and discussion

Graph 1:



- **HITS**

  這是一條從 1 到 6，方向為單向的路徑，可以發現因為節點 1 沒有被任何節點指向，所以 Authority 分數為 0，而結點 6 沒有指向任何的節點，所以其 Hub 分數為 0。

Authority	
Node	Score
1	0
2	0.4472
3	0.4472
4	0.4472
5	0.4472
6	0.4472

Hub	
Node	Score
1	0.4472
2	0.4472
3	0.4472
4	0.4472
5	0.4472
6	0

- **PageRank【以下實驗參數 d 皆使用 0.15】**

  此圖為一簡單直線的連結，依 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。計算結果如下表所示，從結果來看便可驗證 PageRank 由起點 Node 1 到終點 Node 6 確實有逐漸上升的趨勢。

Node	PageRank
1	0.15
2	0.2775
3	0.385875
4	0.477994
5	0.556295
6	0.62285

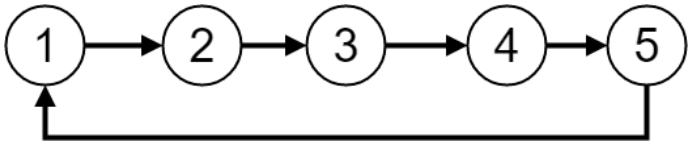
- **SimRank【以下實驗參數 C 皆使用 0.85】**

  由於此圖各節點沒有相同的 in-link，即為相異的兩個點間不被同一個節點指向，因此只有在當 a = b 時，s(a,b) = 1，其他任兩點間的 SimRank 皆為 0。

	1	2	3	4	5	6
1	1	0	0	0	0	0
2	0	1	0	0	0	0
3	0	0	1	0	0	0

4	0	0	0	1	0	0
5	0	0	0	0	1	0
6	0	0	0	0	0	1

Graph 2:



- **HITS**

  這是一條從 1 到 6，再從 6 回到 1，方向為單向的 cycle，可以發現每個節點都被一個節點指向，也各自指向一個節點，所以 Authority 和 Hub 數值都相同。

Authority	
Node	Score
1	0.4472
2	0.4472
3	0.4472
4	0.4472
5	0.4472
6	0.4472

Hub	
Node	Score
1	0.4472
2	0.4472
3	0.4472
4	0.4472
5	0.4472
6	0.4472

- **PageRank**

  此圖為一單向的環狀圖形，所有 Node 都將自己的分數完整傳遞給下一個 Node，因為沒有 Node 得到較多的 PageRank，故所有 Node 的 PageRank 值皆相同，代表著所有 Node 擁有同等的重要性。

Node	PageRank
1	0.999467
2	0.999467
3	0.999467
4	0.999467
5	0.999467
6	0.999467

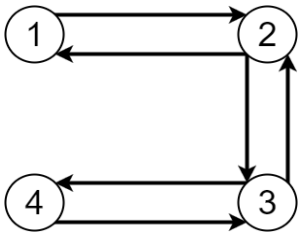
- **SimRank**

  此圖為單向循環，基本上因為與 Graph 1 同樣狀況，在相異的兩個點間不被同一個節點指向，因此也是只有當 a = b 時，s(a,b) = 1，其他任兩點間的 SimRank 皆為 0。

	1	2	3	4	5	6
1	1	0	0	0	0	0

2	0	1	0	0	0	0
3	0	0	1	0	0	0
4	0	0	0	1	0	0
5	0	0	0	0	1	0
6	0	0	0	0	0	1

Graph 3:



- **HITS**

  這是一條從 1 到 4，方向為雙向的路徑，節點 2、3 同時指向兩個節點，也同時被兩個節點指向。而節點 1、4 只有指向一個節點，也只有被一個節點指向。所以可以看出 2 和 3 分數一樣，1 和 4 分數一樣，而 2、3 分數又高於 1、4。

Authority	
Node	Score
1	0.3717
2	0.6015
3	0.6015
4	0.3717

Hub	
Node	Score
1	0.3717
2	0.6015
3	0.6015
4	0.3717

- **PageRank**

  此圖中，節點 2、3 除了可以分得節點 1、3 的完整 PageRank，還可以互相傳遞一半的 PageRank 給對方，故此兩個節點的分數相同且最高，而節點 1、4 由於只能分別從節點 2、3 分到 PageRank，所以分數較低，由結果顯示節點 2、3 有較高的重要性。

