Data Mining 2022 Project3 Grading Policy v3

P76111262 林晨鈞

* Find a way
* Find a way (e.g., add/delete some links) to increase hub, authority, and PageRank of Node 1 in first 3 graphs respectively
  + graph\_1
    - 原本:

auth: {'1': 0.000, '2': 0.200, '3': 0.200, '4': 0.200, '5': 0.200, '6': 0.200}

hub: {'1': 0.200, '2': 0.200, '3': 0.200, '4': 0.200, '5': 0.200, '6': 0.000}

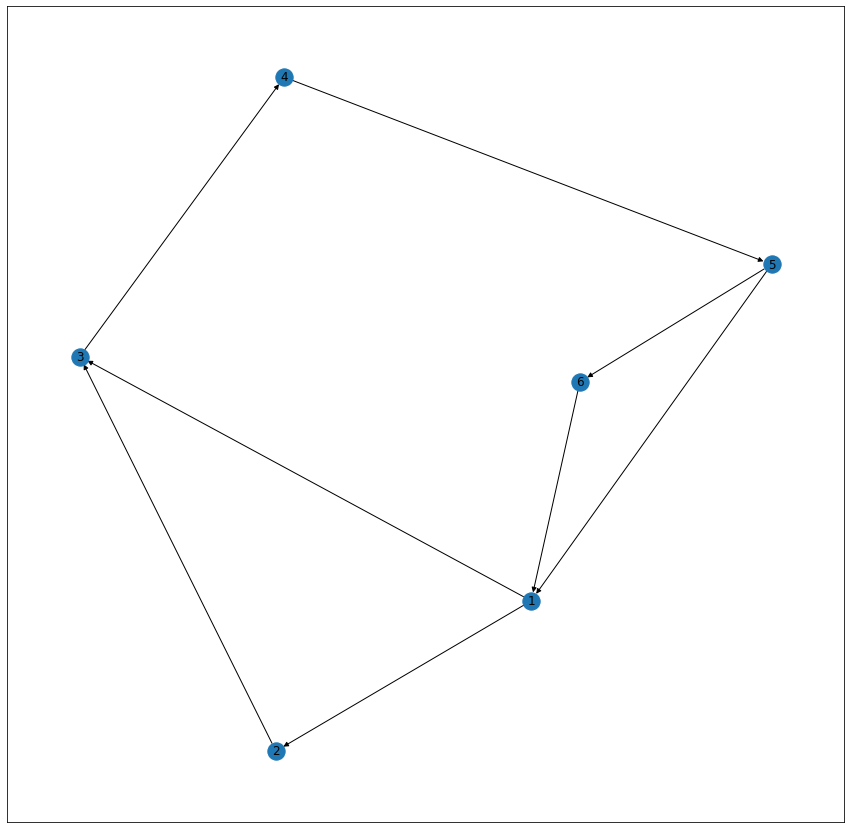
pagerank: {'1': 0.017, '2': 0.032, '3': 0.045, '4': 0.057, '5': 0.068, '6': 0.078}

* 增加(6,1)、(5,1)、(1,3)，可增加Node 1的authority、hub:

auth: {'1': 0.309, '2': 0.191, '3': 0.309, '4': 0.000, '5': 0.000, '6': 0.191}

hub: {'1': 0.309, '2': 0.191, '3': 0.000, '4': 0.000, '5': 0.309, '6': 0.191}

pagerank: {'1': 0.198, '2': 0.106, '3': 0.201, '4': 0.197, '5': 0.194, '6': 0.104}



* + graph\_2
    - 原本:

auth: {'1': 0.200, '2': 0.200, '3': 0.200, '4': 0.200, '5': 0.200,}

hub: {'1': 0.200, '2': 0.200, '3': 0.200, '4': 0.200, '5': 0.200}

pagerank: {'1': 0.200, '2': 0.200, '3': 0.200, '4': 0.200, '5': 0.200}

* 增加(1,3)、(2,3)、(4,1)，可增加Node 1的authority、hub:

auth: {'1': 0.309, '2': 0.191, '3': 0.309, '4': 0.000, '5': 0.191}

hub: {'1': 0.309, '2': 0.191, '3': 0.000, '4': 0.309, '5': 0.191}

pagerank: {'1': 0.247, '2': 0.131, '3': 0.249, '4': 0.244, '5': 0.130}

一張含有 文字, 天空 的圖片

自動產生的描述

* + graph\_3
    - 原本:

