THE UNIVERSITY OF SYDNEY

COMP2129 Assignment 4

Due: 11:59pm Tuesday 17th May

Task

This task will improve your parallel programming skills dealing with pthreads in C and will improve your performance engineering skills to make the execution of programs fast.

Internet search engines compare words in search phrases to words on web-pages and in titles of web-pages to determine a list of sites relevant to the search of a user. The algorithm that is employed by *Google*® to compute the search is called the **PageRank algorithm**, and was developed in 1996 by Larry Page and Sergey Brin when they were graduate students at Stanford University.

The underlying idea of the PageRank algorithm is to construct matrices that describe the referencing structure of web-pages, and then use the dominant eigenvectors of those matrices to list the pages in descending order of importance.

Your task is to implement the PageRank algorithm. Your mark is split into two components, i.e., correctness and performance. The performance mark is determined by the relative performance to other student's submissions.

The PageRank Algorithm

The PageRank algorithm is an iterative algorithm. We repeat the algorithm until a threshold is met. The last iteration gives us the result of the search. The result of the PageRank algorithm is a score per web-page. A high scoring web-page indicates a highly relevant web-page whereas a low scoring web-page indicates not so relevant web-page for a search. By sorting the web-page by their scores in descending order we obtain the order for the result list of a search query.

For describing the PageRank algorithm we introduce the following symbols:

- S is the set of all web pages we are computing the PageRank scores for
- $N = |S|$; the total number of web pages
- $\mathbf{P}$ is the vector of PageRank scores
- d is a dampening factor
- ϵ is the convergence threshold
- $\text{IN}(p)$ is the set of all pages in S which link to page p
- $\text{OUT}(p)$ is the set of all pages in S which page p links to

For the PageRank vector, we use the notation $\mathbf{P}_p^{(t)}$ to represent the PageRank score for page p at iteration t. We initialise the PageRank scores for all pages with an initial value of $\frac{1}{N}$ so that the sum of the PageRank scores equals 1.

$$\mathbf{P}_u^{(0)} = \frac{1}{N} \tag{1}$$

During each iteration of the algorithm, the current iteration of $\mathbf{P}$ is calculated as follows:

$$\mathbf{P}_u^{(t+1)} = \frac{1-d}{N} + d \sum_{v \in \text{IN}(u)} \frac{\mathbf{P}_v^{(t)}}{|\text{OUT}(v)|} \tag{2}$$

The algorithm continues to iterate until the convergence threshold is reached; that is, the point in time when the PageRank scores stop varying between iterations. The PageRank scores have converged when the following condition is met:

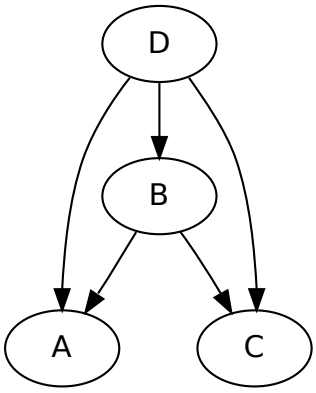
$$||\mathbf{P}^{(t+1)} - \mathbf{P}^{(t)}|| \leq \epsilon \tag{3}$$

The vector norm is defined to be the standard Euclidean vector norm; that is, for some vector $\mathbf{x} = \{x_1, x_2, \ldots\}$:

$$||\mathbf{x}|| = \sqrt{\sum_i x_i^2} \tag{4}$$

Motivating Example

Our motivating example has four web pages, i.e., $S = \{A, B, C, D\}$. In this example, we are fixing $d = 0.85$ and $\epsilon = 0.005$. The referencing structure of the web-pages A, B, C, and D is given in the graph below



$$\begin{aligned} \text{IN}(A) &= \{B, D\} & \qquad \text{OUT}(A) &= \emptyset \\ \text{IN}(B) &= \{D\} & \qquad \text{OUT}(B) &= \{A, C\} \\ \text{IN}(C) &= \{B, D\} & \qquad \text{OUT}(C) &= \emptyset \\ \text{IN}(D) &= \emptyset & \qquad \text{OUT}(D) &= \{A, B, C\} \end{aligned}$$

where a node represents a web-page and an edge in the graph indicates that the source of the edge is linking to the destination of the edge. Note that the edges of the graph are encoded in the sets IN and OUT.

