
CS481/CS583: Bioinformatics Algorithms

Can Alkan

EA509

`calkan@cs.bilkent.edu.tr`

<http://www.cs.bilkent.edu.tr/~calkan/teaching/cs481/>

MOTIFS

Random Sample

atgaccgggatactgataccgtatTTTggcctaggcgtacacattagataaacgtatgaagtacgttagactcggcgccgccg
accctatTTTTtgagcagatttagtgacctggaaaaaaatttgagtacaaaactTTTccgaatactgggcataaggtaca
tgagtatccctgggatgactTTTgggaacactatagtgctctccgattTTTgaatatgtaggatcattcgccagggtccga
gctgagaattggatgaccttgtaagtgtTTTccacgcaatcgcgaaaccaacgcggacccaaaggcaagaccgataaaggaga
tcctTTTgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaatggcccacttagtccacttatag
gtcaatcatgttcttgtgaatggattTTTtaactgagggcatagaccgcttggcgcacccaaattcagtgtgggcgagcgcaa
cggTTTTggcccttgttagaggccccgtactgatggaaactTTTcaattatgagagagctaatttatcgcggtgcgtgttcat
aacttgagttggTTTcgaaaatgctctggggcacatacaagaggagtcttcttatcagttaatgctgtatgacactatgta
ttggccattggctaaaagcccaacttgacaaatggaagatagaatccttgcatttcaacgtatgccgaaccgaaagggaag
ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagcttctgggtactgatagca

Implanting Motif **AAAAAAAAAGGGGGGGG**

atgaccgggatactgat**AAAAAAAAAGGGGGGGG**ggcggtacacattagataaacgtatgaagtacgttagactcggcgccgccg
accctattttttgagcagatttagtgacctggaaaaaaatttgagtacaaaacttttccgaata**AAAAAAAAAGGGGGGGG**a
tgagtatccctgggatgactt**AAAAAAAAAGGGGGGGG**tgctctccgatttttgaatatgtaggatcattcgccagggtccga
gctgagaattggatg**AAAAAAAAAGGGGGGGG**tccacgcaatcgcgaaaccaacgcggacccaaaggcaagaccgataaaggaga
tcctttttgcggtaatgtgccgggaggctgggttacgtagggaagccctaacggacttaat**AAAAAAAAAGGGGGGGG**cttatag
gtcaatcatgttcttgtgaatggattt**AAAAAAAAAGGGGGGGG**gaccgcttggcgcacccaaattcagtgtgggcgagcgcaa
cggttttggcccttggttagaggccccgt**AAAAAAAAAGGGGGGGG**caattatgagagagctaattctatcgctgctgtttcat
aacttgagtt**AAAAAAAAAGGGGGGGG**ctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgta
ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcatt**AAAAAAAAAGGGGGGGG**accgaaagggaag
ctggtgagcaacgacagattcttacgtgcattagctcgcttcggggatctaatagcacgaagctt**AAAAAAAAAGGGGGGGG**a

Where is the Implanted Motif?

atgaccgggatactgataaaaaaaagggggggggcggtacacattagataaacgtatgaagtacgttagactcggcgccgccg
accctattttttgagcagatttagtgacctggaaaaaaatttgagtacaaaacttttccgaataaaaaaaaaggggggga
tgagtatccctgggatgacttaaaaaaaaggggggggtgctctccgatttttgaatatgtaggatcattcgccagggtccga
gctgagaattggatgaaaaaaaaggggggggtccacgcaatcgcgaccaacgcggacccaaaggcaagaccgataaaggaga
tcccttttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaataaaaaaaagggggggccttatag
gtcaatcatgttcttgtgaatggatttaaaaaaaaggggggggaccgcttggcgcacccaaattcagtgtggcgagcgcaa
cggttttggcccttgtagaggccccgtaaaaaaaagggggggcaattatgagagagctaattctatcgcggtgcgtgttcat
aacttgagttaaaaaaaaggggggggctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgta
ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcataaaaaaaaagggggggaccgaaagggaag
ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagcttaaaaaaaaggggggga

Implanting Motif AAAAAAGGGGGGGG with Four Mutations

atgaccgggatactgatAgAAgAAAGGttGGGggcgtacacattagataaacgtatgaagtacgttagactcggcgccgccg
accctattttttgagcagatttagtgacctggaaaaaaatttgagtacaaaacttttccgaataCAAtAAAAcGGcGGGa
tgagtatccctgggatgacttAAAAtAATGGaGtGGTgctctcccgatttttgaatatgtaggatcattcgccagggtccga
gctgagaattggatgcAAAAAAGGGattGttccacgcaatcgcgaaaccaacgcggacccaaaggcaagaccgataaaggaga
tcctttttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaatAtAAtAAAGGaaGGGcttatag
gtcaatcatgttcttgtgaatggatttAAcAAAtAAGGGctGGgaccgcttggcgcacccaaattcagtggtggcgagcgcaa
cggttttggcccttgtagaggccccgtAtAAAcAAGGaGGGccaattatgagagagctaattctatcgctgcgtgttcat
aacttgagttAAAAAAtAGGGaGccctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgta
ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcatActAAAAGGaGcGGGaccgaaagggaag
ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagcttActAAAAAGGaGcGGGa

Where is the Motif???

atgaccgggatactgatagaagaaagggttgggggcggtacacattagataaacgtatgaagtacgttagactcggcgccgccg
accctattttttgagcagatttagtgacctggaaaaaaatttgagtacaaaacttttccgaatacaataaaacggcgggga
tgagtatccctgggatgacttaaaataatggagtggtgctctccgatttttgaatatgtaggatcattcgccagggtccga
gctgagaattggatgcaaaaaaagggttgtccacgcaatcgcgaaaccaacgcggacccaaaggcaagaccgataaaggaga
tcctttttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaataataaaggaagggttatag
gtcaatcatgttcttgtgaatggatttaacaataagggttgggaccgcttggcgcacccaaattcagtggtggcgagcgcaa
cggttttggcccttgtagaggccccgtataaacaaggaggggccaattatgagagagctaatactatcgcggtgcgtgttcat
aacttgagttaaaaaatagggagccctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgta
ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcataactaaaaaggagcggaccgaaagggaag
ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagcttactaaaaaggagcgga

Finding (15,4) Motif

atgaccgggatactgatAgAAGAAAGGttGGGggcggtacacattagataaacgtatgaagtacgttagactcggcgccgccg
accctattttttgagcagatttagtgacctggaaaaaaatttgagtacaaaacttttcgaatacAAtAAAcGGcGGGa
tgagtatccctgggatgacttAAAtAAtGGaGtGGTgctctccgatttttgaatatgtaggatcattcgccagggtccga
gctgagaattggatgcAAAAAAGGGattGttccacgcaatcggaaccaacgcggacccaaaggcaagaccgataaaggaga
tcctttttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaatAtAAtAAAGGaaGGGcttatag
gtcaatcatgttcttgtgaatggatttAAcAAtAAGGGctGGgaccgcttggcgcacccaaattcagtgtgggcgagcgcaa
cggttttggcccttgtagaggccccgtAtAAAcAAGGaGGGcCaattatgagagagctaatttatcgctgcgtgttcat
aacttgagttAAAAAAtAGGGaGccctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgta
ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcatActAAAAAGGaGcGGaccgaaagggaag
ctggtgagcaacgacagattcttacgtgcattagctcgcttcggggatctaatagcacgaagcttActAAAAAGGaGcGGa

AgAAGAAAGGttGGG
..|..|||.|||
cAAtAAAcGGcGGG

Challenge Problem

- Find a motif in a sample of
 - 20 “random” sequences (e.g., 600 chars long)
 - each sequence containing an implanted pattern of length 15,
 - each pattern appearing with 4 mismatches as $(15,4)$ -motif.
-

