CS481/CS583: Bioinformatics Algorithms

Can Alkan
EA509
calkan@cs.bilkent.edu.tr

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

MOTIFS

Random Sample

atgaccgggatactgataccgtatttggcctaggcgtacacattagataaacgtatgaagtacgttagactcggcgccgccg

acccctattttttgagcagatttagtgacctggaaaaaaaatttgagtacaaaacttttccgaatactgggcataaggtaca

tgagtatccctgggatgacttttgggaacactatagtgctctcccgatttttgaatatgtaggatcattcgccagggtccga

gctgagaattggatgaccttgtaagtgttttccacgcaatcgcgaaccaacgcggacccaaaggcaagaccgataaaggaga

tccctttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaatggcccacttagtccacttatag

gtcaatcatgttcttgtgaatggatttttaactgagggcatagaccgcttggcgcacccaaattcagtgtgggcgagcgcaa

cggttttggcccttgttagaggcccccgtactgatggaaactttcaattatgagagagctaatctatcgcgtgcgtgttcat

aacttgagttggtttcgaaaatgctctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgta

ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcatttcaacgtatgccgaaccgaaagggaag

ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagcttctgggtactgatagca

Implanting Motif **AAAAAAAGGGGGGG**

atgaccgggatactgatAAAAAAAAGGGGGGGggcgtacacattagataaacgtatgaagtacgttagactcggcgccgccg

acccctattttttgagcagatttagtgacctggaaaaaaaatttgagtacaaaacttttccgaataAAAAAAAAGGGGGGGa

tgagtatccctgggatgacttAAAAAAAAGGGGGGGtgctctcccgatttttgaatatgtaggatcattcgccagggtccga

gctgagaattggatgAAAAAAAAGGGGGGGtccacgcaatcgcgaaccaacgcggacccaaaggcaagaccgataaaggaga

tccctttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaatAAAAAAAAGGGGGGGcttatag

gtcaatcatgttcttgtgaatggatttAAAAAAAAGGGGGGGgaccgcttggcgcacccaaattcagtgtgggcgagcgcaa

cggttttggcccttgttagaggcccccgtAAAAAAAAGGGGGGGcaattatgagagagctaatctatcgcgtgcgtgttcat

aacttgagttAAAAAAAAGGGGGGGctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgta

ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcatAAAAAAAAGGGGGGGaccgaaagggaag

ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagcttAAAAAAAAGGGGGGGa

Where is the Implanted Motif?

atgaccgggatactgataaaaaaaaggggggggcgtacacattagataaacgtatgaagtacgttagactcggcgccgccg

acccctattttttgagcagatttagtgacctggaaaaaaaatttgagtacaaaacttttccgaataaaaaaaaggggggga

tgagtatccctgggatgacttaaaaaaaaggggggtgctctcccgatttttgaatatgtaggatcattcgccagggtccga

gctgagaattggatgaaaaaaaaggggggtccacgcaatcgcgaaccaacgcggacccaaaggcaagaccgataaaggaga

tccctttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaataaaaaaaaggggggcttatag

gtcaatcatgttcttgtgaatggatttaaaaaaaaggggggggaccgcttggcgcacccaaattcagtgtgggcgagcgcaa

cggttttggcccttgttagaggcccccgtaaaaaaaaggggggcaattatgagagagctaatctatcgcgtgcgtgttcat

aacttgagttaaaaaaaaggggggctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgta

ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcataaaaaaaaggggggaccgaaagggaag

ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagcttaaaaaaaaggggggga

Implanting Motif **AAAAAAGGGGGGG** with Four Mutations

atgaccgggatactgatAgAAgAAAGGttGGGggcgtacacattagataaacgtatgaagtacgttagactcggcgccgccg

acccctattttttgagcagatttagtgacctggaaaaaaaatttgagtacaaaacttttccgaatacAAtAAAAcGGcGGGa

tgagtatccctgggatgacttAAAAtAAtGGaGtGGtgctctcccgatttttgaatatgtaggatcattcgccagggtccga

gctgagaattggatgcAAAAAAAGGGattGtccacgcaatcgcgaaccaacgcggacccaaaggcaagaccgataaaggaga

tccctttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaatAtAAtAAAGGaaGGGcttatag

gtcaatcatgttcttgtgaatggatttAAcAAtAAGGGctGGgaccgcttggcgcacccaaattcagtgtgggcgagcgcaa

cggttttggcccttgttagaggcccccgtAtAAAcAAGGaGGGccaattatgagagagctaatctatcgcgtgcgtgttcat

aacttgagttAAAAAAtAGGGaGccctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgta

ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcatActAAAAAGGaGcGGaccgaaagggaag

ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagcttActAAAAAGGaGcGGa

Where is the Motif???

atgaccgggatactgatagaagaaaggttgggggcgtacacattagataaacgtatgaagtacgttagactcggcgccgccg

accccctattttttgagcagatttagtgacctggaaaaaaaatttgagtacaaaacttttccgaatacaataaaacggcggga

tgagtatccctgggatgacttaaaataatggagtggtgctctcccgatttttgaatatgtaggatcattcgccagggtccga

gctgagaattggatgcaaaaaaagggattgtccacgcaatcgcgaaccaacgcggacccaaaggcaagaccgataaaggaga

tccctttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaatataataaaggaagggcttatag

gtcaatcatgttcttgtgaatggatttaacaataagggctgggaccgcttggcgcacccaaattcagtgtgggcgagcgcaa

cggttttggcccttgttagaggcccccgtataaacaaggagggccaattatgagagagctaatctatcgcgtgcgtgttcat

aacttgagttaaaaaatagggagccctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgta

ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcatactaaaaaggagcggaccgaaagggaag

ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagcttactaaaaaggagcgga

Finding (15,4) Motif

```
atgaccgggatactgatAgAAgAAAGGttGGGggcgtacacattagataaacgtatgaagtacgttagactcggcgccgccg
acccctattttttgagcagatttagtgacctggaaaaaaaatttgagtacaaaacttttccgaatacAAtAAAAcGGcGGGa
tgagtatccctgggatgacttAAAAtAAtGGaGtGGtgctctcccgatttttgaatatgtaggatcattcgccagggtccga
gctgagaattggatgcAAAAAAAGGGattGtccacgcaatcgcgaaccaacgcggacccaaaggcaagaccgataaaggaga
tccctttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaatAtAAtAAAGGaaGGGcttatag
gtcaatcatgttcttgtgaatggatttAAcAAtAAGGGctGGgaccgcttggcgcacccaaattcagtgtgggcgagcgcaa
cggttttggcccttgttagaggcccccgtAtAAAcAAGGaGGGccaattatgagagagctaatctatcgcgtgcgtgttcat
aacttgagttAAAAAAtAGGGaGccctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgta
ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcatActAAAAAGGaGcGGaccgaaagggaag
ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagcttActAAAAAGGaGcGGa

AgAAgAAAGGttGGG
..|..|||.|..|||
cAAtAAAAcGGcGGG
```

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.

