**資料探勘DATA MINING**

Project 3

Q36071059 蔡榮漾

**Implementation detail**

分成三個程式，分別為SimRank.py、HITS.py與PageRank.py

**HITS.py：**

利用pygraph.classes.digraph中的digraph製作圖結構，首先讀取txt資料後，將邊與點等資料加入digraph結構中，之後進行authority與hub，並標準化authority與hub值。

**PageRank.py：**

利用networkx中的DiGraph製作圖結構，讀取原資料後加入點資料，並加入邊資料；之後進行pagerank的計算，並繪圖呈現結果。

其他詳細的程式內容與註解在資料夾中的SimRank.py、HITS.py與PageRank.py

**Result analysis and discussion**

下列為各演算法之結果

1. **HITS：**

graph\_1：

The best authority page: ('2', 0.447213595499958)

The best hub page: ('1', 0.447213595499958)

graph\_2：

The best authority page: ('1', 0.447213595499958)

The best hub page: ('1', 0.447213595499958)

graph\_3：

The best authority page: ('2', 0.6015031709352148)

The best hub page: ('2', 0.6015001085928517)

graph\_4：

The best authority page: ('5', 0.5006306330112105)

The best hub page: ('1', 0.6464226045058622)

graph\_5：

The best authority page: ('61', 0.49135076349766016)

The best hub page: ('274', 0.191943875392767)

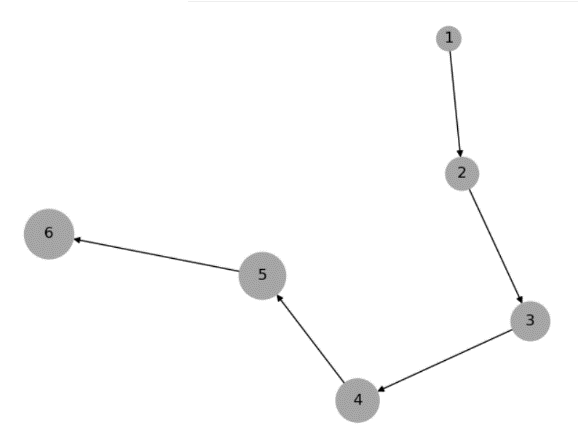
graph\_6：

The best authority page: ('761', 0.27506387286601447)

The best hub page: ('171', 0.15626248215295482)

1. **PageRank：**

graph\_1：

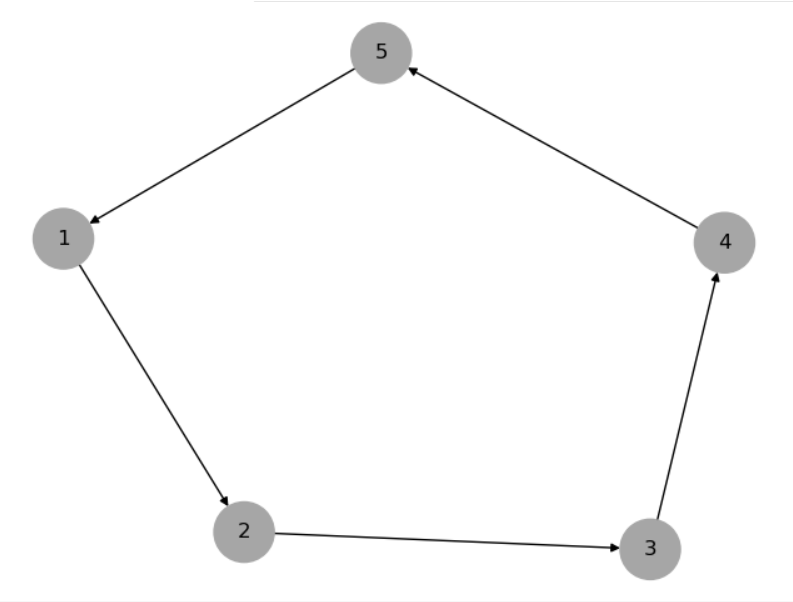
graph\_1的所有node最終會全部指向6(因為6沒有指向任何人)，若未設定damping factor，最後大家的Pagerank都會為0，此處設定damping factor為0.15得到右圖；若點越大，代表其Pagerank值越高，此途中我們可見6為Pagerank值最高的點，這是因為在此圖中他是所有點的終點。