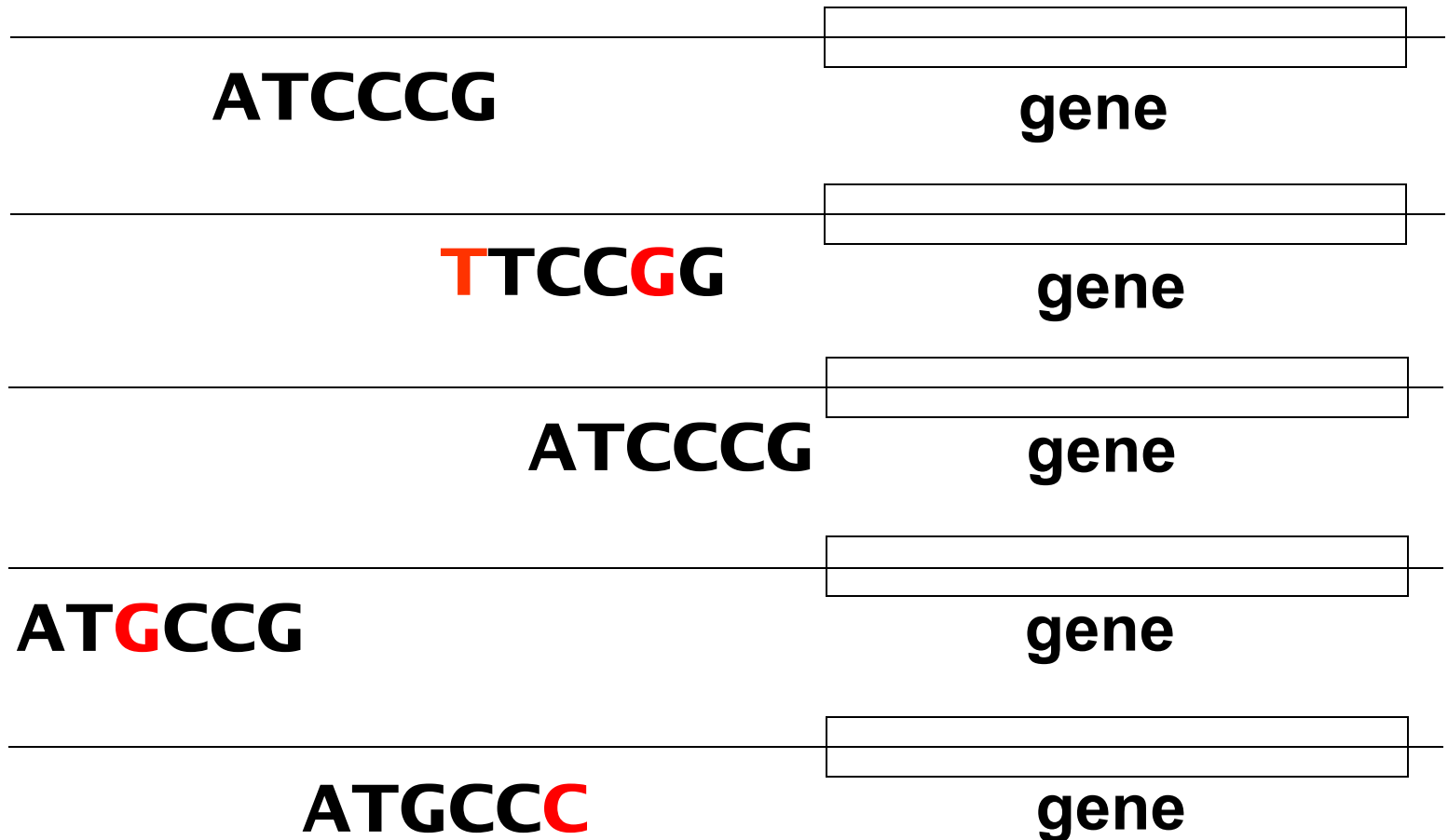
Regulatory Regions

- Every gene contains a regulatory region (RR) typically stretching 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 Start Sites



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 amongst those genes
-

Identifying Motifs: Complications

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

The Motif Finding Problem

- Given a random sample of DNA sequences:

```
cctgatagacgctatctggctatccacgtacgtaggtcctctgtgcgaatctatgcgtttccaaccat  
agtactggtgtacatttgatacgtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc  
aaacgtacgtgcaccctctttcttctgtggctctggccaacgagggctgatgtataagacgaaaatttt  
agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatcttacgtacgtataca  
ctgttatacaacgcgtcatggcgggggtatgcgtttttggtcgtcgtacgctcgatcgttaacgtacgtc
```

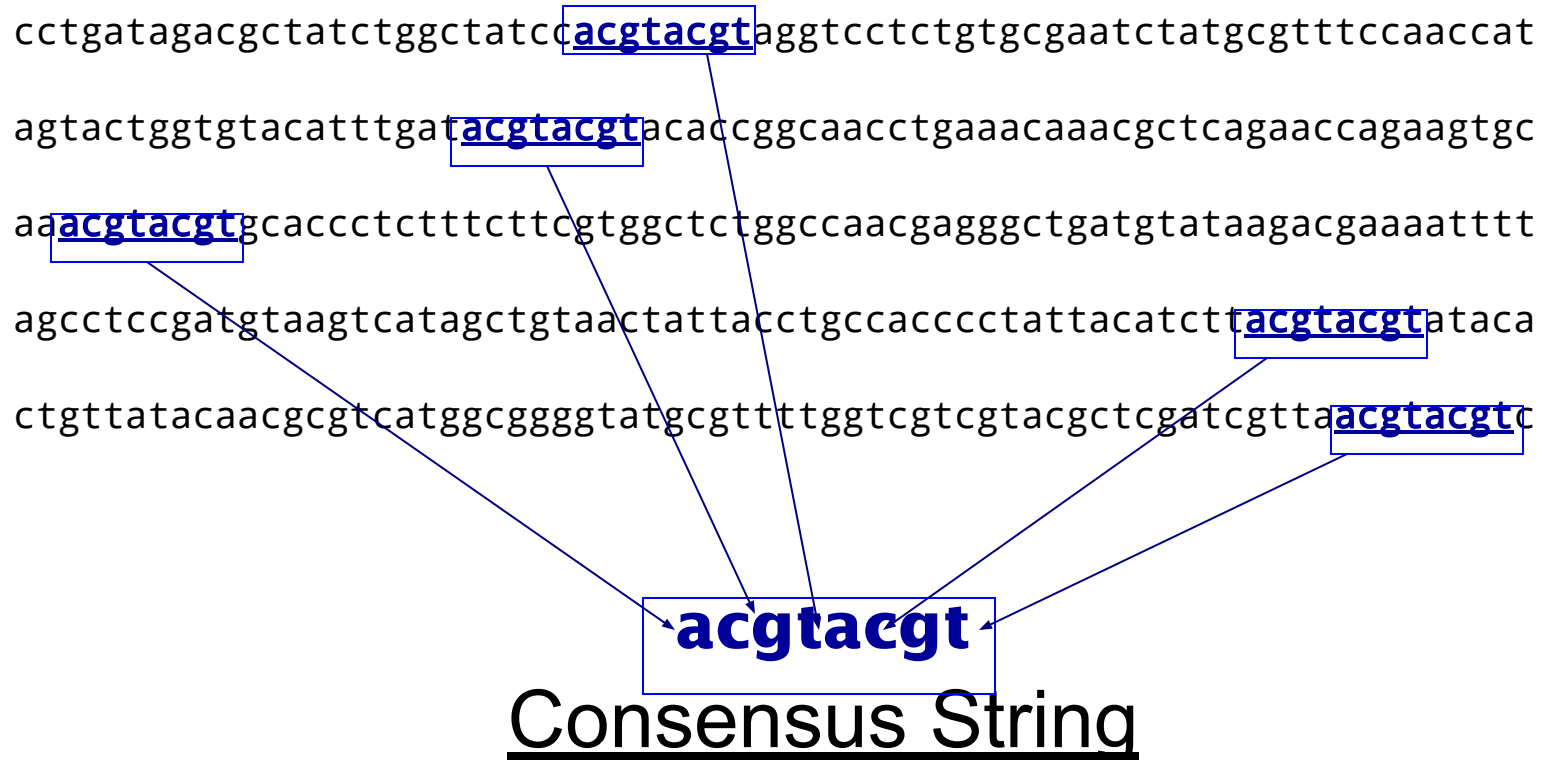
- Find the pattern that is implanted in each of the individual sequences, namely, the motif

The Motif Finding Problem (cont'd)

- Additional information:
 - The hidden sequence is of length 8
 - The pattern is not exactly the same in each array because random point mutations may occur in the sequences
-

The Motif Finding Problem (cont'd)

- The patterns revealed with no mutations:



The Motif Finding Problem (cont'd)

- The patterns with 2 point mutations:

cctgatagacgctatctggctatccaGgtacItaggtcctctgtgCGaatctatgcgttttccaaccat
agtactggtgtacatttgatCcAtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc
aaacgtIAgtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt
agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatcttacgtCcAtataca
ctgttatacaacgcgtcatggcgggggtatgcgtttttggtcgtcgtacgctcgatcgттаCcgtacgGc

The Motif Finding Problem (cont'd)

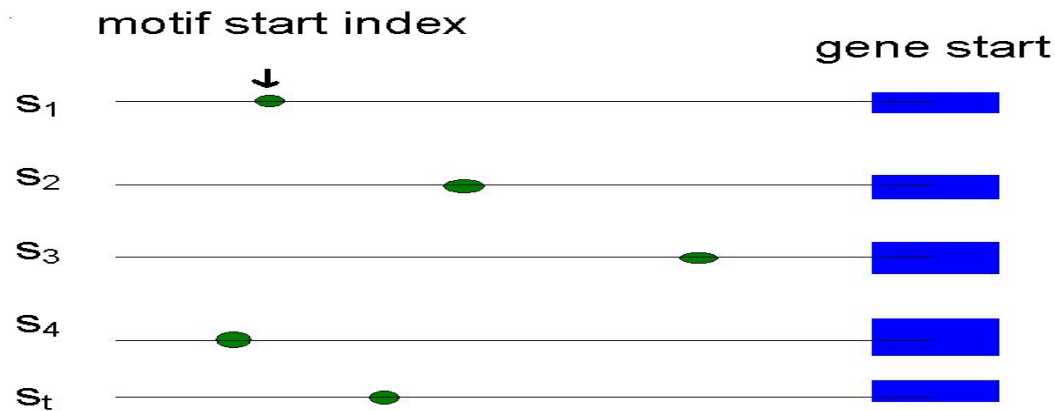
- The patterns with 2 point mutations:

cctgatagacgctatctggctatccaGgtacItaggtcctctgtgcgaatctatgcgtttccaacat
agtactggtgtacatttgatCcAtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc
aaacgtIAgtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt
agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatcttacgtCcAtataca
ctgttatacaacgcgtcatggcggggtatgcgtttttggtcgtcgtacgctcgatcgттаCcgtacgGc

Can we still find the motif, now that we have 2 mutations?

