Linear Algebra HW4 Page Rank

助教 許博竣 r07942095@ntu.edu.tw

Outline

- Introduction

- ToDo

- Data Format

- Grading

- Submission

- Rules

Introduction - PageRank

- PageRank is an algorithm to measure the importance of a website.

- It was developed by Larry Page and Sergey Brin in 1996.

- The more links a website received from other websites, the more important it is.

- The algorithm iteratively calculate the importance of a website according to the websites that have links to it and their importance.

Introduction

$$PR(A) = PR(B) + PR(C) + PR(D)$$

$$PR(A) = \frac{PR(B)}{2} + \frac{PR(C)}{1} + \frac{PR(D)}{3}$$

$$PR(A) = \left(\frac{PR(B)}{2} + \frac{PR(C)}{1} + \frac{PR(D)}{3} \right) d + \frac{1-d}{4}$$
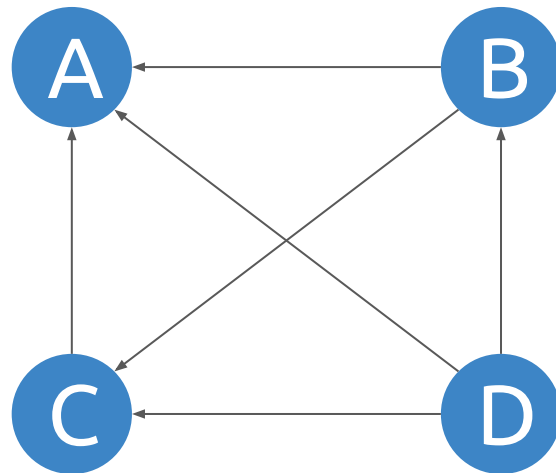
Introduction

$$\text{PageRank}(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{\text{PageRank}(p_j)}{L(p_j)}$$

$$\mathbf{R} = \begin{bmatrix} \text{PageRank}(p_1) \\ \text{PageRank}(p_2) \\ \vdots \\ \text{PageRank}(p_N) \end{bmatrix}$$

$$\mathbf{R}' = \begin{bmatrix} (1-d)/N \\ (1-d)/N \\ \vdots \\ (1-d)/N \end{bmatrix} + d \begin{bmatrix} \ell(p_1,p_1)_{/L(p_1)} & \ell(p_1,p_2)/L(p_2) & \cdots & \ell(p_1,p_N)_{/L(p_N)} \\ \ell(p_2,p_1)_{/L(p_1)} & \ddots & & \\ \vdots & & \ell(p_i,p_j)_{/L(p_j)} & \\ \ell(p_N,p_1)_{/L(p_1)} & & & \ell(p_N,p_N)_{/L(p_N)} \end{bmatrix}$$

$\ell(p_i, p_j)$: 1 if p_j has a link to p_i, else 0

$\mathbf{R}$, repeat util R' ≈ R

transition matrix

ToDo

- Download hw4.zip [here](#).

- Write hw4.py that can

    - Make the transition matrix.

    - Calculate the rank of every node.

- Answer the question [here](#).

ToDo

- Finish 3 ToDos in hw4.py

    - get_tran() -> get the transition matrix

    - cal_rank() -> calculate the rank of every node

        - d = 0.85

    - save() -> save the transition matrix, ranks to 1.txt, 2.txt

- Run the code to get 1.txt, 2.txt

    - python3 hw4.py graph.txt

    - use graph_n.txt according to last number of your student ID (b01234567 -> graph_7.txt)

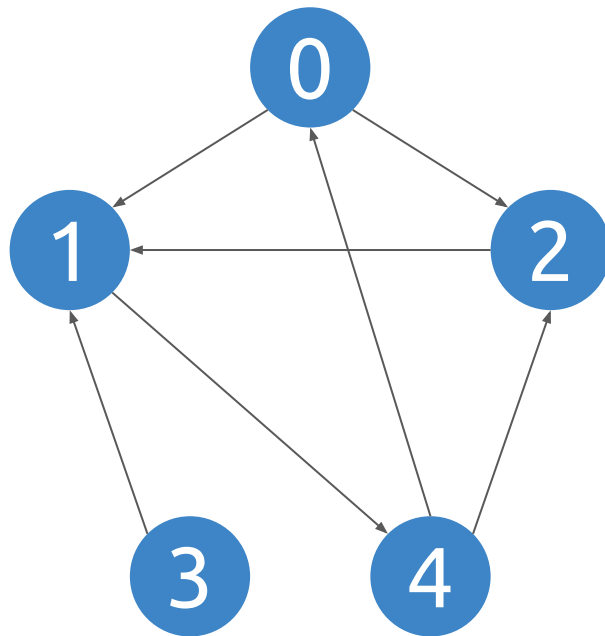   - 1.txt and 2.txt should be in the same directory as where hw4.py is

