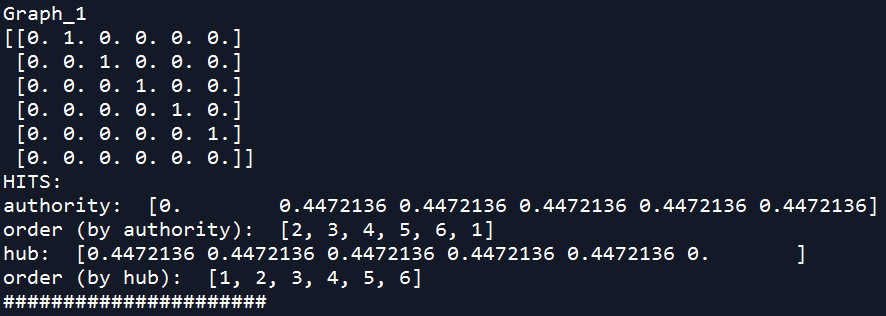
DataMining Project3 林嘉源 P76051072

1. HITS Experiment (Graph 1~8)

CODE: project3\_HITS.py

GRAPH-1



由auth 及 hub 的計算方式: at = At\*ht-1, ht = A\*at-1, A為graph-1 的adjacency matrix

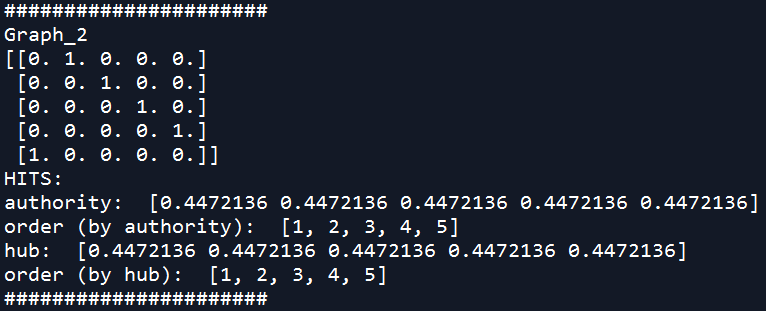
at[i] (向量at的第i個element)正比於A(i) 的和，即adjacency matrix A的第i行向量

ht[i] (向量ht的第i個element)正比於A(i) 的和，即adjacency matrix A的第i列向量

因為A(1) 的和為0，顯然Node-1(以下稱N1)的authority 為0，若以authority的值來做排序: N2 =N3=N4=N5=N6>N1

同理，因為A(6) 的和為0，顯然Node-6的hub 為0，若以hub的值來做排序: N1=N2 =N3=N4=N5>N6

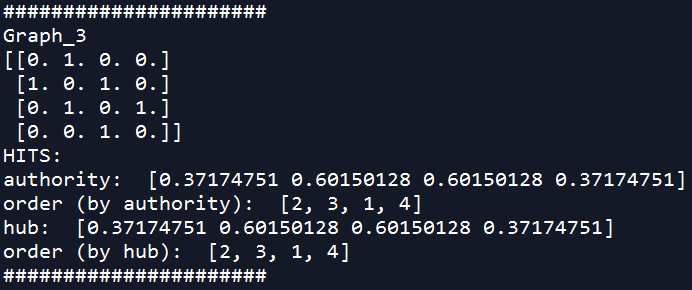
GRAPH-2



因為A各個行向量的和為1，顯然N1~N5的authority均相同，若以authority的值來做排序: N1=N2 =N3=N4=N5

同理，因為A各個列向量的和為1，顯然N1~N5的hub均相同，若以hub的值來做排序: N1=N2 =N3=N4=N5

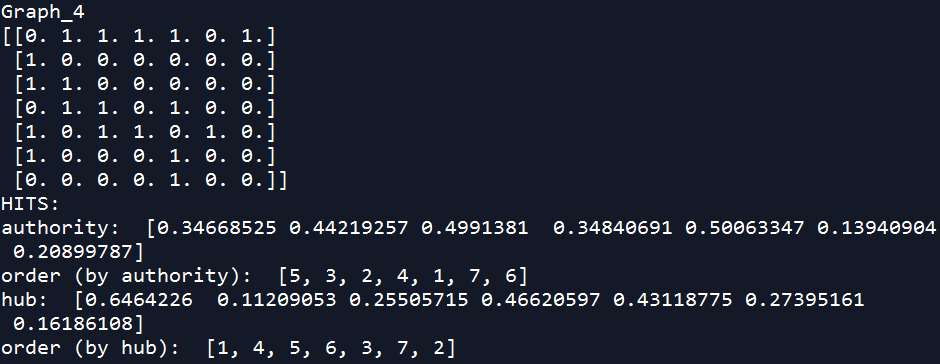
GRAPH-3



因為A(2) ,A(3)行向量的和為2，A(1) ,A(4)行向量的和為1，顯然N2,N3的authority相同，N1,N4的authority相同，且N2會比N1大，若以authority的值來做排序: N2 =N3> N1=N4

