Project in String Processing Algorithms

Spring 2014, period III

Juha Kärkkäinen
Who is this course for?

- Master’s level course in Computer Science, 2 cr
- Continuation of String Processing Algorithms course
- Requires some programming experience
- Subprogram of Algorithms and Machine Learning
  - Together with String Processing Algorithms one of the three special course combinations, one of which must be included in the Master’s degree.
- Suitable addition to Master’s degree program for Bioinformatics, particularly for those interested in biological sequence analysis
- Good fit for Subprogramme of Software systems
Course structure

- Three main tasks
  1. Implementation of string processing algorithms
  2. Experimental analysis and/or comparison of the algorithms
  3. Presentation of the results as a poster

- Each task has about the same weight in grading

- Can be done in groups of at most three
  - Each group member implements something
Algorithm implementation

• Each student in a group implements a significant part of the core algorithms
  – Separate grading for each student

• Can be based on existing implementations

• Any programming language, provided that:
  – Compiles and runs on department computers
  – Same within a group

• Important qualities:
  – correct, well tested
  – readable, well documented
  – efficient, well tuned

• Degree of difficulty is taken into account
Algorithm implementation (continued)

Return to instructor:

• Implementation code

• Scripts for compiling and running tests

• Documentation
  – description of what was done: existing code used, main design decisions, tuning details etc.
  – roles of group members
  – guidance for understanding the code
  – instructions for compiling and running
  – format is free, even comments to code is OK

• By email in a single package (zip, tar.gz, or something like that)
Experiments

• The purpose of the experiments:
  – Determine the performance of algorithms under different conditions
  – Find best algorithms, variations or parameter settings

• Choice of test data is important
  – Try to find best and worst cases for each algorithm.
  – Compare theory and practice.
  – Use generated, artificial data for fine control of parameters, real world data for real world performance.
  – Avoid too trivial experiments. For example, exact string matching time is trivially linear in the length of the text.

• Mainly joint responsibility of a group, but each student should make sure that her or his algorithms are well represented.
Poster

- Includes:
  - Description of the problem
  - Description of algorithms and implementations
  - Experimental setting (repeatability)
  - Experimental results and their interpretation

- Presented to an audience of other students and staff of the department
  - Not all have taken the String Processing Algorithms course (recently)

- Visual clarity is important
  - Avoid too much detail, include only main points and results. Additional details may be explained verbally.
  - Use figures, graphs, colors, etc.

- See examples
**Tentative schedule**

14.1. Formation of groups, selection of topics
   - Study the topic

21.1. Finalization of topic details
   - Study implementation details

28.1. Additional details on implementations
   - Implement

4.2. Initial design of experiments
   - Implement, study experimenting

11.2. Implementations (nearly) finished, final design of experiments, initial design of poster

14.2. Return of implementations
   - Experimenting, poster making

18.2. Poster (nearly) finished

???.2. Poster presentation
Topic: Exact String Matching

- Extensive implementations and experiments using C
  - http://www.dmi.unict.it/~faro/smart/
  - S. Faro and T. Lecroq: The exact online string matching problem: A review of the most recent results. ACM Computing Surveys 45, 2, Article 13 (March 2013), 42 pages.
    http://doi.acm.org/10.1145/2431211.2431212

- Other programming languages?
Topic: String Range Matching

- Generalization of exact string matching
- Given a text $T$ and two patterns $P$ and $Q$, list suffixes of $T$ that are lexicographically between $P$ and $Q$
Topic: Multiple Exact String Matching

- Aho-Corasick

- Multi-pattern versions of Shift-Or, Horspool, BOM, Karp-Rabin, ...

Topic: Approximate String Matching

- Standard dynamic programming, Ukkonen’s cut-off heuristic, Myers’ bitparallel algorithm, filtering algorithms, ...


Topic: String sorting

- Extensive set of implementations and experiments in C++:
  - https://panthema.net/2013/parallel-string-sorting/
- Other programming languages?
- Cache misses are important
Other topics

- string search trees
- suffix array construction
- ...
- Own topic