582670 Algorithms for Bioinformatics

Lecture 2: Exhaustive search and motif finding

5.9.2013

These slides are based on previous years’ slides of Leena Salmela and Veli Mäkinen. These slides use material from http://bix.ucsd.edu/bioalgorithms/slides.php
Outline

Biological motivation

Implanted motifs - an introduction

Motif Finding Problem and Median String Problem

Structuring the search: search tree

Branch and bound

Weeder

Study group assignments
Biological Motivation
Biological Motivation (cont’d)

Microarray experiment

[Diagram showing gene interactions and expression patterns]
Gene Regulation

- Microarray experiments can be used to measure gene activity
- A gene can be knocked out to see what effect that has on gene activity
- An experiment can show that when one gene (gene X) is knocked out, 20 other genes stop being expressed.
- How can one gene have such a drastic effect?
Gene X encodes a regulatory protein, a.k.a. a transcription factor (TF)
The 20 unexpressed genes rely on gene X’s TF to induce transcription
A single TF may regulate multiple genes
Regulatory Regions

- Every gene contains a regulatory region (RR) typically stretching 100-1000 bp upstream of the transcriptional start site.
- Located within the RR are the Transcription Factor Binding Sites (TFBS), also known as motifs, specific for a given transcription factor.
- TFs influence gene expression by binding to a specific location in the respective gene’s regulatory region - TFBS.
Transcription Factor Binding Sites

- A TFBS can be located anywhere within the regulatory region
- TFBS may vary slightly across different regulatory regions since non-essential bases could mutate
Motifs and Transcriptional Starting Sites

- ATCCCG
- TTCCGG
- ATCCCG
- ATGCCG
- ATGCCCG
Motifs can mutate on non important bases

The five motifs in five different genes have mutations in positions 3 and 5

Representations called motif logos illustrate the conserved and variable regions of a motif
Identifying Motifs

- Genes are turned on or off by regulatory proteins.
- These proteins bind to upstream regulatory regions of genes to either attract or block an RNA polymerase.
- Regulatory protein (TF) binds to a short DNA sequence called a motif (TFBS).
- So finding the same motif in multiple genes’ regulatory regions suggests a regulatory relationship among those genes.
Identifying Motifs: Complications

- We do not know the motif sequence
- We do not know where it is located relative to the gene’s start
- Motifs can differ slightly from one gene to the next
- How to discern it from “random” motifs?
Outline

Biological motivation

Implanted motifs - an introduction

Motif Finding Problem and Median String Problem

Structuring the search: search tree

Branch and bound

Weeder

Study group assignments
Random Sample

atgaccgggatactgataccttgctatttggcctaggcgtacacattagataaagctatgaaggtacgttagactccggtacgccg
accctattttttgagcagatttaggtgacctggaaaaaattttgagttacaaactttccgaatactgggcataaggatc
tgagtatccctggttgaactgggtaatgctttggaacactatagtgctctccccagatttttgaatattggatgatcattcc
gcgccaggggtcccggtgagaattggatgaccttgtaagtgttttccacgcaatcgcgaaccacctggaacacacgccgac
ctgtgaggttagagcgctgggttaggggccctgttagaggcccccgtactgatggaaactttcaattatgagagagctaatctatcgcgtgcgtgttc
tctgtagttgcgtaatgtgccgggaggtctgtactagttacgtagggaagctgtcatttattagcacttattggttcgttggttca
aacttgagtttggtttcgaaaatgtccttggggacatacaagagaggtctttctttatcagttaatgctgtatgacactatgt
ttgcccaattggctaaaagcccaacttgacaaatggaagatagaatccttgcatttcaacgtatgccgaaccgaaagggactggtgagcaacgacagattcttttagcccccttgacttggaacactttcaattttagagagagctaatctatcgcgtgcgtgttc
ttgsatctggctaaagcccaacccattggctaaaagcccaacttgacaaatggaagatagaatccttgcatttcaacgtatgccgaaccgaaagggactggtgagcaacgacagattcttttagcccccttgacttggaacactttcaattttagagagagctaatctatcgcgtgcgtgttc
Implanting Motif AAAAAAAAAGGGGGGGGGG

