Instructor: Brian Roark
Class time: Tu/Th 4:00 PM - 5:30 PM
Jan. 4 - Mar. 17, 2011
Class location: WCC 403, videoconferenced to BICC 131B
Office hours: Tu 10-12, Central 115, or by appt
Required texts:
Course overview

• Not a course on molecular biology
  – Rather a course on effective computational approaches to problems posed in molecular biology (and linguistics)

• What the hell do molecular bio and linguistics have in common?
  – Sequences of symbols from a discrete alphabet
  – Large collections of such sequences to search
  – Notions of functional similarity and approximate matching
  – Evolution
  – Dependencies in sequences between symbols arbitrarily far apart
  – Algorithms and models for effective/efficient analysis
Algorithms and models

- Two texts for this course, with differing emphasis
  - Gusfield focuses on efficient algorithms
  - Durbin et al. focuses on effective models
- In fact, decisions on modeling are impacted by considerations of efficient algorithms
- For example, hidden Markov models (HMMs)
  - Most widely used approach for modeling these sequences
  - Not chosen to accurately model underlying dependencies
  - Rather, they are passably effective and have more efficient inference algorithms than richer models such as PCFGs
Agenda for today

• Introduction to sequences and strings
  – Biological sequences
  – Linguistic sequences
  – Formal representation

• Overview of main problems
  – String matching
  – Sequence alignment
  – Evolution and change
  – Higher order structures

• Overview of course structure
Sequences and strings

• A string $S \in \Sigma^*$ is an ordered list of zero or more symbols from an alphabet $\Sigma$
  – “told John to shut up” ($S_1 \in \Sigma^{20}$, $\Sigma = \text{ASCII}$)
  – “LSFAAAMNGLA” ($S_2 \in \Sigma^{11}$, $\Sigma = \text{amino acids}$)
  – Special string: $\epsilon$ of length 0
  – $|S|$ denotes the length of $S$, i.e., $S \in \Sigma^{|S|}$

• A substring $S[i, i+j] \in \Sigma^{j+1}$ for some $j \geq 0$ is a continuous list beginning at the $i^{\text{th}}$ symbol and consisting of $j+1$ symbols, e.g., $S_1[11, 17] = \text{“to shut”}$
  – $S[1, 1+j]$ is a prefix
  – $S[k, |S|]$ is a suffix

• A subsequence need not be contiguous (“toldJohnntoshutup”)

• A proper substring or subsequence is neither $S$ nor $\epsilon$
Biological sequences

• Two kinds of biological sequences of interest
  – Nucleic acids (DNA and RNA), which are sequences of nucleotides
    · DNA consists of Adenine (A), Guanine (G), Cytosine (C) & Thymine (T)
      \[ \Sigma = \{A, G, C, T\} \]
    · RNA consists of A, G, C and Uracil (U) \[ \Sigma = \{A, G, C, U\} \]
  – Proteins, which are sequences of amino acids (|\(\Sigma\)| = 20)

• The Central Dogma of Molecular Biology: DNA → RNA → protein
  – DNA information can be transferred to DNA (*replication*)
  – DNA information can be transferred to RNA (*transcription*)
  – RNA information can be transferred to proteins (*translation*)
DNA

- *Genes* are strands of DNA that contain *coding* and *non-coding* regions

- Gene expression results in the functional product of the gene
  - Coding regions are transcribed to RNA
    - *messenger* RNA (mRNA) for translation to proteins
    - *transfer* RNA (tRNA) and *ribosomal* RNA (rRNA) for mediation of translation
  - Non-coding regions regulate this expression

- DNA strands are typically paired, with the familiar double helix structure
  - A and T can bond to form a base pair
  - C and G can bond to form a base pair
Like DNA, RNA also has nucleotide $\Sigma = \{A, G, C, U\}$, with U in place of T

mRNA encode protein sequences through codons

- units of three nucleotides which identify an amino acid
- creates a “reading frame” problem: where do codons begin?

tRNA is required for translation to protein sequences

No double strands with RNA, resulting in secondary structures

- nucleotides in the sequence form base pairs, resulting in helices and loops
# Amino acids and codons

