Data Structures

Fall 2020, Programming Assignment #1

1 Introduction

PageRank is an algorithm used by Google Search to rank websites in their search engine results. PageRank was named after Larry Page, one of the founders of Google. PageRank is a way of measuring the importance of website pages. According to Google:

PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.

It is not the only algorithm used by Google to order search engine results, but it is the first algorithm that was used by the company, and it is the best-known.

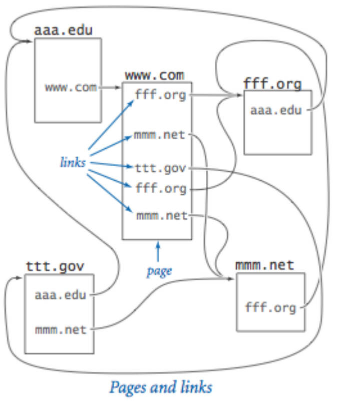

Figure 1: A simple web graph.

Quoting from the original Google paper, PageRank is defined like this:

> We assume page A has pages $T_1, ..., T_n$ which point to it (i.e., are citations). The parameter d is a damping factor which can be set between 0 and 1. We usually set d to 0.85. Also $C(X)$ is defined as the number of links going out of page X. The PageRank of a page A is given as follows:
>
> $PR(A) = (1-d)/N + d * (PR(T_1)/C(T_1) + ... + PR(T_n)/C(T_n))$, where N is the total number of pages,
>
> Note that the PageRanks form a probability distribution over web pages, so the sum of all web pages' PageRanks will be one.

See Figure 2 for an example.

2 Assignment

Your project is to build a simple search engine. Here is what you must do:

You will be given a zip file which contains 500 files named page0 through page499. Each page contains a list of other pages which it points to (there may be anywhere from zero to three hundred of these, pages may refer back to themselves). Following this there will be a row of hyphens (always the same amount) and then a list of 20 words which the page contains (all pages will contain 20 words).

$$\begin{tikzcd}
A \arrow[r] \arrow[d] \arrow[dr] & B \arrow[l, bend right] \arrow[d, bend left] \\
C \arrow[loop left] & D \arrow[u] \arrow[l]
\end{tikzcd}$$

$$\mathbf{v}' = \begin{bmatrix} 0 & 2/5 & 0 & 0 \\ 4/15 & 0 & 0 & 2/5 \\ 4/15 & 0 & 4/5 & 2/5 \\ 4/15 & 2/5 & 0 & 0 \end{bmatrix} \mathbf{v} + \begin{bmatrix} 1/20 \\ 1/20 \\ 1/20 \\ 1/20 \end{bmatrix}$$

$$\begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}, \begin{bmatrix} 9/60 \\ 13/60 \\ 25/60 \\ 13/60 \end{bmatrix}, \begin{bmatrix} 41/300 \\ 53/300 \\ 153/300 \\ 53/300 \end{bmatrix}, \begin{bmatrix} 543/4500 \\ 707/4500 \\ 2543/4500 \\ 707/4500 \end{bmatrix}, \ldots, \begin{bmatrix} 15/148 \\ 19/148 \\ 95/148 \\ 19/148 \end{bmatrix}$$

Figure 2: An example with $d = 0.8$.

From these you need to create a program that produces two output files - a page rank file named PageRank and a reverse index file named ReverseIndex. Your program will then enter a loop which takes as inputs a list of one or more words and then returns the top ten hits (list of pages) for those words. If only one word is used, then the program should print the top ten pages for that word. If multiple words are used, then the program should print two sets of output - the top ten pages that contain all the words (AND semantics), and the top ten pages that contain any of those words (OR semantics).

Your program should loop until the input is *end* (including the asterisks).

Your program will take two command line arguments - the page rank stopping difference and the "d" value, in that order, details below.

Please note that there is one special page – it is page500. This is a page that is pointed to in the crawl, but for which there is no page file. Consider this page as if it was an image or other such document. Thus, it does participate in the computation, but it does not itself link out to anything. To handle this page you need to

- save page500 in your structure
- calculate its pagerank (like all other pages)
- include it in the pagerank list

(note that this means there are 501 pages in the system – i.e. that N=501.)

