# CS342: Bioinformatics
# Lecture 5

# Knuth-Morris-Pratt Algorithm

- Solves a version of the basic pattern matching problem.
- Rather than shifting $p$ by one at each iteration (brute-force), use info about $p$ to never go “backwards".

**Input:** Text $t = t_0 \ldots t_m$ and pattern $p = p_0 \ldots p_n$ (0 index)

**Output:** Index of the first occurrence of $p$ in $t$.

- Step 1: Compute a table $T$ based only on pattern $p$ that tells us where the pattern contains potential repeats.
- Step 2: Use $T$ to search for the first occurrence of $p$ in $t$.

**Computing T**

- $T$ is table of size length of $p$.

# Knuth-Morris-Pratt Algorithm Analysis

Runtime? $O(n + m)$

- Build T? $O(n)$
- Search for match? $O(m)$

# 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:
  - Assume the hidden sequence is of length 8
  - The pattern is not exactly the same in each sequence because random point mutations have been introduced

# Motif Finding Example

- Finding motifs if there are no mutations
- Probability of a given 8-mer in an infinite sequence is $1/4^8 \approx 1.5 \times 10^{-5}$ (1 every 65Kb)
- Assuming 5 strings of length 68, there are 5 (68 - 8) = 300 distinct 8-mers
- Probability of any one 8-mer is $300/4^8 \approx 0.005$
- So *any* repeat is rare

cctgatagacgctatctggctatcc**acgtacgt**aggtcctctgtgcgaatctatgcgtttccaaccat
agtactggtgtacatttgat**acgtacgt**acaccggcaacctgaaacaaacgctcagaaccagaagtgc
aa**acgtacgt**gcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt
agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatctt**acgtacgt**ataca
ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgtta**acgtacgt**c

**acgtacgt**

# The Problem Becomes Harder

- Introduce 2 point mutations into each pattern:

```
cctgatagacgctatctggctatccaGgtacTtaggtcctctgtgcgaatctatgcgtttccaaccat
agtactggtgtacatttgatCcAtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc
aaacgtTAgtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt
agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatcttacgtCcAtataca
ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgttaCcgtacgGc
```

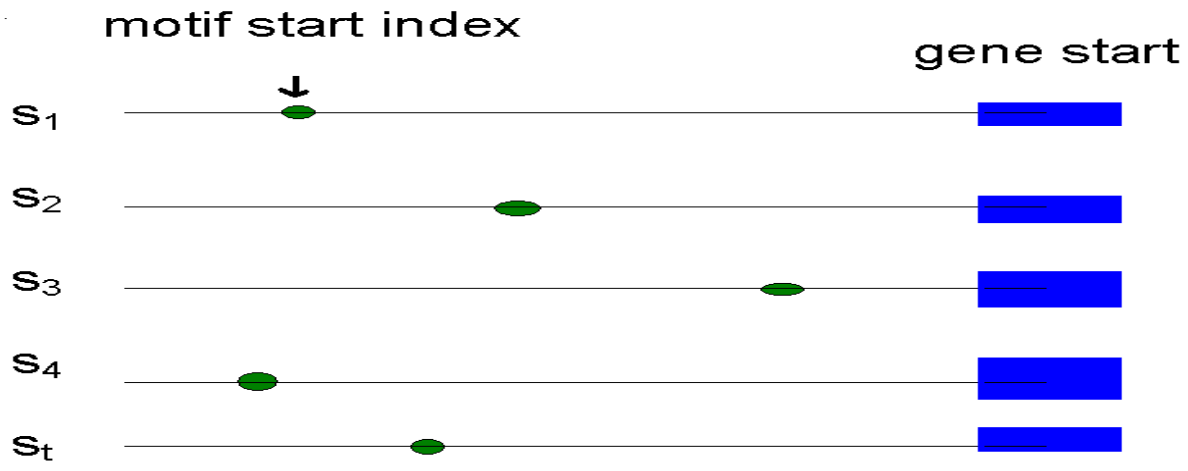
- Our original target pattern no longer appears in any sequence!

