02-514/02-714: String Algorithms

Carl Kingsford

1. Course Information

1.1 Course description

Provides an in-depth look at modern algorithms used to process string data, particularly those relevant to genomics. The course will cover the design and analysis of efficient algorithms for processing enormous amounts of collections of strings. Topics will include string search; inexact matching; string compression; string data structures such as suffix trees, suffix arrays, and searchable compressed indices; and the Borrows-Wheeler transform. Applications of these techniques in genomics will be presented, including genome assembly, transcript assembly, whole-genome alignment, gene expression quantification, read mapping, and search of large sequence databases. No knowledge of biology is assumed; programming proficiency is required.

1.2 Pre-requisites

- Equivalent of 15-210 ("Parallel & Sequential Data Structures and Algorithms") or 02-513/02-713 ("Algorithms and Data Structures for Scientists").
- Equivalent of 15-151 or 21-127.
- Programming proficiency.

Computational Genomics (02-710) is not a pre-requisite.

1.3 Textbooks

- Gusfield, *Algorithms on Strings, Trees and Sequences*, strongly recommended
- Crochemore and Rytter, *Jewels of Stringology*, recommended

1.4 Coursework

The coursework for the class will consist of: several written homework problem sets and in-class quizzes (25%), a midterm (25%), and a class project (50%) selected by the student in consultation with the instructor.

Homeworks and Quizes: During the first half of the semester, there will be 2–4 written homework assignments, designed to help you prepare for the midterm. During the second half of the semester, there will be a weekly short quiz on an assigned paper. The quiz will be 5 minutes and designed to test whether you read the paper.

Project: Deliverables on the project are: (1) A brief 1-page project proposal (due at the start of the project); (2) a write-up of 6 polished pages (in two-column IEEE format) describing the problem you are trying to solve, your solution, and its relation to other work on the same or similar problems; (3) a short (10–15 minute) presentation during the scheduled final exam time; and (4) either a runnable piece of software that can be used to solve your chosen problem OR a meaningful theoretical result described in publication-quality detail (in addition to the writeup). Depending on the class size, projects must be completed in groups of 3 students.

2. Tentative Topics

The first two units below (comprising about the first half of the semester) provide the theoretical background in string algorithms and data structures needed to tackle the research topics in the second half of the semester. This set of topics is probably slightly too long for a single semester; a couple of the topics will probably be dropped. The topics marked with a $\star$ are especially important and would be covered no matter how the schedule unfolds.

- **Searching and string comparison (4 weeks):**

  1. Exact string matching (Gusfield, chapters 1-4): the Z-algorithm; Knuth-Morris-Pratt; Boyer-Moore; seminumerical string matching (Rabin-Karp).
  2. Advanced inexact matching (Gusfield, chapters 10–14): Computing basic edit distance; string alignment in linear space; Four Russians speed up; approximation algorithms for multiple sequence alignment (MSA); Tool: whole genome alignment with MUMmer.
  3. $\star$ Parallel string matching (Crochemore & Rytter, chapter 16).

- **String data structures (3 weeks):**

  4. $\star$ Suffix trees/arrays (Gusfield, chapters 5–9): definition of suffix trees and arrays; Ukkonen's suffix-tree construction algorithm; applications of suffix trees and arrays.
  5. $\star$ Subword graphs and their construction (Crochemore & Rytter, chapter 6).

- **Compression, compressed indices, and compressed search (4 weeks):**
  (aka. making 'big data' small)

  6. String compression algorithms.
  7. $\star$ Compressed self-indices (i.e. data structures that support fast searching and reconstruction of the full sequence in sublinear space; these are the basis of current read-mapping techniques): Burrows-Wheeler transform; the FM-index; other compressed, self-indices; wavelet trees.
  8. Read mapping: Bowtie, BWA, TopHat.
  9. $\star$ Compressive genomics (i.e. doing genomic analyses directly on compressed data): searching compressed collections of sequences; comparative assembly from compressed read databases; outcome-directed lossy compression (saving what matters).