同理，因為A(2) ,A(3)列向量的和為2，A(1) ,A(4)列向量的和為1，顯然N2,N3的hub相同，N1,N4的hub相同，且N2會比N1大，若以hub的值來做排序: N2 =N3> N1=N4

GRAPH-4



因為A(5)行向量的和為4(較大)，A(6) 行向量的和為1(較小)，顯然若以authority的值來做排序: N5 > N6

同理，因為A(1)列向量的和為5(較大)，A(2) 列向量的和為1(較小)，顯然若以hub的值來做排序: N1 > N2

GRAPH-5

請run code

GRAPH-6

請run code

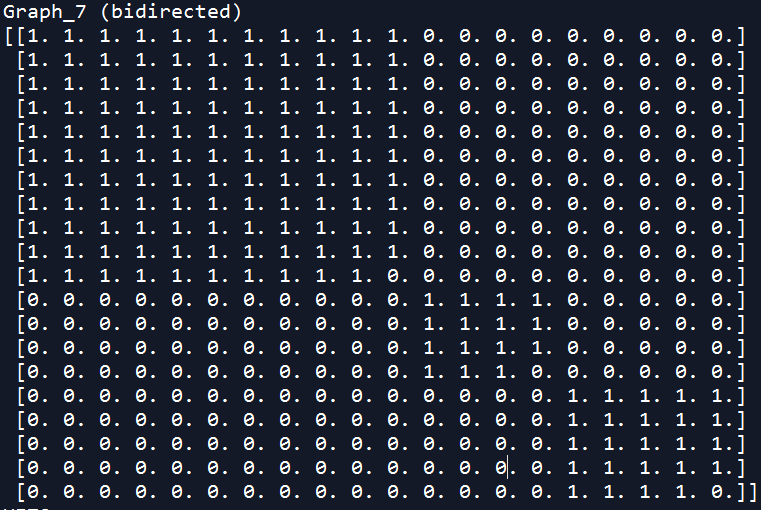
GRAPH-7

DATASET: data.csv

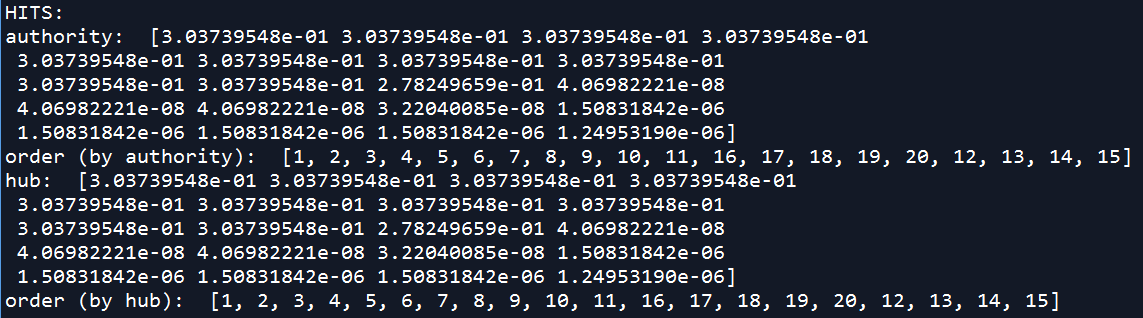
因為檔案內transaction id過大，僅擷取前20筆(size)，並設定為bidirected

輸出g7為adjacency matrix，g7dict為對應item id至unique id的dict

Note: 每個item id (unique id) 對應graph-7內唯一node (vertex)



