

Data Mining

Project3 Link Analysis

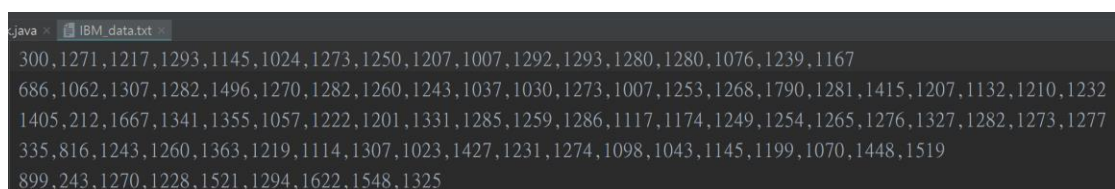
孫啟慧

P78063033

一、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 PageRank/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 對於內存有所限制，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，並按照 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：



```
java IBM_data.txt
300,1271,1217,1293,1145,1024,1273,1250,1207,1007,1292,1293,1280,1280,1076,1239,1167
686,1062,1307,1282,1496,1270,1282,1260,1243,1037,1030,1273,1007,1253,1268,1790,1281,1415,1207,1132,1210,1232
1405,212,1667,1341,1355,1057,1222,1201,1331,1285,1259,1286,1117,1174,1249,1254,1265,1276,1327,1282,1273,1277
335,816,1243,1260,1363,1219,1114,1307,1023,1427,1231,1274,1098,1043,1145,1199,1070,1448,1519
899,243,1270,1228,1521,1294,1622,1548,1325
```

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。處理後的數據有兩欄，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。

graph7.txt	graph8.txt
1 300,1271	1 300,1271
2 300,1217	2 1271,300
3 300,1293	3 300,1217
4 300,1145	4 1217,300
5 300,1024	5 300,1293
6 300,1273	6 1293,300
7 300,1250	7 300,1145
8 300,1207	8 1145,300
9 300,1007	9 300,1024
10 300,1292	10 1024,300
11 300,1293	11 300,1273
12 300,1280	12 1273,300
13 300,1280	13 300,1250
14 300,1076	14 1250,300
15 300,1239	15 300,1207
16 300,1167	16 1207,300
17 1271,1217	17 300,1007
18 1271,1293	18 1007,300
19 1271,1145	19 300,1292
20 1271,1024	20 1292,300

二、 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。

I. HITS

根據網頁的入度和出度來衡量一個網頁的重要性，其中入度指的是指向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，並將結果進行顯示。HitsTest 則是用來傳遞 input graph 進行 test 的檔案。

II. PageRank

主要利用網頁間的連接性對網頁進行排序並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 $n \times n$ 維的 Adjacent Matrix，其中如果 node a 有邊指向 node b 則 $\text{Matrix}[a][b]=1$ 否則等於零。

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

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

$$D(\text{damping factor}) = 0.1 \sim 0.15$$
$$n = |\text{page set}|$$

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 iterator 並將結果按照 rank 進行 sort 并輸出。

III. SimRank

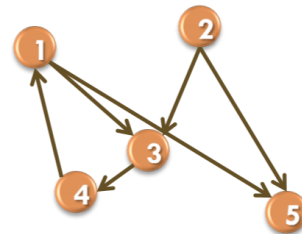
該演算法基於圖論，假設用於推薦的時候，則將用戶和物品分成一張二部圖，一邊是用戶而另外一邊是物品，圖中任意一條邊的兩個端點分別來自兩邊，且兩邊內部的點並沒有連接。

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。

□ SimRank formula

$$S(a,b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} S(I_i(a), I_j(b))$$

- $I(a), I(b)$: all in-neighbors
- C is decay factor, $0 < C < 1$
- $S(a, b) \in [0, 1]$
- $S(a, a) = 1$



1'st iteration
 $S(3, 5) = C/4 * 2$
 $S(4, 5) = 0$

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。

三、 實驗結果

各種演算法的不同 input graph 下的計算結果

I. 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。

```
node0:authority : 0.0, hub:1.0
node1:authority : 1.0, hub:1.0
node2:authority : 1.0, hub:1.0
node3:authority : 1.0, hub:1.0
node4:authority : 1.0, hub:1.0
node5:authority : 1.0, hub:0.0
-----
Time consumption of graph_1 is 36.099614
```

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。

```
Normalization
node0:authority : 1.0, hub:1.0
node1:authority : 1.0, hub:1.0
node2:authority : 1.0, hub:1.0
node3:authority : 1.0, hub:1.0
node4:authority : 1.0, hub:1.0
-----
Time consumption of graph_2 is 33.10201
```

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。

```
node0:authority : 0.6, hub:0.6
node1:authority : 1.0, hub:1.0
node2:authority : 1.0, hub:1.0
node3:authority : 0.6, hub:0.6
-----
Time consumption of graph_3 is 30.422765
```