Defining Motifs

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



Motifs: Profiles 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

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

- Line up the patterns by their start indexes

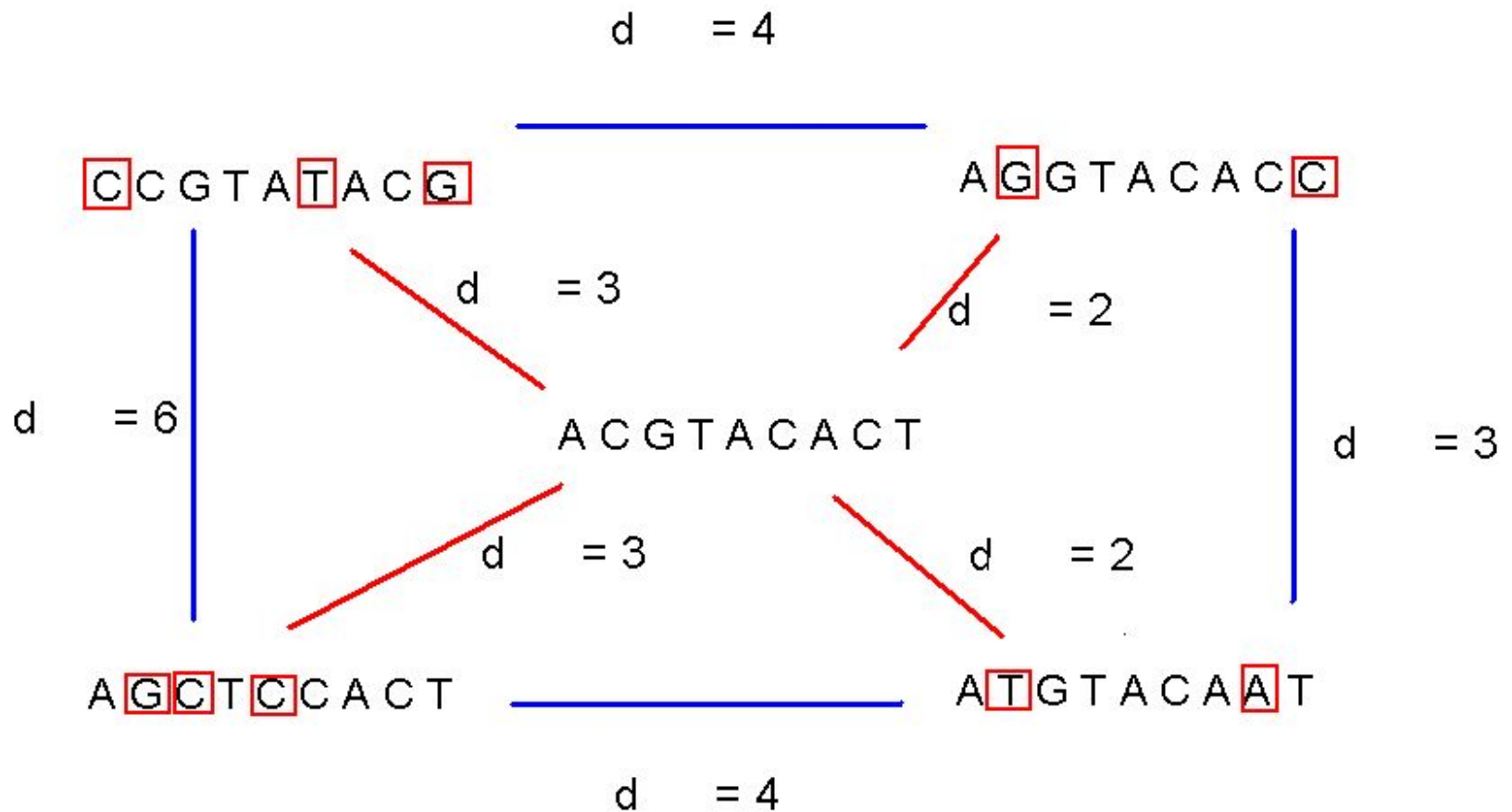
$$\mathbf{s} = (s_1, s_2, \dots, s_t)$$

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

Consensus

- Think of consensus as an “ancestor” motif, from which mutated motifs emerged
- The *distance* between a real motif and the consensus sequence is generally less than that for two real motifs

Consensus (cont'd)



Evaluating Motifs

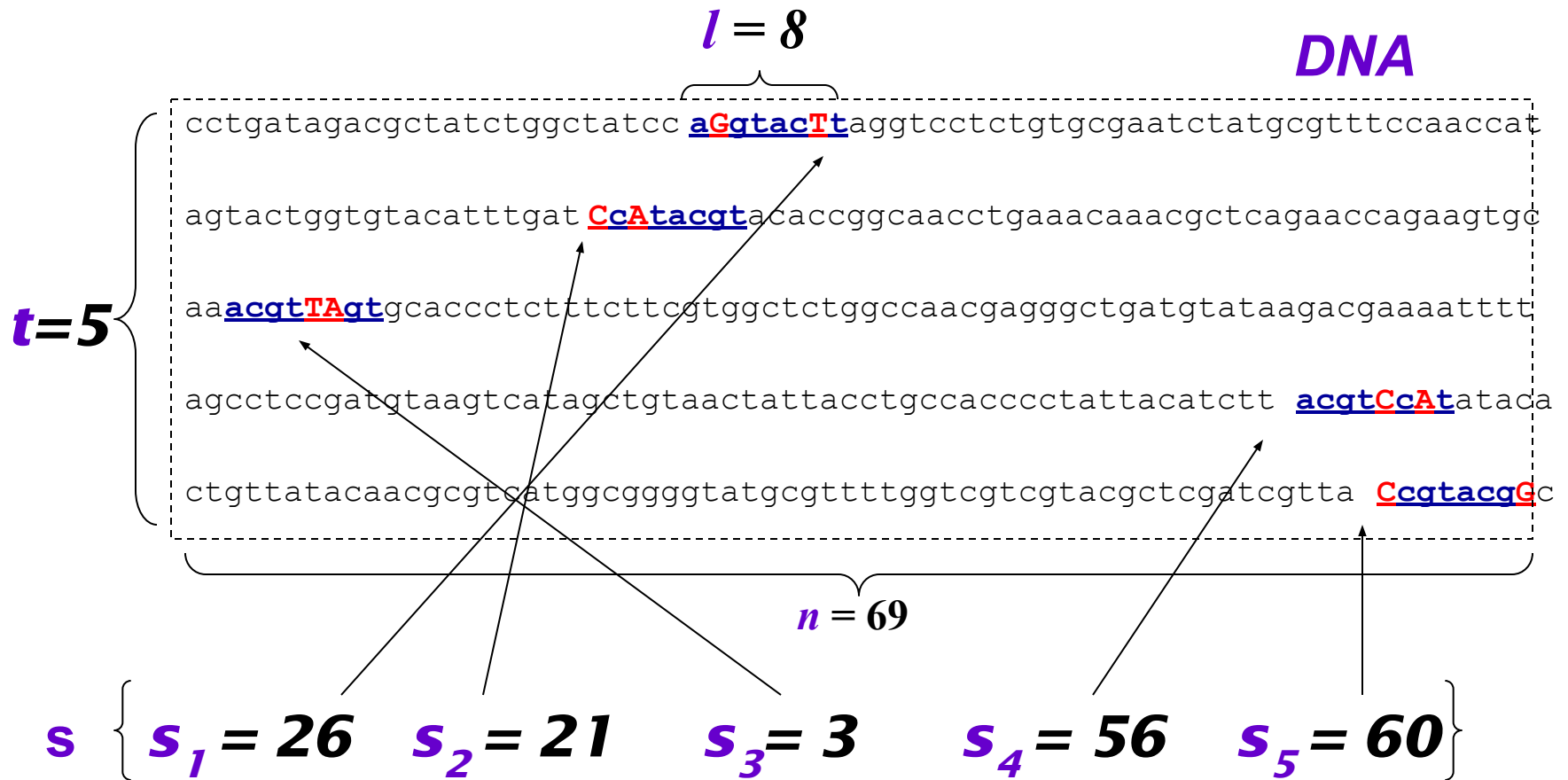
- We have a guess about the consensus sequence, but how “good” is this consensus?
 - Need to introduce a scoring function to compare different guesses and choose the “best” one.
-

Defining Some Terms

- t - number of sample DNA sequences
- n - length of each DNA sequence
- **DNA** - sample of DNA sequences ($t \times n$ array)

- 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



Scoring Motifs

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

$$\text{Score}(\mathbf{s}, \mathbf{DNA}) = \sum_{i=1}^l \max_{k \in \{A, T, C, G\}} \text{count}(k, i)$$

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

Consensus a c g t a c g t

Score 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 of **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 $\text{Score}(\mathbf{s}, \mathbf{DNA})$

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 $\text{Score}(\mathbf{s}, \mathbf{DNA})$ by varying the starting positions \mathbf{s}_i , where:

$$\begin{aligned}\mathbf{s}_i &= [1, \dots, n-\ell+1] \\ i &= [1, \dots, t]\end{aligned}$$

