

582670 Algorithms for Bioinformatics

Lecture 2: Exhaustive search and motif finding

6.9.2012

Outline

Implanted motifs - an introduction

Biological motivation

Motif Finding Problem and Median String Problem

Structuring the search: search tree

Branch and bound

Weeder

Study group assignments

Random Sample

atgaccgggatactgataccgtatattggcctagggctacacattagataaacgtatgaagtacgttagactcggcgccgcc
accctatTTTTTgagcagatttagtgacctggaaaaaaatttgagtacaaaacttttccgaatactgggcataaggtac
tgagtatccctgggatgacttttgggaacactatagtgtctctccgatttttgaatatgtaggatcattcgccagggtccg
gctgagaattggatgaccttgtaagtgttttccacgcaatcggaaccaacgcggacccaaaggcaagaccgataaaggag
tcccttttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaatggcccacttagtccacttata
gtcaatcatgttcttgtgaatggatttttaactgagggcatagaccgcttggcgcacccaaattcagtggtggcgagcgca
cggttttggcccttgtagaggccccgtactgatggaaactttcaattatgagagagctaattctatcgctgctgtttca
aacttgagttggtttcgaaaatgctctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgt
ttggcccatgggctaaaagcccaacttgacaaatggaagatagaatccttgcatTTTcaacgtatgccgaaccgaaagggaa
ctgggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagcttctgggtactgatagc

Implanting Motif AAAAAAAAGGGGGGGG

atgaccgggatactgatAAAAAAAGGGGGGGGggcggtacacattagataaacgtatgaagtacgttagactcggcgccgcc
accctatTTTTTgagcagatttagtgacctggaaaaaaatttgagtacaaaacttttccgaataAAAAAAAGGGGGGGG
tgagtatccctgggatgacttAAAAAAAGGGGGGGGtgctctccgatttttgaatatgtaggatcattcgccagggtccg
gctgagaattggatgAAAAAAAGGGGGGGGtccacgcaatcggaaccaacgcggacccaaaggcaagaccgataaaggag
tccttttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaatAAAAAAAGGGGGGGGcttata
gtcaatcatgttcttgtgaatggatttAAAAAAAGGGGGGGGgaccgcttggcgcacccaaattcagtggtggcgagcgca
cggttttggccttgttagaggccccgtAAAAAAAGGGGGGGGcaattatgagagagctaattctatcgctgctgttca
aacttgagttAAAAAAAGGGGGGGGctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgt
ttggccattggctaaaagcccaacttgacaaatggaagatagaatccttgcatAAAAAAAGGGGGGGGaccgaaagggaa
ctgggtgagcaacgacagattcttacgtgcattagctcgcttcggggatctaatagcacgaagcttAAAAAAAGGGGGGGG

Where is the implanted motif?

atgaccgggatactgataaaaaaagggggggggcgctacacattagataaacgtatgaagtacgttagactcggcgccgcc
acccctatTTTTTgagcagatttagtgacctggaaaaaaatttgagtacaaaacttttccgaataaaaaaaaagggggggg
tgagtatccctgggatgacttaaaaaaaggggggggtgctctcccgatTTTTgaatatgtaggatcattcgccagggtccg
gctgagaattggatgaaaaaaaggggggggtccacgcaatcgcgaccaacgcggacccaaaggcaagaccgataaaggag
tccTTTTgCGGtaatgtgccgggaggctggttacgtagggaagccctaacggacttaataaaaaaaggggggggcttata
gtcaatcatgttcttTgaatggatttaaaaaaaggggggggaccgcttggcgcacccaaattcagtgtgggcgagcgca
cggtTTTTggcccttgtagaggccccgtaaaaaaaggggggggcaattatgagagagctaattctatcgctgctgtttca
aacttgagttaaaaaaaggggggggctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgt
ttggcccatTggctaaaagcccaacttgacaaatggaagatagaatccttgcataaaaaaaggggggggaccgaaagggaa
ctggTgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagcttaaaaaaagggggggg

Implanting Motif AAAAAAAGGGGGGG with four mutations

atgaccgggatactgatAgAAgAAAGGttGGGggcggtacacattagataaacgtatgaagtacgttagactcggcgccgcc
accctatTTTTTgagcagatttagtgacctggaaaaaaatttgagtacaaaactTTTTccgaatacAAtAAAAcGGcGGG
tgagtatccctgggatgacttAAAAtAAtGGaGtGGtgctctccgattttgaatatgtaggatcattcgccagggtccg
gctgagaattggatgcAAAAAAGGGattGttccacgcaatcggaaccaacgcggacccaaaggcaagaccgataaaggag
tcccttttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaataAAtAAAGGaAGGGcttata
gtcaatcatgttcttgtgaatggattAACAAtAAGGGctGGgaccgcttggcgcacccaaattcagtgtgggcgagcgca
cggttttggcccttgtagaggccccgtAATAAaAAGGaAGGGccaattatgagagagctaattctatcgctgcgtgttca
aacttgagttAAAAAAtAGGGaGccctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgt
ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcatActAAAAAGGaGcGGaccgaaaggga
ctggtgagcaacgacagattcttacgtgcattagctcgcttcgggggatctaatagcacgaagcttActAAAAAGGaGcGG

Where is the implanted motif???

