APA: Advanced Programming, Algorithms and Data Structures

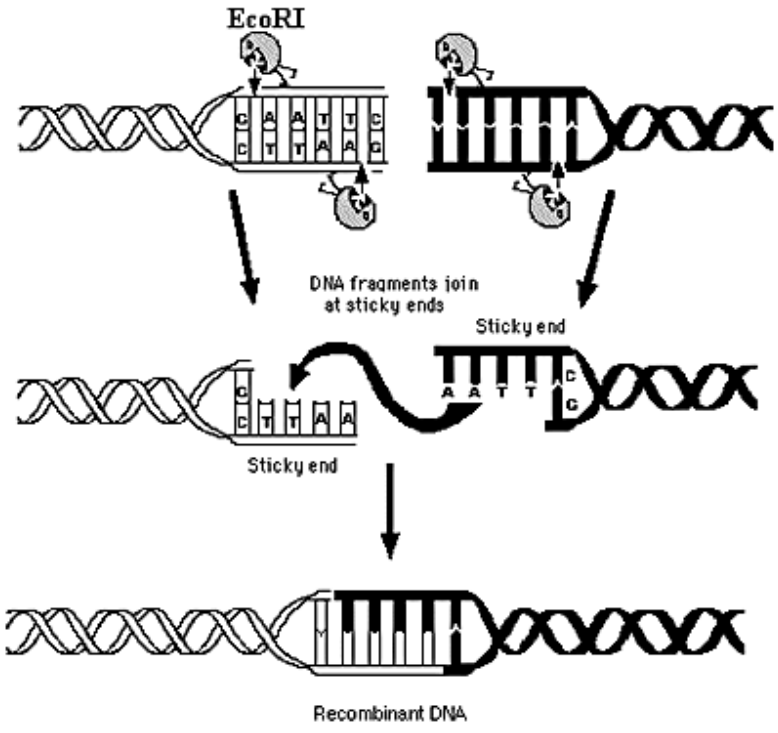
Emre Güney, PhD

Master in Bioinformatics for Health Sciences
Universitat Pompeu Fabra

Lecture 11

November 5, 2019

Recall concept of a restriction enzyme ...



Restriction Enzyme Action of EcoRI

A *restriction enzyme* surrounds DNA molecule at specific point, called *restriction site* (sequence GAATTC in this case). It cuts one strand of DNA double helix at one point and second strand at a different, complementary point (between G and A base).

The separated pieces have single-stranded *sticky ends*, which allow complementary pieces to combine.

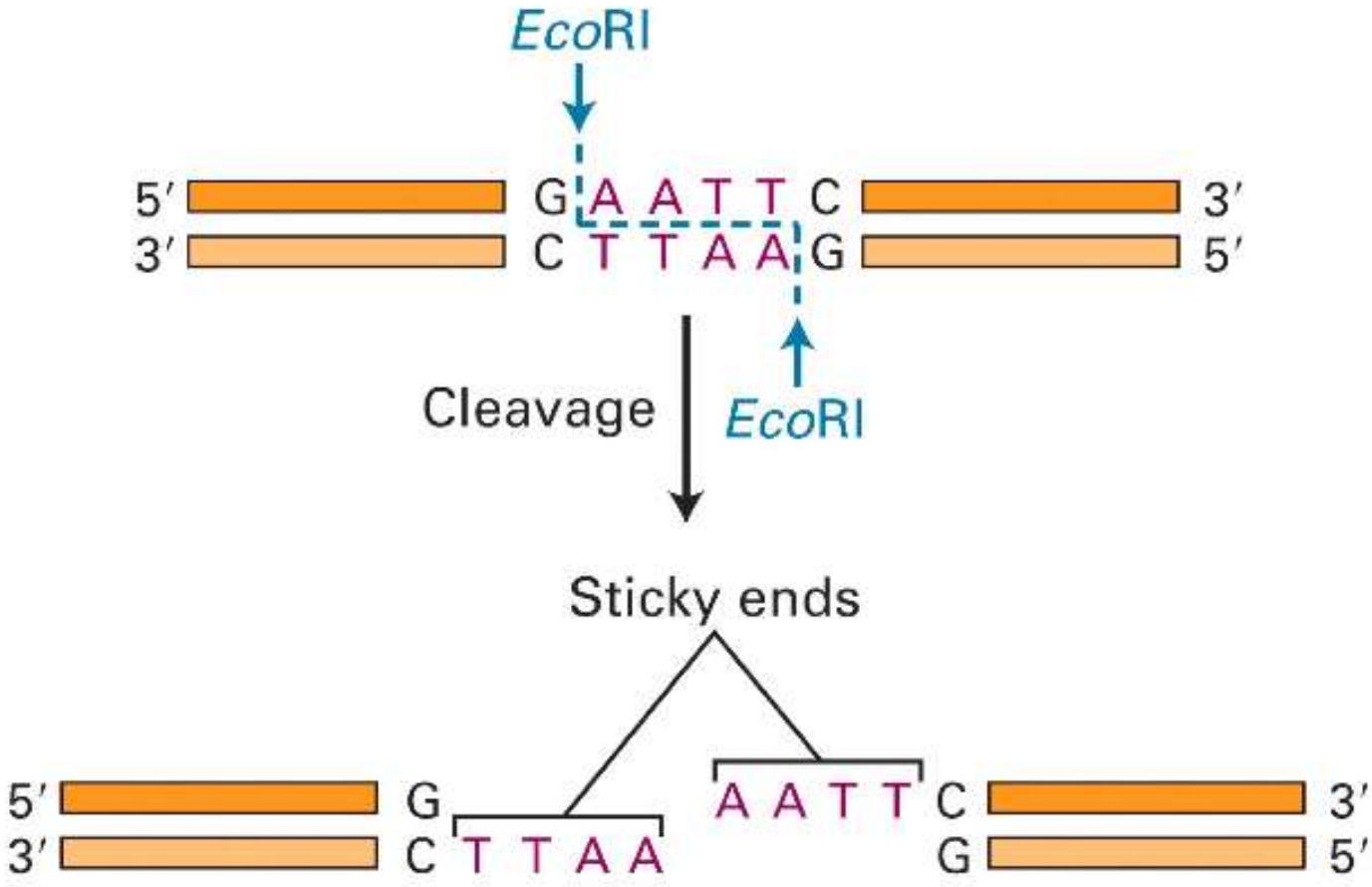
Note that $\overline{\text{GAATTC}} \rightarrow \text{CTTAAG} \rightarrow \text{GAATTC}$ (i.e., palindrome).

http://www.accessexcellence.org/AB/GG/restriction.html

Source: Lehigh Uni. - Daniel Lopresti

LEHIGH UNIVERSITY

"Molecular scissors"



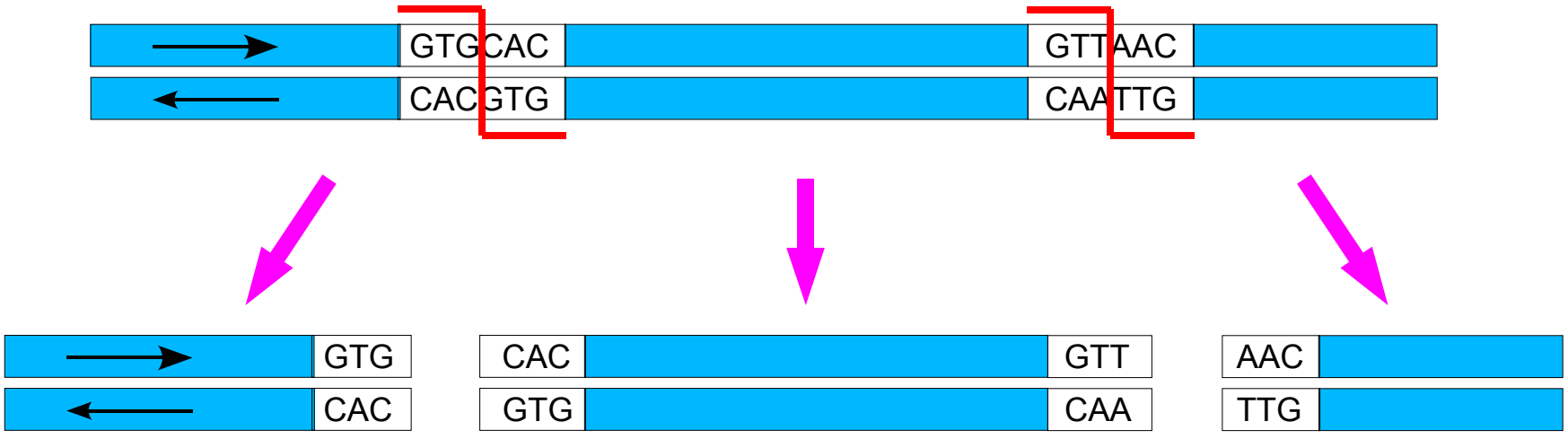
Molecular Cell Biology, 4th edition

Source: Lehigh Uni. - Daniel Lopresti

LEHIGH UNIVERSITY

History of restriction enzymes

In 1970, Hamilton Smith discovered that the restriction enzyme *Hin*dII cleaves DNA at every occurrence of the sequence GTGCAC or GTTAAC.



It is important to note, however, that this discovery occurred decades before we were able to sequence DNA.

Source: Lehigh Uni. - Daniel Lopresti

LEHIGH UNIVERSITY

Uses of restriction enzymes

- Recombinant DNA technology
- Cloning
- cDNA / genomic library construction
- DNA mapping

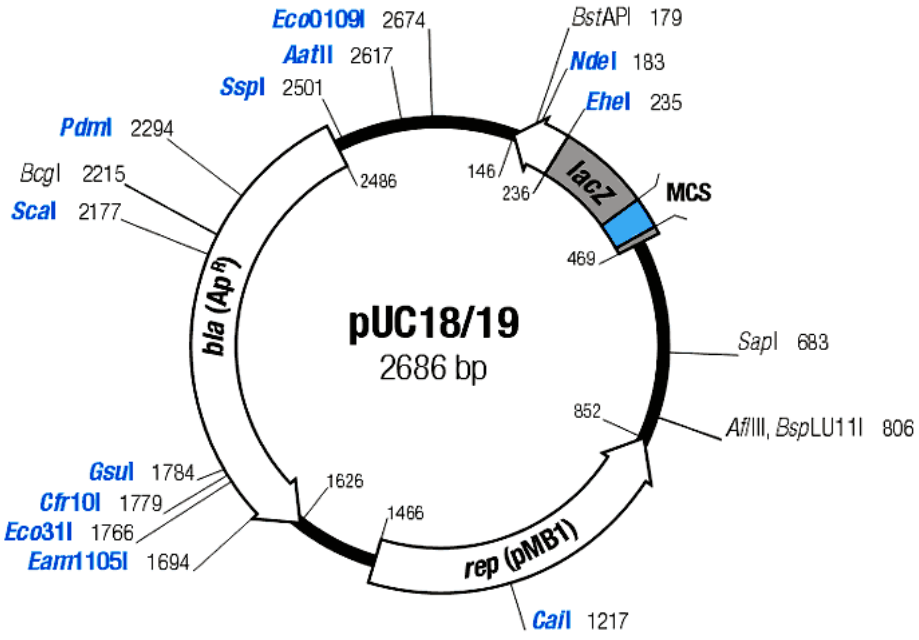
http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

LEHIGH UNIVERSITY

Restriction maps

- A map showing positions of restriction sites in a DNA sequence.
- If DNA sequence is known, then construction of restriction map is a trivial exercise.
- In early days of molecular biology, DNA sequences were often unknown.
- Biologists had to solve problem of constructing restriction maps without knowing DNA sequences.



http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

Measuring length of restriction fragments

- Restriction enzymes break DNA into restriction fragments.
- Gel electrophoresis is process for separating DNA by size and measuring sizes of restriction fragments.
- Can separate DNA fragments that differ in length by only 1 nucleotide for fragments up to 500 nucleotides long.

Gel electrophoresis:

- DNA fragments injected into gel positioned in electric field.
- DNA is negatively charged near neutral pH: ribose phosphate backbone of each nucleotide is acidic.
- DNA molecules move towards positive electrode.

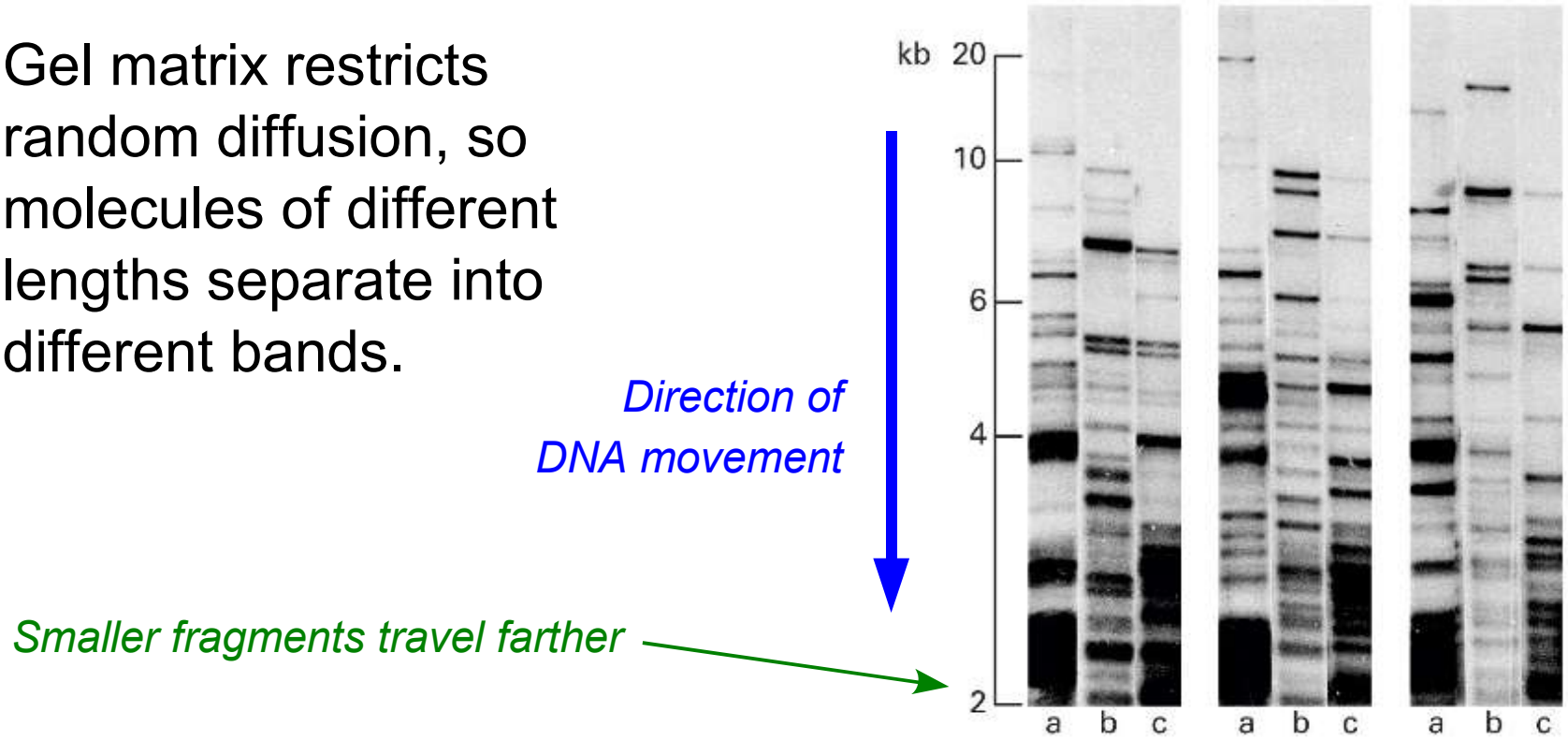
http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

LEHIGH UNIVERSITY

Gel electrophoresis (continued)

- DNA fragments of different lengths separated based on size.
- Smaller molecules move through gel matrix more readily than larger molecules.
- Gel matrix restricts random diffusion, so molecules of different lengths separate into different bands.



http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

Detecting DNA bands

Autoradiography:

- DNA is radioactively labeled
- Gel is laid against sheet of photographic film in the dark, exposing film at positions where DNA is present.