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。

```
node0:authority : 0.8666666666666667, hub:1.0
node1:authority : 0.8666666666666667, hub:0.2777777777777778
node2:authority : 1.0, hub:0.5
node3:authority : 0.7333333333333333, hub:0.7222222222222222
node4:authority : 1.0, hub:0.7777777777777778
node5:authority : 0.3333333333333333, hub:0.5555555555555556
node6:authority : 0.4, hub:0.2777777777777778
-----
Time consumption of graph_4 is 35.348025
```

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。


```
node463:authority : 0.027472527472527472, hub:0.0
node464:authority : 0.03663003663003663, hub:0.0
node465:authority : 0.02564102564102564, hub:0.0
node466:authority : 0.016483516483516484, hub:0.0
node467:authority : 0.1575091575091575, hub:0.29213483146067415
node468:authority : 0.005494505494505495, hub:0.0
-----
Time consumption of graph_5 is 80.58085
```

```
node1222:authority : 0.022026431718061675, hub:0.0
node1223:authority : 0.00961153384060873, hub:0.0
node1224:authority : 0.020024028834601523, hub:0.0
node1225:authority : 0.012014417300760914, hub:0.0
node1226:authority : 0.3400080096115338, hub:0.6364421416234888
node1227:authority : 0.022827392871445733, hub:0.0
-----
Time consumption of graph_6 is 85.323044
```

```
node1787:authority : 0.0, hub:0.0
node1788:authority : 0.0, hub:0.0
node1789:authority : 0.5043177892918825, hub:0.2642706131078224
node1790:authority : 0.0, hub:0.0
-----
Time consumption of graph_7 is 95.44634
```

```
node1787:authority : 0.0, hub:0.0
node1788:authority : 0.0, hub:0.0
node1789:authority : 0.4826771653543307, hub:0.4826771653543307
node1790:authority : 0.0, hub:0.0
-----
Time consumption of graph_8 is 106.645783
```

II. 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。

```
Sort Page Rank :  
Page Rank of 6 is : 0.622850484375  
Page Rank of 5 is : 0.5562946875  
Page Rank of 4 is : 0.47799375  
Page Rank of 3 is : 0.38587499999999997  
Page Rank of 2 is : 0.27749999999999997  
Page Rank of 1 is : 0.15  
Time consumption of graph_1 is 0.051419
```

從排序過後的結果可以看出從 node6 到 node1 的 pr 逐漸下降。

```
Sort Page Rank :  
Page Rank of 1 is : 0.9994667367635582  
Page Rank of 2 is : 0.9994667367635582  
Page Rank of 3 is : 0.9994667367635582  
Page Rank of 4 is : 0.9994667367635582  
Page Rank of 5 is : 0.9994667367635582  
Time consumption of graph_2 is 0.082051
```

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。

```
Sort Page Rank :  
Page Rank of 2 is :    1.2976790218463679  
Page Rank of 3 is :    1.2976790218463679  
Page Rank of 1 is :    0.7014710898705524  
Page Rank of 4 is :    0.7014710898705524  
Time consumption of graph_3 is 0.304136
```

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。

```
Sort Page Rank :  
Page Rank of 1 is :    1.961475972960648  
Page Rank of 5 is :    1.2890693788829775  
Page Rank of 2 is :    1.111056606904432  
Page Rank of 3 is :    0.9719232655784318  
Page Rank of 4 is :    0.7573500938456161  
Page Rank of 7 is :    0.48343475701433414  
Page Rank of 6 is :    0.4239153368312819  
Time consumption of graph_4 is 0.162644
```

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

```
Sort Page Rank :
```

```
Page Rank of 61 is : 1.3428405333118376
```

```
Page Rank of 122 is : 1.3216575530679007
```

```
Page Rank of 104 is : 0.9614907270789692
```

```
Page Rank of 212 is : 0.730684365408361
```

```
Page Rank of 282 is : 0.6930784136694196
```

```
Page Rank of 185 is : 0.6808070451214052
```

```
Page Rank of 325 is : 0.6728898922893586
```

```
Page Rank of 348 is : 0.6443126330653359
```

```
...
```

```
Page Rank of 5 is : 0.15
```

```
Page Rank of 1 is : 0.15
```

```
Page Rank of 2 is : 0.15
```

```
Page Rank of 3 is : 0.15
```

```
Page Rank of 4 is : 0.15
```

```
Time consumption of graph_5 is 18.158114
```

從 graph 5 的結果可以看出 in-degree 越高則 pr 值越高

```
Sort Page Rank :  
Page Rank of 1052 is : 0.8499898579209137  
Page Rank of 761 is : 0.6868260117239122  
Page Rank of 1151 is : 0.6868260117239122  
Page Rank of 62 is : 0.6826959946554018  
Page Rank of 394 is : 0.6666276048861735  
Page Rank of 78 is : 0.6665907021168043
```

