Data Mining

Project3 Link Analysis

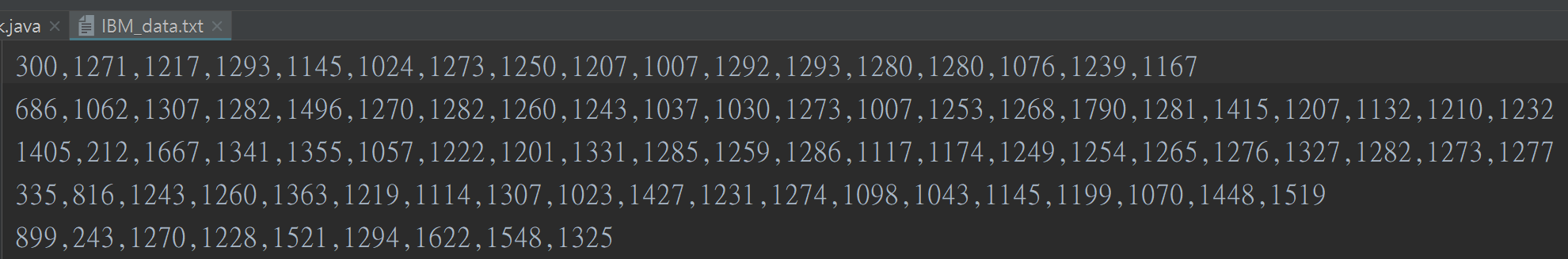
孫啟慧

P78063033

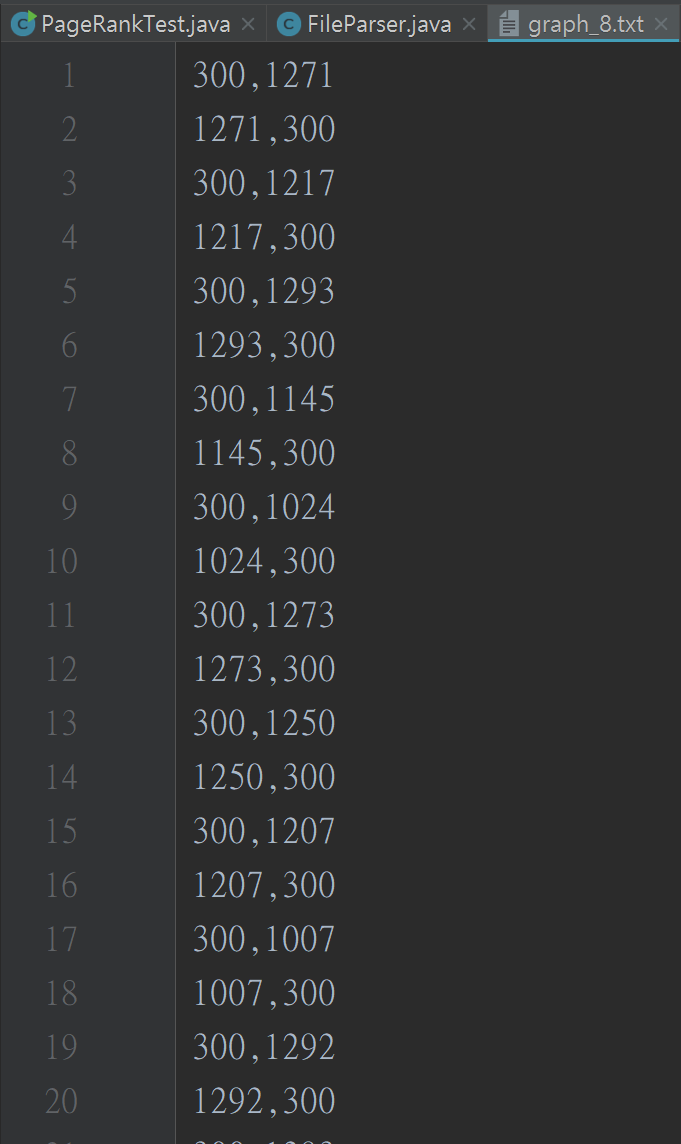
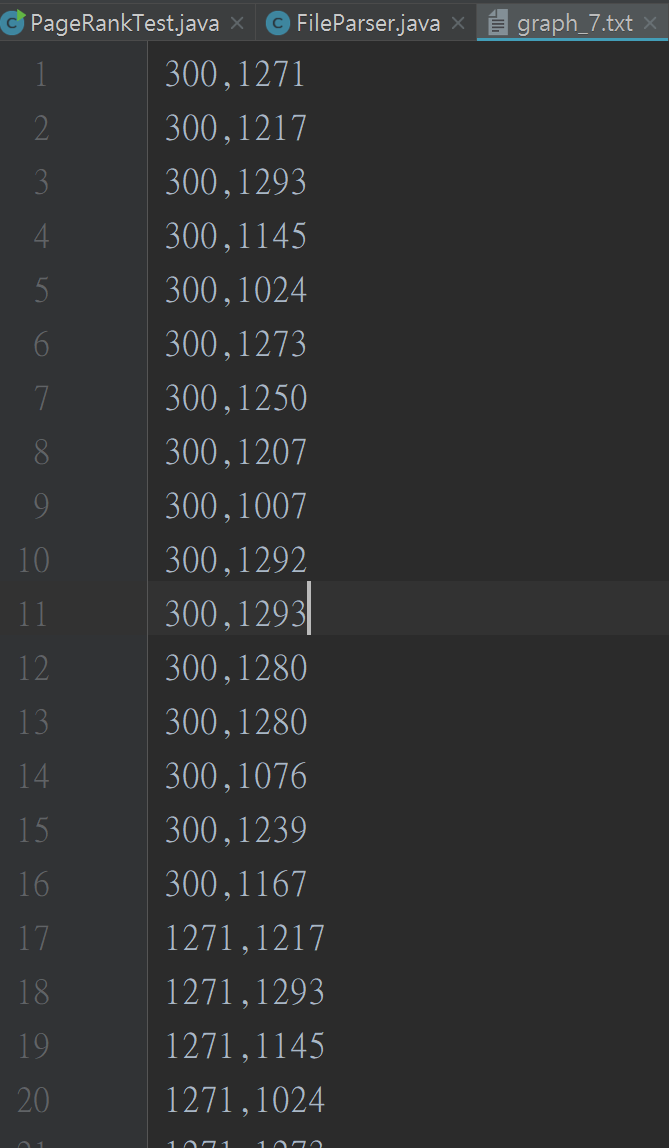
1. **實驗設計**