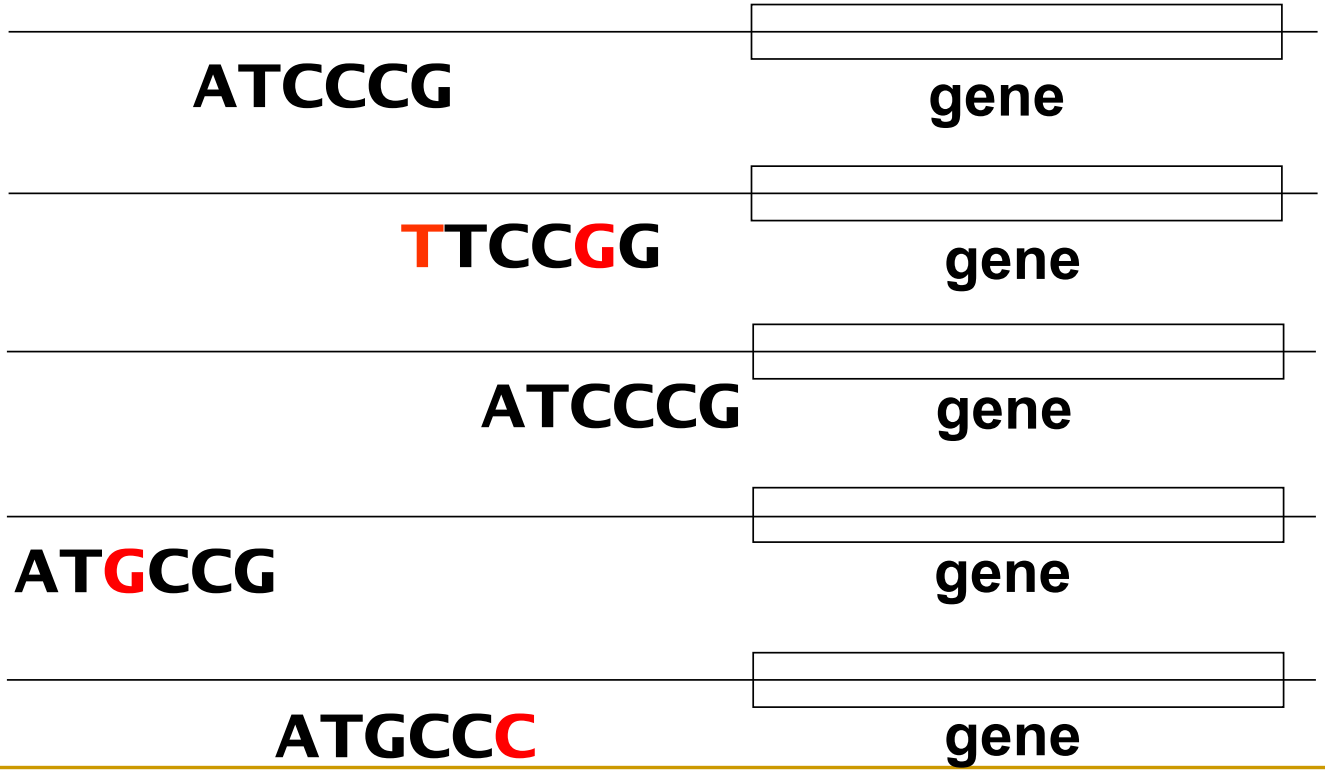
Regulatory Regions

- 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
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:

```
cctgatagacgctatctggctatccacgtacgtaggtcctctgtgcgaatctatgcgtttccaaccat
agtactggtgtacatttgatacgtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc
aaacgtacgtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt
agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatcttacgtacgtataca
ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgttaacgtacgtc
```

acgtacgt

Consensus String

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

The Motif Finding Problem (cont'd)

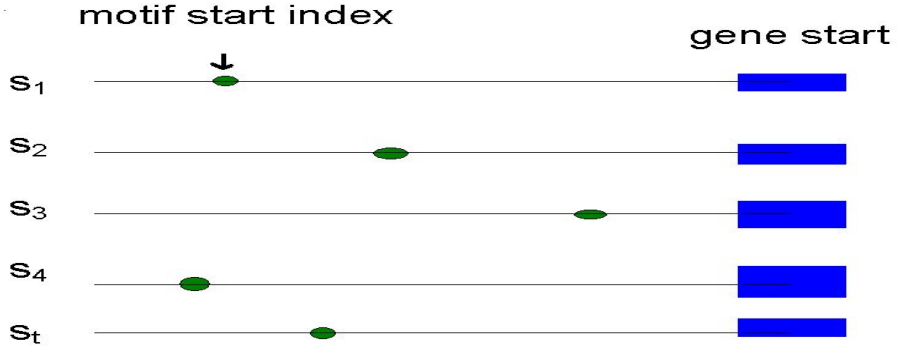
- The patterns with 2 point mutations:

```
cctgatagacgctatctggctatccaGgtacTtaggtcctctgtgcgaatctatgcgtttccaaccat

agtactggtgtacatttgatCcAtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc

aaacgtTAgtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt

agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatcttacgtCcAtataca

ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgttaCcgtacgGc
```

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 $\boldsymbol{s} = (s_1, s_2, s_3, \ldots, s_t)$



