Polytechnic Institute of NYU March 6, 2013
Computer Science and Engineering
CS 6913, Spring 2013

Assignment #3 (due March 25)

In this assignment, you are asked to write a program that uses the inverted index structure from the second homework to answer queries typed in by a user. Please use the `nz` data set supplied to you for this assignment, or ask me for a larger data set of 25 million pages if you are adventurous. (If you were unable to complete HW2, but want to do HW3, please talk to me about what to do.) In the following, I will discuss the minimum requirements for your program, and also point out some opportunities for extra credit.

(0) **Languages:** You may use C/C^{++}, Java, or Python for this assignment. Or contact me if you prefer to use some other language. You may also use a mix of languages. But efficiency is very important.

(1) **Query Execution:** It is recommended that your program should execute queries by using the simple interface presented in class based on operations *open()*, *close()*, *nextGEQ()*, and *getFreq()* on inverted lists. (Recall that *nextGEQ(l, id)* returns the first posting in an already opened list l with docID at least id, when searching in a forward direction from the current position of the list pointer.) This way, issues such as file input and inverted list compression technique should be completely hidden from the higher-level query processor.

(2) **Ranked Queries:** Your query execution program should compute ranked queries according to the BM25 measure presented in class. Return the top 10 or 20 results according to the scoring function. You may implement conjunctive or disjunctive queries, or both. You may also experiment with additional ranking factors (e.g., context, proximity, Pagerank) for extra credit, as optional features.

(3) **Interface:** Your program may read user queries via simple command line prompt, and you only need to return the URL of each result and its BM25 score. However, for extra credit, you could make your program web accessible, and also return some surrounding text (text snippets) for each result.

(4) **Startup:** Upon startup, your query processor may read the lexicon and URL table data structures from disk into main memory. However, you should not read the index itself into memory! After the user inputs a query, your program should then perform seeks on disk in order to read only those inverted lists from disk that correspond to query words, and then compute the result. After returning the result, your program should wait for the next query to be input. For extra credit, you may implement caching of inverted list data in main memory (say, using 64KB blocks), but you should not assume that the entire index fits in main memory.

(5) **Efficiency:** Your query processor may take a little longer during startup, but each query should then be answered fairly quickly, say in *at most* a second, even on the complete `nz` data set. So make sure to be efficient. Use at least a simple form of index compression, such as Variable-Byte or Simple9. For extra credit, you may use an index that is chunk-wise compressed, and then have your query processor skip over chunks for additional efficiency. But if done efficiently, a simple forward scan without skips should also be fast enough.

(6) **Data Size:** For a minimal solution, use the `nz10` data set, but it would be better to use the entire `nz` data set. For extra credit, ask me for a larger 25 million page data set.

For this assignment, please hand in the following in electronic form: (a) your well-commented program source, and (b) a 10-15 page paper (written in Word or Latex or similar) explaining what your program can do, how to run the program, how it works internally, how long it takes on the provided data set and how large the resulting index files are, what limitations it has, and what the major functions and modules are. This paper should be readable to a generic computer scientist who knows what an inverted index is, but otherwise does not know much about search engines. So start very high-level! There will also be a demo for each group, which will be scheduled in the week after spring break.