Fluorescence:

- Gel is incubated with solution containing fluorescent dye ethidium.
- Ethidium binds to DNA.
- DNA lights up when gel is exposed to ultraviolet light.

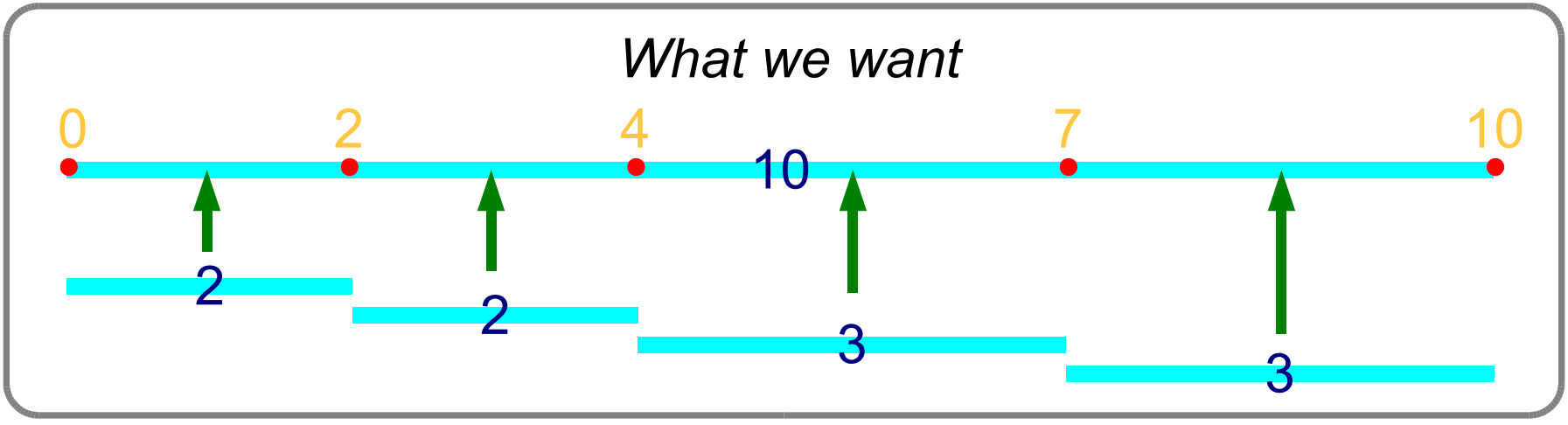
http://www.bioalgorithms.info

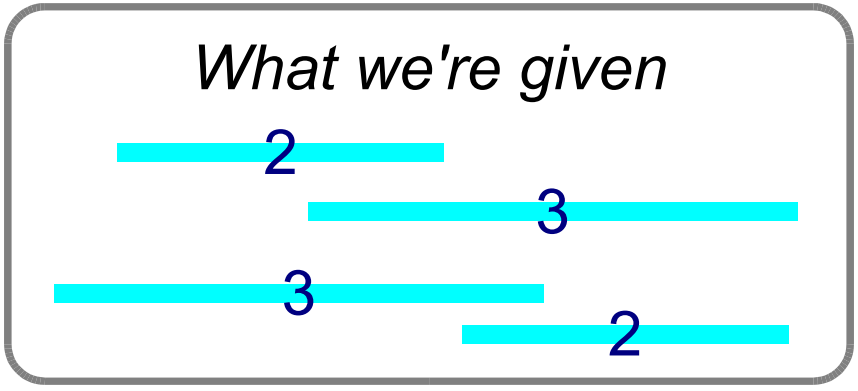
Source: Lehigh Uni. - Daniel Lopresti

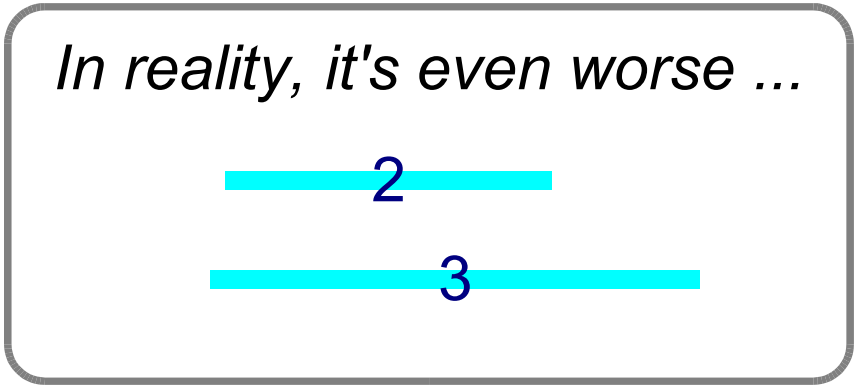
LEHIGH UNIVERSITY

Complete digest

Normally, we might think of a restriction enzyme cutting at all possible restriction sites. This is known as a *complete digest* :



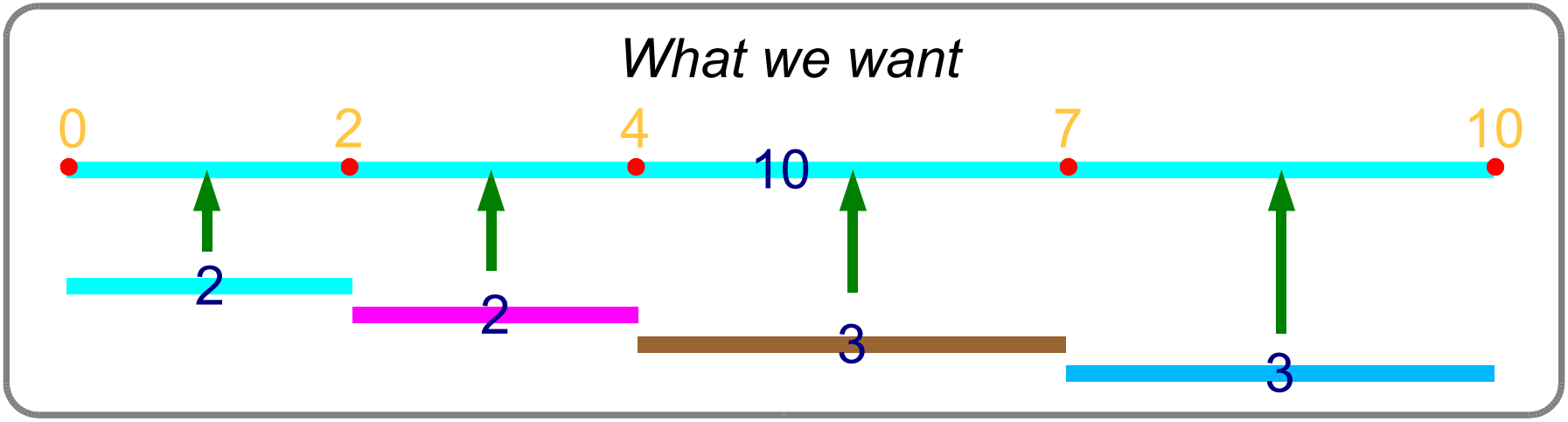




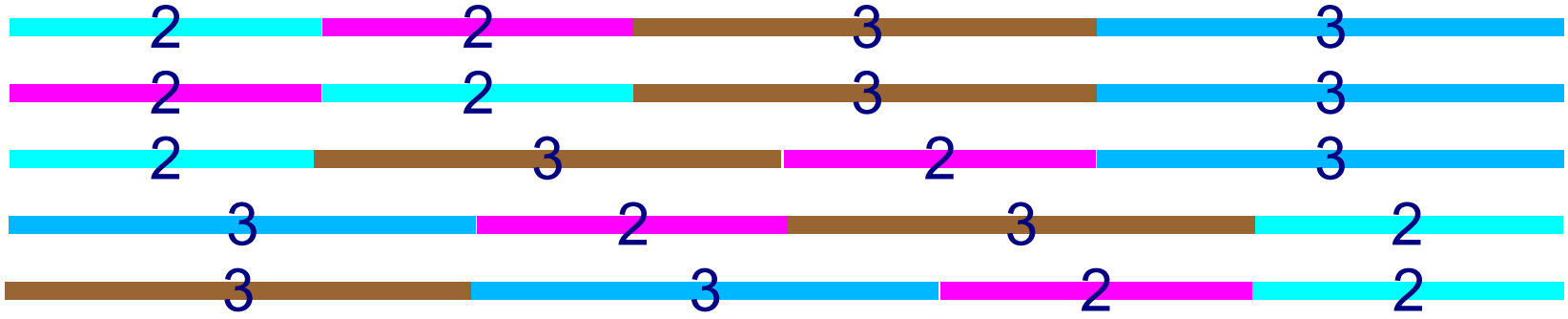
Source: Lehigh Uni. - Daniel Lopresti

Complete digest

The situation here is essentially hopeless ...



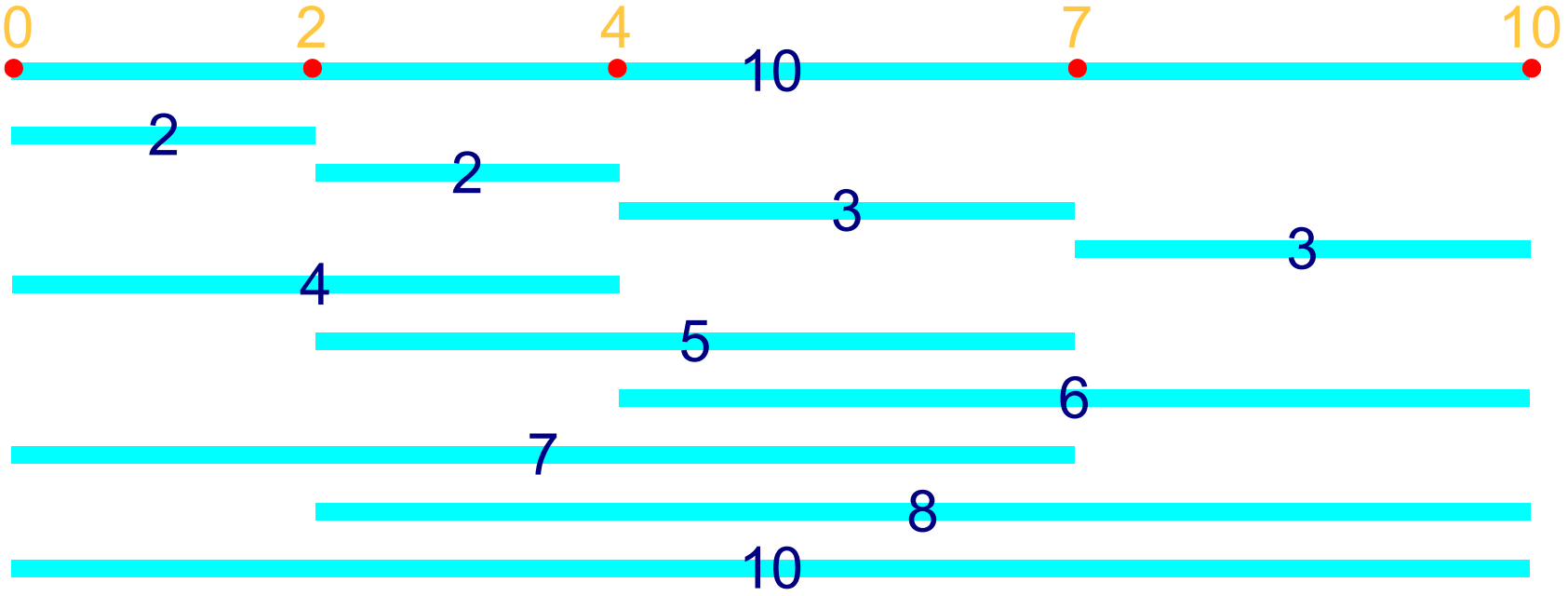
Even if we somehow knew there were two different fragments of lengths 2 and 3, there are still a large number of solutions:



Source: Lehigh Uni. - Daniel Lopresti

Partial digest

Solving a complete digest is essentially impossible – there's too much ambiguity. Fortunately, if we don't let the enzyme digest the DNA completely, we obtain much more information:



We have fragment lengths corresponding to all possible distances between sites. This is known as a *partial digest* .

Source: Lehigh Uni. - Daniel Lopresti

Some definitions

A *multiset* is a set that allows duplicate elements. E.g.,

$$\{2, 2, 3, 3, 4, 5\}$$

Let $X = \{x_1 = 0, \mathbf{x}_2, \ldots, x_n\}$ be a set of n points along a line, and define ΔX to be the multiset of all pairwise distances:

$$\Delta X = \{x_j - x_i \ : \ 1 \le i < j \le n\}$$

For example, if $X = \{0, 2, 4, 7, 10\}$, then ΔX is:

$$\{2, 2, 3, 3, 4, 5, 6, 7, 8, 10\}$$

In general, if X contains n points, how big will ΔX be?

