



The daily schedules of animals, plants, and even bacteria are controlled by an internal timekeeper called the **circadian clock**. Anyone who has experienced the misery of jet lag knows that this clock never stops ticking. Rats and research volunteers alike, when placed in a bunker, naturally maintain a roughly 24-hour cycle of activity and rest in total darkness. And like any timepiece, the circadian clock can malfunction, resulting in a genetic disease known as **delayed sleep-phase syndrome (DSPS)**.

The circadian clock must have some basis on the molecular level, which presents many questions. How do *individual cells* in animals and plants (let alone bacteria) know when they should slow down or increase the production of certain proteins? Is there a "clock gene"? Can we explain why heart attacks occur more often in the morning but asthma attacks are more common at night? And can we identify genes that are responsible for "breaking" the circadian clock to cause DSPS?

In the early 1970s, Ron Konopka and Seymour Benzer identified mutant flies with abnormal circadian patterns and traced the flies' mutations to a single gene. Biologists needed two more decades to discover a similar clock gene in mammals, which was just the first piece of the puzzle. Today, many more circadian genes have been discovered; these genes, having names like *timeless*, *clock*, and *cycle*, orchestrate the behavior of hundreds of other genes and display a high degree of evolutionary conservation across species.

We will first focus on plants, since maintaining the circadian clock in plants is a matter of life and death. Consider how many plant genes should pay attention to the time when the sun rises and sets; indeed, biologists estimate that over a thousand plant genes are circadian, including the genes related to photosynthesis, photo reception, and flowering. These genes must somehow know what time it is in order to change their gene transcript production, or **gene expression**, throughout the day (see [DETOUR: Gene Expression](#)).

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

- 36. [Motif Finding Is More Difficult Than You Think](#)
- 37. [Scoring Motifs](#)
- 38. [From Motif Finding to Finding a Median String](#)
- 39. [Greedy Motif Search](#)
- 40. [Motif Finding Meets Oliver Cromwell](#)
- 41. [Randomized Motif Search](#)
- 42. [How Can a Randomized Algorithm Perform So Well?](#)
- 43. [Gibbs Sampling](#)
- 44. [Gibbs Sampling in Action](#)



It turns out that plants have a **cell-autonomous circadian clock**, meaning that each cell keeps track of day and night independently of other cells, and that just 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.

For example, regulatory proteins controlling the circadian clock in plants coordinate circadian activity as follows. **TOC1** promotes the expression of **LCY** and **CCA1**, whereas **LCY** and **CCA1** repress the expression of **TOC1**, resulting in a **negative feedback loop**. In the morning, sunlight activates the transcription of **LCY** and **CCA1**, triggering the repression of **TOC1** transcription. As light diminishes, so does the production of **LCY** and **CCA1**, which in turn do not repress **TOC1** any more. Transcription of **TOC1** peaks at night and starts promoting the transcription of **LCY** and **CCA1**, which in turn repress the transcription of **TOC1**, and the cycle begins again.

The **LCY**, **CCA1**, and **TOC1** genes are able to control the transcription of other genes because the regulatory proteins that they encode are **transcription factors**, or master regulatory proteins that turn other genes on and off. Transcription factors regulate genes by binding to specific short DNA intervals called **regulatory motifs** (or **transcription factor binding sites (TFBS)**) in the **upstream regions** of these genes, e.g., a 600-1000 nucleotide-long region preceding the start of the gene. For example, **CCA1** binds to **AAAAAATCT** in the upstream region of many genes regulated by **CCA1**.

The life of a bioinformatician would be easy if regulatory motifs were completely conserved, but the reality is more complex, as regulatory motifs may vary at some positions; e.g., **CCA1** may instead bind to **AAGAACTCT**. But how can we locate these regulatory motifs without knowing what they look like in advance? We need to develop algorithms for **motif finding**, the problem of discovering a "hidden message" shared by a collection of strings.

CENTRAL QUESTION: How can we design an *in silico* approach to identify regulatory motifs?

Bioinformatics Algorithms

Chapter 1: Where Does DNA

Replication Begin?: ...

Chapter 2: How Do We Sequence

Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. [Motif Finding Is More Difficult Than You Think](#)

37. [Scoring Motifs](#)

38. [From Motif Finding to Finding a Median String](#)

39. [Greedy Motif Search](#)

40. [Motif Finding Meets Oliver Cromwell](#)

41. [Randomized Motif Search](#)

42. [How Can a Randomized Algorithm Perform So Well?](#)

43. [Gibbs Sampling](#)

44. [Gibbs Sampling in Action](#)

In 2000, Steve Kay used **DNA arrays** (see [DETOUR: DNA Arrays](#)) to determine which genes in the plant *Arabidopsis thaliana* are activated at different times of the day. He then extracted the upstream regions of nearly 500 genes that exhibited circadian behavior and looked for frequently appearing patterns in their upstream regions. If you concatenated these upstream regions into a single string, you would find that **AAAATATCT** is a surprisingly frequent word, appearing 46 times. Kay named **AAAATATCT** the “evening element” and performed a simple experiment to prove that it is indeed the regulatory motif responsible for circadian gene expression in *Arabidopsis thaliana* — after he mutated the evening element in the upstream region of one gene, the gene lost its circadian behavior.

EXERCISE BREAK: What is the expected number of occurrences of a 9-mer in 500 sequences, each of length 1000? Assume that the sequences are formed by selecting each nucleotide (A, C, G, T) with the same probability (0.25).

Note: Express your answer as a decimal; allowable error = 0.0001.

Press **Enter** to submit

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action

45. Complications in Motif Finding

46. Epilogue: How Does Tuberculosis



Whereas the evening element in plants is very conserved (and thus easy to find), motifs having many mutations are more elusive. For example, if you infect a fly with a bacterium, the fly will switch on its **immunity genes** to fight the infection. Thus, many of the genes with elevated expression levels after the infection are likely to be immunity genes. Indeed, some of these genes have 12-mers similar to **TCGGGGATTCC** in their upstream regions, the binding site of a transcription factor called **NF- κ B** that activates various immunity genes in flies. However, NF- κ B binding sites are nowhere near as conserved as the evening element. Below is a sample of ten NF- κ B binding sites from *Drosophila melanogaster*; the most popular nucleotides in every column are shown by upper case colored letters.

```
TCGGGGgTTTtt
cCGGtGAcTTaC
aCGGGGATTtC
TtGGGGAcTTtt
TCGGGGATTCC
TtGGGGAcTTCC
TCGGGGATTcat
TCGGGGATTcCt
TaGGGGAacTaC
TCGGGtATaaCC
```

Bioinformatics Algorithms

Chapter 1: Where Does DNA

Replication Begin?: ...

Chapter 2: How Do We Sequence

Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action

45. Complications in Motif Finding



Our aim is to turn the biological challenge of finding regulatory motifs into a well- defined computational problem. First, consider the 10 randomly generated nucleotide sequences below, which have a hidden message implanted at a randomly chosen position in every sequence (like transcription factor binding sites implanted in the upstream regions of some genes). We will go ahead and tell you that the hidden message is a 15-mer.

```
atgaccgggatactgataaaaaaagggggggcgctacacattagataaacgtatgaagtacgtagactcggcgccgcccg
acccctatttttgagcagatttagtgacctggaaaaaaatttgagtacaaaacttttcgaataaaaaaaaggggggga
tgagtatccctgggatgacttaaaaaaaaggggggggtgctctccgatttttgaatatgtaggatcattccagggtccga
gctgagaattggatgaaaaaaaggggggggtccacgcaatcggaaccaacgcggaacccaaaggcaagaccgataaaggaga
tcccttttgcggtaattgtccgggaggctggttacgtaggaagccctaacggacttaataaaaaaaagggggggccttag
gtcaatcatgttcttgtgaatggatttaaaaaaaaggggggggacccgttggcgacccaaattcagtggtggcgagcgcaa
cggttttggcccttgtagaggcccccgtaaaaaaaaggggggggcaattatgagagagctaattctatcgctgctgttcat
aacttgagttaaaaaaagggggggctgggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgta
ttggccattggctaaaagcccaacttgacaaatggaagatagaatccttgcataaaaaaaaggggggggaccgaaagggaag
ctggtgagcaacgacagattcttacgtgcattagctcgcttcgggggatctaatagcacgaagcttaaaaaaaaggggggga
```

STOP and Think: Can you find the hidden message?

Bioinformatics Algorithms

Chapter 1: Where Does DNA

Replication Begin?: ...

Chapter 2: How Do We Sequence

Antibiotics?: ...

Chapter 3: Which DNA Patterns Play

the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action

45. Complications in Motif Finding



This is a simple problem – running the Frequent Words Problem on the concatenation of these strings will immediately reveal the implanted pattern **AAAAAAAAAGGGGGG** as the most frequent 15-mer, as shown below. Since these short strings were randomly generated, it is unlikely that they contain other frequent 15-mers.

```
atgaccgggatactgatAAAAAAAAAGGGGGGggcgctacacattagataaacgtatgaagtacgttagactcggcgccgcccg
acccctattttttgagcagatttagtgacctggaaaaaaatttgagtacaaaacttttccgaataAAAAAAAAAGGGGGGGa
tgagtatccctgggatgacttAAAAAAAAAGGGGGGGtgcctcccgattttgaaatgtaggatcattcggcagggtccga
gctgagaattggatgAAAAAAAAAGGGGGGGtccacgcaatcggaaccaacgcggacccaaggcaagaccgataaaggaga
tcccttttgccgtaattgtgcgggaggctggttacgtagggaagccctaaccggacttaatAAAAAAAAAGGGGGGGttatag
gtcaatcatgttcttgtgaatggatttAAAAAAAAAGGGGGGGaccgcttggcgacccaattcagtggtggcgagcgcaa
cggttttggcccttgtagaggcccccgtAAAAAAAAAGGGGGGGcaattatgagagagctaattctatcgctgctgttcat
aacttgagttAAAAAAAAAGGGGGGGctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgta
ttggccattggctaaaagcccaacttgacaaatggaagatagaatccttgcatAAAAAAAAAGGGGGGGaccgaaagggaag
ctggtgagcaacgacagattcttacgtgcattagctcgttcggggatctaatagcacgaagcttAAAAAAAAAGGGGGGGa
```

Bioinformatics Algorithms

Chapter 1: Where Does DNA

Replication Begin?: ...

Chapter 2: How Do We Sequence

Antibiotics?: ...

Chapter 3: Which DNA Patterns Play

the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action

45. Complications in Motif Finding



Now imagine that instead of inserting exactly the same pattern into all sequences, we mutate the pattern before inserting it into each sequence by randomly changing the nucleotides at 4 randomly selected positions within each implanted 15-mer:

```
atgaccgggatactgatAgAagAAAGGttGGGggcggtacacattagataaacgtatgaagtacgttagactcggcgccgccc
acccctatTTTTTgagcagatttagtgacctggaaaaaaatttgagtacaaaacttttcgaatacAAATAAAcGGcGGGa
tgagtatccctgggatgacttAAAAAATAGGaGtGGtgctctcccgatttttgaatatgtaggatcattcgccagggtccga
gctgagaattggatgcAAAAAAGGGattGtccacgcaatcggaaccaacgcggacccaaaggcaagaccgataaaggaga
tcccttttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaataATAATAAGGaaGGGcttatag
gtcaatcatgttcttgtgaatggatttAAcAAATAGGGctGGgaccgcttggcgacccaaattcagtggtggcgagcgcaa
cggttttggcccttgttagaggcccccgtATAAAcAAGGaGGGCcaattatgagagagctaatactatcgctgctgttcat
aacttgagttAAAAAATAGGGaGccctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgta
ttggccattggctaaaagcccaacttgacaaatggaagatagaatccttgcatAcTAAAAAGGaGcGGaccgaagggaag
ctggtgagcaacgacagattcttacgtgcattagctcgcttcggggatctaatagcacgaagcttActAAAAAGGaGcGGa
```

The Frequent Words Problem is not going to help us, since **AAAAAAAAAGGGGGG** does not even appear in the sequences above. Maybe we should try an approach similar to the algorithm for the Frequent Words with Mismatches Problem? However, in Chapter 1, we implemented an algorithm for the Frequent Words with Mismatches Problem aimed at finding hidden messages with a small number of mismatches and a small *k*-mer size (e.g., 1-2 mismatches for *DnaA* boxes of length 9). This algorithm is likely to become too time consuming when searching for the implanted motif above, which is longer and has more mutations. We need to design a faster algorithm!