**Can we still find the motif?**

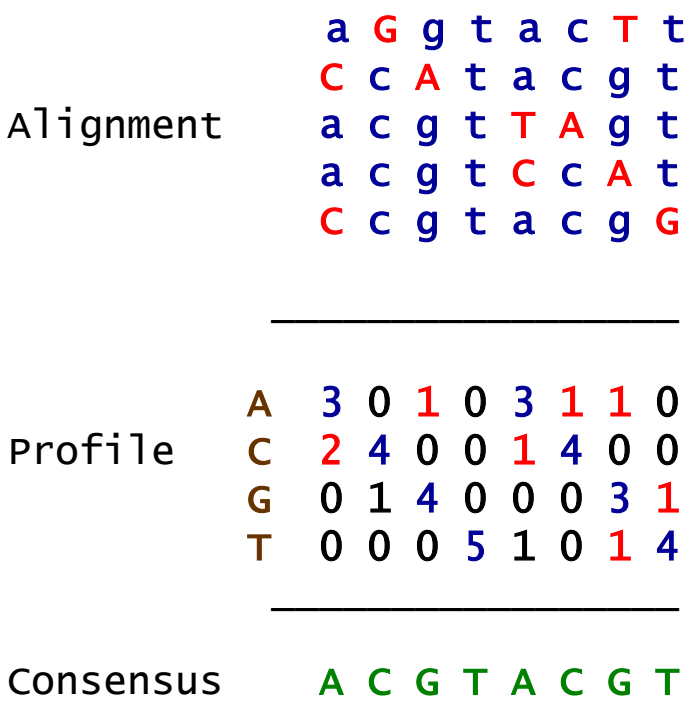
# Defining a Motif

- To define a motif, let's assume that we know where the motif starts in each sequence
- The start positions can be represented as

$$\boldsymbol{s} = [s_1, s_2, s_3, \ldots, s_t]$$



# Motifs: Profiles and Consensus




- Line up the patterns by their start indexes

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

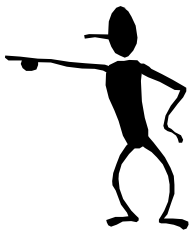
- Construct a matrix profile with the frequencies of each nucleotide in columns

- 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 an actual motif and the consensus sequence is generally less than that for any two actual motifs
- *Hamming distance* is number of positions that differ between two strings

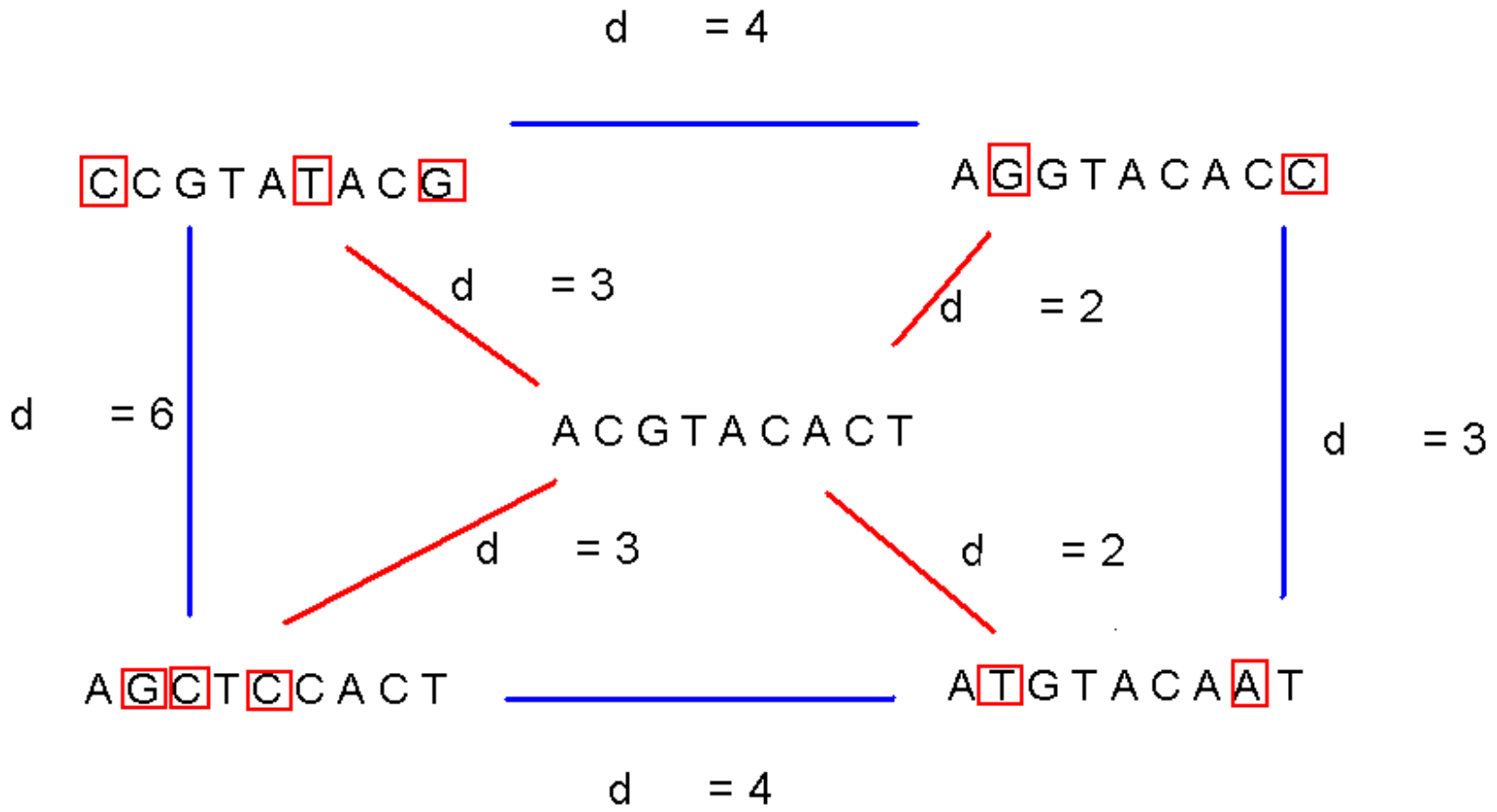
```
G A G A C T C A T
X         X
T A G A C G C A T
```



