CSCI 1820 - Algorithmic Foundations of Computational Biology
&
CSCI 2820 - Advanced Algorithmic Foundations of Computational Biology
Spring 2024

Prof. Sorin Istrail
Department of Computer Science
Brown University

Course meeting: Tues./Thurs. 2:30-3:50pm, CIT 241 (SWIG)

Course website: `https://brown-cs182-spring24.github.io/`

TA email list: `cs1820tas@lists.brown.edu`

Staff:

- Professor Istrail. `sorin_istrail@brown.edu`
- Justin Currie (HTA). `justin_currie@brown.edu`
- Colin Baker (UTA). `colin_baker@brown.edu`
- Melinda Zhang (UTA). `melinda_zhang@brown.edu`
- Sanyu Rajakumar (UTA). `sanyu_rajakumar@brown.edu`

1 Course Description

The aim of this course is to provide mathematical and computer science foundations, as well as biological insights, for numerous seminal algorithms in the field of computational biology.

Course Topics

The course is organized into seven chapters:

1. The BLAST Algorithm and Karlin-Altschul Statistics
2. Genome Assembly Algorithms and Haplotype Assembly Algorithms
3. Hidden Markov Model (HMM) Algorithms: The Learning Problem
4. Recombination and Ancestral Recombination Graph Algorithms
5. Rigorous Clustering: Spectral Graph Theory Algorithms
6. Algorithms for Constructing Suffix Trees in Linear Time
7. Protein Folding Algorithms (An Introduction)*

Each chapter is devoted to a class of fundamental computational problems in genomics related to the analysis of DNA, RNA, protein sequences and structures, and their molecular biological functions. Our journey in each chapter is driven by a set of **beautiful** algorithms, presented together with their **theoretical foundations**, in comprehensive analytical detail. "Beautiful" algorithms are rigorous, practical and elegant, yet intuitive enough to be successfully implemented. These algorithms draw upon state-of-the-art theory and practice in order to solve the computational problems presented in each chapter. The **Algorithmic Foundations** section in each chapter presents a detailed account of the biological problems discussed and the theoretical computer science and statistical results that led to the invention of these algorithms. The algorithms are presented together with their underlying data structures, the mathematical analysis of their performance, and at times, the exciting story of the researchers quest for algorithmic optimality (speed). The overall work in the class will help in providing an algorithmically advanced journey through today's most indispensable software genomics tools of a bioinformatician and computational biologist.

Learning Goals

- Gain an understanding of the most fundamental algorithms of computational biology, bioinformatics, and genomics, and develop the software implementations of these genomic tools, as well as their test on real data and simulated data.
- Critically analyze the computer science, biology, and statistical sciences pillars of the state-of-the-art genomic tools rigorously presented in class, as well as what makes them practical tools and their programming code elegance.
- Improve your algorithm design and software development skills as well as analytical skills required to understand the mathematically correctness and computational complexity of the algorithms presented in class.

*We will give an impressionistic overview and first introduction to the most recent advances in AI-driven protein structure prediction.

Prerequisites

Required: any intro CS sequence, and CS 181 (Computational Molecular Biology).

Recommended: CS 220 (Introduction to Discrete Structures and Probability), or another course which introduces concepts from discrete math and probability theory.

Course overrides are available at the professor's discretion.

2 Course Format

Meeting times and place:

Tuesday and Thursday, 2:30-3:50pm in the SWIG, CIT 241.

You are expected to attend all classes. Class lecture notes will be made available.

Assignments

Homeworks and projects will alternate throughout the semester. Each homework (HW) will focus on the algorithmic theory and mathematical/biological basis of the current chapter, and you will generally be given 1-2 weeks to complete these assignments. Each project (PR) will involve implementing algorithms discussed in lectures in the programming language of your choice, and you will generally be given 2 weeks to complete these assignments. You will also be responsible for writing the aforementioned lecture notes for the class two times during the semester. There will be one in-class midterm exam and a take-home final exam.

Grading

- Class participation 5%
- Homework 30%
- Projects 30%
- Midterm exam 15%
- Final exam 20% (take-home)

Grades will be determined by your overall performance according to these metrics. At the end of the class, a *Pastiche Pie* award will be given to the student(s) with the overall most impressive performance in the class as judged by the TAs and the professor. All final grades will be determined by the professor. For CS 2820 students the final grade will be the average between the CS 1820 grade and the final project grade (given by the professor).

Literature

There is no textbook for this course. However, suggested readings will be provided on the course website to complement the lecture content of the class.

3 CS 2820: Graduate Credit

In addition to all assignments listed above, graduate students must complete a final project selected in consultation with the professor to receive the CS 2820 graduate credit. Undergraduates may choose to complete

an optional final project for extra credit. Details regarding the final project assignment will be made available midway through the semester and will require half a semester of work devoted to it.

4 Course Policies

Collaboration Policy

In addition to Brown's Academic Code, CS 1820 and CS 2820 follow the collaboration policy below:

- You may discuss **HW problems** with other students in the class; however, all solutions must be written up independently and reflect your own understanding of the material.
- You may discuss **PR assignments** and compare output on test cases with other students/groups in the class; however, each student/group must write up their code independently. You may not examine code written by other students/groups.
- You may not collaborate with anyone on the **midterm exam** nor the **final exam**. You may only discuss the content of the exams with members of the course staff. All solutions must be entirely your own.
- You will be required to accept this collaboration policy electronically at the beginning of the semester as a prerequisite for receiving grades for all subsequent assignments.