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. [Do We Have a "Clock" Gene?](#)

36. Motif Finding Is More Difficult Than You Think

37. [Scoring Motifs](#)

38. [From Motif Finding to Finding a Median String](#)

39. [Greedy Motif Search](#)

40. [Motif Finding Meets Oliver Cromwell](#)

41. [Randomized Motif Search](#)

42. [How Can a Randomized Algorithm Perform So Well?](#)

43. [Gibbs Sampling](#)

44. [Gibbs Sampling in Action](#)

45. [Complications in Motif Finding](#)



Let's first reformulate this problem to account for multiple sequences, since simply applying it to the concatenated strings will miss the requirement that the implanted motif must occur in *every* string. The reason why the "concatenated string" approach is inadequate is that it does not correctly model the biological problem at hand, as a regulatory motif is a different kind of hidden message than a *DnaA* box. A *DnaA* box is a pattern that clumps, or appears frequently, within a relatively short interval of the genome. In contrast, a regulatory motif is a pattern that appears at least once (perhaps with variation) in each of many different regions that are scattered throughout the genome.

Let's formulate a computational problem to account for motifs occurring in multiple different strings. Given a collection of strings *Dna* and an integer *d*, a *k*-mer is a **(*k*, *d*)-motif** if it appears in every string from *Dna* with at most *d* mismatches. For example, the implanted 15-mer in the strings above represents a (15,4)-motif. Below we will discuss algorithms for finding elusive implanted (*k*, *d*)-motifs.

Implanted Motif Problem: Find all (*k*, *d*)-motifs in a collection of strings.

Input: A collection of strings *Dna*, and integers *k* and *d*.

Output: All (*k*, *d*)-motifs in *Dna*.

A simple approach for solving the Implanted Motif Problem is based on the observation that any (*k*, *d*)-motif must be at most *d* mismatches apart from some *k*-mer appearing in one of the strings of *Dna*. Therefore, we can generate all such *k*-mers and then check which of them are (*k*, *d*)-motifs.

```
MOTIFENUMERATION(Dna, k, d)
  for each k-mer a in Dna
    for each k-mer a' differing from a by at most d mutations
      if a' appears in each string from Dna with at most d mutations
        output a'
```

Bioinformatics Algorithms

Chapter 1: Where Does DNA

Replication Begin?: ...

Chapter 2: How Do We Sequence

Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action

45. Complications in Motif Finding

46. Epilogue: How Does Tuberculosis



CODE CHALLENGE: Implement `MOTIFENUMERATION` (reproduced below).

Input: Integers k and d , followed by a collection of strings Dna .

Output: All (k, d) -motifs in Dna .

```
MOTIFENUMERATION( $Dna, k, d$ )
  for each  $k$ -mer  $a$  in  $Dna$ 
    for each  $k$ -mer  $a'$  differing from  $a$  by at most  $d$  mutations
      if  $a'$  appears in each string from  $Dna$  with at most  $d$  mutations
        output  $a'$ 
```

Sample Input:

```
3 1
ATTGGC
TGCCTTA
CGGTATC
GAAAATT
```

Sample Output:

```
ATA ATT GTT TTT
```

Extra Dataset

Start Quiz (limit: 5 minutes)

Bioinformatics Algorithms

Chapter 1: Where Does DNA

Replication Begin?: ...

Chapter 2: How Do We Sequence

Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action

45. Complications in Motif Finding

STOP and Think: Estimate the running time of `MOTIFENUMERATION`.

`MOTIFENUMERATION` is unfortunately rather slow for large values of k and d , and so we will instead try a different approach. Maybe we can detect an implanted motif simply by identifying the two most similar k -mers between each pair of strings in *Dna*? However, consider the implanted 15-mers `AgAAgAAAGGttGGG` and `cAAtAAAAcGGGGcG`, each of which differs from `AAAAAAAAAGGGGGGG` by 4 mismatches. Although these motifs are similar to the correct motif `AAAAAAAAAGGGGGGG`, they are not so similar when compared to each other (having $4+4 = 8$ mismatches):

```

AgAAgAAAGGttGGG
| | | |
AAAAAAAAAGGGGGGG
| | | |
cAAtAAAAcGGGGcG

```

Since these two implanted patterns are so different, we should be concerned whether we will be able to find them by finding the most similar k -mers.

In the rest of the chapter, we will benchmark our motif finding algorithms by using a particularly challenging instance of the Implanted Motif Problem with 10 randomly generated strings of length 600 (the typical length of many upstream regulatory regions). The **Subtle Motif Problem** refers to implanting a 15-mer with 4 random mutations in 10 randomly generated 600 nucleotide-long strings. The dataset below, which we will refer to throughout this chapter, corresponds to choosing `AAAAAAAAAGGGGGGG` as this 15-mer.

[Download Subtle Motif Dataset](#)

Yet as you can verify after downloading this data it turns out that thousands of pairs of randomly occurring 15-mers in our dataset for the Subtle Motif Problem are 7 or fewer nucleotides apart from each other, which prevents us from identifying the true implanted motifs by pairwise comparisons.

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. [Do We Have a "Clock" Gene?](#)

36. Motif Finding Is More Difficult Than You Think

37. [Scoring Motifs](#)

38. [From Motif Finding to Finding a Median String](#)

39. [Greedy Motif Search](#)

40. [Motif Finding Meets Oliver Cromwell](#)

41. [Randomized Motif Search](#)

42. [How Can a Randomized Algorithm Perform So Well?](#)

43. [Gibbs Sampling](#)

44. [Gibbs Sampling in Action](#)

Although the Implanted Motifs Problem offers a useful abstraction of the biological problem of motif finding, it has some limitations.

For example, when Steve Kay used a DNA array to infer the set of evening genes in plants, he did not expect that *all* genes in the resulting set would have the evening element (or its variants) in their upstream regions. Similarly, biologists do not expect that all genes with an elevated expression level in infected flies must be regulated by NF- κ B. DNA array experiments are inherently noisy, and some genes identified by these experiments have nothing to do with the circadian clock in plants or immunity genes in flies. For such noisy datasets, any algorithm for the Implanted Motif Problem would fail, because as long as a single sequence does not contain the TFBS, a (k, d) -motif does not exist! A more appropriate problem formulation would score individual instances of motifs depending on how similar they are to an "ideal" motif (i.e., a transcription factor binding site that binds the best to the Transcription Factor). However, since the ideal motif is unknown, we attempt to select a k -mer from each string and score these k -mers depending on how similar they are to *each other*.

To define scoring, consider t DNA sequences, each of length n , and select a k -mer from each sequence to form a collection *Motifs*, which we represent as a $t \times k$ **motif matrix**. In the figure below, which shows the motif matrix for the NF- κ B binding sites, we indicate the most popular nucleotide in each column of the motif matrix by upper case letters.

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

If there are multiple most popular nucleotides in a column, we arbitrarily select one of them and declare all other nucleotides as unpopular.

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action

45. Generalization in Motif Finding

By varying the choice of k -mers in each sequence, we can construct a large number of different motif matrices from a given sample of DNA sequences. Our goal is to select k -mers resulting in the most "conserved" motif matrix, meaning the matrix with the most upper case letters (and thus the fewest number of lower case letters). Leaving aside the question of how we select such k -mers, we will first focus on how to score the resulting motif matrices, defining $Score(Motifs)$ as the number of unpopular (lower case) elements in $Motifs$. Our goal is to find a collection of k -mers that *minimizes* this score.

Motifs	T	C	G	G	G	G	g	T	T	T	t	t
c	C	G	G	G	t	G	A	c	T	T	a	C
a	C	G	G	G	G	G	A	T	T	T	t	C
T	t	G	G	G	G	G	A	c	T	T	t	t
a	a	G	G	G	G	G	A	c	T	T	C	C
T	t	G	G	G	G	G	A	c	T	T	C	C
T	C	G	G	G	G	G	A	T	T	c	a	t
T	C	G	G	G	G	G	A	T	T	c	C	t
T	a	G	G	G	G	G	A	a	c	T	a	C
T	C	G	G	G	t	A	T	a	a	a	C	C
Score	3 + 4 + 0 + 0 + 1 + 1 + 1 + 5 + 2 + 3 + 6 + 4 = 30											

EXERCISE BREAK: Certainly, the minimum possible value of $Score(Motifs)$ is 0 (if all the k -mers in $Motifs$ are the same). What is the maximum possible value of $Score(Motifs)$ for 10 motifs of length 15?

[Start Quiz](#)

Bioinformatics Algorithms

Chapter 1: Where Does DNA

Replication Begin?: ...

Chapter 2: How Do We Sequence

Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action



Using the motif matrix, we can construct the $4 \times k$ **count matrix** $Count(Motifs)$ counting the number of occurrences of each nucleotide in each column of the motif matrix; the (i, j) -th element of $Count(Motifs)$ stores the number of times that nucleotide i appears in column j of $Motifs$. We will further divide all of the elements in the count matrix by t , the number of rows in $Motifs$. This results in a **profile matrix** $P = Profile(Motifs)$ for which P_{ij} is the *frequency* of the i -th nucleotide in the j -th column of the motif matrix. Note that the elements of any column of the profile matrix sum to 1. The figure below shows the motif, count, and profile matrices for the NF- κ B binding sites; note that positions 3 and 4 are the most conserved (nucleotide G is completely conserved in these positions), whereas position 11 is the least conserved.

Motifs		T	C	G	G	G	G	g	T	T	T	t	t
	c	C	G	G	t	G	A	c	T	T	T	a	C
	a	C	G	G	G	G	A	T	T	T	T	t	C
	T	t	G	G	G	G	A	c	T	T	T	t	t
	a	a	G	G	G	G	A	c	T	T	T	C	C
	T	t	G	G	G	G	A	c	T	T	T	C	C
	T	C	G	G	G	G	A	T	T	c	c	a	t
	T	C	G	G	G	G	A	T	T	c	c	C	t
	T	a	G	G	G	G	A	a	c	T	a	C	C
	T	C	G	G	G	t	A	T	a	a	C	C	C
Score		3 + 4 + 0 + 0 + 1 + 1 + 1 + 5 + 2 + 3 + 6 + 4 = 30											
Count													
	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													
	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

Bioinformatics Algorithms

Chapter 1: Where Does DNA

Replication Begin?: ...

Chapter 2: How Do We Sequence

Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action

Finally, we form a **consensus string** (denoted $\text{Consensus}(\text{Motifs})$) from the most popular elements (i.e., the upper case letters) in each column of the motif matrix. The consensus string for the NF- κ B binding sites in the figure below is **TCGGGGATTTC**. If we select Motifs correctly from the collection of upstream regions, then $\text{Consensus}(\text{Motifs})$ provides an ideal candidate regulatory motif for these regions.