atgaccgggatactgat AAAAAAAAAGGGGGGGG gcgtacacattagataaaacgtatgaagtacgttagactcgccgccgccc
accctattttttgagcagatttttagtgacctggaaaaaatttgagtacaaaacttttccgaata AAAAAAAAAGGGGGGGG
tgagtatccctgggatgacct AAAAAAAAAGGGGGGGG tcgtctcccgatttttgaatatgtgagatcattgcaccagggtccgc
ctgagaatttggatg AAAAAAAAAGGGGGGGG tccacgcaatcgcgaaccaacgcggacccaaaggcaagaccgataaaggag
tcccttttgcggtaatgtggcgggaggctgtggtaacgttaggggaagccctaacggacatttaat AAAAAAAAAGGGGGGGGcttata
gtcaatcatgttcttttgtagatgtg AAAAAAAAAGGGGGGGG accgctttggcgacccaaattcagtggtgggccagcgcaggcgttttgcggcccttttgtagggcctggcggcctggccggagtgttcagtgaacgcagaattatgcattgtgagacaaggtgctttttgaaggggattcatagccagcacgaagctt AAAAAAAAAGGGGGGGG
acggccacctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
acggccacctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
acggccacctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
acggccacctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
acggccacctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
acggccacctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
acggccacctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
acggccacctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
acggccacctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
acggccacctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
acggccacctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
acggccacctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
acggccacctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
acggccacctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
acggccacctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
acggccacctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
acggccacctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
acggccacctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
acggccacctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
acggccacctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
acggccacctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
acggccacctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
acggccacctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
acggcc acctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
acggccacctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
acggccacctacacaagaggagtctttctttatcagttatagtgtatgacactatgtttgcccaatttgctaaagcccaacttgacaaatgggagatagaatccttgcaa AAAAAAAAAGGGGGGGG
Implanting Motif AAAAAAAAAGGGGGGGG with four mutations
Where is the implanted motif???

atgaccgggatactgatagaagaaggttgggggctacacatgatgataaactgatagttacgtagacgtcggcgcgcgtcccctttttttagcgtagtttgtgactgctctttcgatttttttagttttagctatggatcattgcaggtcggcgcgtgctgcaattgtcgttcgtgctttcttgagcagccgagcttttctgtggagttttggtgcgtggtaacctggagtttttagaagtagtaagtagtaaactgttagatctggcggccggcccttttttttcgtgatgctgtggattccagcttttttcttttttcgtctgctgtgtttggtgttggagaagccggcccttttttttccggtttttttcgtttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttt
Why finding (15,4)-motifs is hard?

Aligning two first occurrences of the motif

AgAAgAAAGGttGGG
AgAAgAAAGGttGGG

..|..|||.|..|||

cAAtAAAAcGGcGGG
cttata

Aligning two first occurrences of the motif

AgAAgAAAGGttGGG

..|..|||.|..|||

cAAtAAAAcGGcGGG
The Implanted Motif Problem

Finding a motif in a sample of

- 20 “random” sequences (e.g. 600 nt long)
- Each sequence containing an implanted pattern of length 15
- Each pattern appearing with 4 mismatches as (15,4)-motif
Outline

Biological motivation

Implanted motifs - an introduction

Motif Finding Problem and Median String Problem

Structuring the search: search tree

Branch and bound

Weeder

Study group assignments
The Motif Finding Problem

- Given a random sample of DNA sequences:

  cctgataagcgcctatctggttatccacgtacgtaggtcctctgtgcgaatctatgcgtttccaaccat
  agtactggtgtacattttgatacgtacgtacaccggcaacctgaacaacgcagtcagaaccagaagtgc
  aaacgtacgtgcaccctcttttttctcggtcgtgctctggccaacgagggctgatgtataagacgaaat
  ttttatccgtcatgtaagtcagctgtactattacctgccacccctattacatcttacgtacgtataca
  ctttttataaaccgcgtcatggccggggtatgcgtttttggtcgctgcgtacgcctcgatcgttaacgcg
tc

- Find the pattern that is implanted in each of the individual sequences, namely, the motif
The Motif Finding Problem

- Given a random sample of DNA sequences:
  
  cctgatagacgtatcttgtcctatccacgtacgttaggtcctctgtgcgaatctatgcgttttccaaccat
  agtacttgtggtacatttgatacgtacgtacaccggcaacctgaaacaaacgcctcagaacagagaagtgc
  aaacgtacgtgcaccctctttttcttcgtggctctggccaacgagggctgatgtataagacgaaatatattt
  agcctccgatgtaagtgctagctgtaactattacctgccacccctattacatcttacgtacgatataca
  cttgattatacaacgcgtcatggcggggtatgcggttttggtcgtcgtacgtacgtcgtacgttaacgtacgtc

- Find the pattern that is implanted in each of the individual sequences, namely, the motif

- Additional information:
  - The hidden sequence is of length 8
  - The pattern is not exactly the same in each sequence because random point mutations may occur in the sequences
The Motif Finding Problem (cont’d)

The patterns revealed with no mutations:

cctgatagacgctatctggctatccagtacgtaggtcctctgtgcgaatctatgcgttttccaaccat
agtactggtgtacatttgatacgtaacctgaacaaacgctcagaaccagaagtgc
aadacgtacgtgcacccctctttctttcgtggctcttgccaaacgagggctgaatgtataagacga
agctccgatgtaagtcatagctgtaactattacctgccacccctattacatcttacgtacgtacgtataca
ctgttatacaacgcgtcatggccggtatgcgttttgcgtctgtaacgctcgtacgtcgttaacgtacgt