ToDo - cal_rank()

- Step 1: Initiala all PageRank in R_0 with 1/N. (N: the number of nodes)

- Step 2: Use R_t and transition matrix to calculate R_{t+1}

- Step 3: Repeat Step 2 until:

  - $\| R_{t+1} - R_t \|_1 \leq$ alpha (Use the funcion dist(R_{t+1} , R_t) to calculate $\| R_{t+1} - R_t \|_1$)

  - or when t ≥ max iterations
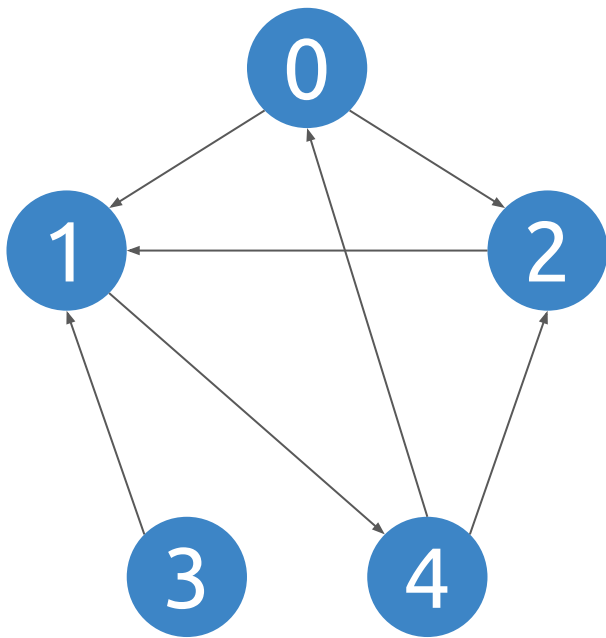
  - max iterations = 1000, alpha = 0.001

Data Format

- graph.txt

Data Format

- 1.txt


```
0 0 0 0 1
1 0 1 1 0
1 0 0 0 1
0 0 0 0 0
0 1 0 0 0
```
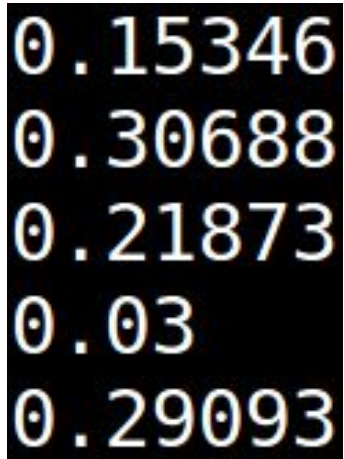
```
0.0 0.0 0.0 0.0 0.5
0.5 0.0 1.0 1.0 0.0
0.5 0.0 0.0 0.0 0.5
0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0
```

this is the transition matrix we want

Data Format

- 2.txt

```
0.15346
0.30688
0.21873
0.03
0.29093
```

- Note: don't worry about the number of the digits, small deviation is allowable

Grading

- Save the transition matrix of the graph in 1.txt. (1%)

  - Check ex/1.txt to make sure that your submission is in correct format.

- Save the rank score of every node in 2.txt. (3%)

  - Check ex/2.txt to make sure that your submission is in correct format.

- Q: If we initial R_0 with random numbers and keep SUM(R_0) = 1, will the

  ranking be different ? Try to explain it. (2%)

  - Write your answer in report.pdf.

  - No template for report.pdf, but write the answer in 1 page.

Submission

- You should put all your files in a folder and compress it in a zip file.

  b01234567_hw4.zip

  |—./b01234567_hw4.zip

    |—hw4.py

    |—1.txt

    |—2.txt

    |—report.pdf

- Note: **do not** upload any graph.txt or other files

Rules

- Plagiarism = 0 point

- Upload b01234567_hw4.zip to CEIBA

- DEADLINE: 2018/12/11 (Tue.) 23:59 (GMT+8:00)

- Late submission: total score x 0.8 (per day)

- Any other error: total score x 0.8

- Python packages: numpy, pandas, and any built-in package

- 請確定投影片中出現的紅字都注意看過 :）

FAQ