CS3100/5100: Programming Assignment #3: Link Analysis (Code due at 11am on Nov 26, report due in the class of Nov 26)

1 Definitions

We will be working on the link structure of a set of web pages. Figure 1 shows an example.

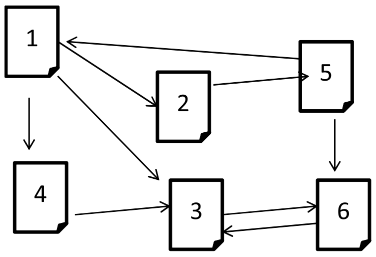


Figure 1: Web pages and hyperlinks.

The link structure is stored in a file. The file starts with the number of vertices, N, at the first line, followed by the directed links of the link structure. We assume the vertices are labeled with numbers 1,2,...,N. Each link $i \rightarrow j$ is represented with a pair of vertices ”i,j” in the file. For example, the above link structure is stored as

```
6
1,2
1,4
2,5
3,6
4,3
1,3
6,3
```

```
5,6
5,1
```

Two sample graph files in this format are provided for your experiments.

2 PageRank Algorithm

The PageRank algorithm is used to rank the global importance of a web page in the entire web. It uses the random walk model to simulate how users surf the Web. It tries to derive the probability that he/she can reach any page X_i by randomly starting with a page in the Web. The original PageRank algorithm, patented by Google, is described by the following equation.

$$PR(X_i) = (1-d)/N + d\sum_{j=1}^{m} \frac{PR(X_j)}{C(X_j)}$$

where

- d is a damping factor which can be set between 0 and 1 (0.85 by default), and N is the total number of pages in the web. $(1-d)/N$ means the probability of randomly choosing a page in the web. These parameters are given in advance.
- $PR(X_i)$ is the PageRank of the page X_i, i.e., a node in Figure 1.
- $PR(X_j)$ is the PageRank of pages X_j which has outbound links pointing to the page X_i. There are m pages having links pointing to X_i. m varies for different pages.
- $C(X_j)$ is the number of total outbound links on the page X_j, for example, page 1 has three outbound links.

There are some features you may want to know. First of all, the PageRank of the page X_i is recursively defined by the PageRanks of other pages that have links pointing to the page X_i. Second, the pages $\{X_j\}$ do not influence the PageRank of page A uniformly, but reversely weighted by the number of outbound links $C(X_j)$ on page X_j. This means that the more outbound links a page X_j has, the less (probability) it will contribute to X_i's PageRank. Finally, any pages that have links to X_i will more or less contribute to X_i's PageRank. Thus, any additional inbound link for page X_i will always increase page X_i's PageRank[1].

Calculating PageRank. In calculating PageRank, you can use an iterative method. Let the superscribe indicate the number of iteration. In iteration 0, you initialize the initial

[1]That is the reason spammers tried the "link farm" method to increase the PageRank of their pages. Google had to find ways to identify and eliminate the PageRank contributions from link spams.

PageRank for all pages as an array $PR^{(0)} = \{PR^{(0)}(X_1), PR^{(0)}(X_2), \ldots, PR^{(0)}(X_N)\}$ with some values (It is to be investigated whether different initialization methods have different effects). At iteration i, $i > 0$, each element in the PageRank array $PR^{(i)}$ is calculated based on the elements in $PR^{(i-1)}$. Use the following formula

$$PR^{(i)}(X_i) = (1-d)/N + d \sum_{j=1}^{m} \frac{PR^{(i-1)}(X_j)}{C(X_j)}$$

It has been proved that the iterations will converge - starting from certain iteration, the pagerank of each page will not change from the last iteration. You may consider using two arrays to keep $PR^{(i)}$ and $PR^{(i+1)}$ in your implementation.

3 What you need to do?

Similar to previous, the skeleton code is provided at /common/users3/group44/w033kxc/project3. This directory also includes the two sample graphs graph1.txt and graph2.txt represented in the described format. Please copy the whole directory to your directory.

You will need to finish the following tasks.

Task1 Read the input file and store the graph into an adjacency list. Use the command line

```
./p3 -i graph_filename -o exp1
```

to print out the adjacency structure in the following format. The numbers, starting from 1, denote the ID of the corresponding vertex, e.g.,
1 $->$ 2,3,4
2 $->$ 3,5
...

Task 2 Implement the PageRank algorithm: WebGraph::page_rank(d, i) with the adjacency list data structure, where d is the damping factor, $0 < d < 1$, and i is the maximum number of iterations. This function will print out the PageRank for each vertex in the end.

Try a number of iterations on sample graphs to learn the number of iterations that the PageRank values will converge. The convergence is defined by the change between two consecutive iterations as follows

$$\mu_i = \max\{|PR^{(i)}(X_1) - PR^{(i-1)}(X_1)|, |PR^{(i)}(X_2) - PR^{(i-1)}(X_2)|, \ldots, |PR^{(i)}(X_N) - PR^{(i-1)}(X_N)|\}.$$

If μ_i is less than some tiny value, e.g., 0.0001, then we can say the PageRank algorithm converges. Set the number of iterations to a sufficiently large value, e.g., 50. Save the μ_i value from each iteration for the report. Once the algorithm converges or the maximum number of iteration is reached, WebGraph::page_rank terminates.

Use the command line

```
./p3 -i graph_file -o exp2
```

to run Task 2.

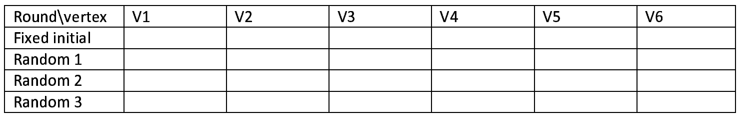
Task 3 Try two different PageRank initialization methods and observe the results. (1) Use $1/N$ to initialize the PageRank values. (2) Use a random value in (0,1) to initialize the PageRank values. Observe the final PageRank values after the algorithm converges. Use the command line

```
./p3 -i graph_file -o exp3
```

to output the results.

4 Deliverables and grading

1. the source code should be well commented and compilable with g++ on unixapps1;
2. the report should briefly describes how you implement the algorithms. Analyze the time complexity of your implementation with the big O notation.
3. the report should include the result of Task 2 and plot the changes for **both** sample graphs. The x-axis represents the number of iterations; the y-axis represents the change. Try different damping factors, e.g., 0.5 and 0.85. Discuss what you have observed.
4. report the result of Task 3 for graph1.txt. Use a sufficiently large number of iterations that leads to the convergence, which is observed in Task 3. First, use the fixed initial values. Then, repeat the random initialization experiments for *three* times, and report the results in the following table, where V_i represents a vertex. Discuss what you have observed.



Round\vertex	V1	V2	V3	V4	V5	V6
Fixed initial						
Random 1						
Random 2						
Random 3						

Figure 2: Sample Task 4 output.

The source code should be zipped into one submission file and submitted to Pilot. You should turn in the report in the class.