Computing PageRank

IN3200/IN4200 Obligatory assignment 1, Spring 2022

Note: This is one of the two obligatory assignments that both need to be approved so that a student is allowed to take the final exam. (The grade of the final exam is otherwise *not* related to these two assignments.)

Note: Discussions between the students are allowed, but each student should write her/his own implementations. The details about the submission can be found at the end of this document.

1 Introduction

An important part of Google's search engine is the **PageRank** algorithm, which computes a numerical score for each webpage inside a group of webpages. The score of a webpage is meant to reflect its "importance", which in turn depends on both the number of webpages that link to it and the "importance" of these inbound webpages.

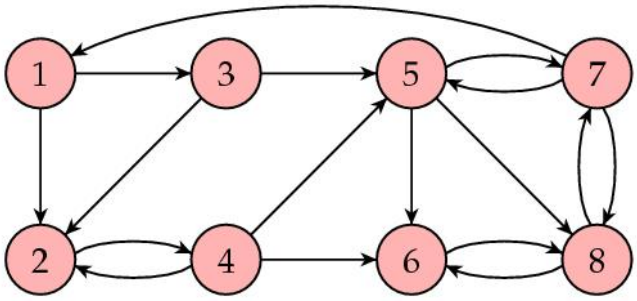

Figure 1: A very simple example of eight webpages that are linked to each other.

A simple example of eight linked webpages is depicted in Figure 1, also called a **web graph**. For webpage No. 1, it has two outbound links (to webpages No. 2 and No. 3) and one inbound link (from webpage No. 7). For webpage No. 2, it has one outbound link (to webpage No. 4) and three inbound links (from webpages No. 1, No. 3 and No. 4). Similarly, the inbound and outbound links for the other six webpages can be identified in Figure 1. In this example, No. 8 has the highest score of PageRank, because it has inbound links from "important" webpages No. 5, No. 6 and No. 7, which have several inbound links of their own. (The details about how to compute the PageRank scores will be shown in the next section.)

2 The PageRank algorithm

Before diving into the numerical scheme of the PageRank algorithm, a few words of "warning" are in order. There are numerous online articles discussing the mathematical foundation of the PageRank algorithm, but you absolutely don't have to understand the mathematical details in order to implement this numerically practical algorithm. In the following, we will show a minimum numerical

"skeleton" of the PageRank algorithm. If you want to understand more, a good reference can be the lecture notes authored by K. Shum [1].

2.1 The hyperlink matrix

For a group of linked webpages, with N denoting the total number of webpages, the linkage connection between them can be represented by a so-called hyperlink matrix $\mathbf{A}$ of dimension $N \times N$. Each row of $\mathbf{A}$, with index i, records all the inbound links towards webpage No. i. More specifically,

$$a_{ij} = \begin{cases} \frac{1}{L(j)} & \text{if there's an inbound link from webpage No. } j, \ (i \neq j) \\ 0 & \text{otherwise,} \end{cases}$$

where $L(j)$ denotes the number of outbound links from webpage No. j.

For example, the hyperlink matrix for the eight webpages shown in Figure 1 will be as follows:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{3} & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & \frac{1}{3} & 1 & \frac{1}{3} & 0 \end{bmatrix}$$

We remark that the main diagonal of a hyperlink matrix $\mathbf{A}$ contains only zeros (because self-linkage should not affect the PageRank score). The values of each column of $\mathbf{A}$ either sum up to 1, or are all zeros. The latter case corresponds to a webpage that has no outbound links, which is also called a **dangling webpage**. Another important remark is that $\mathbf{A}$ is a sparse matrix, where most of its values are zero (especially when the number of webpages N is large).

2.2 The iterative procedure

The objective of the PageRank algorithm is to find a vector of numerical values

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

where x_i is the PageRank score of webpage No. i.