⇒ Consensus String: acgtacgt
The Motif Finding Problem (cont’d)

The patterns revealed with 2 mutations:

cctgatagacgctatctggctatccaggtcctctggtcgtatctatgcgttttccaaccat
agttacgtttgatacccggaacctgaacaaacgctcagaaccagaaggtaagcgtatattaacgcgtcatggcggggttttttggtctggtacgctcgtacgtacgcgctcgtacgG
Can we still find the motifs now that we have 2 mutations?
The Motif Finding Problem (cont’d)

The patterns revealed with 2 mutations:

cctgatagacgctatctggctatcc\textcolor{blue}{aGgtacTt}aggtcctctgtgcgaatctatgcgttccccacccat
agtactgggtgtaaattttagtt\textcolor{blue}{CcAtacgt}acaccggcaacctgaacaagaacgcctcagaaccagaag\textcolor{blue}{tg}
\textcolor{blue}{aaacgtTAg\textcolor{blue}{t}}gcaccctctttcttcgtgctctgcgaaccgagggcctgatgtatataagacgaaaaat\textcolor{blue}{tttt}
agcctccgatgtaaagtcatagctgtaacta\textcolor{blue}{ct}actacttgtccacccctattatcatcttt\textcolor{blue}{acgtCcAtataca}
ctgttatataacgcgtcatggcgggtatgcgtttttgtcgtcgtacgcgtcgtcgtta\textcolor{blue}{CcgtacgGc}

Can we still find the motifs now that we have 2 mutations?
To define a motif, let’s say we know where the motif starts in each sequence.

The motif start positions in their sequences are $s = (s_1, s_2, s_3, \ldots, s_t)$.
Motifs: Profile and Consensus

<table>
<thead>
<tr>
<th>Alignment</th>
<th>a G g t a c T t</th>
<th>C c A t a c g t</th>
<th>a c g t T A g t</th>
<th>a c g t C c A t</th>
<th>C c g t a c g G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile</td>
<td>A 3 0 1 0 3 1 1 0</td>
<td>C 2 4 0 0 1 4 0 0</td>
<td>G 0 1 4 0 0 0 3 1</td>
<td>T 0 0 0 5 1 0 1 4</td>
<td></td>
</tr>
<tr>
<td>Consensus</td>
<td>A C G T A C G T</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Line up the patterns by their start indexes
  \[ s = (s_1, s_2, s_3, \ldots, s_t) \]
- Construct matrix profile with frequencies of each nucleotide in columns
- Consensus nucleotide in each position has the highest score in column
Some Term Definitions

- $t$: number of sample DNA sequences
- $n$: length of each DNA sequence
- $\text{DNA}$: sample of DNA sequences ($t \times n$ array)
  - $\text{DNA}[i]$ denotes the $i$:th DNA sequence
  - $\text{DNA}[i][j \ldots j']$ denotes substring $T[j \ldots j']$ where $T = \text{DNA}[i]$
- $\ell$: length of the motif ($\ell$-mer)
- $s_i$: starting position of an $\ell$-mer in sequence $i$
- $s = (s_1, s_2, \ldots, s_t)$: array of motif’s starting positions
Parameters

\[ \ell = 8 \]

DNA

\[
\begin{align*}
&\text{cctgatagacgctatctggctatcc}_aGgta\text{cTtaggtcctctggtgcgaatctatgcgttttccaaccat} \\
&\text{agtactggtgtacatttgatCcAtacgtaacccggaacctgaaacaaacgctcagaaccagaagtgc} \\
&\text{aaca}_c\text{gtTAg}tcaccctctttcttctctggctctggccaaccgagggctgtatataagacgaaatttt} \\
&\text{agcctccgatgtcataagctgctaactattacctgccacccttattacatctttacgTcATataca} \\
&\text{ctgttataacacgcgtcatggggtatgctttttggtcgtcgtacgcctcgatcgttacGc}
\end{align*}
\]

\[ t = 5 \]

\[ n = 69 \]

\[ s = \{ s_1 = 26, s_2 = 21, s_3 = 3, s_4 = 56, s_5 = 60 \} \]
Scoring Motifs

Given \( s = (s_1, \ldots, s_t) \) and DNA:

\[
\text{Score}(s, \text{DNA}) = \sum_{j=1}^{\ell} \max_{c \in \{A, C, G, T\}} \text{count}(c, j)
\]

where \( \text{count}(c, j) \) gives the number of times symbol \( c \) equals \( \text{DNA}[i][s_i + j - 1] \), that is,

\[
\text{count}(c, j) = |\{i|\text{DNA}[i][s_i + j - 1] = c, i \in [1, t]\}| .
\]

Consensus

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>

Score \( 3+4+4+5+3+4+3+4 = 30 \)
The Motif Finding Problem

- If starting positions \( s = (s_1, s_2, \ldots, s_t) \) are given, finding consensus is easy even with mutations in the sequences because we can simply construct the profile to find the motif (consensus)
- But... the starting positions \( s \) are usually not given. How can we find the “best” profile matrix?
The Motif Finding Problem: Formulation