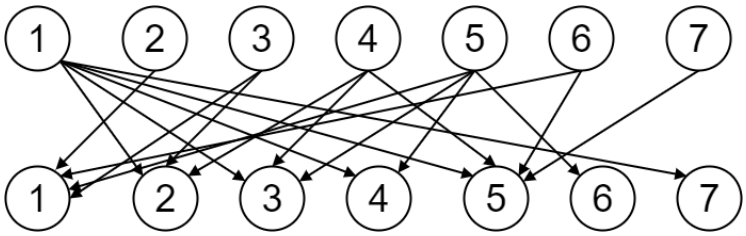
Node	PageRank
1	0.701471
2	1.297679
3	1.297679
4	0.701471

- **SimRank**

  此圖可發現兩點若中間有一節點間隔，此兩點的相似度為 0.739130，原因為此兩點都有來自中間節點的 in-link，以節點 1、2、3 來說，節點 1 和 3，同時有來自節點 2 的邊，若沒有則 SimRank 為 0

	1	2	3	4
1	1	0	0.739126	0
2	0	1	0	0.739126
3	0.739126	0	1	0
4	0	0.739126	0	1

Graph 4:



- **HITS**

  可以發現節點 1 和節點 5 同樣被 4 個節點指向，但節點 5 的 Authority 明顯高於節點 1，可能是因為指向節點 1 的節點不是好的 Hub。

Authority	
Node	Score
1	0.3467
2	0.4221
3	0.4991
4	0.3484
5	0.5006
6	0.1394
7	0.2089

Hub	
Node	Score
1	0.6464
2	0.1120
3	0.2550
4	0.4662
5	0.4311
6	0.2739
7	0.1618

- **PageRank**

  此圖因為節點 1 和節點 5 有最多的 In-degree，故分數較高，節點 6 和節點 7 的 In-degree 最少，所以分數最低。

Node	PageRank
1	1.961476
2	1.111057
3	0.971923
4	0.75735
5	1.289069
6	0.423915
7	0.483435

- **SimRank**

  此圖比前述 3 個圖的節點與邊數都較多一些，經過 SimRank 迭代過程兩兩排列組合後，得出的結果可以看出即便節點間 皆互有

影響力，然而卻有高低之分。

	1	2	3	4	5	6	7
1	1	0.446005	0.435005	0.438759	0.424222	0.494996	0.382522
2	0.446005	1	0.487587	0.453016	0.494343	0.375157	0.530874
3	0.435005	0.487587	1	0.526210	0.472583	0.524608	0.527827
4	0.438759	0.453016	0.52621	1	0.427394	0.605294	0.605284
5	0.424222	0.494343	0.472583	0.427394	1	0.362602	0.492207
6	0.494996	0.375157	0.524608	0.605294	0.362602	1	0.360612
7	0.382522	0.530874	0.527827	0.605284	0.492207	0.360612	1

Graph 5:

- **HITS**

下列是列出 Authority 分數最高的五個節點，可以明顯發現 Authority 高的節點 Hub 不一定會高。

Authority	
Node	Score
6	0.4914
12	0.4826
21	0.2951
10	0.2867
282	0.2548

Hub	
Node	Score
6	0
12	0
21	0.1411
10	0
282	0

- **PageRank**

由於此圖 Node 數較多，於是我們只找 Top-5 高分的 Node，下表為結果，根據分數排名與 Node 的 in-degree 可發現，越高的 in-degree 通常 PageRank 值也越高。

Top-5 PageRank		
Node	PageRank	in-degree
61	1.342841	48
122	1.321657	43
10	0.961491	27
21	0.730684	25
282	0.693078	25

- **SimRank**

此圖是 SimRank 的測試中，節點數最多的一張，但經過 SimRank 得到的結果，可以看出多數的點與點間相似度並不高，甚至為 0，由此猜測這張圖的節點數雖多，但互連的邊數卻相對較少，也可說是兩點來自同一節點的 in-link 的狀況較少，因此結果呈現兩極化，當節點間有關聯 SimRank 就會偏高分。

Graph 6:

- **HITS**

  最高的 5 個節點分數都很靠近，而且都不高於 0.3，可能是因為迭代比較多次，而且指向這幾個節點的節點 Hub 分數差不多的關係。

Authority	
Node	Score
76	0.2751
1151	0.2751
62	0.2730
78	0.2717
394	0.2653

Hub	
Node	Score
76	0
1151	0
62	0.0872
78	0.0968
394	0

- **PageRank**

  由於此圖 Node 數較多，於是我們也只找 Top-5 高分的 Node，下表為結果，仍然可以根據分數排名與 Node 的 in-degree 可發現，越高的 in-degree 通常 PageRank 值也越高。

