Chair for Algorithms
and Data Structures
Prof. Dr. Hannah Bast
Elmar Haußmann

Information Retrieval
WS 2015/2016

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

UNI FREIBURG

Exercise Sheet 1

Submit until Tuesday, October 27 at **2:00pm**

Exercise 1 (10 points)

Extend the code from the lecture by the following two methods. The method *merge* should merge two given inverted lists (5 points). The method *process_query* should compute the list of ids of all records containing at least one word from the given query (5 points).

See the TIP file on the Wiki for the precise specification. See the second page of this sheet for general comments about coding exercises in this course

Exercise 2 (5 points)

Extend your code by a *main* method that uses your code from Exercise 1 to build an inverted index from a given text file and then answers arbitrary keyword queries entered by the user (in an infinite loop). For each query, output the top-3 records containing the query words. Again, see the TIP file on the Wiki for the precise specification.

Try your code on the file *movies.txt* on the Wiki. Find a query that gives good results (the top records match what you had in mind when you asked the query), and one that does not. Write them in your *experiences.txt* (see below), and briefly explain why one works and the other doesn't.

Exercise 3 (5 points)

Register with our course system Daphne (using your RZ account + password for authentication). Make sure that your data is correct, in particular, that you are reachable under the specified e-mail address.

Check out a working copy of your folder in the SVN repository of the course, and add your code to a new subdirectory *sheet-01*, and commit it. Make sure that everything runs through without errors on Jenkins.

Also commit, in that subdirectory, a text file *experiences.txt* where you briefly describe your experiences with the first exercise sheet and the corresponding lecture. As a minimum, say how much time you invested and if you had major problems, and if yes, where.

[please turn over]

Exercise 4 (Optional)

Highlight the query words in the output of your program, e.g. using ANSI escape codes. Play around with better ways (for the user) to sort the result list and obtain the top-3 results.

Think/research about why word frequencies follow Zipf's law. Prove that if a text is created by choosing each character (including space) independently and uniformly at random, Zipf's law follows mathematically.

General comments about the coding exercises (valid throughout the course):

1. You can code in Python, Java, or C++. You can make different choices for different exercise sheets. In the lecture, we will often use Python, which will allow us to focus on the conceptual issues. When efficiency is a central issue, we will use Java or C++.

2. For some exercise sheets, the code from the lecture will be available only in Python. You are still free to use Java or C++ for those sheets, but it will be more work for you then.

3. If the exercise sheet mentions a TIP file, read it. It contains the exact specifications of what to compute. It may also contain valuable implementation advice (which you can but do not have to follow).

4. Our coding conventions must be followed at all times. In particular:

4.1 You have to write a unit test for each non-trivial method. One non-trivial example per unit test is enough. All unit tests must be fast. If your test reads an input file, that file must be small.

4.2 You have to adhere to our coding style.

4.3 You have to provide a standard build/make file along with your code and you have to make sure that everything runs through without errors on our continuous build system (Jenkins).

You find example code for all three languages in the *public/code/lecture-01* folder in the SVN.

5. Make sure that you do not upload any "by-products" to our SVN (e.g., class files or executables or any stuff from your local environment). Also *never ever* upload large data files to our SVN; this is considered sin and will result in excommunication.

6. When you encounter implementation problems proceed as follows. First do the obvious Google search: very often, this leads to a page describing exactly the problem you are having and the solution for it. Then search/read our forum (link on the Wiki) to see if someone has already asked a similar question. Then you are very welcome to ask a question on the forum yourself. In general: please ask before you spent a lot of time on minor implementation issues.

7. You are allowed to work on the exercises in groups of at most two. If you want to work in such a group, send a mail with the name of both of your RZ accounts to Axel Lehmmann (lehmann@cs.uni-freiburg.de). He will then create a joint subfolder for you in our SVN.