Finding Regulatory Motifs in DNA Sequences

Saeedeh Akbari

Department of Computer Engineering

Sharif University of Technology

Fall 2023

Adapted with modifications from lecture notes prepared by Phillip Compeau

Bioinformatics Algorithms: An Active LearningApproach

Outline

- **Motif finding problem**
- Implanting Patterns in Random Text
- The Gold Bug Problem
- The Motif Finding Problem
- Brute Force Motif Finding
- The Median String Problem
- Search Trees
- Branch-and-Bound Motif Search
- Branch-and-Bound Median String Search

What is a DNA Motif?

- DNA motifs are short, recurring patterns that are presumed to have a biological function.
  - As regulatory motifs may vary at some positions
- Every plant cell keeps track of day and night independently of other cells
  - e.g. three plant genes, called LCY, CCA1, and TOC1, are the clock's master timekeepers
  - Such regulatory genes, and the regulatory proteins that they encode, are often controlled by external factors (e.g., nutrient availability or sunlight) in order to allow organisms to adjust their gene expression

Identifying Motifs: Complications

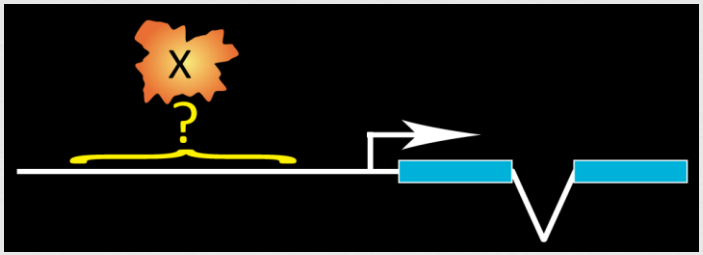
- We do not know the motif sequence
- We do not know where it is located relative to the genes start
- Motifs can differ slightly from one gene to the next
- How to discern it from "random" motifs?

Two Methods

Pattern Matching

Finding known motifs

- Does protein X bind upstream of my genes?

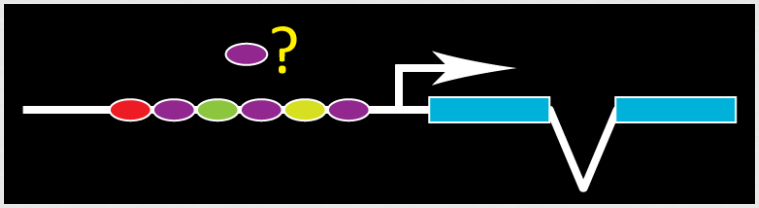


e.g. Patser, Pscan, Mast..

Pattern Discovery

Finding unknown motifs

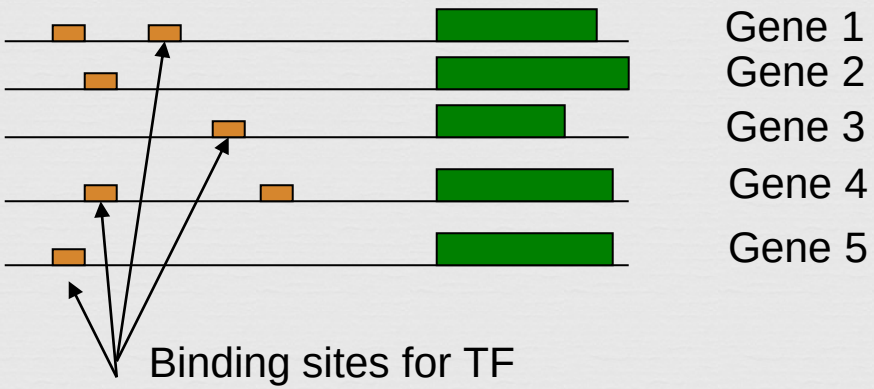
- What motifs are upstream of my genes?



e.g. MEME, Weeder, MDScan ...

Motif Finding Problem

- Suppose a transcription factor (TF) controls five different genes
- Each of the five genes should have binding sites for TF in their promoter region



Motif Finding Problem

- Now suppose we are given the promoter regions of the five genes G1, G2, … G5
- Can we find the binding sites of TF, without knowing about them *a priori* ?
  - Binding sites are similar to each other, but not necessarily identical
- This is the motif finding problem
- To find a motif that represents binding sites of an unknown TF

Identifying Motifs: Complications

- We do not know the motif sequence
  - May know its length
- We do not know where it is located relative to the genes start
- Motifs can differ slightly from one gene to the next
  - Non-essential bases could mutate…
- How to discern functional motifs from random ones?

Outline

- Motif finding problem
- **Implanting Patterns in Random Text**
- The Gold Bug Problem
- The Motif Finding Problem
- Brute Force Motif Finding
- The Median String Problem
- Search Trees
- Branch-and-Bound Motif Search
- Branch-and-Bound Median String Search

Random Sample

atgaccgggatactgataccgtatttggcctaggcgtacacattagataaacgtatgaagtacgttagactcggcgccgccg

acccctatttttgagcagatttagtgacctggaaaaaaaatttgagtacaaaacttttccgaatactgggcataaggtaca

tgagtatccctgggatgacttttgggaacactatagtgctctcccgatttttgaatatgtaggatcattcgccagggtccga

gctgagaattggatgaccttgtaagtgttttccacgcaatcgcgaaccaacgcggacccaaaggcaagaccgataaaggaga

tccctttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaatggcccacttagtccacttatag

gtcaatcatgttcttgtgaatggatttttaactgagggcatagaccgcttggcgcacccaaattcagtgtgggcgagcgcaa

cggttttggcccttgttagaggcccccgtactgatggaaactttcaattatgagagagctaatctatcgcgtgcgtgttcat

aacttgagttggtttcgaaaatgctctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgta

ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcatttcaacgtatgccgaaccgaaagggaag

ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagcttctgggtactgatagca

Implanting Motif **AAAAAAAAGGGGGGGG**

atgaccgggatactgat**AAAAAAAAGGGGGGGG**ggcgtacacattagataaacgtatgaagtacgttagactcggcgccgccg

acccctatttttgagcagatttagtgacctggaaaaaaaatttgagtacaaaacttttccgaata**AAAAAAAAGGGGGGGG**a

tgagtatccctgggatgactt**AAAAAAAAGGGGGGGG**tgctctcccgattttgaatatgtaggatcattcgccagggtccga

gctgagaattggatg**AAAAAAAAGGGGGGGG**tccacgcaatcgcgaaccaacgcggacccaaaggcaagaccgataaaggaga

tccctttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaat**AAAAAAAAGGGGGGGG**cttatag

gtcaatcatgttcttgtgaatggattt**AAAAAAAAGGGGGGGG**gaccgcttggcgcacccaaattcagtgtgggcgagcgcaa

cggttttggcccttgttagaggcccccgt**AAAAAAAAGGGGGGGG**caattatgagagagctaatctatcgcgtgcgtgttcat

aacttgagtt**AAAAAAAAGGGGGGGG**ctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgta

ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcat**AAAAAAAAGGGGGGGG**accgaaagggaag

ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagctt**AAAAAAAAGGGGGGGG**a

Implanting Motif **AAAAAAGGGGGGG** with Four Mutations

atgaccgggatactgatAgAAgAAAGGttGGGggcgtacacattagataaacgtatgaagtacgttagactcggcgccgccg

acccctatttttgagcagatttagtgacctggaaaaaaaatttgagtacaaaacttttccgaatacAAtAAAAcGGcGGGa

tgagtatccctgggatgacttAAAAtAAtGGaGtGGtgctctcccgattttgaatatgtaggatcattcgccagggtccga

gctgagaattggatgcAAAAAAAGGGattGtccacgcaatcgcgaaccaacgcggacccaaaggcaagaccgataaaggaga

tccctttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaatAtAAtAAAGGaaGGGcttatag

gtcaatcatgttcttgtgaatggatttAAcAAtAAGGGctGGgaccgcttggcgcacccaaattcagtgtgggcgagcgcaa

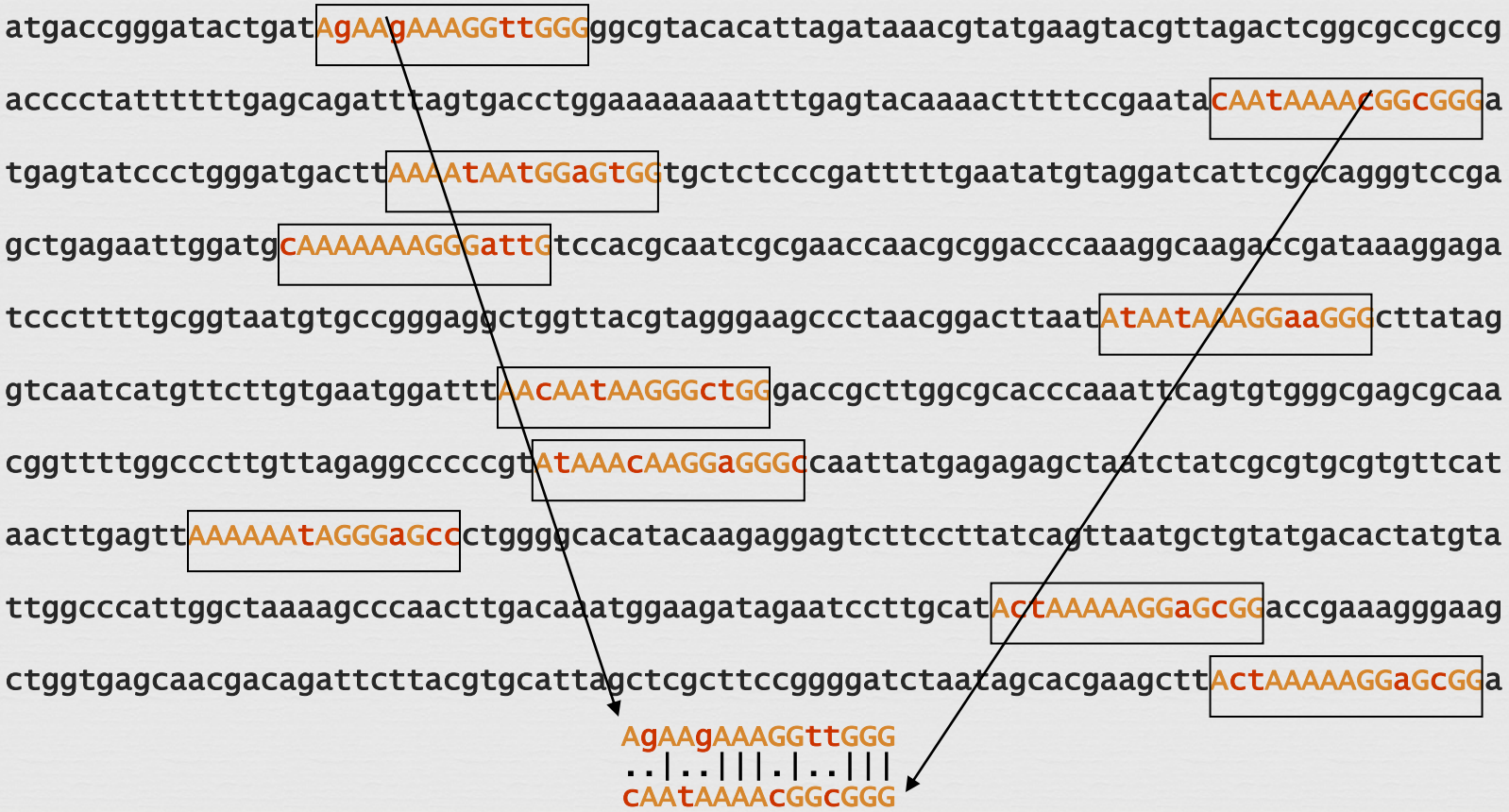
cggttttggcccttgttagaggcccccgtAtAAAcAAGGaGGGccaattatgagagagctaatctatcgcgtgcgtgttcat

aacttgagttAAAAAAtAGGGaGccctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgta

ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcatActAAAAAGGaGcGGaccgaaagggaag

ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagcttActAAAAAGGaGcGGa

Why Finding (15,4) Motif is Difficult?



Challenge Problem

- Find 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

- What is Motif
- Motif finding problem
- Implanting Patterns in Random Text
- **The Gold Bug Problem**
- The Motif Finding Problem
- Brute Force Motif Finding
- The Median String Problem
- Search Trees
- Branch-and-Bound Motif Search
- Branch-and-Bound Median String Search

A Motif Finding Analogy

- The Motif Finding Problem is similar to the problem posed by Edgar Allan Poe (1809 – 1849) in his *Gold Bug* story

The Gold Bug Problem

- Given a secret message:

```
53++!305))6*;4826)4+.)4+);806*;48!8`60))85;]8*:+*8!83(88)5*!;
46(;88*96*?;8)*+(;485);5*!2:*+(;4956*2(5*-4)8`8*; 4069285);)6
!8)4++;1(+9;48081;8:8+1;48!85;4)485!528806*81(+9;48;(88;4(+?3
4;48)4+;161;:188;+?;
```

- Decipher the message encrypted in the fragment

Hints for The Gold Bug Problem

- Additional hints:
  - The encrypted message is in English
  - Each symbol correspond to one letter in the English alphabet
  - No punctuation marks are encoded

The Gold Bug Problem: Symbol Counts

- Naive approach to solving the problem:
  - Count the frequency of each symbol in the encrypted message
  - Find the frequency of each letter in the alphabet in the English language
  - Compare the frequencies of the previous steps, try to find a correlation and map the symbols to a letter in the alphabet

Symbol Frequencies in the Gold Bug Message

Gold Bug Message:

Symbol	8	;	4	)	+	*	5	6	(	!	1	0	2	9	3	:	?	`	-	]	.
Frequency	34	25	19	16	15	14	12	11	9	8	7	6	5	5	4	4	3	2	1	1	1

English Language:

e t a o i n s r h l d c u m f p g w y b v k x j q z

Most frequent → Least frequent

The Gold Bug Message Decoding: First Attempt

- By simply mapping the most frequent symbols to the most frequent letters of the alphabet:

```
sfiilfcsoorntaeuroaikoaiotecrntaeleyrcooestvenpinelefheeosnlt
arhteenmrnwteonihtaesotsnlupnihtamsrnuhsnbaoeyentacrmuesotorl
eoaiitdhimtaecedtepeidtaelestaoaeslsueecrnedhimtaetheetahiwfa
taeoaitdrdtpdeetiwt
```

- The result does not make sense

The Gold Bug Problem: *l*-tuple count

- A better approach:
  - Examine frequencies of *l*-tuples, combinations of 2 symbols, 3 symbols, etc.
  - "The" is the most frequent 3-tuple in English and ";48" is the most frequent 3-tuple in the encrypted text
  - Make inferences of unknown symbols by examining other frequent *l*-tuples

The Gold Bug Problem: the ;48 clue

- Mapping "the" to ";48" and substituting all occurrences of the symbols:

```
53++!305))6*the26)h+.)h+)te06*the!e`60))e5t]e*:+*e!e3(ee)5
*!t
h6(tee*96*?te)*+(the5)t5*!2:*+(th956*2(5*h)e`e*th0692e5)t)
6!e
)h++t1(+9the0e1te:e+1the!e5th)he5!52ee06*e1(+9thet(eeth(+?
3ht
he)h+t161t:1eet+?t
```

The Gold Bug Message Decoding: Second Attempt

- Make inferences:

```
53++!305))6*the26)h+.)h+)te06*the!e`60))e5t]e*:+*e!e3(ee)5*!t
h6(tee*96*?te)*+(the5)t5*!2:*+(th956*2(5*h)e`e*th0692e5)t)6!e
)h++t1(+9the0e1te:e+1the!e5th)he5!52ee06*e1(+9thet(eeth(+?3ht
he)h+t161t:1eet+?t
```

- "thet(ee" most likely means "the tree"
  - Infer "(" = "r"
- "th(+?3h" becomes "thr+?3h"
  - Can we guess "+" and "?"?

The Gold Bug Problem: The Solution

- After figuring out all the mappings, the final message is:

AGOODGLASSINTHEBISHOPSHOSTELINTHEDEVILSSEATWENYONEDEGRE
ESANDTHIRTEENMINUTESNORTHEASTANDBYNORTHMAINBRANCHSEVENT
HLIMBEASTSIDESHOOTFROMTHELEFTEYEOFTHEDEATHSHEADABEELINE
FROMTHETREETHROUGHTHESHOTFIFTYFEETOUT

The Solution (cont'd)

- Punctuation is important:

A GOOD GLASS IN THE BISHOP'S HOSTEL IN THE DEVIL'S SEA,
TWENY ONE DEGREES AND THIRTEEN MINUTES NORTHEAST AND BY NORTH,
MAIN BRANCH SEVENTH LIMB, EAST SIDE, SHOOT FROM THE LEFT EYE OF
THE DEATH'S HEAD A BEE LINE FROM THE TREE THROUGH THE SHOT,
FIFTY FEET OUT.

Solving the Gold Bug Problem

- Prerequisites to solve the problem:
  - Need to know the relative frequencies of single letters, and combinations of two and three letters in English
  - Knowledge of all the words in the English dictionary is highly desired to make accurate inferences

Motif Finding and The Gold Bug Problem: Similarities

- **Motif Finding**:
  - In order to solve the problem, we analyze the frequencies of patterns in the nucleotide sequences
  - Knowledge of established regulatory motifs makes the Motif Finding problem simpler.

- **Gold Bug Problem**:
  - In order to solve the problem, we analyze the frequencies of patterns in the text written in English
  - Knowledge of the words in the English dictionary helps to solve the Gold Bug problem.

Outline

- Motif finding problem
- Implanting Patterns in Random Text
- The Gold Bug Problem
- **The Motif Finding Problem**
- Brute Force Motif Finding
- The Median String Problem
- Search Trees
- Branch-and-Bound Motif Search
- Branch-and-Bound Median String Search

Motif Finding and The Gold Bug Problem: Differences

Motif Finding is harder than **Gold Bug problem**:

- We don't have the complete dictionary of motifs
- The "genetic" language does not have a standard "grammar"
- Only a small fraction of nucleotide sequences encode for motifs; the size of data is enormous

The Motif Finding Problem

- Given a random sample of DNA sequences:

```
cctgatagacgctatctggctatccacgtacgtaggtcctctgtgcgaatctatgcgtttccaaccat
agtactggtgtacatttgatacgtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc
aaacgtacgtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt
agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatcttacgtacgtataca
ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgttaacgtacgtc
```

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

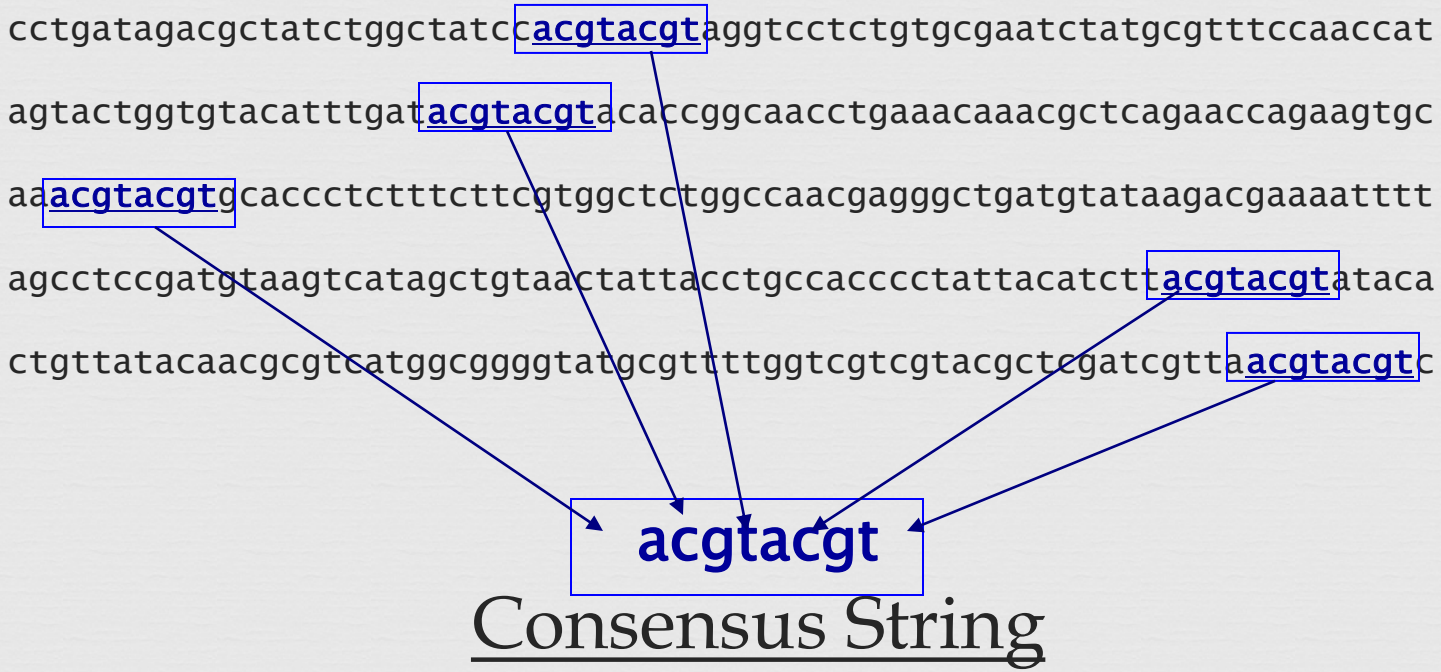
The Motif Finding Problem (cont'd)

