# COS 435, Spring 2016 - Problem Set 3

***Due 11:59 pm Wednesday March 2, 2016 by DropBox submission***
***Due at 5:00 pm, Wednesday, March 2, 2016 if submitting handwritten work on paper.***

---

## 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.

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 10am Thursday (3/3/16).
- Penalized 25% of the earned score if submitted by 4:30 pm Friday (3/4/16).
- Penalized 50% if submitted later than 4:30 pm Friday (3/4/16).

---

## Submission

Submit your solutions as a PDF file using the Computer Science Department DropBox submission system for COS435 at
https://dropbox.cs.princeton.edu/COS435_S2016/HW3 Name your file HW3.pdf. If you have not used this facility before, consult the instructions at
https://csguide.cs.princeton.edu/academic/csdropbox - student
Note that you are automatically enrolled in CS DropBox using the registrar's COS435 enrollment list.

You may hand write your solutions as long as they are legible. In this case, you may either scan your writing to produce a PDF file for submission through DropBox or turn in your document by 5:00 PM Wed. March 2 in the bin outside Prof. LaPaugh's office.

## Problem 1

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 $B^+$ tree example on slide 4 of the slides posted for Feb. 17.)

**Part a.** Suppose there are 14 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 (32*1024 byte) 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 40 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. Assume that leaves contain between **m** and **2m** terms just as internal nodes do.

## Problem 2

Section 4.3 and Figure 4.4 of our textbook *An Introduction to Information Retrieval* present the “single-pass in-memory” algorithm (SPIMI) for constructing an index given a sequence of pairs (term, docID) for a corpus. Using pairs of this type, one constructs an inverted index that only contains the list of documents for each term.

**Part a.** Modify the “single-pass in-memory” algorithm to process a sequence of tuples (term, docID, position in doc., attributes) and produce an inverted index that, for each term, records all positions at which the term occurs in each document and attributes of each occurrence. Clearly describe how each tuple is processed. Also include in your description the parsing of documents to produces the sequence of tuples processed by SPIMI. (The textbook explicitly excludes document parsing in the description of SPIMI. It is included in the description of Blocked sort-based indexing: Figure 4.2.)

**Part b.** Assume there is one buffer in RAM allocated for the algorithm of Part a. The buffer has a capacity of B pages. Describe how the buffer is used by your modified “single-pass in-memory” algorithm (SPIMI), including the document parsing phase. Make clear when disk reads and/or writes occur.

**Part c.** Compare your answer in Part b to the description of "blocked sort-based indexing" given on pg. 71 of *An Introduction to Information Retrieval*. Assume that the main memory used in the description on pg. 71 is the RAM buffer of capacity B pages. Does one algorithm lead to fewer disk pages reads and writes than the other? Justify your answer.

## Problem 3

**Part a.** Give the Elias-***gamma*** code for each of these integers: 4228, $2^{75}$-1. How many ***bits*** does each encoded integer require? How many bytes?

**Part b.** Give the Elias-***delta*** code for each of these integers: 4228, $2^{75}$-1. How many ***bits*** does each encoded integer require? How many bytes?

**Part c.** Decode the following ***sequence*** of numbers encoded with the Elias-delta encoding:

100101100100101101110