- **Goal**: Given a set of DNA sequences, find a set of $\ell$-mers, one from each sequence, that maximizes the consensus score.
- **Input**: A $t \times n$ matrix $DNA$, and $\ell$, the length of the pattern to find
- **Output**: An array of $t$ starting positions $s = (s_1, s_2, \ldots, s_t)$ maximizing $Score(s, DNA)$
The Motif Finding Problem: Brute Force Solution

- Compute the scores for each possible combination of starting positions \(s\)
- The best score will determine the best profile and the consensus pattern in \(DNA\)
- The goal is to maximize \(Score(s, DNA)\) by varying the starting positions \(s_i\), where

\[
\begin{align*}
s_i &\in [1, \ldots, n - \ell + 1] \\
i &\in [1, \ldots, t]
\end{align*}
\]
BruteForceMotifSearch

BruteForceMotifSearch\( (DNA, t, n, \ell) \)

1. \( \text{bestScore} \leftarrow 0 \)
2. \textbf{for} each \( s = (s_1, s_2, \ldots, s_t) \) from \( (1, 1, \ldots, 1) \) to \( (n - \ell + 1, \ldots, n - \ell + 1) \) \textbf{do}
3. \hspace{1em} \textbf{if} \( \text{Score}(s, DNA) > \text{bestScore} \) \textbf{then}
4. \hspace{2em} \text{bestScore} \leftarrow \text{score}(s, DNA)
5. \hspace{1em} \text{bestMotif} \leftarrow (s_1, s_2, \ldots, s_t)
6. \textbf{return} \text{ bestMotif }
Running Time of BruteForceMotifSearch

- Varying \((n - \ell + 1)\) position in each of \(t\) sequences, we get \((n - \ell + 1)^t\) sets of starting positions.
- For each set of starting positions, computing the scoring function requires \(\ell t\) operations, so complexity is

\[
\ell t(n - \ell + 1)^t = O(\ell t n^t)
\]

- E.g. for \(t = 8\), \(n = 1000\), \(\ell = 10\) we must perform approximately \(10^{20}\) computations – it will take billions of years.
The Median String Problem

- Given a set of $t$ DNA sequences find a pattern that appears in all $t$ sequences with the minimum number of mutations
- This pattern will be the motif
The Hamming distance $d_H(v, w)$ is the number of nucleotide pairs that do not match when $v$ and $w$ are aligned.

For example:

$$d_H(\text{AAAAAA}, \text{ACAAAC}) = 2$$
Given $v = \text{"acgtacgt"}$ and $s$:

\[
\begin{align*}
\text{acgtacgt} & \rightarrow d_H(v, x) = 1 \\
\text{cctgatagacgtatctggtatccacgtacAtaggtcctctgtgcgaatctatgctgtttccacccat} & \rightarrow d_H(v, x) = 0 \\
\text{acgtacgt} & \rightarrow d_H(v, x) = 2 \\
\text{agtactgggtgatcatttggatacgtacgtacaccggcaacctgaaacaacgcgtcagacagaacgatgc} & \rightarrow d_H(c, x) = 0 \leftarrow \text{acgtacgt} \\
\text{aaaAgtCcggtgcacccctttttctggtgctctggccacacgagggctgtgatgtataagacgaaaaatttt} & \rightarrow d_H(c, x) = 1 \leftarrow \text{acgtacgt} \\
\text{agccctccgatgtaagtcatagctgtaactattacctgccacccctattacatcttacgtacgtatataca} & \texttt{ctgttatacaacgcgtcatggtgccgggtatgcgttggggtacgtgtcgagctcgatcgttacgtacgtacgtaGgtc}\end{align*}
\]

\[
\text{TotalDistance}(v, DNA) = 1 + 0 + 2 + 0 + 1 = 4
\]
Total Distance: Definition

- For each DNA sequence \( i \), compute all \( d_H(v, x) \), where \( x \) is an \( \ell \)-mer with starting position \( s_i \) (\( 1 \leq s_i \leq n - \ell + 1 \))
- Find minimum of \( d_H(v, x) \) among all \( \ell \)-mers in sequence \( i \)
- \( \text{TotalDistance}(v, DNA) \) is the sum of the minimum Hamming distances for each DNA sequence \( i \)

\[
\text{TotalDistance}(v, DNA) = \sum_{i \in [1,t]} \min_{s_i \in [1,m-\ell+1]} d_H(v, DNA[i][s_i, \ldots, s_i + \ell - 1])
\]
The Median String Problem: Formulation

- **Goal**: Given a set of DNA sequences, find a median string
- **Input**: a $t \times n$ matrix $DNA$, and $\ell$, the length of the pattern to find
- **Output**: A string $v$ of $\ell$ nucleotides that minimizes $\text{TotalDistance}(v, DNA)$ over all strings of that length
Median String Search Algorithm