  The course staff takes violations of the collaboration policy seriously and will prosecute with the standing committee on the academic code as necessary.

Late Handin Policy

You will receive 4 late days for use throughout the course. As all handins will be electronic, you may use these late days at your discretion, with two caveats:

- You may use a maximum of 2 late days per individual assignment
- You may not use late days on exams (only on HWs and PRs)

Extra late days will be penalized 15% each. Additional extensions on HWs and PRs will only be granted by the professor under extenuating circumstances and at his discretion. TAs cannot grant extensions.

Coursework Hours

Students spend 3 hours in class for 12 weeks for a total of 36 hours. Software development is expected to take 7 hours per week for a total of 84- hours. Reading and Writing assignments are expected to take a total of -60 hours-. The total amount of time for the entire course will be 180 hours.

Diversity, Inclusion, Accessibility & Accommodations

Brown is committed to the full inclusion of all students, and CS 1820 and CS 2820 strive to be a welcoming and inclusive place for the diverse student body. Please reach out to the professor if you have any concerns regarding inclusivity, accessibility, or SEAS accommodations.

5 Lecture Topics

- **Chapter 1: The BLAST Algorithm**

  Algorithms

  - Data structures for sequences database
  - Ungapped sequence alignment
  - Finite automata data structures
  - Seeds and faster searches

  Statistical Theory

  - The Karlin-Altschul statistical theory of local alignment (un-gapped local alignment)
  - Substitution Matrices and Information Theory

  PIONEER: Margaret Dayhoff the *"mother and father of Bioinformatics"* - pioneer of statistical methods in bioinformatics.

- **Chapter 2: Genome Assembly Algorithms and Haplotype Assembly Algorithms**

  Algorithms

  - De Brujin Assembly Algorithms (de Brujin graphs and Eulerian paths)
  - Idury-Wateran Assembly Algorithm (Poisson statistics of DNA k-mers)
  - EULER Algorithm of Pevzner-Tang-Waterman
  - Celera Genomics Assembly Algorithm and the software engineering pipeline (an overveiw)
  - Haplotype Assembly - the HapCompass Algorithm

  Statistical Theory

  - Introduction to Genome Assembly - The Sequence of the Human Genome
  - Statistical Theory of Genome Assembly: The Lander-Waterman Formulas
  - Three Fundamental problems and the derivation of their mathematical statistics solution/formulas
    * Problem 1: What is the mean portion of the genome covered by the contigs of the assembly
    * Problem 2: What is the mean number of contigs of the assembly?
    * Problem 3: What is the man contig size of the assembly
  - Ham Smith's DNA Sequencing Lab - "No Windows Allowed" problem

  PIONEER: Hamilton Smith, Nobel Prize for Restriction Enzymes, pioneer of genome sequencing at Celera Genomics

- **Chapter 3: Hidden Markov Models Algorithms: The Learning Problem**

  Algorithms

  - Hidden Markov Models: Three Fundamental Problems and their Algorithmic Solutions.
    1. Problem 1: THE MODEL EVALUATION PROBLEM (computing the probability, covered in CSCI 1810)
    2. Problem 2: THE DECODING/REVEALING THE "HIDDEN PATH" PROBLEM ("Best explanation"/Viterbi maximum likelihood, covered in CSCI 1810)

3. Problem 3: THE LEARNING PROBLEM

Statistical Theory

- Maximum likelihood and the Expectation-Maximization (EM) Algorithm
- Probabalistic finite automata

PIONEER: Andrew Viterbi - pioneer of coding and decoding theory algorithms.

- **Chapter 4: Recombination and Ancestral Recombination Graphs Algorithms**

  Algorithms

  - The Miniciello-Durbin Algorithm
  - Ancestral Recombination Graphs reconstruction algorithms

  Statistical Theory

  - The Zolner-Pritchard Theory - mapping cases and controls on the ARG
  - Coalescent theory

- **Chapter 5: Rigorous Clustering: Spectral Graph Theory Algorithms**

  Algorithms and Statistical Theory

  - An introduction to Linear Algebra foundations for graph theory
  - Theory of Clustering Algorithms
  - Graph Laplacians
  - Graph cuts and random walks intuition for spectral clustering
  - Unnormalized Spectral Clustering Algorithms
  - Normalized Spectral Clustering Algorithms
  - Algorithmic Fairness and Clustering

- **Chapter 6: Algorithms for Constructing Suffix Trees in Linear Time**

  Algorithms

  - Linear time Suffix Tree Algorithm
  - Burrows-Wheeler Transform Algorithm and the Positional BW Transform Algorithm

  PIONEER: Alberto Apostolico, pioneer of Combinatorial Pattern Matching Algorithms in Bioinformatics

- **Chapter 7: Protein Folding Algorithms (introduction)**

  The Computational Protein Folding Problem is one of the grand challenge problems in computational biology, biotechnology, biophysics, biochemistry, mathematics, statistics, and computer science. We will discuss some puzzle problems related to lattice protein folding problems that witness the exceedingly difficult computational problems of this area.

Finally... Please come often to both the professor's office hours as well as to the TAs office hours.