atgaccgggatactgatagaagaaaggttggggggcgctacacattagataaacgtatgaagtacgttagactcggcgccgcc
acccctatTTTTTgagcagatttagtgacctggaaaaaaatttgagtacaaaacttttccgaatacaataaaaacggcggg
tgagtatccctgggatgacttaaaataatggagtggtgctctcccgatTTTTgaatatgtaggatcattcgccagggtccg
gctgagaattggatgcaaaaaagggattgtccacgcaatcgcgaaaccaacgcggacccaaaggcaagaccgataaaggag
tccctTTTgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaataataaaaggaagggttata
gtcaatcatgttcttTgaatggatttaacaataagggtgaggaccgcttggcgcacccaaattcagtgTgggcgagcgca
cggtTTTggcccttgtagaggccccgtataaacaaggaggggccaattatgagagagctaattctatcgctgTgctgttca
aacttgagTtaaaaaataggagccctgggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgt
ttggcccatTggctaaaagcccaacttgacaaatggaagatagaatccttgcatactaaaaaggagcggaccgaaagggaa
ctggtgagcaacgacagattcttacgtgcattagctcgcttcgggggatctaatagcacgaagcttactaaaaaggagcgg

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

atgaccgggatactgatAgAAgAAAGGttGGGggcggtacacattagataaacgtatgaagtacgttagactcggcgccgcc
accctatTTTTTgagcagatttagtgacctggaaaaaaatttgagtacaaaacttttccgaatacAAtAAAAcGGcGGG
tgagtatccctgggatgacttAAAAtAAtGGaGtGGtgctctccgatttttgaatatgtaggatcattcgccagggtccg
gctgagaattggatgcAAAAAAGGGattGttccacgcaatcggaaccaacgcggacccaaaggcaagaccgataaaggag
tcccttttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaataAtAAtAAAGGaAGGccttata
gtcaatcatgttcttgtgaatggatttAAcAAtAAGGGctGGgaccgcttggcgcacccaaattcagtgtgggcgagcgca
cggTTTTggcccttgtagaggccccgtAtAAAcAAGGaAGGcCaattatgagagagctaattctatcgctgcgtgttca
aacttgagttAAAAAAtAGGGaGccctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgt
ttggcccatTggctaaaagcccaacttgacaaatggaagatagaatccttgcatActAAAAAGGaGcGGaccgaaagggaa
ctggtgagcaacgacagattcttacgtgcattagctcgcttcgggggatctaatagcacgaagcttActAAAAAGGaGcGG

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

Implanted motifs - an introduction

Biological motivation

Motif Finding Problem and Median String Problem

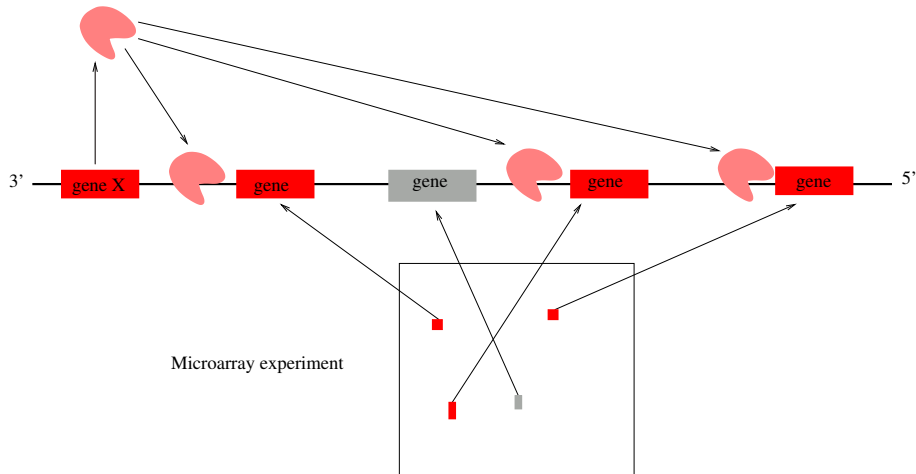
Structuring the search: search tree

Branch and bound

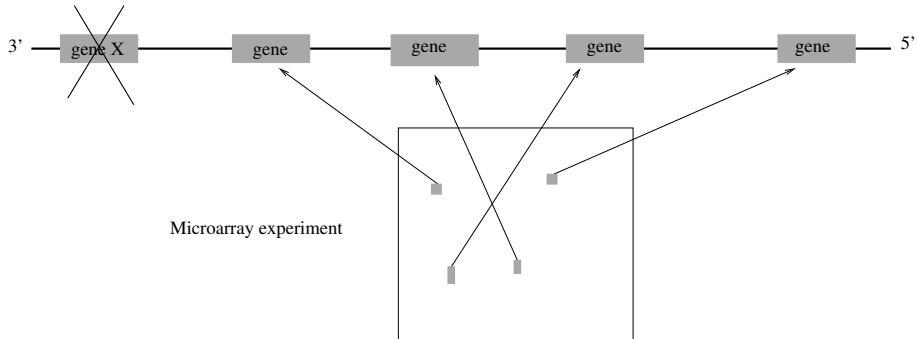
Weeder

Study group assignments

Biological Motivation



Biological Motivation (cont'd)



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?