執行結果



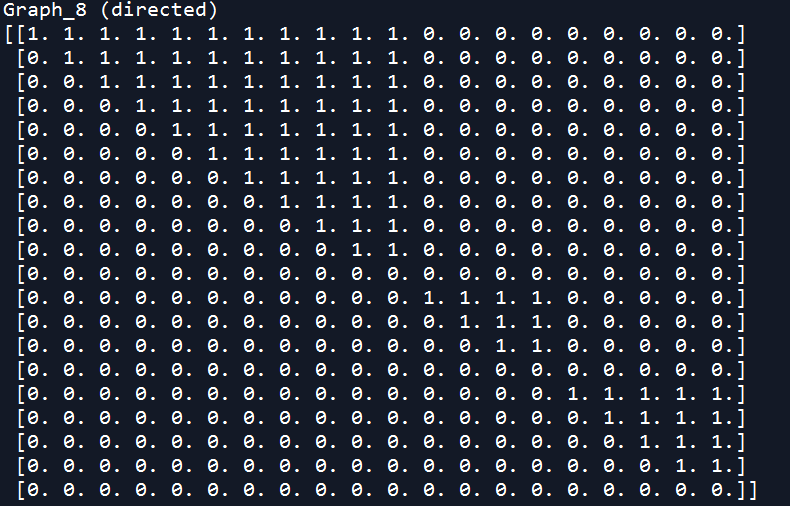
GRAPH-8



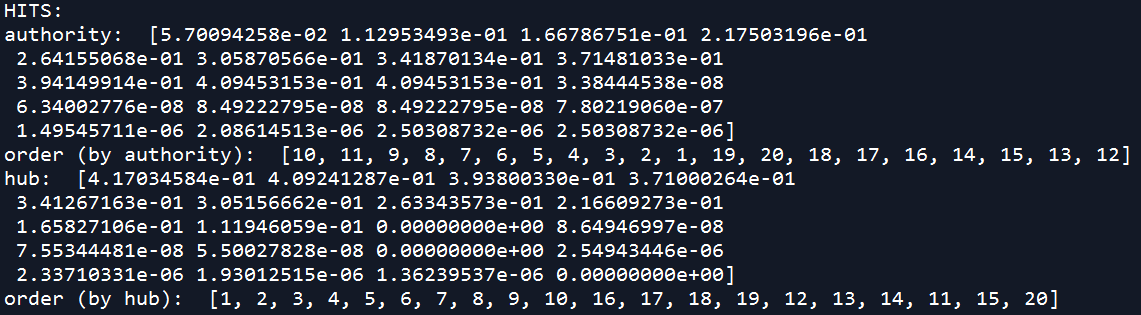
因為檔案內transaction id過大，僅擷取前20筆(size)，並設定為directed

輸出g8為adjacency matrix，g8dict為對應item id至unique id的dict

Note: 每個item id (unique id) 對應graph-8內唯一node (vertex)



執行結果

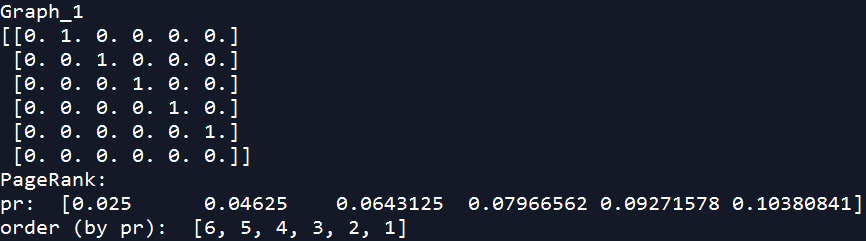


1. PageRank Experiment (Graph 1~8)

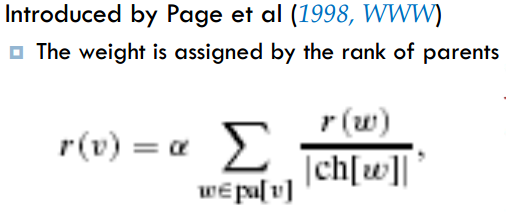
CODE: project3\_PR.py

dumpingFactor = 0.15

GRAPH-1



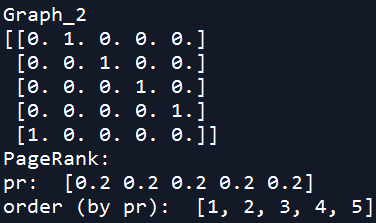
由PageRank演算法



rank值可以由其parent node累積，隨著演算不斷迭代，若某parent node有較大的rank值，則其child node的rank值也會受到影響而變大

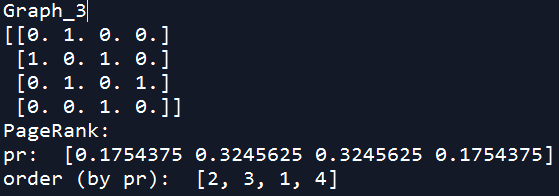
接著觀察graph-1，為由N1->N2->N3->N4->N5>N6的directed graph，顯然由rank值排序N6> N5> N4> N3> N2> N1