A Hamming distance of 2

# Consensus Properties

- A consensus string has a minimal hamming distance to all source strings

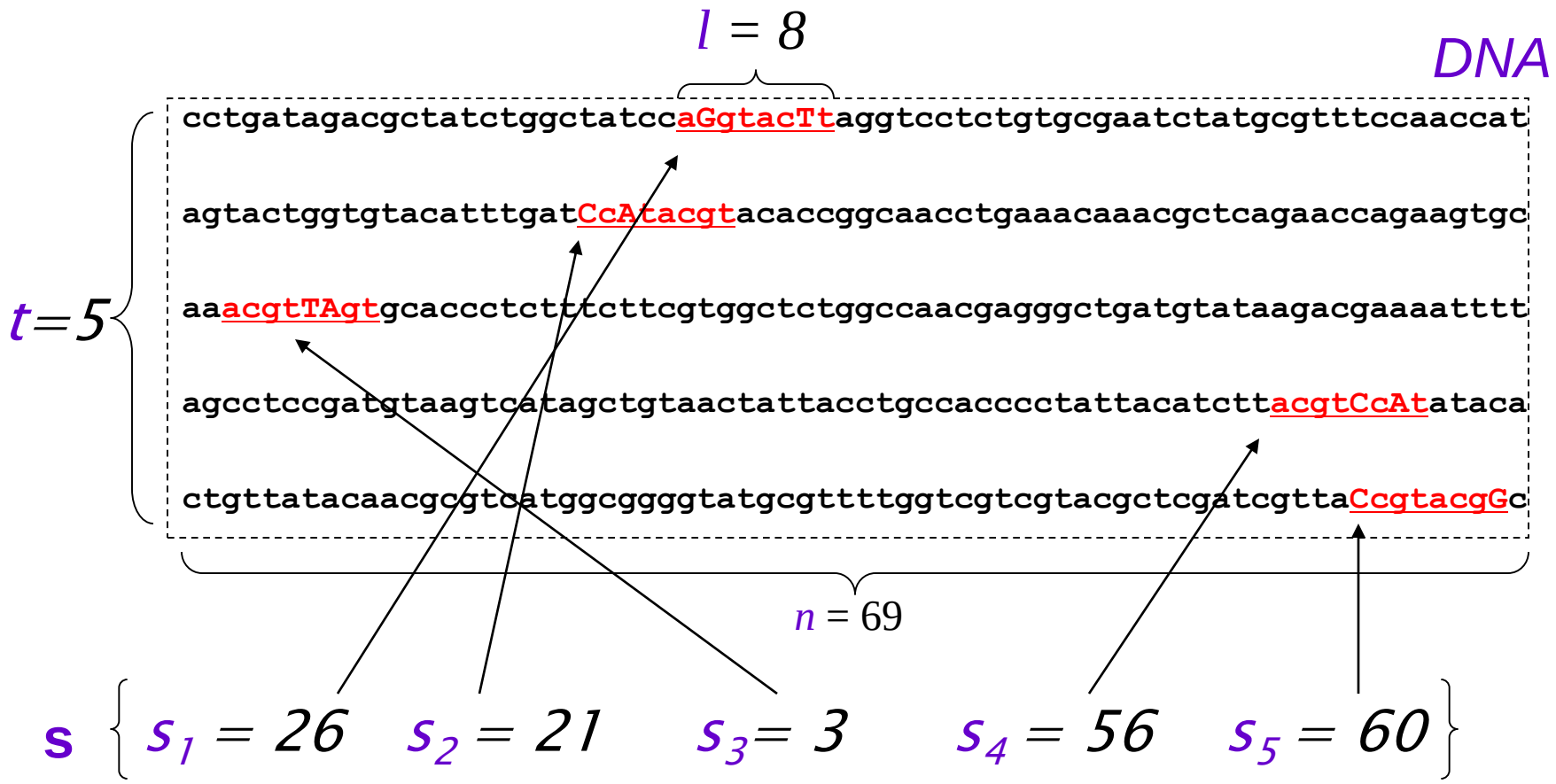


# Defining Some Terms

- ***DNA*** – array of sequence fragments
- ***t*** - number of sample DNA sequences
- ***n*** - length of each DNA sequence

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

# Illustration of Terms



# Scoring Motifs

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

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