Chair for Algorithms
and Data Structures
Prof. Dr. Hannah Bast
Patrick Brosi

Information Retrieval
WS 2016/2017

http://ad-wiki.informatik.uni-freiburg.de/teaching

UNI FREIBURG

Exercise Sheet 3

Submit until Tuesday, November 15 at **2:00pm**

For this exercise sheet, you can use Java or C++, **but not Python**. Subtle algorithmic improvements and performance tuning make little sense in Python, due to the large overhead of the underlying data structures, in particular of the Python lists/arrays.

Copy the code provided on the Wiki (which is similar to the basic version of the code written during the lecture, without the performance improvements) and proceed from there. It is available in both Java and C++. It contains code for reading posting lists from the given files, for the baseline intersection, and for time measurement, as well as tests for the intersection method(s).

Exercise 1 (20 points)

You have been hired to prove that the city of Bielefeld's existence is, in fact, an illusion created by **them**. Your employer provides you with Wikipedia's posting lists for "bielefeld", "them" and "existence". Your job is to find documents relevant to the investigation. You only have slow code at hand and time is of the essence!

1. (10 points) Implement a new method for intersecting two posting lists that uses at least three non-trivial ideas presented in the lecture. The goal is to beat the baseline implementation from the lecture for all scenarios. Note that you can also implement several algorithms and switch between them depending on the sizes of the input lists (or depending on any information that you find to be useful). Each implemented method must pass the test case provided for the baseline implementation in the example code!

2. (10 points) Evaluate your algorithm on all three pairwise combinations of the three posting lists *bielefeld.txt*, *them.txt* and *existence.txt* on the Wiki. Report the results from your evaluation in the table on the Wiki, following the baseline rows already there.

[**they** want you to turn over]

Commit your code to a new sub-directory *sheet-03* of your folder in the course SVN.

You must implement (at least) the test cases from the *IntersectTest.java* or *IntersectTest.cpp* file in the example code and your code must run through on Jenkins without errors. Otherwise your submission will not be graded. This goes without saying from now on. The reasons for these requirements were explained in Lecture 1 and repeated / elaborated on at the beginning of Lectures 2 and 3.

As usual, summarize your experiences and insights in your *experiences.txt*. As a minimum, say how much time you invested and if you had major problems, and if so, where. Please be brief and informative: that makes reading your feedback much more pleasant for us.