GRAPH-2



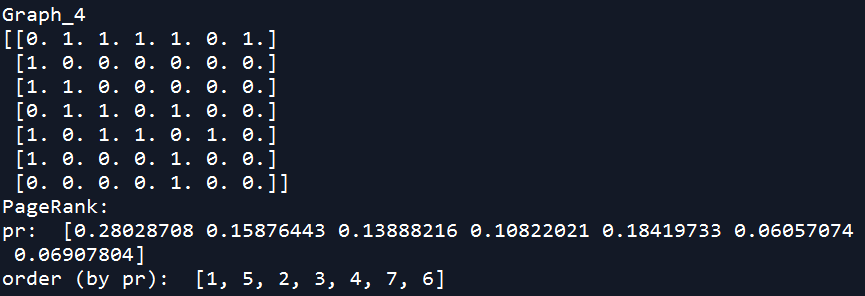
graph-2為directed cycle，N1~N5的rank皆相同

GRAPH-3



graph-3 N2,N3的rank皆比N1,N4大

GRAPH-4



考慮In-degree，N1, N5為4，N2,N3為3，N4為2，N7,N6為1

GRAPH-5

請run code

GRAPH-6

請run code

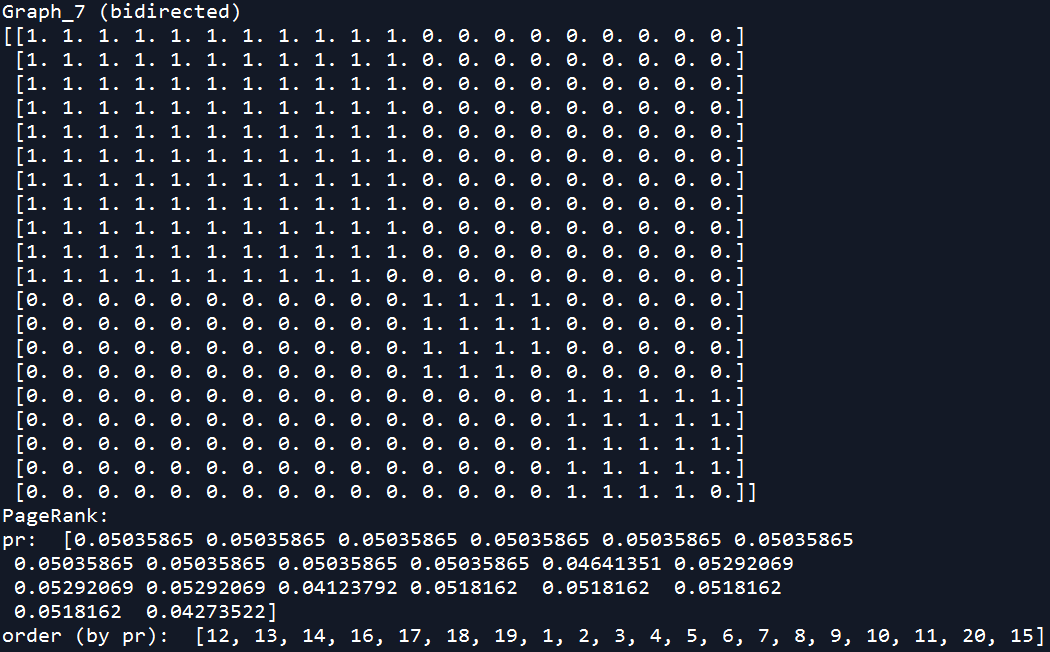
GRAPH-7

DATASET: data.csv

因為檔案內transaction id過大，僅擷取前20筆(size)，並設定為bidirected

輸出g7為adjacency matrix，g7dict為對應item id至unique id的dict

Note: 每個item id (unique id) 對應graph-7內唯一node (vertex)



