COMP 4820 – Bioinformatics

Course Description

Calendar entry

An exploration of bioinformatics problems through the lens of Computer Science. Students will discover novel data structures, algorithmic tools, and techniques used to manage, index, and analyze large amounts of data. May not be held with the former COMP 3820. Prerequisite: COMP 3170.

General Course Description

By this point students have learnt the basics of analyzing the time and space requirements of algorithms. They have seen how simple algorithms are designed and how they can be implemented in different languages.

In this course we explore classical bioinformatics problems through the lens of computer science. We teach how these problems have been solved using different types of approaches, data structures and algorithmic techniques. We analyze the time/space requirements, advantages and disadvantages of these different approaches. We see how the algorithms behave on different types of input datasets and practice how to implement them efficiently.

Along the way, we explore different problems (exact and approximate text searching, database indexing, etc.), data structures (prefix/suffix trees, graphs, Bloom filters, etc.) and techniques (dynamic programming, branch and bound, Hidden Markov Models, etc.) that also have many applications in other areas of computer science.

Detailed Prerequisites

Before entering this course, a student should be able to:

- Understand the basic concepts of time and space complexity of an algorithm.
- Implement and use data structures such as lists, matrices, dictionaries and binary trees in a high-level language.
- Find strategies to translate algorithmic steps into code.

Course Goals

By the end of this course students will:

- Have a basic understanding of the types of datasets and problems that are encountered in bioinformatics.
- Associate how different algorithmic techniques and data structures can be used to solve some specific types of problems.
- Gain experience in the processing of large inputs and the implementation of efficient algorithms.
- Begin to understand how to deal with problems that have a very large search space, and evaluate the trade-offs between speed and accuracy of algorithms.

Learning Outcomes

Biology review

Students should be able to:

1. Use clear and precise vocabulary to describe the biological data, molecules and concepts that will be used in the course.
2. Describe the central dogma of molecular biology.

Exact pattern matching

Students should be able to:

1. Implement different algorithms for single pattern matching, such as the naive sliding-window approach, Boyer-Moore-Horspool (BMH) and Knuth-Morris-Pratt (KMP).
2. Build the trie that is required for running the Aho-Corasick algorithm for multiple pattern matching.
3. Describe how the search algorithm of the Aho-Corasick algorithm uses the trie.
4. Compare and contrast the different algorithms for single and multiple pattern matching in terms of time complexity.

Sequence alignments

Students should be able to:

1. Choose and justify the use of an appropriate sequence alignment algorithm for a given problem.
2. Implement the Needleman-Wunsch (with and without affine gap penalties), Smith-Waterman and ClustalW algorithms.
3. Choose and justify the use of nucleotide (transition/transversion) and amino-acid (PAM and BLOSUM) substitution matrices.
4. Describe the drawbacks of progressive multiple sequence alignment approaches and how the T-COFFEE algorithm attempts to mitigate them.

5. Compare and contrast these algorithms in terms of time and space complexity.

Database searching

Students should be able to:

1. Describe the main steps of the BLAST and Gapped BLAST search algorithms.
2. Compare and contrast regular search seeds with spaced seeds.
3. Compare and contrast the advantages/disadvantages in databases of using full genomes vs a k-mer representation of genomes (as in Bloom filters).

Phylogenetic trees

Students should be able to:

1. Use correct and precise vocabulary to describe the main characteristics of phylogenetic trees, distance matrices and metric/ultrametric spaces.
2. Implement distance-based phylogenetic tree construction methods such as WPGMA, UPGMA, Fitch-Margoliash and Neighbor-Joining.
3. Explain the difference between the small parsimony problem and the large parsimony problem in character-based phylogenetic construction.
4. Implement Fitch's and Sankoff's algorithms for the small parsimony problem.
5. Describe a variety of strategies for solving the large parsimony problem, such as branch and bound, local exploration (using tree rearrangements) and probabilistic (MCMC) approaches.
6. Compare and contrast the advantages and disadvantages of distance-based and character-based methods of phylogenetic tree construction.
7. Use the Newick format to represent trees in a text file.

Gene prediction

Students should be able to:

1. Explain why predicting/finding genes in a genome is a difficult problem.
2. Describe the main components of a Hidden Markov Model, or HMM: states, alphabet of output tokens, transition probability matrix and emission probability matrix.
3. Calculate the probability that a given path of states through a HMM produced a given output sequence.
4. Implement and describe the worst-case time complexity of the Forward and Viterbi algorithms.
5. Train an HMM using a labeled dataset.
6. Evaluate the performance of an HMM using sensitivity and precision.

Genome sequencing

Students should be able to:

1. Explain and compare the advantages and disadvantages of the 3 generations of sequencing technologies.
2. Describe the genome assembly problem and a general framework to solve it (overlap-layout-consensus).