CS236620 – Introduction to Search Engine Technology

HW Assignment #2 – due January 10, 2016
(Please submit electronically and in pairs)

1. Lecture 6: in the following sub-questions, “size” refers to number of nodes.
   a. Show that the size of a Trie is not bounded by any function of the size of the vocabulary, i.e. construct arbitrarily small vocabularies that require arbitrarily large Tries.
   b. Prove that the size of a Pat-Tree is linear in the size of the vocabulary kept in the Pat-Tree.

2. Lecture 7: prove that when finding the top-r scores of a random permutation of M scores using a min-heap of size r, the expected number of required heap insertions in the last M-r steps is about \( r \log (M/r) \).

3. Lecture 8: when performing top-n retrieval over a locally partitioned index into m segments (assume \( m \geq n \) and that documents are indexed uniformly at random across the segments):
   a. What is the probability that retrieving \( k=1 \) results from each of the m segments would indeed collect all top-n results?
   b. What is the probability that retrieving \( k=n-1 \) results from each of the m segments would fail to collect all top-n results?

4. Wet assignment: download version 5.3.1 of the Lucene search library (http://lucene.apache.org/). Then, modify Lucene’s query evaluation code according to the requirement below. In addition to working Java code, please also submit:
   • A brief documentation of your changes/additions to Lucene.
   • A small collection of documents and a test program that runs some queries which demonstrate (e.g. by print-outs) that your code works. This should be accompanied by an explanation of why the test setup is convincing.

The required modification: modify Lucene’s TF/IDF scoring model so that a term’s frequency in a document (the TF) will be counted differently. Instead of every appearance of the term in the doc contributing 1 to its frequency, an appearance of the term in position \( j \) of a document (actually field) of length \( L \) should contribute \( (2L-j)/L \) to the frequency. Since Lucene’s term positions are numbered 1,…,L, this means that an appearance at the beginning of the document will be almost twice as important as an appearance at the end of the document (which adds exactly 1 to the frequency).

Note: the evaluation code is part of Lucene’s core, and is found in the package org.apache.lucene.search. You should read online documentation on Lucene’s scoring model to understand what you need to change. Please modify/add as little code as possible.