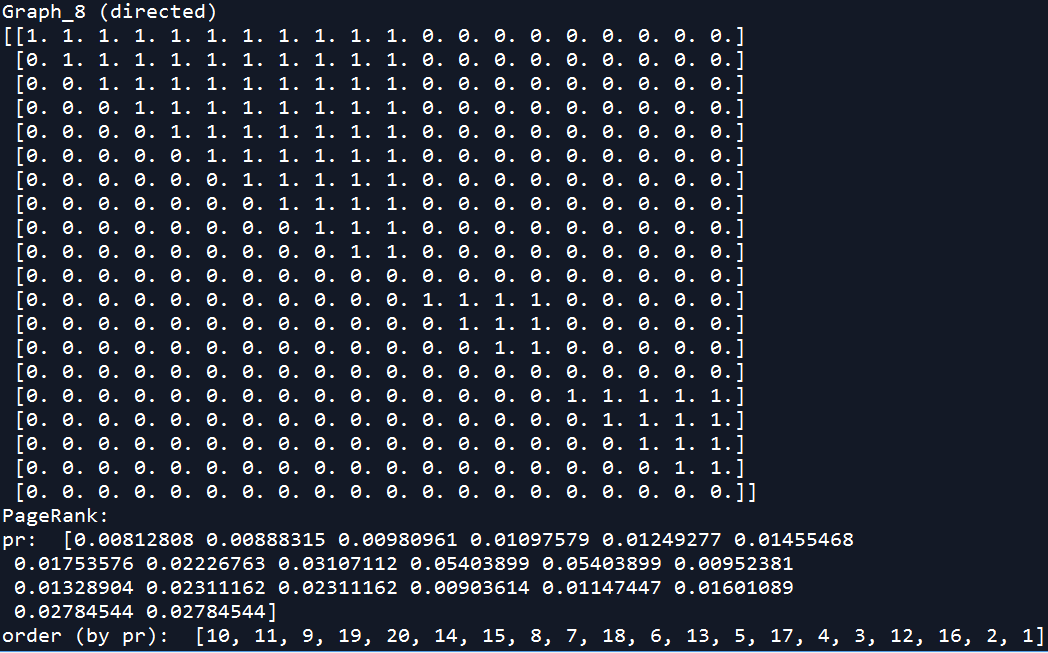
GRAPH-8



因為檔案內transaction id過大，僅擷取前20筆(size)，並設定為directed

輸出g8為adjacency matrix，g8dict為對應item id至unique id的dict

Note: 每個item id (unique id) 對應graph-8內唯一node (vertex)

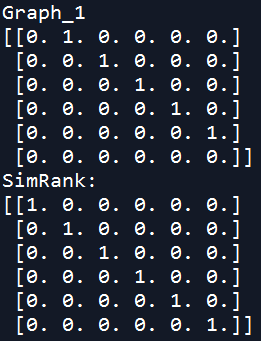


1. SimRank Experiment (Graph 1~5)

CODE: project3\_SimRank.py

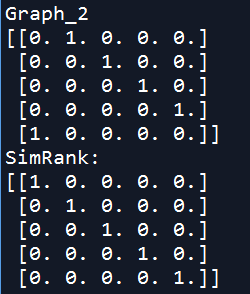
C=0.8

GRAPH-1



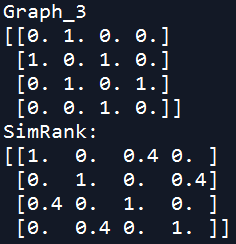
從A (Adajcency Matrix) 來看，每個Node沒有共同的parent node，顯然graph-1的simRank除了主對角線外皆為0

GRAPH-2



從A (Adajcency Matrix) 來看，每個Node沒有共同的parent node，顯然graph-1的simRank除了主對角線外皆為0

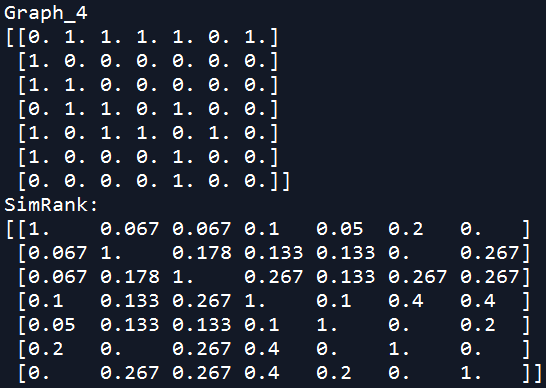
GRAPH-3



從A (Adajcency Matrix) 來看，N1,N3共同的parent node 為N2，N1及N3的in-degree各為1及2 (對應A(1)及A(3)的和)，再加上C = 0.8

故simRank(1,3) = 1/2 \* 0.8 = 0.4

GRAPH-4



從A (Adajcency Matrix) 來看，N1,N2共同的parent node 為N3，N1及N3的in-degree各為4及3 (對應A(1)及A(2)的和)，再加上C = 0.8