- 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:

cctgatagacgctatctggctatccaGgtacTtaggtcctctgtgcgaatctatgcgtttccaaccat

agtactggtgtacatttgatCcAtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc

aaacgtTAgtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt

agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatcttacgtCcAtataca

ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgttaCcgtacgGc

The Motif Finding Problem (cont'd)

- The patterns with 2 point mutations:

cctgatagacgctatctggctatcc**aGgtacTt**aggtcctctgtgcgaatctatgcgtttccaaccat

agtactggtgtacatttgat**CcAtacgt**acaccggcaacctgaaacaaacgctcagaaccagaagtgc

aa**acgtTAgt**gcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt

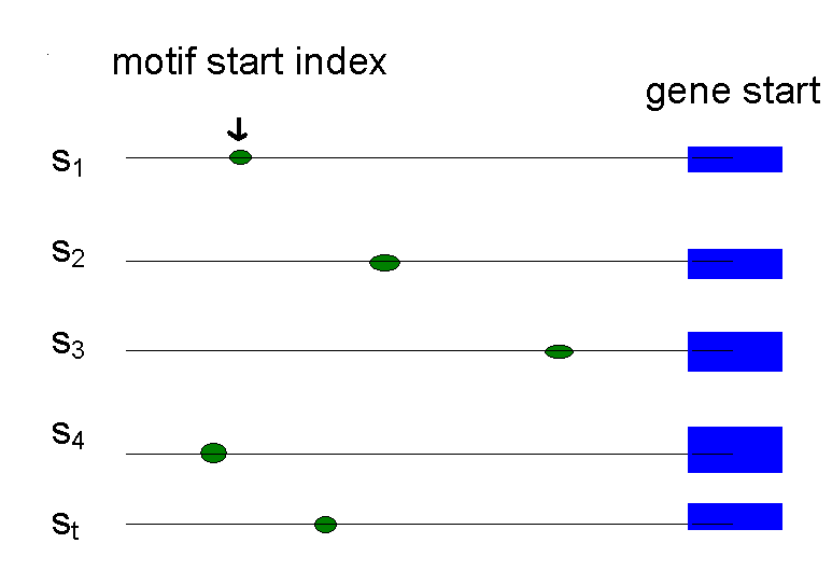
agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatctt**acgtCcAt**ataca

ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgtta**CcgtacgG**c

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

Defining Motifs

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



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

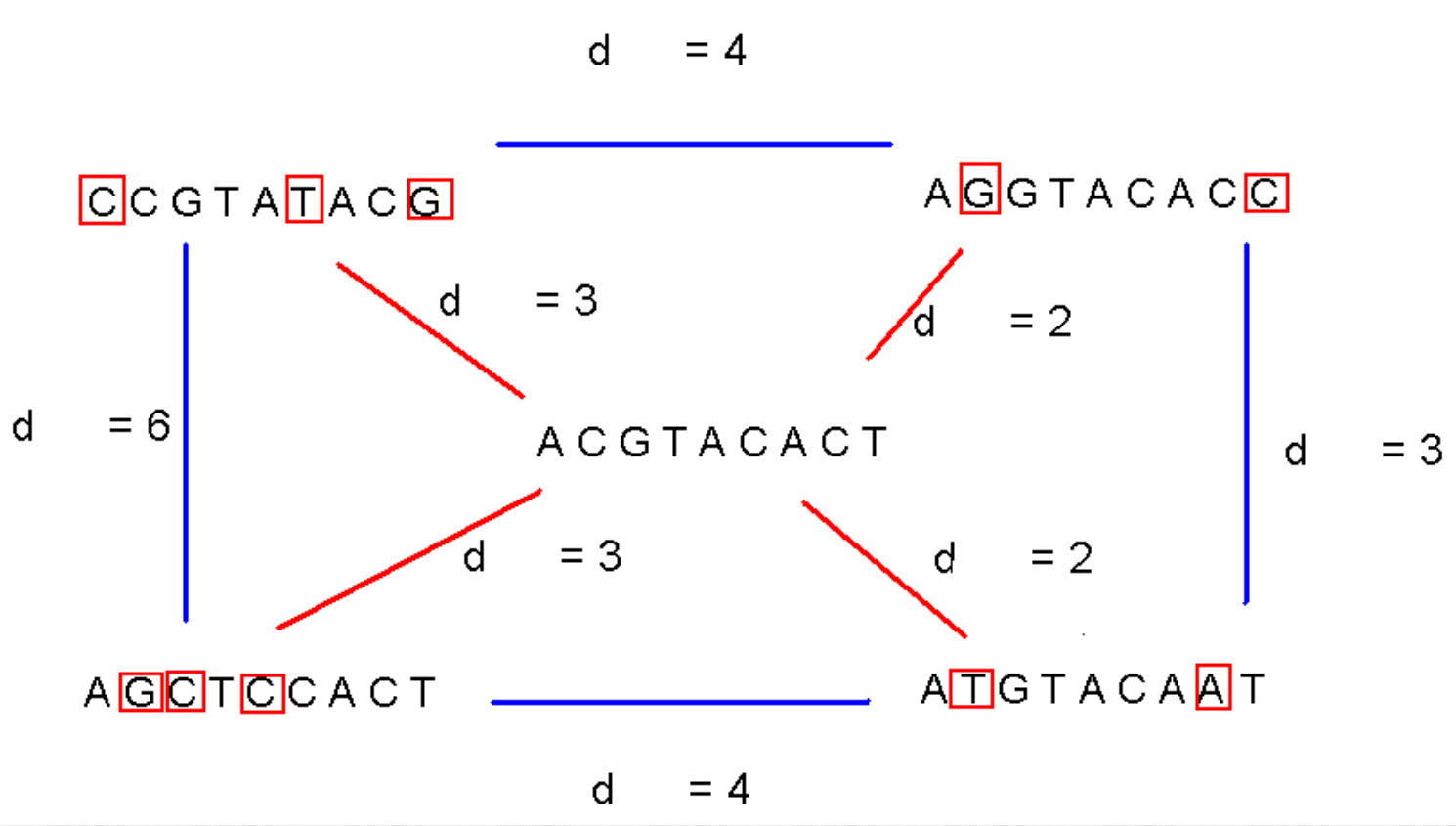
$$\mathbf{s} = (s_1, s_2, \ldots, s_t)$$

- Construct matrix profile 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?

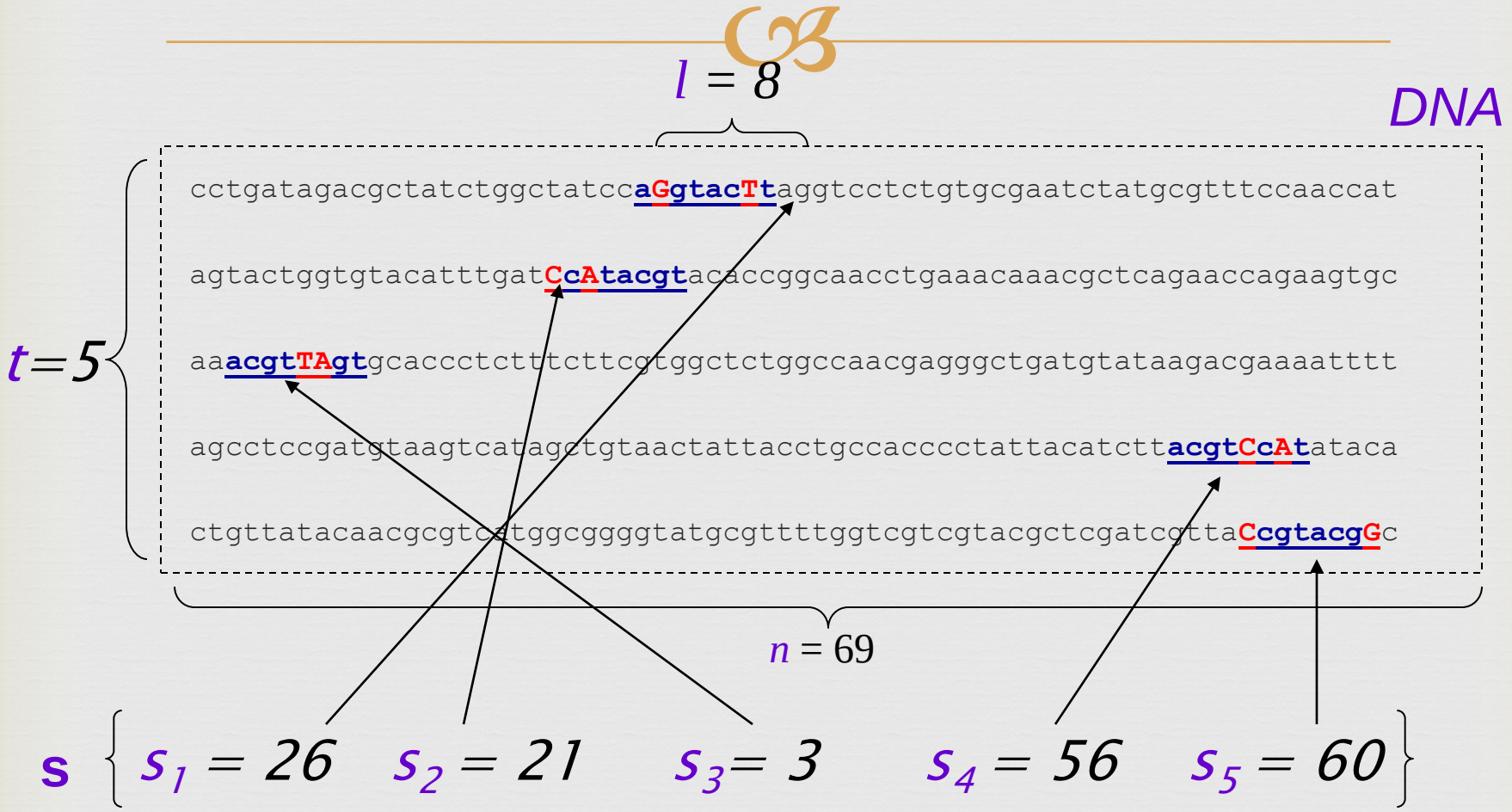
- Need to introduce a scoring function to compare different guesses and choose the "best" one.