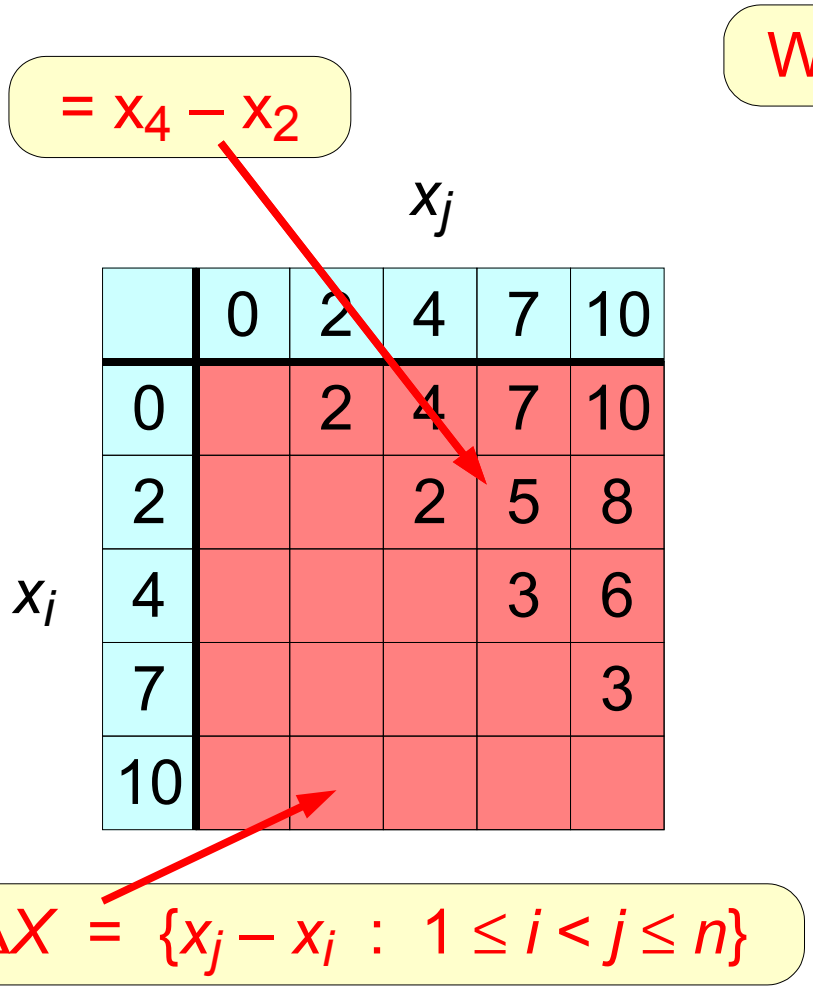
$$\binom{n}{2} = \frac{n(n-1)}{2}$$

"n choose 2"

Source: Lehigh Uni. - Daniel Lopresti

Some definitions

We can build a table of ΔX values given X as input.



What we want

What we're given

|X|	|ΔX|
2	1
3	3
4	6
5	10
10	45
100	4,950

The input gets big fast

Source: Lehigh Uni. - Daniel Lopresti

Formal definition of the problem

The Partial Digest Problem.

Given all pairwise distances between points on a line, reconstruct the positions of those points.

Input: The multiset of pairwise distances L.

Output: A set X of integers such that $\Delta X = L$.

Note: If there are $\binom{n}{2}$ integers in L, there are n integers in X.

Interesting sidenote: this problem is also known as the "Turnpike problem": given all of the distances between exits on a turnpike, determine the layout of the exits.

Source: Lehigh Uni. - Daniel Lopresti

LEHIGH UNIVERSITY

Homometric sets

A less trivial example is the following:

{0, 1, 3, 4, 5, 7, 12, 13, 15}

{0, 1, 3, 8, 9, 11, 12, 13, 15}

	0	1	3	4	5	7	12	13	15
0		1	3	4	5	7			
1			2	3	4	6	11	12	14
3				1	2	4	9	10	12
4					1	3	8	9	11
5						2	7	8	10
7							5	6	8
12									
13									
15									

	0	1	3	8	9	11	12	13	15
0			3	8	9	11	12	13	15
1			2	7	8	9	11	12	14
3				5	6	8	9	10	12
8					1	3	4	5	7
9						2	3	4	6
11							1	2	4
								1	3
									2

Multisets are identical

Two sets *A* and *B* are *homometric* if $\Delta A = \Delta B$

Source: Lehigh Uni. - Daniel Lopresti

Homometric sets

As we saw earlier, the complete digest problem is hopeless because there's too much ambiguity.

Unfortunately, we can't completely eliminate this here either.

For any integer v and set A, note ΔA is equal to $\Delta(A \oplus \{v\})$:

$$\Delta(A \oplus \{v\}) = \{a + v : a \in A\}$$

Also, $\Delta A = \Delta(-A)$.

E.g., if A = {0, 2, 4, 7, 10}, then the same partial digest is also produced by:

$$A \oplus \{100\} = \{100, 102, 104, 107, 110\}$$

and

$$-A = \{-10, -7, -4, -2, 0\}$$

These cases are utterly indistinguishable

Source: Lehigh Uni. - Daniel Lopresti

A brute force solution ...

Given all pairwise distances between points on a line, reconstruct the positions of those points.

Input: The multiset of pairwise distances L.

Output: A set X of integers such that $\Delta X = L$.

```
BruteForcePDP ( L, n )
M ← maximum element in L
for every set of n – 2 integers 0 < x_2 < ... < x_{n-1} < M
    X ← {0, x_2, ..., x_{n-1}, M}
    Form ΔX from X
    if ΔX = L
        return L
output "No Solution"
```

How many such sets are there? $\binom{M-1}{n-2}$

Time complexity is $O(M^{n-2})$

Source: Lehigh Uni. - Daniel Lopresti

A better brute force solution ...

It seems like overkill to try all possible subsets of integers. Some integers certainly cannot occur in a given problem. If

$$L = \{2, 2, 3, 3, 4, 5, 6, 7, 8, 10\}$$

Then we shouldn't bother trying $\mathbf{x}_i = 9$ because it can't occur.

```
AnotherBruteForcePDP ( L, n )
M ← maximum element in L
for every set of n – 2 integers 0 < x2 < ... < xn-1 < M from L
    X ← {0, x2, ..., xn-1, M}
    Form ΔX from X
    if ΔX = L
        return L
output "No Solution"
```

How many such sets are there? $\binom{|L|}{n-2}$

Since $|L| = \frac{n(n-1)}{2}$ time complexity is $O(n^{2n-4})$

Source: Lehigh Uni. - Daniel Lopresti

Brute force solutions

Both of these approaches require exponential time, so neither is an effective solution to the general partial digest problem.

For BruteForcePDP, time complexity is $O(M^{n-2})$.

For AnotherBruteForcePDP, time complexity is $O(n^{2n-4})$.

Still, is it possible that one is better than the other?

Consider this example:

$$L = \{2, 998, 1000\}$$

Here, $M = 1000$, so BruteForcePDP takes a very long time.
On the other hand, $n = 3$, so AnotherBruteForcePDP is fast.
Ratio should be about 100:1.

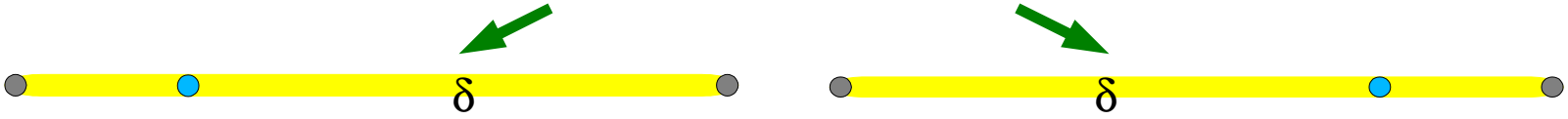
Source: Lehigh Uni. - Daniel Lopresti

A more intelligent solution

Developed by Skiena in 1990, this approach works from the largest possible distances down to smaller ones.

(1) First, find largest distance in L. This must determine two outermost points. Delete this distance from L.

(2) Select next largest distance in L, call it δ. One of the points responsible for δ must be an outermost point.



Case 1 (right) *Case 2 (left)*

(3) Evaluate whether remaining distances are consistent with either choice. Repeat recusively, backtracking if necessary.

Source: Lehigh Uni. - Daniel Lopresti

A more intelligent solution

Consider this example:

$$L = \{2, 2, 3, 3, 4, 5, 6, 7, 8, 10\}$$

Since $|L| = \frac{n(n-1)}{2} = 10$ we know that n must be 5.

Hence, solution will consist of five points $\mathbf{x_1 = 0}$, $\mathbf{x_2}$, $\mathbf{x_3}$, $\mathbf{x_4}$, $\mathbf{x_5}$.

Because 10 is largest distance in L, we set $\mathbf{x_5} = 10$, giving us:

$X = \{0, 10\}$ $L = \{2, 2, 3, 3, 4, 5, 6, 7, 8\}$

Now largest distance is 8. Either $\mathbf{x_4} = 8$ or $\mathbf{x_2} = 2$. For latter:

$X = \{0, 2, 10\}$ $L = \{2, 3, 3, 4, 5, 6, 7\}$

Source: Lehigh Uni. - Daniel Lopresti

A more intelligent solution

$X = \{0, 2, 10\}$ $L = \{2, 3, 3, 4, 5, 6, 7\}$

Now largest distance is 7. Either $\mathbf{x}_4 = 7$ or $\mathbf{x}_3 = 3$.

In latter case, $\mathbf{x}_3 - \mathbf{x}_2 = 1$ should be in set L, but it isn't. Hence, the former case must be the correct one and $\mathbf{x}_4 = 7$. We must remove following distances from L:

$\mathbf{x}_5 - \mathbf{x}_4 = 3$ $\mathbf{x}_4 - \mathbf{x}_2 = 5$ $\mathbf{x}_4 - \mathbf{x}_1 = 7$

$X = \{0, 2, 7, 10\}$ $L = \{2, 3, 4, 6\}$

Now 6 is largest distance. Either $\mathbf{x}_3 = 4$ or $\mathbf{x}_3 = 6$.

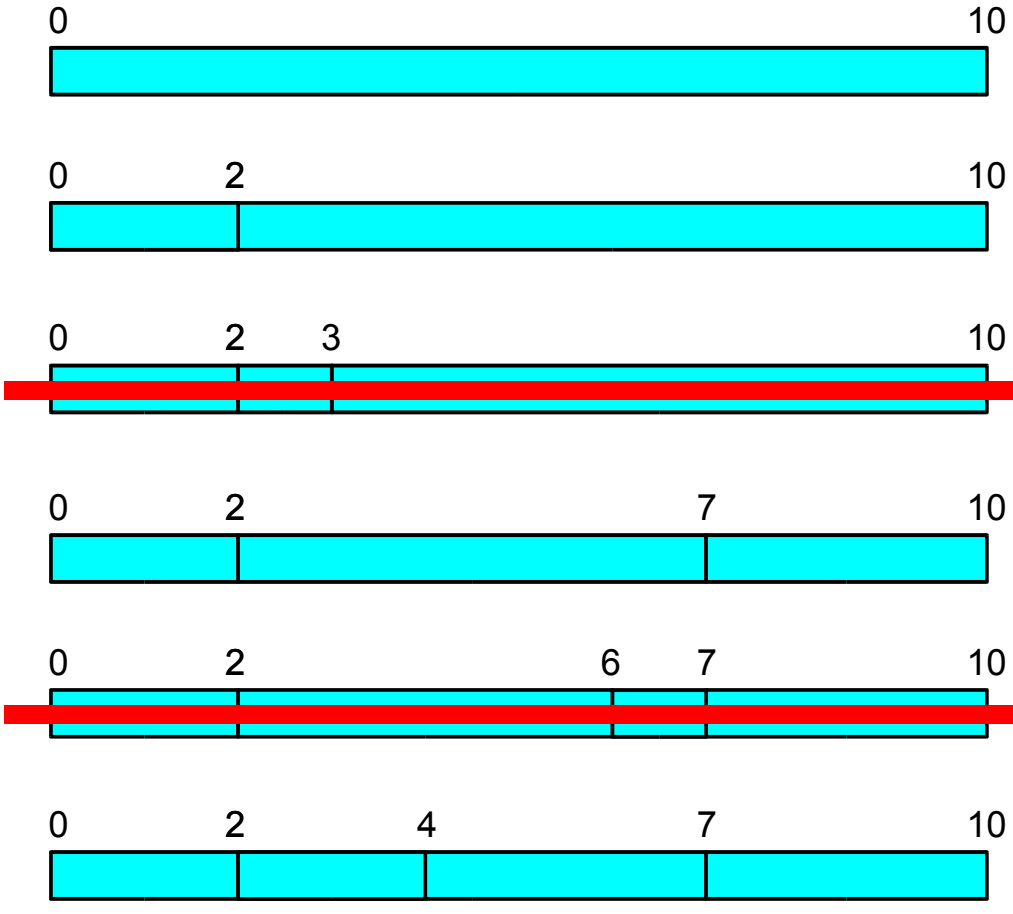
In latter case, $\mathbf{x}_4 - \mathbf{x}_3 = 1$ should be in set L, but it isn't. Hence:

$X = \{0, 2, 4, 7, 10\}$ $L = \{\}$ Done!

