CSC 483/583: Assignment #1 (75 pts)

Due by 11:59 P.M., January 27
(upload both code and report to D2L)

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 the first four problems do not require coding. Only problem 5 requires coding.

Problem 1 (15 points)

Consider these documents:

Doc 1 breakthrough drug for schizophrenia

Doc 2 new approach for treatment of schizophrenia

Doc 3 new hopes for schizophrenia patients

Doc 4 new schizophrenia drug

1. Draw the term-document incidence matrix for this document collection.
2. Draw the inverted index representation for this collection, as in Figure 1.3 in IIR.
3. What are the returned results for these queries:
   (a) schizophrenia `AND` drug
   (b) for `AND NOT`(drug `OR` approach)

Problem 2 (20 points)

1. Write out a postings merge algorithm, in the style of Figure 1.6 in IIR, for an x `OR` y query.
2. Write out a postings merge algorithm, in the style of Figure 1.6 in IIR, for an x `AND NOT` y query.

Problem 3 (10 points)

Recommend a query processing order for:

(tangerine `OR` trees) `AND` (marmalade `OR` skies) `AND` (kaleidoscope `OR` eyes)

given the following postings list sizes:

Term	Postings size
eyes	213312
kaleidoscope	46653
marmalade	107913
skies	271658
tangerine	87009
trees	316812

Problem 4 (10 points)

How should the Boolean query x `OR NOT` y be handled? Why is the naive evaluation of this query normally very expensive? Write out a postings merge algorithm that evaluates this query efficiently.

Demorgens law twice

Problem 5 (20 points undergrad / 30 grad)

Implement an inverted index that supports Boolean search. Your program must take in one file containing one document per line, in a format close to the one used in Problem 1. For example, you can use the file below to test your code:

```
Doc 1    breakthrough drug for schizophrenia
Doc 2    new approach for treatment of schizophrenia
Doc 3    new hopes for schizophrenia patients
Doc 4    new schizophrenia drug
```

In particular, the format requires:

- One document per line in the input file.
- The first token in each line be the document id. The id is not to be indexed as a term!
- The rest of tokens in a line be all terms appearing in the document. Assume that the text is already tokenized and the tokens are normalized.

To code this problem, you can use any programming language that is familiar to the instructor (C/C++, Java, Scala, Python). You can use data structures available in your programming language of choice, e.g., dictionaries in Python or hash maps in Java/Scala, but you are **not** allowed to use open-source code that implements inverted indices, such as Lucene. You have to implement the inverted index and corresponding search operations from scratch.

The code submitted **must** compile and run. You **must** also include in your submission a `Makefile/pom.xml/build.sbt` file or shell script that allows the instructor to run the code, together with a `README` file that describes usage.

Please implement the following:

1. **(20 points)** Construct the inverted index and implement support for binary `AND` queries. What does your code return for the file above and the query: schizophrenia `AND` drug?
2. **(GRAD STUDENTS ONLY: 5 points)** Add support for binary `OR` queries. What does your code return for the query: breakthrough `OR` new?
3. **(GRAD STUDENTS ONLY: 5 points)** Add support for queries that combine multiple binary `AND` and `OR` operators, such as: (drug OR treatment) AND schizophrenia. You can choose a default operator priority, e.g., `AND` has a higher priority than `OR`, or use parentheses to make processing order explicit. Support for parentheses is optional. What does your code return for this query?