Defining Some Terms

- t - number of sample DNA sequences
- n - length of each DNA sequence
- DNA - sample of DNA sequences (t x n array)

- l - length of the motif (l-mer)
- s_i - starting position of an l-mer in sequence i
- $\mathbf{s}=(s_1, s_2, \ldots s_t)$ - array of motif's starting positions

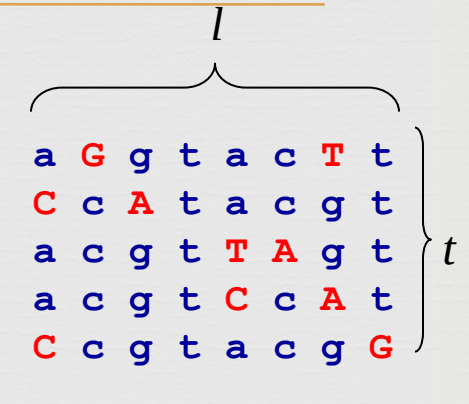
Parameters


l = 8
DNA
cctgatagacgctatctggctatccaGgtacTtaggtcctctgtgcgaatctatgcgtttccaaccat
agtactggtgtacatttgatCcAtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc
aaacgtTAgtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt
agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatcttacgtCcAtataca
ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgttaCcgtacgGc
t=5
n = 69
s
s1 = 26
s2 = 21
s3 = 3
s4 = 56
s5 = 60

Scoring Motifs

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

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



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 **s**=(s_1, s_2,… 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 l-mers, one from each sequence, that maximizes the consensus score

- Input: A t x n matrix of DNA, and l, the length of the pattern to find

- Output: An array of t starting positions $\mathbf{s} = (s_1, s_2, \ldots s_t)$ maximizing $Score(\mathbf{s}, DNA)$

The Matrix

- A position weight matrix (PWM)
  - also called position-specific weight matrix (PSWM)
  - also called position-frequency matrix (PFM)
  - also called position-specific scoring matrix (PSSM)
  - or just matrix
- There is a matrix element for all possible bases at every position.

	1	2	3	4	5	6	7	8	9	10	11
A	4	13	5	3	0	0	0	0	17	0	6
C	4	1	2	0	0	0	0	0	0	1	0
G	3	3	0	0	18	0	0	0	1	4	3
T	7	1	11	15	0	18	18	18	0	13	9

Pattern Matching

Counts

A	4	13	5	3	0	0	0	0	17	0	6
C	4	1	2	0	0	0	0	0	0	1	0
G	3	3	0	0	18	0	0	0	1	4	3
T	7	1	11	15	0	18	18	18	0	13	9

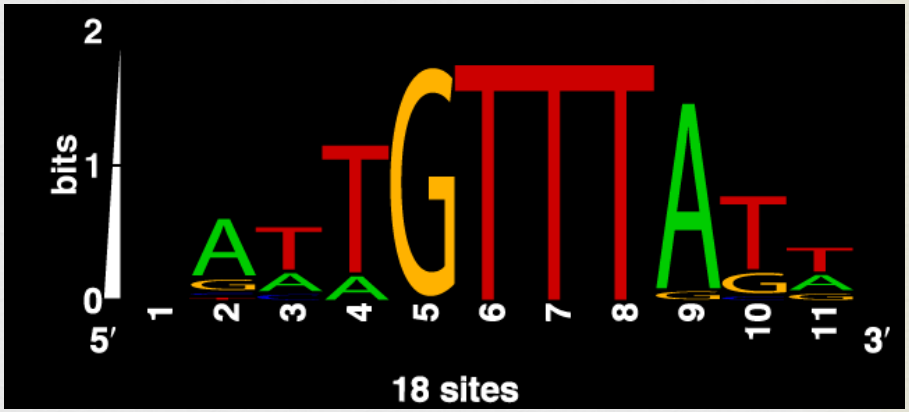
Frequency

A	0.2	0.7	0.3	0.2	0.0	0.0	0.0	0.0	0.9	0.0	0.3
C	0.2	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0
G	0.2	0.2	0.0	0.0	1.0	0.0	0.0	0.0	0.1	0.2	0.2
T	0.4	0.1	0.6	0.8	0.0	1.0	1.0	1.0	0.0	0.7	0.5

Motif Logos

- A visual representation of the motif
- Each column of the matrix is represented as a stack of letters whose size is proportional to the corresponding residue frequency
- The total height of each column is proportional to its information content.

A	4	13	5	3	0	0	0	0	17	0	6
C	4	1	2	0	0	0	0	0	0	1	0
G	3	3	0	0	18	0	0	0	1	4	3
T	7	1	11	15	0	18	18	18	0	13	9



Motifs	T	C	G	G	G	G	g	T	T	T	t	t
	c	C	G	G	t	G	A	c	T	T	a	C
	a	C	G	G	G	G	A	T	T	T	t	C
	T	t	G	G	G	G	A	c	T	T	t	t
	a	a	G	G	G	G	A	c	T	T	C	C
	T	t	G	G	G	G	A	c	T	T	C	C
	T	C	G	G	G	G	A	T	T	c	a	t
	T	C	G	G	G	G	A	T	T	c	C	t
	T	a	G	G	G	G	A	a	c	T	a	C
	T	C	G	G	G	t	A	T	a	a	C	C

SCORE(*Motifs*) $3 + 4 + 0 + 0 + 1 + 1 + 1 + 5 + 2 + 3 + 6 + 4 = 30$

COUNT(*Motifs*)

A:	2	2	0	0	0	0	9	1	1	1	3	0
C:	1	6	0	0	0	0	0	4	1	2	4	6
G:	0	0	10	10	9	9	1	0	0	0	0	0
T:	7	2	0	0	1	1	0	5	8	7	3	4

PROFILE(*Motifs*)

A:	.2	.2	0	0	0	0	.9	.1	.1	.1	.3	0
C:	.1	.6	0	0	0	0	0	.4	.1	.2	.4	.6
G:	0	0	1	1	.9	.9	.1	0	0	0	0	0
T:	.7	.2	0	0	.1	.1	0	.5	.8	.7	.3	.4

CONSENSUS(*Motifs*) T C G G G G A T T T C C

Outline

- Motif finding problem
- Implanting Patterns in Random Text
- The Gold Bug Problem
- The Motif Finding Problem
- **Brute Force Motif Finding**
- The Median String Problem
- Search Trees
- Branch-and-Bound Motif Search
- Branch-and-Bound Median String Search

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:

$$s_i = [1, \ldots, n-l+1]$$
$$i = [1, \ldots, t]$$

BruteForceMotifSearch

1. BruteForceMotifSearch(***DNA, t, n, l***)
2. ***bestScore*** ← 0
3. **for** each **s**=(s_1,s_2 , . . ., s_t) from (1,1 . . . 1)
   to (***n***−***l***+1, . . ., ***n***−***l***+1)
4. **if** (*Score*(**s**,***DNA***) > ***bestScore***)
5. ***bestScore*** ← *score*(**s**, ***DNA***)
6. **bestMotif** ← (s_1,s_2 , . . . , 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 – it will take billions years

Outline

- Motif finding problem
- Implanting Patterns in Random Text
- The Gold Bug Problem
- The Motif Finding Problem
- Brute Force Motif Finding
- **The Median String Problem**
- Search Trees
- Branch-and-Bound Motif Search
- Branch-and-Bound Median String Search

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:
  - $d_H(v,w)$ 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: Definition

- For each DNA sequence i, compute all $d_H(v, x)$, where x is an l-mer with starting position s_i