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

Consider the 2nd column (containing 6 C, 2 a, and 2 t) and the final column (containing 6 C and 4 t) in the motif matrix above; both of these columns contribute 4 to the computation of $\text{Score}(\text{Motifs})$.

STOP and Think: Does scoring these columns equally make sense biologically?

Bioinformatics Algorithms

Chapter 1: Where Does DNA

Replication Begin?: ...

Chapter 2: How Do We Sequence

Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You
Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median
String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm
Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action

For many biological motifs, certain motif positions feature two nucleotides with roughly the same ability to bind to a transcription factor. For example, the 16 nucleotide-long CSRE transcription factor binding site in the yeast species *S. cerevisiae* consists of 5 conservative positions (1, 8, 9, 12, and 13) in addition to 11 variable positions, each of which features two nucleotides with similar frequencies.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
C	G/C	G/T	T/A	C/T	G/C	C/G	A	T	G/T	C/G	A	T	C/T	C/T	G/T

Following this example, a more appropriate representation of the consensus string **TCGGGATTTC** for the NF- κ B binding sites should include viable alternatives to the most popular nucleotides in each column, e.g., nucleotides with frequencies equal to or exceeding 0.4. In this sense, the last column (6 C, 4 T) in the NF- κ B motif matrix (reproduced below) is "more conserved" than the 2nd column (6 C, 2 A, 2 T) and should receive a lower score.

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

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action



Every column of *Profile(Motifs)* corresponds to a **probability distribution**, or a collection of nonnegative numbers that sum to 1. For example, the 2nd column in the profile matrix for the NF-κB binding sites corresponds to the probabilities 0.2, 0.6, 0.0, and 0.2 for A, C, G, and T, respectively.

Profile	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

Entropy is a measure of the uncertainty of a probability distribution (p_1, \dots, p_N) , and is defined as

$$H(p_1, \dots, p_N) = - \sum_{i=1}^N p_i \cdot \log_2 p_i$$

For example, the entropy of the probability distribution (0.2, 0.6, 0.0, 0.2) corresponding to the 2nd column of the NF-κB profile matrix is:

$$-(0.2 \log_2 0.2 + 0.6 \log_2 0.6 + 0.0 \log_2 0.0 + 0.2 \log_2 0.2) \approx 1.371$$

whereas the entropy of the more conserved final column (0.0, 0.6, 0.0, 0.4) is:

$$-(0.0 \log_2 0.0 + 0.6 \log_2 0.6 + 0.0 \log_2 0.0 + 0.4 \log_2 0.4) \approx 0.971$$

and the entropy of the very conserved 5th column (0.0, 0.0, 0.9, 0.1) is:

$$-(0.0 \log_2 0.0 + 0.0 \log_2 0.0 + 0.9 \log_2 0.9 + 0.1 \log_2 0.1) \approx 0.467$$

Note: Technically, $\log_2(0)$ is undefined, but in the computation of entropy, we assume that $0 \cdot \log_2(0)$ is equal to 0.

STOP and Think: What are the maximum and minimum possible values for the entropy of a column?

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

The entropy of the completely conserved 3rd column is 0, which is the minimum possible entropy. On the other hand, a column with equally-likely nucleotides (all probabilities equal to $1/4$) has maximum possible entropy $-4 \cdot 1/4 \cdot \log_2(1/4) = 2$. In general, the more conserved the column, the smaller its entropy. Thus, entropy offers an improved method of scoring motif matrices: the **entropy** of a motif matrix is equal to the sum of the entropies of its columns. We will continue to use *Score(Motifs)* for simplicity, but the entropy score is used more often in practice.

EXERCISE BREAK: Compute the entropy of the NF- κ B motif matrix (reproduced below).

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

[Start Quiz](#)

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action

Another application of entropy is the **motif logo**, a diagram for visualizing motif conservation that consists of a stack of letters at each position. We show a motif logo for the NF- κ B motif matrix at the bottom. The relative sizes of the letters indicate their frequency in the column. The total height of the letters in a column is based on the **information content** of the column, which is defined as $2 - H(p_1, \dots, p_N)$. The lower the entropy, the higher the information content, meaning that tall columns in the motif logo are highly conserved.

Motifs		T	C	G	G	G	G	g	T	T	T	t	t
	c	C	G	G	t	G	A	c	T	T	T	a	C
	a	C	G	G	G	G	A	c	T	T	T	t	C
	T	t	G	G	G	G	A	c	T	T	T	t	C
	a	a	G	G	G	G	A	c	T	T	T	C	C
	T	t	G	G	G	G	A	c	T	T	T	C	C
	T	C	G	G	G	G	A	T	T	T	T	C	C
	T	a	G	G	G	G	A	a	c	T	T	a	C
	T	C	G	G	G	t	A	T	a	a	a	C	C
Profile	A:	.2	.2	0	0	0	0	.9	.1	.1	.1	.3	0
	C:	.1	.6	0	0	0	0	.4	.1	.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		T	C	G	G	G	G	A	T	T	T	C	C



Bioinformatics Algorithms

Chapter 1: Where Does DNA

Replication Begin?: ...

Chapter 2: How Do We Sequence

Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action

You may like to see all the components of the NF- κ B motif matrix that we have worked with, so we reproduce them below.

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	3 + 4 + 0 + 0 + 1 + 1 + 1 + 5 + 2 + 3 + 6 + 4 = 30												
Count	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	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	T	C	G	G	G	G	A	T	T	T	C	C	



Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action



Now that we have a good grasp of scoring a collection of k -mers, we are ready to formulate the Motif Finding Problem.

Motif Finding Problem: Given a collection of strings, find a set of k -mers, one from each string, that minimizes the score.

Input: A collection of strings Dna and an integer k .

Output: A collection of k -mers, one from each string in Dna , minimizing $Score(Motifs)$.

A brute force algorithm for the Motif Finding Problem (referred to as **BRUTEFORCEMOTIFSEARCH**) considers every possible choice of k -mers $Motifs$ from Dna (one k -mer from each string) and returns the collection $Motifs$ having minimum score. Because there are $n - k + 1$ choices of k -mers in each of t sequences, there are $(n - k + 1)^t$ different ways to form $Motifs$. For each choice of $Motifs$, the algorithm calculates $Score(Motifs)$, which requires $k \cdot t$ steps. Thus, assuming that k is smaller than n , the overall running time of the algorithm is $O(n^t \cdot k \cdot t)$. We need to come up with a faster algorithm!

Because **BRUTEFORCEMOTIFSEARCH** is inefficient, we will think about motif finding in a different way. Instead of exploring all $Motifs$ in Dna and deriving the consensus string from $Motifs$ afterwards:

$$Motifs \rightarrow Consensus(Motifs)$$

we will explore all potential k -mer consensus strings first and then find the best possible collection $Motifs$ for each consensus string:

$$Consensus(Motifs) \rightarrow Motifs$$

Bioinformatics Algorithms

Chapter 1: Where Does DNA

Replication Begin?: ...

Chapter 2: How Do We Sequence

Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

To reformulate the motif finding problem, we need to devise an alternative way of computing $\text{Score}(\text{Motifs})$. Until now, we have computed $\text{Score}(\text{Motifs})$, the the number of lower case letters in the motif matrix, column-by-column ($3 + 4 + 0 + 0 + 1 + 1 + 1 + 5 + 2 + 3 + 6 + 4 = 30$). The figure below illustrates that $\text{Score}(\text{Motifs})$ can just as easily be computed row-by-row ($3 + 4 + 2 + 4 + 3 + 2 + 3 + 2 + 4 + 3 = 30$). Note that each element in the sum $3 + 4 + 2 + 4 + 3 + 2 + 3 + 2 + 4 + 3 = 30$ represents the number of mismatches between the consensus string **TCGGGGATTCC** and a motif represented by a row in the motif matrix. The number of mismatches between two k -mers v and w is called the **Hamming distance** between v and w and is denoted by $d(v, w)$. For example, $d(\text{TCGGGGATTCC}, \text{TCGGGGgTTTt}) = 3$ for the 1st row of the motif matrix below.

Motifs	T	C	G	G	G	G	g	T	T	T	t	t	3
c	C	G	G	t	G	A	c	T	T	T	a	C	4
a	C	G	G	G	G	A	T	T	T	T	t	C	2
T	t	G	G	G	G	A	c	T	T	T	t	t	4
a	a	G	G	G	G	A	c	T	T	T	C	C	3
T	t	G	G	G	G	A	c	T	T	T	C	C	2
T	C	G	G	G	G	A	T	T	c	a	a	t	3
T	C	G	G	G	G	A	T	T	c	C	C	t	2
T	a	G	G	G	G	A	a	c	T	a	C	C	4
T	C	G	G	G	t	A	T	a	a	C	C	C	3
Score	3 + 4 + 0 + 0 + 1 + 1 + 1 + 5 + 2 + 3 + 6 + 4 = 30												
Consensus	T	C	G	G	G	G	A	T	T	T	C	C	

The motif, score, and consensus matrix for the NF- κ B binding sites. Rather than sum the non-consensus elements column-by-column, we can sum them row-by-row, as highlighted on the right of the motifs matrix. Each value at the end of a row corresponds to the Hamming distance between that row and the consensus string.

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

- 35. Do We Have a "Clock" Gene?
- 36. Motif Finding Is More Difficult Than You Think
- 37. Scoring Motifs
- 38. From Motif Finding to Finding a Median String**
- 39. Greedy Motif Search
- 40. Motif Finding Meets Oliver Cromwell
- 41. Randomized Motif Search
- 42. How Can a Randomized Algorithm Perform So Well?
- 43. Gibbs Sampling
- 44. Gibbs Sampling in Action

Given a set of k -mers $Motifs = \{Motif_1, \dots, Motif_t\}$ and a k -mer $Pattern$, we now define $d(Pattern, Motifs)$ as the sum of Hamming distances between $Pattern$ and each $Motif_i$:

$$d(Pattern, Motifs) = \sum_{i=1}^t d(Pattern, Motif_i)$$

Because $Score(Motifs)$ corresponds to adding the lower case elements of $Motifs$ column-by-column and $d(Consensus(Motifs), Motifs)$ corresponds to adding these elements row-by-row, we obtain the following:

$$Score(Motifs) = d(Consensus(Motifs), Motifs)$$

This equation gives us an idea. Instead of searching for a collection of k -mers $Motifs$ minimizing

$$Score(Motifs)$$

let's instead search for a potential consensus string $Pattern$ minimizing

$$d(Pattern, Motifs)$$

among all possible k -mers $Pattern$ and all possible sets of k -mers $Motifs$ in Dna . This problem is equivalent to the Motif Finding Problem.

Equivalent Motif Finding Problem: Given a collection of strings, find a set of k -mers, one from each sequence, that minimizes the distance between all patterns and all collections of k -mers.

Input: A collection of strings Dna and an integer k .

Output: A k -mer $Pattern$ and a collection of k -mers, one from each string in Dna , minimizing $d(Pattern, Motifs)$ among all possible choices of $Pattern$ and $Motifs$.

Bioinformatics Algorithms

Chapter 1: Where Does DNA

Replication Begin?: ...

Chapter 2: How Do We Sequence

Antibiotics?: ...

Chapter 3: Which DNA Patterns Play

the Role of Molecular Clocks?:

35. [Do We Have a "Clock" Gene?](#)

36. [Motif Finding Is More Difficult Than You Think](#)

37. [Scoring Motifs](#)

38. From Motif Finding to Finding a Median String

39. [Greedy Motif Search](#)

40. [Motif Finding Meets Oliver Cromwell](#)

41. [Randomized Motif Search](#)

42. [How Can a Randomized Algorithm Perform So Well?](#)

43. [Gibbs Sampling](#)

44. [Gibbs Sampling in Action](#)

But wait a second — have we not just made our task more complex? Instead of having to search for all *Motifs*, we now have to search all *Motifs* as well as all *k*-mers *Pattern*. The key observation for solving the Equivalent Motif Finding Problem is that, given *Pattern*, we don't need to explore all possible collections *Motifs* in order to minimize $d(\text{Pattern}, \text{Motifs})$.