本次作業通過要求我們實現三種常見的不同的Link Analysis算法，HITS, PageRank和SimRank，從而加深我們對Link Analysis算法理解。為了進行各種算法的比較，moodle中有提供六中基本的graph，另外根據project1中的transaction data生成相應的有向圖和無向圖可以得到graph 7和8。

因為三個演算法中我先實作pagerank，所以相應的input graph的處理文件（原始data和相應的程式碼都在pagerank的文件夾中）。其中PagrRank/dataset/IBM\_data.txt是project1中所生成的transaction data。（因為我自己的pagerank中會將整個graph存成matrix的形式，而我所使用的JAVA IDEA對于內存有所限制，如果直接採用全部project1的data，則java的memory會爆掉，因此在這裡對原始的input進行了部分修改，并按照transaction id和customer id對商品id進行了重新整理，每一行代表同一transaction id和customer id的一筆transaction下購買的物品（詳細內容可以參見project1的報告））。相應的IBM\_data.txt如下圖所示：



為了得到相應graph7和8，考慮到其有向邊和無向邊的關係，PageRank/src/DataPrepare.java中是對數據進行預處理的過程，因為每一筆transaction內的物品間均有關係，且其關係將表示成邊的關係。處理后的數據有兩欄，具體結果如 PageRank/dataset/graph7.txt和同目錄下的graph8.txt所示，而其中點和邊的詳細信息在Read\_me.txt中可以找到。下圖分別是graph7和graph8的部分數據形式。



1. **Link Analysis演算法的原理及實作**

在本次實驗中我們選取了HITS, PageRank和SimRank三種演算法進行實作。相應的代碼在project3下面的文件夾中均可以找到。

1. **HITS**

根據網頁的入度和出度來衡量一個網頁的重要性，其中入度指的是指想這個網頁的超鏈接即Authority，而出度則表示這個網頁指向其他網頁的超鏈接即Hub。

用authority[k]和hub[k]來分別表示，第k個node相應的authority和hub值。首先將其均初始化為全1的數組，之後迭代計算authority和hub，其中authority權值為所有指向node k的hub權值之和，而hub權值則為所有指向node k的authority權值之和。對authority和hub分別處以各自最高的值來進行標準化。重複上述過程直到兩者的計算收斂。

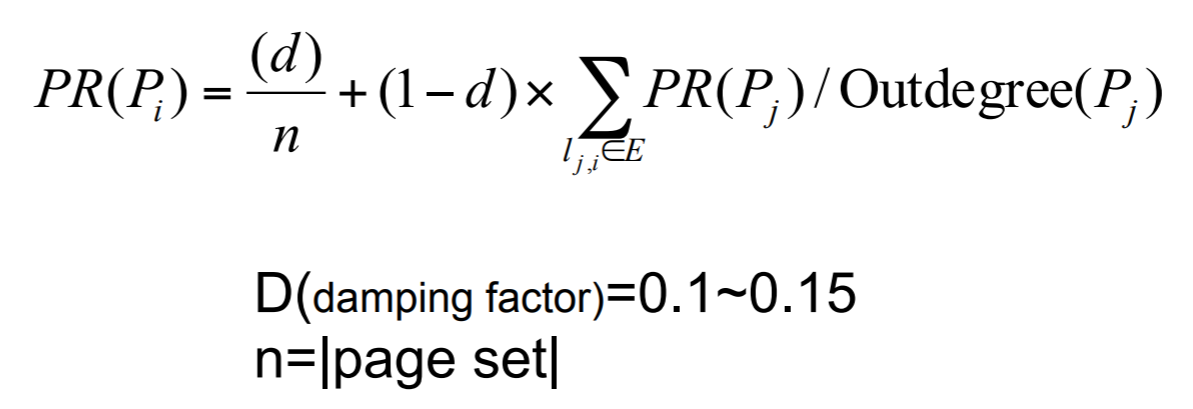
在這裡我有寫三個檔案，分別是FilePaser負責讀取相應graph檔案，生成鄰接矩陣（Adjacent Matrix）。而Hits則會根據給定的threshold和Adjacent Matrix一起計算出各個node的authority和hub值，并將結果進行顯示。HitsTest則是用來傳遞input graph進行test的檔案。

1. **PageRank**

主要利用網頁間的連接性對網頁進行排序并選取名詞較高的網頁推送給使用者。

在這裡我有寫三個檔案，分別是FilePaser負責讀取相應graph檔案，生成鄰接矩陣（Adjacent Matrix）。而PageRank則會根據給定的Damping factor，threshold和Adjacent Matrix一起計算出各個node的Rank值，在排序之後將結果進行顯示。PageRankTest則是用來傳遞input graph進行test的檔案。

對於FilePaser來說，如果input的graph有n個節點，則會產生n\*n維的Adjacent Matrix，其中如果node a有邊指向node b則Matrix[a][b]=1否則等於零。