BruteForceMotifSearch

1. BruteForceMotifSearch(***DNA***, ***t***, ***n***, \emptyset)
2. ***bestScore*** \leftarrow 0
3. **for** each ***s***=(s_1, s_2, \dots, s_t) from (1,1 . . . 1)
to (***n***-***l***+1, . . . , ***n***-***l***+1)
4. **if** (Score(***s***,***DNA***) > ***bestScore***)
5. ***bestScore*** \leftarrow score(***s***, ***DNA***)
6. **bestMotif** \leftarrow (s_1, s_2, \dots, s_t)
7. **return bestMotif**

Running Time of BruteForceMotifSearch

- Varying $(n - \ell + 1)$ positions in each of t sequences, we're looking at $(n - \ell + 1)^t$ sets of starting positions
- For each set of starting positions, the scoring function makes ℓ operations, so complexity is $\ell(n - \ell + 1)^t = O(\ell n^t)$
- That means that for $t = 8$, $n = 1000$, $\ell = 10$ we must perform approximately 10^{20} computations

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

- Hamming distance:
 - $d_H(\mathbf{v}, \mathbf{w})$ is the number of nucleotide pairs that do not match when \mathbf{v} and \mathbf{w} are aligned. For example:

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

Total Distance: An Example

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

$$d_H(\mathbf{v}, \mathbf{x}) = 0$$

cctgatagacgctatctggctatcc **acgtacgt** aggtcctctgtgcgaatctatgcgtttccaacccat
 $d_H(\mathbf{v}, \mathbf{x}) = 0$ **acgtacgt**
agtactgggtgtacatttgat **acgtacgt** acaccggcaacctgaaacaaacgctcagaaccagaagtgc
acgtacgt
aa **acgtacgt** gcaccctcttttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt
 $d_H(\mathbf{v}, \mathbf{x}) = 0$ **acgtacgt**
agcctccgatgtaagtcatagctgtaactattacctgccaccctattacatctt **acgtacgt** ataca
 $d_H(\mathbf{v}, \mathbf{x}) = 0$ **acgtacgt**
ctgttatatacaacgcgtcatggcgggggtatgcgttttggtcgctcgtagcgtcgcgatcgтта **acgtacgt**c
 $d_H(\mathbf{v}, \mathbf{x}) = 0$

\mathbf{v} is the sequence in red, \mathbf{x} is the sequence in blue

- $TotalDistance(\mathbf{v}, \mathbf{DNA}) = 0$

$$d_H(\mathbf{v}, \mathbf{x}) = 0$$

Total Distance: Example

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

$$d_H(v, x) = 1$$

cctgata~~g~~acgctatctggctatcc~~acgtacgt~~acgtac~~gt~~ataggtcctctgtgcgaatctatgcggtttccaaccat

$$d_H(v, x) = 0$$

agta~~ct~~gggtgtacatttgat~~acgtacgt~~acaccggcaacctgaaacaaacgctcagaaccagaaagtgc

$$d_H(v, x) = 2$$

aaa~~acgt~~~~acgt~~Ccgtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt

$$d_H(v, x) = 0$$

agcctccgatgtaagtcata~~g~~ctgtaactattacctgcccaccctattacatctt~~acgtacgt~~acgtacgtatataca

ctgttatacaacgcgctcatggcgggggtatgcggttttggtcgtcgtacgctcgcgatcgтта~~acgtacgt~~acgtacgtc

v is the sequence in red, x is the sequence in blue

$$d_H(v, x) = 1$$

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

Total Distance: Definition

- For each DNA sequence i , compute all $d_H(\mathbf{v}, \mathbf{x})$, where \mathbf{x} is an ℓ -mer with starting position s_i ($1 \leq s_i \leq n - \ell + 1$)
- Find minimum of $d_H(\mathbf{v}, \mathbf{x})$ among all ℓ -mers in sequence i
- $TotalDistance(\mathbf{v}, \mathbf{DNA})$ is the sum of the minimum Hamming distances for each DNA sequence i
- $TotalDistance(\mathbf{v}, \mathbf{DNA}) = \min_{\mathbf{s}} d_H(\mathbf{v}, \mathbf{s})$, where \mathbf{s} is the set of starting positions s_1, s_2, \dots, s_t

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 \mathbf{v} of ℓ nucleotides that **minimizes** $TotalDistance(\mathbf{v}, \mathbf{DNA})$ over all strings of that length

Median String Search Algorithm

1. MedianStringSearch (*DNA*, *t*, *n*, *l*)
2. ***bestWord*** \leftarrow AAA...A
3. ***bestDistance*** $\leftarrow \infty$
4. **for** each *l*-mer **s** **from** AAA...A to TTT...T
 if *TotalDistance*(**s**,*DNA*) < ***bestDistance***
5. ***bestDistance*** \leftarrow *TotalDistance*(**s**,*DNA*)
6. ***bestWord*** \leftarrow **s**
7. **return** ***bestWord***

Motif Finding Problem = Median String Problem

- The *Motif Finding* is a maximization problem while *Median String* is a minimization problem
 - However, the *Motif Finding* problem and *Median String* problem are computationally equivalent
 - Need to show that minimizing *TotalDistance* is equivalent to maximizing *Score*
-

We are looking for the same thing

Alignment	<div><div><div><div><div></div><div>a</div><div>G</div><div>g</div><div>t</div><div>a</div><div>c</div><div>T</div><div>t</div></div><div><div>C</div><div>c</div><div>A</div><div>t</div><div>a</div><div>c</div><div>g</div><div>t</div></div><div><div>a</div><div>c</div><div>g</div><div>t</div><div>T</div><div>A</div><div>g</div><div>t</div></div><div><div>a</div><div>c</div><div>g</div><div>t</div><div>C</div><div>c</div><div>A</div><div>t</div></div><div><div>C</div><div>c</div><div>g</div><div>t</div><div>a</div><div>c</div><div>g</div><div>G</div></div></div><div></div><div>t</div></div></div>																																				
Profile	<table><tr><td>A</td><td>3</td><td>0</td><td>1</td><td>0</td><td>3</td><td>1</td><td>1</td><td>0</td></tr><tr><td>C</td><td>2</td><td>4</td><td>0</td><td>0</td><td>1</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><td>0</td><td>0</td><td>3</td><td>1</td></tr><tr><td>T</td><td>0</td><td>0</td><td>0</td><td>5</td><td>1</td><td>0</td><td>1</td><td>4</td></tr></table>	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
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
 $Score_i + TotalDistance_i = t$
- Because there are l columns
 $Score + TotalDistance = l * t$
- Rearranging:
 $Score = l * t - TotalDistance$
- $l * t$ is constant 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**. That is $(n - \ell + 1)^t$ combinations!!!
 - The Median String Problem needs to examine all 4^ℓ combinations for **v**. This number is relatively smaller

