CS 667 : Homework 5(Due: Dec 2, 2009)

Problem 1. (50 points)

Find the page rank of the graphs of Figures 1 and 2. Use the algorithm of page 14 of Subject 8. Initial values are $1/N$. Also use $a = (d =) = 0.85$ in the page rank formula. Iterate as many times as needed for the error to be less than 10^{-4}. Use the synchronous update version.

Is there any difference if you use an asynchronous approach (say a lower indexed vertex gets updated before a higher indexed one) where as long as a rank if computed for a given vertex than rank is being used for the rank computation of other vertices pointed by that vertex? Explain.

Problem 2. (50 points)

Find the hub/authority rank of the graphs of Figures 1 and 2. Use initial values both 1 and $1/N$. Repeat as many times as long as the error is less than 10^{-4}. This is page 4 of Subject 8 with line 4 modified as needed to accommodate both cases.

Problem 3. (50 points)

The prime number theorem (Theorem 31.37) states that the number of primes less than x is about $x/\ln x$. Therefore the density of primes is $1/\ln x$. In other words if we pick uniformly at random an integer z between 1 and N with probability $1/\ln N$ we choose a prime number and with probability $1 - 1/\ln N$ we choose a composite number. Let $P = P_m \ldots P_1$ be an m-bit integer. Let $Q = Q_m \ldots T_1$ be an m-bit integer. Then $P = P_m 2^{m-1} + \ldots + P_1$ and $Q = Q_m 2^{m-1} + \ldots + Q_1$.

(a) Is $P \leq 2^m$? Explain. Is $Q \leq 2^m$? Explain.

(b) If $p/(P-Q)$ then show that $P \bmod p \equiv Q \bmod p$. Then show that if $P \bmod p \equiv Q \bmod p$, the $p/(P-Q)$.

(c) Show P or Q can have at most m distinct prime divisors.

(d) Let us pick a random number prime number p between 1 and $R = 4n^2 m \ln 4n^2 m$, where $n > m$. Show that such a p divides $P - Q$ with probability at most $O(1/n^2)$.

Problem 4. (50 points)

This is a continuation of the previous problem. In Rabin-Karp let P be the pattern of length m and T be the text of length n. Choose prime R so that $R = \Theta(4n^2 m \ln(4n^2 m))$; R is the prime number used to perform fingerprint computations modulo R.

(i) Show that the probability a spurious hit occurs in position i of the text is $O(1/n^2)$.

(ii) Show that the probability a spurious hit occurs anywhere in the text is $O(1/n)$.

(iii) Show that the expected running time of the algorithm is $O(n)$.

Problem 5. (50 points)

(a) Give an input T, P for which NaiveMatching exhibits worst-case performance over a binary alphabet of letters a, b.

(b) Give an input T, P for which Boyer-Moore exhibits worst-case performance over a binary alphabet of letters a, b.

(c) How many comparisons does BM perform if $P = a^m$ and $T = (a^{m-1}b)^{n/m}$. Assume m/n and a^m means m consecutive occurrences of a.

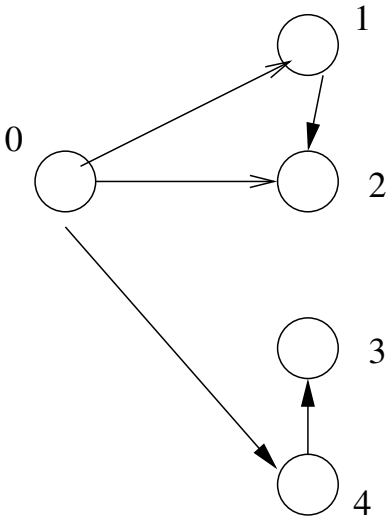

Figure 1.

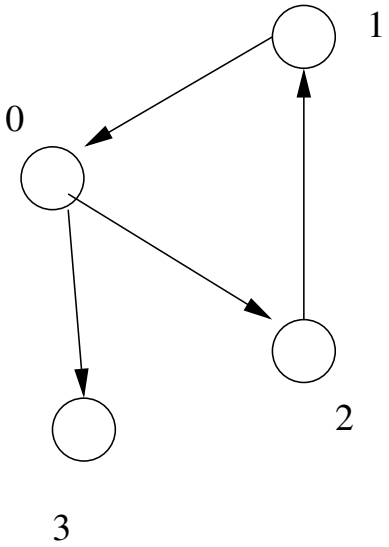

Figure 2.