故simRank(1,2) = 1/12 \* 0.8 = 0.0667

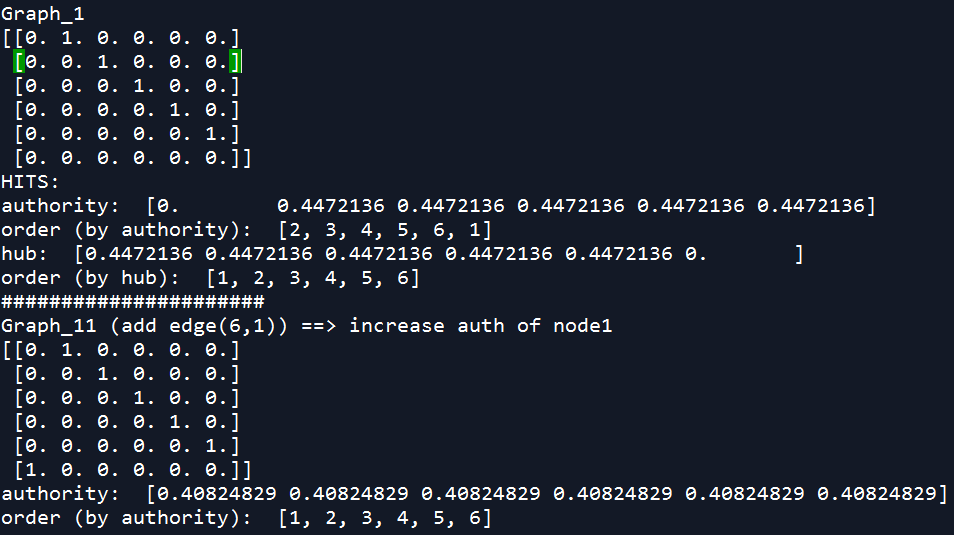
GRAPH-5

請run code

1. Increase Node-1 (hub, authority and PageRank) value

CODE: project3\_HITS\_Miscellaneous.py, project3\_PR\_Miscellaneous.py

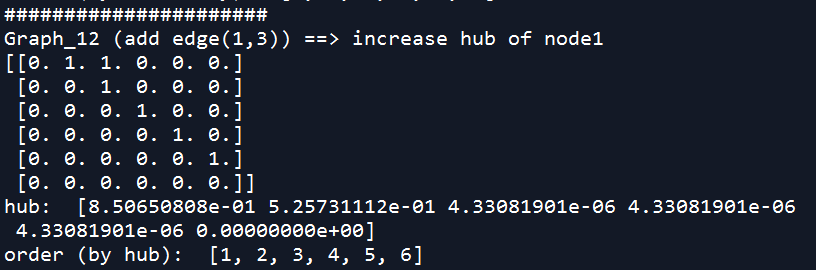
GRAPH-1



**Authority**

要增加N1的authority，要增加N1的indegree，不妨加入edge(6,1)，from=N6, to=N1

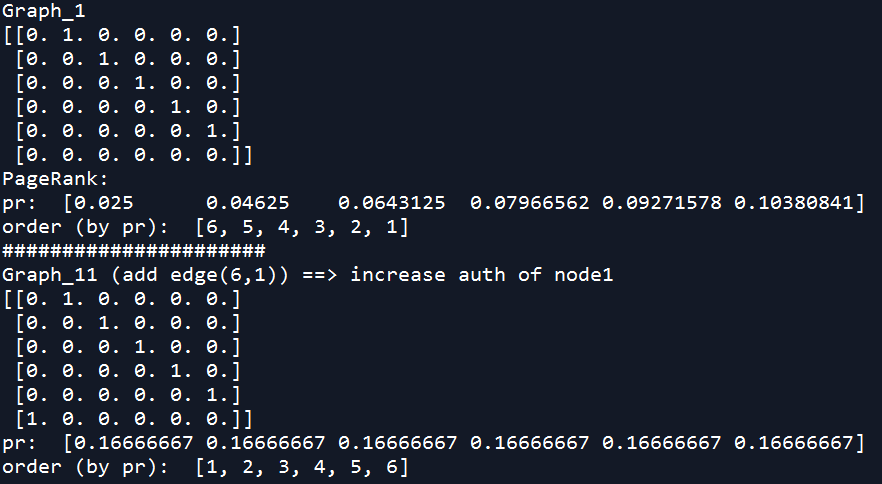
可以將N1的authority從0 🡺0.40824



**Hub**

要增加N1的hub，要增加N1的outdegree，不妨加入edge(1,3)，from=N1, to=N3

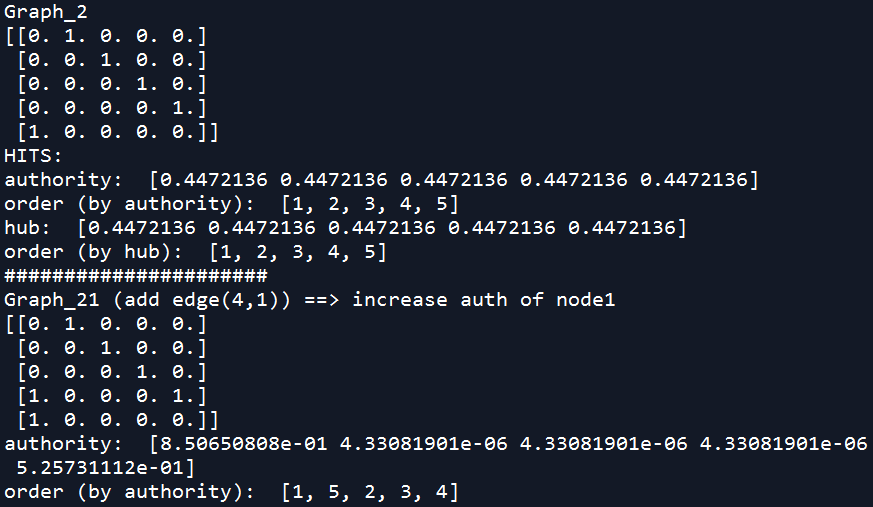
可以將N1的hub從0 🡺0.8506



**PageRank**

要增加N1的rank，要增加N1的indegree，不妨加入edge(6,1)，from=N6, to=N1，可以將N1的pagerank從0.025 🡺0.1667