auth: {'1': 0.191, '2': 0.309, '3': 0.309, '4': 0.191}

hub: {'1': 0.191, '2': 0.309, '3': 0.309, '4': 0.191}

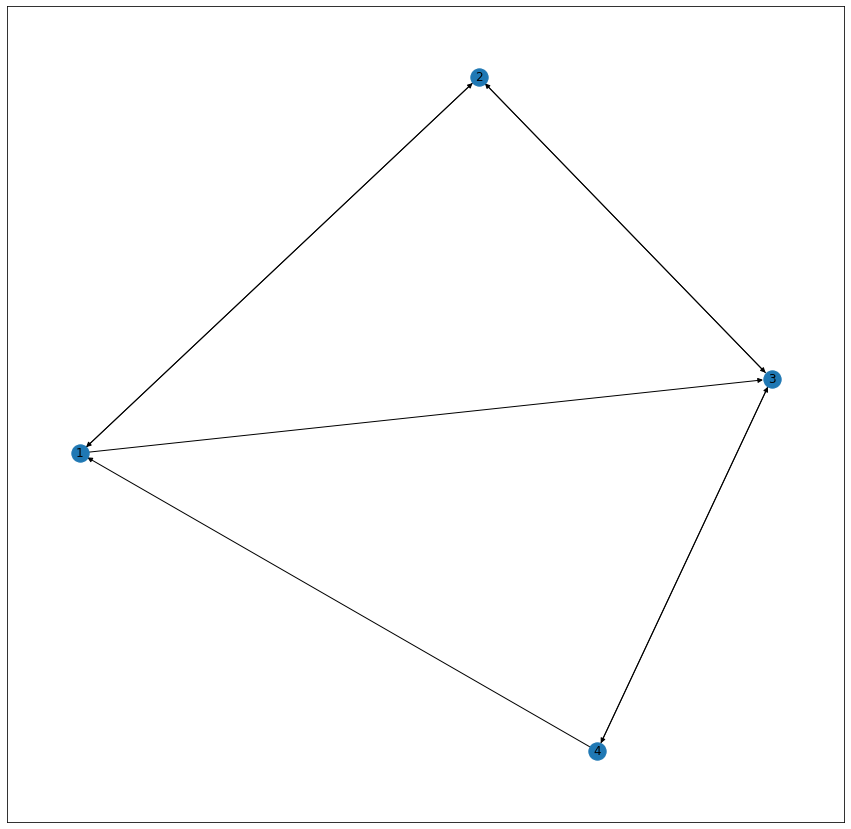
pagerank: {'1': 0.172, '2': 0.328, '3': 0.328, '4': 0.172}

* 增加(1,3)、(2,3)、(4,1)，可增加Node 1的authority、hub:

auth: {'1': 0.322, '2': 0.178, '3': 0.453, '4': 0.047}

hub: {'1': 0.262, '2': 0.322, '3': 0.093, '4': 0.322}

pagerank: {'1': 0.226, '2': 0.274, '3': 0.328, '4': 0.172}



* Algorithm description
* 解釋每個演算法步驟流程
  + HITS(Hypertext Induced Topic Selection)

1. 初始化所有節點的hub和auth值為1
2. 在每個迭代中，更新每個節點的hub和auth值
3. 新的authority值的計算方法是該節點的所有父親(指向自己的節點)節點的hub值的加總
4. 新的hub值的計算方法是該節點的所有兒子(自己指出去指到的點)節點的authority值的加總
5. 對所有新計算出的authority和hub值做正規化

一張含有 文字 的圖片

自動產生的描述

* PageRank
  1. 初始化所有節點的PR值為(1/節點總數)
  2. 接著使用以下公式迭代更新各節點PR值

d: damping factor

n: 節點總數

一張含有 文字 的圖片

自動產生的描述

一張含有 文字, 監視器, 螢幕, 電視 的圖片

自動產生的描述

* SimRank

1. 首先，為整張圖儲存一個n\*n的單位矩陣
2. 並根據以下公式進行迭代，以更新點對點的simrank值

一張含有 文字 的圖片

自動產生的描述

1. S(a,b)依據以下規則更新: (a,b為圖中的兩個節點)

If ( a == b ):

Simrank(a,b) = 1

Else:

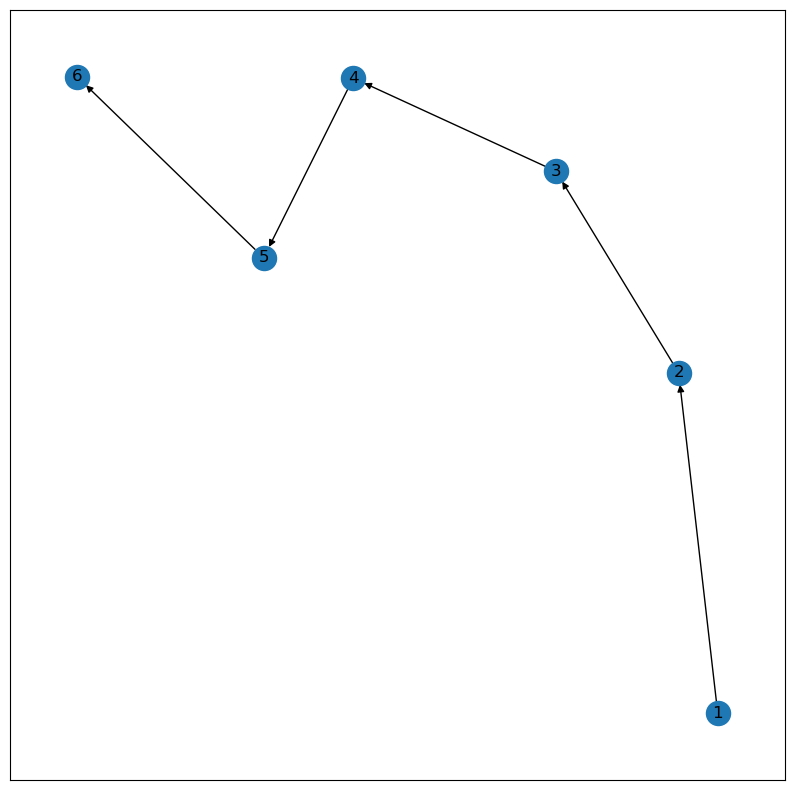
Simrank(a,b) = 0

1. I(a): 節點a的所有父親節點(所有指向自己的節點)
2. C: decay factor

一張含有 文字 的圖片

自動產生的描述

* Result Analysis and Discussion
  + graph\_1



* + - HITS:
      * Authority: 0.000 0.200 0.200 0.200 0.200 0.200
      * Hub: 0.200 0.200 0.200 0.200 0.200 0.000
      * auth值的大小取決於有多少Node指向自己，hub值取決於自己指向多少個Node。
      * Node1 的authority值為0，因為沒有任何Node指向Node1。
      * Node6 的hub值為0，因為Node6沒有指向任何點。
    - PageRank(PR):
      * 0.017 0.032 0.045 0.057 0.068 0.078
      * 因為此圖是一個單向流的圖，而pagerank的算法，Node2的PR值會加上加權後Node1的PR值，Node3的PR值會加上加權後Node2的PR值，以此類推，所以Node1到Node6的PR值是由小到大。
    - SimRank:
      * 1.000 0.000 0.000 0.000 0.000 0.000

