Project in String Processing Algorithms

Spring 2015, 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
- Regular course of Algorithms, Data Analytics and Machine Learning subprogramme
- Suitable for Algorithmic Bioinformatics subprogramme (or MBI) particularly for those interested in biological sequence analysis
- Good fit for Software Systems subprogramme

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
  - simple example(s)
  - correctness 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 in the 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 their 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

Week 1 (13.1.): Formation of groups, selection of topics

- Study the topic

Week 2 (20.1.): Finalization of topic details

- Study implementation details, start coding

Week 3 (27.1.): Additional details on implementations

- Coding, start documenting, study experimenting

Week 4 (3.2.): Initial design of experiments

- Coding, documenting, design experiments, study poster making

Week 5 (10.2.): Final design of experiments, initial design of poster

- (12.2.): Return of implementations
- Experimenting, poster making

Week 6 (17.2.): Final design of poster, show draft poster

- Finalize poster

Week 7 (24.2.?): 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
- J. Kärkkäinen, D. Kempa, S. Puglisi: String Range Matching. Proceedings of the 25th Symposium on Combinatorial Pattern Matching (CPM), pp. 232-241, 2014. `http://dx.doi.org/10.1007/978-3-319-07566-2_24`

Topic: Multiple Exact String Matching

- Aho-Corasick
- Multi-pattern versions of Shift-Or, Horspool, BOM, Karp-Rabin, ...
- L. Salmela, J. Tarhio, and J. Kytöjoki: Multipattern string matching with q-grams. Journal of Experimental Algorithmics 11, Article 1.1 (February 2007). `http://doi.acm.org/10.1145/1187436.1187438`

Topic: Approximate String Matching

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

- G. Navarro: A guided tour to approximate string matching. ACM Computing Surveys 33(1): 31–88, 2003.
  `http://doi.acm.org/10.1145/375360.375365`

- L. Salmela and J. Tarhio: Approximate String Matching with Reduced Alphabet. Workshop on Algorithms and Applications, LNCS 6060, Springer 2010. `http://dx.doi.org/10.1007/978-3-642-12476-1_15`

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 tree construction
  - McCreight vs. transform from suffix array
- ...
- Topics from an earlier year: `www.cs.helsinki.fi/u/vmakinen/strproject12/strproject12.pdf`
- Own topic