Source: Lehigh Uni. - Daniel Lopresti

Branch-and-bound

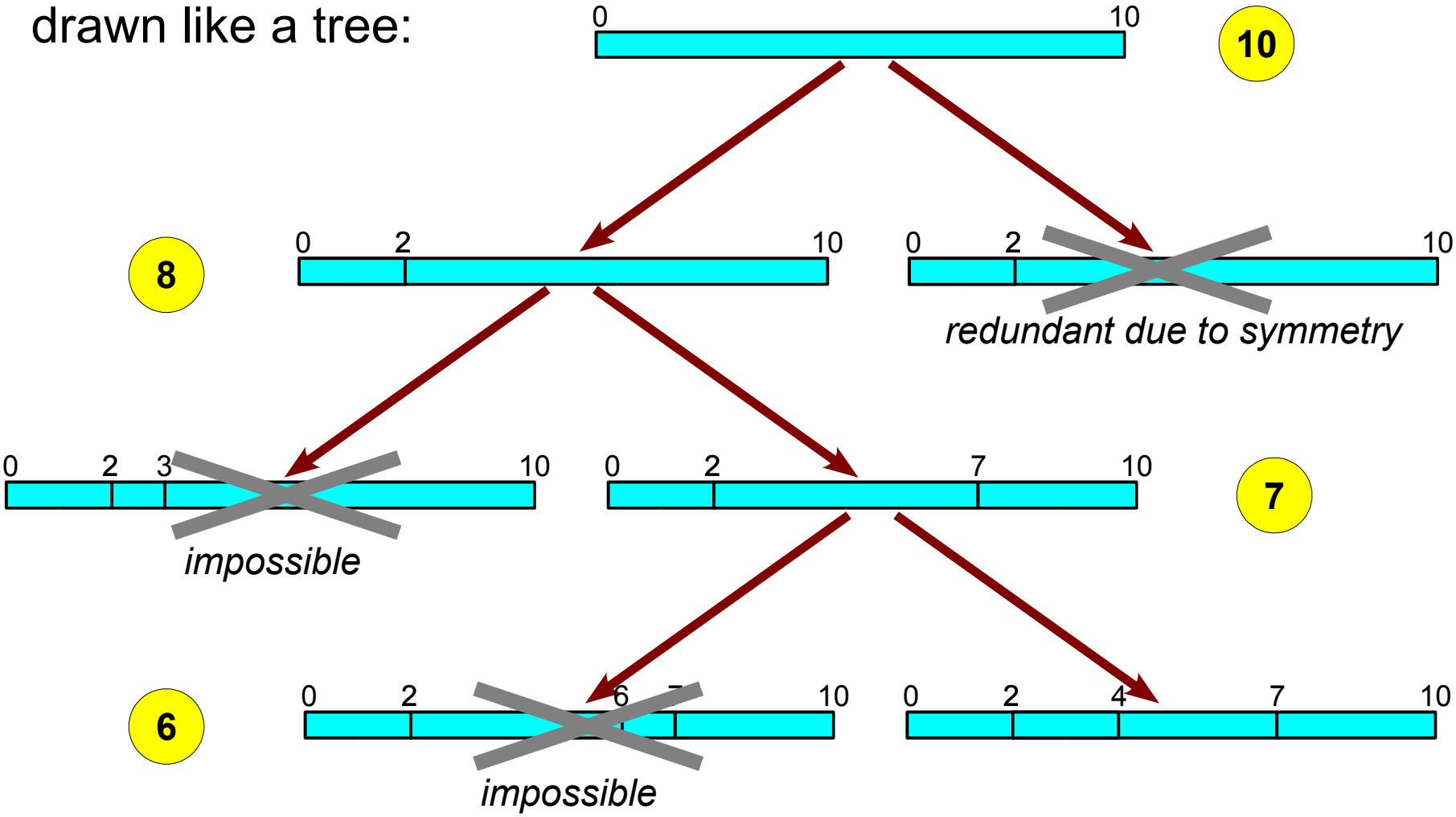
Here's the complete branch-and-bound tree we explored:



Source: Lehigh Uni. - Daniel Lopresti

Branch-and-bound

Or, if you prefer,
drawn like a tree:



Source: Lehigh Uni. - Daniel Lopresti

Partial digest algorithm

```
Place ( L , X )
if L is empty
    output X; return
y ← maximum element in L
if Δ(y, X) ⊆ L
    Add y to X; remove lengths Δ(y, X) from L
    Place(L, X)
    Remove y from X; add lengths Δ(y, X) to L
if Δ(width – y, X) ⊆ L
    Add width – y to X; remove lengths Δ(width – y, X) from L
    Place(L, X)
    Remove y from X; add lengths Δ(y, X) to L
return
```

```
PartialDigest ( L )
width ← maximum element in L
Delete(width, L)
X ← {0, width}
Place(L, X)
```

Removes width from L

Multiset of distances between point & set

Recursively generates all possible solutions

Source: Lehigh Uni. - Daniel Lopresti

Partial digest algorithm

What is the computational efficiency?

So long as only one of the two alternatives (left or right) is viable, algorithm is quite fast. We have to worry about case where both alternatives are viable, however – then number of possible solutions we must explore grows exponentially.

Let $T(n)$ be maximum number of steps taken by PartialDigest on an input of size n.

When only one alternative viable:

$$T(n) = T(n-1) + O(n)$$

When both alternatives viable:

$$T(n) = 2T(n-1) + O(n)$$

Big difference here!

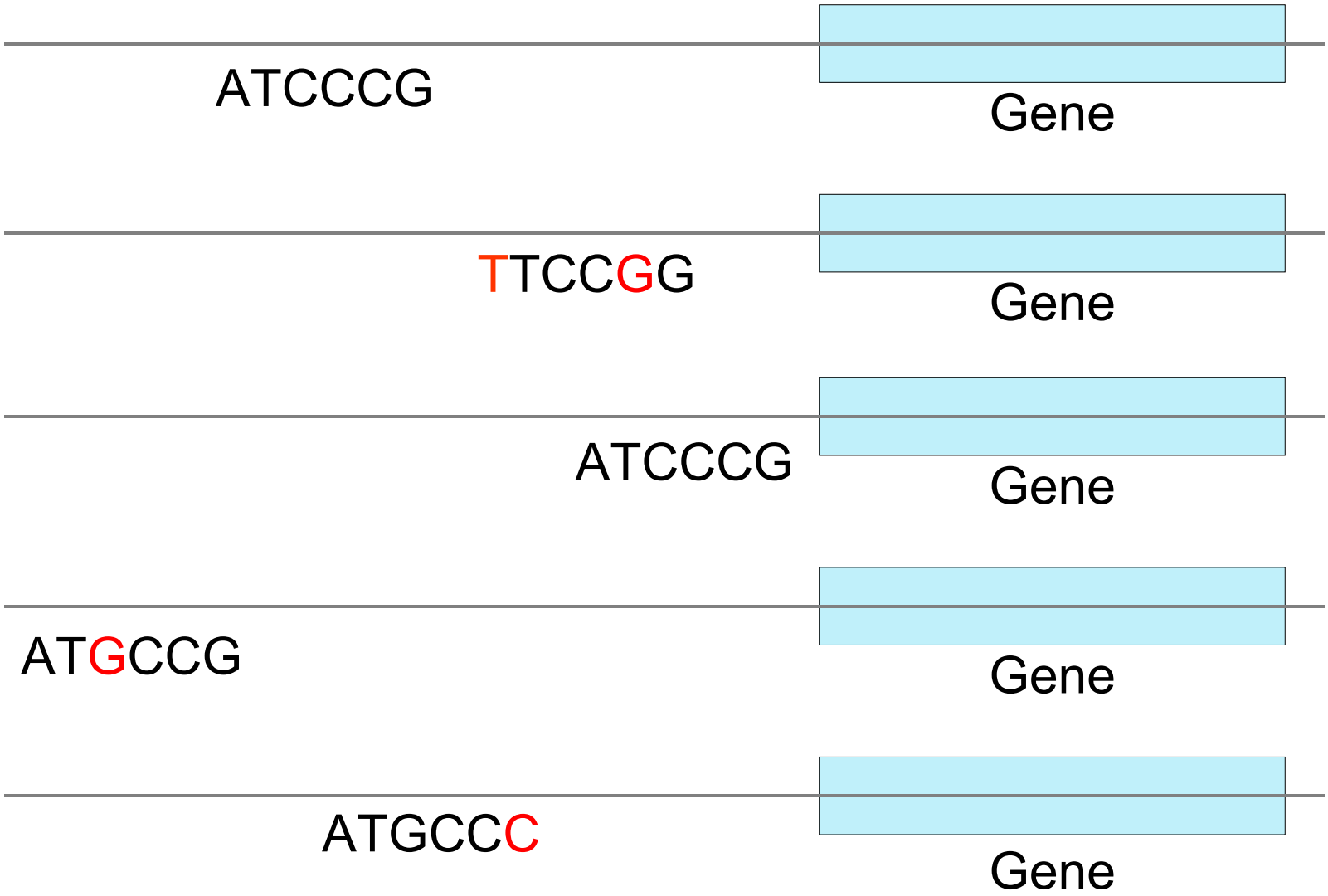
Time complexity is $O(n^2)$

Time complexity is $O(2^n)$

Source: Lehigh Uni. - Daniel Lopresti

LEHIGH UNIVERSITY

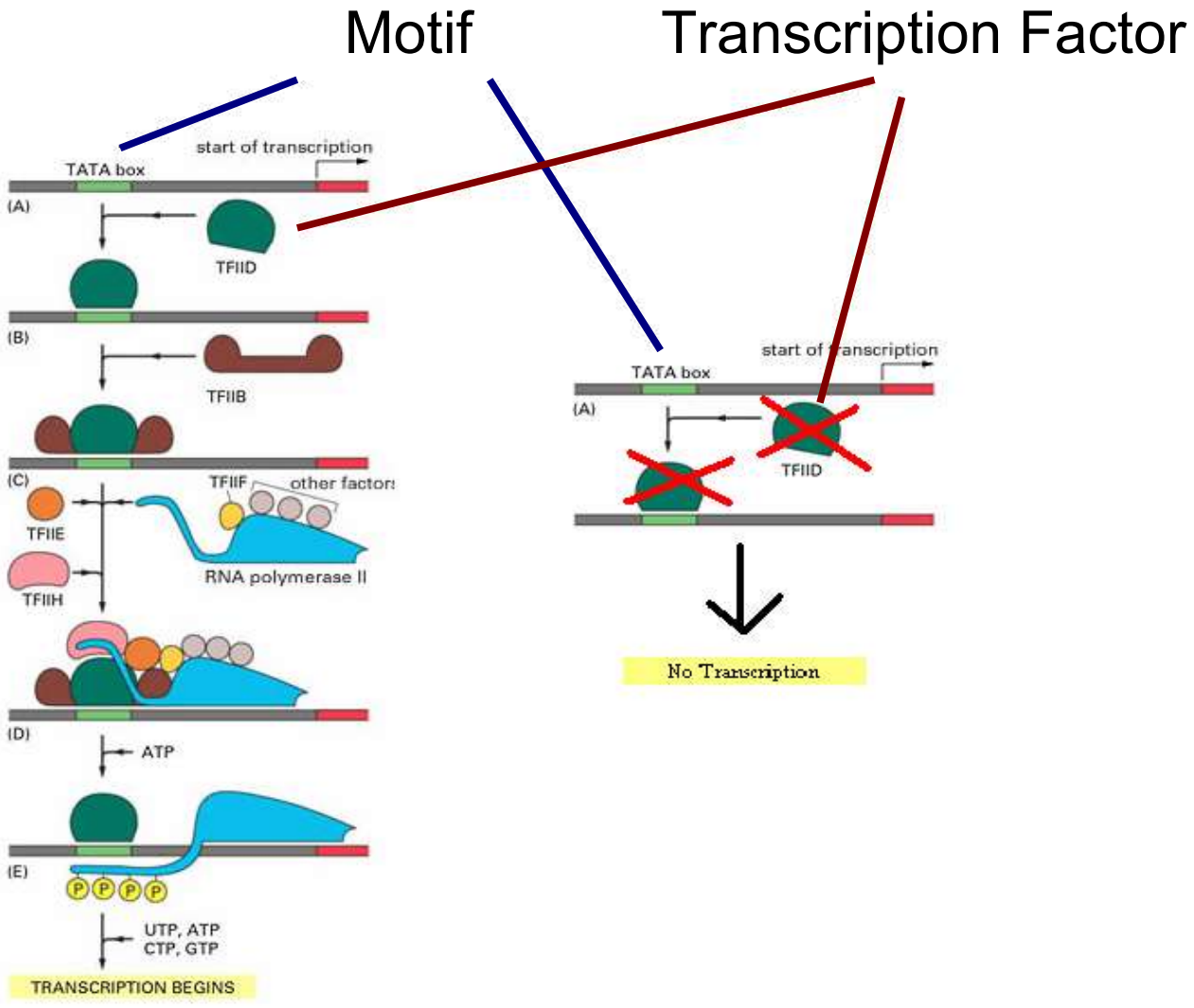
Motifs and transcriptional start sites



http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

Transcription factors and motifs



http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

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 same motif in multiple genes' regulatory regions suggests a regulatory relationship amongst those genes.

Complications

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

http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

LEHIGH UNIVERSITY

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*.

