INF3800/INF4800

Assignment B

WHAT TO DELIVER:
Please submit a zip containing both the implementation task (the entire project) and the pencil-and-paper tasks. Exercise 4 is optional for INF3800 students.

DEADLINE:
The assignment must be submitted, using Devilry, by 20.02.2017 at the latest.

On paper

Exercise 1:

If you wanted to search for `s*ng` in a permuterm wildcard index, what key(s) would one do the lookup on?

(Exercise 3.3, page 56)

Exercise 2:

Compute the edit distance between `paris` and `alice`. Write down the `5x5` array of distances between all prefixes as computed by the algorithm in Figure 3.5.

(Exercise 3.6, page 62)

Exercise 3:

String matching algorithms

a) Kurt has a large dictionary with millions of elements. As part of a document processing system, he wants to develop a module which can efficiently detect whether a word in the document is also present in the dictionary. Describe (i) what kind of data structure Kurt should use to represent his dictionary, and (ii) what kind of algorithm should be used to perform the search.

b) Kurt also has a big dictionary containing the surface forms of the most common words in Norwegian. As part of a spellchecking application, Kurt wants to be able to query the dictionary with a word *w*, and get back the set of all words which have an edit distance of at most *k* from *w*. More formally, the resulting set is thus defined as *{w' : editdistance(w' ,w) <= k }*. You can assume that the maximum distance *k* is small. Describe how Kurt should represent his dictionary and perform the search.

Before answering this exercise, you should read carefully the additional selected papers provided on the course website.

(This exercise is taken from the V11 final exam)

Exercise 4: (Optional for INF3800)

Permuterm indexes & Suffix arrays

a) Write down the entries in a permuterm index that are generated by the term `sting`. If you wanted to search for `s*ng` in a permuterm wildcard index, what key(s) would one do the lookup on?

b) Consider the term `mississippi`. Write down the suffix array for this term, and explain how you can use this to efficiently locate all occurrences of the substring is.

(This exercise is taken from the V14 final exam)

In code

Some types of searches can very well be carried out without the use of an inverted index. In this assignment you will implement a simple in-memory index that uses a suffix array to do phrase searches. E.g., the user query *foo bar baz* would require that we find the exact phrase *foo bar baz* somewhere in each document in the result set.

Download the code for the assignment B, which contains the SimpleSearch code, and notably a partial implementation of SuffixArray.java. Your task is to develop the remaining part of the code. The Junit test ObligBTest.java will allow you to easily check if your implementation of the suffix array behaves according to its specification. Basically the unit test should be able to build a suffix array over 2 alternative collections (respectively stored in data/cran.xm and data/wescience.txt), and perform simple phrase searches on it.

Mind the lookup speed as well – if your lookup takes significantly longer than 0.1 second for each document collection, you should see if you can optimize the code. It is expected that your implementation not only works, but will scale reasonably if the index was significantly larger.

Optional: If you have some free time, find a way to minimize the memory space requirements of your implementation, by using the minimal number of bytes necessary to record the suffix array, and avoid the creation of thousands of unnecessary objects.