MedianStringSearch(\(DNA, t, n, \ell\))

1. \(bestWord \leftarrow AAA\ldots A\)
2. \(bestDistance \leftarrow \infty\)
3. for each \(\ell\)-mer \(v\) from AAA\ldots A to TTT\ldots T do
4. \(\text{if } \text{TotalDistance}(v, DNA) < bestDistance\) then
5. \(bestDistance \leftarrow \text{TotalDistance}(v, DNA)\)
6. \(bestWord \leftarrow v\)
7. return \(bestWord\)
Motif Finding Problem = Median String Problem

- *Motif Finding* is a maximization problem while *Median String* is a minimization problem.
- However, *Motif Finding* and *Median String* problems are computationally equivalent.
- We need to show that minimizing *TotalDistance* is equivalent to maximizing *Score*.
We are looking for the same thing

\[ \ell \]

\[
\begin{array}{ccccccccc}
\text{Alignment} & a & G & g & t & a & c & T & t \\
& C & c & A & t & a & c & g & t \\
& a & c & g & t & T & A & g & t \\
& a & c & g & t & C & c & A & t \\
& C & c & g & t & a & c & g & G \\
\end{array}
\]

\[ t \]

\[ \begin{array}{ccccccccc}
\text{A} & 3 & 0 & 1 & 0 & 3 & 1 & 1 & 0 \\
\text{C} & 2 & 4 & 0 & 0 & 1 & 4 & 0 & 0 \\
\text{G} & 0 & 1 & 4 & 0 & 0 & 0 & 3 & 1 \\
\text{T} & 0 & 0 & 0 & 5 & 1 & 0 & 1 & 4 \\
\end{array}
\]

\[ \text{Consensus} \quad \text{A} \quad \text{C} \quad \text{G} \quad \text{T} \quad \text{A} \quad \text{C} \quad \text{G} \quad \text{T} \\
\text{Score} \quad 3+4+4+5+3+4+3+4 \\
\text{TotalDistance} \quad 2+1+1+0+2+1+2+1 \\
\text{Sum} \quad 5 \quad 5 \quad 5 \quad 5 \quad 5 \quad 5 \quad 5 \quad 5 \quad 5
\]

\[ \begin{align*}
\text{At any column } i \\
\text{Score}_i + \text{TotalDistance}_i &= t \\

\ell \text{ columns} \implies \\
\text{Score} + \text{TotalDistance} &= \ell \cdot t \\

\text{Rearranging:} \\
\text{Score} &= \ell \cdot t - \text{TotalDistance}
\end{align*} \]

\[ \ell \cdot t \text{ is constant so the minimization of the right side is equivalent to the maximization of the left side} \]
Motif Finding Problem vs Median String Problem

Why bother reformulating the Motif Finding problem into the Median String problem?

- The Motif Finding problem needs to examine all the combinations for $s$
  \[\Rightarrow \text{Total running time } O(\ell tn^t)\]
- The Median String problem needs to examine all $4^\ell$ combinations for $v$
  \[\Rightarrow \text{Total running time } O(\ell tn4^\ell)\]
Outline

Biological motivation

Implanted motifs - an introduction

Motif Finding Problem and Median String Problem

Structuring the search: search tree

Branch and bound

Weeder

Study group assignments
Recall the BruteForceMotifSearch:

BruteForceMotifSearch(DNA, t, n, ℓ)

1: $bestScore \leftarrow 0$
2: for each $s = (s_1, s_2, \ldots, s_t)$ from $(1, 1, \ldots 1)$ to $(n - ℓ + 1, \ldots n - ℓ + 1)$ do
3: \hspace{1em} if $Score(s, DNA) > bestScore$ then
4: \hspace{2em} $bestScore \leftarrow score(s, DNA)$
5: \hspace{2em} $bestMotif \leftarrow (s_1, s_2, \ldots, s_t)$
6: return $bestMotif$
Structuring the Search

- How can we perform the line

  \[
  \text{for each } s = (s_1, s_2, \ldots, s_t) \text{ from } (1, 1, \ldots 1) \text{ to } (n - \ell + 1, \ldots n - \ell + 1) \text{ do}
  \]

- We need a method for efficiently structuring and navigating the many possible motifs

- This is not very different from exploring all \( t \)-digit numbers
Recall the MedianStringSearch:

\[ \text{MedianStringSearch}(\text{DNA}, t, n, \ell) \]

1. \( \text{bestWord} \leftarrow \text{AAA...A} \)
2. \( \text{bestDistance} \leftarrow \infty \)
3. \textbf{for each} \( \ell \)-mer \( v \) \textbf{from AAA...A to TTT...T} \textbf{do}
4. \textbf{if} \ \text{TotalDistance}(v, \text{DNA}) < \text{bestDistance} \textbf{then}
5. \( \text{bestDistance} \leftarrow \text{TotalDistance}(v, \text{DNA}) \)
6. \( \text{bestWord} \leftarrow v \)
7. \textbf{return} \ \text{bestWord}
Structuring the Search

