Programming Assignment 2 - Implement Google's Page Rank

Submit Assignment

Due Nov 30 by 11:59pm **Points** 100 **Submitting** a file upload (Turnitin enabled)

Test your code in Stepik. We will create additional test cases to test your code on. Your code must compile and run in Stepik. We will retest your last Stepik submission.

Your submission in Canvas should be your commentary, worth 20% of your grade.

Your commentary should answer the following questions.

What data structure did you use for your table of URLs? Why did you choose it? If you could do the project over would you choose something different?

What graph implementation did you choose? Why did you choose it? If you could do the project over would you choose something different?

What is the computational complexity of the methods in your project?

What was the hardest part of this project?

What did you learn from this project? Try to be more specific than "I learned about page rank algorithm."

Goal:

Implement Google's page rank.

Input

Line 1 contains the number of lines (n) that will follow and the number of power iterations you need to perform. Each line from 2 to n will contain two URL's – *from_page* *to_page* separated by a space. This means from_page points to the URL to_page.

the first power iteration is simply the starting point for page ranks

2 power iterations means one matrix multiplication

3 power iterations means two matrix multiplications

Output

Print the PageRank of all pages after n powerIterations in ascending alphabetical order of webpage. Also, round off the rank of the page to two decimal places.

Google Page Rank

In late 90's as the number of webpages on the internet were growing exponentially different search engines were trying different approaches to rank the webpages. At Stanford, two computer science PhD students, Sergey Brin and Larry Page were working on the following questions: How can we trust information? Why are some web pages more important than others? Their research led to the formation of the Google search engine. In this programming assignment, you are required to implement a simplified version of the original PageRank algorithm on which Google was built.

Representing the Web as a Graph

The idea that the entire internet can be represented as a graph. Each node represents a webpage and each edge represents a link between two webpages. This graph can be implemented as an Adjacency Matrix or an Adjacency List.

We are explaining the assignment in the form of an Adjacency Matrix. We represent the graph in the form of |V|x|V| matrix where |V| is the total number of vertices in the graph. A vertex represents a webpage in the internet. Thus, if there is an edge from V_i to V_j page i points to page j. In the adjacency matrix M_{ji} >0 if there is an edge and 0 otherwise. Note that this is flipped compared to the adjacency matrix format we studied.

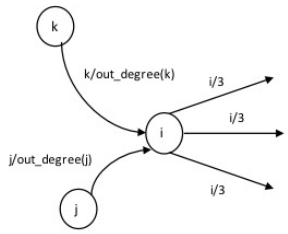
Core Ideas of PageRank

1. Important web pages will point to other important webpages.

2. Each page will have a score and the results of the search will be based on the page score (called page rank).

1.

Rank(i) = j/out_degree(j) + k/out_degree(k)



Each webpage is thus a node in the directed graph and has incoming edges and outgoing edges. Each node has a rank. According to PageRank, this rank is equally split among the node's outgoing links and this rank is equal to the sum of the incoming ranks. The rank is based on the indegree (the number of nodes pointing to it) and the importance of incoming node. This is important considering let's say you create your personal website and have a million links to other pages of importance. If this was not the case and rank used out links, we can easily dupe the algorithm. Therefore, the rank is based on in-links.

Sample Problem

Input:

7 2

google.com gmail.com

google.com maps.com

facebook.com ufl.edu

ufl.edu google.com

ufl.edu gmail.com

maps.com facebook.com

gmail.com maps.com

Step 1: Map URLs to a unique ID

1 google.com

2 gmail.com

3 facebook.com

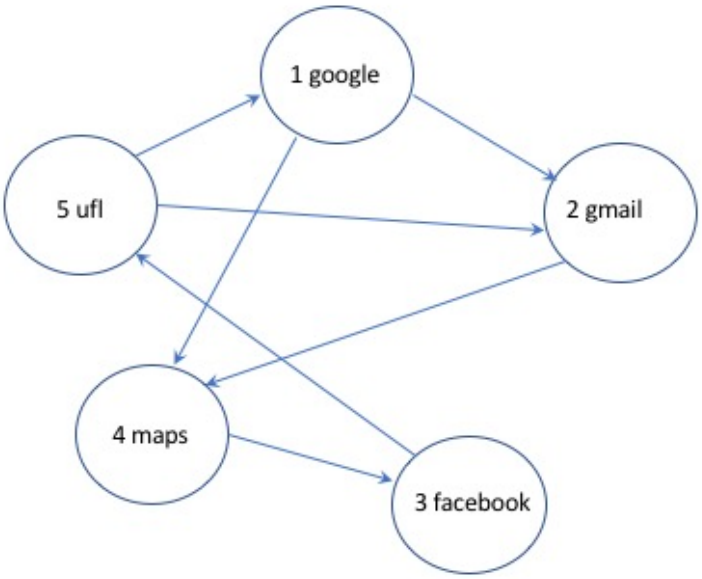
4 maps.com

5 ufl.edu

	Data Structure 1
1	google.com
2	gmail.com
3	facebook.com
4	maps.com
5	ufl.edu

Step 2. Graph Representation

Here is the graph for our example:



The initial values M_{ji} in the adjacency matrix are $1/d_i$ where d_i is the outdegree of vertex i.

For our graph, the adjacency matrix will look like:

	1	2	3	4	5
1	0	0	0	0	1/2
2	1/2	0	0	0	1/2
3	0	0	0	1/1	0
4	1/2	1/1	0	0	0
5	0	0	1/1	0	0

"5 UFL" points to "1 google". 5 has outdegree 2, so sends 1/2 its pagerank to 1. So M_{15} = 1/2.

Step 3: Power Iteration r(t+1) = M*r(t)

This means that a rank of the webpage at time t+1 is equal to the rank of that page at time t multiplied by matrix, M. To achieve this, we create our matrix M based on input. Next, we initialize r(t) which is a matrix of size |V|x1 and consists of the ranks of every webpage. We initialize r(t) to 1/|V|. Next we compute power_iterations based on our input.

$r(t+1)=r(0+1)=r(1) = M.r(0)=$

	1	2	3	4	5
1	0	0	0	0	1/2
2	1/2	0	0	0	1/2
3	0	0	0	1	0
4	1/2	1	0	0	0
5	0	0	1	0	0

x

	1
1	1/5
2	1/5
3	1/5
4	1/5
5	1/5

=

	1
1	1/10
2	1/5
3	1/5
4	3/10
5	1/5