0.000 1.000 0.000 0.000 0.000 0.000

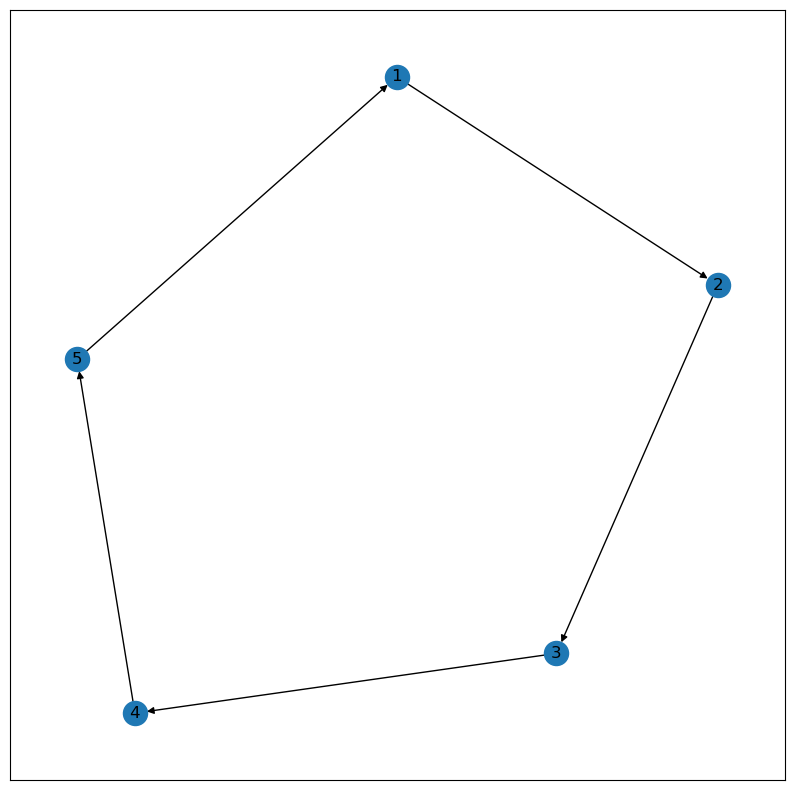
0.000 0.000 1.000 0.000 0.000 0.000

0.000 0.000 0.000 1.000 0.000 0.000

0.000 0.000 0.000 0.000 1.000 0.000

0.000 0.000 0.000 0.000 0.000 1.000

* + - * Simrank的計算中，兩個節點是否有相同父親是非常重要的，此圖中任兩節點都不存在相同父親，所以除了對角線外，其餘的值都是0。
  + graph\_2



* + - HITS:
      * Authority: 0.200 0.200 0.200 0.200 0.200
      * Hub: 0.200 0.200 0.200 0.200 0.200
      * 此圖的所有節點連結形成一個迴圈，每個節點都有一個父親和兒子，所以每個節點的auth和hub值理所當然都是相同的。
    - PageRank(PR):
      * 0.200 0.200 0.200 0.200 0.200
      * 此圖的所有節點連結形成一個迴圈，每個節點的PR值會收斂為相同的值。
    - SimRank:
      * 1.000 0.000 0.000 0.000 0.000

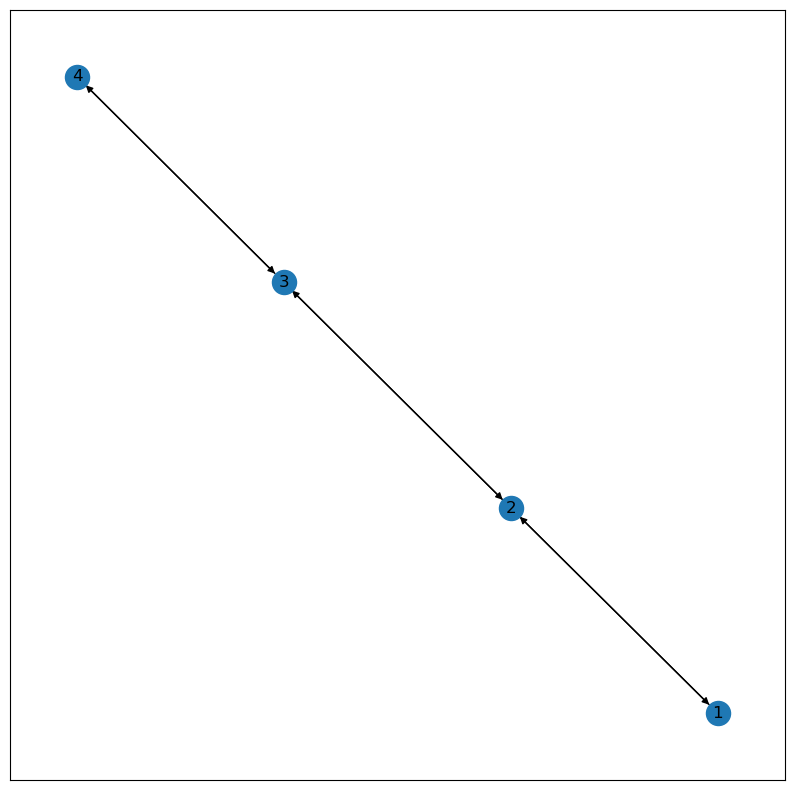
0.000 1.000 0.000 0.000 0.000

0.000 0.000 1.000 0.000 0.000

0.000 0.000 0.000 1.000 0.000

0.000 0.000 0.000 0.000 1.000

* + - * Simrank的計算中，兩個節點是否有相同父親是非常重要的，此圖中任兩節點都不存在相同父親，所以除了對角線外，其餘的值都是0。
  + graph\_3



* + - HITS:
      * Authority: 0.191 0.309 0.309 0.191
      * Hub: 0.191 0.309 0.309 0.191
      * Node2和Node3都有2個父親和2個兒子，所以auth和hub值會較高。
      * Node1和Node4則都只有1個父親和1個兒子，所以auth和hub值會較小。
    - PageRank:
      * 0.172 0.328 0.328 0.172
      * Node2和Node3都有2個父親和2個兒子，所以PR值會較高。
      * Node1和Node4則都只有1個父親和1個兒子，所以PR值會較小。
    - SimRank:
      * 1.000 0.000 0.538 0.000