  $(1 \leq s_i \leq n - l + 1)$

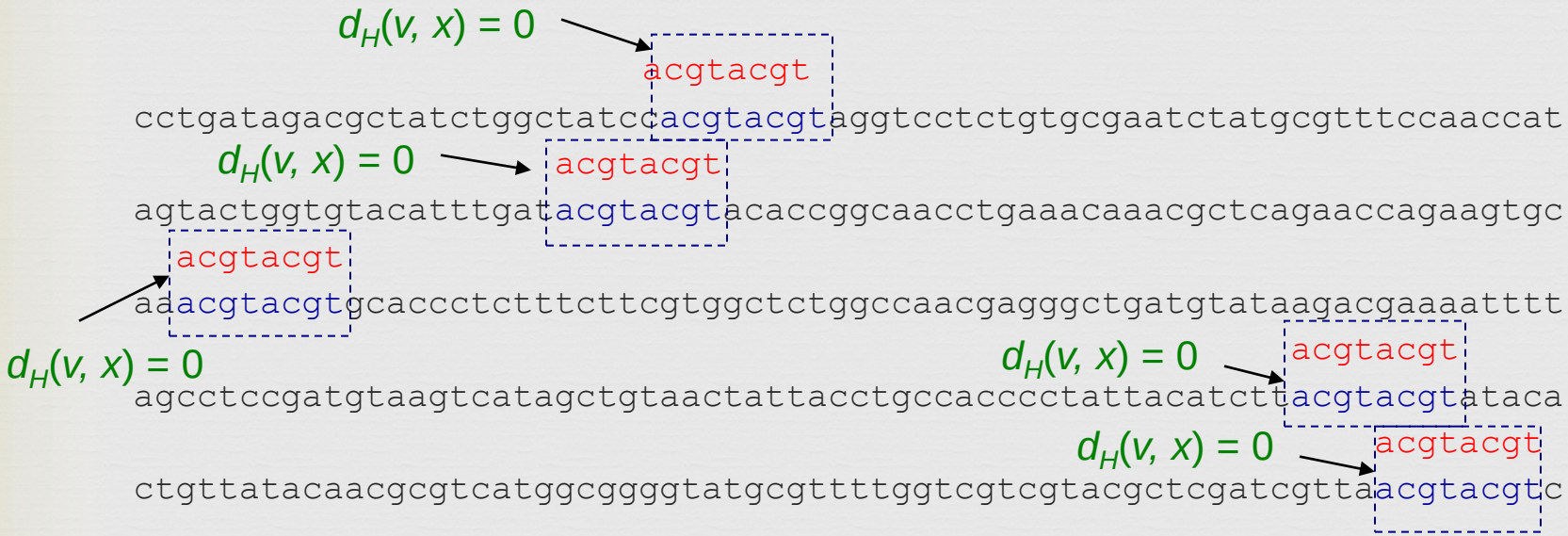
- Find minimum of $d_H(v, x)$ among all l-mers in sequence i

- *TotalDistance*(v,**DNA**) is the sum of the minimum Hamming distances for each DNA sequence i

- *TotalDistance*(v,**DNA**) = $min_{\mathbf{s}}\, d_H(v, s)$, where s is the set of starting positions $s_1, s_2, \ldots s_t$

Total Distance: An Example

- Given v = "acgtacgt" and **s**



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

- $TotalDistance(v, DNA) = 0$

Total Distance: Example

- Given v = "acgtacgt" and **s**

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

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

The Median String Problem: Formulation

- Goal: Given a set of DNA sequences, find a median string
- Input: A t x 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

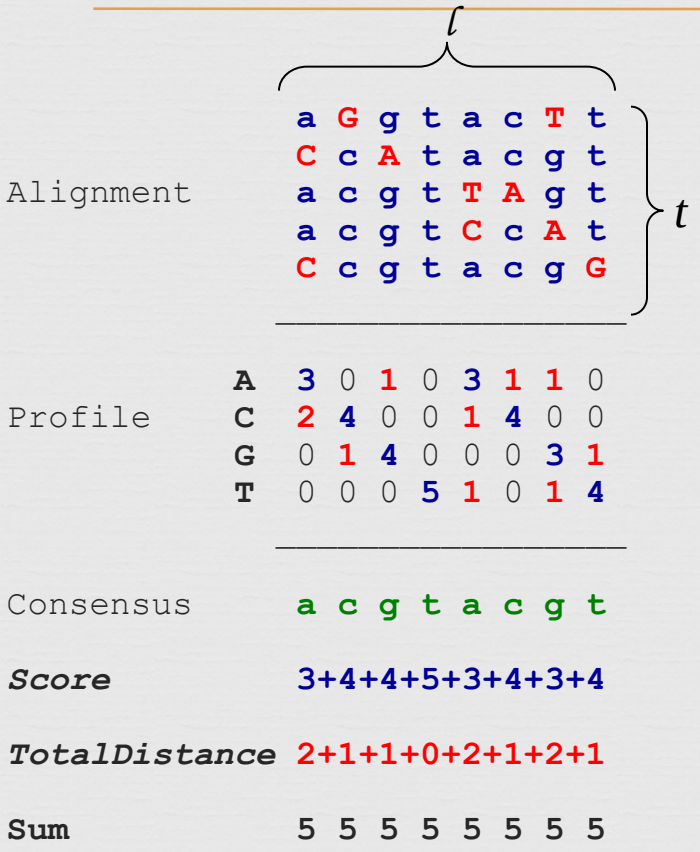
Median String Search Algorithm

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***

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



- At any column i
  $Score_i + TotalDistance_i = t$
- Because there are ℓ columns
  $Score + TotalDistance = \ell * t$
- Rearranging:
  $Score = \ell * t - TotalDistance$
- $\ell * t$ 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 $(n - l + 1)^t$ combinations!!!
  - The Median String Problem needs to examine all 4^l combinations for v. This number is relatively smaller

Motif Finding: Improving the Running Time

Recall the BruteForceMotifSearch:

```
1.    BruteForceMotifSearch(DNA, t, n, l)
2.    bestScore ← 0
3.    for each s=(s_1,s_2 , . . ., s_t) from (1,1 . . . 1) to (n−l+1, . . ., n−l+1)
4.          if (Score(s,DNA) > bestScore)
5.                bestScore ← Score(s, DNA)
6.                bestMotif ← (s_1,s_2 , . . . , s_t)
7.    return bestMotif
```

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***

Structuring the Search

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

aa… aa
aa… ac
aa… ag
aa… at
.
.
tt… tt

How to organize this search?

Alternative Representation of the Search Space

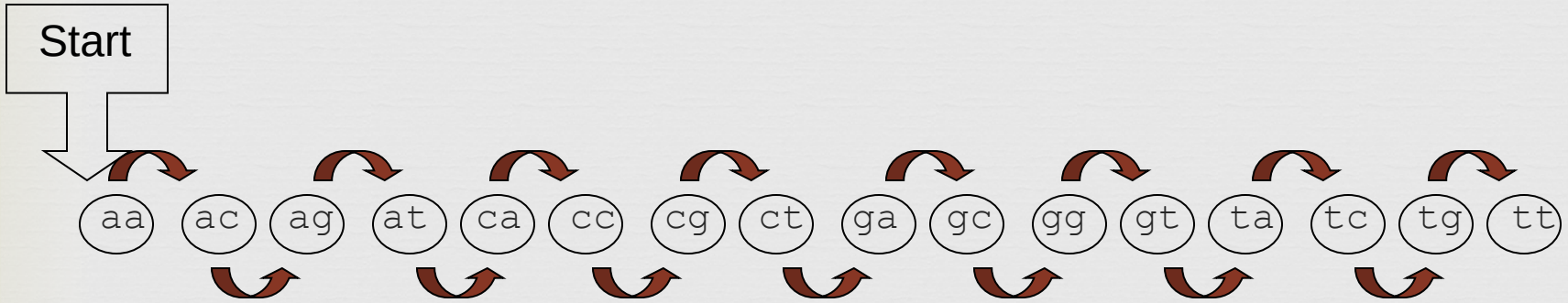
- Let **A** = 1, **C** = 2, **G** = 3, **T** = 4
- Then the sequences from AA…A to TT…T become:

$$\underbrace{11\ldots11}_{l}$$
$$11\ldots12$$
$$11\ldots13$$
$$11\ldots14$$
$$\vdots$$
$$44\ldots44$$

- Notice that the sequences above simply list all numbers as if we were counting on base 4 without using 0 as a digit

Linked List

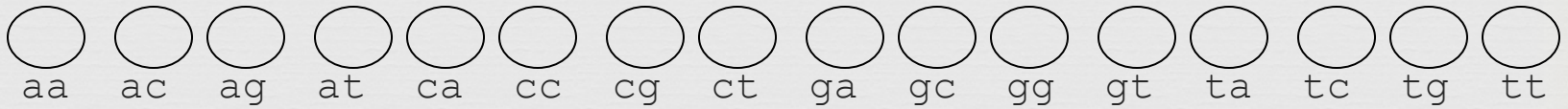
- Suppose $\ell = 2$



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

Linked List (cont'd)

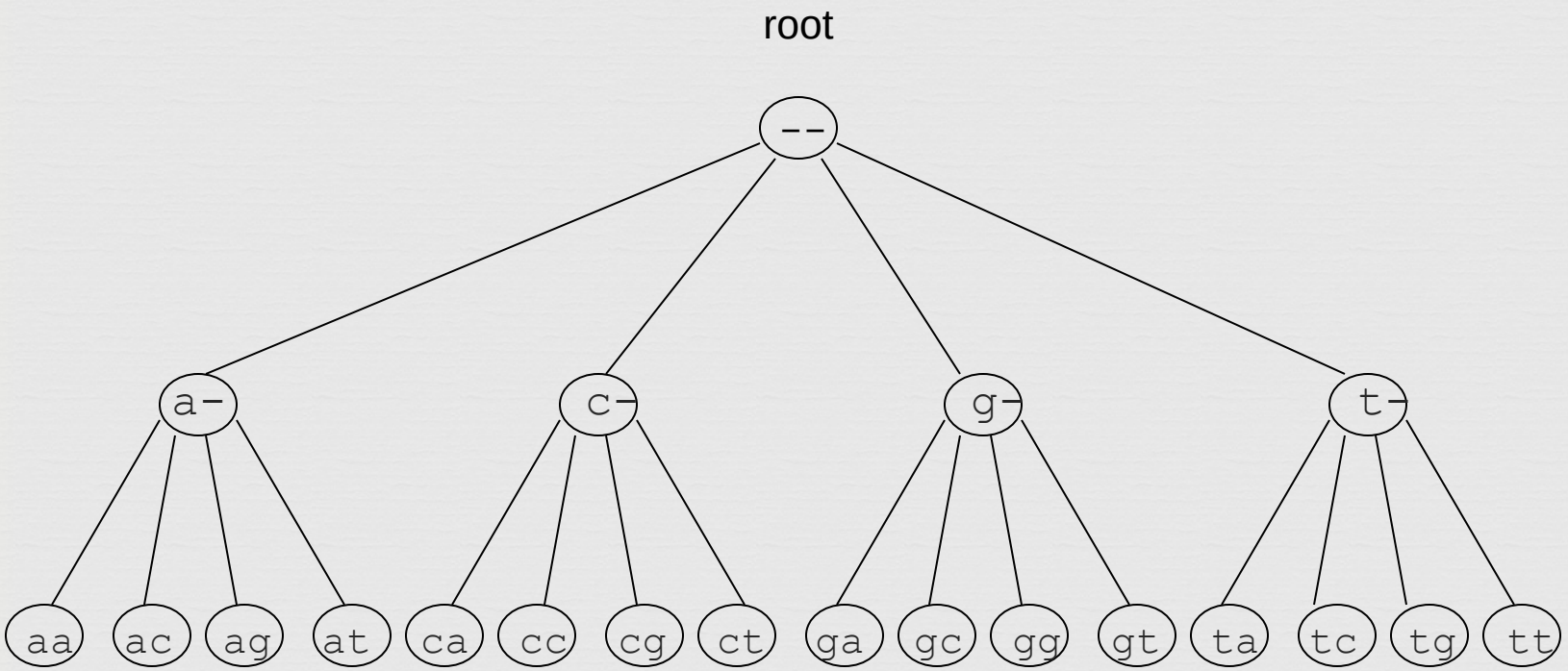
- Linked list is not the most efficient data structure for motif finding
- Let's try grouping the sequences by their prefixes



