Warmup: Is the product of two symmetric matrices symmetric?
If yes, prove it. If no, give a counterexample.

The page rank algorithm

Problem: Given a network of webpages, rank the webpages according to its importance.

Example:

1 ➔ 2 ➔ web page ➔ 3

Idea: Rank webpages according to:
1. importance of the webpage
2. the number of other webpages it links to
3. the number of other webpages it recommends

The Page Rank is based on a recommender system.

The rank of a webpage increases if there are other webpages linking to it. Each web page linking to webpage x contributes to the rank of webpage x.
Assume y links to x. The amount webpage y contributes to the rank of webpage x is low if y links to
notation: let \( W \) be the set containing web page indices.

For the example, \( W = \{1, 2, 3, 4\} \)

- let \( n_j \) be the number of web-page that web-pages \( j \) links to.

For the example, \( n_1 = 2, n_2 = 0, n_3 = 3, n_4 = 0 \)

- let \( I_i \) be the set of web-pages that \( i \) links into web page \( i \).

For the example, \( I_1 = \{3\}, I_2 = \{1, 3\}, I_3 = \{1\}, I_4 = \{3\} \)

rank of web page

- let \( v_i \) be the rank of \( i \) th web-page.

How is \( v_i \) computed?

Intuitively, \( v_i = \sum_{j \in I_i} \frac{v_j}{n_j} \), \( i = 1, \ldots, n \) captures importance of \( i \) th web-page.

But we need to know the rank of all the web-pages that link into web-page \( i \) to compute \( v_i \). However, the Page Ranks are unknown!

So we use an iterative method to estimate \( v_1, \ldots, v_n \).
Iterative method: Given an initial guess of PageRank \( \{ r_i^{(0)} \}_{i=1}^n \), compute \( \{ r_i^{(k+1)} \}_{i=1}^n \) iteratively using \( \{ r_i^{(k)} \}_{i=1}^n \).

i.e. let \( r_i^{(k+1)} = \sum_{j \in I_i} \frac{r_j^{(k)}}{\eta_j}, \quad j = 1, \ldots, n. \)

Computing a priori Page Rank iteratively requires a prior knowledge of rank of web pages \( \{ r_i^{(0)} \}_{i=1}^n \).

For the initial guess, we could assume all the web pages are equally important, i.e.
assume \( r_i^{(0)} = \frac{1}{n}, \quad i = 1, \ldots, n. \)

Important question:

Question: For the example, what is \( r_2^{(1)} \)?

\[ r_2^{(1)} = \frac{V_4}{2} + \frac{V_4}{3} = \frac{12}{3} = 0.2857 \]

Important questions:
- Does the algorithm converge?
- What does it converge to?
- How fast does it converge?
We first represent \( \gamma_i^{(k+1)} = \sum_{j \in I_i} \frac{\gamma_j^{(k)}}{\eta_j} \) for \( i = 1, \ldots, n \) in matrix form.

The adjacency matrix of a network is defined as by

\[
\begin{align*}
\alpha_{ij} &= \begin{cases}
1 & \text{if web page } i \text{ links into web page } j \\
0 & \text{otherwise}
\end{cases}
\end{align*}
\]

Note that \( j \in I_i \) iff \( \alpha_{ji} = 1 \),

set of web pages that link into \( i \).

So, for the example

\[
\begin{array}{c}
1 \rightarrow 2 \\
3 \rightarrow 2 \\
4 \rightarrow 2
\end{array}
\]

what is \( A \)?

The adjacency matrix is

\[
A = \begin{bmatrix}
0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0
\end{bmatrix}
\]

So, \( \gamma_i^{(k+1)} = \sum_{j \in I_i} \frac{\gamma_j^{(k)}}{\eta_j} = \sum_{j=1}^{n} \frac{\alpha_{ji} \gamma_j^{(k)}}{\eta_j} = \sum_{j=1}^{n} h_{ij} \gamma_j^{(k)} \)

let \( h_{ij} = \frac{\alpha_{ji}}{\eta_j} \) for all \( i, j \in \{1, \ldots, n\} \)

So, \( H = \begin{bmatrix}
\frac{a_{11}/n_1}{n_1} & \frac{a_{12}/n_2}{n_1} & \cdots & \frac{a_{1n}/n_n}{n_1} \\
\frac{a_{21}/n_1}{n_2} & \frac{a_{22}/n_2}{n_2} & \cdots & \frac{a_{2n}/n_n}{n_2} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{a_{n1}/n_1}{n_n} & \frac{a_{n2}/n_2}{n_n} & \cdots & \frac{a_{nn}/n_n}{n_n}
\end{bmatrix} = A^T \text{diag} (\gamma_1, \gamma_2, \ldots, \gamma_n). \)
So, in matrix form, the iterative algorithm is

\[ \mathbf{v}(k+1) = \mathbf{H} \mathbf{v}(k) \]

\[ \mathbf{v}(0) = \mathbf{v}_0 \]

**Matlab example**

(Example 2)

The problem with using matrix \( \mathbf{H} \):

1. Rank sinks (accumulate more ranks as iteration progresses).
2. Cycle in graph.

2 Modification to \( \mathbf{H} \):

D. Note that column 2 of \( \mathbf{H} \) corresponds to "likelihood" of jumping to other webpages after landing on webpage 2. Since 2nd column is 0, it is equally acceptable to go to any other webpage. So, replace 2nd column by \( \frac{e}{n} \), as \( \mathbf{e} = [1, \cdots, 1]^T \).

So, for example 1, replace \( \mathbf{H} \) with

\[
\mathbf{S} = \begin{bmatrix} 0 & \frac{e}{n} & \frac{e}{n} & \frac{e}{n} \\ \frac{e}{n} & 0 & \frac{e}{n} & \frac{e}{n} \\ \frac{e}{n} & \frac{e}{n} & 0 & \frac{e}{n} \\ \frac{e}{n} & \frac{e}{n} & \frac{e}{n} & 0 \end{bmatrix} = \mathbf{H} + \frac{1}{n} \mathbf{e} \mathbf{e}^T,
\]

where \( \mathbf{e}^2 = \frac{1}{n} \mathbf{1} \mathbf{1}^T \) of \( \mathbf{H} \) is zero.
Also,

2) Assume there is a "random surfer."

\( \alpha \) is the probability that the surfer will continue clicking on a link.

\((1-\alpha)\) is the probability that the surfer will randomly choose a webpage and start clicking on links.

This is captured by

\[
G = \alpha S + (1-\alpha) n^4 e e^T
\]

So, the Page Rank iteration is

\[\gamma^{(k+1)} = G \gamma^{(k)}\]

\[\gamma^{(0)} = \gamma_0\]

Note that matrix \( G \) is dense, hence, matrix \( H \) is sparse. It is computationally more efficient to work with \( H \cdot (H \gamma) \).