Initialise $\mathbf{P}^{(0)} = \frac{1}{N}$. Then perform the first iteration for each page.

$$\mathbf{P}_A^{(1)} = \frac{1-0.85}{4} + 0.85\left(\frac{\mathbf{P}_B^{(0)}}{|\{A,C\}|} + \frac{\mathbf{P}_D^{(0)}}{|\{A,B,C\}|}\right) = \frac{0.15}{4} + 0.85\left(\frac{0.25}{2} + \frac{0.25}{3}\right) \approx 0.214$$

$$\mathbf{P}_B^{(1)} = \frac{1-0.85}{4} + 0.85\left(\frac{\mathbf{P}_D^{(0)}}{|\{A,B,C\}|}\right) = \frac{0.15}{4} + 0.85\left(\frac{0.25}{3}\right) \approx 0.108$$

$$\mathbf{P}_C^{(1)} = \frac{1-0.85}{4} + 0.85\left(\frac{\mathbf{P}_B^{(0)}}{|\{A,C\}|} + \frac{\mathbf{P}_D^{(0)}}{|\{A,B,C\}|}\right) = \frac{0.15}{4} + 0.85\left(\frac{0.25}{2} + \frac{0.25}{3}\right) \approx 0.214$$

$$\mathbf{P}_D^{(1)} = \frac{1-0.85}{4} + 0.85\,(0) = \frac{0.15}{4} + 0 \approx 0.038$$

Which leaves us with the following two iterations of $\mathbf{P}$:

t	A	B	C	D
0	0.250	0.250	0.250	0.250
1	0.214	0.108	0.214	0.038

Next, we check whether or not the algorithm has converged.

$$\begin{aligned} ||\mathbf{P}^{(1)} - \mathbf{P}^{(0)}|| &= ||\{-0.036, -0.142, -0.036, -0.215\}|| \\ &= \sqrt{0.0677} \\ &\approx 0.260 \end{aligned}$$

Since $0.260 \not\leq 0.005$ (ϵ) we then perform another iteration.

$$\mathbf{P}_A^{(2)} = \frac{1-0.85}{4} + 0.85\left(\frac{\mathbf{P}_B^{(1)}}{|\{A,C\}|} + \frac{\mathbf{P}_D^{(1)}}{|\{A,B,C\}|}\right) = \frac{0.15}{4} + 0.85\left(\frac{0.108}{2} + \frac{0.038}{3}\right) \approx 0.094$$

$$\mathbf{P}_B^{(2)} = \frac{1-0.85}{4} + 0.85\left(\frac{\mathbf{P}_D^{(1)}}{|\{A,B,C\}|}\right) = \frac{0.15}{4} + 0.85\left(\frac{0.038}{3}\right) \approx 0.048$$

$$\mathbf{P}_C^{(2)} = \frac{1-0.85}{4} + 0.85\left(\frac{\mathbf{P}_B^{(1)}}{|\{A,C\}|} + \frac{\mathbf{P}_D^{(1)}}{|\{A,B,C\}|}\right) = \frac{0.15}{4} + 0.85\left(\frac{0.108}{2} + \frac{0.038}{3}\right) \approx 0.094$$

$$\mathbf{P}_D^{(2)} = \frac{1-0.85}{4} + 0.85\,(0) = \frac{0.15}{4} + 0 \approx 0.038$$

Which leaves us with the following three iterations of $\mathbf{P}$:

t	A	B	C	D
0	0.250	0.250	0.250	0.250
1	0.214	0.108	0.214	0.038
2	0.094	0.048	0.094	0.038

Again, we check whether or not the algorithm has converged:

$$||\mathbf{P}^{(2)} - \mathbf{P}^{(1)}|| \approx 0.180 \not\leq \epsilon$$

Since convergence has not been reached, we perform another iteration.

$$\begin{aligned}
\mathbf{P}_A^{(3)} &= \frac{1-0.85}{4} + 0.85\left(\frac{\mathbf{P}_B^{(2)}}{|\{A,C\}|} + \frac{\mathbf{P}_D^{(2)}}{|\{A,B,C\}|}\right) = \frac{0.15}{4} + 0.85\left(\frac{0.048}{2} + \frac{0.038}{3}\right) \approx 0.069 \\
\mathbf{P}_B^{(3)} &= \frac{1-0.85}{4} + 0.85\left(\frac{\mathbf{P}_D^{(2)}}{|\{A,B,C\}|}\right) = \frac{0.15}{4} + 0.85\left(\frac{0.038}{3}\right) \approx 0.048 \\
\mathbf{P}_C^{(3)} &= \frac{1-0.85}{4} + 0.85\left(\frac{\mathbf{P}_B^{(2)}}{|\{A,C\}|} + \frac{\mathbf{P}_D^{(2)}}{|\{A,B,C\}|}\right) = \frac{0.15}{4} + 0.85\left(\frac{0.048}{2} + \frac{0.038}{3}\right) \approx 0.069 \\
\mathbf{P}_D^{(3)} &= \frac{1-0.85}{4} + 0.85\,(0) = \frac{0.15}{4} + 0 \approx 0.038
\end{aligned}$$