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

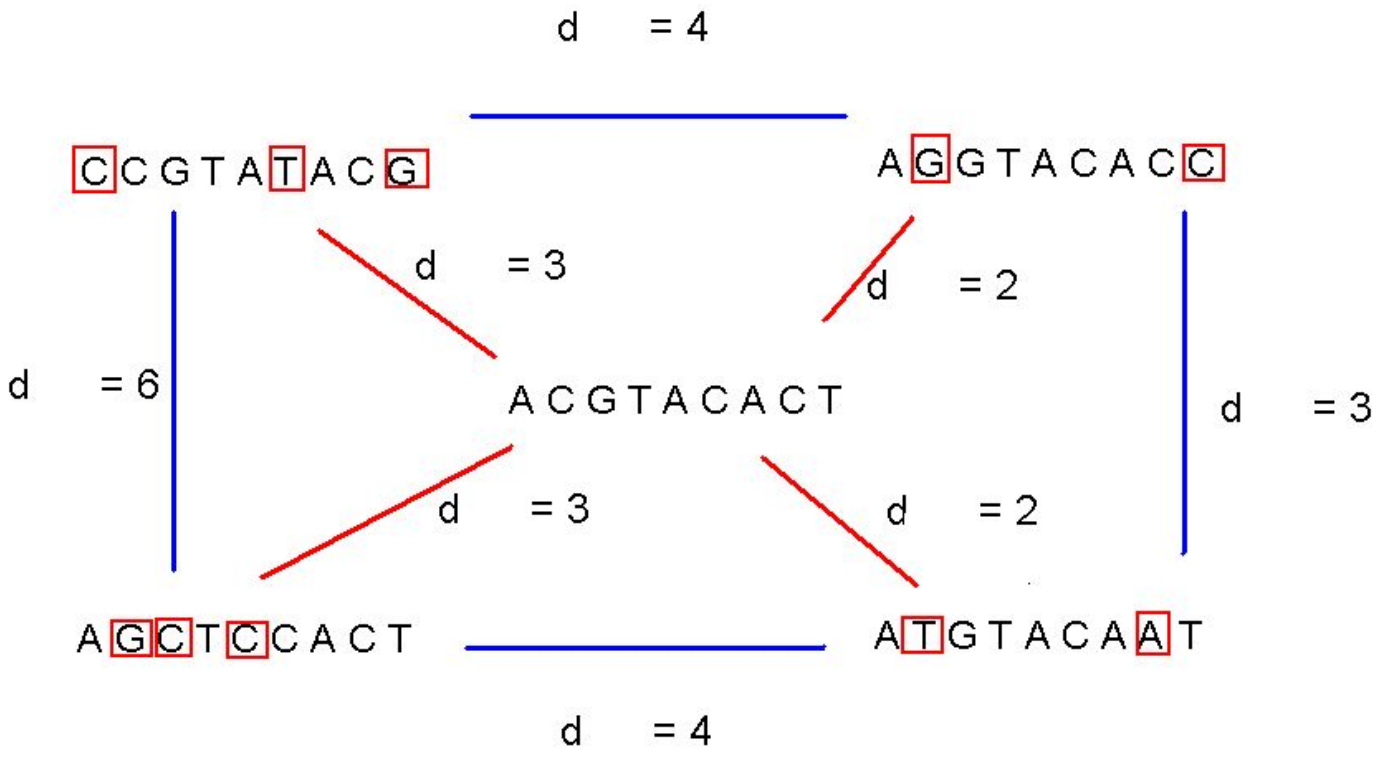
$$\mathbf{s} = (s_1, s_2, \ldots, s_t)$$

- Construct 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 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

cctgatagacgctatctggctatcc aGgtacTtaggtcctctgtgcgaatctatgcgtttccaaccat

agtactggtgtacatttgat CcAtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc

aaacgtTAgtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt

agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatctt acgtCcAtataca

ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgtta CcgtacgGc

$t=5$

$n = 69$

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

Scoring Motifs

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

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

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	t
a	c	g	t	C	c	A	t	
C	c	g	t	a	c	g	G	

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

Basic Scoring

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 **s** = (s_1, s_2, … s_t) maximizing *Score*(**s**,***DNA***)

The Motif Finding Problem: Brute Force Solution

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

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

BruteForceMotifSearch

```
BruteForceMotifSearch(DNA, t, n, ℓ)
bestScore ← 0
for each s=(s_1,s_2 , . . ., s_t) from (1,1 . . . 1)
                        to (n-ℓ+1, . . ., n-ℓ+1)
    if (Score(s,DNA) > bestScore)
        bestScore ← score(s, DNA)
        bestMotif ← (s_1,s_2 , . . . , s_t)
return bestMotif
```

Running Time of BruteForceMotifSearch

- Varying $(n - \ell + 1)$ positions in each of $\boldsymbol{t}$ 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 $\boldsymbol{t} = 8$, $\boldsymbol{n} = 1000$, $\ell = 10$ 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:
  - $d_H(\boldsymbol{v},\boldsymbol{w})$ 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: An Example

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

```
                                  acgtacgt
cctgatagacgctatctggctatcc acgtacgtaggtcctctgtgcgaatctatgcgtttccaaccat
                          acgtacgt
agtactggtgtacatttgat acgtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc
  acgtacgt
aaacgtacgtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt
                                                          acgtacgt
agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatctt acgtacgtataca
                                                             acgtacgt
ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgtta acgtacgtc
```

$d_H(v, x) = 0$

$d_H(v, x) = 0$

$d_H(v, x) = 0$

$d_H(v, x) = 0$

$d_H(v, x) = 0$

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

- *TotalDistance*(***v***,***DNA***) = 0

Total Distance: Example

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

$d_H(v, x) = 1$

```
                                  acgtac gt
cctgatagacgctatctggctatcc acgtacAtaggtcctctgtgcgaatctatgcgtttccaaccat
```

$d_H(v, x) = 0$

```
                                  acgtacgt
agtactggtgtacatttgat acgtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc
```

```
  acgtacgt
aaaAgtCcgtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt
```

$d_H(v, x) = 2$

$d_H(v, x) = 0$

```
                                                          acgtacgt
agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatctt acgtacgtataca
                                                              acgtacgt
ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgtta acgtaGgtc
```

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 $d_H(\boldsymbol{v}, \boldsymbol{x})$, where ***x*** is an ℓ-mer with starting position s_i
  $(1 \leq s_i \leq \boldsymbol{n} - \ell + 1)$
- Find minimum of $d_H(\boldsymbol{v}, \boldsymbol{x})$ 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**)* = $min_{\mathbf{s}}\ d_H(\boldsymbol{v}, \boldsymbol{s})$, where ***s*** is the set of starting positions $s_1, s_2, \ldots s_t$