Additional information:

- Hidden sequence is of length 8.
- Pattern is not exactly same in each position because random point mutations may occur in sequences.

http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

The Motif-Finding Problem

The patterns revealed with no mutations:

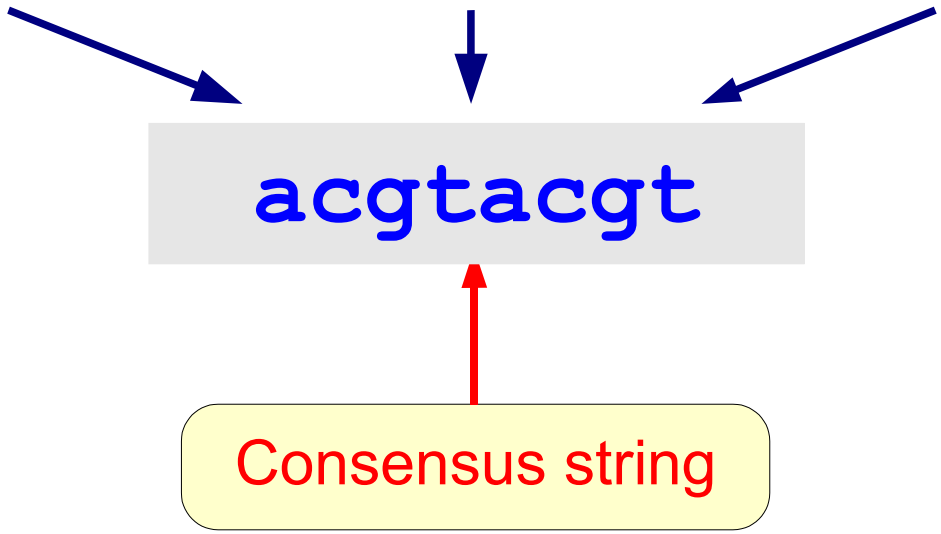
cctgatagacgctatctggctatccacgtacgtaggtcctctgtgcgaatctatgcgtttccaaccat

agtactggtgtacatttgatacgtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc

aaacgtacgtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt

agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatcttacgtacgtataca

ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgttaacgtacgtc



http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

The Motif-Finding Problem

The patterns with 2 point mutations:

cctgatagacgctatctggctatccaggtacttaggtcctctgtgcgaatctatgcgtttccaaccat

agtactggtgtacatttgatccatacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc

aaacgttagtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt

agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatcttacgtccatataca

ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgttaccgtacggc

http://www.bioalgorithms.info

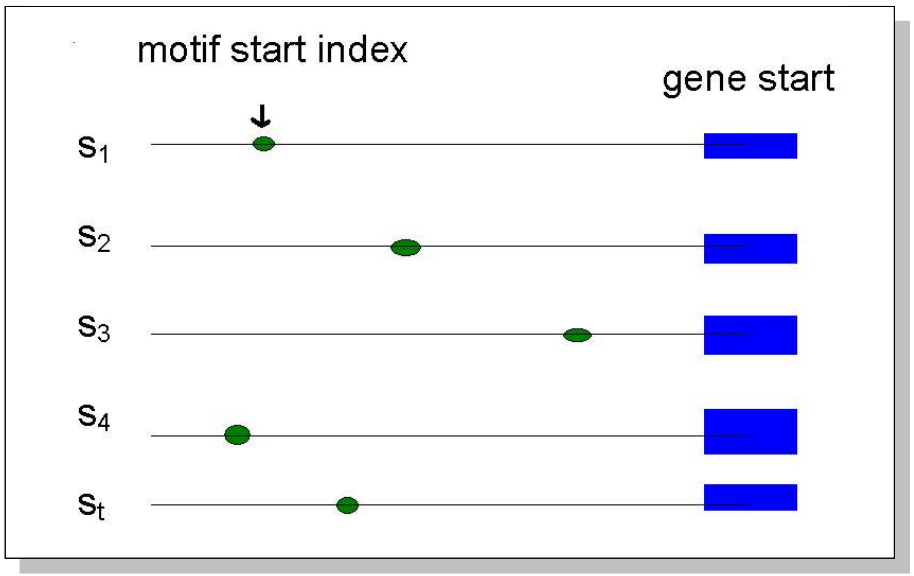
Source: Lehigh Uni. - Daniel Lopresti

Defining motifs

Let's start by saying we know where motif starts in sequence.

Say we have *t* sequences. Motif starting positions can be represented as:

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



http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

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
```

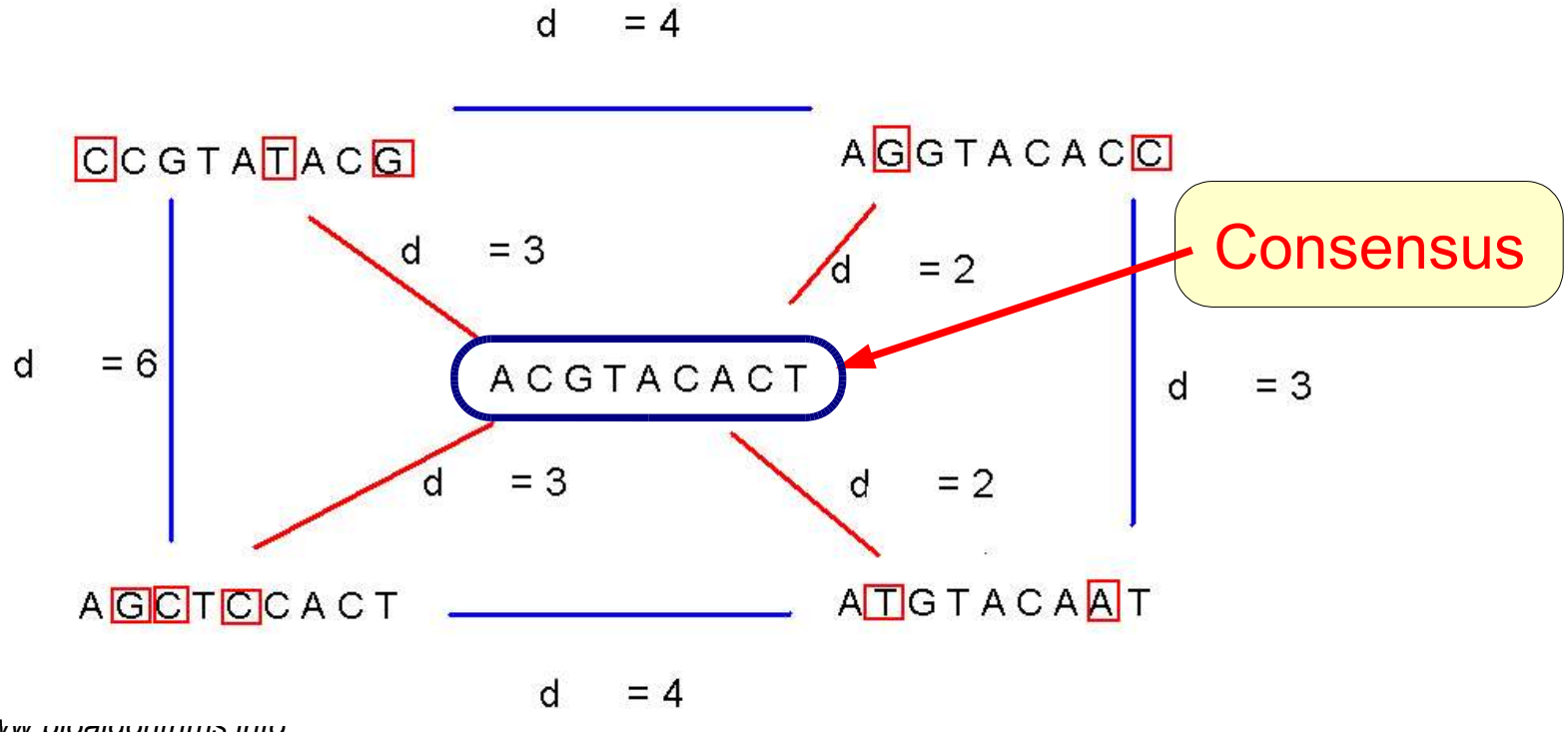
- Line up the patterns by their start indexes:
  $s = (s_1, s_2, s_3, \ldots, s_t)$
- Construct matrix profile with frequencies of each nucleotide in columns.
- Consensus nucleotide in each position has the highest score in column.

http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

Consensus

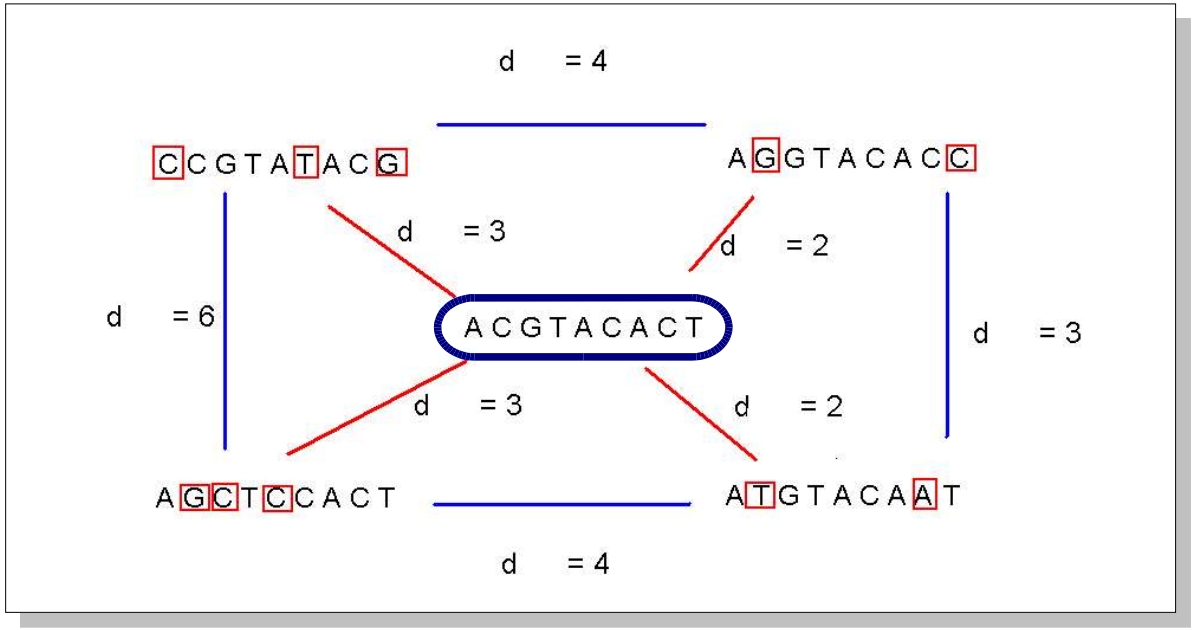
- Think of consensus as an “ancestor” motif, from which mutated motifs emerged.
- Distance between a real motif and consensus sequence is generally less than that for two real motifs.



http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

Evaluating motifs



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

http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

Defining some terms

$t \equiv$ number of DNA sequences

$n \equiv$ length of each DNA sequence

$DNA \equiv$ collection of DNA sequences (t x n array)

$l \equiv$ length of motif (l-mer)

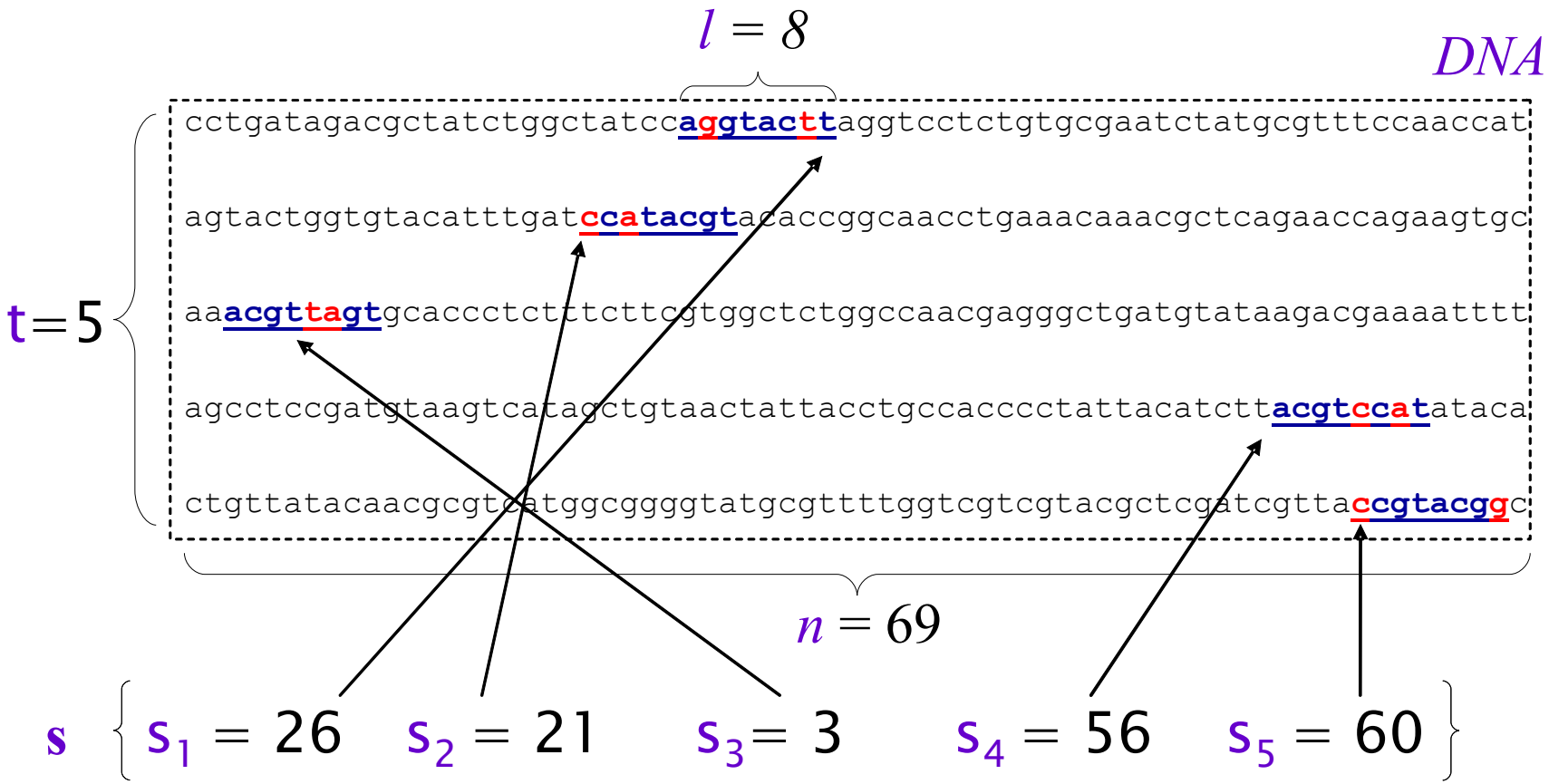
$s_i \equiv$ starting position of an l-mer in sequence i

$(s_1, s_2, s_3, \ldots, s_t) \equiv$ array of motif's starting positions

http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

Parameters



Source: Lehigh Uni. - Daniel Lopresti

Scoring motifs

Given $s = (s_1, s_2, s_3, \ldots, s_t)$ and *DNA, define*:

Score(*s*,*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
	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

a	c	g	t	a	c	g	t

Score

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

http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

Formal definition of the Motif-Finding Problem

- If starting positions $s = (s_1, s_2, s_3, \ldots, s_t)$ are given, finding consensus is easy, even with mutations in sequences because we can construct profile to find motif (consensus).
- But … starting positions $s = (s_1, s_2, s_3, \ldots, s_t)$ are usually not given. How can we find "best" profile matrix?

The Motif-Finding Problem.

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 pattern to find.

Output: An array of t starting positions $s = (s_1, s_2, s_3, \ldots, s_t)$ that maximizes *Score*(*s*,*DNA*).

http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

The Motif-Finding Problem: Brute Force Solution

- Compute scores for each possible combination of starting positions $s = (s_1, s_2, s_3, \ldots, s_t)$.
- Best score will determine best profile and hence consensus pattern in DNA.
- Goal is to maximize Score(s,DNA) by varying starting positions s_i, where:

$s_i = [1, \ldots, n - l + 1]$ ← All possible starting positions

$i = [1, \ldots, t]$ ← All sequences

http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

The Motif-Finding Problem: Brute Force Solution