Some Other Details

Note: the search should be case sensitive, and punctuation counts (i.e. "don't" is not the same as "dont") – in essence, use string equality.

When you produce the output, please use ASCII order (i.e. uppercase letters before lower case).

Here is the pseudo-code of the PageRank algorithm:

```
PageRank(page(0..$n-1$), $d, DIFF$);
1    INITIALIZE:
     $PR(i) = 1/N$ for all $i$;
2    diff = 0;
     For each $P$ in Page
```

```
        PRbefore= $PR(P)$;
        $PR(P) = (1-d)/N + d * ((PR(t_1)/CR(t_1) + PR(t_2)/CR(t_2)...)$
                for all $t_i$ that point to $P$
                ($CR(t_i)$ is the outbranching of $t_i$)
        $diff = diff + AbsoluteValue(PRbefore - PR(P);$
3   if $diff >= DIFF$ then goto 2
4   done.
```

1) Page Rank list

Your program will take two arguments, the stopping difference and the "*d*" value, and compute the page rank for all 500 pages, stopping when the cumulative *diff* for all 500 goes below that number (a real number greater than zero) - note, be careful when debugging - for some settings of these two parameters your program could loop forever (we'll provide some useful values for debugging).

Your output should be a file with a list of the pages in page rank order (highest to lowest), for each page listing its outbranching (how many pages it connects to) and the page rank for that page (to 8 significant digits).

Example

```
page500    137    .0023781
page503    196    .0022139
page1500   7      .0021101
```

2) Reverse index

Your program should also produce a file which shows its reverse index. That is, for every word (note: including commonly used ones) you should output the word followed by a list of the pages that include that word (please alphabetize the word list, the pages can be produced in any order you choose - but each page should only appear once!).

Example

```
a         page500 page522 page916 page803
an        page999 page921 page998 page763 page1227 page501 page937 page862 page500
          page522 page916 page803 page919 page997
at        page900 page2000 page1227 page501 page937 page862
zuchini   page1000
```

3) Search engine

Given the data structures containing the above, your search engine is quite simple. You take as input a list of words and you output the top ten pages (sorted by page rank) that contain them (if less than 10, then print all of them). When a single word is entered, just output the pages for that word. When multiple words are entered, output two lists - the top ten pages with all the words and the top ten pages with any of the words (multiple words do not effect the ranking here).

Example

```
Enter Word: moose
page1000 page2001 page2000


Enter Word: moose cow
AND (moose cow) page1000
OR (moose cow) page1000 page2006 page2001 page1983 page776 page842 page777 page2000 page963
page871

Enter Word: *end*
```

NOTE To compute the PageRank for a large graph representing the Web, we have to perform a matrixVvector multiplication many times, until the vector is close to unchanged at one iteration. Since the transition matrix of the Web M is very sparse, representing it by all its elements is highly inefficient. Rather, we want to represent the matrix by its nonzero elements.

An example of a page file:

page351 page352 page119 page354 page449 page445 page111 page440 page117 page114 page115 page219 page191 page196 page197 page195 page213 page215 page214 page290 page292 page65 page297 page299 page298 page207 page447 page472 page475 page362 page163 page160 page147 page58 page288 page286 page287 page284 page51 page273 page378 page275 page373 page279 page278 page178 page177 page172 page48 page47 page46 page44 page312 page260 page454 page265 page266 page268 page269 page149 page148 page140 page491 page497 page144 page301 page494 page383 page381 page466 page69 page118 page460 page36 page35 page33 page31 page90 page154 page94 page95 page97 page98 page257 page256 page158 page489 page319 page306 page480 page316 page484 page143 page486 page397 page478 page331 page399 page411 page28 page415 page25 page27 page227 page416 page123 page87 page86 page241 page83 page328 page321 page320 page323 page327 page401 page402 page404 page409 page15 page17 page336 page132 page335 page137 page138 page9 page205 page239 page238 page4 page5 page0 page230 page464 page438 page436 page433 page105 page456 page107 page106 page101 page452 page185 page186 page180 page224 page209 page189 page221 page73 page71 page422 page427
———————— her clearing instrument of singular every lenses and He the him at which false successfully to the and In Street