The PageRank algorithm is an iterative procedure that starts with an initial guess

$$\mathbf{x}^0 = \frac{1}{N} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

and uses the following formula to calculate $\mathbf{x}^k$ of iteration k based on $\mathbf{x}^{k-1}$:

$$\mathbf{x}^k = \frac{\left(1 - d + d \cdot W^{k-1}\right)}{N} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + d \cdot \mathbf{A}\mathbf{x}^{k-1} \tag{1}$$

Here, d is a prescribed scalar constant between 0 and 1, which is called the **damping constant**. (A typical choice of d is 0.85 in practice.) The term of $\mathbf{A}\mathbf{x}^{k-1}$ represents a sparse matrix-vector multiplication. The scalar value W^{k-1} is the summation of the PageRank scores, from iteration $k-1$, of all the dangling webpages, that is,

$$W^{k-1} = \sum_{m \in \mathcal{D}} x_m^{k-1} \quad \mathcal{D}\text{: set of indices for all the dangling webpages}$$

Note: if there is no dangling webpage at all, then the scalar value of W^{k-1} is simply zero.

2.3 Stopping criterion

The iterations should be stopped if the difference between vectors $\mathbf{x}^{k-1}$ and $\mathbf{x}^k$ is small enough (that is, the iterations have converged). Then, the vector $\mathbf{x}^k$ is deemed as a good enough numerical approximation of the vector of true PageRank scores: $\mathbf{x}$.

The following is a suggested way of testing that the difference between $\mathbf{x}^{k-1}$ and $\mathbf{x}^k$ is small enough:

$$\max_{1 \le j \le N} \left| x_j^k - x_j^{k-1} \right| < \varepsilon \tag{2}$$

where ε is a prescribed convergence threshold value (typically very small).

3 Required work

3.1 Three serial functions

The following three functions should be implemented:

1. `void read_graph_from_file (char *filename, int *N, int **row_ptr, int **col_idx, double **val)` This function should read a text file that contains a web graph, so that the corresponding hyperlink matrix is built up in the CRS format (see Section 3.6.1 of the textbook). **Note:** The reason for `row_ptr`, `col_idx` and `val` being double pointers in the argument list is because these 1D arrays need to be allocated inside the function.

2. `void PageRank_iterations (int N, int *row_ptr, int *col_idx, double *val, double d, double epsilon, double *scores)` This function should implement the iterative procedure of the PageRank algorithm. The input arguments include the hyperlink matrix in CRS format, the damping constant (`d`), and the convergence threshold value (`epsilon`). The computed PageRank scores are to be contained in the pre-allocated 1D array `scores`.

3. `void top_n_webpages (int N, double *scores, int n)` This function should go through the computed PageRank score vector `scores` and print out the top `n` webpages, with both their scores and webpage indices. **Note:** The first input argument `N` is the length of the PageRank score vector.

3.2 OpenMP parallelization

The `PageRank_iterations` function must be parallelized using OpenMP. IN4200 students must also parallelize the `top_n_webpages` function (not required for IN3200 students). The `read_graph_from_file` function does not need to be parallelized.

3.3 File format of a web graph

It can be assumed that a text file containing a web graph (that is, the webpage linkage connection information) has the following format:

- The first two lines both start with the `#` symbol and contain free text (listing the name of the data file, authors etc.);
- Line 3 is of the form "`# Nodes: integer1 Edges: integer2`", where `integer1` is the total number of webpages, and `integer2` is the total number of links. (Here, nodes mean the same as webpages, and edges mean the same as links.)
- Line 4 is of the form "`# FromNodeId ToNodeId`";
- The remaining part of the file consists of a number of lines, the total number equals the number of links. Each line simply contains two integers: the index of the outbound webpage and the index of the inbound webpage;
- Some of the links can be self-links (same outbound as inbound), these should be excluded when creating the hyperlink matrix;
- Note: the webpage indices start from 0 (C convention).
- It can be assumed that each web link is uniquely listed, that is, there will NOT be multiple text lines describing the same web link.
- You may however NOT assume that the links are sorted with respect to `FromNodeId`.
- For each `FromNodeId`, you may NOT assume that the links are sorted with respect to `ToNodeId`.