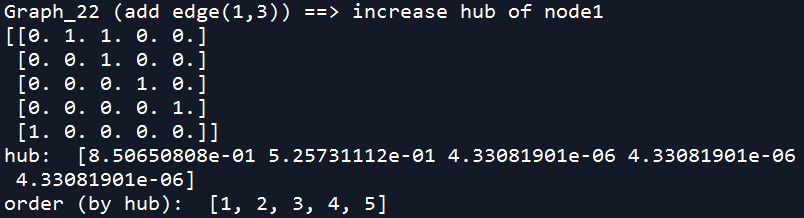
GRAPH-2



**Authority**

要增加N1的authority，要增加N1的indegree，不妨加入edge(4,1)，from=N4, to=N1

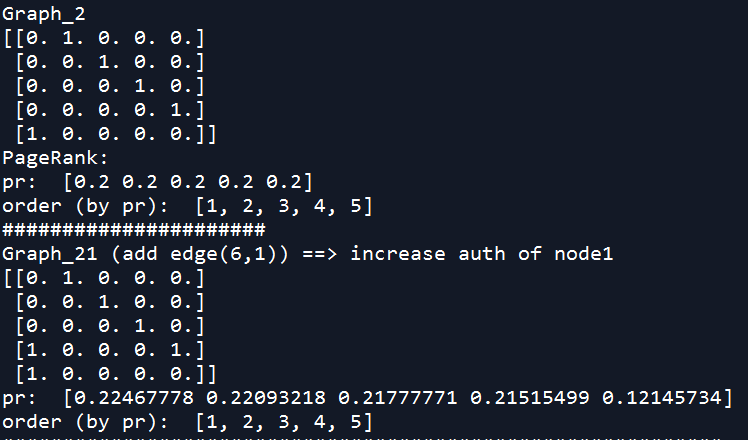
可以將N1的authority從0.4472 🡺0.8506



**Hub**

要增加N1的hub，要增加N1的outdegree，不妨加入edge(1,3)，from=N1, to=N3

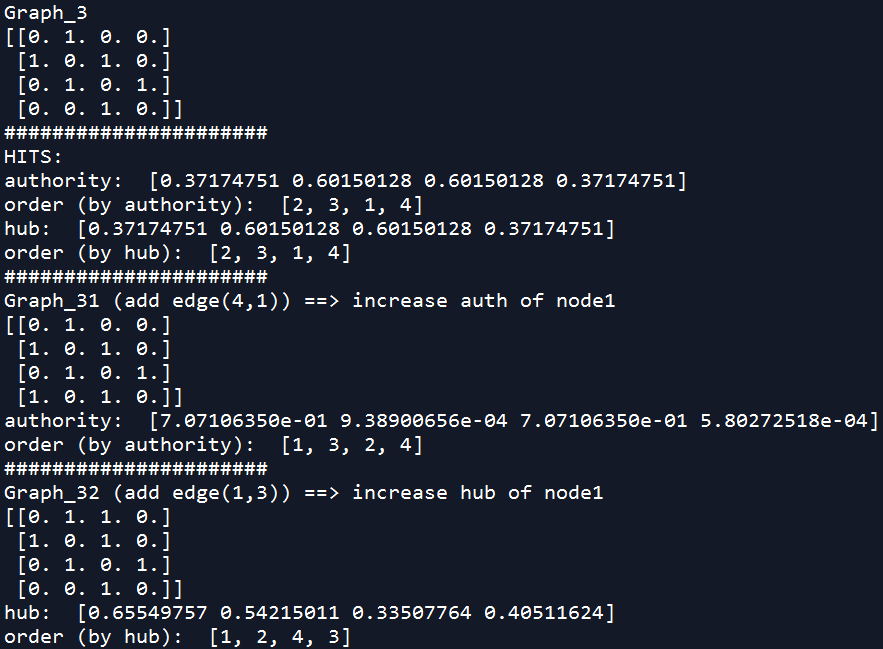
可以將N1的hub從0.4472 🡺0.8506



**PageRank**

要增加N1的rank，要增加N1的indegree，不妨加入edge(4,1)，from=N4, to=N1，可以將N1的pagerank從0.2 🡺0.2246

GRAPH-3



**Authority**

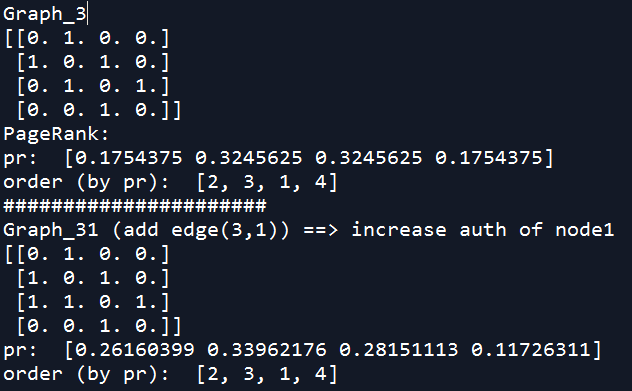
要增加N1的authority，要增加N1的indegree，不妨加入edge(4,1)，from=N4, to=N1