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

```
MedianStringSearch (DNA, t, n, ℓ)
bestWord ←AAA…A
bestDistance ← ∞
    for each ℓ-mer s from AAA…A to TTT…T
    if TotalDistance(s,DNA) < bestDistance
        bestDistance ← TotalDistance(s,DNA)
        bestWord ← s
  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

ℓ

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
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

t

- At any column $\boldsymbol{i}$
  $Score_i + TotalDistance_i = \boldsymbol{t}$
- Because there are ℓ columns
  $Score + TotalDistance = \ell * \boldsymbol{t}$
- Rearranging:
  $Score = \ell * \boldsymbol{t} - TotalDistance$
- $\ell * \boldsymbol{t}$ 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 $(\boldsymbol{n} - \ell + 1)^t$ 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:

```
BruteForceMotifSearch(DNA, t, n, ℓ)
bestScore ← 0
for each s=(s1,s2 , . . ., st) from (1,1 . . . 1) to (n-ℓ+1, . . ., n-ℓ+1)
    if (Score(s,DNA) > bestScore)
        bestScore ← Score(s, DNA)
        bestMotif ← (s1,s2 , . . . , st)
return bestMotif
```

Structuring the Search

- How can we perform the line

for each **s**=(s_1,s_2 , . . ., s_t) from (1,1 . . . 1) to (***n***-ℓ+1, . . ., ***n***-ℓ+1) ?

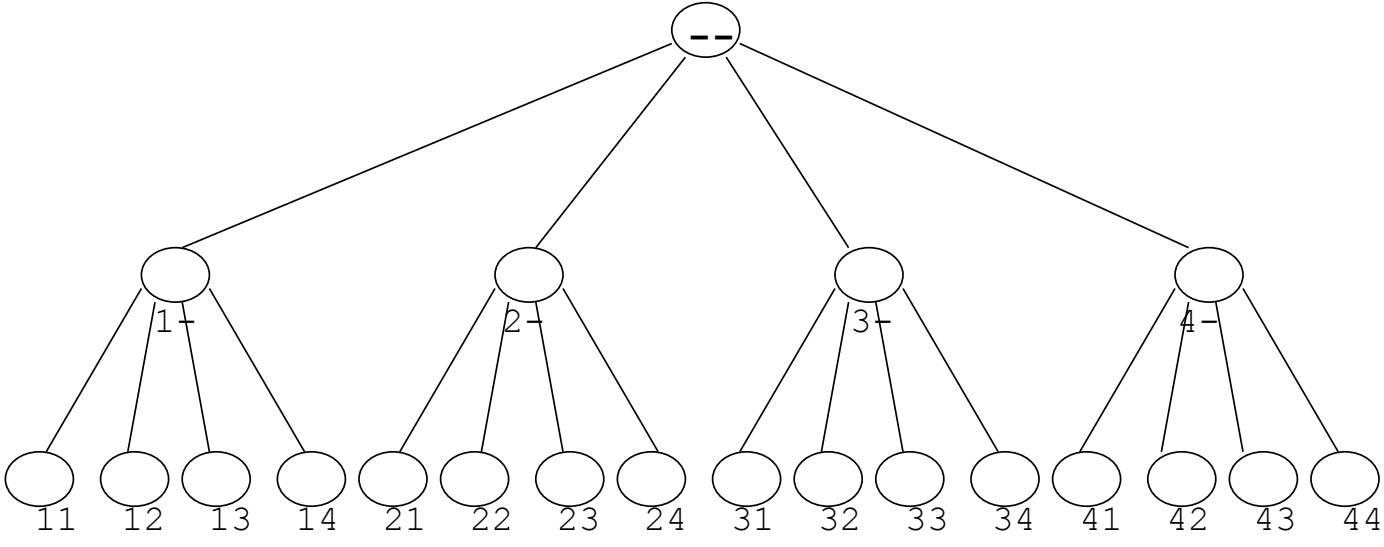
- We need a method for efficiently structuring and navigating the many possible motifs
- This is not very different than exploring all *t*-digit numbers

Median String: Improving the Running Time

```
MedianStringSearch (DNA, t, n, ℓ)
bestWord ← AAA…A
bestDistance ← ∞
    for each ℓ-mer s from AAA…A to TTT…T
    if TotalDistance(s,DNA) < bestDistance
        bestDistance ← TotalDistance(s,DNA)
        bestWord ← s
  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:

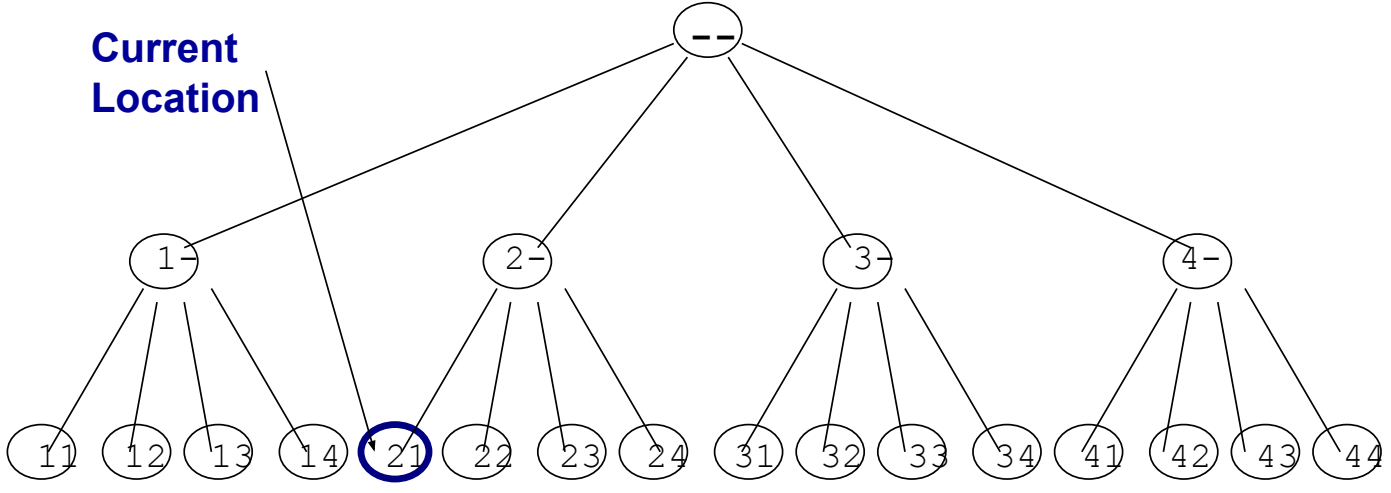
```
NextLeaf( a, L, k )          // a : the array of digits
for i ← L to 1               // L: length of the array
    if a_i < k               // k : max digit value
        a_i ← a_i + 1
            return a
    a_i ← 1
return a
```