Regulatory Proteins

- ▶ 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

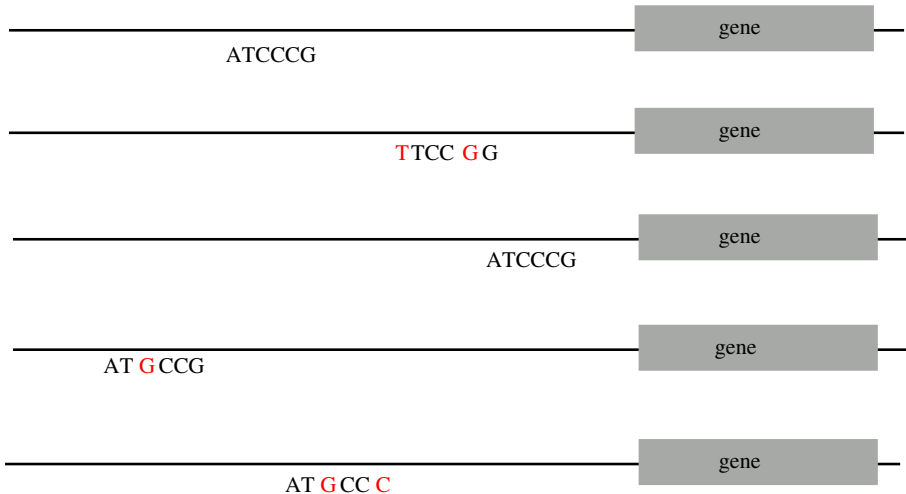
Regulatory Regions

- ▶ 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



Motif Logo

- ▶ 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

TGGGGGA
TGAGAGA
TGGGGGA
TGAGAGA
TGAGGGA



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

Implanted motifs - an introduction

Biological motivation

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:

```
cctgatagacgctatctggctatccacgtacgtaggtcctctgtgcgaatctatgcgtttccaaccat  
agtactgggtgtacatttgatacgtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc  
aaacgtacgtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaat  
agcctccgatgtaagtcatacgtgtaactattacctgccaccctattacatcttacgtacgtataca  
ctgttataacaacgcgtcatggcgggggtatgcgttttggtcgtcgtacgctcgatcgttaacgtacgtc
```

- ▶ 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:

```
cctgatagacgctatctggctatccacgtacgtaggtcctctgtgcgaatctatgcgtttccaacct  
agtactgggtgtacatttgatacgtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc  
aaacgtacgtgcaccctctttcttcgtggctctggccaacgagggtgatgtataagacgaaaatttt  
agcctccgatgtaagtcatacgtgtaactattacctgccaccctattacatcttacgtacgtataca  
ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgttaacgtacgtc
```

- ▶ 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:

```
cctgatagacgctatctggctatccacgtacgtaggtcctctgtgcgaatctatgcgtttccaaccat
agtactgggtgtacatttgatacgtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc
aaacgtacgtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatctt
agcctccgatgtaagtcatagctgtaactattacctgccaccctattacatcttacgtacgtataca
ctgttatacaacgcgtcatggcgggggtatgcgttttggctcgtcgtacgctcgatcgttaacgtacgtc
```

⇒ Consensus String: **acgtacgt**

The Motif Finding Problem (cont'd)

The patterns revealed with 2 mutations:

```
cctgatagacgctatctggctatccaGgtacTtaggtcctctgtgcaatctatgcgtttccaaccat
agtactggtgtacatttgatCcAtacgtacaccggcaacctgaacaaacgctcagaaccagaagtgc
aaacgtTAgtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt
agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatctttacgtCcAtatatac
ctgttatacaacgcgtcatggcgggggtatgcgttttggtcgtcgtacgctcgatcgttaCcgtacgGc
```

The Motif Finding Problem (cont'd)

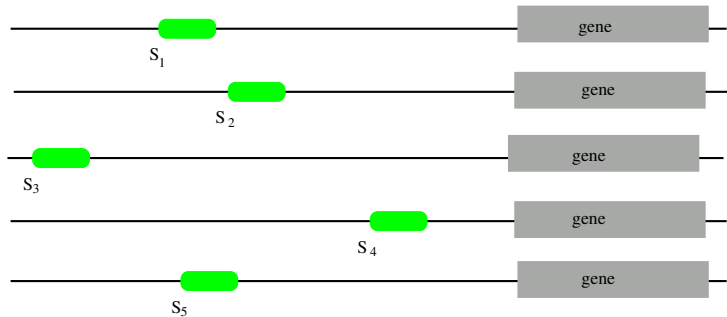
The patterns revealed with 2 mutations:

```
cctgatagacgctatctggctatccaGgtacTtaggtcctctgtgcgaatctatgcgtttccaaccat
agtactggtgtacatttgatCcAtacgtacaccggcaacctgaacaaacgtcagaaccagaagtgc
aaacgtTAgtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt
agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatcttacgtCcAtataca
ctgttatacaacgcgtcatggcgggggtatgcgttttggtcgtcgtacgctcgatcgttaCcgtacgGc
```

Can we still find the motifs now that we have 2 mutations?