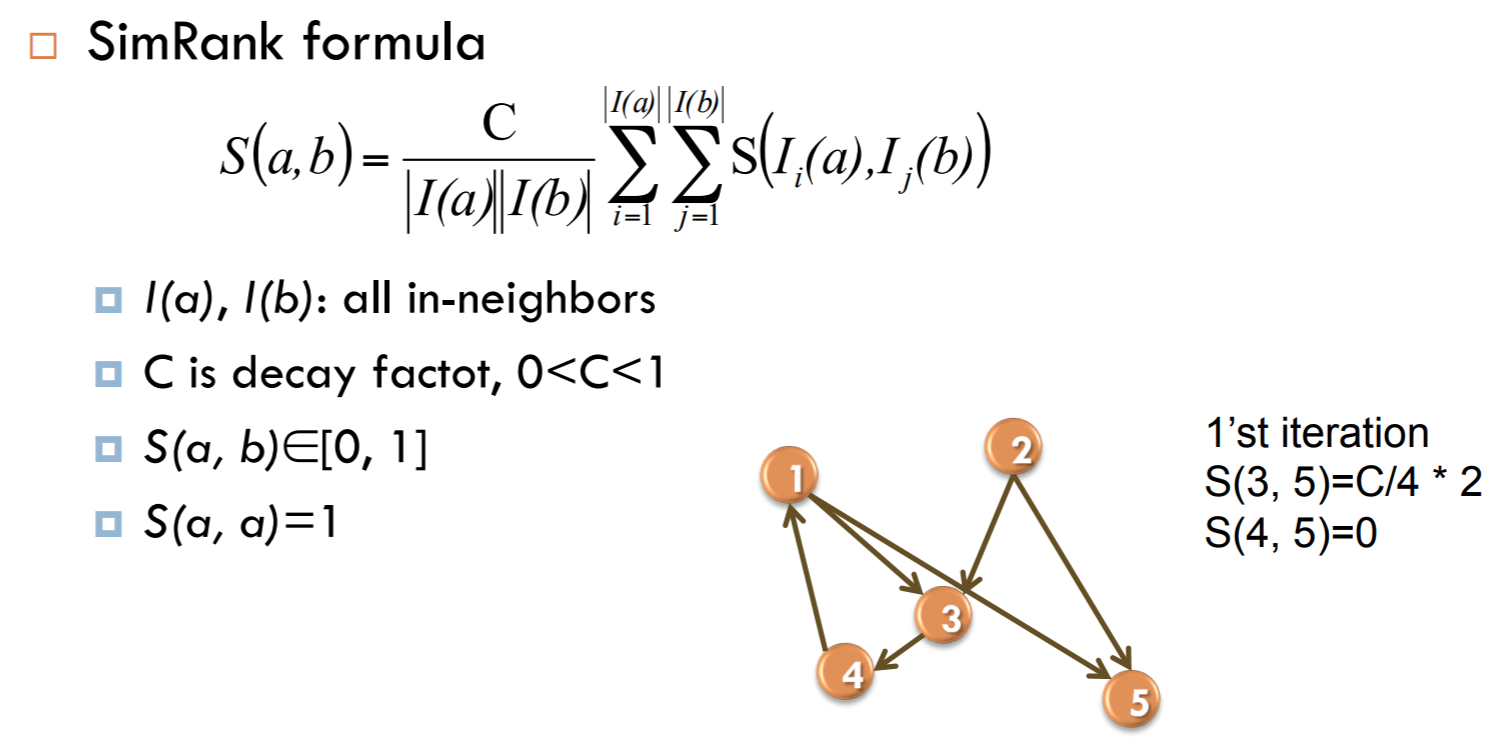
在PageRank中首先各個node的pagerank (pr) 會被初始化成1/totalnodes，之後按照下面公式進行計算

當前後兩個iteration得到的pr的差值之和小於threshold，也就是說計算收斂，則停止整個ierator并將結果按照rank進行sort并輸出。

1. **SimRank**

該演算法基於圖論，假設用於推薦的時候，則將用戶和物品分成一張二部圖，一邊是用戶而另外一邊是物品，圖中任意一條邊的兩個端點分別來自兩邊，且兩邊內部的點並沒有連接。

在這裡我只有寫一個檔案simRank.java，首先會讀取相應的graph input，之後生成其in-neighbors矩陣，通過下面的公式來計算兩個點之間的SimRank。當兩次迭代之間的結果差大於threshold時會重複計算SimRank直至停止。



在SimRank的計算過程中，如果node a和b依賴於相同的節點，則認為這兩個node相似，s(a,b)表示兩個節點之間的相似度，而I(a)則表示所有指向a節點的集合。