- How can we perform the line
  \[
  \text{for each } \ell\text{-mer } v \text{ from AAA...A to TTT...T do}
  \]

- We need a method for efficiently structuring and navigating all \(4^\ell\) possible \(\ell\)-mers

- This is not very different from exploring all \(\ell\)-digit numbers (in base 4):

  \[
  \begin{align*}
  \{ &\text{AA...AA} \\
  &\text{AA...AC} \\
  &\text{AA...AG} \\
  &\text{AA...AT} \\
  &\text{TT...TT} \\
  \}
  \]

\[
\begin{array}{c}
\ell \\
\hline
\text{AA...AA} \\
\text{AA...AC} \\
\text{AA...AG} \\
\text{AA...AT} \\
\text{TT...TT}
\end{array}
\]
Alternative Representation of the Search Space

- Let $A = 1$, $C = 2$, $G = 3$, $T = 4$
- Then the sequences from $AA\ldots A$ to $TT\ldots T$ become:

$$\ell \begin{cases} 11\ldots11 \\ 11\ldots12 \\ 11\ldots13 \\ 11\ldots14 \\ \vdots \\ 44\ldots44 \end{cases}$$

- Notice that the sequences above simply list all $\ell$-digit numbers in base 4 using digits 1, 2, 3, and 4
Search Tree

```
A
AA AC AG AT

C
CA CC CG CT

G
GA GC GG GT

T
TA TC TG TT
```

Analyzing Search Trees

- Characteristics of search trees
  - The sequences are contained in its leaves
  - The parent of a node is the prefix of its children
- How can we move through the tree?
Analyzing Search Trees

- Characteristics of search trees
  - The sequences are contained in its leaves
  - The parent of a node is the prefix of its children

- How can we move through the tree?

- Four common moves in a search tree that we are about to explore:
  - Move to next leaf
  - Visit all the leaves
  - Visit next node
  - Bypass the children of a node
Visit the Next Leaf

Given a current leaf \( a \), we need to compute the next leaf:

\[
\text{NextLeaf}(a, L, k)
\]

1. \textbf{for} \( i \leftarrow L \) \textbf{to} 1 \textbf{do} \\
2. \textbf{if} \( a_i < k \) \textbf{then} \\
3. \quad \quad \quad \quad \quad \quad \quad a_i \leftarrow a_i + 1 \\
4. \quad \quad \quad \textbf{return} \quad a \\
5. \quad \quad a_i \leftarrow 1 \\
6. \quad \textbf{return} \quad a

\begin{itemize}
\item \textbf{Arguments:}
  \begin{itemize}
  \item \( a \): the array of digits
  \item \( L \): length of the array
  \item \( k \): max digit value
  \end{itemize}
\item The algorithm is common addition in radix \( k \):
  \begin{itemize}
  \item Increment the least significant digit
  \item “Carry the one” to the next digit position when the digit is at maximal value
  \end{itemize}
\end{itemize}
NextLeaf: Example
NextLeaf: Example
Visit All Leaves

Print all permutations in ascending order:

```
AllLeaves(L, k)
1: a ← (1, ..., 1)
2: while forever do
3: output a
4: a ← NextLeaf(a, L, k)
5: if a = (1, 1, ..., 1) then
6: return
```

- **L**: length of the sequence
- **k**: max digit value
- **a**: array of digits
Visit All Leaves: Example

Moving through all the leaves in order:
Depth First Search

- The previous slides showed how to search leaves
- How about searching all vertices of the tree?
- We can do this with a depth first search
Visit the Next Vertex

Given a current vertex \( a \), we need to compute the next vertex:

\[ \text{NextVertex}(a, i, L, k) \]

1. if \( i < L \) then
   2. \( a_{i+1} \leftarrow 1 \)
   3. return \((a, i + 1)\)
2. else
   4. for \( j \leftarrow L \) to 1 do
      5. if \( a_j < k \) then
         6. \( a_j \leftarrow a_j + 1 \)
         7. return \((a, j)\)
   8. return \((a, 0)\)

- Arguments:
  - \( a \): the array of digits
  - \( i \): prefix length
  - \( L \): length of the array
  - \( k \): max digit value

- The algorithm returns a prefix which is represented by digits in the array \( a \) and the prefix length
NextVertex: Example

Moving to the next vertices:

Current vertex
NextVertex: Example

Moving to the next vertices:

Location after 5 next vertex moves
Bypass Move

Given a prefix (internal vertex), find next vertex after skipping all its children

Bypass\((a, i, L, k)\):