Defining Motifs

- ▶ To define a motif, let's say we know where the motif starts in each sequence
- ▶ The motif start positions in their sequences are $\mathbf{s} = (s_1, s_2, s_3, \dots, s_t)$



Motifs: Profile and Consensus

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

- Line up the patterns by their start indexes

$$\mathbf{s} = (s_1, s_2, s_3, \dots, s_t)$$

Profile

A	3	0	1	0	3	1	1	0
C	2	4	0	0	1	4	0	0
G	0	1	4	0	0	0	3	1
T	0	0	0	5	1	0	1	4

- Construct matrix profile with frequencies of each nucleotide in columns
- Consensus nucleotide in each position has the highest score in column

Consensus A C G T A C G T

Some Term Definitions

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

Parameters

$$\ell = 8$$

DNA

$$t = 5 \left\{ \begin{array}{l} \text{cctgatagacgctatctggctatcca}\underline{\text{aGgtacTt}}\text{aggtcctctgtgcgaatctatgcgtttccaacat} \\ \text{agtactggtgtacatttgat}\underline{\text{CcAtacgt}}\text{acaccggcaacctgaaacaaacgctcagaaccagaagtgc} \\ \text{aa}\underline{\text{acgtTAgt}}\text{gcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt} \\ \text{agcctccgatgtaagtcatagctgtaactattacctgccaccctattacatctt}\underline{\text{acgtCcAt}}\text{ataca} \\ \text{ctgttataacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgтта}\underline{\text{CcgtagcGc}} \end{array} \right.$$

$n = 69$

$$\mathbf{s} = \{s_1 = 26, s_2 = 21, s_3 = 3, s_4 = 56, s_5 = 60\}$$

Scoring Motifs

- Given $\mathbf{s} = (s_1, \dots, s_t)$ and *DNA*:

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

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

$\text{count}(c, j) =$

$|\{i \mid \text{DNA}[i][s_i + j - 1] = c, i \in [1, t]\}|$. Consensus A C G T A C G T

Score

ℓ							
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
t							

A	3	0	1	0	3	1	1	0
C	2	4	0	0	1	4	0	0
G	0	1	4	0	0	0	3	1
T	0	0	0	5	1	0	1	4

3+4+4+5+3+4+3+4=30

The Motif Finding Problem

- ▶ If starting positions $\mathbf{s} = (s_1, s_2, \dots, s_t)$ 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 $\mathbf{s}$ 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 ℓ -mers, one from each sequence, that maximizes the consensus score.
- ▶ Input: A $t \times n$ matrix DNA , and ℓ , the length of the pattern to find
- ▶ Output: An array of t starting positions $\mathbf{s} = (s_1, s_2, \dots, s_t)$ maximizing $Score(\mathbf{s}, DNA)$

The Motif Finding Problem: Brute Force Solution

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

$$s_i \in [1, \dots, n - \ell + 1]$$

$$i \in [1, \dots, t]$$

BruteForceMotifSearch

BruteForceMotifSearch(DNA, t, n, ℓ)

- 1: $bestScore \leftarrow 0$
- 2: **for** each $\mathbf{s} = (s_1, s_2, \dots, s_t)$ from $(1, 1, \dots, 1)$ to $(n - \ell + 1, \dots, n - \ell + 1)$ **do**
- 3: **if** $Score(\mathbf{s}, DNA) > bestScore$ **then**
- 4: $bestScore \leftarrow score(\mathbf{s}, DNA)$
- 5: $bestMotif \leftarrow (s_1, s_2, \dots, s_t)$
- 6: **return** $bestMotif$

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 ℓt 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

Hamming Distance

- ▶ 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:

$$d_H(\text{AAAAAA}, \text{ACAAAC}) = 2$$

Total Distance: Example

- ▶ Given $v = \text{"acgtacgt"}$ and s :

$$\text{acgtacgt} \rightarrow d_H(v, x) = 1$$

cctgatagacgctatctggctatccacgtacAtaggtcctctgtgcgaatctatgcgtttccaacccat

$$\text{acgtacgt} \rightarrow d_H(v, x) = 0$$

agtactggtgtacatttgatacgtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc

$$\text{acgtacgt} \rightarrow d_H(v, x) = 2$$

aaaAgtCcgtagcaccctctttcttcgtggctctggccaacgagggtgatgtataagacgaaaatttt

$$d_H(c, x) = 0 \leftarrow \text{acgtacgt}$$

agcctccgatgtaagtcatactgtaactattacctgccaccctattacatcttacgtacgtataca

$$d_H(c, x) = 1 \leftarrow \text{acgtacgt}$$

ctgttatacaacgcgctcatggcggggtatgcgttttggtcgtcgtacgctcgatcggttaacgtaGgtc

- ▶ $TotalDistance(v, DNA) = 1 + 0 + 2 + 0 + 1 = 4$

Total Distance: Definition

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

$$TotalDistance(v, DNA) = \sum_{i \in [1, t]} \min_{s_i \in [1, m - \ell + 1]} d_H(v, DNA[i][s_i, \dots, s_i + \ell - 1])$$

The Median String Problem: Formulation

- ▶ Goal: Given a set of DNA sequences, find a median string
- ▶ Input: a $t \times n$ matrix DNA , and ℓ , the length of the pattern to find
- ▶ Output: A string v of ℓ nucleotides that **minimizes** $TotalDistance(v, DNA)$ over all strings of that length

Median String Search Algorithm

MedianStringSearch(DNA, t, n, ℓ)

