Topic: Google's original search engine architecture (and relevant paper)

**Rules.** This is to be completed no later than the start of class on the day it is due. One page is enough!

**Due Date:** No later than start of class on Mon Oct 17, 2011.

# 1 Objectives of the assignment

Read the paper that describes Google as it was originally designed and answer some questions on the paper (section C5, link L1, has the URL or use local copy `P1.GoogleBrinPage.pdf`). Subject 3 and a bit of Subject 4 might also prove useful as far as hash table are concerned, if you are not familiar with it from CS 114 or CS 435. Link `http://en.wikipedia.org/wiki/Hash_table` might also help.

# 2 Part 1 (100 points)

Answer the following questions after reading the paper. The answers you provide are and should be drawn from the paper and thus justification will be to paper-available information.

**(a)** What was the size (in bytes) of Google's dictionary/vocabulary around 1998? Justify your answer.

**(b)** What is the number of bits used for `docID` at that time (circa 1998)? Justify your answer with material from the paper.

**(c)** Describe the data structures used by Google for indexing only.

**(d)** Does Google (1998) use a hash table for the dictionary? What do they use? Why ?

**(e)** How many bytes are assigned to each hit (of the hitlist)? How many types of hits? What are they (types of hits)?

# 3 Part 2 (25 points)

You have 256MB of memory of which 30MB are being used by the operating system plus related programs. You are asked to organize the additional space to support a hash table along the lines of the paper where strings of words are stored in a contiguous table of characters delimited by a null character and the hash table itself stores in each entry a wordID identifying the corresponding word, plus some form of a pointer to the table of characters entry for the word. In an efficient implementation (a) how big would the hash table size $m$ be, (b) how many words $n$ can the scheme support, (c) how many bits for a `wordID`, (d) how many bits for the pointers, The total number of bits for (c) and (d) combined needs to be a multiple of 8 (byte aligned) but individual bit counts need not be (although if they are, it will count towards efficiency).