<table>
<thead>
<tr>
<th>Amino Acid</th>
<th>RNA codons</th>
<th>Amino Acid</th>
<th>RNA codons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ala (A)</td>
<td>GCU, GCC, GCA, GCG</td>
<td>Leu (L)</td>
<td>UUA, UUG, CUU, CUC, CUA, CUG</td>
</tr>
<tr>
<td>Arg (R)</td>
<td>CGU, CGC, CGA, CGG, AGA, AGG</td>
<td>Lys (K)</td>
<td>AAA, AAG</td>
</tr>
<tr>
<td>Asn (N)</td>
<td>AAU, AAC</td>
<td>Met (M)</td>
<td>AUG</td>
</tr>
<tr>
<td>Asp (D)</td>
<td>GAU, GAC</td>
<td>Phe (F)</td>
<td>UUU, UUC</td>
</tr>
<tr>
<td>Cys (C)</td>
<td>UGU, UGC</td>
<td>Pro (P)</td>
<td>CCU, CCC, CCA, CCG</td>
</tr>
<tr>
<td>Glu (Q)</td>
<td>CAA, CAG</td>
<td>Ser (S)</td>
<td>UCU, UCC, UCA, UCG, AGU, AGC</td>
</tr>
<tr>
<td>Glu (E)</td>
<td>GAA, GAG</td>
<td>Thr (T)</td>
<td>ACU, ACC, ACA, ACG</td>
</tr>
<tr>
<td>Gly (G)</td>
<td>GGU, GGC, GGA, GGG</td>
<td>Trp (W)</td>
<td>UGG</td>
</tr>
<tr>
<td>His (H)</td>
<td>CAU, CAC</td>
<td>Tyr (Y)</td>
<td>UAU, UAC</td>
</tr>
<tr>
<td>Ile (I)</td>
<td>AUU, AUC, AUA</td>
<td>Val (V)</td>
<td>GUU, GUC, GUA, GUG</td>
</tr>
</tbody>
</table>
Reading frame problem

- Consider the substring \ldots AUGGACGGA\ldots

- Three possible codon sequences:
  \ldots(AUG)(GAC)(GGA)\ldots \Rightarrow (\text{Met})(\text{Asp})(\text{Gly})
  \ldots A(UGG)(ACG)GA\ldots \Rightarrow (\text{Trp})(\text{Thr})
  \ldots AU(GGA)(CGG)A\ldots \Rightarrow (\text{Gly})(\text{Arg})

- Exercise 10 in Chapter 1 of Gusfield asks:
  How can we find all three translations for a string of length $n$ in at most $n$ character examinations and indexing steps

- Efficient algorithms for general problems
Amino acids

- Amino acid consists of
  - a central carbon atom ($C_\alpha$)
  - an amino group ($NH_2$)
  - a carboxyl (or acid) group (COOH)
  - and a side chain (R)
Peptide bonds

• H from the Amino group of one amino acid combines with OH of the carboxyl group of another to produce H₂O and a peptide bond

• For example, R=CH₃ (Alanine, Ala/A) and R=H₃CHCH₃ (Valine, Val/V) combine via the amino group of A and the carboxyl group of V

• The peptide unit (or group) consists of the C=O and N-H bonds linking the Cα atoms of the amino acids
Polypeptide chains

- These chains are the amino acid sequences that constitute proteins.
- Like RNA, protein sequences fold into higher order structures.
Protein secondary structure

- Two key kinds of secondary structure
  - $\alpha$ helices
  - $\beta$ sheets

- $\beta$ sheets are formed through loops
  - bonds can form parallel or anti-parallel sheets
α helices
Hairpin loops
Parallel $\beta$ sheets
Anti-parallel $\beta$ sheets
Tertiary structure and beyond

- Beyond secondary structures of these sorts, higher order structures:
Biological sequences (summary)

- Nucleic acid (DNA/RNA) and Amino acid (protein) sequences
- DNA → RNA → protein
- Alphabets $\Sigma$ of size 4 and 20
- Secondary structures in RNA and proteins
  - folding and looping
Linguistic sequences

- Phonology
  - Sequences of phones form words, e.g.,
    \[\theta r o\] is the pronunciation of the word ‘throw’

- Morphology
  - Sequences of morphemes form words, e.g.,
    ‘work’+‘ed’ forms the past tense ‘worked’

- Syntax
  - Sequences of words form phrases: ‘the dog food’
  - Sequences of phrases form sentences:
    ‘the dog food tastes delicious’
Spoken versus written language

- **Spoken language**
  - Primary human faculty, hence primary focus of the field of Linguistics
  - Physical articulatory constraints on sounds and sound sequences
  - Large differences between languages in phonetic inventory

- **Written language**
  - Relatively recent technological development
  - Different kinds of writing systems, e.g.,
    - Logographic – symbols represent morphemes (Chinese)
    - Alphabetic – symbols represent phonemes (Latin)
    - Syllabic – symbols represent syllables (Japanese kana)
  - Many ordering and delimiter differences between languages, e.g., Arabic (right to left) and Chinese (no spaces between words)
Higher order structures

- Like bio-sequences, both primary and secondary structures
- Primary structures: phonological and morphological sequences
- Secondary structures: syntactic structures
  - Requires context-free grammars to describe:

```
S
 / 
NP VP
 / 
NP PP tastes ADJP
 / the dog food in NP delicious
 / the green can
```

Linguistic sequences (summary)

• Sequences from different $\Sigma$
  – Phones (sounds)
  – Syllables
  – Words
  – Letters
  – ASCII/Unicode Characters

• Long-distance dependencies in syntax

• World languages differ in these sequences, yet can express the same semantic content (translation)
Formal representation