- 1: $bestWord \leftarrow AAA \dots A$
- 2: $bestDistance \leftarrow \infty$
- 3: **for** each ℓ -mer v from $AAA \dots A$ to $TTT \dots T$ **do**
- 4: **if** $TotalDistance(v, DNA) < bestDistance$ **then**
- 5: $bestDistance \leftarrow TotalDistance(v, DNA)$
- 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

Alignment	$\overbrace{\hspace{1.5cm}}^{\ell}$							
	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

Profile	A	3	0	1	0	3	1	1	0
	C	2	4	0	0	1	4	0	0
	G	0	1	4	0	0	0	3	1
	T	0	0	0	5	1	0	1	4

Consensus A C G T A C G T

Score 3+4+4+5+3+4+3+4

TotalDistance 2+1+1+0+2+1+2+1

Sum 5 5 5 5 5 5 5 5

- ▶ At any column i

$$Score_i + TotalDistance_i = t$$

- ▶ ℓ columns $\implies$

$$Score + TotalDistance = \ell \cdot t$$

- ▶ Rearranging:

$$Score = \ell \cdot t - TotalDistance$$

- ▶ $\ell \cdot t$ 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
 $\implies$ Total running time $O(\ell tn^t)$
 - ▶ The Median String problem needs to examine all 4^ℓ combinations for v
 $\implies$ Total running time $O(\ell tn4^\ell)$

Outline

Implanted motifs - an introduction

Biological motivation

Motif Finding Problem and Median String Problem

Structuring the search: search tree

Branch and bound

Weeder

Study group assignments

Motif Finding: Improving the Running Time

Recall the BruteForceMotifSearch:

BruteForceMotifSearch(DNA, t, n, ℓ)

- 1: $bestScore \leftarrow 0$
- 2: **for** each $\mathbf{s} = (s_1, s_2, \dots, s_t)$ from $(1, 1, \dots, 1)$ to $(n - \ell + 1, \dots, n - \ell + 1)$ **do**
- 3: **if** $Score(\mathbf{s}, DNA) > bestScore$ **then**
- 4: $bestScore \leftarrow score(\mathbf{s}, DNA)$
- 5: $bestMotif \leftarrow (s_1, s_2, \dots, s_t)$
- 6: **return** $bestMotif$

Structuring the Search

- ▶ How can we perform the line
 for each $\mathbf{s} = (s_1, s_2, \dots, s_t)$ from $(1, 1, \dots, 1)$ to
 $(n - \ell + 1, \dots, n - \ell + 1)$ **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

Median String: Improving the Running Time

Recall the MedianStringSearch:

MedianStringSearch(DNA, t, n, ℓ)

- 1: $bestWord \leftarrow AAA \dots A$
- 2: $bestDistance \leftarrow \infty$
- 3: **for** each ℓ -mer v from $AAA \dots A$ to $TTT \dots T$ **do**
- 4: **if** $TotalDistance(v, DNA) < bestDistance$ **then**
- 5: $bestDistance \leftarrow TotalDistance(v, DNA)$
- 6: $bestWord \leftarrow v$
- 7: **return** $bestWord$

Structuring the Search

- ▶ How can we perform the line
for each ℓ -mer v **from** AAA...A **to** TTT...T **do**
- ▶ We need a method for efficiently structuring and navigating all 4^ℓ possible ℓ -mers
- ▶ This is not very different from exploring all ℓ -digit numbers (in base 4):

$$\begin{array}{c} \overbrace{\text{AA} \dots \text{AA}}^{\ell} \\ \text{AA} \dots \text{AC} \\ \text{AA} \dots \text{AG} \\ \text{AA} \dots \text{AT} \\ \cdot \\ \cdot \\ \text{TT} \dots \text{TT} \end{array}$$

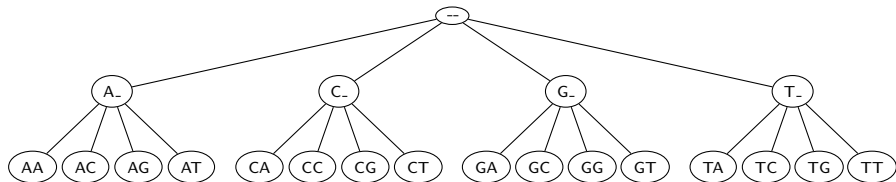
Alternative Representation of the Search Space

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

$$\begin{array}{c} \overbrace{11 \dots 11}^{\ell} \\ 11 \dots 12 \\ 11 \dots 13 \\ 11 \dots 14 \\ \cdot \\ \cdot \\ 44 \dots 44 \end{array}$$

- ▶ Notice that the sequences above simply list all ℓ -digit numbers in base 4 using digits 1, 2, 3, and 4

Search 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?

