Chair for Algorithms
and Data Structures
Prof. Dr. Hannah Bast
Marjan Celikik

Search Engines WS 09/10

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

UNI FREIBURG

Exercise Sheet 9

complete until Thursday, January 14th

Exercise 1

Characterize the set of all *invalid* UTF-8 multi-byte sequences. If you think about it, this boils down to saying which bytes cannot occur as the first byte in a UTF-8 multi-byte sequence, and which bytes cannot occur as a second / third / fourth byte in a UTF-8 multi-byte sequence. You should of course justify your answer and not just claim things.

Exercise 2

Write the following program in either Java or C++. Generate a random sequence of n bytes, where n is an input parameter. Then go over the string, and using your characterization from Exercise 1, replace as few characters as possible such that the resulting string is completely valid UTF-8. Make an effort to write efficient code. In any case, your program should run in $O(n)$ time. (Note: "as few characters as possible" is not meant in a mathematical optimal sense, it's just so you avoid extremes like replacing all characters with ASCII code ≥ 128 by zero.)

Exercise 3

If you wrote the program for Exercise 1 in Java, now write the same program in C++. If you wrote it in C++, now write it in Java. Again, make an effort to write efficient code.

Exercise 4

Write the program from Exercise 1 in a script language. You may choose between Perl, Python, and PHP. Again, make an effort to write efficient code.

Exercise 5

Run all three programs for $n = 10^3, 10^6, 10^9$. Measure the running time for each program and each setting of n, averaged over 10 runs in each case. Measure only the time for the actual string repair, not the time for generating the random sequence. Summarize your results in a table with 3 columns (one for each programming language) and 3 rows (one for each setting of n). Briefly discuss your results.