To explain how this can be done, we define $\text{Motifs}(\text{Pattern}, \text{Dna})$ as a collection of *k*-mers that minimizes

$$d(\text{Pattern}, \text{Motifs})$$

for a given *Pattern* and all possible sets of *k*-mers *Motifs* in *Dna*. For example, for the strings *Dna* shown below, the five colored 3-mers represent $\text{Motifs}(\text{AAA}, \text{Dna})$.

```
ttaccttAAc
gAtAtctgtc
Acggcgttcg
ccctAAAgag
cgtcAgAggt
```

STOP and Think: Given a *k*-mer *Pattern*, design a fast algorithm for generating $\text{Motifs}(\text{Pattern}, \text{Dna})$.

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action

The reason why we don't need to consider all possible *Motifs* is that we can generate the strings in $\text{Motifs}(\text{Pattern}, \text{Dna})$ one at a time. Given a k -mer *Pattern* and a longer string *Text*, we use $d(\text{Pattern}, \text{Text})$ to denote the minimum Hamming distance between *Pattern* and any k -mer in *Text*. For example:

$$d(\text{GATTCTCA}, \text{GCAAAGACGCTGACCAA}) = 3$$

A k -mer in *Text* that achieves the minimum Hamming distance with *Pattern* is denoted $\text{Motif}(\text{Pattern}, \text{Text})$. For the above example,

$$\text{Motif}(\text{GATTCTCA}, \text{GCAAAGACGCTGACCAA}) = \text{GACGCTGA}$$

We note that the notation $\text{Motif}(\text{Pattern}, \text{Text})$ is slightly ambiguous because there may be multiple k -mers in *Text* that achieve the minimum Hamming distance with *Pattern*. For example, $\text{Motif}(\text{AAG}, \text{GCAATCCTCAGC})$ could be either **AAT** or **CAG**. However, this ambiguity does not affect the analysis below, and so we will keep this definition for the sake of convenience.

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action



Given a k -mer *Pattern* and a set of strings $Dna = \{Dna_1, \dots, Dna_t\}$, we define

$$d(Pattern, Dna) = \sum_{i=1}^t d(Pattern, Dna_i)$$

as the sum of distances between *Pattern* and all strings in *Dna*. For example, for the following strings *Dna*, $d(AAA, Dna) = 1 + 1 + 2 + 0 + 1 = 5$:

```
ttaccttAAc 1
gAtAtctgtc 1
Acggcgttcg 2
ccctAAAgag 0
cgtcAgAggt 1
```

Our goal is to find a k -mer *Pattern* that minimizes $d(Pattern, Dna)$ over all k -mers *Pattern*, the same task that the Equivalent Motif Finding Problem is trying to achieve. We call such a *Pattern* a **median string** for *Dna*.

Median String Problem: Find a median string.

Input: A collection of strings *Dna* and an integer k .

Output: A k -mer *Pattern* that minimizes $d(Pattern, Dna)$ among all k -mers *Pattern*.

Notice that finding a median string requires solving a double minimization problem: we must find a k -mer *Pattern* that minimizes $d(Pattern, Dna)$, where this function is itself computed by taking a minimum over all choices of k -mers from each string in *Dna*.

Bioinformatics Algorithms

Chapter 1: Where Does DNA

Replication Begin?: ...

Chapter 2: How Do We Sequence

Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action

We now give the pseudocode for a brute force solution to the Median String Problem.

```

MEDIAN STRING(Dna, k)
  BestPattern ← AAA...AA
  for each k-mer Pattern from AAA...AA to TTT...TT
    if  $d(\text{Pattern}, \text{Dna}) < d(\text{BestPattern}, \text{Dna})$ 
      BestPattern ← Pattern
  output BestPattern

```

CODE CHALLENGE: Implement MEDIAN STRING.

Input: An integer *k*, followed by a collection of strings *Dna*.

Output: A *k*-mer *Pattern* that minimizes $d(\text{Pattern}, \text{Dna})$ among all *k*-mers *Pattern*.

Sample Input:

```

3
AAATTGACGCAT
GACGACCACGTT
CGTCAGCGCCTG
GCTGAGCACCGG
AGTACGGGACAG

```

Sample Output:

```

GAC

```

[Extra Dataset](#)

[Start Quiz \(limit: 5 minutes\)](#)

Bioinformatics Algorithms

[Chapter 1: Where Does DNA Replication Begin?: ...](#)

[Chapter 2: How Do We Sequence Antibiotics?: ...](#)

[Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?: ...](#)

[35. Do We Have a "Clock" Gene?](#)

[36. Motif Finding Is More Difficult Than You Think](#)

[37. Scoring Motifs](#)

[38. From Motif Finding to Finding a Median String](#)

[39. Greedy Motif Search](#)

[40. Motif Finding Meets Oliver Cromwell](#)

[41. Randomized Motif Search](#)

[42. How Can a Randomized Algorithm Perform So Well?](#)

[43. Gibbs Sampling](#)

[44. Gibbs Sampling in Action](#)



We have demonstrated that the Median String Problem is equivalent to the Equivalent Motif Finding Problem, which in turn is equivalent to the Motif Finding Problem. To see why we have reformulated the Motif Finding Problem, consider the runtime of the algorithms that we have presented. **MEDIANSTRING** computes $d(\text{Pattern}, \text{Dna})$ for each of the 4^k k -mers *Pattern*. Each computation of $d(\text{Pattern}, \text{Dna})$ requires a single pass over each string in *Dna*, which requires $|\text{Pattern}| \cdot |\text{Dna}| = n \cdot k \cdot t$ operations. Therefore, **MEDIANSTRING** has a running time of $O(4^k \cdot n \cdot k \cdot t)$, which in practice compares favorably with the $O(n^2 \cdot k \cdot t)$ running time of **BRUTEFORCEMOTIFSEARCH** because the length of a motif (k) typically does not exceed 20 nucleotides.

The Median String Problem teaches an important lesson, which is that sometimes rethinking how a problem is formulated can lead to dramatic improvements in the runtime required to solve it. In this case, our simple observation that $\text{Score}(\text{Motifs})$ could just as easily be computed row-by-row as column-by-column produced a much faster algorithm in **MEDIANSTRING**.

Of course, the ultimate test of **MEDIANSTRING** is how it performs in practice. Unfortunately, since **MEDIANSTRING** has to consider 4^k k -mers, it becomes too slow for the Subtle Motif Problem, for which $k = 15$. We will run **MEDIANSTRING** with $k = 13$ in the hope that it will capture a substring of the correct 15-mer motif. The algorithm still requires half a day to run on our very fast computer and returns the median string **AAAAA**t**AGaGGGG** (with distance 29). This 13-mer is not a substring of the implanted pattern **AAAAAAAAGGGGGGGG**, but it does come close.

STOP and Think: How can a slightly incorrect median string of length 13 help us find the correct median string of length 15?

Furthermore, we have thus far assumed that the value of k is known in advance, which is not true in practice. As a result, we are forced to run our motif finding algorithms for different values of k and then try to deduce the correct motif length. Since some regulatory motifs are rather long (below in this chapter we will search for a biologically important motif of length 20), **MEDIANSTRING** may be too slow for finding them. We need to design a faster algorithm!

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

- 35. Do We Have a "Clock" Gene?
- 36. Motif Finding Is More Difficult Than You Think
- 37. Scoring Motifs
- 38. From Motif Finding to Finding a Median String**
- 39. Greedy Motif Search
- 40. Motif Finding Meets Oliver Cromwell
- 41. Randomized Motif Search
- 42. How Can a Randomized Algorithm Perform So Well?
- 43. Gibbs Sampling
- 44. Gibbs Sampling in Action

Many algorithms are iterative procedures that must choose among a number of alternatives at each iteration. Some of these alternatives may lead to correct solutions, whereas others may not. **Greedy algorithms** select the “most attractive” alternative at each iteration. For example, a greedy algorithm in chess might attempt to capture an opponent’s most valuable piece at every move. Yet anyone who has played chess knows that this strategy of looking only one move ahead may lead to disastrous results. In general, most greedy algorithms typically fail to find an exact solution of the problem; instead, they are often fast heuristics that find an *approximate* solution. Nevertheless, for many biological problems that we will study in this book, greedy algorithms will prove quite useful.

In this section, we will explore a greedy approach to motif finding. Again, let *Motifs* be a collection of *k*-mers taken from *t* strings *Dna*. Recall from our discussion of entropy that we can view each column of *Profile(Motifs)* as a probability distribution, or a 4-sided biased die. Thus, a profile matrix with *k* columns can be viewed as *k* dice that we can roll to randomly generate a *k*-mer. For example, if the 1st column of the profile matrix is (0.2, 0.1, 0.0, 0.7), then we generate **A** as the 1st nucleotide with probability 0.2, **C** with probability 0.1, **G** with probability 0.0, and **T** with probability 0.7.

Below, we show a profile matrix with one entry in every column boldfaced so that the boldfaced entry in the *i*-th column corresponds to the *i*-th nucleotide in **ACGGGGATTACC**. The probability $\Pr(\text{ACGGGGATTACC} \mid \text{Profile})$ that *Profile* generates **ACGGGGATTACC** is computed by simply multiplying the highlighted entries in the profile matrix.

Profile	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
String		A	C	G	G	G	G	A	T	T	A	C	C
Probability		.2	.6	1	1	.9	.9	.9	.5	.8	.1	.4	.6

= 0.000839808

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action

Profile	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		T	C	G	G	G	G	A	T	T	T	C	C

A k -mer tends to have a higher probability when it is more similar to the consensus string of a profile. For example, for the same *Profile* and its consensus string TCGGGGATTTCC (shown above):

$$\Pr(\text{TCGGGGATTTCC} \mid \text{Profile}) = 0.7 \cdot 0.6 \cdot 1.0 \cdot 1.0 \cdot 0.9 \cdot 0.9 \cdot 0.9 \cdot 0.5 \cdot 0.8 \cdot 0.7 \cdot 0.4 \cdot 0.6 \\ = 0.0205753$$

which is larger than the value $\Pr(\text{ACGGGGATTACC} \mid \text{Profile}) = 0.000839808$ that we computed previously.

EXERCISE BREAK: For the *Profile* above, compute $\Pr(\text{TCGCGGATTTCC} \mid \text{Profile})$.

Start Quiz

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

- 35. Do We Have a "Clock" Gene?
- 36. Motif Finding Is More Difficult Than You Think
- 37. Scoring Motifs
- 38. From Motif Finding to Finding a Median String
- 39. Greedy Motif Search**
- 40. Motif Finding Meets Oliver Cromwell
- 41. Randomized Motif Search
- 42. How Can a Randomized Algorithm Perform So Well?
- 43. Gibbs Sampling
- 44. Gibbs Sampling in Action



Given a profile matrix *Profile*, we can evaluate the probability of every *k*-mer in a string *Text* and find a **Profile-most probable *k*-mer** in *Text*, i.e., a *k*-mer that was most likely to have been generated by *Profile*. For example, **ACGGGGATTACC** is the *Profile*-most probable 12-mer in GGTACGGGGATTACCT. Indeed, every other 12-mer in this string has probability 0.

Profile-most Probable *k*-mer Problem: Find a *Profile*-most probable *k*-mer in a string.