...

```
Page Rank of 226 is : 0.15202293802097427  
Page Rank of 1144 is : 0.15202293802097427  
Page Rank of 1126 is : 0.15202293802097427  
Page Rank of 285 is : 0.15202293802097427  
Page Rank of 1 is : 0.15  
Time consumption of graph_6 is 114.28785
```

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。

```
Sort Page Rank :  
Page Rank of 1232 is : 1.8742087395018587  
Page Rank of 1519 is : 1.568437600647432  
Page Rank of 1167 is : 1.351963105902832  
Page Rank of 1273 is : 1.0289768088714488  
Page Rank of 1210 is : 1.0130912109571997  
Page Rank of 1325 is : 0.9369599702694571
```

...

```
Page Rank of 1788 is : 0.15
Page Rank of 1789 is : 0.15
Page Rank of 71 is : 0.15
Page Rank of 1791 is : 0.15
Time consumption of graph_7 is 527.281845
```

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。

```
Sort Page Rank :
Page Rank of 1273 is : 1.6654004515161416
Page Rank of 1282 is : 1.4076436264900092
Page Rank of 1270 is : 1.316852251918117
Page Rank of 1260 is : 1.2750763801795264
Page Rank of 1243 is : 1.2750763801795264
Page Rank of 1307 is : 1.2750763801795262
Page Rank of 1145 is : 1.1031520334842868
```

...

```
Page Rank of 1788 is : 0.15
Page Rank of 1789 is : 0.15
Page Rank of 76 is : 0.15
Page Rank of 1791 is : 0.15
Time consumption of graph_8 is 1136.171819
```

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。

III. 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。

```
Time consuming of graph_1 is 73.0ms
iterations are 2
1.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0
```

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.

```
Time consuming of graph_2 is 71.0ms
iterations are 2
1.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0
0.0 0.0 0.0 0.0 1.0
```

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

```
Time consuming of graph_3 is 101.0ms
iterations are 5
1.0 0.0 0.14285707473754883 0.0
0.0 1.0 0.0 0.1428571343421936
0.14285707473754883 0.0 1.0 0.0
0.0 0.1428571343421936 0.0 1.0
```

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

```
Time consuming of graph_4 is 411.0ms
iterations are 6
1.0 0.03501987610008259 0.03319189482512566 0.040192482679286164 0.028719146440863583 0.06995553694553203 0.010429428413040308
0.035020229122513076 1.0 0.06456672722896802 0.04979847170106703 0.0566305320194192 0.010149010054429248 0.0894479333477048
0.03319250382643296 0.06456704062780824 1.0 0.08905211366853341 0.053448773221922985 0.08902872564796002 0.08907550168910683
0.04019324883494081 0.049799002192426306 0.08905236755625959 1.0 0.03963058017164851 0.12858989330510795 0.1285896537022092
0.028720166318325076 0.05663109750833831 0.053449054371021966 0.0396307706812735 1.0 0.009225528802893068 0.07003601255965392
0.06995660504264083 0.010149999280885044 0.08902924474996654 0.12859002078979062 0.009225779897634097 1.0 0.007180041579581269
0.010430403394475242 0.08944869812536765 0.08907596909334581 0.12858989330510795 0.07003609050236616 0.007179786610215896 1.0
```

由於 graph 5 對應的 SimRank 矩陣過大（此處不能全部顯示），故此

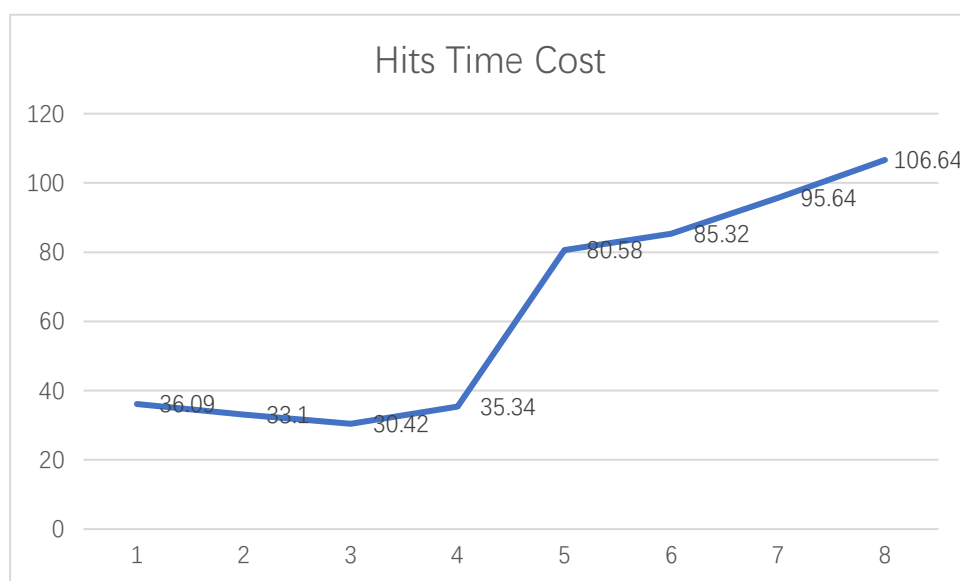
處結果只包含時間。相比前面的 graph，這張 graph 中邊增加的很多，很多節點之間都有相互影響力