1: \textbf{for} \(j \leftarrow i\) \textbf{to} 1 \textbf{do}
2: \hspace{1em} \textbf{if} \(a_j < k\) \textbf{then}
3: \hspace{2em} \(a_j \leftarrow a_j + 1\)
4: \hspace{1em} \textbf{return} \((a, j)\)
5: \hspace{1em} \textbf{return} \((a, 0)\)

- \(a\): array of digits
- \(i\): prefix length
- \(L\): maximum length
- \(k\): max digit value
Bypass Move: Example

Bypassing descendants of “2-”

Current vertex
Bypass Move: Example

Bypassing descendants of “2-”

Next vertex
Outline

Biological motivation

Implanted motifs - an introduction

Motif Finding Problem and Median String Problem

Structuring the search: search tree

Branch and bound

Weeder

Study group assignments
Now that we have a method for navigating the tree, let’s look again at BruteForceMotifSearch.

BruteForceMotifSearchAgain(DNA, t, n, ℓ)

1: \( s \leftarrow (1, 1, \ldots, 1) \)
2: \( \text{bestScore} \leftarrow \text{score}(s, DNA) \)
3: \( \textbf{while} \ \text{forever} \ \textbf{do} \)
4: \( s \leftarrow \text{NextLeaf}(s, t, n - ℓ + 1) \)
5: \( \text{if Score}(s, DNA) > \text{bestScore} \ \text{then} \)
6: \( \text{bestScore} \leftarrow \text{score}(s, DNA) \)
7: \( \text{bestMotif} \leftarrow (s_1, s_2, \ldots, s_t) \)
8: \( \text{if } s = (1, 1, \ldots, 1) \ \text{then} \)
9: \( \textbf{return } \text{bestMotif} \)
Can We Do Better?

- Sets of $s = (s_1, s_2, \ldots, s_t)$ may have a weak profile for the first $i$ positions $(s_1, s_2, \ldots, s_i)$
- Every row of the alignment may add at most $\ell$ to $\text{Score}$
- Define $\text{Score}(s, i, DNA)$ to be the score involving only the $i$ first rows of the alignment matrix

\[
\text{Score}(s, i, DNA) = \sum_{j=1}^{\ell} \max_{c \in \{A, C, G, T\}} \text{count}_i(c, j)
\]

where $\text{count}_i(c, j)$ gives the number of times nucleotide $c$ occurs in column $j$ in the first $i$ rows of the alignment matrix

- Optimism: if all remaining $(t - i)$ positions $(s_{i+1}, \ldots, s_t)$ add $(t - i) \cdot \ell$ to $\text{Score}(s, i, DNA)$
- If $\text{Score}(s, i, DNA) + (t - i) \cdot \ell < \text{BestScore}$, it makes no sense to search in vertices of the current subtree $\implies$ Use ByPass()!
Branch and Bound Algorithm for Motif Search

- Since each level of the tree goes deeper into the search, discarding a prefix discards all following branches.
- This saves us from looking at $(n - \ell + 1)^{t-i}$ leaves.
  - Use NextVertex() and ByPass() to navigate the tree.
Pseudocode for Branch and Bound Motif Search

BranchAndBoundMotifSearch($DNA, t, n, \ell$)

1: $s \leftarrow (1, 1, \ldots, 1)$
2: $bestScore \leftarrow 0$
3: $i \leftarrow 1$
4: while $i > 0$ do
5:   if $i < t$ then
6:     optimisticScore $\leftarrow \text{Score}(s, i, DNA) + (t - i) \cdot \ell$
7:     if optimisticScore $< bestScore$ then
8:       $(s, i) \leftarrow \text{Bypass}(s, i, n - \ell + 1)$
9:     else
10:       $(s, i) \leftarrow \text{NextVertex}(s, i, t, n - \ell + 1)$
11:   else
12:     if $\text{Score}(s, DNA) > bestScore$ then
13:       $bestScore \leftarrow \text{score}(s, DNA)$
14:       $bestMotif \leftarrow (s_1, s_2, \ldots, s_t)$
15:       $(s, i) \leftarrow \text{NextVertex}(s, i, t, n - \ell + 1)$
16: return $bestMotif$
Median String Search Improvements

- Recall the computational differences between motif search and median string search
  - The Motif Finding Problem needs to examine all \((n - \ell + 1)^t\) combinations for \(s\)
  - The Median String Problem needs to examine \(4^\ell\) combinations of \(v\). This number is relatively small
- We want to use median string algorithm with the Branch and Bound trick!
Branch and Bound Algorithm for Median String Search

- Note that if the total distance for a prefix is greater than that for the best word so far:

  \[ \text{TotalDistance}(\text{prefix}, \text{DNA}) > \text{BestDistance} \]

  there is no use exploring the remaining part of the word

- We can eliminate that branch and ByPass exploring it further
Bounded Median String Search

BranchAndBoundMedianStringSearch(DNA, t, n, ℓ)