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

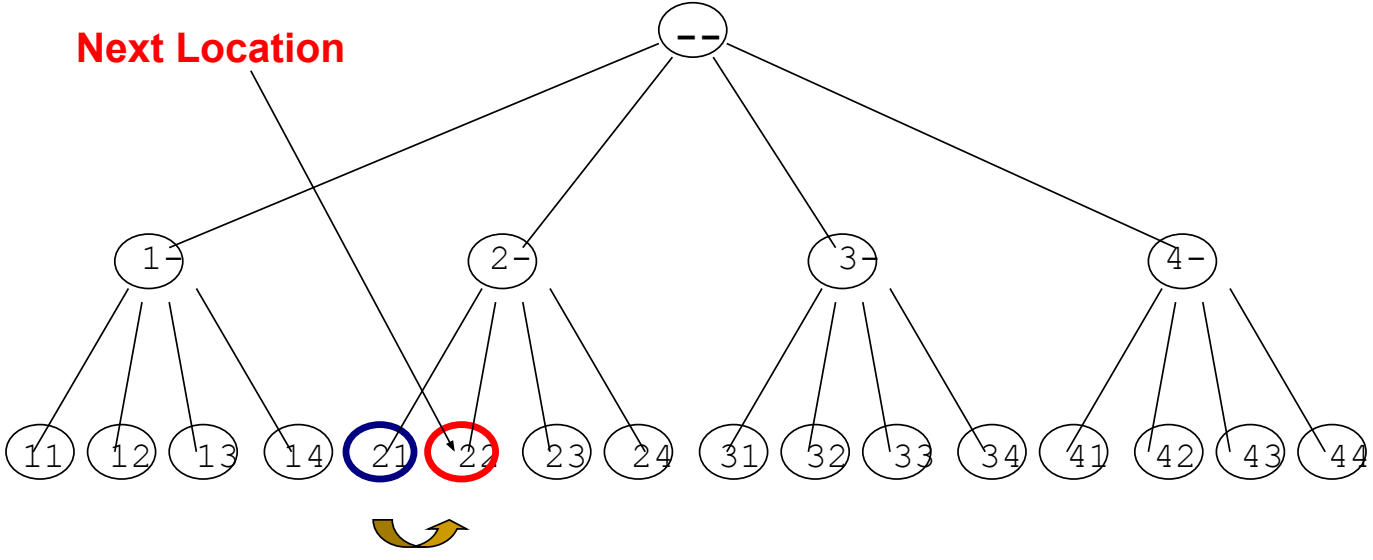
NextLeaf: Example

- Moving to the next leaf:



NextLeaf: Example (cont'd)

- Moving to the next leaf:



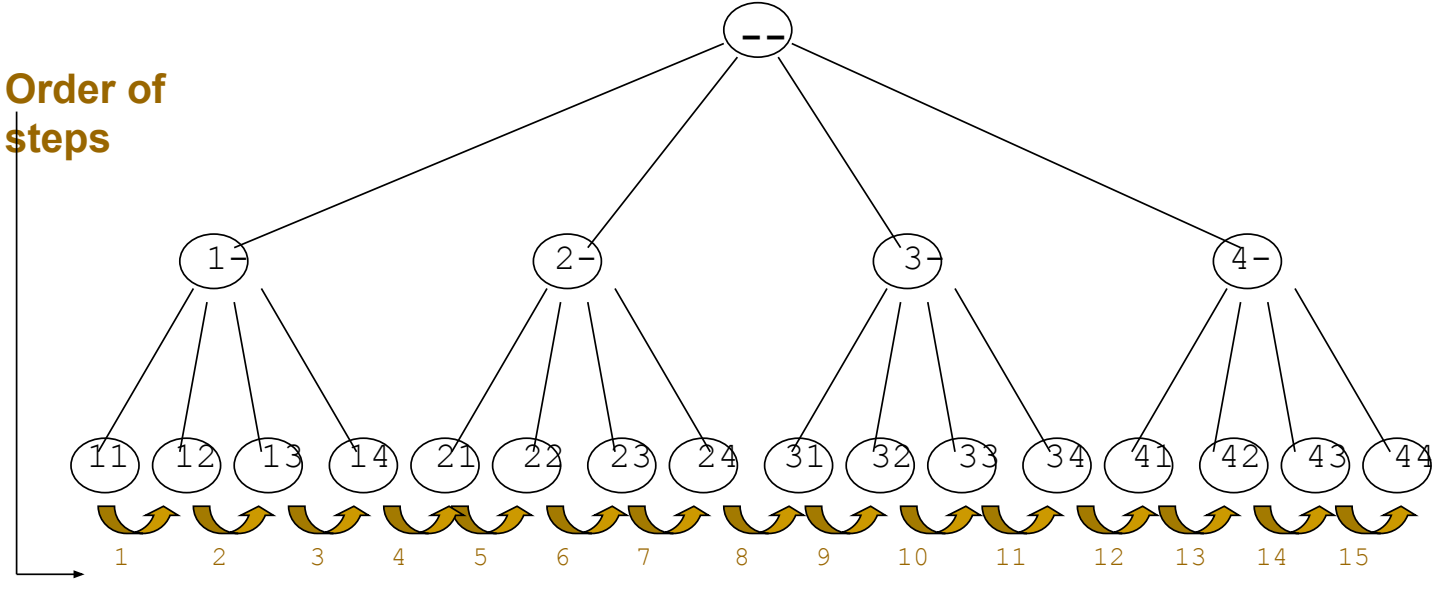
Visit All Leaves

- Printing all permutations in ascending order:

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

Visit All Leaves: Example

- Moving through all the leaves in order:



Depth First Search

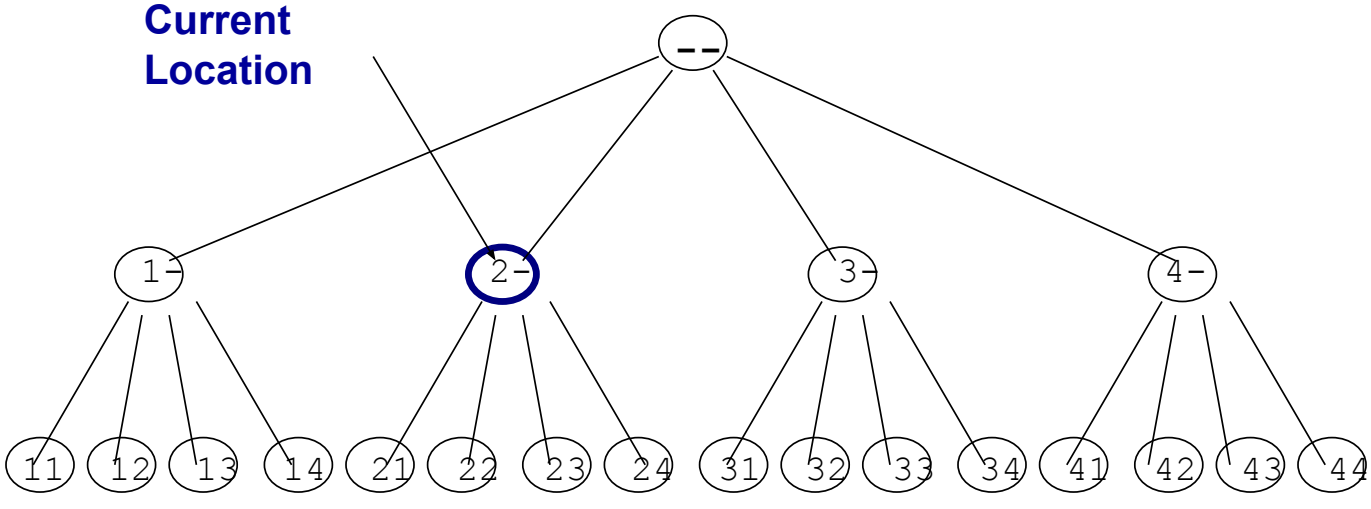
- So we can search leaves
- How about searching all vertices of the tree?
- We can do this with a *depth first* search

Visit the Next Vertex

```
NextVertex(a,i,L,k)        // a : the array of digits
  if i < L                 // i  : prefix length
    a_{i+1} ← 1            // L: max length
    return ( a,i+1)        // k : max digit value
  else
    for j ← ℓ to 1
      if a_j < k
        a_j ← a_j + 1
        return( a,j )
  return(a,0)
```

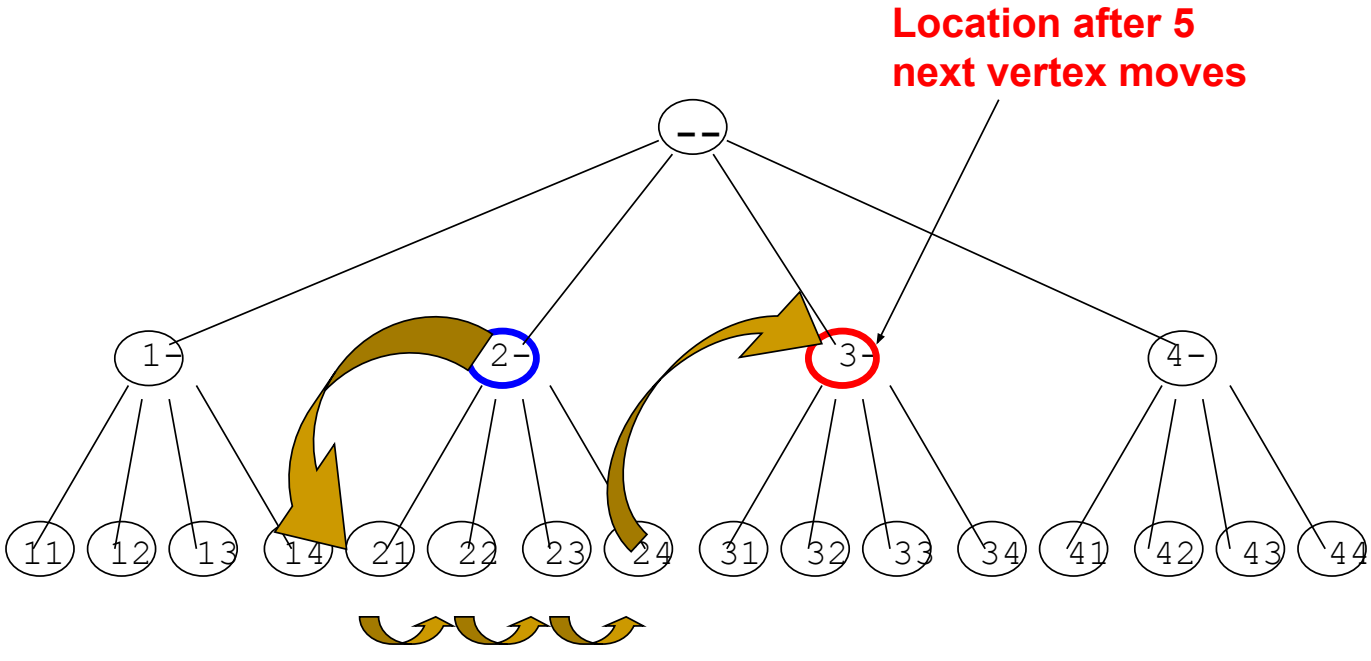
Example

- Moving to the next vertex:



Example

- Moving to the next vertices:



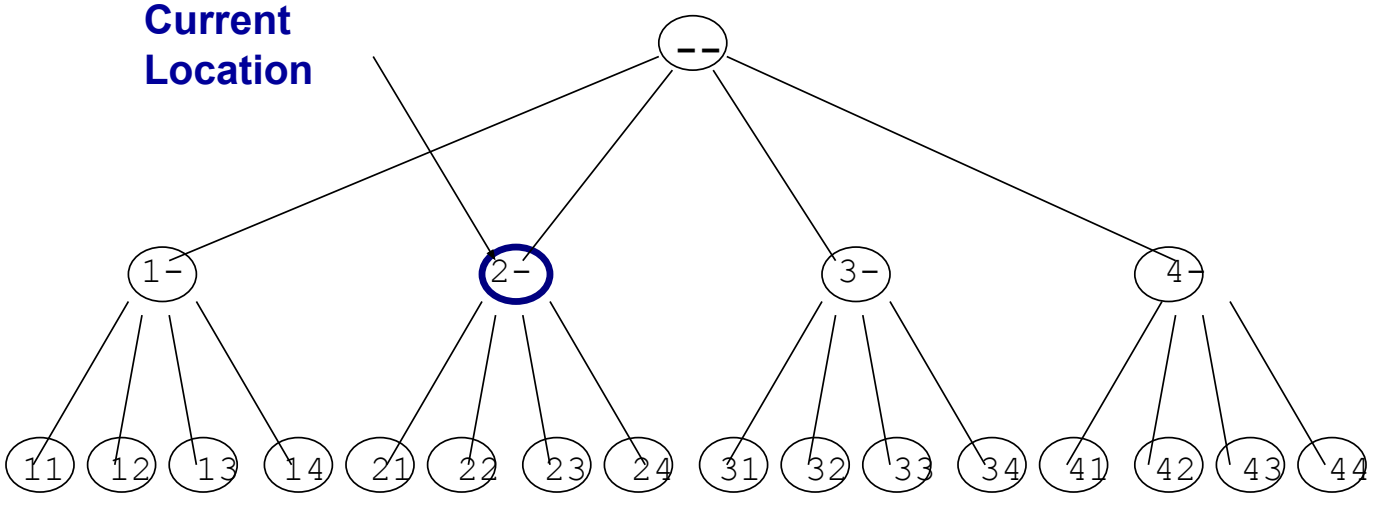
Bypass Move

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

```
Bypass(a,i,L,k)        // a: array of digits
for j ← i to 1         // i : prefix length
    if a_j < k         // L: maximum length
        a_j ← a_j + 1  // k : max digit value
        return(a,j)
return(a,0)
```

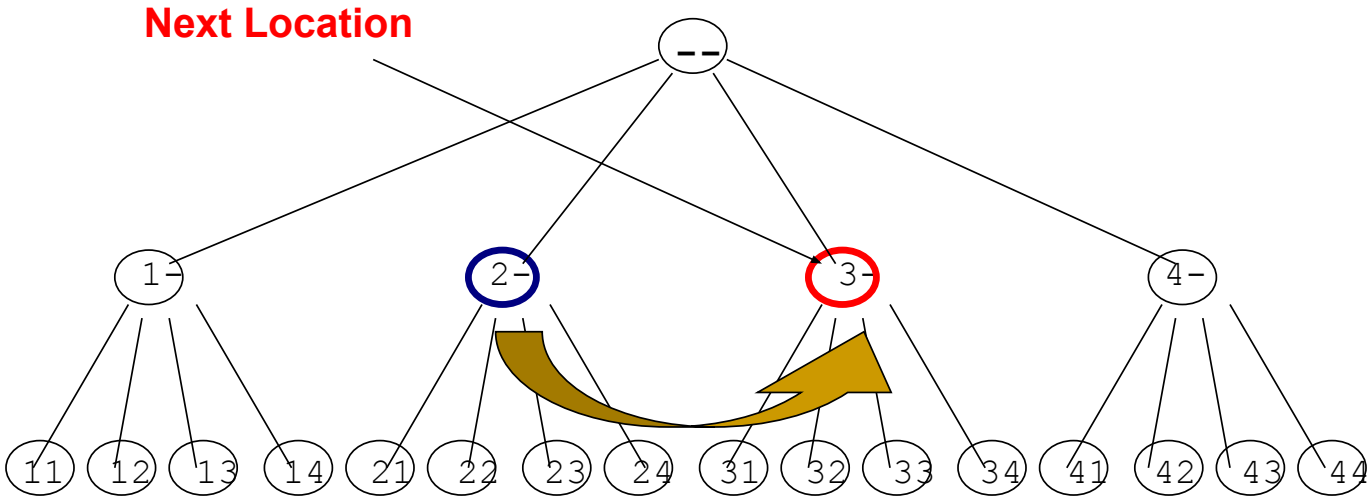
Bypass Move: Example

- 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

```
BruteForceMotifSearchAgain(DNA, t, n, ℓ)
s ← (1,1,…, 1)
bestScore ← Score(s,DNA)
while forever
    s ← NextLeaf (s, t, n- ℓ+1)
    if (Score(s,DNA) > bestScore)
        bestScore ← Score(s, DNA)
        bestMotif ← (s_1,s_2 , . . . , s_t)
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
- <u>Optimism</u>: if all subsequent *(t-i)* positions (s_{i+1}, …s_t) add

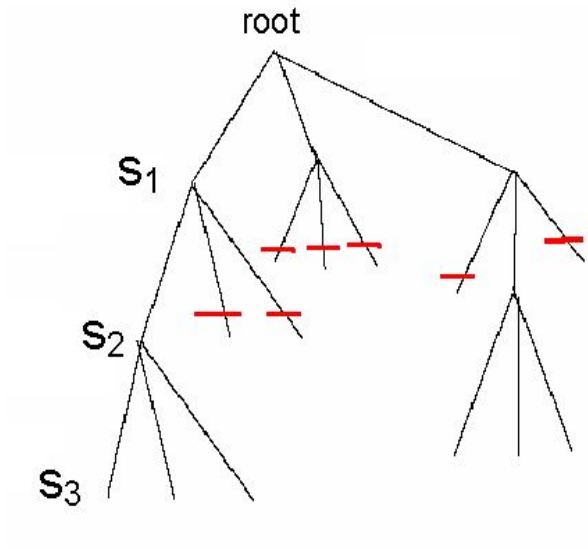
$$(\boldsymbol{t} - \boldsymbol{i}) * \ell \text{ to } \mathrm{Score}(\mathbf{s}, \boldsymbol{i}, \boldsymbol{DNA})$$

- If Score(**s**,***i***,***DNA***) + (***t*** – ***i***) * ℓ < **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 $(\boldsymbol{n} - \ell + 1)^{t-i}$ leaves
  - Use **NextVertex()** and **ByPass()** to navigate the tree



Pseudocode for Branch and Bound Motif Search

```
BranchAndBoundMotifSearch(DNA,t,n,ℓ)
s ← (1,…,1)
bestScore ← 0
i ← 1
while i > 0
    if i < t
        optimisticScore ← Score(s, i, DNA) +(t – i ) * ℓ
        if optimisticScore < bestScore
          (s, i) ← Bypass(s,i, n-ℓ+1)
        else
          (s, i) ← NextVertex(s, i, n-ℓ+1)
    else
        if Score(s,DNA) > bestScore
           bestScore ← Score(s)
           bestMotif ← (s1, s2, s3, …, st)
             (s,i) ← NextVertex(s,i,t,n-ℓ+ 1)
return bestMotif
```

Structuring the Search: median string

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

$$\overbrace{\text{aa}\ldots\text{aa}}^{\ell}$$
$$\text{aa}\ldots\text{ac}$$
$$\text{aa}\ldots\text{ag}$$
$$\text{aa}\ldots\text{at}$$
$$\vdots$$
$$\text{tt}\ldots\text{tt}$$

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:

$$\overbrace{11\ldots11}^{\ell}$$
$$11\ldots12$$
$$11\ldots13$$
$$11\ldots14$$
$$\vdots$$
$$44\ldots44$$

- Notice that the sequences above simply list all numbers as if we were counting on base 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 $\mathbf{s}$.
  - 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 Applied to Median String Search

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

  TotalDistance (***prefix***, ***DNA***) > ***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