Top-5 PageRank		
Node	PageRank	in-degree
105	0.849990	89
76	0.686826	68
115	0.686826	68
62	0.682696	68
394	0.666627	65

Graph 7:

- **HITS**

  下列數據取 Authority 和 Hub 分數最高的前五個節點。圖形的特性和實驗結果可以看出，Authority 和 Hub 剛好以數值相似排序相反的方式呈現，因為被指向和指向的數目隨著編號呈規律的遞增和遞減。

Authority	
Node	Score
1	0.3285
2	0.3261
3	0.3214
4	0.3143
5	0.3050

Hub	
Node	Score
1	0.3284
2	0.3261
3	0.3214
4	0.3143
5	0.3050

- **PageRank**

  由於此圖的規律是編號越低的 Node 會指向編號高的 Node，也因此 in-degree 會與根據編號遞增，下表為執行結果，Node 19 有最

大的 in-degree，所以分數最高，之後慢慢遞減。

Top-5 PageRank		
Node	PageRank	in-degree
19	1.931389	19
18	1.043997	18
17	0.732300	17
16	0.570880	16
15	0.470829	15

Graph 8:

- **HITS**

可以發現節點 2、8、17 被指向和指向別節點的情況很類似，他們同時是最好的 Authority 也是最好的 Hub。

Authority	
Node	Score
2	0.4180
8	0.4180
17	0.4180
5	0.3941
1	0.3158

Hub	
Node	Score
2	0.4108
8	0.4108
17	0.4108
5	0.3470
1	0.2732

- **PageRank**

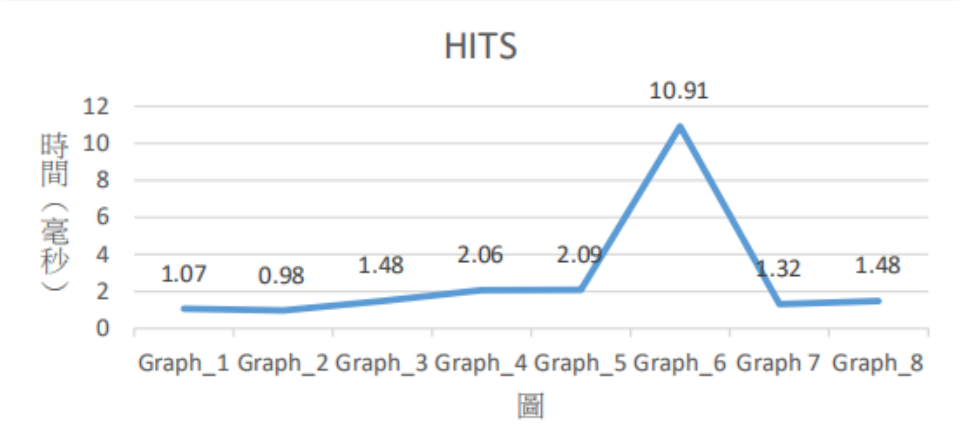
下表為此圖的結果，仍然可以根據分數排名與 Node 的 in-degree 發現，越高的 in-degree 通常 PageRank 值也越高，但兩者之間並沒有比例關係，就如同 Node 2 的 in-degree 為 Node 1 的 4 倍，但 PageRank 上並沒有四倍的關係。

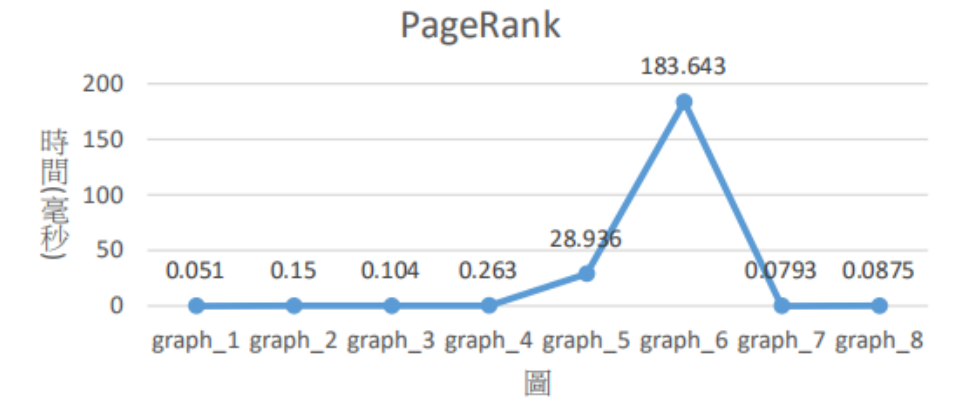
Top-5 PageRank		
Node	PageRank	in-degree
2	1.393720	8
8	1.393720	8
17	1.393720	8
5	1.203025	7
1	0.894612	2