可以將N1的authority從0.3717 🡺0.7071

**Hub**

要增加N1的hub，要增加N1的outdegree，不妨加入edge(1,3)，from=N1, to=N3

可以將N1的hub從0.3717 🡺0.6554



**PageRank**

要增加N1的rank，要增加N1的indegree，不妨加入edge(4,1)，from=N4, to=N1，可以將N1的pagerank從0.1754 🡺0.2616

1. Implementation Detail

CODE: utils.py

HITS

由auth 及 hub 的計算方式: at = At\*ht-1, ht = A\*at-1, A為graph 的adjacency matrix

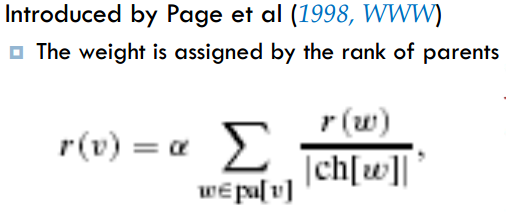
at[i] (向量at的第i個element)正比於A(i) 的和，即adjacency matrix A的第i行向量

ht[i] (向量ht的第i個element)正比於A(i) 的和，即adjacency matrix A的第i列向量

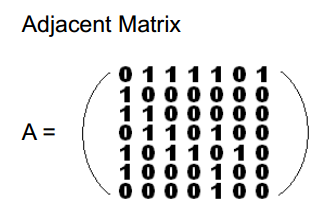
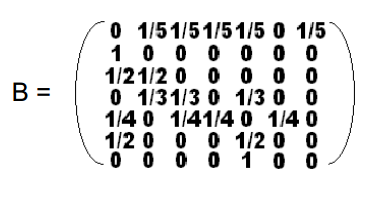
演算法只要起始值a0,h0(單位向量)，不斷透過at = At\*ht-1, ht = A\*at-1來迭代即可求得authority及hub

PageRank

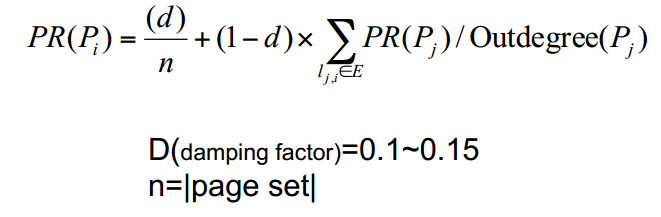
由PageRank演算法



rank值可以由其parent node累積，但累積的加成值與其parent node 的child node的個數(outdegree)成反比，在做矩陣運算時不妨將A的每個列都除以列的和 (A🡺B)

🡺 

接著演算法只要起始值pr(單位向量)，透過以下方式不斷迭代



SimRank

令SimRank Matrix 為S，a,b為Graph G的node，C為decay factor，Adjacency Matrix為A

，令B為將A的每個行除以每個行的和,e.g.

由公式:

由上述公式迭代即可求得S

1. Computation Performance Analysis
2. HITS、PageRank

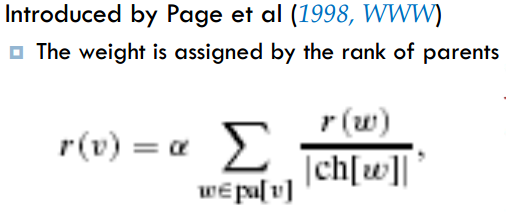
HITS 演算法使用a0,h0(單位向量)，分別與A,At做矩陣乘法迭代

PageRank演算法使用 單位向量，分別與除以列和後的A(即B)做矩陣乘法迭代，兩者的計算複雜度皆與收斂所需的迭代次數成正比

1. SimRank

SimRank演算法使用矩陣間的乘法(BtSB)，每一輪皆須做2次n\*n矩陣的乘法，每次迭代的計算成本較HITS及PageRank高

1. Discussion
2. 就output而言，HITS及PageRank產生每個node的權值，便於做搜尋的排序，而SimRank產生的是Graph內每個node與其他node之間相關性，較適合做個個node之間相似度的比較
3. 由先前討論知PageRank演算法



顯然若v的parent越多，即indegree越高，summation的項次會越多，rank值會越高。舉極端的情形，graph內全部的node都是其parent node，則其rank值會最高。

1. 因為以前沒有寫過python，從這次project中發現很多python語法簡潔的地方，舉例像是做summation，在C++得要用loop來做加總，在攥寫速度上提升不少。