Input: A string *Text*, an integer *k*, and a $k \times 4$ matrix *Profile*.

Output: A *Profile*-most probable *k*-mer in *Text*.

CODE CHALLENGE: Solve the *Profile*-most Probable *k*-mer Problem.

Sample Input:

ACCTGTTTATTGCCTAAGTTCCGAACAAACCCAATATAGCCCGAGGGCCT

5

A C G T

0.2 0.4 0.3 0.1

0.2 0.3 0.3 0.2

0.3 0.1 0.5 0.1

0.2 0.5 0.2 0.1

0.3 0.1 0.4 0.2

Sample Output:

CCGAG

Extra Dataset

Start Quiz (limit: 5 minutes)

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action

The greedy motif search algorithm iteratively finds k -mers in the 1st, string from Dna , 2nd string from Dna , 3rd string from Dna , etc. After finding $i - 1$ k -mers $Motifs$ in the first $i - 1$ strings of Dna , this algorithm constructs $Profile(Motifs)$ and selects the $Profile$ -most probable k -mer from the i -th string based on this profile matrix (ties are broken arbitrarily).

To form the initial motif matrix $BestMotifs$, the algorithm starts by selecting arbitrary k -mers in each string from Dna ; the following pseudocode selects the first k -mer in each string.

```

GREEDYMOTIFSEARCH( $Dna, k, t$ )
  form a set of  $k$ -mers  $BestMotifs$  by selecting 1st  $k$ -mers in each string from  $Dna$ 
  for each  $k$ -mer  $Motif$  in the 1st string from  $Dna$ 
     $Motif_1 \leftarrow Motif$ 
  for  $i = 2$  to  $t$ 
    form  $Profile$  from motifs  $Motif_1, \dots, Motif_{i-1}$ 
     $Motif_i \leftarrow Profile$ -most probable  $k$ -mer in the  $i$ -th string in  $Dna$ 
   $Motifs \leftarrow (Motif_1, \dots, Motif_t)$ 
  if  $Score(Motifs) < Score(BestMotifs)$ 
     $BestMotifs \leftarrow Motifs$ 
  output  $BestMotifs$ 

```

Bioinformatics Algorithms

Chapter 1: Where Does DNA

Replication Begin?: ...

Chapter 2: How Do We Sequence

Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action



CODE CHALLENGE: Implement `GREEDYMOTIFSEARCH`.

Input: Integers k and t , followed by a collection of strings Dna .

Output: A collection of strings *BestMotifs* resulting from applying `GREEDYMOTIFSEARCH(Dna, k, t)`. If at any step you find more than one *Profile*-most probable k -mer in a given string, use the one occurring first.

Sample Input:

```
3 5
GGCGTTCAGGCA
AAGAATCAGTCA
CAAGGAGTTCGC
CACGTCATCAC
CAATAATATTCG
```

Sample Output:

```
CAG
CAG
CAA
CAA
CAA
```

Extra Dataset

Start Quiz (limit: 5 minutes)

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. [Do We Have a "Clock" Gene?](#)

36. [Motif Finding Is More Difficult Than You Think](#)

37. [Scoring Motifs](#)

38. [From Motif Finding to Finding a Median String](#)

39. Greedy Motif Search

40. [Motif Finding Meets Oliver Cromwell](#)

41. [Randomized Motif Search](#)

42. [How Can a Randomized Algorithm Perform So Well?](#)

43. [Gibbs Sampling](#)

44. [Gibbs Sampling in Action](#)



In contrast to `MEDIANSTRING`, `GREEDYMOTIFSEARCH` is fast and can be run with $k = 15$ to solve the Subtle Motif Problem (recall that we settled for $k = 13$ in the case of `MEDIANSTRING`). However, it trades speed for accuracy and returns `gtAAAtAgaGatGtG` (total distance: 58), which is very different from the true implanted motif `AAAAAAAAGGGGGGG`.

STOP and Think: Why does `GREEDYMOTIFSEARCH` (reproduced below for your convenience) perform so poorly?

```

GREEDYMOTIFSEARCH(Dna, k, t)
  form a set of k-mers BestMotifs by selecting 1st k-mers in each string from Dna
  for each k-mer Motif in the 1st string from Dna
    Motif1 ← Motif
  for i = 2 to t
    form Profile from motifs Motif1, ..., Motifi-1
    Motifi ← Profile-most probable k-mer in the i-th string in Dna
  Motifs ← (Motif1, ..., Motift)
  if Score(Motifs) < Score(BestMotifs)
    BestMotifs ← Motifs
  output BestMotifs

```

Bioinformatics Algorithms

Chapter 1: Where Does DNA

Replication Begin?: ...

Chapter 2: How Do We Sequence

Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action



At first glance, GREEDYMOTIFSEARCH may seem like a reasonable algorithm, but it is not! Let's see whether GREEDYMOTIFSEARCH will find the (4, 1)-motif **ACGT** implanted in the following strings *Dna*:

```
ttACCTtaac
gATGTctgtc
acgGCGTtag
ccctaACGAg
cgtagAGGT
```

The algorithm is now ready to search for a *Profile*-most probable 4-mer in the 2nd sequence. The issue, however, is that there are so many zeros in the profile matrix that the probability of every 4-mer but **ACCT** is 0! Thus, unless **ACCT** is present in every string in *Dna*, there is little chance that GREEDYMOTIFSEARCH will find the implanted motif. Zeroes in the profile matrix are not just a minor annoyance but rather a persistent problem that we must address.

Bioinformatics Algorithms

Chapter 1: Where Does DNA

Replication Begin?: ...

Chapter 2: How Do We Sequence

Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action



In 1650, after the Scots proclaimed Charles II as king, English statesman Oliver Cromwell made a famous appeal to the Church of Scotland, urging them to see the error of their royal alliance:

I beseech you, in the bowels of Christ, think it possible that you may be mistaken.

His appeal was rejected, and Cromwell invaded Scotland in response. His quote later inspired the statistical maxim called **Cromwell's rule**, which states that we should not use probabilities of 0 or 1 unless we are talking about logical statements that can only be true or false. In other words, we should allow a small probability for extremely unlikely events, such as "this book was written by aliens" or "the sun will not rise tomorrow". We cannot speak to the likelihood of the former event, but in the 18th Century, the great French mathematician Pierre-Simon Laplace actually estimated the probability that the sun will not rise tomorrow (1 in 1826251), given that it has risen every day for the past 5000 years. Although his estimate was ridiculed by contemporaries, Laplace's approach to this question now plays an important role in statistics.

In any observed data set, there is the possibility, especially with low-probability events and with small data sets, of a possible event not occurring. Its observed frequency is therefore zero; however, setting the empirical probability of the event to 0 represents an inaccurate oversimplification that may cause problems in randomized algorithms. By artificially adjusting the probability of rare (but not impossible) events, these problems can be mitigated.

Bioinformatics Algorithms

Chapter 1: Where Does DNA

Replication Begin?: ...

Chapter 2: How Do We Sequence

Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action

Cromwell's rule is relevant to the calculation of the probability of a string based on a profile matrix. For example, consider the following *Profile*.

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

= 0

As illustrated above, $\Pr(\text{TCGTGGATTCC} \mid \text{Profile})$ is equal to 0 because TCGTGGATTCC has a symbol corresponding to zero probability at just a single position, and so the entire string is assigned a zero probability, even though TCGTGGATTCC differs from the consensus string only at the fourth position. As a result, TCGTGGATTCC receives the same probability as AAATCTTGGAA, which is very different from the consensus string. In order to avoid columns that contain zeroes in profiles, bioinformaticians often substitute zeroes with small numbers called **pseudocounts**.

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

The simplest approach to introducing pseudocounts, called Laplace's Rule of Succession, is similar to the principle that Laplace used to calculate the probability that the sun will not rise tomorrow. In the case of motifs, pseudocounts often amount to adding 1 to each element of $\text{Count}(\text{Motifs})$. For example, say we have the following motif, count, and profile matrices:

Motifs		T	A	A	C
		G	T	C	T
		A	C	T	A
		A	G	G	T
Count	A:	2	1	1	1
	C:	0	1	1	1
	G:	1	1	1	0
	T:	1	1	1	2
Profile	A:	2/4	1/4	1/4	1/4
	C:	0	1/4	1/4	1/4
	G:	1/4	1/4	1/4	0
	T:	1/4	1/4	1/4	2/4

Laplace's Rule of Succession updates these count and profile matrices to the following:

Count	A:	2+1	1+1	1+1	1+1
	C:	0+1	1+1	1+1	1+1
	G:	1+1	1+1	1+1	0+1
	T:	1+1	1+1	1+1	2+1
Profile	A:	3/8	2/8	2/8	2/8
	C:	1/8	2/8	2/8	2/8
	G:	2/8	2/8	2/8	1/8
	T:	2/8	2/8	2/8	3/8

STOP and Think: How would you use Laplace's Rule of Succession to address the shortcomings of GREEDYMOTIFSEARCH?

Bioinformatics Algorithms

Chapter 1: Where Does DNA

Replication Begin?: ...

Chapter 2: How Do We Sequence

Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

The only change we need to introduce to **GREEDYMOTIFSEARCH** in order to eliminate zeroes from the profile matrices that it constructs is to replace the sixth line of pseudocode in **GREEDYMOTIFSEARCH**, which is highlighted below in green.

```

GREEDYMOTIFSEARCH(Dna, k, t)
  form a set of k-mers BestMotifs by selecting 1st k-mers in each string from Dna
  for each k-mer Motif in the 1st k-mers in each string from Dna
    Motif1 ← Motif
    for i = 2 to t
      form Profile from motifs Motif1, ..., Motifi-1
      Motifi ← Profile-most probable k-mer in the i-th string in Dna
    Motifs ← (Motif1, ..., Motift)
    if Score(Motifs) < Score(BestMotifs)
      BestMotifs ← Motifs
  output BestMotifs
  
```

After incorporating Laplace's Rule of Succession, **GREEDYMOTIFSEARCH** is given by the following:

```

GREEDYMOTIFSEARCH(Dna, k, t)
  form a set of k-mers BestMotifs by selecting 1st k-mers in each string from Dna
  for each k-mer Motif in the 1st k-mers in each string from Dna
    Motif1 ← Motif
    for i = 2 to t
      apply Laplace's Rule of Succession to form Profile from motifs Motif1, ..., Motifi-1
      Motifi ← Profile-most probable k-mer in the i-th string in Dna
    Motifs ← (Motif1, ..., Motift)
    if Score(Motifs) < Score(BestMotifs)
      BestMotifs ← Motifs
  output BestMotifs
  
```

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. [Do We Have a "Clock" Gene?](#)

36. [Motif Finding Is More Difficult Than You Think](#)

37. [Scoring Motifs](#)

38. [From Motif Finding to Finding a Median String](#)

39. [Greedy Motif Search](#)

40. Motif Finding Meets Oliver Cromwell

41. [Randomized Motif Search](#)

42. [How Can a Randomized Algorithm Perform So Well?](#)

43. [Gibbs Sampling](#)

Stepic



Full screen

We now apply Laplace's Rule of Succession to search for a (4, 1)-motif **ACCT** implanted in the following strings *Dna*:

```

ttACCTtaac
gATGTctgtc
acgGCGTtag
ccctaACGAg
cgtcagAGGT

```

Again, let's assume that the algorithm has already chosen the implanted 4-mer **ACCT** from the 1st sequence and construct the corresponding profile matrix using Laplace's Rule of Succession:

Count	A:	1+1	0+1	0+1	0+1
	C:	0+1	1+1	1+1	0+1
	G:	1+1	1+1	1+1	0+1
	T:	1+1	1+1	1+1	2+1

Profile	A:	2/5	1/5	1/5	1/5
	C:	1/5	2/5	2/5	1/5
	G:	1/5	1/5	1/5	1/5
	T:	1/5	1/5	1/5	2/5

We now use this profile matrix to compute the probabilities of all 4-mers in the 2nd row of *Dna*:

g ATG	ATGT	TGTc	GTct	Tctg	ctgt	tgtc
$1/5^4$	$4/5^4$	$1/5^4$	$4/5^4$	$2/5^4$	$2/5^4$	$1/5^4$

There are two *Profile*-most probable 4-mers in the second sequence (**ATGT** and **GTct**); let's assume that we get lucky again and choose the implanted 4-mer **ATGT**.