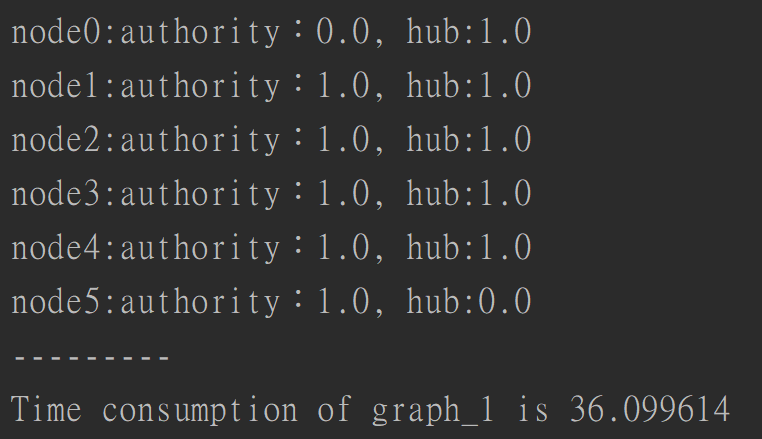
1. **實驗結果**

各種演算法的不同input graph下的計算結果

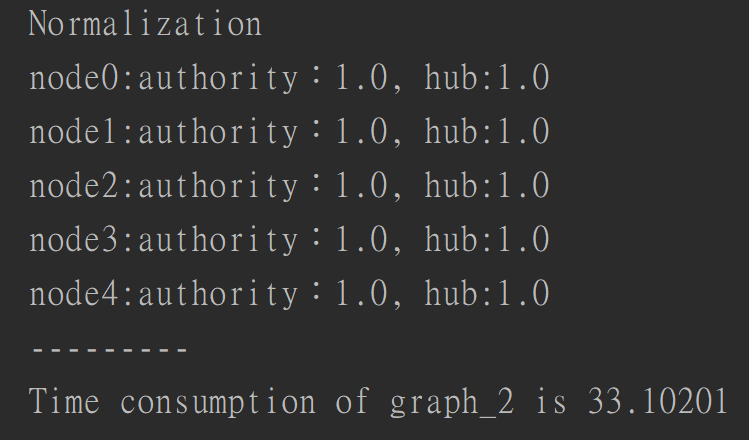
1. **HITS**

此處展示的結果包括排authority和hub以及完成整個迭代運

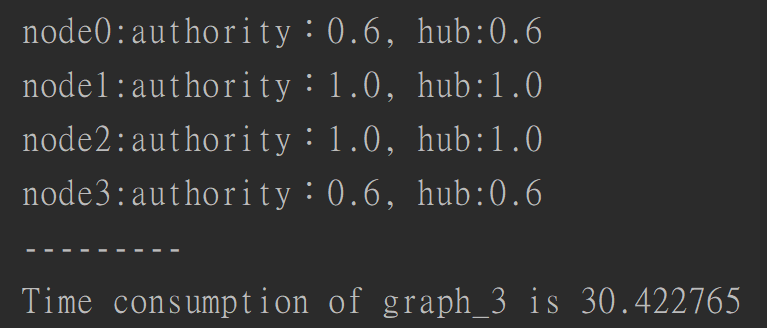
的時間。



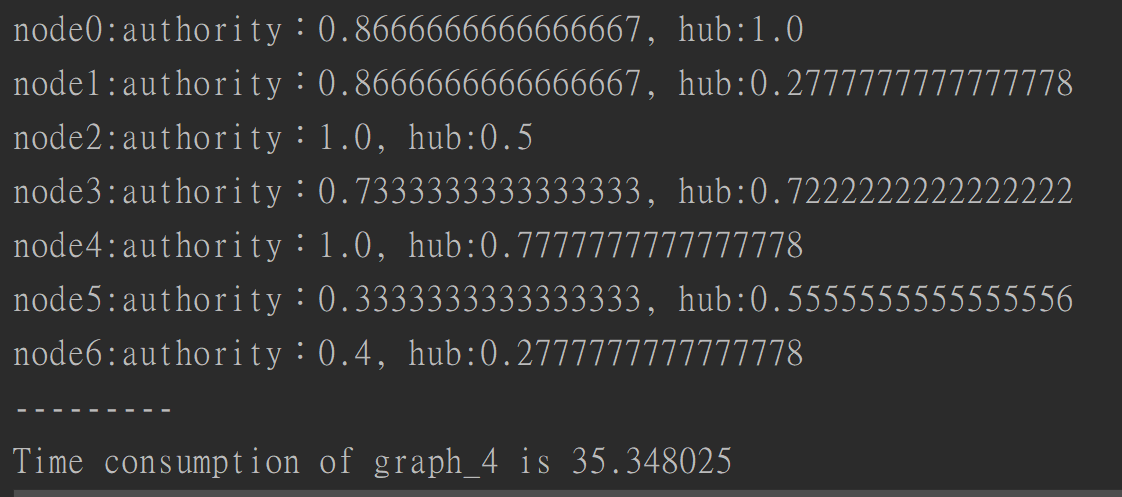
因為graph1是一條從node 1到node 6的單向圖，並沒有任何node指向node1，故其authority為1，同理，因為node 6並沒有指向任何node，所以其hub為0.



因為graph2是一個從node1依次指到node6再指回node1的單向圖，可以發現圖中的每個節點均有被其他節點指到，故其authority和hub是相同的。

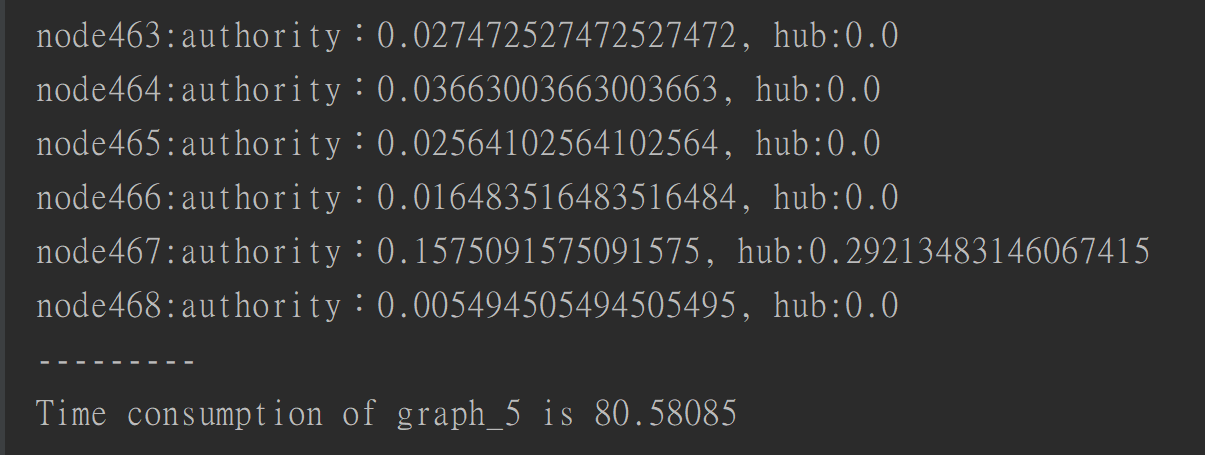


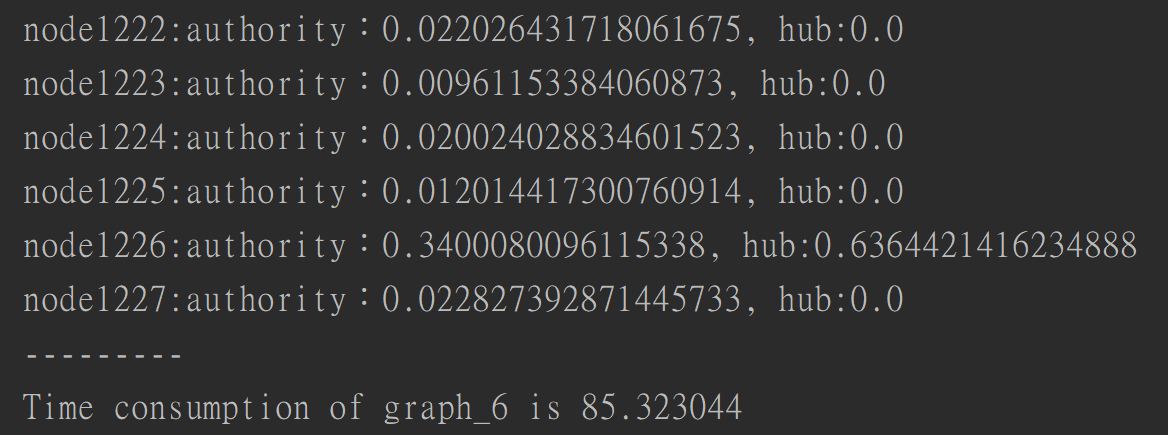
graph 3是一條從node1到node4的雙向路徑圖，有兩個link-in和兩個link-out，所以1和4的分數一樣而2和3的分數也一樣，相對來講2和3的分數會略高。

