Lecture 1: Introduction

- Arrangements and Overview of the Course
- Motivation: Molecular Sequence Data
- Base model: Strings and Sequences
- Methodology: Analysis of Algorithms
See course home page
http://www.cs.uku.fi/~kilpelai/BSA07 for details, notes, assignments and announcements

- graduate course in Computer Science (6 ECTS)
- about algorithmic methods applicable to molecular sequence data (DNA, RNA, protein)
- Language of instruction: English or Finnish(?)

For a rough syllabus, see
http://www.cs.uku.fi/~kilpelai/BSA07/syllabus.html
Arrangements

- 16 lectures, 7 exercise sessions
- Final exam on March 23; retake on April 20
- Grading: \( \text{round}(6 \times E + 2 \times H - 2.5) \), where
  \[ E = \frac{\text{(exam points)}}{\text{(max exam points)}} \], and
  \[ H = \text{fraction of solved homework assignments} \];
  \( E \geq 0.5 \) required to pass

Course is based on the textbook *Algorithms on Strings, Trees, and Sequences* by D. Gusfield, out of which we cover, selectively, Parts I–III.
What is this course about?

String algorithms that operate on molecular sequence data. These are treated

- in CS within Stringology (merkkijonoalgoritmien tutkimus) or Combinatorial Pattern Matching (kombinatorinen hahmonsovitus)

- in Biology within Computational Biology or Bioinformatics

Emphasis on ideas and methods that are applicable to bio-sequence related problems of today, and, hopefully, of the future
What this course is NOT about?

This is **NOT** a complete course on bioinformatics. We **do NOT**

- treat statistical methods, or molecular structures other than sequences
- study the use of specific computer packages, databases or services

Also, little attention is paid to the implementation (programming) of the methods.
Rough Syllabus

0. Introduction
1. Exact String Matching
2. Suffix Trees
3. Approximate Matching and Alignments
Motivation: Molecular Sequence Data

(I try to outline some central issues, even though I’m NOT an expert in Biology.)

Genetic material: DNA (deoxyribonucleic acid)

- a polymer of nucleotides
  - phosphate group + ribose sugar + base
- essentially a string of bases (emäs) denoted by A, C, G and T (adenine, cytosine, guanine, and thymine)

The sequence of nucleotides, identified by their bases, determines the genome of an organism.
Production of Proteins

In a process called gene expression:

1. **Transcription** DNA information is copied into RNA, with base U (uracil) replacing T (thymine)
   - in so called 5′ → 3′ direction
2. RNA is translated (by ribosomes) into a protein.

Proteins are

- polymers made of 20 different **amino acids**;
- central for life, for example, as material of cells and as enzymes.

Protein is, roughly, “the meaning of a gene”. (They also produce RNA for ribosomes)
Ribosomes locate triplets of bases (codons) in the RNA, and create amino acids for them in the resulting protein.

- Start codon AUG also encodes methionine.
- Triplets UAA, UAG and UGA act as stop codons.

Genetic code is redundant ($4^3 - 3 = 61 > 20$), and thus robust: single-base mistakes do not necessarily effect the encoded amino acid sequence.
Lots of non-coding junk (residue?) appears in the genome (~ 95 % for human)

- between genes and
- as **introns** btw encoding regions called **exons**

For example, the human gene associated with **cystic fibrosis** has

- total length over $10^6$ nucleotides
- about 1000 nucleotides, in 25 exons
  (< 0.1% of total gene length)

In most bacteria, most of DNA (~ 85%) is in genes, and introns are rare
Relevance of Primary Structure

A lot of biologically relevant information can be inferred from amino acid order (primary structure) alone (even though proteins are actually complex 3D structures; also, much less of the latter are known).

First fact of biological sequence analysis (Gusfield, Sect. 10): High sequence similarity usually implies significant functional or structural similarity.

Locating sequences of a data base that are similar to a new one, or locating conserved subsequences (signatures or motifs) in related sequences is central activity in Molecular Biology.
Some statistics of genetic material

The length of …

- a gene is a few kb’s (kb = 1000 base pairs)
  - an average human gene is ~ 3 kb
- most proteins are hundreds of amino acids (≤ 500)
- the entire genome a few million nucleotides for prokaryotes (e.g, bacteria) (esitumalliset)
  - … billions for eukaryotes (aitotumalliset)
  - worm 100 Mb (100 · 10^6 base pairs)
  - human 3 Gb (3 · 10^9 base pairs)
- # of human genes estimated 20,000–30,000
Statistics of molecular sequences

In mid-90’s (Gusfield, Sect. 15.1)

- ~ 300,000 genes (or parts of them, of different organisms) stored in DNA archives, totaling > 500 Mb (and growing ~ 75 %/year)

- ~ 100,000 different protein sequences in major archives, totaling about $25 \cdot 10^6$ amino acids

The GenBank sequence database (in Aug. ’06):

- ~ $60 \cdot 10^6$ nucleotide sequences, of total length $65 \cdot 10^9$ b

- doubles every 10 months
A **string** $S$ (*merkkijono*) is an ordered list of characters $s_1 \ldots s_n$; $|S| = n$ is the **length** of $S$.