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

We now have the following motif matrix with corresponding count and profile matrices:

Motifs	A	C	C	T
	A	T	G	T
Count	A: 2+1 0+1 0+1 0+1	C: 0+1 1+1 1+1 0+1	G: 0+1 0+1 1+1 0+1	T: 0+1 1+1 0+1 2+1
Profile	A: 3/6 1/6 1/6 1/6	C: 1/6 2/6 2/6 1/6	G: 1/6 1/6 2/6 1/6	T: 1/6 2/6 1/6 3/6

We now use this profile matrix to compute the probabilities of all 4-mers in the 3rd row of *Dna*:

acgG	cgGC	gGCG	GCGT	CGTt	GTta	Ttag
12/6 ⁴	2/6 ⁴	2/6 ⁴	12/6 ⁴	3/6 ⁴	2/6 ⁴	2/6 ⁴

Again, there are two *Profile*-most probable 4-mers in the second sequence (acgG and GCGT). This time, we will assume that we don't get lucky; we choose acgG instead of the implanted 4-mer GCGT.

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

We now have the following motif, count, and profile matrices:

Motifs	A	C	C	T
	A	T	G	T
	a	c	g	G
Count	A: 3+1	0+1	0+1	0+1
	C: 0+1	2+1	1+1	0+1
	G: 0+1	0+1	2+1	1+1
	T: 0+1	1+1	0+1	2+1
Profile	A: 4/7	1/7	1/7	1/7
	C: 1/7	3/7	2/7	1/7
	G: 1/7	1/7	3/7	2/7
	T: 1/7	2/7	1/7	3/7

We use this profile to compute probabilities of all 4-mers in the 4th row of *Dna*:

ccct	ccta	ctaA	taAC	aACG	ACGA	CGAg
$18/7^4$	$3/7^4$	$2/7^4$	$1/7^4$	$16/7^4$	$36/7^4$	$2/7^4$

Despite the fact that we missed the implanted 4-mer in the 3rd sequence, we have now found the implanted 4-mer in the 4th sequence as the *Profile*-most probable 4-mer **ACGA**.

Bioinformatics Algorithms

Chapter 1: Where Does DNA

Replication Begin?: ...

Chapter 2: How Do We Sequence

Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling



We have the following motif, count, and profile matrices:

Motifs	A	C	C	T
	A	T	G	T
	a	c	g	G
	A	C	G	A
Count	A: 4+1	0+1	0+1	1+1
	C: 0+1	3+1	1+1	0+1
	G: 0+1	0+1	3+1	1+1
	T: 0+1	1+1	0+1	2+1
Profile	A: 5/8	1/8	1/8	2/8
	C: 1/8	4/8	2/8	1/8
	G: 1/8	1/8	4/8	2/8
	T: 1/8	2/8	1/8	3/8

We now use this profile to compute the probabilities of all 4-mers in the 5th row of *Dna*:

cgtc	gtca	tcag	cagA	agAG	gAGG	AGGT
$1/8^4$	$8/8^4$	$8/8^4$	$8/8^4$	$10/8^4$	$8/8^4$	$60/8^4$

The *Profile*-most probable 4-mer in the 5th row is **AGGT**, the implanted 4-mer. **GREEDYMOTIFSEARCH** has produced the following motif matrix, which implies the correct consensus string **ACGT**:

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

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

Now that you have seen `GREEDYMOTIFSEARCH` perform well on a sample dataset, update your `GREEDYMOTIFSEARCH` implementation to incorporate pseudocounts.

CODE CHALLENGE: Implement `GREEDYMOTIFSEARCH` with pseudocounts.

Input: Integers k and t , followed by a collection of strings Dna .

Output: A collection of strings *BestMotifs* resulting from applying `GREEDYMOTIFSEARCH(Dna, k, t)` with pseudocounts. If at any step you find more than one *Profile*-most probable k -mer in a given string, use the one occurring first.

Sample Input:

```
3 5
GGCGTTCAGGCA
AAGAATCAGTCA
CAAGGAGTTCGC
CACGTCAATCAC
CAATAATATTCG
```

Sample Output:

```
TTC
ATC
TTC
ATC
TTC
```

Extra Dataset

Start Quiz (limit: 5 minutes)

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

- 35. [Do We Have a "Clock" Gene?](#)
- 36. [Motif Finding Is More Difficult Than You Think](#)
- 37. [Scoring Motifs](#)
- 38. [From Motif Finding to Finding a Median String](#)
- 39. [Greedy Motif Search](#)
- 40. Motif Finding Meets Oliver Cromwell**
- 41. [Randomized Motif Search](#)
- 42. [How Can a Randomized Algorithm Perform So Well?](#)
- 43. [Gibbs Sampling](#)
- 44. [Gibbs Sampling in Action](#)

Applying GREEDYMOTIFSEARCH with pseudocounts to the Subtle Motif Problem returns a collection of 15-mers *Motifs* with $\text{Score}(\text{Motifs}) = 41$ and $\text{Consensus}(\text{Motifs}) = \text{AAAAA}\text{tAgaGGGG}\text{tt}$. Thus, Laplace's Rule of Succession provided a significant improvement over the original GREEDYMOTIFSEARCH, which returned the consensus string gtAAAtAgaGatGtG with $\text{Score}(\text{Motifs}) = 58$.

You may be satisfied with the performance of GREEDYMOTIFSEARCH, but you should know by now that your authors are never satisfied – can we design an even more accurate motif finding algorithm?

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling



We will now turn to **randomized algorithms** that toss coins and roll dice in order to search for motifs. Making random algorithmic decisions may sound like a disastrous idea; just imagine a chess game in which every move would be decided by rolling a die. However, an 18th Century French mathematician and naturalist, Comte de Buffon, first proved that randomized algorithms are useful by randomly dropping needles onto parallel strips of wood and using the results of this experiment to accurately approximate the constant π (see [Detour: Buffon's Needle](#)).

Randomized algorithms may be nonintuitive because they lack the control of traditional algorithms, but they can offer an advantage in problems for which polynomial-time algorithms are unknown. Most randomized algorithms typically fail to find an exact solution of the problem; instead, they represent fast algorithms to find an *approximate* solution. Because of their speed, randomized algorithms can be run many times, allowing us to choose their best approximation among thousands of runs.

Although most randomized algorithms are **Monte Carlo algorithms**, which do not guarantee an exact solution, some other randomized algorithms (called **Las Vegas algorithms**) deliver solutions that are guaranteed to be exact, despite the fact that they rely on making random decisions. The randomized motif finding algorithms that we will consider in this chapter are Monte Carlo algorithms.

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. [Do We Have a "Clock" Gene?](#)

36. [Motif Finding Is More Difficult Than You Think](#)

37. [Scoring Motifs](#)

38. [From Motif Finding to Finding a Median String](#)

39. [Greedy Motif Search](#)

40. [Motif Finding Meets Oliver Cromwell](#)

41. Randomized Motif Search

42. [How Can a Randomized Algorithm Perform So Well?](#)

43. [Gibbs Sampling](#)



We previously defined *Profile*(*Motifs*) as the profile matrix constructed from a collection of *k*-mers *Motifs* in *Dna*. Now, given a collection of strings *Dna* and an arbitrary $4 \times k$ matrix *Profile*, we define *Motifs*(*Profile*, *Dna*) as a collection of *k*-mers formed by the *Profile*-most probable *k*-mers in each sequence from *Dna*. For example, consider the following *Profile* and *Dna*:

<i>Profile</i>	A:	4/5	0	0	1/5	<i>Dna</i>	ttaccta
	C:	0	3/5	1/5	0		gagtcgtc
	G:	1/5	1/5	4/5	0		acggcgtag
	T:	0	1/5	0	4/5		cccta
							cgtagaggt

Taking the *Profile*-most probable 4-mer from each row of *Dna* produces the following 4-mers (shown in red):

Motifs(*Profile*, *Dna*)

```

ttACCTaac
gATGTctgtc
acgGCGTtag
ccctaACGAg
cgtagAGGT
  
```

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

Stepic



Full screen

In general, we can begin from a collection of randomly chosen k -mers $Motifs$ in Dna , construct $Profile(Motifs)$, and use this profile to generate a new collection of k -mers:

$Motifs(Profile(Motifs), Dna)$

Why would we do this? Because the hope is that $Motifs(Profile(Motifs), Dna)$ has a better score than the original collection of k -mers $Motifs$. We can then form the profile matrix of these k -mers, $Profile(Motifs(Profile(Motifs), Dna))$, and use it to form the most probable k -mers, $Motifs(Profile(Motifs(Profile(Motifs), Dna)), Dna)$. We can continue to iterate...

$\dots Motifs(Profile(Motifs(Profile(Motifs), Dna)), Dna), Dna) \dots$

...for as long as the score of the constructed motifs keeps improving, which is exactly what **RANDOMIZEDMOTIFSEARCH** does. To implement this algorithm, you will need to randomly select the initial collection of k -mers that form the motif matrix $Motifs$. To do so, you will need a **random number generator** (denoted $Random(N)$) that is equally likely to return any integer from 1 to N . You might like to think about this random number generator as an unbiased N -sided die.

```
RANDOMIZEDMOTIFSEARCH( $Dna, k, t$ )
  randomly select  $k$ -mers  $Motifs = (Motif_1, \dots, Motif_t)$  in each string from  $Dna$ 
   $BestMotifs \leftarrow Motifs$ 
  while forever
     $Profile \leftarrow Profile(Motifs)$ 
     $Motifs \leftarrow Motifs(Profile, Dna)$ 
    if  $Score(Motifs) < Score(BestMotifs)$ 
       $BestMotifs \leftarrow Motifs$ 
  else
    output  $BestMotifs$ 
  return
```

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. [Do We Have a "Clock" Gene?](#)

36. [Motif Finding Is More Difficult Than You Think](#)

37. [Scoring Motifs](#)

38. [From Motif Finding to Finding a Median String](#)

39. [Greedy Motif Search](#)

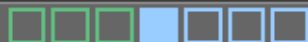
40. [Motif Finding Meets Oliver Cromwell](#)

41. Randomized Motif Search

42. [How Can a Randomized Algorithm Perform So Well?](#)

43. [Gibbs Sampling](#)

Stepic



Full screen

CODE CHALLENGE: Implement `RANDOMIZEDMOTIF SEARCH`.

Input: Integers k and t , followed by a collection of strings Dna .

Output: A collection *BestMotifs* resulting from running `RANDOMIZEDMOTIF SEARCH(Dna, k, t)` 1000 times.