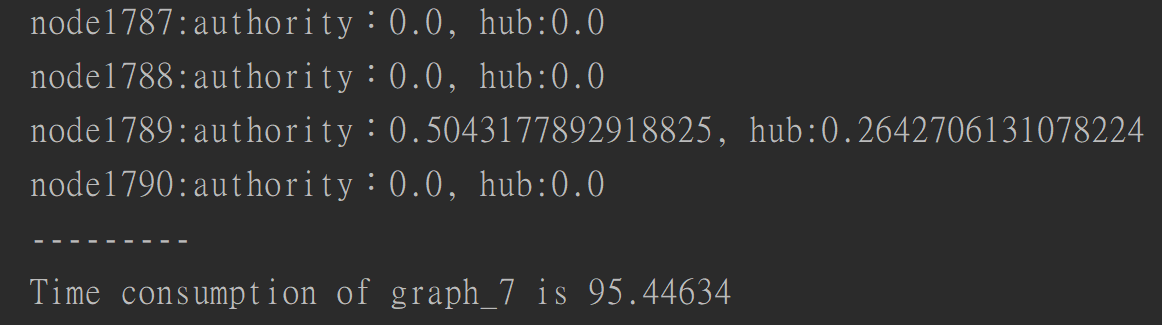


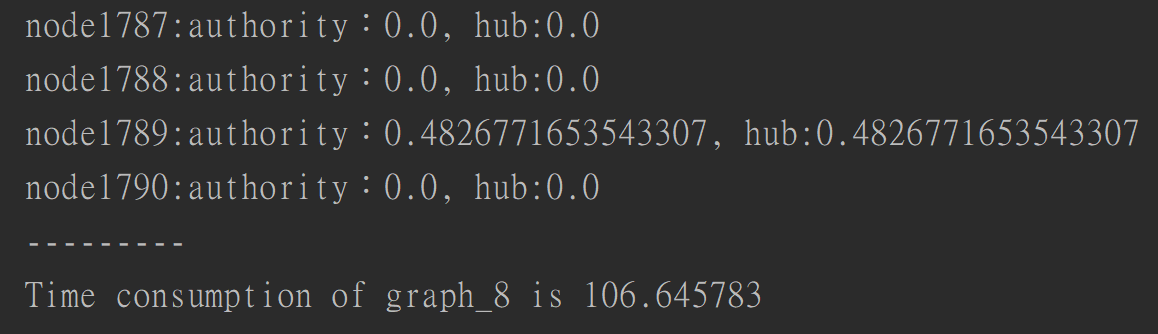
graph 4中node1和5均被四個節點所指向，隨著點變多相應的計算時間也有所上升。