Problem 6. (75 POINTS)

Implement the HITS and PageRank algorithms. The inputs will be graphs represented through an adjacency list.

The command-line interface would be as follows.

```
% ./rank  Algorithm   InitialValue Iterations InputFile
% java rank Algorithm InitialValue Iterations InputFile
```

The command-line parameter `Algorithm` takes one of two values: `0` if Kleinberg's HITS is used (witht he scaling as otherwise shown on page 4 of Subject 8) and `1` if the Brin and Page's PageRank algorithm is used (as shown on page 14 of Subject 8). The second parameter `InitialValue` indicates how the initial values for the ranks will be computed. If it is `0` all ranks are initialized to 0, if it is `1` they are initialized to 1. If it is `2` they are initialized to $1/N$, where N is the number of web-pages (size of the graph.) If the value is a numeric integer value other than 0,1,2 then the ranks are initialized as `InitialValue` divided by 100. Thus an `InitialValue` equal to 50, initializes all ranks to $50/100 = 0.5$. Parameter `Iterations` runs the algorithms for that number of iterations. Parameter `InputFile` describes the input graph and it has the following form. The first line contains two numbers: the number of vertices (in the example below, this is equal to five) and the number of edges that follow on separate lines (i.e. six). In each line an edge (i, j) is presented by `i j`. The graph used in class in a lecture will be represented as follows. (Note that the graphs in class have vertices in the range $1..n$, whereas in this implementation, it is $0..n-1$.

```
5 6
0 3
0 4
1 3
2 3
2 4
3 0
```

```
Kleinberg  might report, at the  10-th iteration, Authority/Hub pair values of
  0.00000/0.65719 0.00000/0.36905 0.00000/0.65719 0.78821/0.00000 0.61541/0.00000
and Google
  0.14131         0.03000         0.03000         0.12726         0.10176
```

Note. Do not forget to use scaling in HITS.

Problem 7. (100 POINTS)

Implement the NaiveAlgorithm, RabinKarp, BoyerMoore, and KMP by providing C/C++/Java implementations **consistent with the descriptions of these algorithms provided in class and in the available notes**. For example implementing descriptions or pseudocode for RabinKarp or any of the other algorithms found on the Web will be rejected. Your implementation should report ALL hits. String searches should be case insensitive. Then test, your algorithms on texts available at the Project Gutenberg web-site. Some texts that you should test your algorithms on are

```
1. US Copyright Law at
   http://www.gutenberg.org/files/4291/4291.txt

2. George Bernard Shaw's  "Caesar and Cleopatra".
   http://www.gutenberg.org/files/3329/old/candc10.txt

3. Homer's Iliad
   http://www.gutenberg.org/dirs/etext00/iliad10.txt

4. Human Genome Project, Chromosome 3,
   http://www.gutenberg.org/dirs/etext00/03hgp10.txt
```

including the preamble.

Search 4-letter, 8-letter and 16-letter strings that appear and do not appear in the text and report the running times for finding all occurrences in a tabular form and the number of successful hits. In addition search the strings "notwithstanding" and "copyright" in the first text, "Caesar" and "Cleopatra" in the second, and "Achilles" and "Argives" in the third, and "TTTTTGGGGG" and "AAAAAAAACCCC" in the fourth, and report the number of hits in addition to running times.

You need to provide the following interface

```
% java  strmatch   XXX file-name

 or
%  ./strmatch XXX file-name  pattern
```

`XXX` is one of `NAL, KMP, RK, BM` and indicates the corresponding algorithm i.e. the naive, Knuth-Morris-Pratt, Rabik-Karp or Boyer-Moore as explained in the Notes (or the textbook). Parameter `file-name` is an arbitrary string that indicates the file that will be used as input. Parameter `pattern` is the string that will be searched in `file-name`. The output should be of the following form.

```
 No instance of  pattern was found in file-name

or

 1. An instance of pattern was found at position 19
 2. An instance of pattern was found at position 192
 3. An instance of pattern was found at position 1922

 A total of 3 instances of pattern were found.
```

depending on whether `pattern` exists or not in `file-name`. Positions are indexes in the file. Local copies of the text files will also be made available through the protected area of the course web-page (hints).