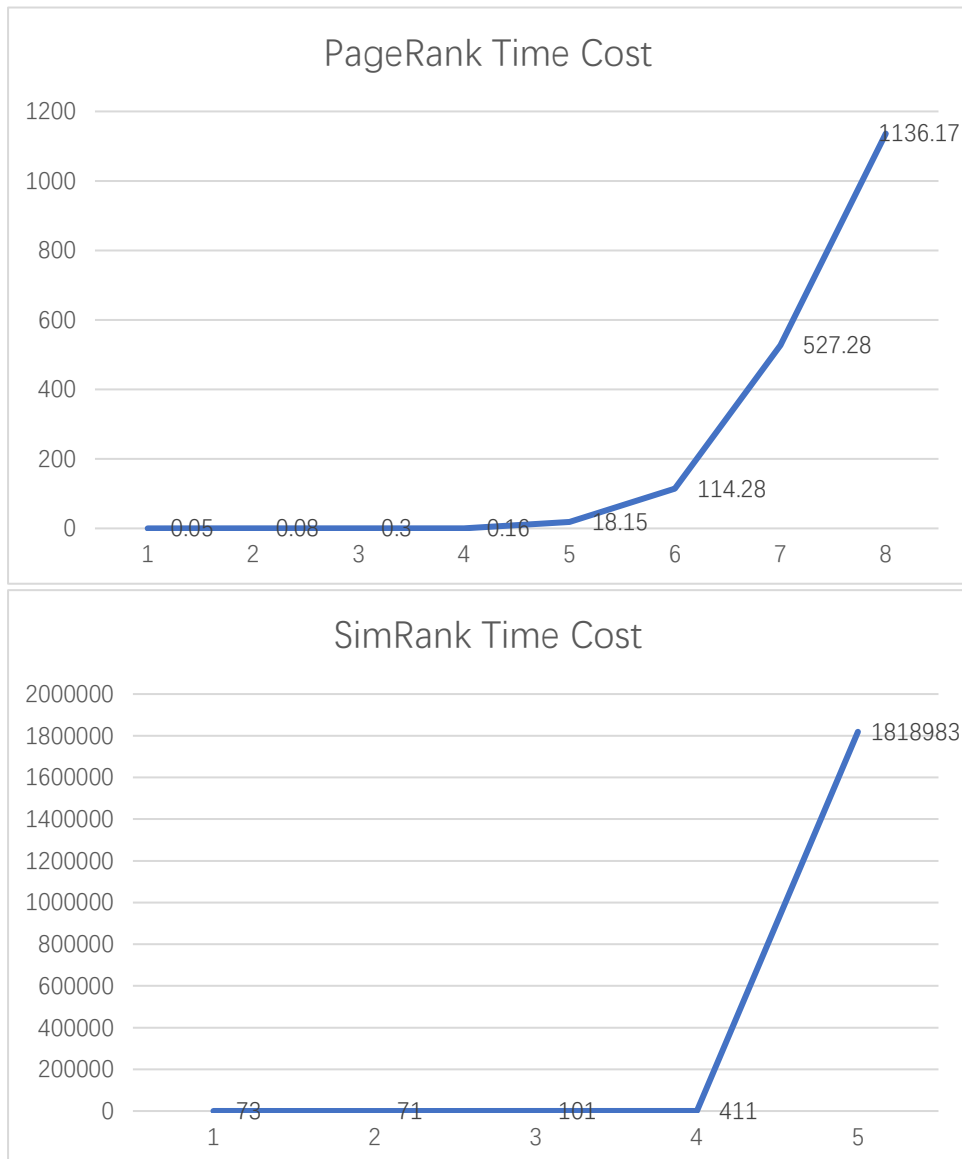
```
Time consuming of graph_5 is 1818983.0ms  
iterations are 7
```

```
Process finished with exit code 0
```

四、 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。





五、總結

1. 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 的连接。

2. 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。

3. 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。

4. 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。

5. 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。

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

所給的 input 中好像沒有 actor/movie graph

7. 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。

8. 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 $I(a)=I(b)=\{A\}$ ，按照公式計算出 $\text{sim}(a, b)=C*\text{sim}(A, A)=C$

9. 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。

六、 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。