因為graph5，6，7和8的點過多，故結果較為龐大，這裡只展示部分結果。



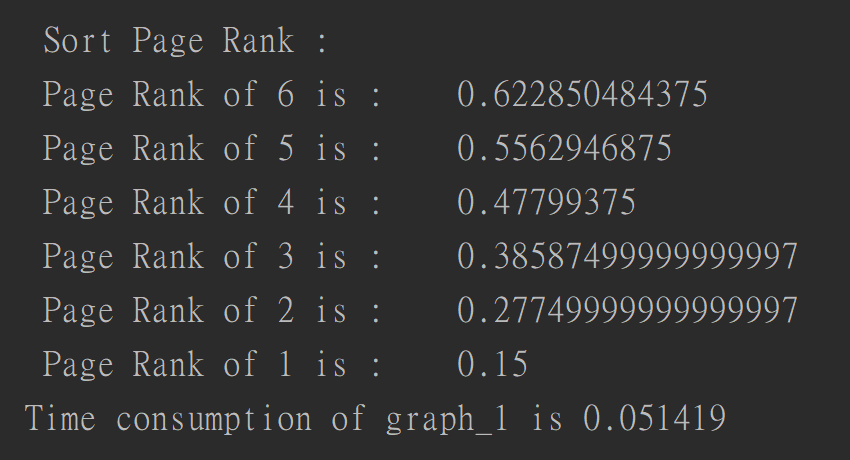


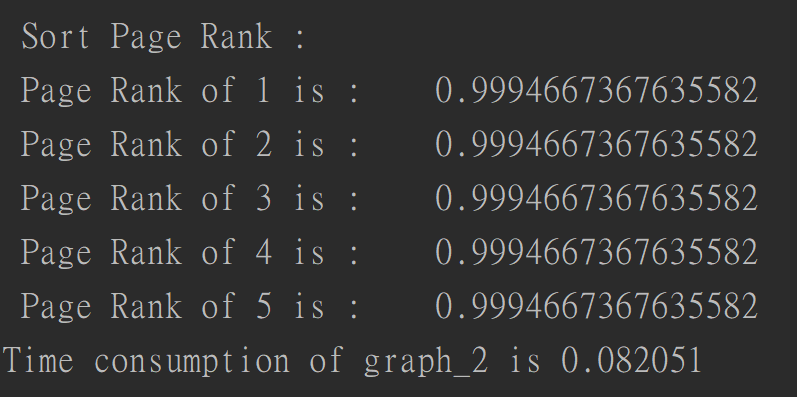




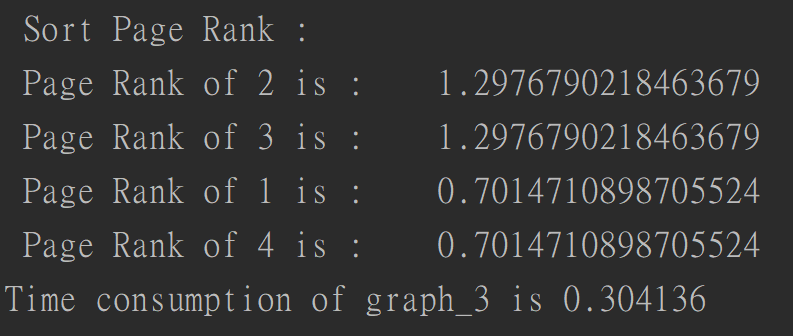
1. **PageRank**

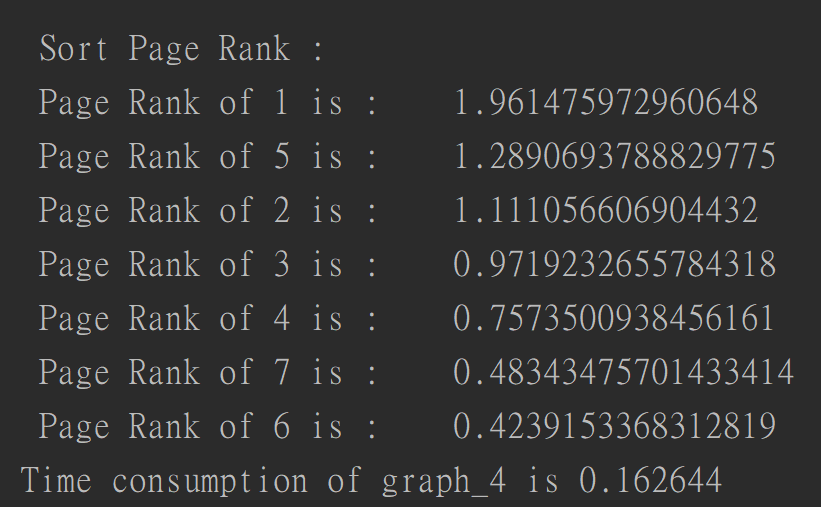
此處展示的結果包括排過序的page rank以及完成整個迭代運算的時間。



從排序過後的結果可以看出從node6到node1的pr逐漸下降。

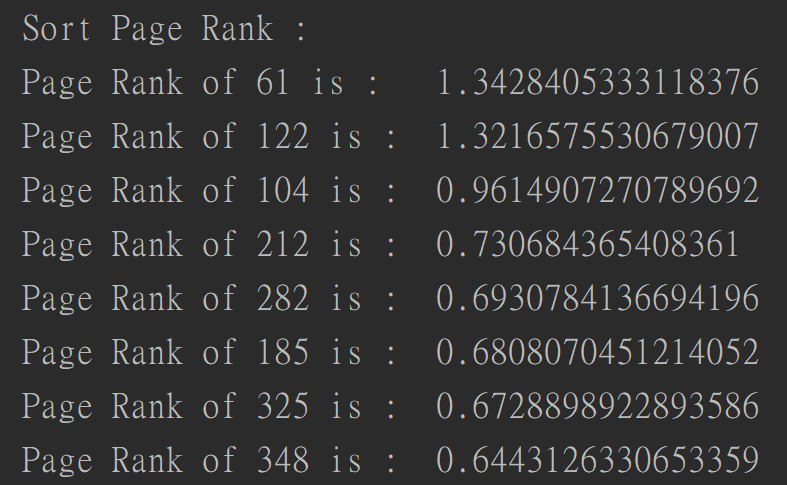
從graph 2的結果可以看出從node 1到node 5，相應的pr值逐漸下降。



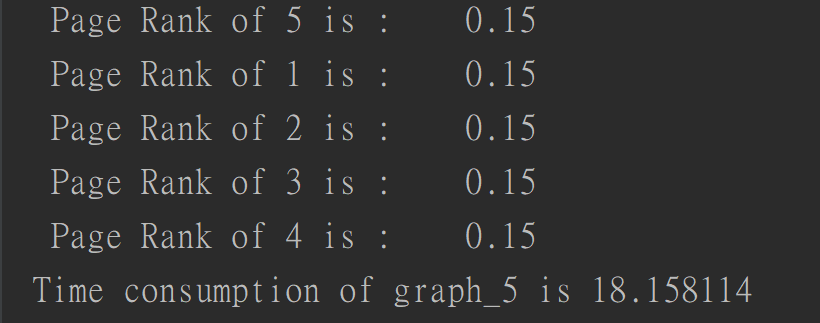
從graph 3的結果可以看出in-degree的值越高，則相應的pr值也越高。

