DM823 String Algorithms

Fall 2010

Department of Mathematics and Computer Science
University of Southern Denmark

September 20, 2010

Introduction

The purpose of this project is try out in practice some of the algorithms learnt in this course.[1] The project is to be done in groups of size between one and four persons.

Requirements

Let x ($1 \leq x \leq 4$) be the number of members of the group. You are to implement at least $x + 1$ algorithms (of your own choice) for exact pattern matching among the following:

- The (K)MP algorithm.
- The BM algorithm.
- The Horspool variant of the BM algorithm.
- The Galil (BMG) variant of the BM algorithm.
- The Fast-on-average algorithm.[2]
- The Shift-AND algorithm.

Additionally, the Brute-force algorithm (with either forward or backward verification) should be implemented. This will serve as a solution against which all other implementations should be thoroughly tested for correctness.

Any necessary preprocessing of the pattern (*MP_Shift* for KMP, *BM_Shift* for BM and BMG) is of course part of the implementation required.

The choice of programming language is left to you.

[1] Alternatively, you can do a theoretical variant of the project, see later.

[2] For this, you will need either a suffix tree or suffix array built on the pattern in order to test in time $O(r)$ whether a given string of length r is a substring of the pattern. For simplicity, we suggest using a suffix array, and building it by simple, non-optimal means, such as radixsort. Come see the lecturer for further details.

After implementation, empirical investigations of the implementations made (including the Brute-force algorithm) should be performed on non-trivial size test data with varying alphabet sizes and data origins. These test data, as well as further details of the experiments required (suggested query patterns, number of experiments, etc.) will be given later by the lecturer.

Finally, the outcome of the experiments should be displayed appropriately, e.g. by graphs on running time (wall-clock time) versus pattern length for each data instance. The results should be discussed.

Formalities

You should hand in: A report of 10–15 pages (including figures) in pdf-format. Files with source code (source code is not to be included in the report, except possibly as snippets in the main report text).

The main focus of the report should be on describing the design choices made during development and the structure of the final implementations, as well as on describing in detail the experiments made and discussing the outcome of these.

For groups of size more than one: For formal reasons, you will need to designate who wrote which part of the programs and report.

All code should be your own. In particular, grabbing code from the net is not legal. Basing you implementation on *pseudo-code* from the net (or elsewhere) is allowed, if you reference the original, and if you *explain* the working of the pseudo-code (if the pseudo-code is different from that/those in the course material).

The project will be evaluated by pass/fail grading. The grading will be based on:

- The clarity of the writing and of the structure of the report.
- The thoroughness of the experiments (execution as well as discussion).
- The amount of work done.

The material should be handed in using the `aflever` command on the Imada system: Move to the directory containing your report and code (hopefully well structured in subdirectories) and issue the command `aflever DM823`. This will copy the contents of the directory to a place accessible by the lecturer. Repeated use of the command is possible (later uses overwrite the contents from earlier uses). In the directory, you must for identification purposes have an ASCII file named `names.txt` containing the names of the group members, with one name per line.

You must hand in the material by

Sunday, October 17, 2010, at 23:59

Variant

If you prefer, you may instead of the above do a theoretical project (no programming), in groups of size at most two. The report will then consist of a written exposition of a research paper relevant to the course. The length is expected to be 10–20 pages. You are to describe, in your own words (avoid as much as possible just copying from the paper), the background (briefly), the main result achieved, the algorithm of the result, and the proof of correctness and time complexity.

Some examples of possible papers are:

- M.I. Abouelhoda, S. Kurtz, E. Ohlebusch. *Replacing suffix trees with enhanced suffix arrays.* Journal of Discrete Algorithms 2 (2004), p. 53–86. [Only sections 1–6 need to be covered. Not all applications discussed in paper need be covered.]
- S. J. Puglisi, W.F. Smyth, M. Yusufu. *Fast, Practical Algorithms for Computing All the Repeats in a String.* Mathematics in Computer Science 3 (2010), p. 373–389. [Only sections 1 and 2 need to be covered.]
- S. Dori, G. M. Landau. *Construction of Aho Corasick automaton in linear time for integer alphabets.* Information Processing Letters 98 (2006), p. 66–72.

You are free to suggest further papers (but they must be approved by the lecturer).