THE UNIVERSITY OF SYDNEY

COMP2129 Assignment 4

Due: 9:00pm Sunday, 5 June 2016

This assignment is worth 8% of your final assessment

Task description

In this assignment your task is to implement the PageRank algorithm in C using the power method described below and then optimise and parallelise your code to ensure peak performance is achieved.

The PageRank algorithm was developed in 1996 by Larry Page and Sergey Brin when they were graduate students at Stanford University. Google and other search engines compare words in search phrases to words in web pages and use ranking algorithms to determine the most relevant results.

PageRank assigns a score to a set of web pages that indicates their importance. The underlying idea behind PageRank is to model a user who is clicking on web pages and following links from one page to another. In this framework, important pages are those which have incoming links from many other pages, or have incoming links from other pages with a high PageRank score, or both.

You are encouraged to ask questions on Ed using the assignments category. As with any assignment, make sure that your work is your own, and you do not share your code or solutions with other students.

Working on your assignment

You can work on this assignment on your own computer or the lab machines. Students can also take advantage of the online code editor on Ed. Simply navigate to the assignment page and you are able to run, edit and submit code from within your browser. We recommend that you use Safari or Chrome.

It is important that you continually back up your assignment files onto your own machine, flash drives, external hard drives and cloud storage providers. You are encouraged to submit your assignment while you are in the process of completing it. By submitting you will obtain some feedback of your progress.

Academic declaration

By submitting this assignment you declare the following:

I declare that I have read and understood the University of Sydney Academic Dishonesty and Plagiarism in Coursework Policy, and except where specifically acknowledged, the work contained in this assignment or project is my own work, and has not been copied from other sources or been previously submitted for award or assessment.

I understand that failure to comply with the Academic Dishonesty and Plagiarism in Coursework Policy, can lead to severe penalties as outlined under Chapter 8 of the University of Sydney By-Law 1999 (as amended). These penalties may be imposed in cases where any significant portion of my submitted work has been copied without proper acknowledgement from other sources, including published works, the internet, existing programs, the work of other students, or work previously submitted for other awards or assessments.

I realise that I may be asked to identify those portions of the work contributed by me and required to demonstrate my knowledge of the relevant material by answering oral questions or by undertaking supplementary work, either written or in the laboratory, in order to arrive at the final assessment mark.

I acknowledge that the School of Information Technologies, in assessing this assignment, may reproduce it entirely, may provide a copy to another member of faculty, and or communicate a copy of this assignment to a plagiarism checking service or in house computer program, and that a copy of the assignment may be maintained by the service or the School of IT for the purpose of future plagiarism checking.

The PageRank algorithm

PageRank is an iterative algorithm that is repeated until a stopping criteria is met. The last iteration gives us the result of the search, which is a score per web page. A high score indicates a very relevant web page whereas a low score indicates a not so relevant web page for a search. Sorting the web pages by their scores in descending order gives us 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 that we are computing the PageRank scores for
- $N = |S|$ is the total number of web pages
- $\mathbf{P} = [P_1^t, P_2^t, ..., P_N^t]$ is the vector of PageRank scores
- d is a dampening factor for the probability that the user continues clicking on web pages
- 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
- $\mathcal{M}_{ij}$ is the transition probability of travelling to page i from j
- $\widehat{\mathcal{M}}$ is a matrix calculated from $\mathcal{M}$, d and N that will be multiplied with $\mathbf{P}$ for each iteration

First we construct the matrix $\mathcal{M}$ as follows:

$$\mathcal{M}_{ij} = \begin{cases} 1 \,/\, N, & \text{if } |\text{OUT}(j)| = 0 \\ 1 \,/\, |\text{OUT}(j)|, & \text{if } j \text{ links to } i \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

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 scores for all pages to an initial value of $\frac{1}{N}$ so that the sum equals 1.

$$\mathbf{P}_u^{(0)} = \frac{1}{N} \tag{2}$$

Next we construct matrix $\widehat{\mathcal{M}}$ as follows:

$$\widehat{\mathcal{M}} = d\mathcal{M} + \frac{1-d}{N}\mathbf{E} \tag{3}$$

where $\mathbf{E}$ is an $N \times N$ matrix of all ones.