STRUCTURING SEARCH

Motif Finding: Improving the Running Time

Recall the BruteForceMotifSearch:

```
1. BruteForceMotifSearch(DNA, t, n, l)
2. bestScore  $\leftarrow$  0
3. for each s=( $s_1, s_2, \dots, s_t$ ) from (1,1 ... 1) to (n-l+1, ..., n-l+1)
4.     if (Score(s,DNA) > bestScore)
5.         bestScore  $\leftarrow$  Score(s, DNA)
6.         bestMotif  $\leftarrow$  ( $s_1, s_2, \dots, s_t$ )
7. 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 $(\mathbf{n-l+1}, \dots, \mathbf{n-l+1})$?

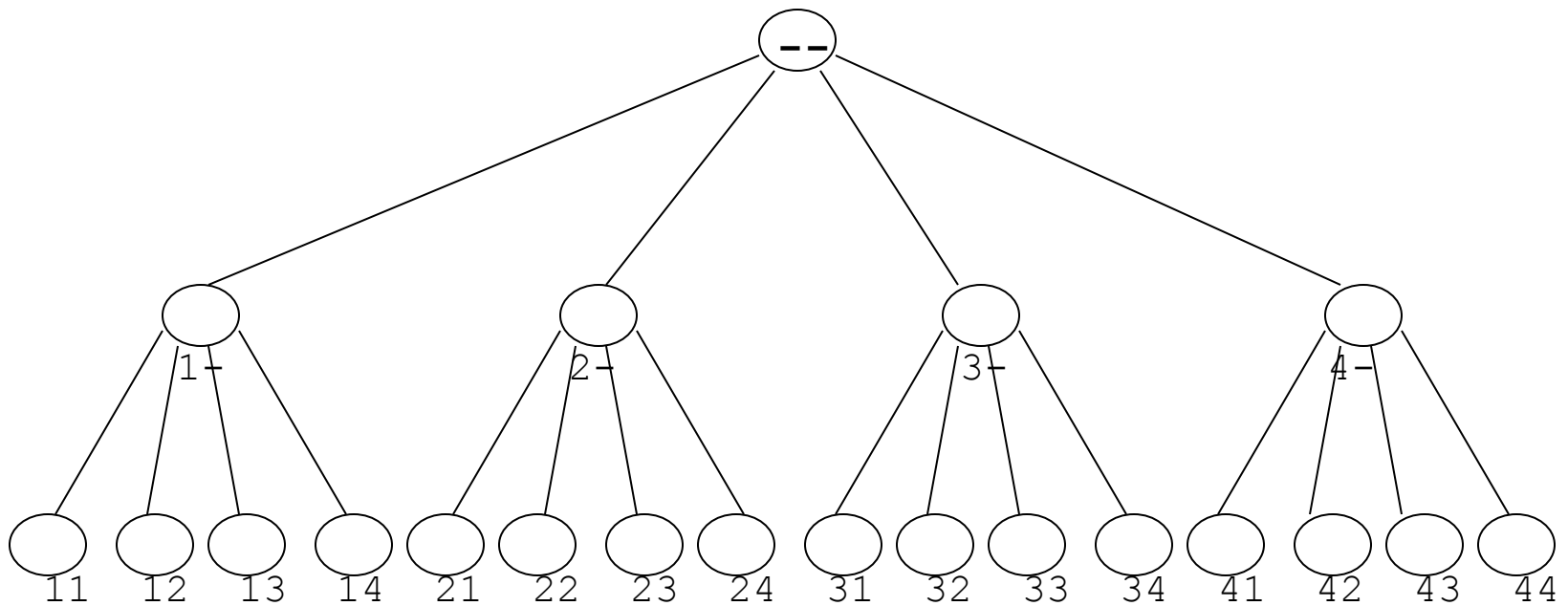
- We need a method for efficiently structuring and navigating the many possible motifs
- This is not very different than exploring all t -digit numbers

Median String: Improving the Running Time

1. MedianStringSearch (*DNA*, *t*, *n*, *l*)
2. ***bestWord* ← AAA...A**
3. ***bestDistance* ← ∞**
4. **for each *l*-mer *s* from AAA...A to TTT...T**
 if *TotalDistance*(*s*,*DNA*) < *bestDistance*
5. ***bestDistance* ← *TotalDistance*(*s*,*DNA*)**
6. ***bestWord* ← *s***
7. **return *bestWord***

Search trees

- Assume $t=2$, $l=2$, $n=5$ ($n-l+1=4$)
 - 2 DNA sequences of length 5, look for 2-mer motif



Analyzing Search Trees

- Characteristics of the 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?
-

Moving through the Search Trees

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

Visit the Next Leaf

Given a current leaf a , we need to compute the “next” leaf:

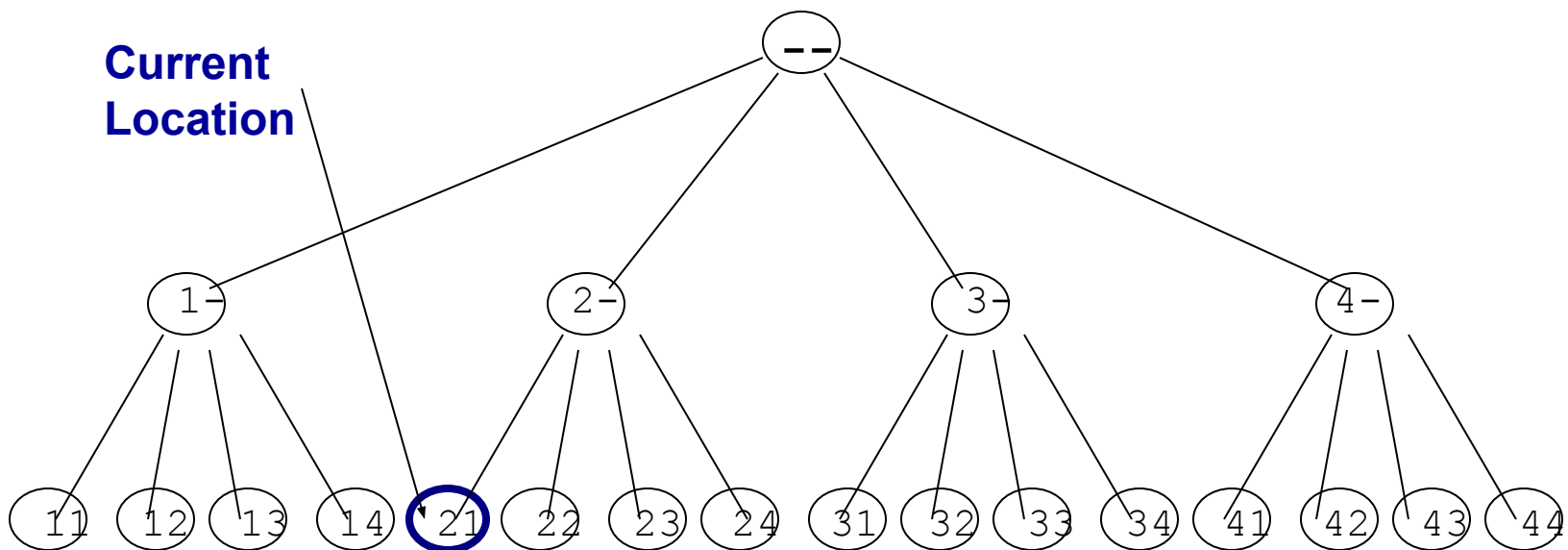
```
1.  NextLeaf(  $\mathbf{a}, L, \mathbf{k}$  )           //  $\mathbf{a}$  : the array of digits
2.  for  $i \leftarrow L$  to 1                //  $L$ : length of the array
3.      if  $a_i < \mathbf{k}$                     //  $\mathbf{k}$  : max digit value
4.           $a_i \leftarrow a_i + 1$ 
5.          return  $\mathbf{a}$ 
6.       $a_i \leftarrow 1$ 
7.  return  $\mathbf{a}$ 
```

NextLeaf (cont'd)

- 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
-

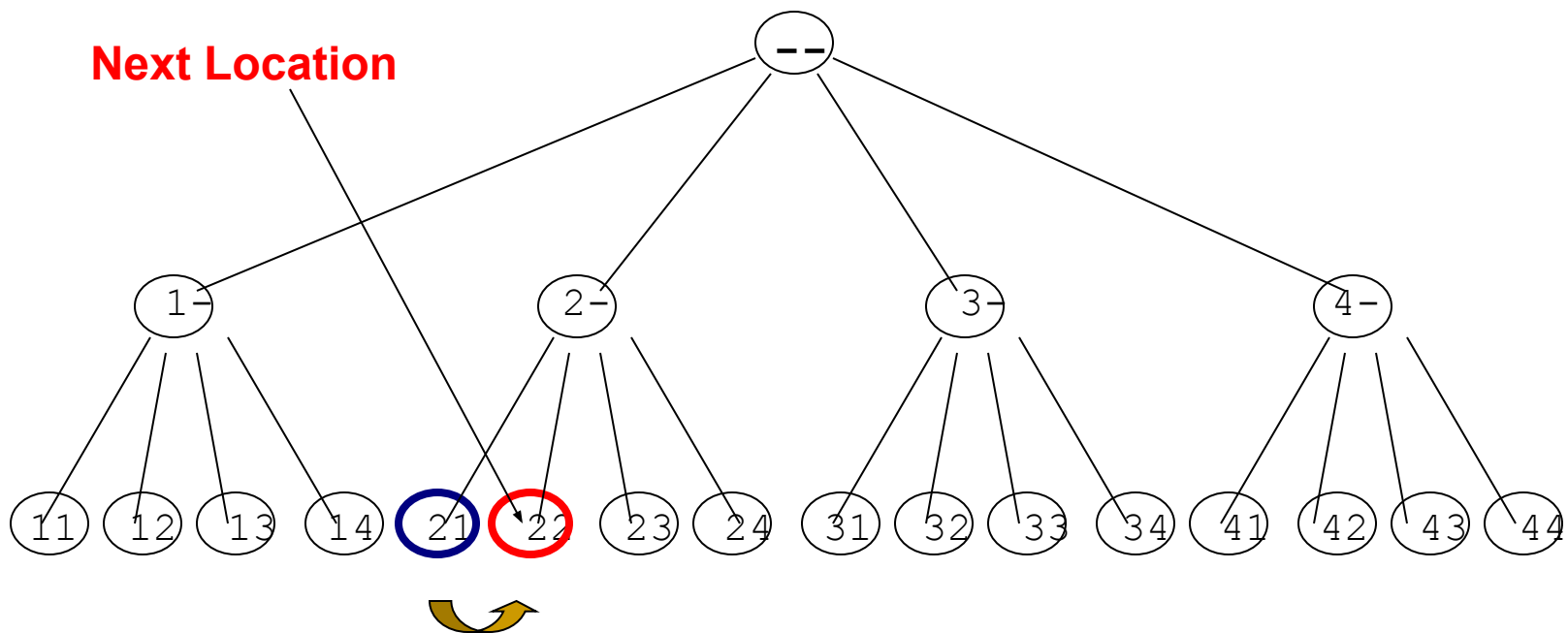
NextLeaf: Example

- Moving to the next leaf:



NextLeaf: Example (cont'd)

- Moving to the next leaf:



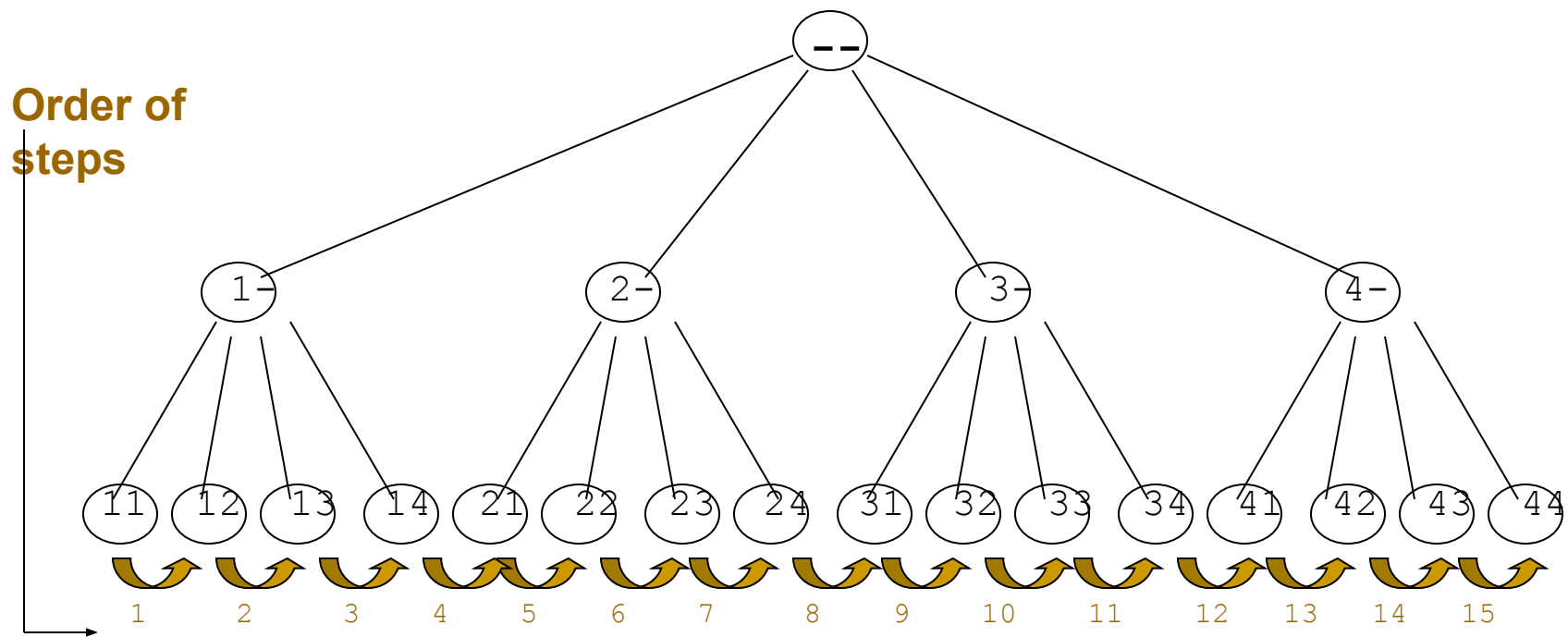
Visit All Leaves

- Printing all permutations in ascending order:

```
1.  AllLeaves(L,k)  // L: length of the sequence
2.  a ← (1,...,1)    // k : max digit value
3.  while forever    // a : array of digits
4.      output a
5.      a ← NextLeaf(a,L,k)
6.      if a = (1,...,1)
7.          return
```

Visit All Leaves: Example

- Moving through all the leaves in order:



Depth First Search

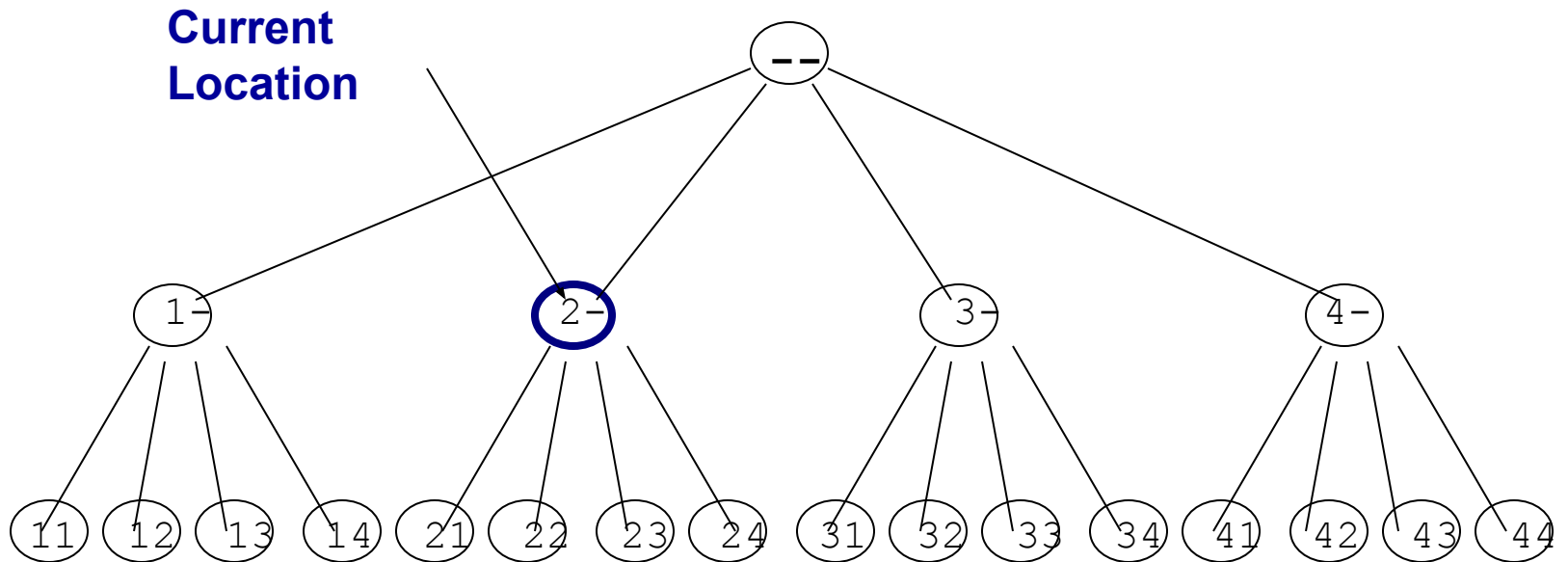
- So we can search leaves
 - How about searching all vertices of the tree?
 - We can do this with a *depth first* search
-

Visit the Next Vertex