從graph 4的結果可以看出in-degree越高則pr越高

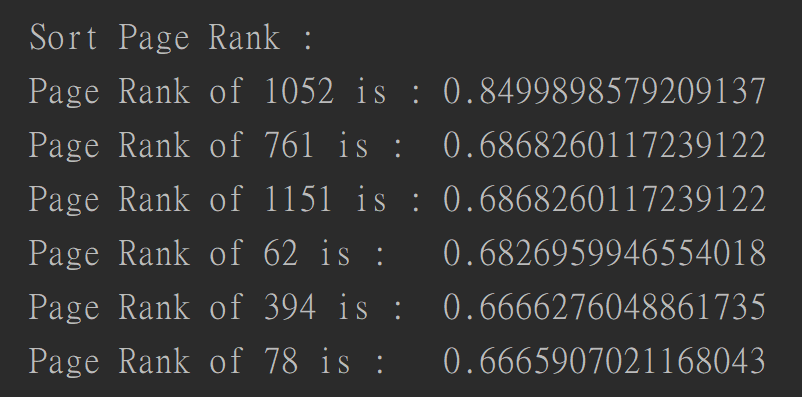
由於graph 5,6和7的input較長，故只放部分結果

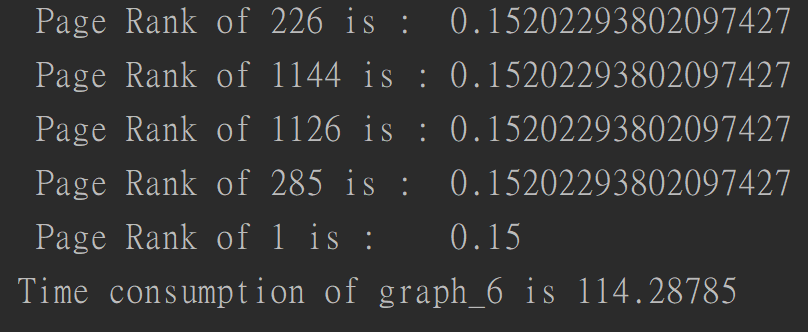


**…**

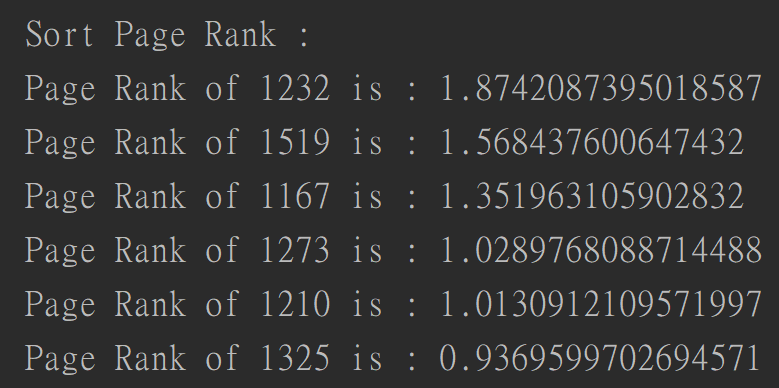


從graph 5的結果可以看出in-degree越高則pr值越高

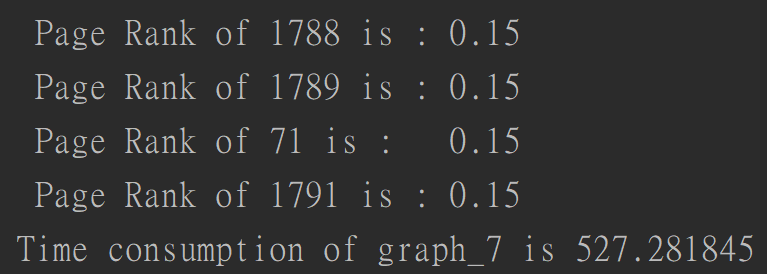
**…**



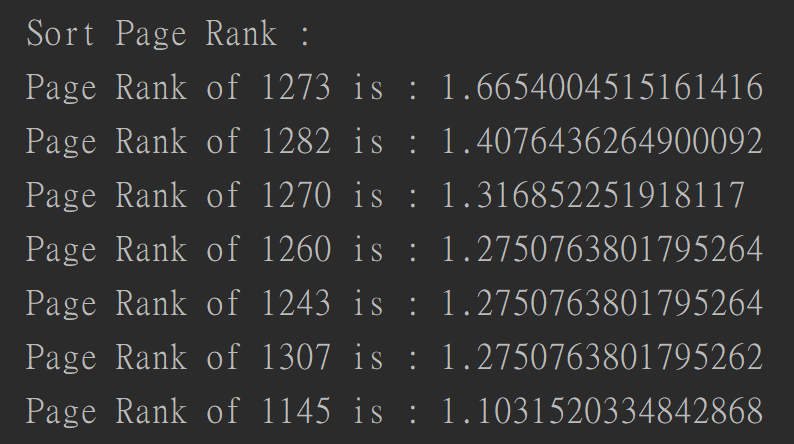
從graph 6的結果可以看出node的排名和in-degree的值有關，相應的in-degree越高則結果的pr值也越高。



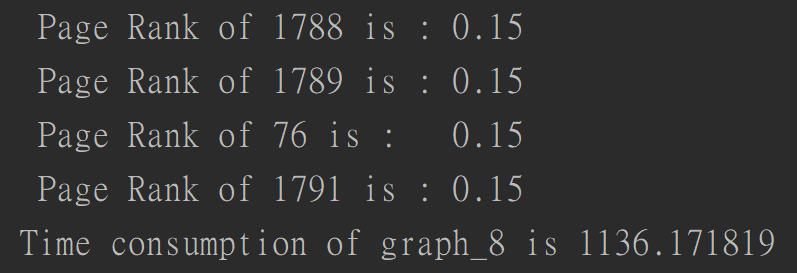
**…**



各個節點間的in-degree呈現一種中間高兩邊低的情況，主要原因應該是input graph是隨機產生的商品id且較多分佈在1200左右，所以才會造成上述現象。



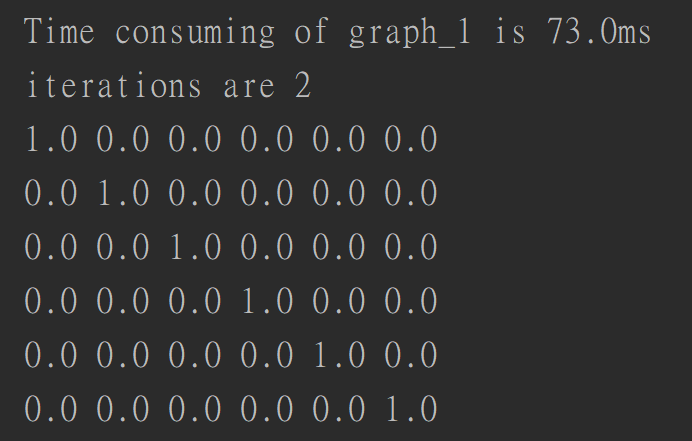
**…**



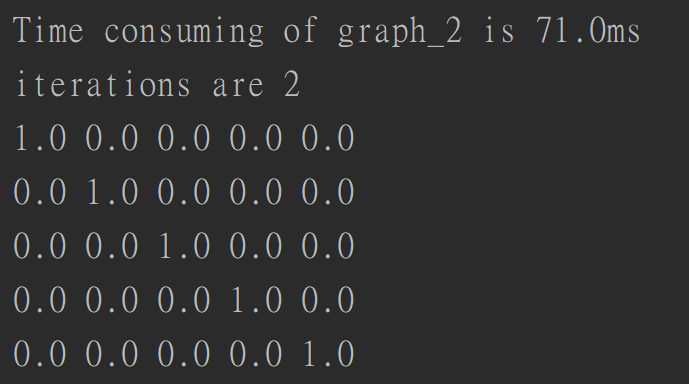
因為是根據隨機產生的商品id構造的graph且雙向，所以點間的in-degree分佈並不均勻。

