COS 435, Spring 2015 - Problem Set 4

Due at 1:30PM, Wednesday, April 1, 2015.

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. For each problem, list the students with whom you discussed general methods of solving the problem (excluding very brief casual conversations).

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:

- Penalized 10% of the earned score if submitted by 11:59 pm Wed. (4/1/15).
- Penalized 25% of the earned score if submitted by 4:30pm Friday (4/6/15).
- Penalized 50% if submitted later than 4:30 pm Friday (4/3/15).

Problem 1 (similar to a 2011 exam 2 problem)

Recall that skip pointers can be used to speed up query evaluation by allowing the algorithm that executes the intersection of postings lists for the different terms of the query to skip sections of a postings list when then next document on one of the other postings list has a much higher docID. This question asks you to estimate the savings in space if the skip pointer representation is combined with the compressed representation of docIDs using gaps.

Assume that a list containing L postings uses floor(sqrt(L))-1 skip pointers that are approximately evenly spaced, starting at the first posting, so that each skip bridges about sqrt(L) postings.

For a collection of one hundred billion documents, and postings that are pairs (DocID, term frequency), let the representation of one posting in a postings list, using ***no*** compression, be one of the following two forms:

form of posting when there is a skip pointer:
| docID | | skip pointer | | term frequency |
5 bytes 3 bytes 2 bytes

form when there is no skip pointer:
| doc ID | |term frequency |
5 bytes 2 bytes

Part A: Suppose we compress each postings list by representing each destination document of a skip pointer by the difference between its docID and the docID of the origin of the skip pointer (i.e. gap between docIDs). Also represent successive docIDs lying between two skip pointers by their successive gaps in docID (see the illustration of skip pointers in the Compression Summary, Part 2 posted under 3/2/15). All gaps should be represented using ***variable byte encoding***. Also use ***variable byte encoding*** to represent the skip pointer. Do not compress the term frequency. Estimate the space in bytes required for a postings list with this compression. Your estimate should be in terms of L. Your answer should be an estimate of the space used, but it will be graded on the ***quality and correctness of the estimate***, i.e. expect deductions for very coarse estimates.

Part B: For a list of one million postings, how much compression is being achieved with the representation of Part A in comparison to the representation without compression presented at the beginning of this problem?

Problem 2 (from a 2010 exam 2 problem)

Consider a Web crawler that uses F different priority levels for fetching URLs, based on the frequency of change of the URL. The crawler also uses different minimum delays between requests for different hosts. It will contact a host known to have a large capacity for handling requests more frequently than a host that has less capacity. For each host, h, that has been contacted, the earliest next contact time t_h is recorded. Assume the fetching priorities and minimum delays between requests are independent.

Part A: In the Mercator Web crawler, the URL frontier is managed by two sets of first in first out (FIFO) queues. Give an example in which a crawler must wait before fetching a URL even though there is a URL that could be fetched immediately in one of the front queues. Your example should show as much of the state of the front and back queues as necessary to make clear that the state is legal and crawler is waiting unnecessarily.

Part B: Consider replacing each FIFO **front** queue in the Mercator URL frontier with a priority queue that is sorted on earliest next contact time of the host of each URL in the queue. That is, the next element removed from the k^{th} front priority queues is the URL with the earliest next contact time among all the URLs with fetch priority k. Does this eliminate the situation that the crawler must wait before fetching a URL even though there is a URL that could be fetched immediately in one of the front queues? Does using such a priority queue for each of the F front queues cause new problems? Explain all your answers.

Part C: What information is needed to compute the earliest next contact times for all previously seen hosts? Where is this information stored? When is it updated? Be explicit.