```
1.  NextVertex(a,i,L,k)      // a : the array of digits
2.    if  $i < L$                 // i : prefix length
3.       $a_{i+1} \leftarrow 1$       // L: max length
4.      return ( a,i+1)        // k : max digit value
5.    else
6.      for  $j \leftarrow \ell$  to 1
7.        if  $a_j < k$ 
8.           $a_j \leftarrow a_j + 1$ 
9.          return( a,j )
10.   return(a,0)
```

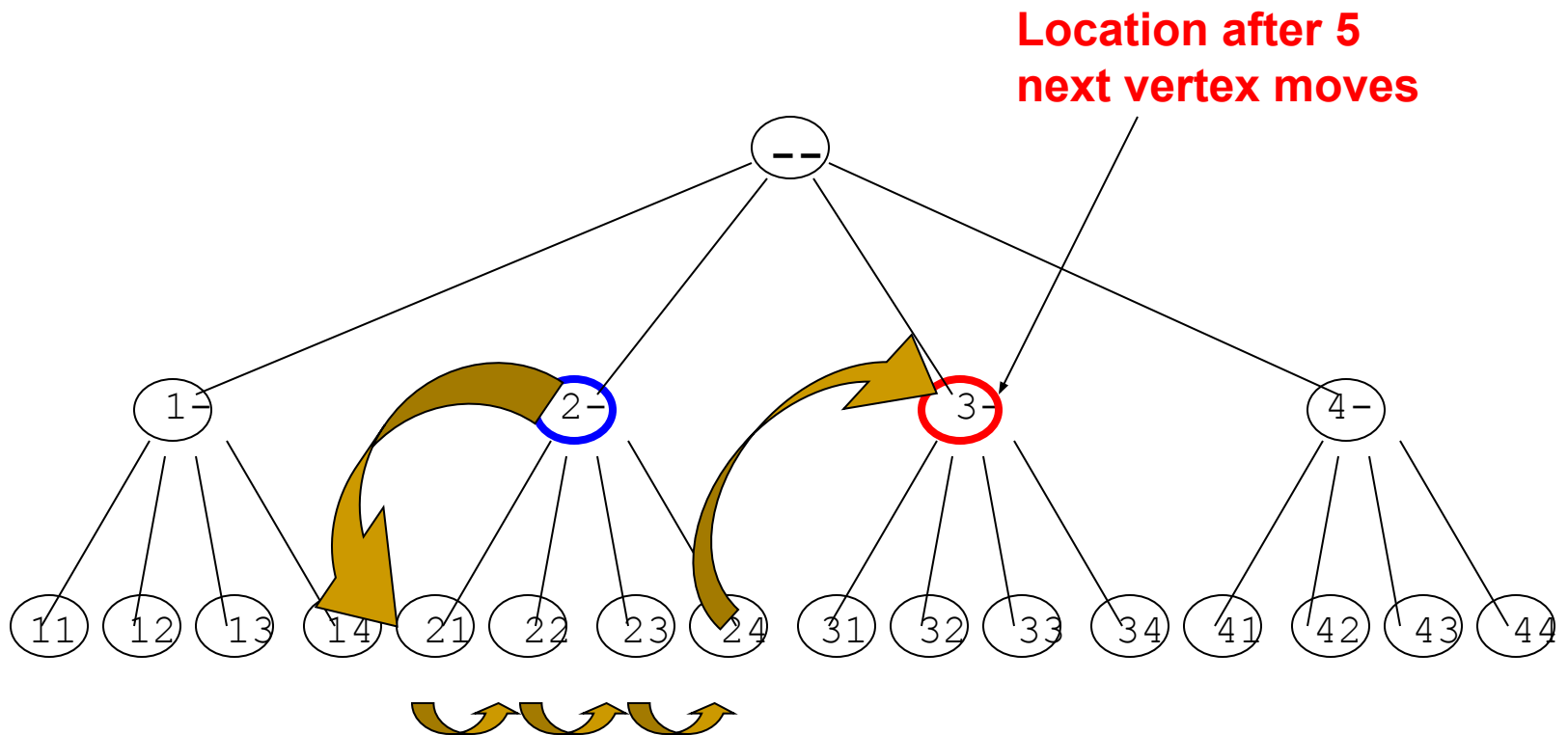
Example

- Moving to the next vertex:



Example

- Moving to the next vertices:



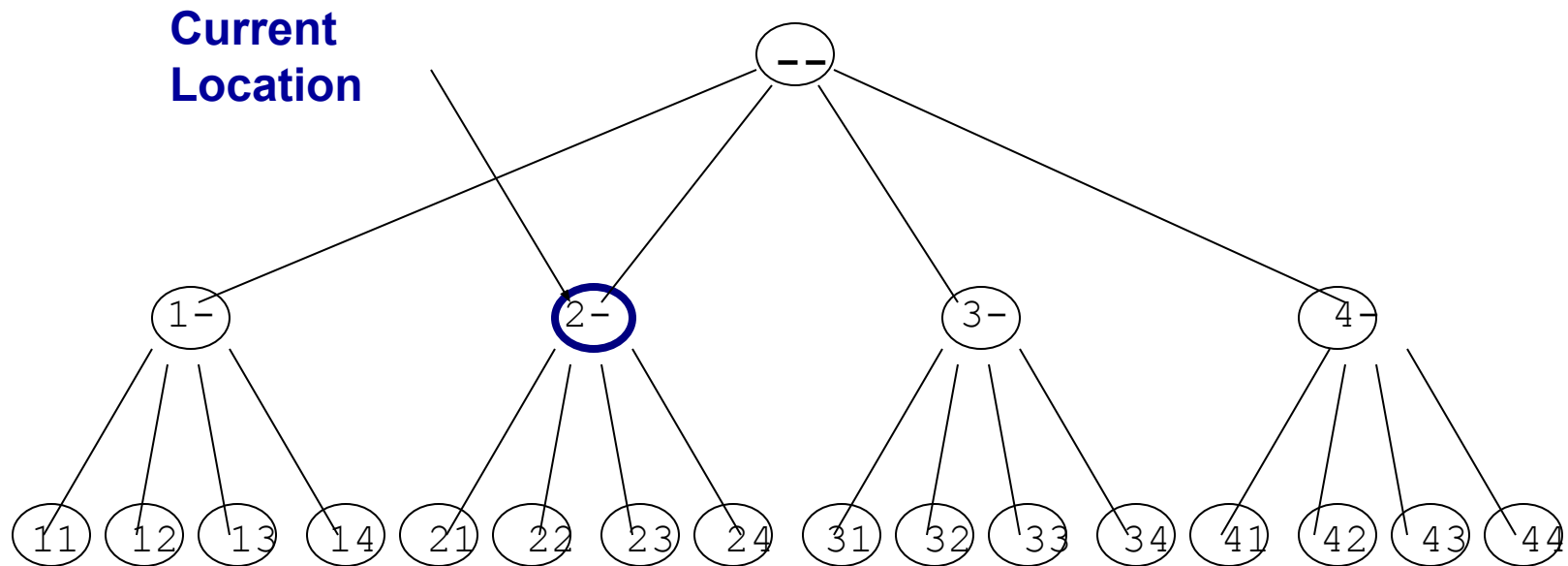
Bypass Move

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

```
1.   Bypass(a,i,L,k)    // a: array of digits
2.   for  $j \leftarrow i$  to 1    // i: prefix length
3.       if  $a_j < k$         // L: maximum length
4.            $a_j \leftarrow a_j + 1$     // k: max digit value
5.       return(a,j)
6.   return(a,0)
```

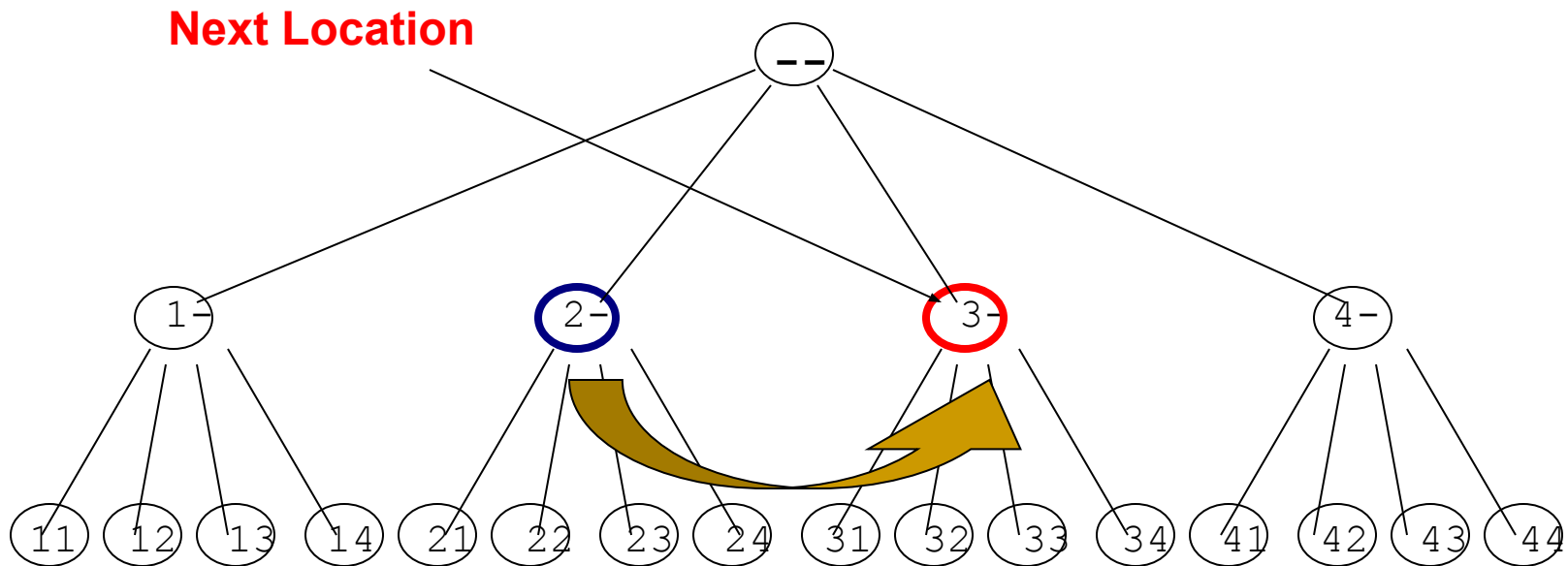
Bypass Move: Example

- Bypassing the descendants of “2-”:



Example

- Bypassing the descendants of “2-”:



Revisiting Brute Force Search

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

Brute Force Search Again

```
1. BruteForceMotifSearchAgain(DNA, t, n,  $\ell$ )
2. s  $\leftarrow$  (1,1,..., 1)
3. bestScore  $\leftarrow$  Score(s,DNA)
4. while forever
5.     s  $\leftarrow$  NextLeaf (s, t, n-  $\ell$  + 1)
6.     if (Score(s,DNA) > bestScore)
7.         bestScore  $\leftarrow$  Score(s, DNA)
8.         bestMotif  $\leftarrow$  ( $s_1, s_2, \dots, s_t$ )
9. return bestMotif
```


Can We Do Better?

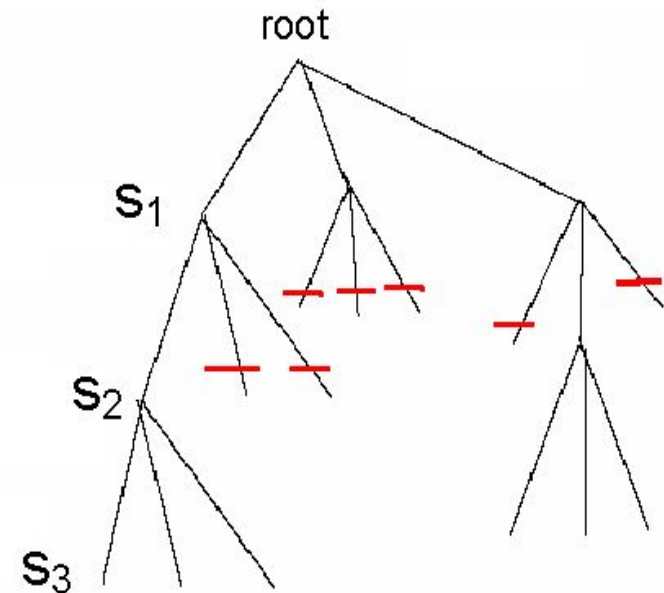
- Sets of $\mathbf{s}=(s_1, s_2, \dots, s_t)$ may have a weak profile for the first i positions (s_1, s_2, \dots, s_i)
- Every row of alignment may add at most ℓ to Score
- Optimism: if all subsequent $(t-i)$ positions (s_{i+1}, \dots, s_t) add

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

- If $\text{Score}(\mathbf{s}, \mathbf{i}, \mathbf{DNA}) + (\mathbf{t} - \mathbf{i}) * \ell < \mathbf{BestScore}$, it makes no sense to search in vertices of the current subtree
 - Use **ByPass()**

Branch and Bound Algorithm for Motif Search

- Since each level of the tree goes deeper into 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