1. **SimRank**

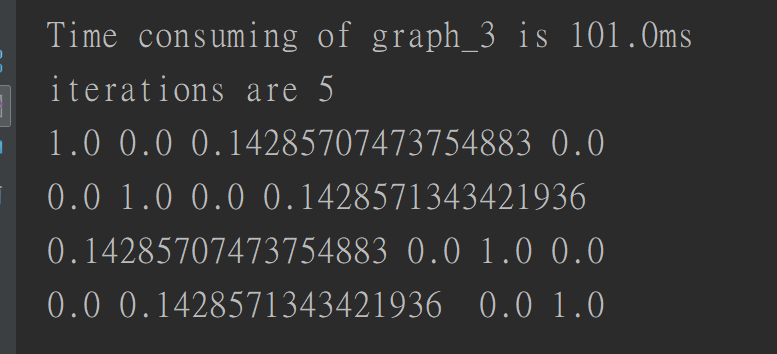
此處展示的結果包括時間花費，迭代次數，以及simrank結果的矩陣。



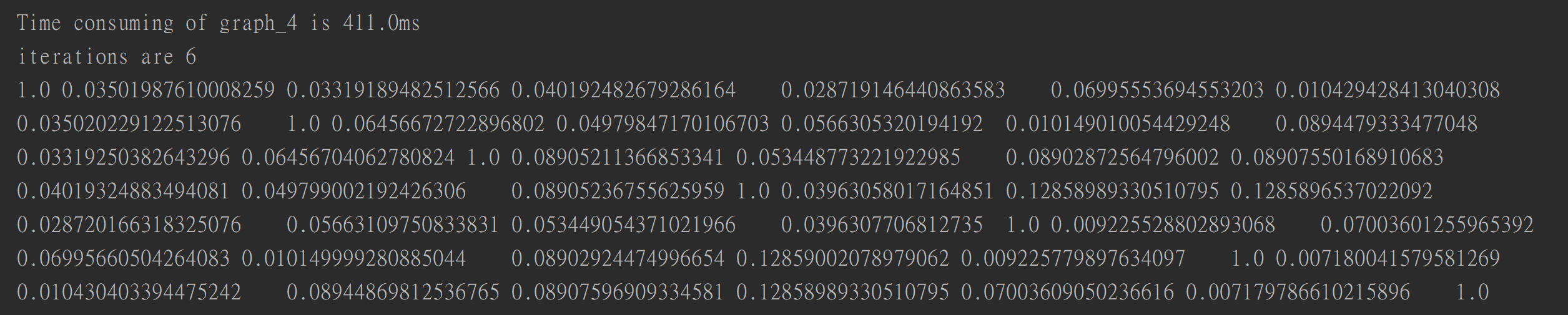
因為圖中並沒有兩個相同的in-link，故只有當a=b的時候其結果為1，剩下則均為0.



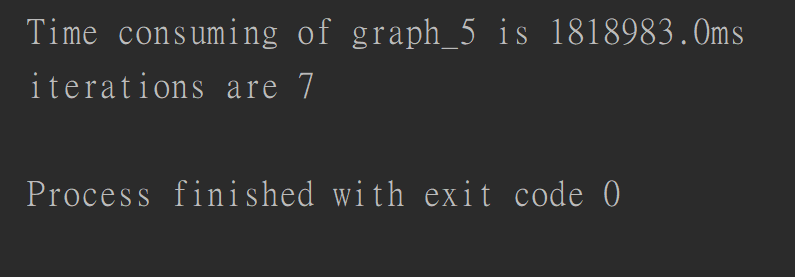
因為graph 2是一張從1到6再回到1的單向環狀圖，不同的節點間沒有被同樣的節點指向，故只有再a=b的時候其值為1



graph 4是一條從1到4的雙向圖，從圖中可以看出若兩個節點之間有in-link則其值不為0，否則則是0



由於graph 5對應的SimRank矩陣過大（此處不能全部顯示），故此處結果只包含時間。相比前面的graph，這張graph中邊增加的很多，很多節點之間都有相互影響力



1. **結果分析**

為了衡量各種input下的性能，我有比較各種input相對應耗費的時間。橫軸表示第幾張graph而縱軸則是相應的時間花費（以ms為單位）。從下面的圖中可以看出三種演算法的耗時逐漸增長，隨著graph的複雜性變高，其相應的耗費時間也會特別上升。以SimRank為例，當圖的複雜性從graph1逐漸到graph6的時候，時間花費會爆炸多。

1. **總結**
2. Find a way (e.g., add/delete some links) to increase hub, authority, and PageRank of Node 1 in first 3 graphs respectively.

關於authority：選取hub值高的node link到node1即可

關於hub：增加Node 1 link到其它的node，若所連的node的authority較高則其值增長較快。

提高pagerank：增加其它node到node1的連接。

1. More limitations about link analysis algorithms

本次實作的三種算法都很容易被作弊者操縱，從而導致垃圾網頁的上升，另外就是推薦的網頁可能包含跟主題不相關的網頁，從而導致結果漂移。

1. Can link analysis algorithms really find the “important”pages from Web?

不一定，因為只要有大量的其它重要網絡對某個網頁進行連接就可以很容易的找到該網頁，通過廣告等行為手段可以增加相應的連接，也就是說根據現有的link analysis算法，再爛的網頁都有可能因為連接的次數和連接網頁的重要程度而出現在用戶需要的首個推薦頁面。

1. What are practical issues when implement these

algorithms in a real Web?

隨著科技的發展，大量的信息湧入網絡，link analysis algorithms可以幫助人們獲取較有用的訊息，但是由於其大量的計算需求，導致對相應server的computation power要求較高。作為企業來講這就意味著成本的上升，將其應用到分佈式的環境中，通過多台性能較弱的機器一起運作可以較好地完成計算需求且成本較少。

1. Performance discussion (time cost)

通過比較上述的三種演算法可以發現hits的用時最少，而SinRank則相對較多。這主要跟計算方式有關，不同的演算法導致相應的實現也不一樣，多輪的迭代和矩陣計算勢必會造成計算的複雜。

1. What do the result say for your actor/movie graph?

所給的input中好像沒有actor/movie graph

1. Any new idea about the link analysis algorithm?

或者可以仿照電影打分的方式，讓用戶對網頁進行打分，相應的link analysis algorithm可以通過相應的分數進一步綜合考慮并推薦效能較高的網頁。

1. What is the effect of “C” parameter in SimRank?

C是一個阻尼係數，通常介於（0,1）之間，假如I(a)=I(b)={A}，按照公式计算出sim(a,b)=C\*sim(A,A)=C

1. Design a new link-based similarity measurement

仿照hits，若兩條link相連的點相似，則link就類似。

1. **展望**

在本次作業中我有實作了三種link analysis的演算法，以前周圍有朋友在做社交網絡，對馬爾科夫鏈略有耳聞，一直有興趣，這次作業讓我對相應的算法的理解有所加深。以前玩mapreduce的時候有稍微碰過pagerank，在巨量資料需要處理的時候會很有效。