COMP2129   Assignment 4

Task description

In this assignment we will implement the PageRank algorithm in the C programming language using parallel programming and code optimisation techniques to ensure that 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:

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

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.

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

During each iteration of the algorithm, the value of $\mathbf{P}$ is updated 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 algorithm terminates when PageRank scores stop varying between iterations. The PageRank scores have converged when the following condition is met:

$$||\mathbf{P}^{(t+1)} - \mathbf{P}^{(t)}|| \le \epsilon \tag{3}$$
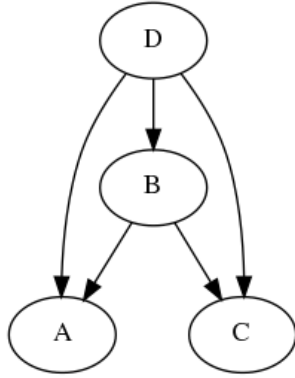
The vector norm is defined to be the standard Euclidean vector norm; that is, for some vector $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$:

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

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.

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

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.

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

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

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

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

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

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	0.214	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 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.

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

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

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

$$\mathbf{P}_D^{(3)} = \frac{1-0.85}{4} + 0.85 \, (0) = \frac{0.15}{4} + 0 \approx 0.038$$

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.

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

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

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

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

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

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

We have now reached convergence, so 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 task is to implement the PageRank algorithm as described above. The header file `pagerank.h` contains an initialisation function which processes the list of pages and edges from standard input and stores a linked list of pages and their corresponding inlinks in the following data structures:

```
struct page {
  char* name;      // name of this page
  int index;       // index of this page
  int 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 -O0 -std=gnu11 -march=native -lm -lpthread 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: $D \rightarrow A$, $D \rightarrow B$, $D \rightarrow C$, $B \rightarrow A$, and $B \rightarrow C$

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

```
A 0.0686
B 0.0481
C 0.0686
D 0.0375
```

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

For printing the score, use the format string `"%s %.4lf\n"`
For all test cases set $\epsilon = 0.005$, use the constant defined as `EPSILON`

Running the prototype

We have provided you with a prototype implemention of the PageRank algorithm that is accessible from the lab machines and undergraduate servers by running:

```
$ /labcommon/comp2129/bin/pagerank
```

We suggest that you experiment with the prototype implementation, and also run and compare your own test cases against it. Your program must match the output of the prototype for all valid input.

Submission details

You must submit your code in the assignment page on ed. To submit, simply place your code into the code editor, click the run button to check your program works and then click the submit button.

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

Marking

In this assignment correctness and performance are equally weighted. However, your submission will only be eligible for the performance component if it produces the correct output for all valid inputs.

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. Any submission that is faster than the benchmark implementation will receive 5 marks, with successively slower solutions receiving lower marks.

Your program will be marked automatically, so make sure that you carefully follow the assignment specifications. Your program must match the exact output of the prototype and the test cases 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.