```
1.  BranchAndBoundMotifSearch(DNA, t, n, l)
2.  s  $\leftarrow (1, \dots, 1)$ 
3.  bestScore  $\leftarrow 0$ 
4.  i  $\leftarrow 1$ 
5.  while i > 0
6.      if i < t
7.          optimisticScore  $\leftarrow \text{Score}(\mathbf{s}, i, \mathbf{DNA}) + (t - i) * l$ 
8.          if optimisticScore < bestScore
9.              (s, i)  $\leftarrow \text{Bypass}(\mathbf{s}, i, n - l + 1)$ 
10.         else
11.             (s, i)  $\leftarrow \text{NextVertex}(\mathbf{s}, i, n - l + 1)$ 
12.         else
13.             if Score(s, DNA) > bestScore
14.                 bestScore  $\leftarrow \text{Score}(\mathbf{s})$ 
15.                 bestMotif  $\leftarrow (s_1, s_2, s_3, \dots, s_t)$ 
16.                 (s, i)  $\leftarrow \text{NextVertex}(\mathbf{s}, i, t, n - l + 1)$ 
17.  return bestMotif
```

Structuring the Search: median string

- For the Median String Problem we need to consider all 4^l possible l -mers:

$\overbrace{\text{aa... aa}}^l$
aa... ac
aa... ag
aa... at
.
.
tt... tt

How to organize this search?

Alternative Representation of the Search Space

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

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

- Notice that the sequences above simply list all numbers as if we were counting on base 5 without using 0 as a digit

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^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 Applied to Median String Search

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

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

there is no use exploring the remaining part of the word

- We can eliminate that branch and BYPASS exploring that branch further
-

Bounded Median String Search

```
1. BranchAndBoundMedianStringSearch(DNA, t, n, l)
2. s  $\leftarrow$  (1,...,1)
3. bestDistance  $\leftarrow$   $\infty$ 
4. i  $\leftarrow$  1
5. while i > 0
6.   if i < l
7.     prefix  $\leftarrow$  string corresponding to the first i nucleotides of s
8.     optimisticDistance  $\leftarrow$  TotalDistance(prefix, DNA)
9.     if optimisticDistance > bestDistance
10.      (s, i)  $\leftarrow$  Bypass(s, i, l, 4)
11.   else
12.     (s, i)  $\leftarrow$  NextVertex(s, i, l, 4)
13.   else
14.     word  $\leftarrow$  nucleotide string corresponding to s
15.     if TotalDistance(s, DNA) < bestDistance
16.       bestDistance  $\leftarrow$  TotalDistance(word, DNA)
17.       bestWord  $\leftarrow$  word
18.     (s, i)  $\leftarrow$  NextVertex(s, i, l, 4)
19. return bestWord
```

previously: AAAAA, AAAAC, AAAAG,....

bestmotif = AACTA bestdist = 34

Search: ACGTA k=5

A totDist = 0

AC totDist = 23

ACG totDist = 55 X

do not search ACGT or ACGTA or ACGA or
ACGG or

Improving the Bounds

- Given an ℓ -mer \mathbf{w} , divided into two parts at point i
 - ▣ \mathbf{u} : prefix w_1, \dots, w_i
 - ▣ \mathbf{v} : suffix w_{i+1}, \dots, w_ℓ
- Find minimum distance for \mathbf{u} in a sequence
- No instances of \mathbf{u} in the sequence have distance less than the minimum distance
- Note this doesn't tell us anything about whether \mathbf{u} is part of any motif. We only get a minimum distance for prefix \mathbf{u}

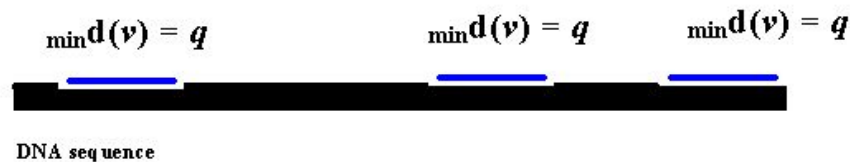
Improving the Bounds (cont'd)

- Repeating the process for the suffix \mathbf{v} gives us a minimum distance for \mathbf{v}
- Since \mathbf{u} and \mathbf{v} are two substrings of \mathbf{w} , and included in motif \mathbf{w} , we can assume that the minimum distance of \mathbf{u} plus minimum distance of \mathbf{v} can only be less than the minimum distance for \mathbf{w}

Better Bounds

Searching for prefix V

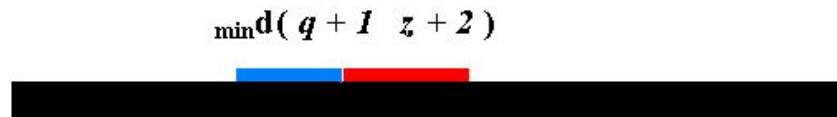
We may find many instances of prefix V with a minimum distance q



Likewise for U



But for U and V combined, U is not at its minimum distance location, neither is V



But at least we know w (prefix u suffix v) cannot have distance *less* than $\min d(v) + \min d(u)$

Better Bounds (cont'd)

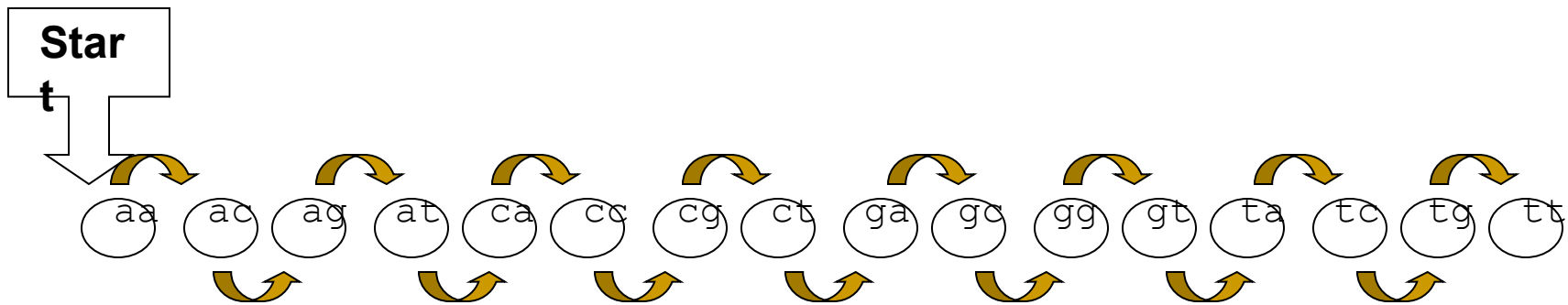
- If $d(\textit{prefix}) + d(\textit{suffix}) \geq \textit{bestDistance}$:
 - Motif w ($\textit{prefix.suffix}$) cannot give a better (lower) score than $d(\textit{prefix}) + d(\textit{suffix})$
 - In this case, we can **ByPass()**

Better Bounded Median String Search

```
1. ImprovedBranchAndBoundMedianString(DNA.t.n.l)
2.   s = (1, 1, ..., 1)
3.   bestdistance =  $\infty$ 
4.   i = 1
5.   while i > 0
6.     if i < l
7.       prefix = nucleotide string corresponding to (s1, s2, s3, ..., si)
8.       optimisticPrefixDistance = TotalDistance (prefix, DNA)
9.       if (optimisticPrefixDistance < bestsubstring[ i ])
10.        bestsubstring[ i ] = optimisticPrefixDistance
11.        if (l - i < i)
12.          optimisticSufxDistance = bestsubstring[l - i]
13.        else
14.          optimisticSufxDistance = 0;
15.        if optimisticPrefixDistance + optimisticSufxDistance  $\geq$  bestDistance
16.          (s, i) = Bypass(s, i, l, 4)
17.        else
18.          (s, i) = NextVertex(s, i, l, 4)
19.      else
20.        word = nucleotide string corresponding to (s1, s2, s3, ..., sl)
21.        if TotalDistance( word, DNA) < bestDistance
22.          bestDistance = TotalDistance(word, DNA)
23.          bestWord = word
24.          (s, i) = NextVertex(s, i, l, 4)
25.   return bestWord
```

Linked List

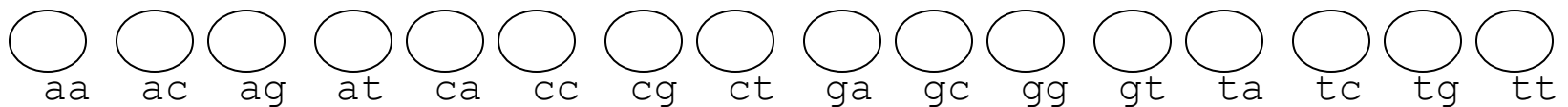
- Suppose $l=2$



- Need to visit all the predecessors of a sequence before visiting the sequence itself

Linked List (cont'd)

- Linked list is not the most efficient data structure for motif finding
- Let's try grouping the sequences by their prefixes



Search Tree