Pagerank:



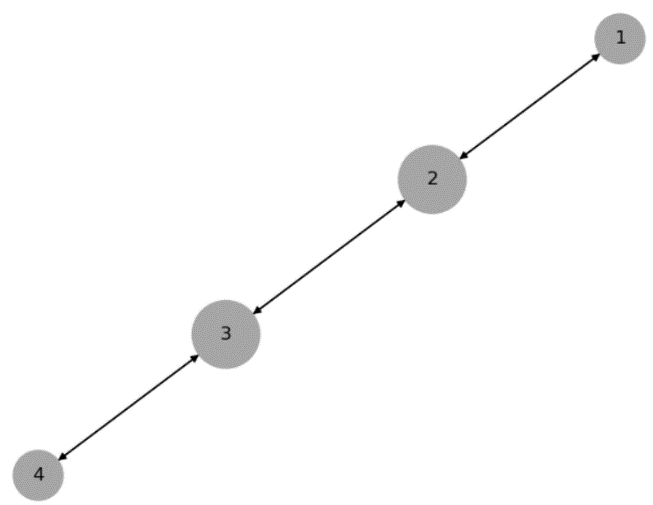
graph\_2：

可以很清楚的看見圖二是一個單向環繞的封閉圖，故每個人的Pagerank收斂後會相同



Pagerank:

graph\_3：



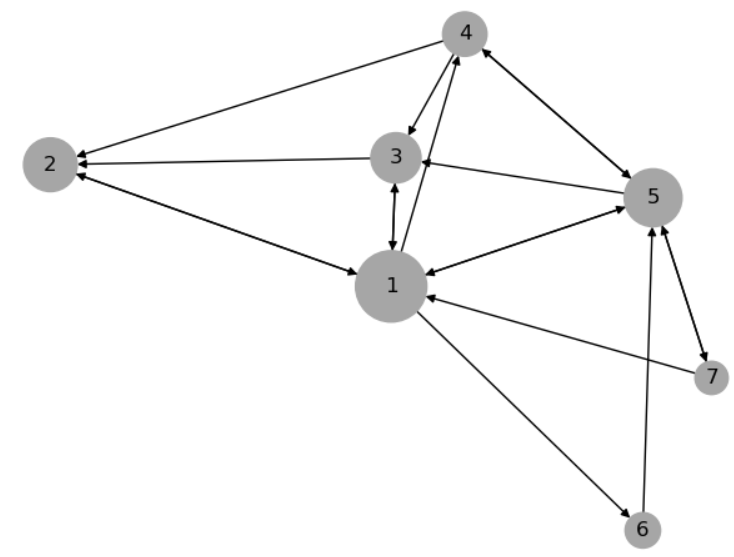
由於node 1.4分別只與node 2.3雙向相連，故node2.3之Pagerank會較node 1.4 高，且node 2.3近乎相同，node 1.4也近乎相同

Pagerank:



graph\_4：

Pagerank最高者為node 1 與node 5

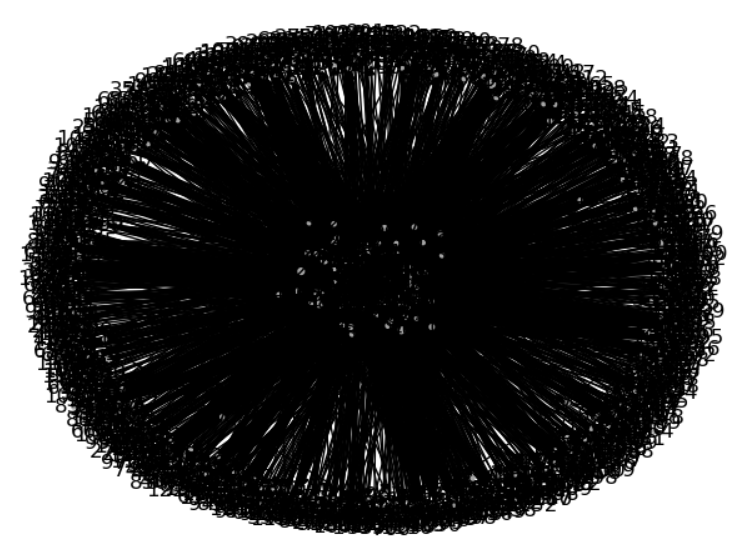
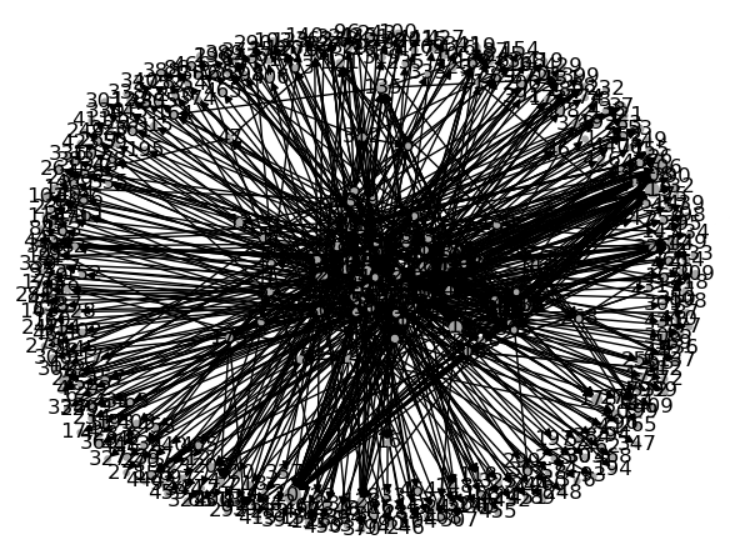


Pagerank:



graph\_5 & graph\_6：

已無法從圖分辨



Pagerank of graph\_5:



Pagerank of graph\_6



1. **SimRank：**

graph\_1：

[[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.]]

graph\_2：

[[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\_3：

[[1.00 0.00 0.67 0.00]

[0.00 1.00 0.00 0.67]

[0.67 0.00 1.00 0.00]

[0.00 0.67 0.00 1.00]]

graph\_4：

[[1.00 0.27 0.33 0.39 0.35 0.38 0.33]

[0.27 1.00 0.51 0.25 0.41 0.28 0.54]

[0.33 0.51 1.00 0.38 0.36 0.30 0.40]

[0.39 0.25 0.38 1.00 0.34 0.47 0.36]

[0.35 0.41 0.36 0.34 1.00 0.27 0.34]

[0.38 0.28 0.30 0.47 0.27 1.00 0.54]

[0.33 0.54 0.40 0.36 0.34 0.54 1.00]]

graph\_5：

[[1.00 0.00 0.00 ... 0.00 0.00 0.00]

[0.00 1.00 0.00 ... 0.00 0.00 0.00]

[0.00 0.00 1.00 ... 0.00 0.00 0.00]

...

[0.00 0.00 0.00 ... 1.00 0.00 0.00]

[0.00 0.00 0.00 ... 0.00 1.00 0.00]

[0.00 0.00 0.00 ... 0.00 0.00 1.00]]

**Computation performance analysis**

Cost time：

HITS：0.76 s

PageRank：22.57 s

SimRank：0.10 s

**Discussion**

這次的報告讓我學到很多，例如networkx套件的使用，以及其所包含的DiGraph結構，與我本來所認識的，經常拿來儲存圖的結構digraph有所不同，除此之外，在課堂外實作出上課時教過的演算法，也是一次難得的經驗，一次實作了SimRank、HITS與PageRank 三種link analysis，也讓我稍微加強了Python的寫作能力，尤其是對於array和dataframe的部分。