Again, we check whether or not the algorithm has converged:

$$||\mathbf{P}^{(3)} - \mathbf{P}^{(2)}|| \approx 0.040 \not\leq \epsilon$$

Since convergence has not been reached, we perform another iteration.

$$\begin{aligned}
\mathbf{P}_A^{(4)} &= \frac{1-0.85}{4} + 0.85\left(\frac{\mathbf{P}_B^{(3)}}{|\{A,C\}|} + \frac{\mathbf{P}_D^{(3)}}{|\{A,B,C\}|}\right) = \frac{0.15}{4} + 0.85\left(\frac{0.048}{2} + \frac{0.038}{3}\right) \approx 0.069 \\
\mathbf{P}_B^{(4)} &= \frac{1-0.85}{4} + 0.85\left(\frac{\mathbf{P}_D^{(3)}}{|\{A,B,C\}|}\right) = \frac{0.15}{4} + 0.85\left(\frac{0.038}{3}\right) \approx 0.048 \\
\mathbf{P}_C^{(4)} &= \frac{1-0.85}{4} + 0.85\left(\frac{\mathbf{P}_B^{(3)}}{|\{A,C\}|} + \frac{\mathbf{P}_D^{(3)}}{|\{A,B,C\}|}\right) = \frac{0.15}{4} + 0.85\left(\frac{0.048}{2} + \frac{0.038}{3}\right) \approx 0.069 \\
\mathbf{P}_D^{(4)} &= \frac{1-0.85}{4} + 0.85\,(0) = \frac{0.15}{4} + 0 \approx 0.038
\end{aligned}$$

Again, we check whether or not the algorithm has converged:

$$||\mathbf{P}^{(4)} - \mathbf{P}^{(3)}|| = 0 \leq \epsilon$$

Since convergence has been reached, the algorithm now terminates, and the values of $\mathbf{P}^{(4)}$ are the final PageRank scores for each of the pages.

Hence, the result of the search query is A, C, B, D. Note that web-page A and web-page C have the same score. If this is the case, we use the name of the web-page to disambiguiate.

Input/Output

The structure of the input is as follows:

```
<number of cores>
<number of pages>
```

```
<name of web-page>
...
<number of edges>
<web-page> <web-page>
...
```

The first line specifies how many cores are available on the computer. You need to take the number of cores into account whilst you are optimizing the performance of the program.

The next input line specifies the number of web-pages followed by the names of the web-pages. After declaring the names of the web-pages, the number of edges in the graph are given. Each edge in the graph is given by the source and the destination of an edge.

For the motivating example, the input is as follows assuming that the number of cores on the computer is 8.

```
8
4
A
B
C
D
5
D A
D B
D C
B A
B C
```

Again, the first line gives the number of cores in the system (based on this number you need to design a parallelization strategy), followed is the number of web-pages which is four for our motivating example. Then, the names of the web-pages follow, i.e., A, B, C, and D. After defining the names of the web-pages, the number of edges is given which is 5. Then, the edges of the graph follow, i.e., $D \rightarrow A$, $D \rightarrow B$, $D \rightarrow C$, $B \rightarrow A$, and $B \rightarrow C$.

We assume that there must be at least one web-page in the graph and the algorithm is executed at least on one core.

If the input does not follow the input format, your program should print **`error`** and terminate. If any page is declared twice or if an edge is defined to a nonexistent page, then your program should print **`error`** and terminate. Additionally, if the number of cores or the number of pages is not positive, your program should print **`error`** and terminate.

The output is the list of scores per web-page, i.e.,

```
A 0.069
B 0.048
C 0.069
D 0.038
```

The order of the scores corresponds to the order how you define the web-pages in the input.

For printing the score, use the format string `"%s %.4lf\n"`. For all test cases set $d = 0.85$ and set $\epsilon = 0.005$. The names of the web-pages will be at most 19 characters long.

Writing your own test cases

Since your assignment will be automatically marked, it is crucial that you follow our instructions carefully. Your output will need to be in exactly the right format. To assist with this, we have made available some sample test cases. If you are logged in to the **`ucpu[01]`** machines, you can run these tests using the Makefile we provided by typing **`make test`**. If your output is incorrect, you will see both the expected output and your output. If you passed the test, then you will only see the name of the test.

The sample test cases are by no means exhaustive. You will need to test your code more thoroughly by thinking carefully about the specifications and writing your own tests.

As with the previous assignment, you can view the sample test cases manually. They are located in the folder **`~comp2129/assignment4/sample/question1`**.