Sample Input:

```
8 5
CGCCCTCTCGGGGTGTTTCAGTAAACGGCCA
GGGCGAGGTATGTGTAAGTGCCAAGGTGCCAG
TAGTACCGAGACCGAAAGAAGTATACAGGCGT
TAGATCAAGTTTCAGGTGCACGTCGGTGAACC
AATCCACCAGCTCCACGTGCAATGTTGGCCTA
```

Sample Output:

```
TCTCGGGG
CCAAGGTG
TACAGGCG
TTCAGGTG
TCCACGTG
```

Note: Because randomized algorithms are difficult for us to test, there is a very small chance that your algorithm may be implemented correctly but not return the correct answer. We suggest running your algorithm on another dataset if you do not get the correct answer the first time.

Start Quiz (limit: 5 minutes)

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

- 35. [Do We Have a "Clock" Gene?](#)
- 36. [Motif Finding Is More Difficult Than You Think](#)
- 37. [Scoring Motifs](#)
- 38. [From Motif Finding to Finding a Median String](#)
- 39. [Greedy Motif Search](#)
- 40. [Motif Finding Meets Oliver Cromwell](#)
- 41. Randomized Motif Search**
- 42. [How Can a Randomized Algorithm Perform So Well?](#)
- 43. [Gibbs Sampling](#)



At first glance, **RANDOMIZEDMOTIFSEARCH** appears to be doomed. How can this algorithm, which starts from a random guess, possibly find anything useful? To explore **RANDOMIZEDMOTIFSEARCH**, let's run it on five short strings with the implanted (4,1)-motif **ACGT** (shown in blue) and imagine that it chooses the following 4-mers *Motifs* (shown in bold) at the first iteration. As expected, it misses the implanted motif in nearly every string.

Dna

```

ttACCTtaac
gATGTctgtc
ccgGCGTtag
cactaACGAg
cgtcagAGGT

```

We now construct the profile matrix *Profile(Motifs)* of the chosen 4-mers:

		t	a	a	c
	<i>Motifs</i>	G	T	c	t
		c	c	g	G
		a	c	t	a
		A	G	G	T
<i>Profile(Motifs)</i>	A:	0.4	0.2	0.2	0.2
	C:	0.2	0.4	0.2	0.2
	G:	0.2	0.2	0.4	0.2
	T:	0.2	0.2	0.2	0.4

and compute the probabilities of every 4-mer in *Dna* based on this profile. For example, the probability of the first 4-mer in the first sequence in *Dna* is $\Pr(\text{ttAC}|\text{Profile}) = 0.2 \cdot 0.2 \cdot 0.2 \cdot 0.2 = 0.0016$.

tt AC (.0016)	t ACC (.0016)	ACCT (.0128)	CCTt (.0064)	CTta (.0016)	Ttaa (.0016)	taac (.0016)
g ATG (.0016)	ATGT (.0128)	TGTc (.0016)	Gtct (.0032)	Tctg (.0032)	ctgt (.0032)	tgtc (.0016)
cc gG (.0064)	c gGC (.0032)	gGCG (.0016)	GCGT (.0128)	CGTt (.0032)	GTta (.0016)	Ttag (.0016)
Cact (.0032)	acta (.0064)	ctaA (.0016)	ta AC (.0016)	a ACG (.0032)	ACGA (.0128)	CGAg (.0016)
cgtc (.0016)	gtca (.0016)	tcag (.0016)	cagA (.0032)	ag AG (.0032)	gAGG (.0032)	AGGT (.0128)

Bioinformatics Algorithms

Chapter 1: Where Does DNA

Replication Begin?: ...

Chapter 2: How Do We Sequence

Antibiotics?: ...

Chapter 3: Which DNA Patterns Play

the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action



The maximum probabilities in every row are shown in green below.

ttAC (.0016)	tACC (.0016)	ACCT (.0128)	CCTt (.0064)	CTta (.0016)	Ttaa (.0016)	taac (.0016)
gATG (.0016)	ATGT (.0128)	TGTc (.0016)	GTct (.0032)	Tctg (.0032)	ctgt (.0032)	tgtc (.0016)
ccgG (.0064)	cgGC (.0036)	gGCG (.0016)	GCGT (.0128)	CGTt (.0032)	GTta (.0016)	Ttag (.0016)
Cact (.0032)	acta (.0064)	ctaA (.0016)	taAC (.0016)	aACG (.0032)	ACGA (.0128)	CGAg (.0016)
cgtc (.0016)	gtca (.0016)	tcag (.0016)	cagA (.0032)	agAG (.0032)	gAGG (.0032)	AGGT (.0128)

We now take the five most probable 4-mers (ACCT, ATGT, GCGT, ACGA, and AGGT) as our new collection *Motifs*:

```

ttACCTtaac
gATGTctgtc
ccgGCGTtag
cactaACGA
cgtcagAGGT
  
```

and miraculously, their consensus string reveals the implanted motif **ACGT**!

```

Motifs(Profile(Motifs), Dna)
      A   C   C   T
      A   T   G   T
      G   C   G   T
      A   C   G   A
      A   G   G   T

Profile(Motifs(Profile(Motifs), Dna))
      A: 0.8 0.0 0.0 0.2
      C: 0.0 0.6 0.2 0.0
      G: 0.2 0.2 0.8 0.0
      T: 0.0 0.2 0.0 0.8
  
```

STOP and Think: How is it possible that randomly chosen k -mers have led us to the correct implanted k -mer? If you think we manufactured this example, select your own initial 4-mers and see what happens.

Bioinformatics Algorithms

Chapter 1: Where Does DNA

Replication Begin?: ...

Chapter 2: How Do We Sequence

Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action



For the Subtle Motif Problem with implanted 15-mer `AAAAAAAAAGGGGGG`, when we run `RANDOMIZEDMOTIFSEARCH` 100,000 times (each time with new randomly selected k -mers), it returns the following 15-mers (as the lowest scoring motif across all iterations), resulting in the consensus string `AAAAAAAAACAGGGG` with score 42.

		Distance
<i>Motifs</i>	<code>AAAtAcAgACAGcGt</code>	5
	<code>AAAAAAtAGCAGGGt</code>	3
	<code>tAAAAtAAACAGcGG</code>	3
	<code>AcAgAAAAAaAGGGG</code>	3
	<code>AAAAtAAAACTGcGa</code>	4
	<code>AtAgAcgAACAcGGt</code>	6
	<code>cAAAAgAgaAGGGG</code>	4
	<code>AtAgAAAAggAaGGG</code>	5
	<code>AAGAAAAAgAGaGG</code>	3
	<code>cAtAAtgAACTGtGa</code>	6
<i>Consensus(Motifs)</i>	<code>AAAAAAAAACAGGGG</code>	42

This motif is only slightly less conserved than the collection of implanted (15, 4)-motifs with score 40 (or the motif returned by `GREEDYMOTIFSEARCH` with score 41), and it largely captures the implanted motif. In contrast with `GREEDYMOTIFSEARCH`, `RANDOMIZEDMOTIFSEARCH` can be run for a larger number of iterations to discover better and better motifs.

STOP and Think: Does your run of `RANDOMIZEDMOTIFSEARCH` return a similar consensus string? How many times do you need to run `RANDOMIZEDMOTIFSEARCH` to get the implanted (15, 4)-motif with distance 40?

Although the motifs returned by `RANDOMIZEDMOTIFSEARCH` are slightly less conserved than the motifs returned by `MEDIANSERACH`, `RANDOMIZEDMOTIFSEARCH` has the advantage of being able to find longer motifs (since `MEDIANSERACH` becomes too slow for longer motifs). In the Epilogue, we will see that this feature is important in practice.

Bioinformatics Algorithms

Chapter 1: Where Does DNA

Replication Begin?: ...

Chapter 2: How Do We Sequence

Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action



If the strings in *Dna* were truly random, then we would expect that all nucleotides in the selected *k*-mers would be equally likely, resulting in an expected *Profile* in which every entry is approximately 0.25:

```
A: 0.25 0.25 0.25 0.25
C: 0.25 0.25 0.25 0.25
G: 0.25 0.25 0.25 0.25
T: 0.25 0.25 0.25 0.25
```

Such a **uniform profile** is essentially useless for motif finding because no string is more probable than any others according to this profile and because it does not provide any clues on what an implanted motif looks like.

At the opposite end of the spectrum, if we were incredibly lucky, we would choose the implanted *k*-mers *Motifs* from the very beginning, resulting in the profile that we obtained at the end of the previous section:

```
A: 0.8 0.0 0.0 0.2
C: 0.0 0.6 0.2 0.0
G: 0.2 0.2 0.8 0.0
T: 0.0 0.2 0.0 0.8
```

In practice, we are likely to obtain a profile somewhere in between these two extremes, such as the following:

```
A: 0.4 0.2 0.2 0.2
C: 0.2 0.4 0.2 0.2
G: 0.2 0.2 0.4 0.2
T: 0.2 0.2 0.2 0.4
```

This profile matrix has already started to point us toward the implanted motif ACGT, i.e., ACGT is the most likely 4-mer that can be generated by this profile. **RANDOMIZEDMOTIFSEARCH** is designed so that subsequent steps have a good chance of leading us toward this implanted motif (although it is not certain).

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

- 35. [Do We Have a "Clock" Gene?](#)
- 36. [Motif Finding Is More Difficult Than You Think](#)
- 37. [Scoring Motifs](#)
- 38. [From Motif Finding to Finding a Median String](#)
- 39. [Greedy Motif Search](#)
- 40. [Motif Finding Meets Oliver Cromwell](#)
- 41. [Randomized Motif Search](#)
- 42. How Can a Randomized Algorithm Perform So Well?**
- 43. [Gibbs Sampling](#)



If you still doubt the efficacy of a randomized algorithm, consider the following argument. We have already noticed that if *Dna* were random strings, then `RANDOMIZEDMOTIFSEARCH` would start from a nearly uniform profile, and there would be nothing to work with. However, the key observation is that the strings in *Dna* are not random because they include the implanted motif! And these multiple occurrences of the same motif may create a bias in the profile matrix, directing it away from the uniform profile and in the direction of the implanted motif. If you look at the original (randomly) selected *k*-mers (shown in bold below):

```
ttACCTtaac
gATGTctgtc
ccgGCGTtag
cactaACGAg
cgtcagAGGT
```

you will see that the bold 4-mer **AGGT** in the last sequence happened to capture the implanted motif simply by chance. In fact, the profile formed from the remaining 4-mers **taac**, **GTct**, **ccgG**, and **acta** is uniform.

EXERCISE BREAK: Compute the probability that 10 randomly selected 15-mers from the ten 600-nucleotide long strings in the Subtle Motif Problem capture at least one implanted 15-mer. (Allowable error: 0.000001)

Start Quiz

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

- 35. Do We Have a "Clock" Gene?
- 36. Motif Finding Is More Difficult Than You Think
- 37. Scoring Motifs
- 38. From Motif Finding to Finding a Median String
- 39. Greedy Motif Search
- 40. Motif Finding Meets Oliver Cromwell
- 41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

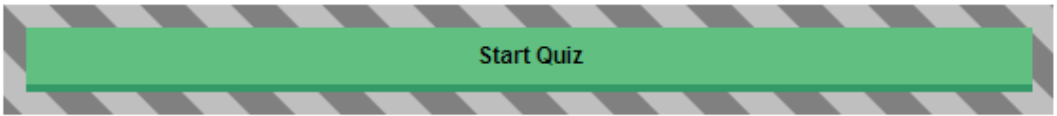
- 43. Gibbs Sampling



Although the probability that randomly selected k -mers match *all* implanted motifs on the first try is negligible, the probability that they capture *at least one* implanted motif is significant. And even in the case of difficult motif finding problems for which this probability is small, we are still running **RANDOMIZEDMOTIFSEARCH** so many times that it will almost certainly catch one implanted motif, thus creating a statistical bias pointing toward the correct motif.