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: for  $i \leftarrow L$  to 1 do  
2:   if  $a_i < k$  then  
3:      $a_i \leftarrow a_i + 1$   
4:     return  $a$   
5:    $a_i \leftarrow 1$   
6: return  $a$ 
```

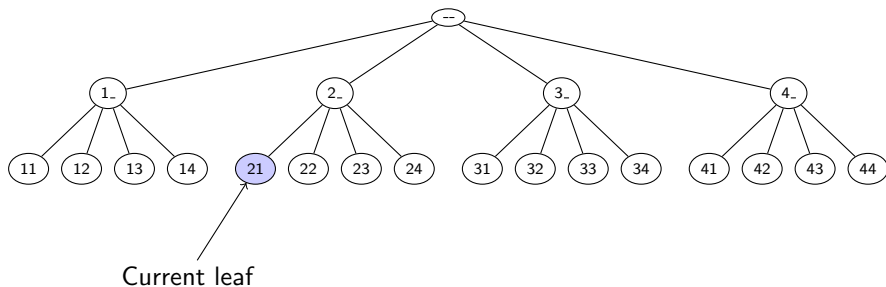
► Arguments:

- a : the array of digits
- L : length of the array
- k : max digit value

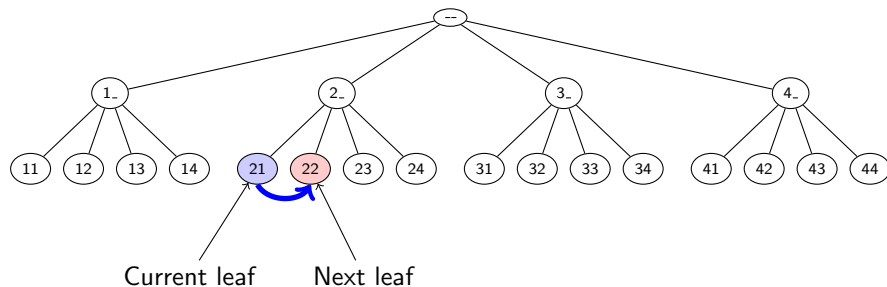
► The algorithm is common addition in radix k :

- Increment the least significant digit
- “Carry the one” to the next digit position when the digit is at maximal value

NextLeaf: Example



NextLeaf: Example



Visit All Leaves

Print all permutations in ascending order:

AllLeaves(L, k)

1: $a \leftarrow (1, \dots, 1)$

2: **while** forever **do**

3: output a

4: $a \leftarrow \text{NextLeaf}(a, L, k)$

5: **if** $a = (1, 1, \dots, 1)$ **then**

6: **return**

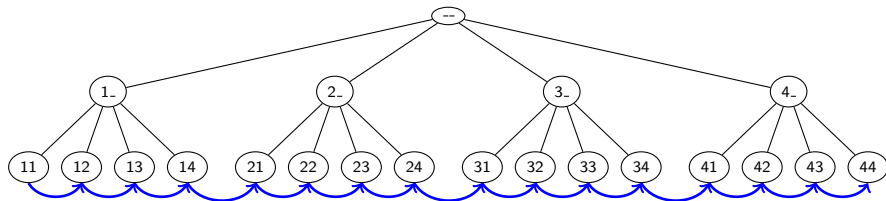
► L : length of the sequence

► 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:

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

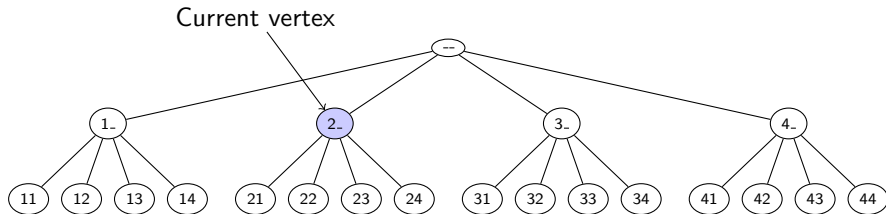
```
1: if  $i < L$  then  
2:    $a_{i+1} \leftarrow 1$   
3:   return  $(a, i + 1)$   
4: else  
5:   for  $j \leftarrow L$  to 1 do  
6:     if  $a_j < k$  then  
7:        $a_j \leftarrow a_j + 1$   
8:       return  $(a, j)$   
9: return  $(a, 0)$ 
```

► 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

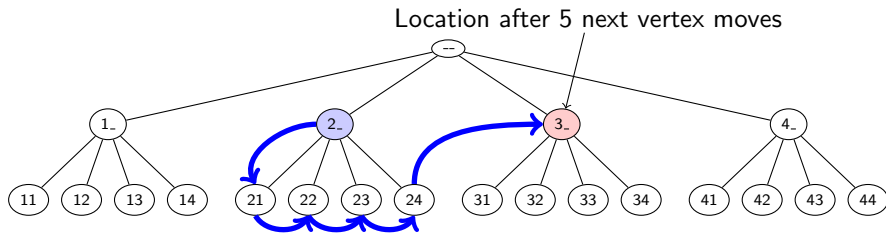
NextVertex: Example

Moving to the next vertices:



NextVertex: Example

Moving to the next vertices:



Bypass Move

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

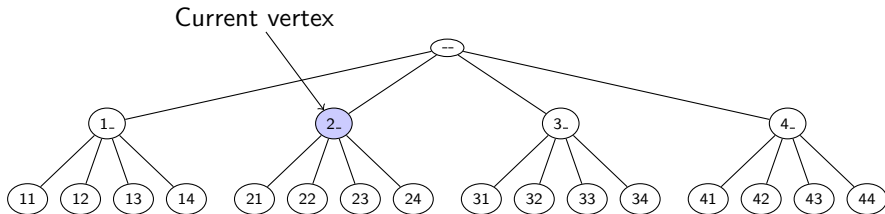
Bypass(a, i, L, k):