- **Other applications involving large collections of sequencing data (4 weeks):**

  10. Genome assembly: Shortest superstring problem, Celera assembler (overlap-layout-consensus); de Bruijn-graph-based assembly (e.g. ALLPATHS or EULER); mixed assembly (e.g. MSR-CA).
  11. Isoform / transcript assembly (e.g. Cufflinks, Trinity).
  12. $\star$ Gene and isoform expression quantification: RSEM, eXpress, Jellyfish, Cufflinks, Sailfish.

3. Relationship to existing courses

Students who have taken 02-710 or 03-711 will find very little duplicate material.

The overlap with 02-710 ("Computational Genomics") is fairly minimal. Both classes cover sequence alignment, but this proposed class focuses on more advanced algorithmic aspects. 02-710 covers HMMs and motif discovery, which we will not cover in any depth. 02-710 includes one lecture on genome assembly, which may overlap with our discussion of genome assembly; but this field is so vast, it is likely that students taking both classes will hear about very different assembly algorithms.

03-711 ("Computational Molecular Biology and Genomics") covers a lot of HMM-related material; we will not cover HMMs. 03-711 also covers sequence alignment, but again, the this course will cover more advanced aspects particularly relating to dealing with large data sets. 03-711 covers BLAST, substitution matrices, and BLAST statistics in depth; we will cover none of that except possibly some ideas from BLAST.

4. Policies

Homework policies:

- Homeworks are due at the start of class. **No late homework will be accepted** — turn in what you have completed. If you will miss class, turn in the homework early.
- Answers to homework problems should be written concisely and clearly. Homeworks must be typeset and submitted as PDFs. Instructions for submission will be posted on the course webpage.
- Homework problems that ask for an algorithm should present: a clear English description or pseudocode, an argument that the algorithm is correct, and an analysis of the running time.
- Graded homeworks (and midterms) should be picked up in class; if you miss the class when the homework is returned, please pick it up during office hours.
- Regrade requests should be made **in writing** within 1 week of the homework being returned. The entire homework or exam in question will be regraded, which may result in a higher or lower grade than originally returned.
- You may discuss homework problems with classmates. You must list the names of the class members with whom you worked at the top of your homework. **You must write up your own solution independently!** "Independently" means — at least — that you cannot look at another person's homework, you cannot have them look at yours to see if it is correct, you cannot take detailed notes from a discussion and edit them into your homework, and you cannot sit in a group and continue discussing the homework while writing it up. The intent of this rule is: you can gather around a whiteboard with your fellow students and discuss how to solve the problems. Then you must all walk away and write the answers up separately. Note: it's really the exams that count for most of your grade, so there's little benefit in writing down a homework answer that you don't understand.
- Except on group assignments and projects, you must write all programming assignments on your own and cannot share code with other students or use code obtained from other students. In addition to manual inspection, we use an automatic system for detecting programming assignments that are significantly similar.

Excused absences: Students claiming an excused absence for an in-class exam or midterm must supply documentation (such as a doctor's note) justifying the absence. Absences for religious observances must be submitted by email to the instructor during the first two weeks of the semester.

Academic honesty: All class work should be done independently unless explicitly indicated on the assignment handout. You may *discuss* homework problems with classmates, but must write your solution by yourself. If you do discuss assignments with other classmates, you must supply their names at the top of your homework / source code. No excuses will be accepted for copying others work (from the current or past semesters), and violations will be dealt with harshly. (Getting a bad grade is much preferable to cheating.)

The university's policy on cheating and plagiarism can be found here: `http://www.cmu.edu/policies/documents/Cheating.html`. In part it reads "In any presentation, creative, artistic, or research, it is the ethical responsibility of each student to identify the conceptual sources of the work submitted. Failure to do so is dishonest and is the basis for a charge of cheating or plagiarism, which is subject to disciplinary action." You should be familiar with the policy in its entirety.