1: \( v \leftarrow (1, \ldots, 1) \) \# recall that 1=A, 2=C, 3=G, 4=T
2: \( \text{bestWord} \leftarrow \text{AAA} \ldots \text{A} \)
3: \( \text{bestDistance} \leftarrow \infty \)
4: \( i = 1 \)
5: while \( i > 0 \) do
6: \quad if \( i < ℓ \) then
7: \quad \quad \text{prefix} \leftarrow \text{string corresponding to the first } i \text{ nucleotides of } v
8: \quad \quad \text{optimisticDistance} \leftarrow \text{TotalDistance(prefix, DNA)}
9: \quad \quad if \ \text{optimisticDistance} > \text{bestDistance} \text{ then}
10: \quad \quad \quad (v, i) \leftarrow \text{ByPass}(v, i, ℓ, 4)
11: \quad \quad else
12: \quad \quad \quad (v, i) \leftarrow \text{NextVertex}(v, i, ℓ, 4)
13: \quad \quad else
14: \quad \quad \quad \text{word} \leftarrow \text{nucleotide string corresponding to } v
15: \quad \quad \quad if \ \text{TotalDistance(word, DNA)} < \text{bestDistance} \text{ then}
16: \quad \quad \quad \quad \text{bestDistance} \leftarrow \text{TotalDistance(word, DNA)}
17: \quad \quad \quad \quad \text{bestWord} \leftarrow \text{word}
18: \quad \quad \quad (v, i) \leftarrow \text{NextVertex}(v, i, ℓ, 4)
19: return bestWord
Some Motif Finding Programs

- CONSENSUS (*Hertz, Stromo (1989)*)
- GibbsDNA (*Lawrence et al (1993)*)
- MEME (*Bailey, Elkan (1995)*)
- Weeder (*Pavesi, Mauri, Pesole (2001)*)
- RandomProjections (*Buhler, Tompa (2002)*)
- MULTIPROFILER (*Keich, Pevzner (2002)*)
- MITRA (*Eskin, Pevzner (2002)*)
- Pattern Branching (*Price, Pevzner (2003)*)
Outline

Biological motivation

Implanted motifs - an introduction

Motif Finding Problem and Median String Problem

Structuring the search: search tree

Branch and bound

Weeder

Study group assignments
Weeder: Suffix Tree -Based Approach

- Suffix tree is a compressed keyword trie of all suffixes of a sequence.
- E.g. suffixes of sequence CATACT are CATACT, ATACT, TACT, ACT, CT, T.
- Suffix tree of CATACT:
Suffix Tree

→ 58093 String Processing Algorithms (4 cr), period II
Exact Search on Suffix Tree

pattern = C
Backtracking on Suffix Tree

pattern = ACA, 1 mismatch
Suffix Tree and Exact Motif Finding

- Concatenate the rows of the $t \times n$ matrix $DNA$ into a string $S$, inserting an endmarker $\#$ between the rows.
- Build suffix tree $T$ of $S$ and for each node $v$ store a bitvector $B(v)[1, t]$ such that $B(v)[i] = 1$ iff any suffix starting from row $t$ is in the subtree of $v$.
- Let $v(s)$ denote an internal node of $T$ such that the path from the root to the incoming edge of $v(s)$ spells $s$.
- String $s$ is an exact motif if $B(v(s))[i] = 1$ for all $i$. 
Suffix Tree and Exact Motif Finding

\[ B(v) = 111 \]

\[ \ldots ACACG \ldots # \ldots ACACG \ldots ACACG \ldots # \ldots ACACG \ldots # \]
Weeder: a Suffix Tree-Based Approach

- Weeder extends the exact motif finding algorithm to approximate motifs
- Backtracking plugged in
- Some heuristics to avoid too extensive branching
Outline

Biological motivation

Implanted motifs - an introduction

Motif Finding Problem and Median String Problem

Structuring the search: search tree

Branch and bound

Weeder

Study group assignments
Study Group Assignments: Group 1 (Lastnames A-K)

- Read Sections 4.1-4.3 (partial digest problem) from the course book before coming to the study group meeting.
- Solve Problem 4.2 (page 119) at study group and use the solution to explain the material to the other groups:
  - Consider partial digest
    \[ L = \{1, 1, 1, 2, 2, 3, 3, 3, 4, 4, 5, 5, 6, 6, 6, 9, 9, 10, 11, 12, 15\} \]
  - Solve the Partial Digest problem for \( L \) (i.e. find \( X \) such that \( \Delta X = L \)).
Study Group Assignments: Group 2 (Lastnames L-San)

▶ Read the following article before coming to the study group:

http://genome.cshlp.org/content/8/11/1202.full

Read especially section METHODS.

▶ At study group, discuss the approach, and draw a pattern trie for some small example input.
Study Group Assignments: Group 3 (Lastnames See-Y)

- Read the following article before coming to the study group:
  http://bioinformatics.oxfordjournals.org/content/17/suppl_1/S207.full.pdf

- At study group, summarize the message of the article and share the message to other groups.