COS 435, Spring 2012 - Problem Set 3

Due at 1:30pm, Wednesday, March 7, 2012.

Collaboration and Reference Policy

You may discuss the general methods of solving the problems with other students in the class. However, each student must work out the details and write up his or her own solution to each problem independently.

Some problems have been used in previous offerings of COS 435. You are NOT allowed to use any solutions posted for previous offerings of COS 435 or any solutions produced by anyone else for the assigned problems. You may use other reference materials; you must give citations to all reference materials that you use.

Lateness Policy

A late penalty will be applied, unless there are extraordinary circumstances and/or prior arrangements:

- No penalty if in Prof. LaPaugh's office or inbox by 5pm Wednesday (3/7/12).
- Penalized 10% of the earned score if submitted by 11:59 pm Wed (3/7/12).
- Penalized 25% of the earned score if submitted by 5pm Friday (3/9/12).
- Penalized 50% if submitted later than 5pm Friday (3/9/12).

Problem 1 (2011 exam problem):

For this problem on PageRank, calculations can easily be done by hand. (You may write a self-contained program in Java to do this calculation, but it is probably easier to do the calculation by hand. If you write a program, turn in the source code. You may **no**t use code written by someone else.)

Part A: Consider the 4-node graph consisting of two 2-cycles: {(A,B), (B,A)} and {(C,D), (D,C)} connected by cross-edges (A,C) and (B,D), as shown below.
Show two iterations of the PageRank calculation on this graph with initial PageRank values of ¼ for all nodes and α=0. Based on the pattern of the two iterations, what four values (one for each node) do you think the PageRank calculation converges to?

Part B: For $\alpha=0$, different results from those of Part A are obtained if one executes the iterative PageRank calculation with initial values of 1 for node C and 0 for the other nodes. What are the results of the calculation with these initial values? Why do they differ?

Part C: Using the steady state PageRank equations, show that for any α, $0<\alpha<1$,

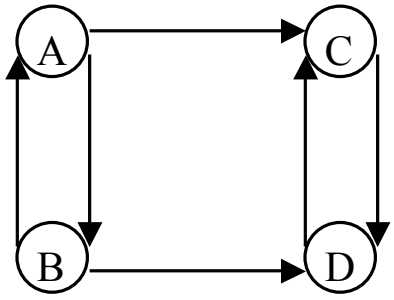
$$\text{PageRank}(A) + \text{PageRank}(B) = \alpha / (1+\alpha)$$
$$\text{PageRank}(C) + \text{PageRank}(D) = 1 / (1+\alpha)$$

Steady state equations, NOT iterative:

$$\text{PageRank}(A) = \alpha/4 + (1-\alpha)*1/2*\text{PageRank}(B)$$
$$\text{PageRank}(B) = \alpha/4 + (1-\alpha)*1/2*\text{PageRank}(A)$$
$$\text{PageRank}(C) = \alpha/4 + (1-\alpha)*(1/2*\text{PageRank}(A) + \text{PageRank}(D))$$
$$\text{PageRank}(D) = \alpha/4 + (1-\alpha)*(1/2*\text{PageRank}(B) + \text{PageRank}(C))$$

Part D: Consider the extremes as α approaches 0 and as α approaches 1. What do the PageRanks of the four nodes approach in each case. Interpret this in terms of the random walk with random jumps model.

Graph for Problem 2



Problem 2

Consider an inverted index containing, for each term, the postings list (i.e. the list of documents and occurrences within documents) for that term. The postings lists are accessed through a B+ tree with the terms serving as search keys. Each *leaf* of the B+ tree holds a sublist of alphabetically consecutive terms, and, with each term, a *pointer to* the postings list for that term. (See the "blow-up of B^+ tree example" posted on the Schedule and Assignments page under Feb. 27.)

Part a. Suppose there are 9 billion terms for a collection of 10 trillion documents of total size 100 petabytes. We would like each internal node of the B+ tree and each leaf of the B+ tree to fit in one 32 kilobyte page of the file system. Recall that a B+ tree has a parameter **m** called the *order* of the tree, and each internal node of a B+ tree has between **m+1** and **2m+1** children (except the root, which has between 2 and **2m+1**). Assume that each term is represented using 28 bytes, and each pointer to a child in the tree or to a postings list is represented using 8 bytes. Find a value for the order **m** of the B+ tree so that one 32 kilobyte page can be assigned to each internal node and leaf, and so that an internal node will come as close as possible to filling its page without overflow its page when it has **2m+1** children. If you need to make additional assumptions, state what assumptions you are making.

Part b. For your **m** of Part a, estimate the height of the B+ tree for the inverted index of the collection described in Part a. (Giving a range of heights is fine.) Also estimate the amount of memory needed to store the tree, including leaves but not including the postings lists themselves.

Part c. Estimate the aggregate size of the postings lists.

Problem 3

Consider again building a B+ tree on the collection described in Problem 2. However, now use compression on the list of terms in each B+ tree node. Apply the dictionary compression technique that uses one string of all the terms and pointers into the string (as discussed in class); this is called "dictionary as string" in our textbook *An Introduction to Information Retrieval* (Section 5.2.1). The goal is to maximize the number of terms that can fit in a node using this compression.

As in Problem 2, there are 9 billion terms for a collection of 10 trillion documents of total size 100 petabytes. Again, each internal node and leaf must fit in a 32 KB page. Each term is no longer represented using 28 bytes, but as a pointer into the string. Each pointer to a child in the tree or to a postings list is still represented using 8 bytes, but you must determine how many bytes to allocate for the pointer into the string. You may assume that the average length of a term is 8 characters and that a character is represented in one

byte. Both the pointer structure and the string must be stored within the 32 KB of a node. The string stored in a node need only contain the terms used in that node. The figure below illustrates the structure.

Part a. Determine the size in bytes of a pointer into the string. This pointer should be as small as possible while still providing enough address space to refer to any character in the string. The size of the pointer must be constant across all nodes of the B+ tree. Justify your choice.

Part b. Find a value for the order **m** of the B+ tree when this compression is used in each node (internal and leaf). As in Problem 2, the goal is to determine the value of **m** so that an internal node will come as close as possible to filling its page without overflow its page when it has **2m+1** children.

Structure of an internal B+ tree node:
Occupies one 32KB page.