MTL-106
Probability and Stochastic Processes

Assignment - 2A

Deadline : 10th April 2024

In this assignment you'll learn about an application of DTMC which is the pagerank algorithm coming from Google

Most search engines, including Google, continually run an army of computer programs that retrieve pages from the web, index the words in each document, and store this information in an efficient format. Each time a user asks for a web search using a search phrase, such as "search engine", the search engine determines all the pages on the web that contains the words in the search phrase. Here is the problem: Roughly 95% of the text in web pages is composed from a mere 10,000 words. This means that, for most searches, there will be a huge number of pages containing the words in the search phrase. What is needed is a means of ranking the importance of the pages that fit the search criteria so that the pages can be sorted with the most important pages at the top of the list.

Google's PageRank algorithm assesses the importance of web pages without human evaluation of the content. In fact, Google feels that the value of its service is largely in its ability to provide unbiased results to search queries; Google claims, "the heart of our software is PageRank". As we'll see, the trick is to ask the web itself to rank the importance of pages.

How to quantify the importance of a page ?

If you've ever created a web page, you've probably included links to other pages that contain valuable, reliable information. By doing so, you are affirming the importance of the pages you link to. The fundamental idea put forth by PageRank's creators, Sergey Brin and Lawrence Page, is this: the importance of a page is judged by the number of pages linking to it as well as their importance.

Here's how the PageRank is determined. Suppose that page P_j has l_j links. If one of those links is to page P_i, then P_j will pass on $1/l_j$ of its importance to P_i. The importance ranking of P_i is then the sum of all the contributions made by pages linking to it. That is, if we denote the set of pages linking to P_i by B_i, then

$$I(P_i) = \sum_{j \in B_i} \frac{I(P_j)}{l_j} \tag{1}$$

Let's first create H matrix (called hyperlink matrix) where

$$H_{ij} = \begin{cases} 1/l_j & \text{if } P_j \in B_i \\ 0 & \text{else} \end{cases}$$

Thus we can represent (1) in matrix form as $I = HI$, in other words I is stationary vector of H.

Computing I : Power-method

Choosing $I^{(0)}$ as initial guess for I, we produce a sequence of vectors as

$$I^{(k+1)} = HI^{(k)} \tag{2}$$

General principle : sequence $I^{(k)}$ will converge to I.

Probabilistic interpretation for H

Imagine that we surf the web at random; that is, when we find ourselves on a web page, we randomly follow one of its links to another page after one second. For instance, if we are on page P_j with l_j links, one of which takes us to page P_i, the probability that we next end up on P_i page is then $1/l_j$.

Of course, there is a complication in this description: If we surf randomly, at some point we will surely get stuck at a dangling node, a page with no links. To keep going, we will choose the next page at random; that is, we pretend that a dangling node has a link to every other page. This has the effect of modifying the hyperlink matrix H by replacing the column of zeroes corresponding to a dangling node with a column in which each entry is $1/n$. We call this new matrix S.

Thus $S = H + A$, where A is the matrix whose entries are all zero except for the columns corresponding to dangling nodes, in which each entry is $1/n$.

Problem with power method

Suppose that our web looks like this,

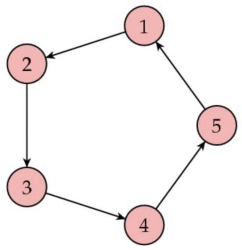

Figure 1: A period 5 chain

$$S = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$I^{(0)}$	$I^{(1)}$	$I^{(2)}$	$I^{(3)}$	$I^{(4)}$	$I^{(5)}$
1	0	0	0	0	1
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0

Thus the sequence $I^{(k)}$ fails to converge.

Consider another scenario

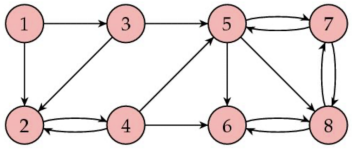

Figure 2: Non irreducible chain

$$S = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 1/3 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/3 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 1/3 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 1/3 & 1 & 1/2 & 0 \end{bmatrix} \text{ with stationary vector } I = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0.12 \\ 0.24 \\ 0.24 \\ 0.4 \end{bmatrix}$$

Notice that the pageranks assigned to the first four web pages are zero. However, this doesn't feel right: each of these pages has links coming to them from other pages. Clearly, they have some importance! generally speaking, we want the importance rankings of all pages to be positive. The problem with this example is that pages $\{1, 2, 3, 4\}$ are transient states and $\{5, 6, 7, 8\}$ forms a separate recurrent communicating class.

Thus to guarantee convergence, we can make modifications to S to make it aperiodic and irreducible (which guarantees existence of a stationary distribution over the web(chain) defined by the stochastic matrix S).

Final Modifications

We will modify the way our random surfer moves through the web. As it stands now, the movement of our random surfer is determined by S. To make our modification, we will first choose a parameter α between 0 and 1. Now suppose that our random surfer moves in a slightly different way. With probability α, he is guided by S. With probability $1-\alpha$, he chooses the next page at random.

If we denote by $\mathbf{1}$ the $n \times n$ matrix whose entries are all one, we obtain the Google matrix:

$$G = \alpha S + \frac{(1-\alpha)}{n}\mathbf{1} \tag{3}$$

G is stochastic matrix and for $\alpha \neq 0, 1$ is aperiodic and irreducible. Therefore G has a unique stationary vector I which can be found using power method.

Computing I

$$G = \alpha H + \alpha A + \frac{1-\alpha}{n}\mathbf{1}$$

Therefore,

$$GI^{(k)} = \alpha H I^{(k)} + \alpha A I^{(k)} + \frac{1-\alpha}{n}\mathbf{1}I^{(k)}$$

Now recall that most of the entries in H are zero. Therefore, evaluating $HI^{(k)}$ requires only few nonzero terms instead of $O(n)$ terms for each entry in the resulting vector. Also, the rows of A are all identical as are the rows of 1. Therefore, evaluating $AI^{(k)}$ and $\mathbf{1}I^{(k)}$ amounts to adding the current importance rankings of the dangling nodes or of all web pages. This only needs to be done once.

Coding Part

Your task is to compute I vector for a given web, with $\alpha = 0.85$ using the algorithm described above.

Input

Every line of input is of form $p\ \ q$, which denotes there is a link from page p to page q.

Output

Output is of form $p = I$, where p is pageid and I is importance of page p(pagerank).
Output is ordered in **increasing order** of pageids.
Last line is of form $s = x$. where x is sum of page ranks of all the pages in the web (x should be 1.0).
Note : Output is considered correct if and only if every page-rank value is within 10^{-4} of the actual values

Constraints and Time-Limit

Web-Size $\leq 100,000$ Time-Limit = 3sec

Sample Inputs and Outputs are provided separately (see MSTeams A2 channel), but your code will be tested on a much larger and exhaustive dataset so make sure to handle all the edge cases.

You are expected to write your own code from scratch using only the standard library of C++ or python. Any instances of plagiarism (either among students or copy from the internet) will be awarded 0.