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

$$\mathbf{P}^{(t+1)} = \widehat{\mathcal{M}}\mathbf{P}^{(t)} \tag{4}$$

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

$$||\mathbf{P}^{(t+1)} - \mathbf{P}^{(t)}|| \leq \epsilon \tag{5}$$

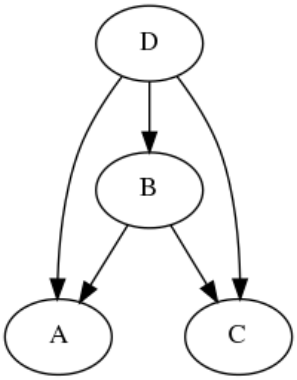
The vector norm is defined to be the standard Euclidean vector norm; that is,
for some vector $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$:

$$||\mathbf{x}|| = \sqrt{\sum_i x_i^2} \tag{6}$$

Example

Our example has four web pages: $S = \{A, B, C, D\}$. In this example, $d = 0.85$ and $\epsilon = 0.005$.

The referencing structure of the web pages A, B, C, and D is given in the graph below.



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

Each node represents a web page.
Edges in the graph indicate that the source of the edge is linking to the destination of the edge.

First calculate $\mathcal{M}$:

$$\mathcal{M} = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 0.25 & 0.5 & 0.25 & 0.333 \\ 0.25 & 0 & 0.25 & 0.333 \\ 0.25 & 0.5 & 0.25 & 0.333 \\ 0.25 & 0 & 0.25 & 0 \end{bmatrix} \end{array}$$

Notice that the sum of each column is 1. Columns A and C have value $1/N$ because

$$|\text{OUT}(A)| = |\text{OUT}(C)| = 0$$

Now calculate $\widehat{\mathcal{M}}$:

$$\widehat{\mathcal{M}} = d\mathcal{M} + \frac{1-d}{N}\mathbf{E} \approx \begin{bmatrix} 0.250 & 0.463 & 0.250 & 0.321 \\ 0.250 & 0.038 & 0.250 & 0.321 \\ 0.250 & 0.463 & 0.250 & 0.321 \\ 0.250 & 0.038 & 0.250 & 0.038 \end{bmatrix}$$

Initialise $\mathbf{P}^{(0)}$:

$$\mathbf{P}^{(0)} = \begin{bmatrix} 0.250 \\ 0.250 \\ 0.250 \\ 0.250 \end{bmatrix}$$

Then perform the first iteration:

$$\mathbf{P}^{(1)} = \widehat{\mathcal{M}}\mathbf{P}^{(0)} = \begin{bmatrix} 0.250 & 0.463 & 0.250 & 0.321 \\ 0.250 & 0.038 & 0.250 & 0.321 \\ 0.250 & 0.463 & 0.250 & 0.321 \\ 0.250 & 0.038 & 0.250 & 0.038 \end{bmatrix} \begin{bmatrix} 0.250 \\ 0.250 \\ 0.250 \\ 0.250 \end{bmatrix} \approx \begin{bmatrix} 0.321 \\ 0.215 \\ 0.321 \\ 0.144 \end{bmatrix}$$

The initialisation and first iteration result in the following values for $\mathbf{P}$:

t	A	B	C	D
0	0.250	0.250	0.250	0.250
1	0.321	0.215	0.321	0.144

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

$$\begin{aligned} ||\mathbf{P}^{(1)} - \mathbf{P}^{(0)}|| &= ||\{0.071, -0.035, 0.071, -0.106\}|| \\ &= \sqrt{0.022543} \\ &\approx 0.150 \end{aligned}$$

Since $0.150 \not\leq 0.005$ (ϵ), we perform another iteration.

$$\mathbf{P}^{(2)} = \widehat{\mathcal{M}}\mathbf{P}^{(1)} = \begin{bmatrix} 0.250 & 0.463 & 0.250 & 0.321 \\ 0.250 & 0.038 & 0.250 & 0.321 \\ 0.250 & 0.463 & 0.250 & 0.321 \\ 0.250 & 0.038 & 0.250 & 0.038 \end{bmatrix} \begin{bmatrix} 0.321 \\ 0.215 \\ 0.321 \\ 0.144 \end{bmatrix} \approx \begin{bmatrix} 0.306 \\ 0.215 \\ 0.307 \\ 0.174 \end{bmatrix}$$