Outline

- What is Motif
- Motif finding problem
- Implanting Patterns in Random Text
- The Gold Bug Problem
- The Motif Finding Problem
- Brute Force Motif Finding
- The Median String Problem
- **Search Trees**
- Branch-and-Bound Motif Search
- Branch-and-Bound Median String Search

Search Tree



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:

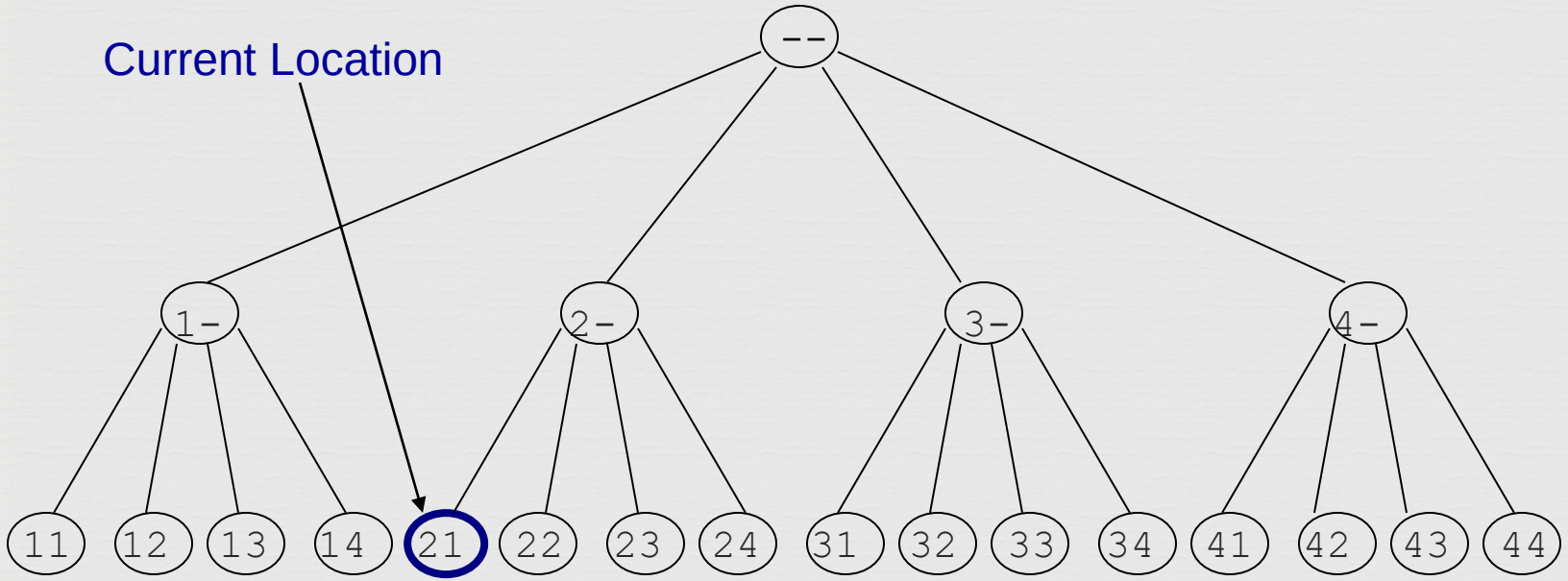
```
1.  NextLeaf( a, L, k )          // a : the array of digits
2.  for i ← 1 to L               // L: length of the array
3.      if a_i < k               // k : max digit value
4.          a_i ← a_i + 1
5.              return a
6.          a_i ← 1
7.  return a
```

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

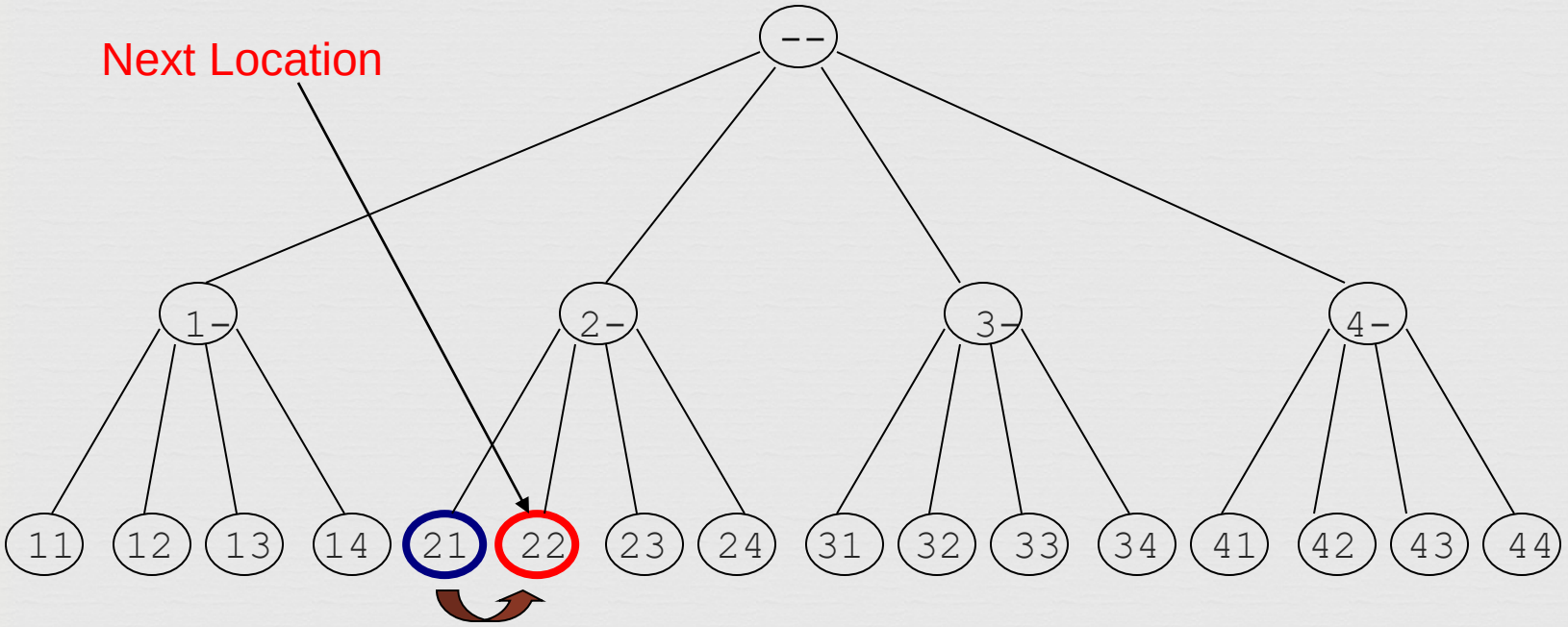
NextLeaf: Example

- Moving to the next leaf:



NextLeaf: Example (cont'd)

- Moving to the next leaf:



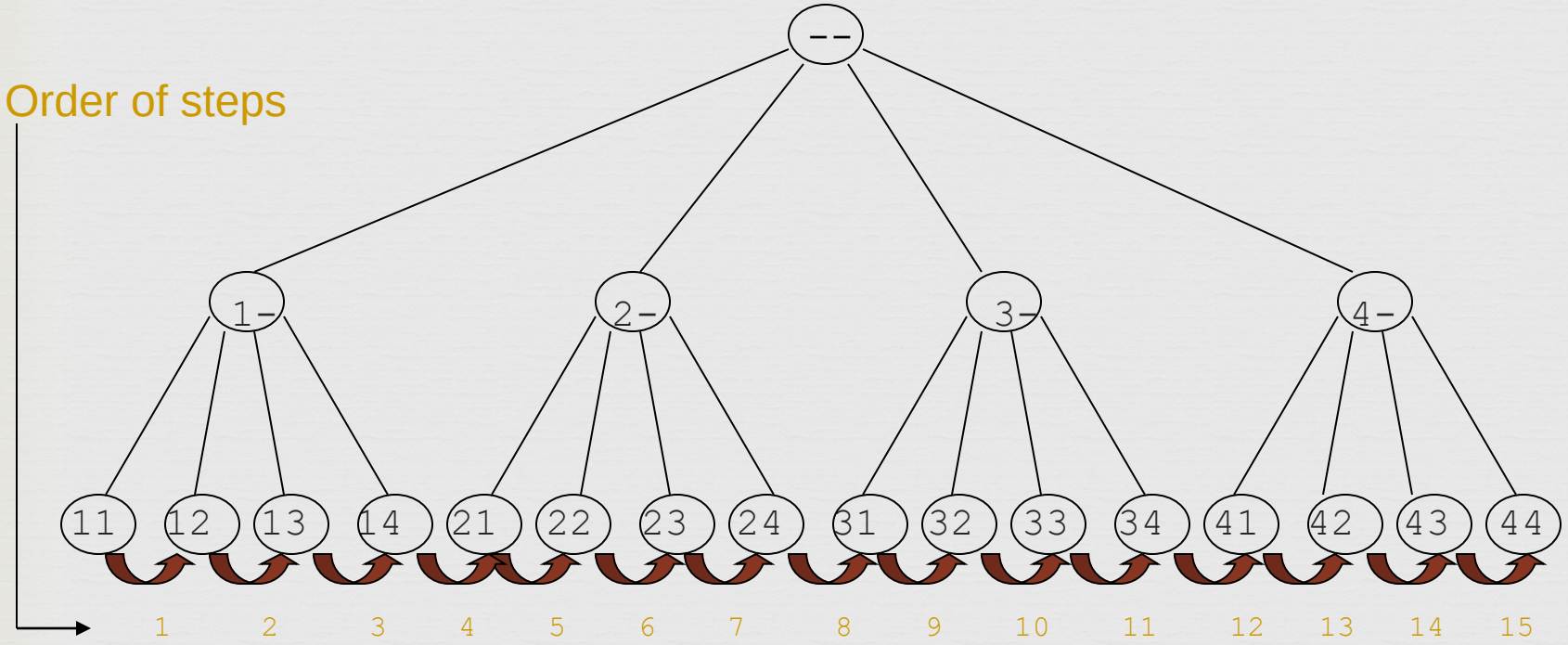
Visit All Leaves

- 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

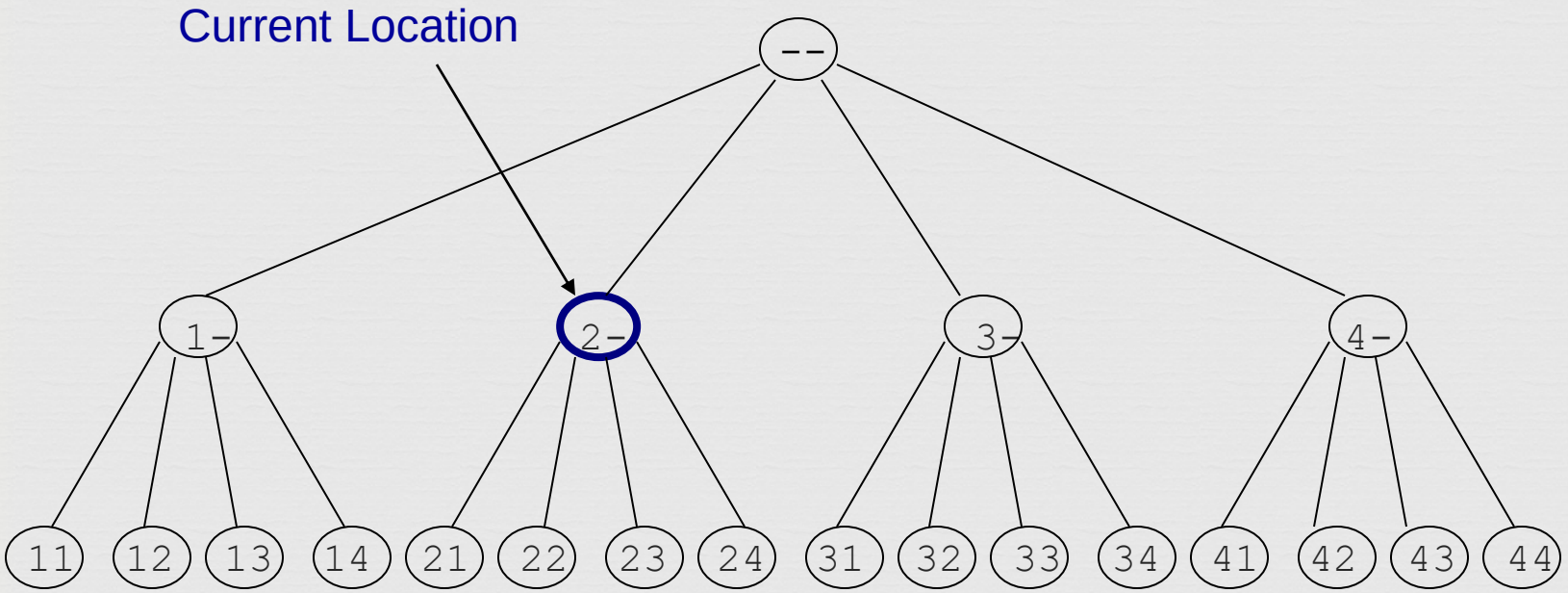
- So we can search leaves
- How about searching all vertices of the tree?
- We can do this with a *depth first* search

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} ← 1            // L: max length
4.      return ( a,i+1)        // k : max digit value
5.    else
6.      for j ← L to 1
7.        if a_j < k
8.          a_j ← a_j + 1
9.          return( a,j )
10.   return(a,0)
```

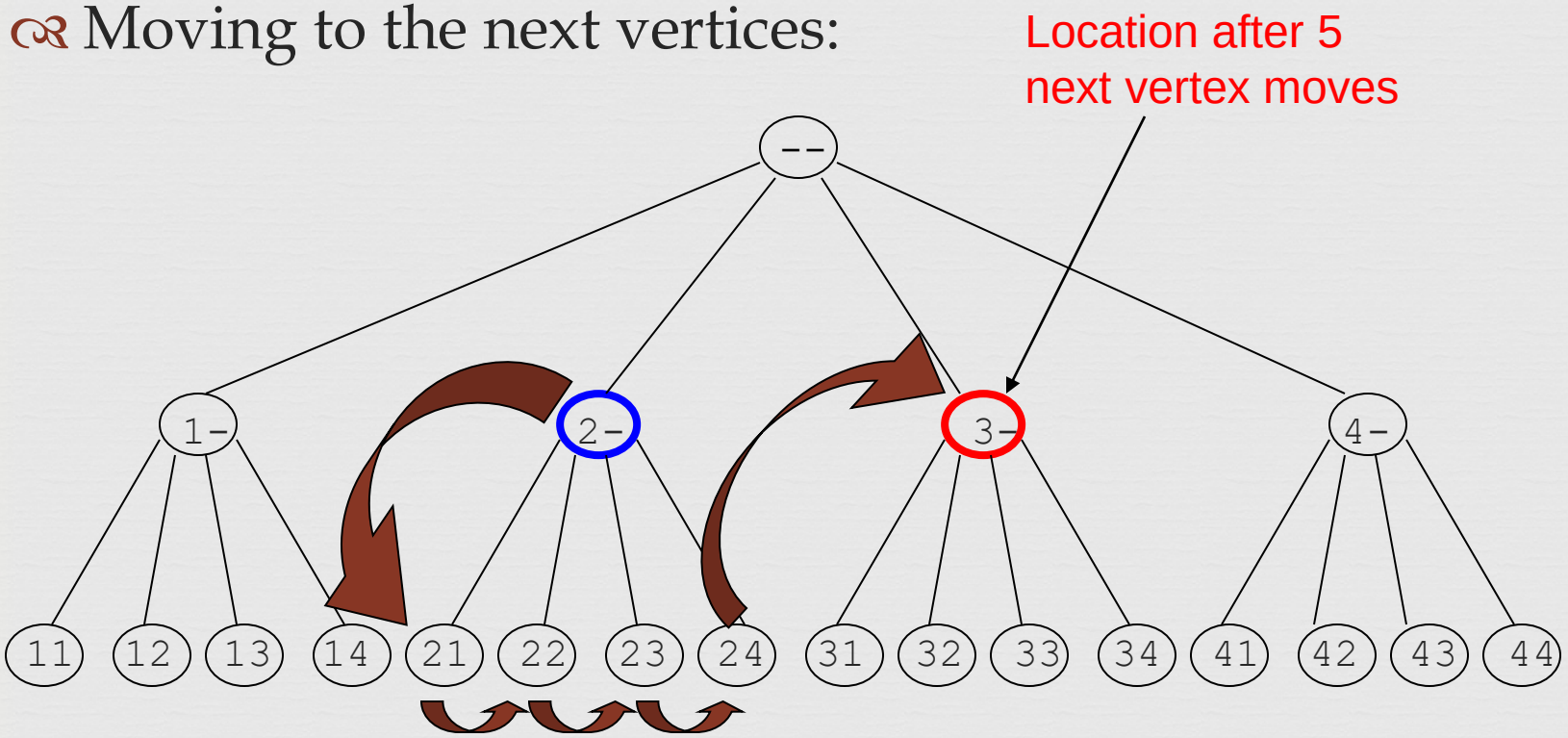
Example

- Moving to the next vertex:



Example

- Moving to the next vertices:



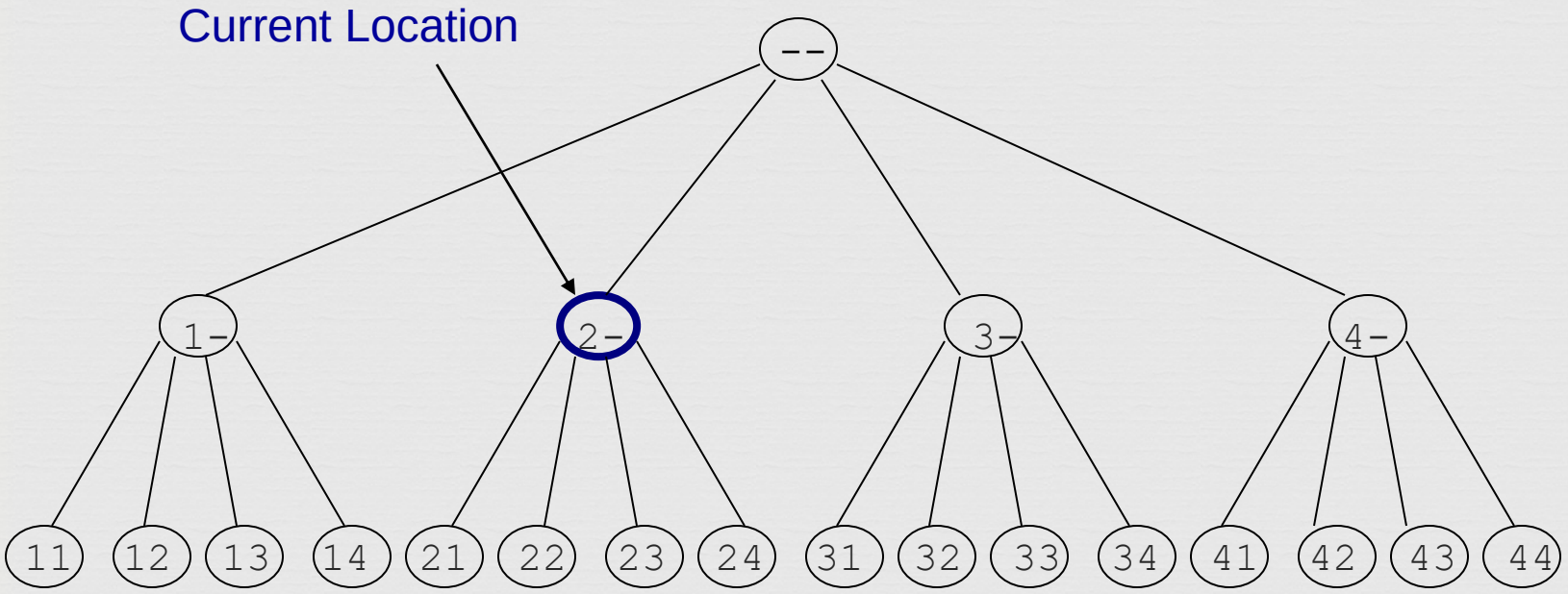
Bypass Move

- 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 ← 1 to i          // i : prefix length
3.       if a_j < k          // L: maximum length
4.           a_j ← a_j + 1   // k : max digit value
5.           return(a,j)
6.   return(a,0)
```