```
1: for  $j \leftarrow i$  to 1 do  
2:   if  $a_j < k$  then  
3:      $a_j \leftarrow a_j + 1$   
4:     return ( $a, j$ )  
5: return ( $a, 0$ )
```

- ▶ a : array of digits
- ▶ i : prefix length
- ▶ L : maximum length
- ▶ k : max digit value

Bypass Move: Example

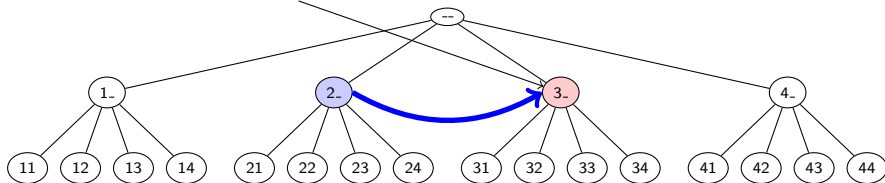
Bypassing descendants of "2-"



Bypass Move: Example

Bypassing descendants of "2-"

Next vertex



Outline

Implanted motifs - an introduction

Biological motivation

Motif Finding Problem and Median String Problem

Structuring the search: search tree

Branch and bound

Weeder

Study group assignments

Brute Force Search Again

Now that we have a method for navigating the tree, let's look again at BruteForceMotifSearch.

BruteForceMotifSearchAgain(DNA, t, n, ℓ)

```
1:  $\mathbf{s} \leftarrow (1, 1, \dots, 1)$ 
2:  $bestScore \leftarrow score(\mathbf{s}, DNA)$ 
3: while forever do
4:    $\mathbf{s} \leftarrow \text{NextLeaf}(\mathbf{s}, t, n - \ell + 1)$ 
5:   if  $Score(\mathbf{s}, DNA) > bestScore$  then
6:      $bestScore \leftarrow score(\mathbf{s}, DNA)$ 
7:      $bestMotif \leftarrow (s_1, s_2, \dots, s_t)$ 
8:   if  $\mathbf{s} = (1, 1, \dots, 1)$  then
9:     return  $bestMotif$ 
```

Can We Do Better?

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

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

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

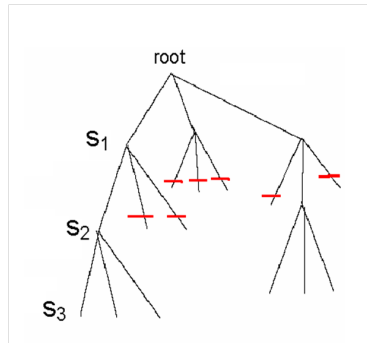
- ▶ Optimism: if all remaining $(t - i)$ positions $(s_{i+1}, \dots, s_t)$ add

$$(t - i) \cdot \ell \text{ to } \text{Score}(s, i, \text{DNA})$$

- ▶ If $\text{Score}(s, i, \text{DNA}) + (t - i) \cdot \ell < \text{BestScore}$, 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, ℓ)

```
1:  $s \leftarrow (1, 1, \dots, 1)$ 
2:  $bestScore \leftarrow 0$ 
3:  $i \leftarrow 1$ 
4: while  $i > 0$  do
5:   if  $i < t$  then
6:      $optimisticScore \leftarrow Score(s, i, DNA) + (t - i) \cdot \ell$ 
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  $Score(s, DNA) > bestScore$  then
13:        $bestScore \leftarrow score(s, DNA)$ 
14:        $bestMotif \leftarrow (s_1, s_2, \dots, s_t)$ 
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 $\mathbf{s}$
 - ▶ The Median String Problem needs to examine 4^ℓ 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:

$$\textit{TotalDistance}(\textit{prefix}, \textit{DNA}) > \textit{BestDistance}$$

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

Implanted motifs - an introduction

Biological motivation

Motif Finding Problem and Median String Problem

Structuring the search: search tree

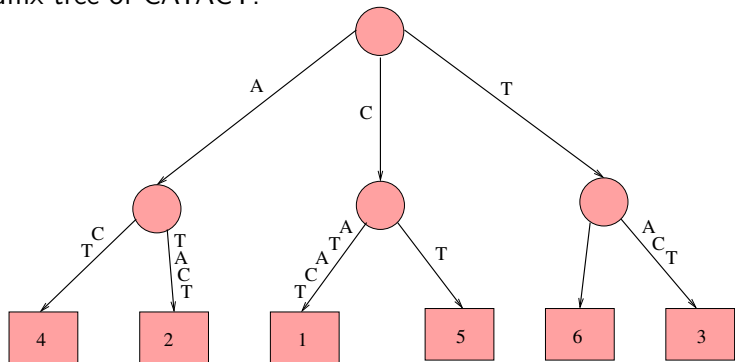
Branch and bound

Weeder

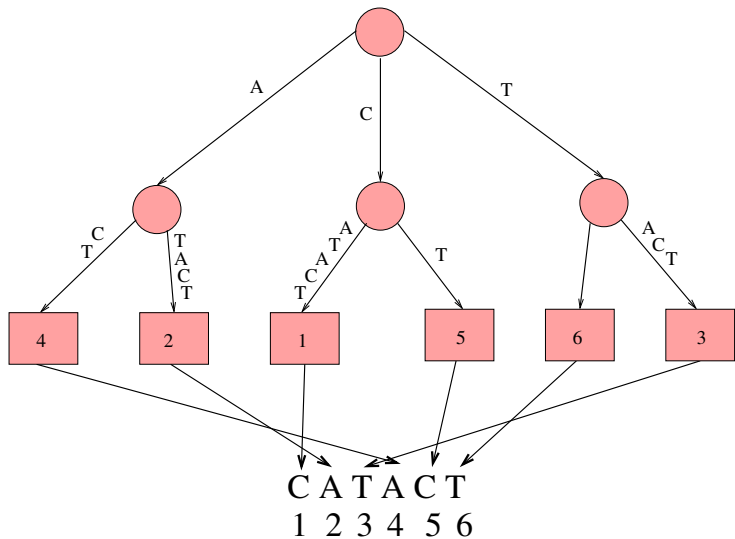
Study group assignments

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:

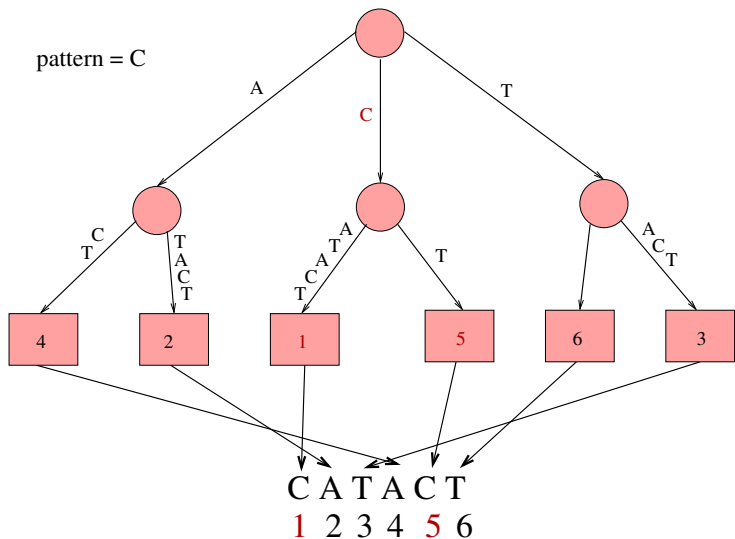


Suffix Tree

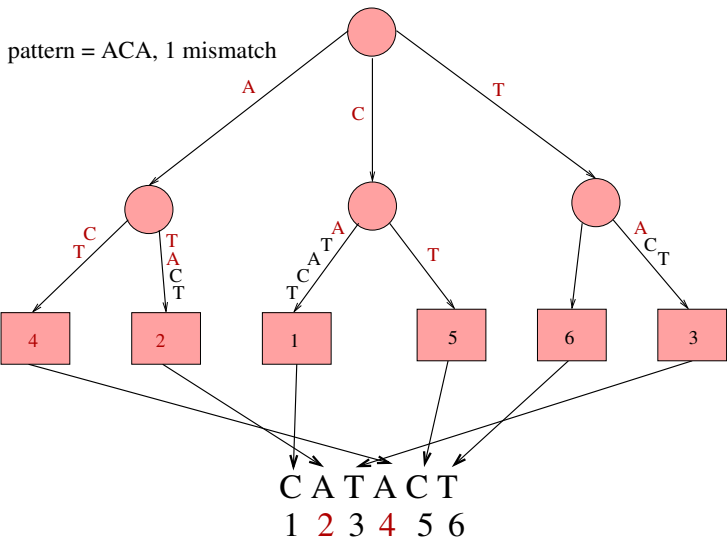


⇒ 58093 String Processing Algorithms (4 cr), period II

Exact Search on Suffix Tree



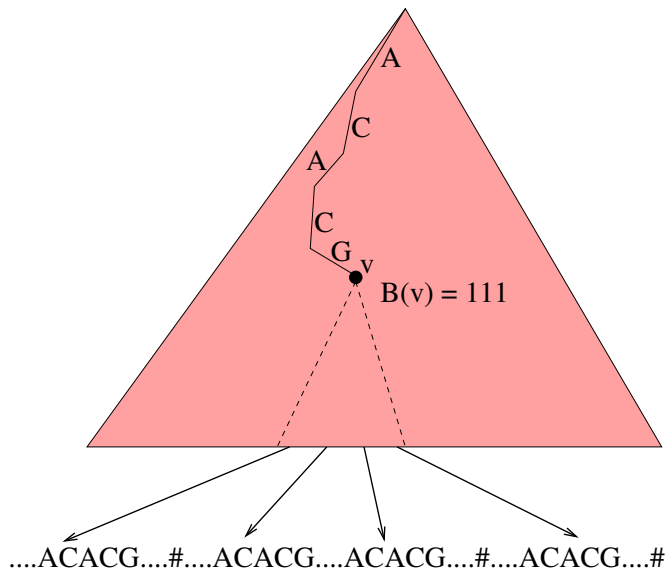
Backtracking on Suffix Tree



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



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

Implanted motifs - an introduction

Biological motivation

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 S-Z)

- ▶ 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 A-G)

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

Alvis Brazma, Inge Jonassen, Jaak Vilo, and Esko Ukkonen.
Predicting Gene Regulatory Elements in Silico on a Genomic Scale.
Genome Res. 1998. 8: 1202-1215.

<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 H-R)

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

Pavesi G, Mauri G, Pesole G. An algorithm for finding signals of unknown length in DNA sequences. *Bioinformatics*, 2001, 17(Suppl 1):S207-S214.

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.