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	0.321	0.215	0.321	0.144
2	0.306	0.215	0.306	0.174

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

$$||\mathbf{P}^{(2)} - \mathbf{P}^{(1)}|| \approx 0.036 \not\leq \epsilon$$

Convergence has not been reached, so we continue with the algorithm.
After another two iterations, we have finally reached convergence:

t	A	B	C	D
0	0.250	0.250	0.250	0.250
1	0.321	0.215	0.321	0.144
2	0.306	0.215	0.306	0.174
3	0.308	0.217	0.308	0.167
4	0.308	0.216	0.308	0.168

$$||\mathbf{P}^{(4)} - \mathbf{P}^{(3)}|| \approx 0.001 \leq \epsilon$$

At this point the algorithm terminates and the values of $\mathbf{P}^{(4)}$ are the final PageRank scores for each of the pages. If this were a query, the resulting ranks would be A, C, B, D where we arbitrarily rank page A before page C since they have the same score.

Implementation details

Your program must implement the PageRank algorithm using the power method described above.

The header file `pagerank.h` contains an `init` function that processes the list of pages and edges from standard input and stores a linked list of pages and corresponding inlinks in the following structs:

```
struct page {
    char* name;        // name of this page
    size_t index;      // index of this page
    size_t noutlinks; // number of outlinks from this page
    node* inlinks;     // linked list of pages with inlinks to this page
};

struct node {
    page* page; // pointer to the page data structure
    node* next; // pointer to the next page in the list
};
```

Your program must produce no errors when built on the lab machines and ed with the command:

```
$ clang -O1 -std=gnu11 -march=native -lm -pthread pagerank.c -o pagerank
```

Your program output must match the exact output format shown by the prototype and on Ed. You are encouraged to submit your assignment while you are working on it, so you can obtain some feedback.

Program input

```
<dampening factor>
<number of pages>
<name of web page>
...
<number of edges>
<source page> <destination page>
...
```

The first line specifies the dampening factor to be used by the program. The second line defines the number of pages to be declared, followed by a list of page names with one name per line. Then, the number of edges in the graph is specified, followed by a list of edges with one edge per line.

The `init` function reads the input and terminates outputting an error message if the input is invalid. The input is considered invalid if a page name exceeds the maximum length, the dampening factor is not in the range $0 \leq d \leq 1$, a page is declared twice, or an edge is defined to a nonexistent page.

Example

```
0.85
4
A
B
C
D
5
D A
D B
D C
B A
B C
```

This example has four web pages with names: A, B, C, and D
There are five edges: $D \to A$, $D \to B$, $D \to C$, $B \to A$, and $B \to C$

The output is the list of scores for each web page, for example:

```
A 0.30791363
B 0.21580945
C 0.30791363
D 0.16836329
```

The scores are ordered in the same order they were defined in the input.

For printing the score, use the format string `"%s %.8lf\n"`
For all test cases set $\epsilon = 0.005$, use the constant defined as **`EPSILON`**

Submission details

Final deliverable for the performance is due in week 13. Further details will be announced on Ed.

You must submit your code using the assignment page on Ed. To submit, simply place all of your files and folders into the workspace, click run to check your program works and then click submit.

You are encouraged to submit multiple times, but only your last submission will be marked.

Marking

In this assignment you will receive marks for correctness and performance of your program. However, submissions are only eligible for the performance component if they pass all of the correctness tests.

5 **marks** are assigned based on automatic tests for the *correctness* of your program.
This component will use our own hidden test cases that cover every aspect of the specification. To pass test cases your solution must produce the correct output within the imposed time limit.

5 **marks** are assigned based on the *performance* of your code relative to the benchmark.
This component is tested on a separate machine in a consistent manner. Submissions faster than the benchmark implementation will receive 5 marks, with successively slower ones receiving lower marks. Submissions faster than our basic implementation will receive at least 2.5 marks.

Your program will be marked automatically, so make sure that you carefully follow the assignment specifications. Your program must match the exact output in the examples and the testcases on Ed.

Warning: Any attempts to deceive or disrupt the marking system will result in an immediate zero for the entire assignment. Negative marks can be assigned if you do not properly follow the assignment specification, or your code is unnecessarily or deliberately obfuscated.