0.000 1.000 0.000 0.538

0.538 0.000 1.000 0.000

0.000 0.538 0.000 1.000

* + - * 可以觀察到Node pair(1,3)、pair(1,4)都有各自共同的父親，所以simrank值不為0。
* 針對不同 damping factor 和 decay factor 進行討論
  + damping factor
    - 當damping factor趨近於1時，會導致所有節點的pagerank值均勻分布。
    - 當damping factor趨近於0時，有最多父親的節點會有較大的可能得到最大pagerank值。
    - graph\_4 搭配 damping factor = 0.1
      * 0.289 0.161 0.140 0.107 0.183 0.055 0.066
    - graph\_4 搭配 damping factor = 0.9
      * 可看出每個節點的pagerank平均分布
      * 0.160 0.143 0.140 0.136 0.156 0.132 0.132
  + decay factor
    - 因為相同node之間的simrank值是1，也就是s(x,x) = 1，所以為了區分極高相似與完全相同之間的差異，也就是避免s(a,b) = 1，所以引進decay factor。
    - graph\_4 搭配 decay factor = 0.7
* 1.000 0.243 0.232 0.239 0.221 0.303 0.175

0.243 1.000 0.294 0.256 0.295 0.170 0.343

0.232 0.294 1.000 0.340 0.275 0.339 0.341

0.239 0.256 0.340 1.000 0.230 0.427 0.427

0.221 0.295 0.275 0.230 1.000 0.159 0.300

0.303 0.170 0.339 0.427 0.159 1.000 0.155

0.175 0.343 0.341 0.427 0.300 0.155 1.000

* graph\_4 搭配 decay factor = 0.1
  + 1.000 0.010 0.010 0.014 0.008 0.026 0.001

0.010 1.000 0.023 0.018 0.019 0.001 0.034

0.010 0.023 1.000 0.034 0.018 0.034 0.034

0.014 0.018 0.034 1.000 0.014 0.050 0.050

0.008 0.019 0.018 0.014 1.000 0.001 0.026

0.026 0.001 0.034 0.050 0.001 1.000 0.001

0.001 0.034 0.034 0.050 0.026 0.001 1.000

* graph\_4 搭配 decay factor = 0.99
  + 1.000 0.934 0.933 0.933 0.931 0.941 0.925

0.934 1.000 0.940 0.935 0.941 0.924 0.946

0.933 0.940 1.000 0.945 0.938 0.944 0.945

0.933 0.935 0.945 1.000 0.931 0.956 0.956

0.931 0.941 0.938 0.931 1.000 0.922 0.940

0.941 0.924 0.944 0.956 0.922 1.000 0.922

0.925 0.946 0.945 0.956 0.940 0.922 1.000

* 可以觀察到decay factor設太小或太大的話，會導致兩個不同節點的simrank值很小或很大，看不出差異，所以simrank值的設定是很重要的。
* Effectiveness analysis

PageRank

|  |  |
| --- | --- |
| Graph | Time (seconds) |
| graph\_1 | 0.0 |
| graph\_2 | 0.0 |
| graph\_3 | 0.0 |
| graph\_4 | 0.0 |
| graph\_5 | 0.024915695190429688 |
| graph\_6 | 0.11833882331848145 |
| IBM\_dataset | 0.0547480583190918 |

HITS

|  |  |
| --- | --- |
| Graph | Time (seconds) |
| graph\_1 | 0.0 |
| graph\_2 | 0.0 |
| graph\_3 | 0.0 |
| graph\_4 | 0.0 |
| graph\_5 | 0.014954090118408203 |
| graph\_6 | 0.05481982231140137 |
| IBM\_dataset | 0.029900074005126953 |

SimRank

|  |  |
| --- | --- |
| Graph | Time (seconds) |
| graph\_1 | 0.0029904842376708984 |
| graph\_2 | 0.0029883384704589844 |
| graph\_3 | 0.0019958019256591797 |
| graph\_4 | 0.007990837097167969 |
| graph\_5 | 22.085312366485596 |
| graph\_6 | 314.83754229545593 |
| IBM\_dataset |  |

* 由上表可以得知，在graph1~graph4這些節點及邊數較少的圖，3個不同的演算法的執行時間都很少，但當遇到graph5、graph6、IBM\_dataset這種節點跟邊數都非常多的圖來說，執行時間就會更多，可以看到SimRank的執行時間又更多得原因是在於要去一一比對兩兩節點間是否存在相同的父親，這又使得SimRank這個演算法要花額外的時間去計算。