Diego F. Rincon Santana
2/21/2016

# Homework 1

## Problem 1

**A.**

Processing posting lists in order does guarantee to minimize the amount of total work. The reason for this is that, if we have *n* lists, then if we intersect the two smallest lists the number of intermediate results will be no bigger than the smallest list. Always working around the smallest list reduces the amount of total work.

**B.**

The goal of the algorithm is to find the intersection of two postings lists. If you have a Hash-Table with the tuple <word, docID> and the length of the postings list as value, then you can adapt the algorithm to accept two terms as arguments instead of the postings list. Then get the smallest list and traverse that list. This is basically a Hash Join. Then for every document in the posting list, we check if the other term appears (if, so, record, otherwise move on). This way the algorithm will take O(min(x,y)) where x and y are the lengths of the posting lists.

## Problem 2

Indeed when you Google 212-998-3123 you get, among other results, the homepage of Prof. Davis, as well as other results that let us know that Google has correctly understood that the query was a phone number (e.g. phone archive results). The same results are thrown if you search for (212) 998 3123 or (212)9983123 or even 212 9983123.

The problem comes when you alter the search by a few characters. If you search for 212-9983-123 Google understands this as an arithmetical expression and outputs the result (-9894). And you get similar results when you don’t comply with the ###-###-#### format but still have three numbers and 2 hyphens. If you search for 2129983123 without the space between the first three digits and the rest, then Google doesn’t understand this as a phone number. More ambiguous are the results when you introduce the country code into the query. Searching for +1 212-998-3123 or 1 212-998-3123 doesn’t show Prof. Davis websites or courses among the results even though it recognizes it as a phone number as many of the results are phone related (the top one is NYU IT Service Desk).

The problem with tokenizing worldwide phone numbers is that they defy the usual regular expression that consists of 10 numbers. Usually the best bet is to exclude the country code from the query. The biggest issue is that worldwide telephone numbers may differ with this standard 10-digit regular expression. In this case it would be necessary to include additional reg. expressions to the ‘telephone’ category; however, this might get in the way of other numerical expressions that follow the same pattern but are not telephone numbers.

## Problem 3

**A.**

$F(x) = x^2/c$

Where x is the amount of change and c is a constant to be determined. Then a page that has seen a very small amount of changes will not need to be refreshed. The exponential factor makes it so that as the website changes the effect on the user's experience is affected exponentially making the need for a refresh more and more evident.

**B.**

The web crawler would need to infer the rate of change of a certain website based on how often the website is updated and information about the usual amount of change of a website of that *kind* (this could be the type of website—sports, news, education, etc—or simply historical data). The way the crawler would use this information in deciding on the refresh rate would be by combining the probability that the webpage has changed (say, in a day) with the average amount of change that that website usually has (during a day). Suppose the result of that is represented by *r* then you can use the following function $G(r) = c*r – r^2/c$ (where c is a constant) to determine whether to refresh or not. Since G has the shape of an upside-down parabola, if the website goes through lots of changes in high frequency, then it's no use in refreshing; if when it changes it changes greatly but it's updated every month, then the refresh rate will be low and so on.

**C.**

You could incorporate a measure of important vs unimportant changes. Important changes could be of structure and unimportant changes could be simple orthography corrections. You could place different weights on these changes: an important change might be worth 10x an unimportant change (or a nonlinear relationship that puts higher/lower weights). If you count anchor links as an important change then, since both functions grow exponentially, your chances of refreshing will also grow exponentially.