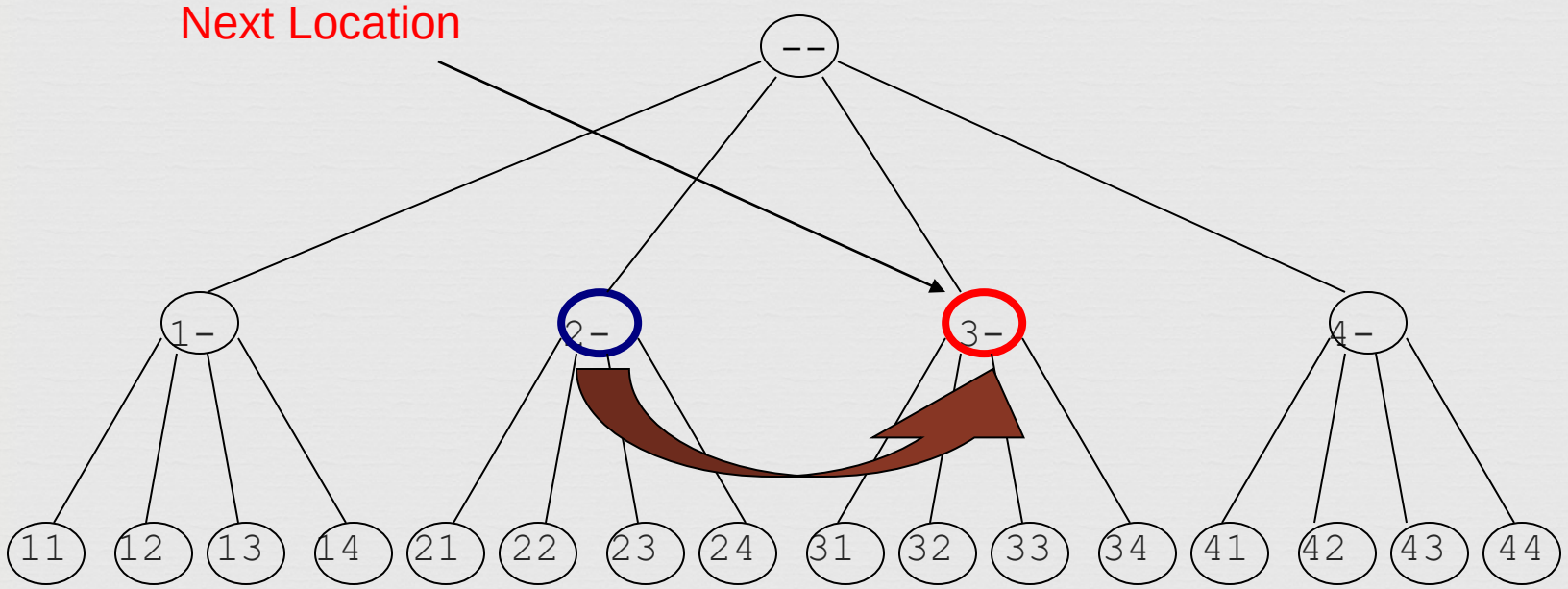
Bypass Move: Example

- Bypassing the descendants of "2-":



Example

- Bypassing the descendants of "2-":



Revisiting Brute Force Search

- Now that we have method for navigating the tree, lets look again at BruteForceMotifSearch

Brute Force Search Again

```
1.    BruteForceMotifSearchAgain(DNA, t, n, l)
2.    s ← (1,1,…, 1)
3.    bestScore ← Score(s,DNA)
4.    while forever
5.          s ← NextLeaf (s, t, n– l + 1)
6.          if (Score(s,DNA) > bestScore)
7.                bestScore ← Score(s, DNA)
8.                bestMotif ← (s1,s2 , . . . , st)
9.    return bestMotif
```

Can We Do Better?

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

  $(t - i) * \ell$ to Score(**s**,*i*,*DNA*)

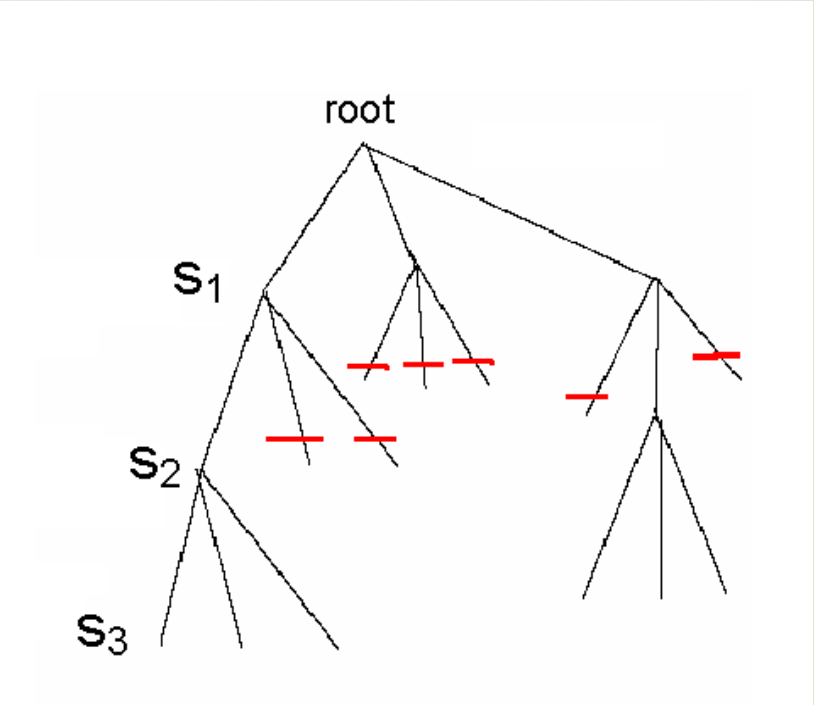
- If Score(**s**,*i*,*DNA*) + $(t - i) * \ell$ < **BestScore**, it makes no sense to search in vertices of the current subtree
  - Use **ByPass()**

Outline

- What is Motif
- Motif finding problem
- Implanting Patterns in Random Text
- The Gold Bug Problem
- The Motif Finding Problem
- Brute Force Motif Finding
- The Median String Problem
- Search Trees
- **Branch-and-Bound Motif Search**
- Branch-and-Bound Median String Search

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 ← (1,…,1)
3.   bestScore ← 0
4.   i ← 1
5.   while i > 0
6.       if i < t
7.           optimisticScore ← Score(s, i, DNA) +(t – i ) * ℓ
8.           if optimisticScore < bestScore
9.               (s, i) ← Bypass(s,i, n-ℓ+1)
10.          else
11.              (s, i) ← NextVertex(s, i, n-ℓ+1)
12.      else
13.          if Score(s,DNA) > bestScore
14.              bestScore ← Score(s)
15.              bestMotif ← (s1, s2, s3, …, st)
16.          (s,i) ← NextVertex(s,i,t,n-ℓ+ 1)
17.  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^ℓ combinations of ***v***. This number is relatively small
- We want to use median string algorithm with the Branch and Bound trick!

Outline

- What is Motif
- Motif finding problem
- Implanting Patterns in Random Text
- The Gold Bug Problem
- The Motif Finding Problem
- Brute Force Motif Finding
- The Median String Problem
- Search Trees
- Branch-and-Bound Motif Search
- **Branch-and-Bound Median String Search**

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

Improving the Bounds

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

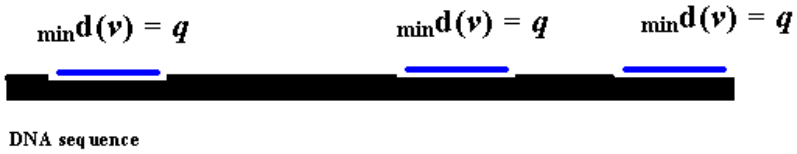
Improving the Bounds (cont'd)

- Repeating the process for the suffix v gives us a minimum distance for v

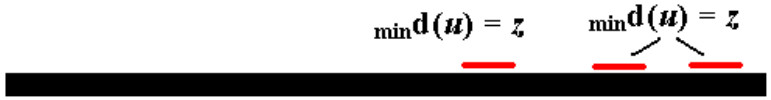
- Since u and v are two substrings of w, and included in motif w, we can assume that the minimum distance of u plus minimum distance of v can only be less than the minimum distance for w

Better Bounds

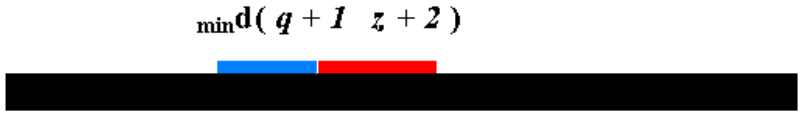
Searching for prefix V
We may find many instances of prefix V with a minimum distance q



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(***prefix***) + d(***suffix***) ≥ ***bestDistance***:
  - Motif ***w*** (*prefix.suffix*) cannot give a better (lower) score than d(***prefix***) + d(***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 = ∞
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 ≥ 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, …, st)
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
```

More on the Motif Problem

- Exhaustive Search and Median String are both exact algorithms
- They always find the optimal solution, though they may be too slow to perform practical tasks
- Many algorithms sacrifice optimal solution for speed