Project in String Processing Algorithms

Spring 2017, period III

Juha Kärkkäinen and Dominik Kempa
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 (24.1.): Formation of groups, selection of topics
  • Study the topic

Week 2 (31.1.): Finalization of topic details
  • Study implementation details, start coding

Week 3 (7.2.): Additional details on implementations
  • Coding, start documenting, study experimenting

Week 4 (14.2.): Initial design of experiments
  • Coding, documenting, design experiments, study poster making

Week 5 (21.2.): Final design of experiments, initial design of poster
  • (23.2.): Return of implementations
  • Experimenting, poster making

Week 6 (28.2.): Final design of poster, show draft poster
  • Finalize poster

Week 7 (2.3.?): Poster presentation
Topic: Exact String Matching

- Extensive implementations and experiments using C
  - http://www.dmi.unict.it/~faro/smart/

- 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 $Q$


  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, ...
**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/](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:

- Own topic