CSC 483/583: Assignment #3 (75 pts)

Due by 11:59 P.M., March 17
(upload to D2L or, if on paper, turn it in in class or in the instructor's office)

Because the credit for graduate students adds to more than 75 points, graduate students' grades will be normalized at the end to be out of 75. For example, if a graduate student obtains 80 points on this assignment, her final grade will be $80 \times 75/85 = 70.6$. Undergraduate students do not have to solve the problems marked "grad students only". If they do, their grades will not be normalized but will be capped at 75. For example, if an undergraduate student obtains 80 points on this project (by getting some credit on the "grad students only" problem), her final grade will be 75.

Note that only problem 1 (parts 1, 3, and 4) requires coding.

Problem 1 (30 points undergraduate students / 40 points graduate students)

For this problem you will ~~be forced~~learn how to use Lucene:

1. (15 points) Write code to index the following documents with Lucene:

   ```
   Doc1 information retrieval is the most awesome class I ever took.
   Doc2 the retrieval of private information from your emails is a job that the NSA loves.
   Doc3 at university of arizona you learn about data science.
   Doc4 the labrador retriever is a great dog.
   ```

   Your indexed documents should have two fields: a tokenized and searchable text field containing the text of the document (i.e., each line without the first token), and another non-tokenized field containing the document name (the first token in each line), e.g., `Doc1`. Search for the query `information retrieval`. Print the list of documents in the search result, sorted in descending order of their relevance score. For each document, print its name and score.

   For this task, you might want to read this tutorial first: `http://www.lucenetutorial.`

com/lucene-in-5-minutes.html. If you have to program in Python, this Python wrapper follows closely the original Lucene syntax: http://lucene.apache.org/pylucene/ (but we do not recommend it! The best solution for this homework is to program in Java/Scala.)

2. (5 points) Describe the default scoring strategy in Lucene. Is it Boolean, vector-based, or a combination of both? To answer this question, you might want to start here: http://lucene.apache.org/core/7_6_0/index.html.

3. (10 points) Translate the following three queries written in pseudocode into the Lucene query syntax, and run them: (a) information AND retrieval, (b) information AND NOT retrieval, and (c) information AND retrieval WITHIN 1 WORD OF EACH OTHER. What documents and what scores are returned for each of these queries? For this task, you might want to read about the syntax of Lucene queries: http://lucene.apache.org/core/2_9_4/queryparsersyntax.html.

4. (10 points – **graduate students only**) Change the default similarity formula in Lucene to a different one. For example, you might want to replace Lucene's default probabilistic formula (Okapi BM25) with cosine similarity over *tf-idf* vectors (what we covered in lecture 6). Does the ordering of documents change using this new formula for the query information retrieval? What are the scores returned for each document now?

Note: The software for this problem must use maven (or sbt), and must be submitted using the instructions we will provide over Piazza. No other software submission will be accepted.

Problem 2 (10 points)

Compute variable byte codes and γ codes for the postings list $< 777, 17743, 294068, 31251336 >$. Use gaps instead of docIDs where possible. Write binary codes in 8-bit blocks. You can use Google, or any other resource, to convert numbers to binary.

Problem 3 (10 points)

From the following sequence of γ-coded gaps, reconstruct first the gap sequence and then the postings sequence: 1110001110101011111101101111011.

Problem 4 (10 points)

Consider the table of term frequencies for 3 documents denoted Doc1, Doc2, Doc3 in the table below. Compute the tf-idf weights for the terms "car", "auto", "insurance", and "best", for each document, using the idf values from the second table.

	Doc1	Doc2	Doc3
car	27	4	24
auto	3	33	0
insurance	0	33	29
best	14	0	17

term	df_t	idf_t
car	18,165	1.65
auto	6,723	2.08
insurance	19,241	1.62
best	25,235	1.5

Problem 5 (15 points)

1. What is the idf of a term that occurs in every document? Compare this with the use of stop word lists.
2. How does the base of the logarithm in the *idf* formula affect the score calculation of the following formula:

$$Score(q, d) = \sum_{t \in q} tf.idf_{t,d} \tag{1}$$

Does a different logarithm base affect the relative scores of two documents on a given query? If so, how? To answer this question, you must know how to change the base of a logarithm. Google "how to change the base of a logarithm" :)