

Biosequence Algorithms Spring 2005 lecture notes

Pekka Kilpeläinen
University of Kuopio
Department of Computer Science

BSA Lecture 1: Introduction - p.1/21

Lecture 1: Introduction

- Arrangements and Overview of the Course
- Motivation: Molecular Sequence Data
- Base model: Strings and Sequences
- Methodology: Analysis of Algorithms

BSA Lecture 1: Introduction - p.2/21

Arrangements and Overview

See course home page
<http://www.cs.uku.fi/~kilpelai/BSA05> for details, notes,
assignments and announcements

- graduate (*laudatur*) course in Computer Science (3 cu)
- about algorithmic methods applicable to the exploitation of molecular sequence data (DNA, RNA, protein)
- Language of instruction: English or Finnish(?)

For a rough syllabus, see
<http://www.cs.uku.fi/~kilpelai/BSA05/syllabus.html>

BSA Lecture 1: Introduction - p.3/21

Arrangements

- 16 lectures, 7 exercise sessions
- Home exam (tentatively) due to March 23
- (Retake on April 29)
- Grading: $[12 * E + 4 * H - 3]$, where
 $E = (\text{exam points}) / (\text{max exam points})$, 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 plan to cover, selectively, Parts I–III.

BSA Lecture 1: Introduction - p.4/21

What is this course about?

Deterministic **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

BSA Lecture 1: Introduction - p.5/21

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.

BSA Lecture 1: Introduction - p.6/21

Motivation: Molecular Sequence Data

(I try to offer simple explanations of 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.

BSA Lecture 1: Introduction - p.7/21

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)

BSA Lecture 1: Introduction - p.8/21

Translation and the Genetic Code

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

BSA Lecture 1: Introduction - p.9/21

Genome vs. Proteins

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

BSA Lecture 1: Introduction - p.10/21

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.

BSA Lecture 1: Introduction - p.11/21

Some statistics of genetic material

The length of ...

- a gene is a few kb's (kb = 1000 base pairs)
 - average human gene is 30 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 ($100 \cdot 10^6$ base pairs)
 - human 3 Gb ($3 \cdot 10^9$ base pairs)
- # of human genes estimated 20,000–25,000 (10^4)

BSA Lecture 1: Introduction - p.12/21

Statistics of molecular sequences

(Gusfield, Sect. 15.1)

Number of ...

- genes (or assumed coding regions) in the first completely sequenced DNA of a free-living organism (*Haemophilus influenzae* rd, 1995) was 1,743

In mid-90's

- ~ 300,000 genes (or parts of them, of different organisms) stored in DNA archives, totaling > 500 Mb (with growth of ~ 75%/year)
- ~ 100,000 different protein sequences in major archives, totaling about 25,000,000 amino acids

BSA Lecture 1: Introduction - p.13/21

Base model: Strings and Sequences

A **string** S (*merkkijono*) is a left-to-right list of characters $s_1 \dots s_n$; $|S| = n$ is the **length** of S .

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

- if $i \leq j$, $S[i \dots j] = s_i \dots s_j$, and $|S[i \dots j]| = j - i + 1$;
- Otherwise $S[i \dots j] = \epsilon$, and $|S[i \dots j]| = 0$ (**empty string**)
- $S[1 \dots i]$ is a **prefix** (*alkuosa*) of S . If $i < |S|$, the prefix is **proper** (*aito alkuosa*)
- $S[i \dots |S|]$ is a **suffix** (*loppuosa*) of S . If $i > 1$, the suffix is **proper** (*aito loppuosa*)

BSA Lecture 1: Introduction - p.14/21

Subsequences of strings

A **subsequence** (*alisekvenssi*) $s_{i_1} s_{i_2} \dots s_{i_k}$ of $S = s_1 \dots s_n$ is an ordered selection of some $k \geq 0$ characters of S , that is, $1 \leq i_1 < i_2 < \dots < i_k \leq |S|$.

Example: String "California"

- prefixes: ϵ , "C", "Ca", "Cal", ..., "California"
- suffixes: ϵ , "a", "ia", "nia", ..., "California"
- some substrings: "lifo", "forni"
- some subsequences: "Carni", "alora"

BSA Lecture 1: Introduction - p.15/21

Methodology: Asymptotic Analysis

We estimate the efficiency of algorithms 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:

- $f(n) = O(g(n))$ ("of order $g(n)$ ")
iff $f(n) \leq cg(n)$ for some c and all sufficiently large n
- $f(n) = \Omega(g(n))$ ("at least of order $g(n)$ ")
iff $g(n) = O(f(n))$, and
- $f(n) = \Theta(g(n))$ ("exactly of order $g(n)$ ")
iff $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$

BSA Lecture 1: Introduction - p.16/21

Observations and refinements

Insignificance of constant coefficients $c > 0$:

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

↪ programmer competence, compiler and HW ignored —
Focus is on *scalability* wrt increasing input size (n)

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

- ⑥ $f(n) = o(g(n))$ iff $\lim_{n \rightarrow \infty} f(n)/g(n) = 0$
("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:

$$\begin{aligned} \sum_{i=1}^n 4i &= 4 \frac{n(n+1)}{2} = 2(n^2 + n) \\ &= \Theta(n^2 + n) \\ &= \Theta(n^2) \end{aligned}$$

Worst case vs Average case

Average case complexity would often be informative, but when compared to worst-case complexity, it

- ⑥ is often much more difficult to derive, and
- ⑥ does not guarantee 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?

- ⑥ provide an **implementation-independent** characterization of the **scalability** of algorithms
- Do they tell of practical efficiency?

- ⑥ In principle, **no**
- ⑥ Often, **yes**: an asymptotically less efficient algorithm *could* be more efficient in practice, but *only* on small inputs

Experimenting practical algorithms on real data sets is the right thing to do!

Relevance for Computational Biology?

- ⑥ Asymptotics \sim inputs growing without limit
- ⑥ Sequence DBs grow, but not infinitely
- ⑥ Patterns of interest, say, proteins, have a fixed size

Personal belief: With current technology, sequence collections and patterns of interest are "large enough" such that asymptotic estimates reflect the real usefulness of algorithms.

Reading assignment: Review basics of complexity analysis from algorithms course notes or some textbook (e.g., Cormen, Leiserson and Rivest: *Intro to Algorithms*)