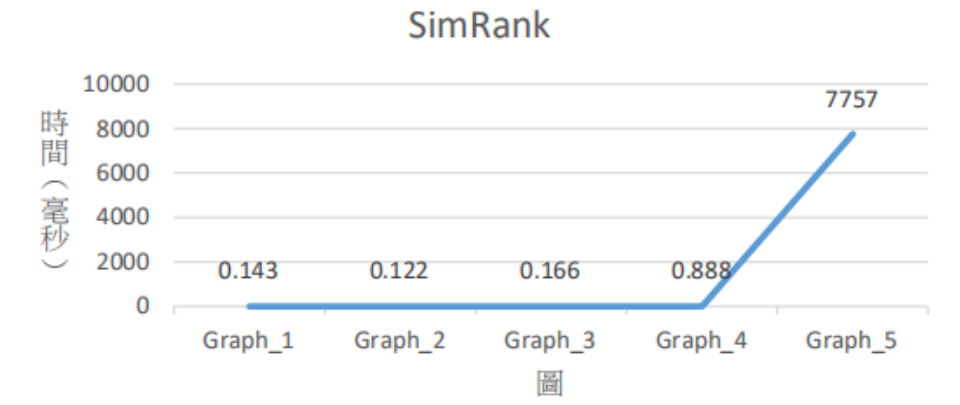
三、Computation performance analysis

針對各個 Graph 在不同演算法下的 runtime 如下圖，可以明顯發現 Graph 5 與 Graph 6 在此三個方法下皆花最多時間，原因是大量的 Node 與複雜的 Link，導致大量的迭代，使執行時間增加。

HITS 與 PageRank 效能上的差異可能主要來自於實作語言的不同，其他可能影響的因素還有運算方式的不同，以致於迭代需求數不一致，即便針對同一張圖，兩個方法執行下來會有不同的迭代次數，而次數高者需要越多的執行時間。







四、Discussion

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

  我們認為這些方法在某些情況下沒辦法找到那些真正重要的網頁，例如：某一個網站只要大量的在其他重要的網站買廣告來產生連結，就可以在不需要考慮到網站的內容品質的情況下，提升本身的分數，使得有更高的機會被搜尋到，但網站內容不一定是重要的。

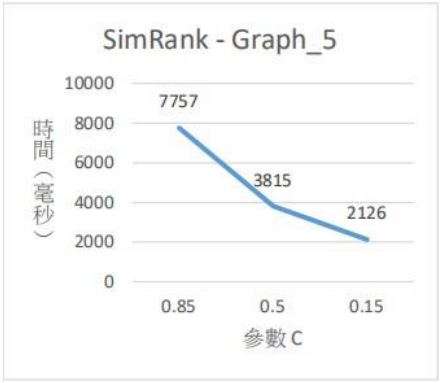
- **What are practical issues when implement these algorithms in a real Web?**

  實際上 Web 數量非常多且網頁內容更新非常快，一台電腦無法負荷這麼大的計算量，因此需要在分散式的架構下，去實做這些方法，才能提升效能。

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

  下圖分別為針對 Graph 5 做 SimRank 計算時，調整不同大小的數 C 所得到的 runtime 及迭代次數結果。

  C 值越小，迭代所產生的前後次結果差異也就越小，而 SimRank 值便能越快收斂，也因此隨著迭代次數的下降，執行時間會變快。



- **心得**

  在這次 Project 中，我了解到 Link Analysis 常見方法的運作原理以及優缺點，也藉此了解早期搜尋引擎是如何找出重要的網站的，雖然原理很簡單，但實際上仍有許多問題存在。

  使用的圖結構並不複雜，但我發現計算過程中，仍有機會產生大量的迭代計算，這讓我更無法想像，當初 Google 是如何實作這個方法，並且用於數百萬筆網站資料上是非常困難和具挑戰的。

  除此之外，這些方法仍然有一些缺點，例如：新的網站分數通常都會非常低，而高分的網站可能培養出自己其它高分的網站，導致最後搜尋的結果並不是最適合使用者的，可能需要搭配一些針對網路內容的過濾機制，或者混合多種演算法，才能解決這些問題，就如同 Google 目前搜尋引擎的運作一般。