```
BruteForceMotifSearch (DNA, t, n, l)
bestScore ← 0
for each s = (s1, s2, s3, …, st) from (1, 1, ..., 1)
                                  to (n – l + 1, ..., n – l + 1)
    if (Score(s, DNA) > bestScore)
        bestScore ← Score(s, DNA)
        bestMotif ← (s1, s2, s3, …, st)
return bestMotif
```

http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

Running time of BruteForceMotifSearch

- Varying $(n - l + 1)$ positions in each of t sequences, we are looking at $(n - l + 1)^t$ sets of starting positions.
- For each set of starting positions, scoring function performs lt operations, so complexity is $lt(n - l + 1)^t = O(ltn^t)$.
- That means that for $t = 8$, $n = 1000$, $l = 10$ we must perform approximately 10^{20} computations.

It will take billions years!

http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

LEHIGH UNIVERSITY

Taking a different perspective ...

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:

- $d_H(v, w)$ is the number of nucleotide pairs that do not match when *v* and *w* are aligned. For example:

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

http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

LEHIGH UNIVERSITY

Total distance: An example

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

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

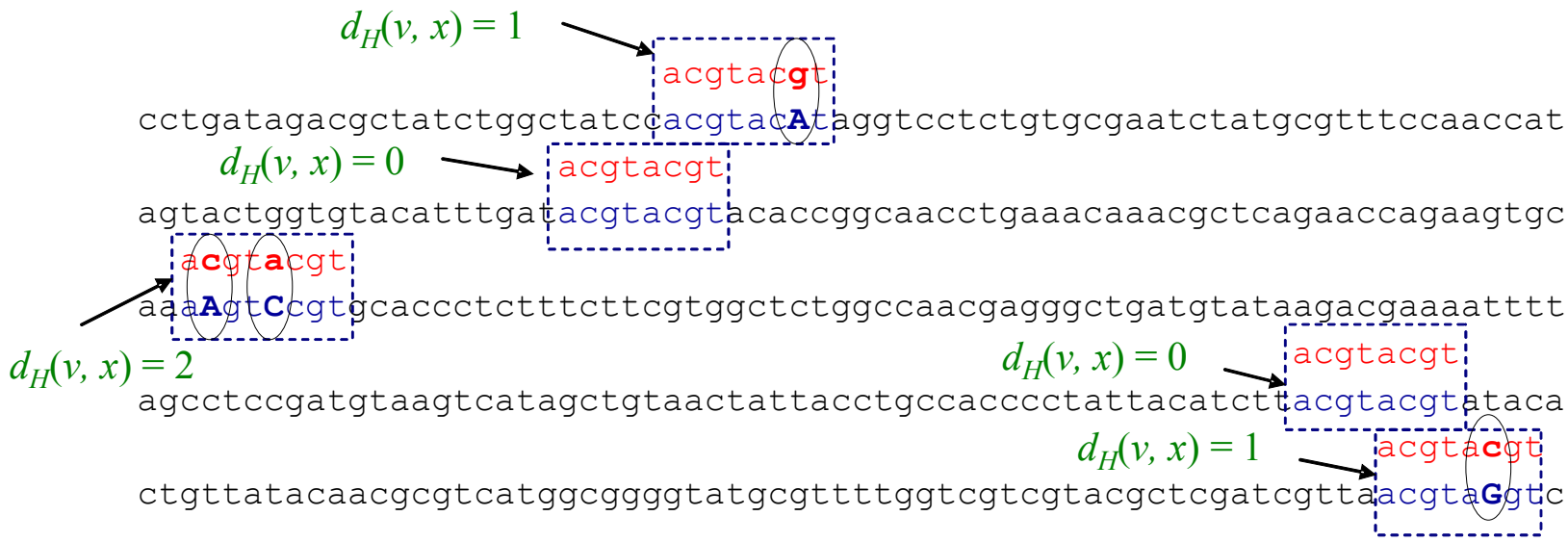
TotalDistance(*v*, *DNA*) = 0

http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

Total distance: An 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

http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

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 , $(1 < s_i < n - l + 1)$.
- 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*) = $\min_s d_H(v, s)$, where s is the set of starting positions $(s_1, s_2, s_3, \ldots, s_t)$.

http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

Formal definition of the Median String Problem

The Median String Problem.

Given a set of DNA sequences, find a median string.

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

Output: A string *v* of *l* nucleotides that minimizes TotalDistance(*v*, *DNA*) over all strings of that length.

http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

LEHIGH UNIVERSITY

Median String search algorithm

```
MedianStringSearch (DNA, t, n, l)
bestWord ← aa…aa
bestDistance ← ∞
for each l-mer s from aa…aa to tt…tt
    if TotalDistance(s,DNA) < bestDistance
        bestDistance ← TotalDistance(s,DNA)
        bestWord ← s
return bestWord
```

- Motif-Finding is a maximization problem while Median String is a minimization problem.
- However, the two problems are computationally equivalent.
- Need to show minimizing TotalDistance ≡ maximizing Score.

http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

Looking for the same thing

	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

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	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: minimization on right side is equivalent to maximization on left.

http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

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. This is $(n - l + 1)^t$ combinations!!!
- The Median String Problem needs to examine all 4^l combinations for v. This number is relatively smaller

Source: Lehigh Uni. - Daniel Lopresti

LEHIGH UNIVERSITY

Recall BruteForceMotifSearch ...

```
BruteForceMotifSearch (DNA, t, n, l)
bestScore ← 0
for each s = (s1, s2, s3, …, st) from (1, 1, ..., 1)
                                 to (n – l + 1, ..., n – l + 1)
    if (Score(s, DNA) > bestScore)
        bestScore ← Score(s, DNA)
        bestMotif ← (s1, s2, s3, …, st)
return bestMotif
```

How can we perform this line?

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

http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

Median String: improving the running time