A **substring** (*osajono*) is a contiguous region $S[i \ldots j]$ of $S$:

1. If $i \leq j$, $S[i \ldots j] = s_i \ldots s_j$, and $|S[i \ldots j]| = j - i + 1$;
2. If $i > j$, $S[i \ldots j] = \epsilon$ (**empty string**), and $|S[i \ldots j]| = 0$;
3. $S[1 \ldots i]$ is a **prefix** (*alkuosa*) of $S$. If $i < |S|$, the prefix is **proper** (*aito alkuosa*)
4. $S[i \ldots |S|]$ is a **suffix** (*loppuosa*) of $S$. If $i > 1$, the suffix is **proper** (*aito loppuosa*)
A subsequence (alisekvenssi) $s_{i_1}s_{i_2}\ldots s_{i_k}$ of $S = s_1\ldots s_n$ is an ordered selection of some $k \geq 0$ characters of $S$, that is, $1 \leq i_1 < i_2 < \cdots < i_k \leq |S|$.

Example: String “California”

- suffixes: $\epsilon$, “a”, “ia”, “nia”, . . . , “California”
- some substrings: “lifo”, “forni”
- some subsequences: “Carni”, “alora”
Methodology: Asymptotic Analysis

Efficiency of algorithms is estimated in terms of (worst-case) **complexity**, i.e., dependency of (maximally needed) resources (time, space) on input size.

Standard **asymptotic notations** for upper and lower bounds:

1. \( f(n) = O(g(n)) \) ("of order \( g(n) \)"")
   \[ \text{iff } f(n) \leq cg(n) \text{ for some } c \text{ and all sufficiently large } n \]

2. \( f(n) = \Omega(g(n)) \) ("at least of order \( g(n) \)"")
   \[ \text{iff } g(n) = O(f(n)), \text{ and} \]

3. \( f(n) = \Theta(g(n)) \) ("exactly of order \( g(n) \)"")
   \[ \text{iff } f(n) = O(g(n)) \text{ and } f(n) = \Omega(g(n)) \]
Observations and refinements

Insignificance of constant coefficients $c > 0$:

$$c \times f(n) = \begin{cases} O(f(n)) \\ \Theta(f(n)) \\ \Omega(f(n)) \end{cases}$$

$\sim$ programmer competence, compiler and HW ignored — Focus is on scalability wrt increasing input size ($n$)

A stronger version of $f(n) = O(g(n))$:

$\circ$ $f(n) = o(g(n))$ iff $\lim_{n \to \infty} f(n)/g(n) = 0$

(“of strictly lower order than $g(n)$”)
Simplification rules

Insignificance of lower-order terms:
\[ g(n) = o(f(n)) \Rightarrow f(n) \pm g(n) = \Theta(f(n)) \]

Transitivity:
\[ f(n) = \Theta(g(n)) \text{ and } g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n)) \]

Example:
\[\sum_{i=1}^{n} 4i = 4 \frac{n(n+1)}{2} = 2(n^2 + n) = \Theta(n^2 + n) = \Theta(n^2)\]
**Worst case vs Average case**

**Average case complexity** would often be informative, but worst-case complexity

- is often much easier to derive, and
- guarantees that the real complexity is never worse than estimated.

~~ most often restricted to worst-case estimates
Relevance of Asymptotics?

Asymptotic estimates hide a lot of information. Are they useful?

- Yes: provide an implementation-independent characterization of the scalability of algorithms

Do they tell of practical efficiency?

- In principle, no

- Often, yes: an asymptotically less efficient algorithm could be more efficient in practice, but only on small inputs

With practical algorithms, experimenting on real data sets is the right thing to do!
Relevance for Computational Biology?

- Asymptotics \(\sim\) inputs growing without limit
- Sequence DBs grow, but not infinitely
- Patterns of interest, say, proteins, have a fixed size

Personal view: With current technology, many sequences of interest are large enough for asymptotics to reflect the real usefulness of algorithms

**Ex:** Processing human genome \((n = 3 \cdot 10^9)\); ass. \(10^9\) ops/s

- with \(n\) ops \(\rightarrow\) 3 sec
- with \(n^2\) ops \(\rightarrow\) \(9 \cdot 10^9\) sec \(\approx\) 285 years
Review basics of complexity analysis from algorithms course notes or some textbook (e.g., Cormen, Leiserson and Rivest: *Introduction to Algorithms*)