```
BranchAndBoundMedianStringSearch(DNA,t,n,ℓ)
s ← (1,…,1)
bestDistance ← ∞
 i ← 1
while i > 0
  if i < ℓ
      prefix ←  string corresponding to the first i nucleotides of s
      optimisticDistance ← TotalDistance(prefix,DNA)
       if optimisticDistance > bestDistance
              (s, i ) ← Bypass(s,i, ℓ, 4)
       else
              (s, i ) ← NextVertex(s, i, ℓ, 4)
  else
       word ← nucleotide string corresponding to s
       if  TotalDistance(s,DNA) < bestDistance
              bestDistance ← TotalDistance(word, DNA)
              bestWord ← word
      (s,i ) ← NextVertex(s,i,ℓ, 4)
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 $\boldsymbol{w}$, divided into two parts at point $\boldsymbol{i}$
  - $\boldsymbol{u}$: prefix $w_1, \ldots, w_i$,
  - $\boldsymbol{v}$: suffix $w_{i+1}, \ldots, w_\ell$
- Find minimum distance for $\boldsymbol{u}$ in a sequence
- No instances of $\boldsymbol{u}$ in the sequence have distance less than the minimum distance
- Note this doesn't tell us anything about whether $\boldsymbol{u}$ is part of any motif. We only get a minimum distance for prefix $\boldsymbol{u}$

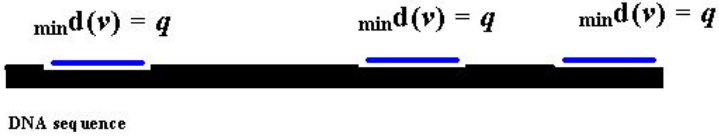
Improving the Bounds (cont'd)

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

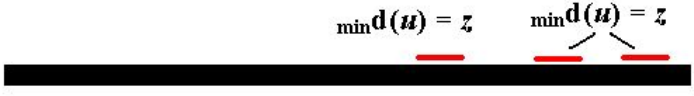
- Since ***u*** and ***v*** are two substrings of ***w***, and included in motif ***w***, we can assume that the minimum distance of ***u*** plus minimum distance of ***v*** can only be less than the minimum distance for ***w***

Better Bounds

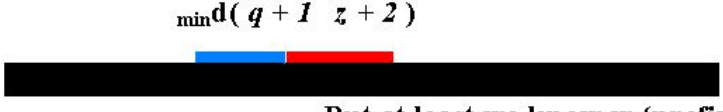
Searching for prefix *V*
We may find many instances of prefix *V* with a minimum distance *q*



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)

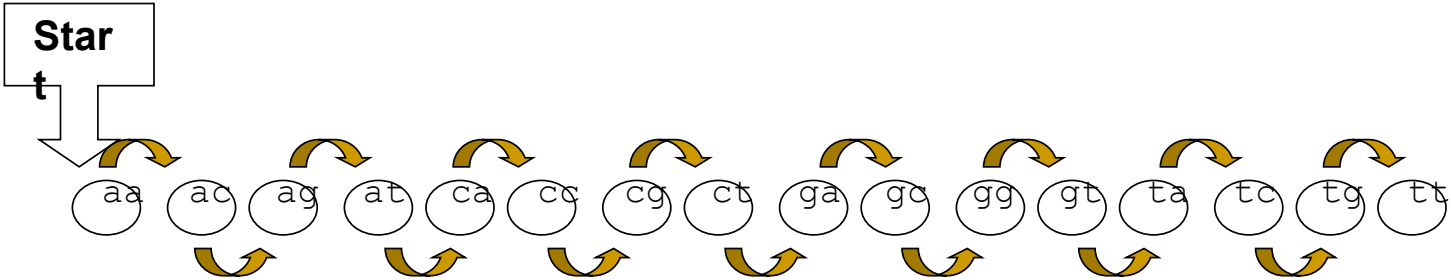
- If d(***prefix***) + d(***suffix***) $\geq$ ***bestDistance***:
  - Motif ***w*** (*prefix.suffix)* cannot give a better (lower) score than d(***prefix***) + d(***suffix***)
  - In this case, we can **ByPass()**

Better Bounded Median String Search

```
ImprovedBranchAndBoundMedianString(DNA,t,n,ℓ)
  s = (1, 1, …, 1)
  bestdistance = ∞
  i = 1
  while i > 0
      if i < ℓ
        prefix = nucleotide string corresponding to (s1, s2, s3, …, si )
        optimisticPrefixDistance = TotalDistance (prefix, DNA)
          if (optimisticPrefixDistance < bestsubstring[ i ])
            bestsubstring[ i ] = optimisticPrefixDistance
            if (ℓ- i < i )
              optimisticSufxDistance = bestsubstring[ℓ-i ]
            else
              optimisticSufxDistance = 0;
            if optimisticPrefixDistance + optimisticSufxDistance ≥ bestDistance
              (s, i ) = Bypass(s, i, ℓ, 4)
            else
              (s, i ) = NextVertex(s, i, ℓ,4)
      else
        word = nucleotide string corresponding to (s1,s2, s3, …, st)
        if TotalDistance( word, DNA) < bestDistance
          bestDistance = TotalDistance(word, DNA)
          bestWord = word
          (s,i) = NextVertex(s, i,ℓ, 4)
return bestWord
```

Linked List

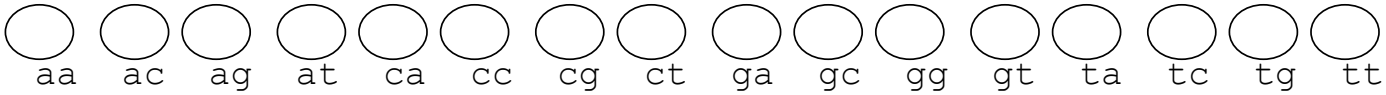
- Suppose $\ell = 2$



- 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



Search Tree