```
MedianStringSearch (DNA, t, n, l)
bestWord ← aa…aa
bestDistance ← ∞
for each l-mer s from aa…aa to tt…tt
    if TotalDistance(s,DNA) < bestDistance
        bestDistance ← TotalDistance(s,DNA)
        bestWord ← s
return bestWord
```

Consider this line

- 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?

http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

Alternative representation of the search space

- Let a = 1, c = 2, g = 3, t = 4.
- Then the sequences from aa...aa to tt…tt become:

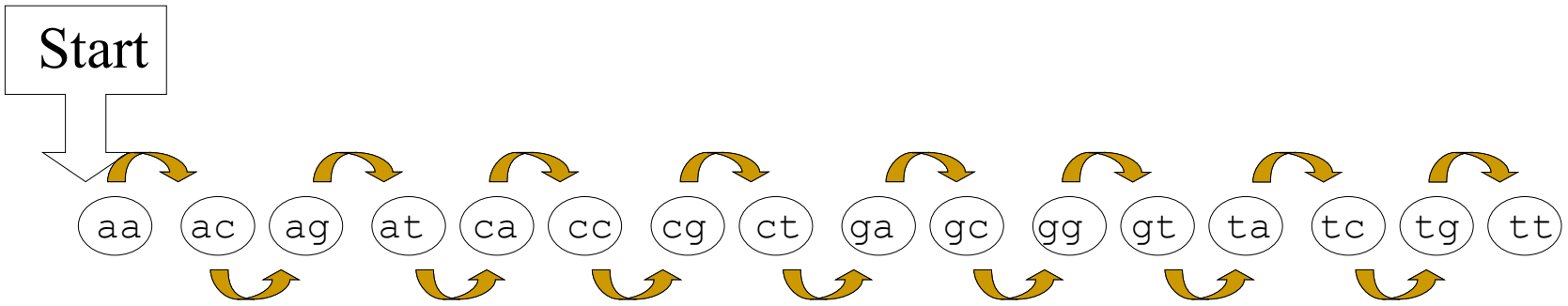
$$\underbrace{11\ldots11}_{l}$$
$$11\ldots12$$
$$11\ldots13$$
$$11\ldots14$$
$$\vdots$$
$$44\ldots44$$

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

Source: Lehigh Uni. - Daniel Lopresti

Linked list representation

Suppose *l* = 2:



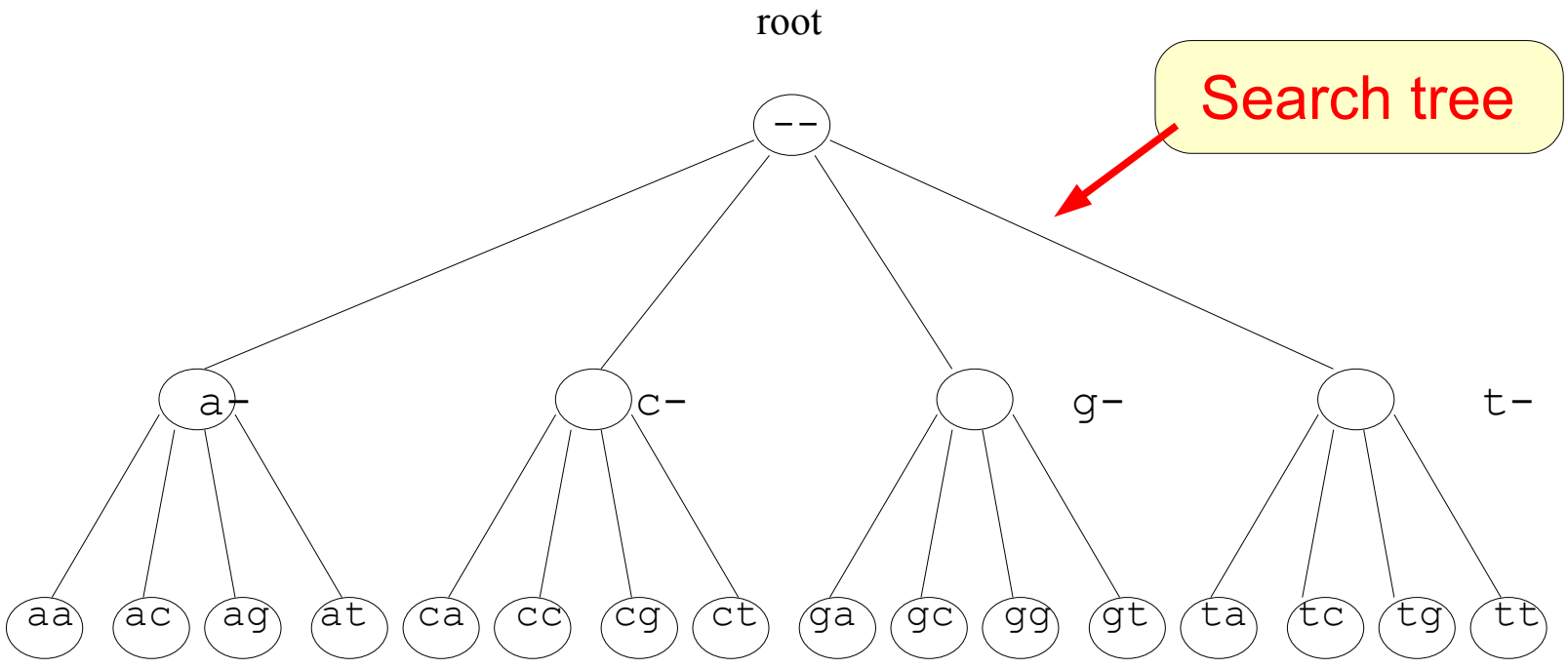
Need to visit all predecessors before visiting a given sequence.

Source: Lehigh Uni. - Daniel Lopresti

Search tree representation

Linked list is not most efficient data structure for motif-finding.

Let's try grouping sequences by their prefixes:



Source: Lehigh Uni. - Daniel Lopresti

Search trees

Characteristics of search trees:

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

How can we move through a search tree?

Four common moves in a search tree:

- Visit next leaf.
- Visit all leaves.
- Visit next vertex.
- Bypass children of a vertex.

Source: Lehigh Uni. - Daniel Lopresti

LEHIGH UNIVERSITY

Visit Next Leaf

Given a current leaf a, we need to compute "next" leaf:

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

Common addition in radix k

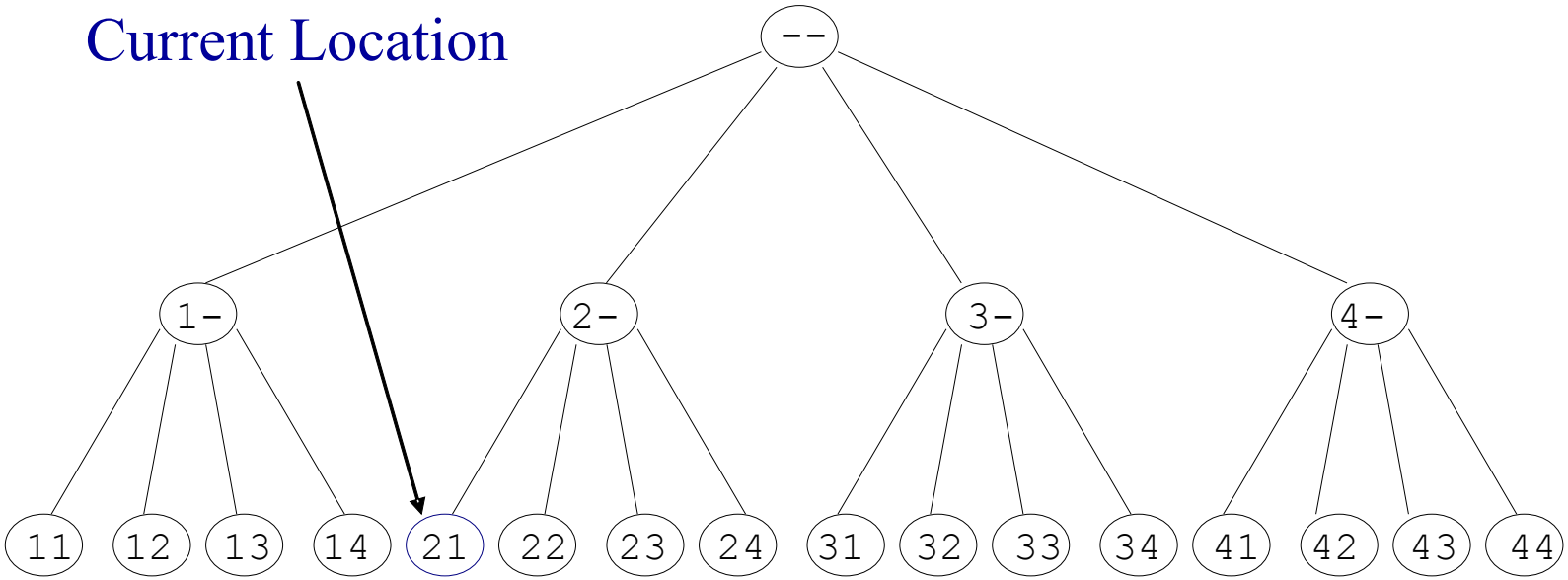
- Increment least significant digit.
- "Carry the one" to next digit position when digit is maximum.

http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

Next Leaf example

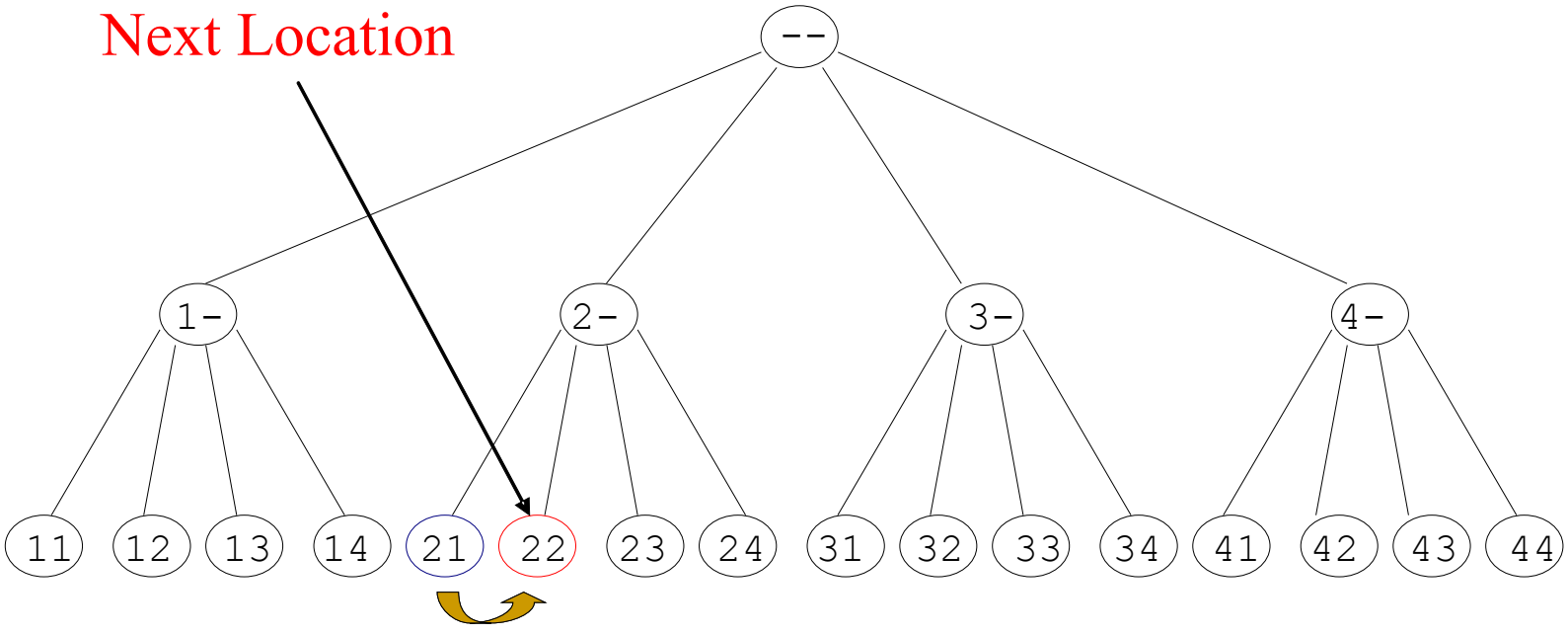
Moving to next leaf:



Source: Lehigh Uni. - Daniel Lopresti

Next Leaf example

Moving to next leaf:



Source: Lehigh Uni. - Daniel Lopresti

Visit All Leaves

Generating all permutations in ascending order:

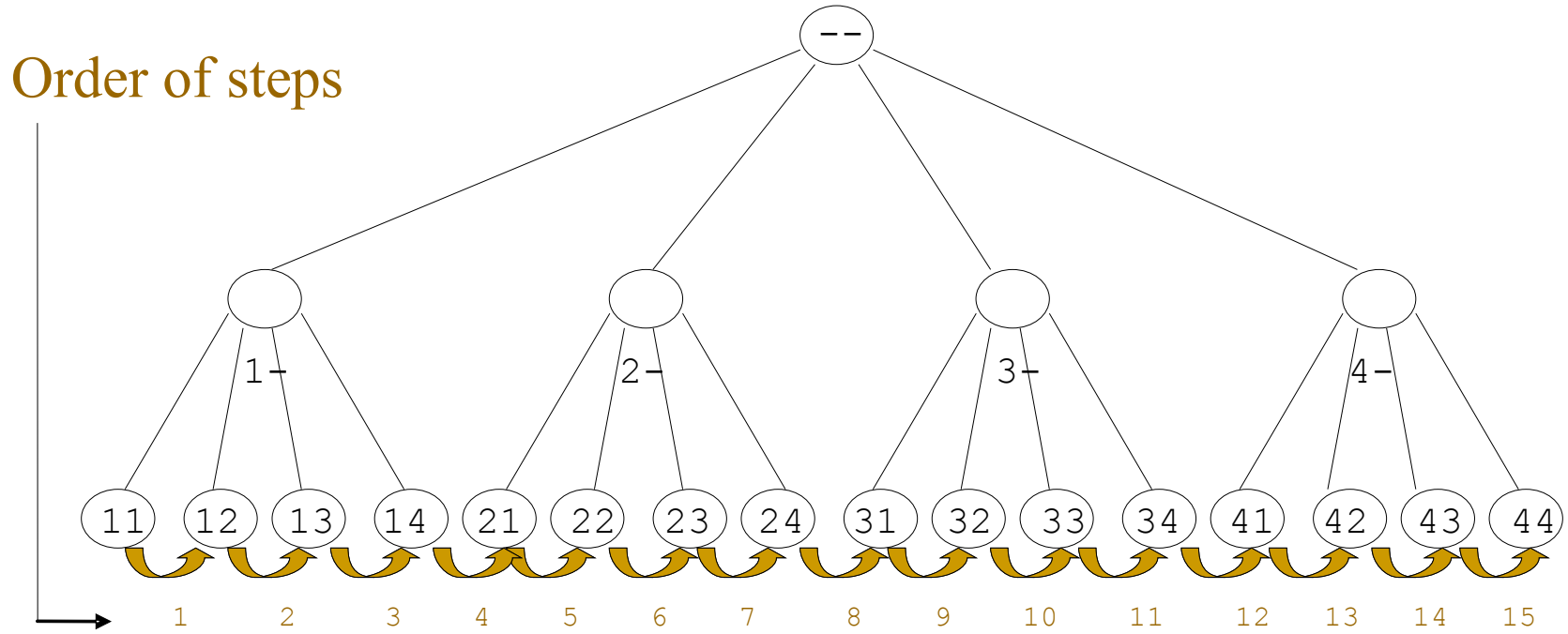
```
AllLeaves( L, k )                 // L : length of sequence
a ← (1,...,1)                     // k : max digit value
while forever                     // a : array of digits
    output a
    a ← NextLeaf(a, L, k)
    if a = (1,...,1)
        return
```

http://www.bioalgorithms.info

Source: Lehigh Uni. - Daniel Lopresti

Visit All Leaves example

Moving through all leaves in order:



Source: Lehigh Uni. - Daniel Lopresti

Depth-first search

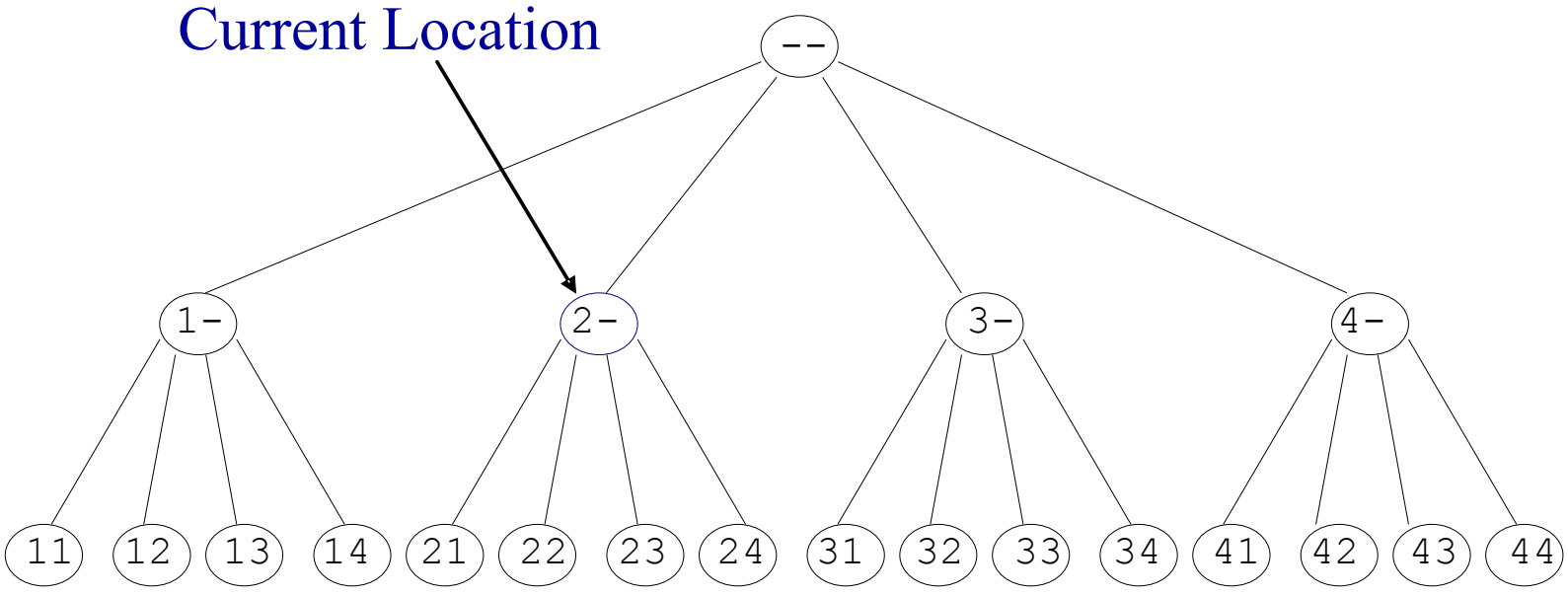
So we can search leaves. How about searching all vertices?