An example web graph file that contains the linkage connection information shown in Figure 1 is as follows:

```
# Directed graph (each unordered pair of nodes is saved once): 8-webpages.txt
# Just an example
# Nodes: 8 Edges: 17
# FromNodeId    ToNodeId
0       1
0       2
1       3
2       4
2       1
3       4
3       5
3       1
4       6
4       7
4       5
5       7
6       0
6       4
6       7
7       5
7       6
```

The above small example of web graph can be downloaded from

`https://www.uio.no/studier/emner/matnat/ifi/IN3200/v22/teaching-material/simple-webgraph.txt`

Another example, with 100 nodes (webpages), can be downloaded from

`https://www.uio.no/studier/emner/matnat/ifi/IN3200/v22/teaching-material/100nodes_graph.txt`

A real-world web graph file (rather large) can be downloaded from

`https://www.uio.no/studier/emner/matnat/ifi/IN3200/v22/teaching-material/web-stanford.txt`

3.4 Submission

Each student should submit, via Devilry, a tarball (`.tar`) or a zip file (`.zip`). Upon unpacking/unzipping it should produce a folder named `IN3200_Oblig1_xxx` or `IN4200_Oblig1_xxx`, where `xxx` should be the candidate number of the student (can be found in StudentWeb). Inside the folder, there should be *at least* the following files:

- `read_graph_from_file.c`
- `PageRank_iterations.c` (The OpenMP parallelization can be done in the same file, or in a separate file `PageRank_iterations_omp.c`)
- `top_n_webpages.c` (The OpenMP parallelization, only required for IN4200 students, can be done in the same file, or in a separate file `top_n_webpages_omp.c`)
- `main.c` should contain a main program that accepts, on the command line, the filename of a web graph file, the damping constant d, the convergence threshold value ε, and value of n related to showing the top n webpages (see above). This main program should call the three functions. Proper comments and output info (using `printf`) should be provided.
- `README.txt` or `README.md` should contain the most essential information about compilation and an example of how to run the main program.

References

[1] Kenneth Shum. *Notes on PageRank Algorithm*. (Lecture notes of ENGG2012B Advanced Engineering Mathematics, The Chinese University of Hong Kong). 2013.
`http://home.ie.cuhk.edu.hk/~wkshum/papers/pagerank.pdf`

Hints

When reading a web graph file to construct a hyperlink matrix in the CRS format, it is recommended to read through the web graph file *twice*. The first pass is to do the necessary counting and fill array `row_ptr` correctly. It is in the second pass where arrays `col_idx` and `val` are correctly filled.

Remark: The brief description of the CRS format given in the textbook has a "flaw". Specifically, the array `row_ptr` should be of length $N+1$, and the last value of `row_ptr` should be the total number of nonzeros.

For the 8-webpage simple example, its hyperlink matrix in the CRS format will be

```
row_ptr=[0,1,4,5,6,9,12,14,17]
col_idx=[6,0,2,3,0,1,2,3,6,3,4,7,4,7,4,5,6]
val=[0.333333,0.500000,0.500000,0.333333,0.500000,
1.000000,0.500000,0.333333,0.333333,0.333333,0.333333,
0.500000,0.333333,0.500000,0.333333,1.000000,0.333333]
```

Rrunning the PageRank algorithm using $d = 1$ (no damping) will produce the following results:

```
---Initial guess:
scores=[0.125000,0.125000,0.125000,0.125000,0.125000,0.125000,0.125000,0.125000]
---After iteration 1:
scores=[0.041667,0.166667,0.062500,0.125000,0.145833,0.145833,0.104167,0.208333]
---After iteration 2:
scores=[0.034722,0.093750,0.020833,0.166667,0.107639,0.194444,0.152778,0.229167]
---After iteration 3:
scores=[0.050926,0.083333,0.017361,0.093750,0.116898,0.206019,0.150463,0.281250]
---After iteration 4:
```

```
scores=[0.050154,0.065394,0.025463,0.083333,0.090085,0.210841,0.179591,0.295139]
---After iteration 5:
scores=[0.059864,0.065586,0.025077,0.065394,0.100373,0.205376,0.177598,0.300733]
....
```

When converged, the PageRank scores will be as follows:

```
scores=[0.060000,0.067500,0.030000,0.067500,0.097500,0.202500,0.180000,0.295000]
```