• For an alphabet of symbols $\Sigma$
  – $\Sigma^*$ is the set of strings of symbols in $\Sigma$ of length 0 or more
  – $\Sigma^+$ is the set of strings of symbols in $\Sigma$ of length 1 or more
  – $\Sigma^k$ is the set of strings of symbols in $\Sigma$ of length $k$
  – $\epsilon$ is the empty string

• A string $S \in \Sigma^*$ is an ordered list of zero or more symbols from an alphabet $\Sigma$
  – Let $S[i, j]$ be the substring beginning at the $i^{th}$ and ending at the $j^{th}$ symbol
  – By convention, if $j < i$, $S[i, j] = \epsilon$
  – Let $S(i) = S[i, i]$
  – By definition, $S = S[1, |S|]$
  – $S[1, k]$ is a prefix of $S$ (proper if $k < |S|$)
  – $S[j, |S|]$ is a suffix (proper if $j > 1$)
Overview of main problems

• String matching
  – Find all examples of a string in a database/corpus
  – Find all examples of a set of strings in a database/corpus

• Sequence alignment
  – Find all “close” examples of a string

• Evolution and change
  – Find functionally similar sequences derived from a common ancestor

• Higher order structures
  – Find functionally critical long distance dependencies
Why are these problems?

• Domain experts (biologists/linguists) characterize key structures and dependencies in sequences they study
  – Plenty that is not well understood
  – Many solutions rely on people as pattern recognizers

• Given a new string or set of strings
  – Number of possible structures is typically exponential in the length of the string(s)
  – Must avoid enumerating all of them while searching for the best
  – The best models often make search more expensive

• Clever efficiencies can make use of rich models feasible
Exact string matching

- Given a short *pattern* and a large *text*, find where the *pattern* occurs in the *text*
- Naive methods have $O(nm)$ worst-case complexity
- Want at least worst-case linear $O(m)$, typical sub-linear
- Want to be able to search for large sets of patterns efficiently
- If the *text* is given well in advance, would like to pre-process it to speed up subsequent searches
- Core exact-match algorithms form the basis of some efficient approximate matching algorithms
Approximate matching / alignment

• Most interesting matching problems involve small differences
  – Functionally similar amino acid substitution, deletions
  – Vowel shift or systematic consonant dropping
• Want to find “functionally similar” sequences
• Problem: how to define and measure similarity
• Approximate match aligns two sequences
• Find best alignment out of all possible alignments
  – Dynamic programming solution helps with efficiency
  – Further efficiency improvements can be had
• Maybe global alignment is irrelevant (best “local” alignments)
Evolution and change

• Organisms evolve – change manifest in sequences

• With multiple, phylogenically related sequences
  – Can find functionally similar regions across multiple sequences
  – Must learn models that model sets of related subsequences

• Evolutionary pressures do not apply uniformly
  – Some regions remain largely intact (presumably changes lethally reduce functionality)
  – Functional structures are often in secondary structure
  – Large differences in primary structure may be functionally equivalent
Higher order structures

- Secondary structure (due to folding, etc.), like syntax in natural language, requires context-free models (even context-sensitive)
- Rich models of long-distance dependencies
  - inference algorithms well studied in NLP
  - less efficient than finite-state (primary structure) models
  - Common structures quite different from language
- Higher order structures, akin to document structure
  - inference becomes less-and-less reliable and more-and-more expensive
Overall themes

• Clever algorithms can make a HUGE difference in speed
• Relatively simple models can be very effective
• Many problems in biological sequence processing are closely related to problems in linguistic sequence processing
  – Some problems are significant variations
• General techniques form the backbone of all processing
  – Fast exact match
  – Dynamic programming
  – Hidden Markov models, probabilistic grammars
Course content

● About general algorithms and models, not linguistics or biology
  – Three+ weeks on exact matching (motivated by approximate)
  – Two weeks on approximate matching / alignment
  – Two weeks on multi-sequence families / evolution
  – One week on higher-order structures and other problems
  – One final week of student presentations

● Four homeworks – hands on, real data, coding (40% of grade)
● In-class midterm (15%) and final project (20%)
● More content on biological than linguistic sequences
Course logistics

• Lectures, readings and homeworks linked from syllabus, found here:
  http://www.cslu.ogi.edu/people/roark/courses/cse555-BLS/cse555w11.html

• Two books: Gusfield, mainly about algorithms; Durbin et al., mainly models

• Some may find homeworks to be hard, don’t procrastinate!
  – Homework policy: Turn in what you have by due date, otherwise no credit
  – More credit for partial solutions at deadline than anything submitted after
  – Prefer email submit; code + writeup; 2 weeks for each

• Wrapper code provided for HWs in C and Java, but you can choose your own
  language, as long as it is sufficiently commented
  – Wrapper code provides I/O and evaluation, with placeholders/comments

• Some changes from prior years: more student presentations in class, some new
  content, less linguistic content