```
NextVertex ( a, i, L, k )        // a : 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)
        a_j ← 1
return(a, 0)
```

Source: Lehigh Uni. - Daniel Lopresti

Next Vertex example

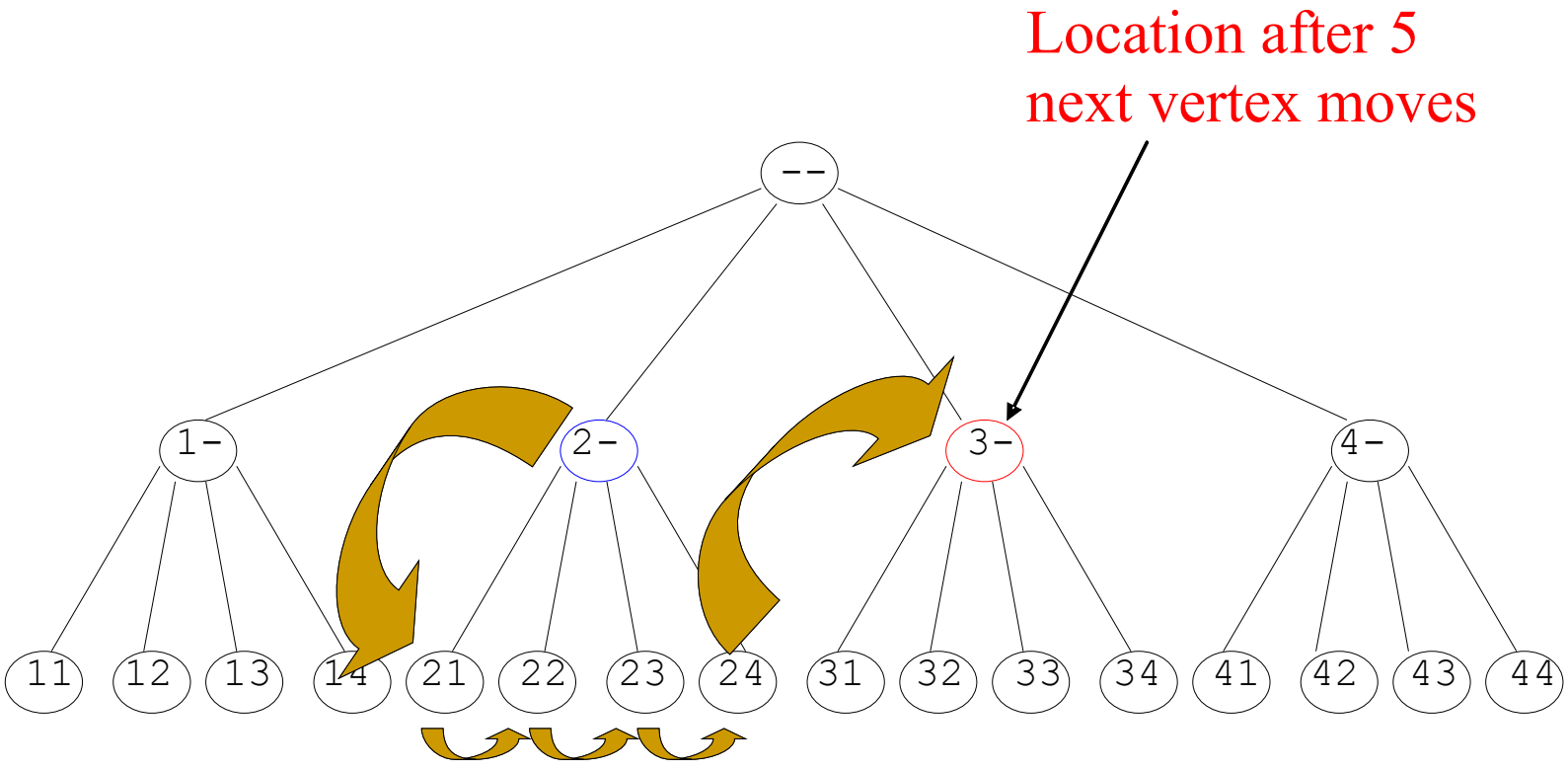
Moving to the next vertex:



Source: Lehigh Uni. - Daniel Lopresti

Next Vertex example

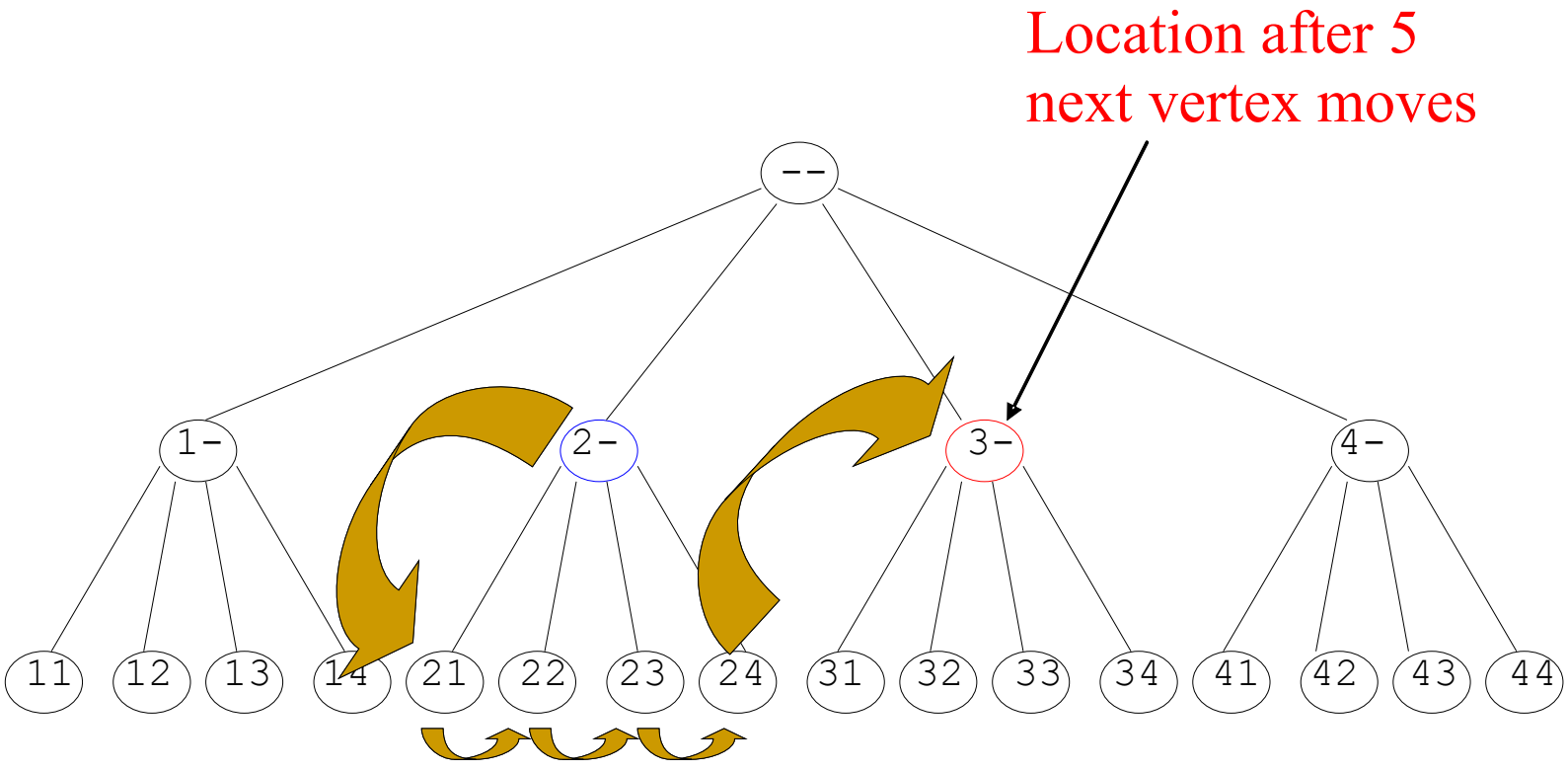
Moving to the next vertex:



Source: Lehigh Uni. - Daniel Lopresti

Next Vertex example

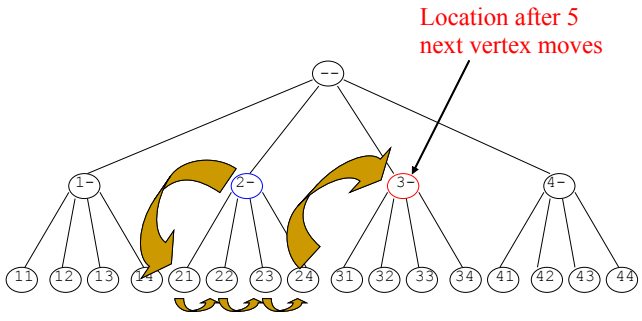
Moving to the next vertex:



Source: Lehigh Uni. - Daniel Lopresti

Next Vertex example

Moving to the next vertex:



LEHIGH UNIVERSITY

Preorder traversal of the tree

```
PREORDER (v)
1       output v
2       if v has children
3           PREORDER ( left child of v )
4           PREORDER ( right child of v )
```

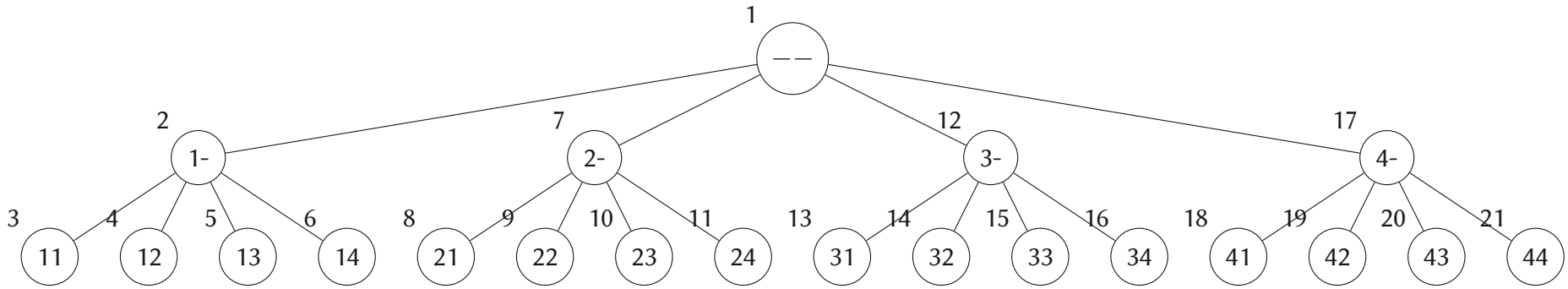
(-, -)
(1, -)
(1, 1)
(1, 2)
(1, 3)
(1, 4)

(2, -)
(2, 1)
(2, 2)
(2, 3)
(3, 4)

(3, -)
(3, 1)
(3, 2)
(3, 3)
(3, 4)

(4, -)
(4, 1)
(4, 2)
(4, 3)
(4, 4)

Search tree representation



Search tree to represent all possible starting positions for l-mers in a library of t sequences

- Motif Finding problem: $L = t$ levels and $k = n - l + 1$ children per vertex
- Median String problem,: $L = l$ levels and $k = 4$ children per vertex

Internal vertices of the tree do not represent a valid state (meaningful starting positions), yet allow to use the branch and bound technique

Median String: improving the running time

```
MedianStringSearch (DNA, t, n, l)
bestWord ← aa…aa
bestDistance ← ∞
for each l-mer s from aa…aa to tt…tt
    if TotalDistance(s,DNA) < bestDistance
        bestDistance ← TotalDistance(s,DNA)
        bestWord ← s
return bestWord
```

Iterate over vertices

Source: Lehigh Uni. - Daniel Lopresti

Median String: improving the running time

```
MedianStringSearch (DNA, t, n, l)
bestWord ← aa…aa
bestDistance ← ∞
for each l-mer s from aa…aa to tt…tt
    if TotalDistance(s,DNA) < bestDistance
        bestDistance ← TotalDistance(s,DNA)
        bestWord ← s
return bestWord
```

Iterate over vertices

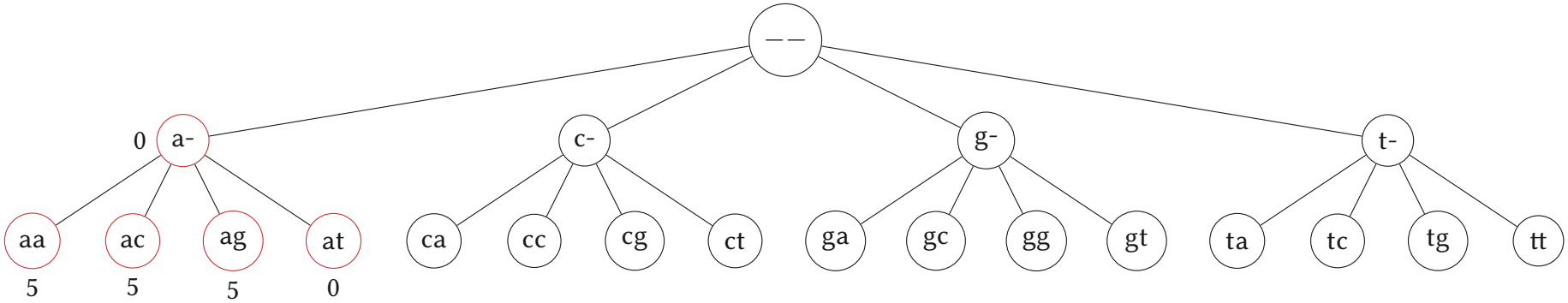
Discard non-promising states*

(*) Prune subtrees not containing high-scoring leaves

```
optimisticDistance ← TotalDistance (s[:i], DNA)
if optimisticDistance > bestDistance
    Bypass(all the children of s[:i])
```

Source: Lehigh Uni. - Daniel Lopresti

Branch and bound



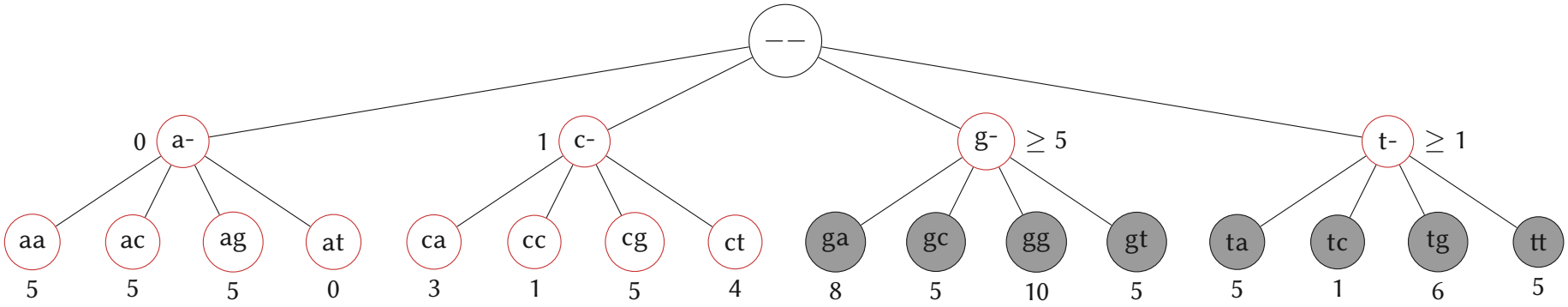
Given the DNA sequence library:

atcc
ccat
atcc
atct
catc

Start traversing the tree and update the bestDistance so far

After traversing all strings starting with an 'a': bestDistance ← ~~inf~~ ~~5~~ 0

Branch and bound



DNA	a-	c-	g-	t-
atcc	0	0	1	0
ccat	0	0	1	1
atcc	0	0	1	0
atct	0	0	1	0
catc	0	0	1	0
TotalDistance(s[:1], DNA)	≥ 0	≥ 0	≥ 5	≥ 1

No point in searchig for strings starting with ‘g’ (and even ‘t’)

Branch-and-bound Median String Search

```
BRANCHANDBOUNDMEDIANSEARCH(DNA, t, n, l)
 1  s ← (1, 1, ..., 1)
 2  bestDistance ← ∞
 3  i ← 1
 4  while i > 0
 5      if i < l
 6          prefix ← nucleotide string corresponding to (s_1, s_2, ..., s_i)
 7          optimisticDistance ← TOTALDISTANCE(prefix, DNA)
 8          if optimisticDistance > bestDistance
 9              (s, i) ← BYPASS(s, i, l, 4)
10          else
11              (s, i) ← NEXTVERTEX(s, i, l, 4)
12      else
13          word ← nucleotide string corresponding to (s_1, s_2, ... s_l)
14          if TOTALDISTANCE(word, DNA) < bestDistance
15              bestDistance ← TOTALDISTANCE(word, DNA)
16              bestWord ← word
17          (s, i) ← NEXTVERTEX(s, i, l, 4)
18  return bestWord
```

Source: An Introduction to Bioinformatics Algorithms, N.C. Jones and P.A. Pevzner, The MIT Press Cambridge, 2004

Algorithms, programming and data structures in Bioinformatics

Subject	4 Exhaustive Search	5 Greedy Algorithms	6 Dynamic Programming Algorithms	7 Divide-and-Conquer Algorithms	8 Graph Algorithms	9 Combinatorial Pattern Matching	10 Clustering and Trees	11 Hidden Markov Models	12 Randomized Algorithms
Mapping DNA	o								
Sequencing DNA					o				
Comparing Sequences			o	o		o			
Predicting Genes			o						
Finding Signals	o	o						o	o
Identifying Proteins					o				
Repeat Analysis						o			
DNA Arrays					o				
Genome Rearrangements		o							
Molecular Evolution							o		

- Read chapters 5 & 6 for the next class

Source: An Introduction to Bioinformatics Algorithms, N.C. Jones and P.A. Pevzner, The MIT Press Cambridge, 2004

http://bioinformaticsinstitute.ru/sites/default/files/an_introduction_to_bioinformatics_algorithms_-_jones_pevzner.pdf