Unfortunately, capturing a single implanted motif is often insufficient to steer **RANDOMIZEDMOTIFSEARCH** to an optimal solution. Therefore, since the computational space of all starting positions of k -mers is huge, the strategy of randomly selected motifs is often not as successful as in the simple example above: the chance that these randomly selected k -mers will be able to guide us to the optimal solution is relatively small. Thus, **RANDOMIZEDMOTIFSEARCH** is typically run a large number of times, followed by selecting the best-scoring motif across all runs.

EXERCISE BREAK: Compute the probability that 10 randomly selected 15-mers from the ten 600-nucleotide long strings in the Subtle Motif Problem capture at least two implanted 15-mers. (Allowable error: 0.000001)

Start Quiz

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

- 35. Do We Have a "Clock" Gene?
- 36. Motif Finding Is More Difficult Than You Think
- 37. Scoring Motifs
- 38. From Motif Finding to Finding a Median String
- 39. Greedy Motif Search
- 40. Motif Finding Meets Oliver Cromwell
- 41. Randomized Motif Search
- 42. How Can a Randomized Algorithm Perform So Well?**
- 43. Gibbs Sampling

Note that **RANDOMIZEDMOTIFSEARCH** may change all t motifs in $Motifs = (Motif_1, \dots, Motif_t)$ in a single iteration. This strategy may prove reckless, since some correct motifs (captured in $Motifs$) may potentially be discarded at the next iteration. **GIBBSAMPLER** is a more cautious iterative algorithm that discards a single k -mer from the current set of motifs at each iteration and replaces it with a new one (or keeps it). It thus moves with more caution in the space of all motifs, as illustrated below.

```
ttaccttaac   ttaccttaac
gatatctgtc   gatatctgtc
acggcggttcg → acggcggttcg
ccctaaagag   ccctaaagag
cgtcagaggt   cgtcagaggt
```

RANDOMIZEDMOTIFSEARCH
(may change all k -mers in a single step)

```
ttaccttaac   ttaccttaac
gatatctgtc   gatatctgtc
acggcggttcg → acggcggttcg
ccctaaagag   ccctaaagag
cgtcagaggt   cgtcagaggt
```

GIBBSAMPLER
(changes one k -mer in a single step)

Like **RANDOMIZEDMOTIFSEARCH**, **GIBBSAMPLER** starts with randomly chosen k -mers in each of t DNA sequences, but it makes a random rather than a deterministic choice at each iteration. It uses randomly selected k -mers ($Motif_1, \dots, Motif_t$) to come up with another (hopefully higher scoring) set of k -mers. In contrast with **RANDOMIZEDMOTIFSEARCH** (which deterministically defines new motifs as $Motifs(Profile(Motifs), Dna)$), **GIBBSAMPLER** randomly selects an integer i between 1 and t and then randomly changes only one $Motif_i$.

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

- 35. Do We Have a "Clock" Gene?
- 36. Motif Finding Is More Difficult Than You Think
- 37. Scoring Motifs
- 38. From Motif Finding to Finding a Median String
- 39. Greedy Motif Search
- 40. Motif Finding Meets Oliver Cromwell
- 41. Randomized Motif Search
- 42. How Can a Randomized Algorithm Perform So Well?
- 43. Gibbs Sampling**
- 44. Gibbs Sampling in Action



To describe how `GIBBSAMPLER` updates *Motifs*, we will need a slightly more advanced random number generator. Given a probability distribution (p_1, \dots, p_n) , this random number generator, denoted $\text{Random}(p_1, \dots, p_n)$, models an n -sided biased die and returns integer i with probability p_i . For example, the standard 6-sided fair die represents the random number generator $\text{Random}(1/6, 1/6, 1/6, 1/6, 1/6, 1/6)$, whereas a biased die might represent the random number generator $\text{Random}(0.1, 0.2, 0.3, 0.05, 0.1, 0.25)$. `GIBBSAMPLER` further generalizes the random number generator by using the function $\text{Random}(p_1, \dots, p_n)$ defined for any set of non-negative numbers, i.e., not necessarily satisfying the condition that the p_i sum to 1. If the p_i sum to some $C > 0$ instead, then $\text{Random}(p_1, \dots, p_n)$ is defined as $\text{Random}(p_1/C, \dots, p_n/C)$, where $(p_1/C, \dots, p_n/C)$ is the probability distribution. For example, for $(0.1, 0.2, 0.3)$ with $0.1 + 0.2 + 0.3 = 0.6$:

$$\text{Random}(0.1, 0.2, 0.3) = \text{Random}(0.1/0.6, 0.2/0.6, 0.3/0.6) = \text{Random}(1/6, 1/3, 1/2)$$

STOP and Think: Write a program computing $\text{Random}(p_1, \dots, p_n)$ that uses $\text{Random}(X)$ (for an appropriately chosen integer X) as a subroutine.

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?

36. Motif Finding Is More Difficult Than You Think

37. Scoring Motifs

38. From Motif Finding to Finding a Median String

39. Greedy Motif Search

40. Motif Finding Meets Oliver Cromwell

41. Randomized Motif Search

42. How Can a Randomized Algorithm Perform So Well?

43. Gibbs Sampling

44. Gibbs Sampling in Action



We have previously defined the notion of a *Profile*-most probable k -mer in a string. We now define a *Profile*-randomly generated k -mer in a string *Text*. For each k -mer *Pattern* in *Text*, compute $\Pr(\text{Pattern} \mid \text{Profile})$, resulting in $n = |\text{Text}| - k + 1$ probabilities (p_1, \dots, p_n) . These probabilities do not necessarily sum to 1, but we can still form a random number generator based on them. GIBBSAMPLER uses this random number generator to randomly select a *Profile*-randomly generated k -mer at each step. If the die rolls the number i , we define the *Profile*-randomly generated k -mer as the i -th k -mer in *Text*. While the pseudocode below repeats this procedure a fixed number of times (N), in practice GIBBSAMPLER depends on various stopping rules that are beyond the scope of this chapter.

```
GIBBSAMPLER(Dna, k, t, N)
  randomly select  $k$ -mers Motifs = (Motif1, ..., Motift) in each string from Dna
  BestMotifs ← Motifs
  for i from 1 to N
    i ← Random(t)
    construct profile matrix Profile from all strings in Motifs except for Motifi
    Motifi ← Profile-randomly generated  $k$ -mer in the  $i$ -th sequence
    if Score(Motifs) < Score(BestMotifs)
      BestMotifs ← Motifs
  output BestMotifs
```

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

- 35. Do We Have a "Clock" Gene?
- 36. Motif Finding Is More Difficult Than You Think
- 37. Scoring Motifs
- 38. From Motif Finding to Finding a Median String
- 39. Greedy Motif Search
- 40. Motif Finding Meets Oliver Cromwell
- 41. Randomized Motif Search
- 42. How Can a Randomized Algorithm Perform So Well?
- 43. Gibbs Sampling**
- 44. Gibbs Sampling in Action

Stepic



Full screen

CODE CHALLENGE: Implement `GIBBS_SAMPLER`.

Input: Integers k , t , and N , followed by a collection of strings Dna .

Output: The strings *BestMotifs* resulting from running `GIBBS_SAMPLER(Dna, k, t, N)` with 20 random starts.

Sample Input:

```
8 5 100
CGCCCTCTCGGGGTGTT CAGTAAACGGCCA
GGGCGAGGTATGTGTAAGTGCCAAGGTGCCAG
TAGTACCGAGACCGAAAGAAGTATACAGGCGT
TAGATCAAGTTTCAGGTGCACGTCGGTGAACC
AATCCACCAGCTCCACGTGCAATGTTGGCCTA
```

Sample Output:

```
TCTCGGGG
CCAAGGTG
TACAGGCG
TTCAGGTG
TCCACGTG
```

Note: As with `RANDOMIZEDMOTIFSEARCH`, there is a very small chance that your algorithm may be implemented correctly but not return the correct answer. We suggest running your algorithm on another dataset if you do not get the correct answer the first time.

Start Quiz (limit: 5 minutes)

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

- 35. [Do We Have a "Clock" Gene?](#)
- 36. [Motif Finding Is More Difficult Than You Think](#)
- 37. [Scoring Motifs](#)
- 38. [From Motif Finding to Finding a Median String](#)
- 39. [Greedy Motif Search](#)
- 40. [Motif Finding Meets Oliver Cromwell](#)
- 41. [Randomized Motif Search](#)
- 42. [How Can a Randomized Algorithm Perform So Well?](#)
- 43. Gibbs Sampling**
- 44. [Gibbs Sampling in Action](#)



We illustrate how GIBBSAMPLER works by using the same example as before. Imagine that, at the initial step, it has chosen the following 4-mers (shown in bold)

```

ttACCTtaac
gATGTctgtc
ccgGCGTtag
cactaACGAg
cgtcagAGGT

```

and that at the first iteration it randomly selected the 3rd string for removal:

```

ttACCTtaac
gATGTctgtc
.....
cactaACGAg
cgtcagAGGT

```

This results in the following motif, count, and profile matrices.

<i>Motifs</i>		t	a	a	c
		G	T	c	t
		a	c	t	a
		A	G	G	T
<i>Count</i>	A:	2	1	1	1
	C:	0	1	1	1
	G:	1	1	1	0
	T:	1	1	1	2
<i>Profile</i>	A:	2/4	1/4	1/4	1/4
	C:	0	1/4	1/4	1/4
	G:	1/4	1/4	1/4	0
	T:	1/4	1/4	1/4	2/4

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

- 35. Do We Have a "Clock" Gene?
- 36. Motif Finding Is More Difficult Than You Think
- 37. Scoring Motifs
- 38. From Motif Finding to Finding a Median String
- 39. Greedy Motif Search
- 40. Motif Finding Meets Oliver Cromwell
- 41. Randomized Motif Search
- 42. How Can a Randomized Algorithm Perform So Well?
- 43. Gibbs Sampling

44. Gibbs Sampling in Action



Note that the profile matrix (reproduced below) is only slightly more “conserved” than the uniform profile, making us wonder whether we have any chance to be steered toward the implanted motif.

<i>Profile</i>	A:	2/4	1/4	1/4	1/4
	C:	0	1/4	1/4	1/4
	G:	1/4	1/4	1/4	0
	T:	1/4	1/4	1/4	2/4

We now use this profile matrix to compute the probabilities of all 4-mers in the deleted string `ccgGCGTtag`:

ccgG	cgGC	gGCG	GCGT	CGTt	GTta	Ttag
0	0	0	1/128	0	0	0

This has brought us to a situation for which we don’t need to roll a die to select one of these 4-mers because all probabilities except for one are now 0. This situation is similar to the one we encountered with `GREEDYMOTIFSEARCH`, and as before, we need to substitute the zero probabilities with small pseudocounts to avoid disastrous results.

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

- 35. [Do We Have a "Clock" Gene?](#)
- 36. [Motif Finding Is More Difficult Than You Think](#)
- 37. [Scoring Motifs](#)
- 38. [From Motif Finding to Finding a Median String](#)
- 39. [Greedy Motif Search](#)
- 40. [Motif Finding Meets Oliver Cromwell](#)
- 41. [Randomized Motif Search](#)
- 42. [How Can a Randomized Algorithm Perform So Well?](#)
- 43. [Gibbs Sampling](#)

44. Gibbs Sampling in Action



Application of Laplace's Rule of Succession yields the following updated count and profile matrices:

Count	A:	3	2	2	2
	C:	1	2	2	2
	G:	2	2	2	1
	T:	2	2	2	3
Profile	A:	3/8	2/8	2/8	2/8
	C:	1/8	2/8	2/8	2/8
	G:	2/8	2/8	2/8	1/8
	T:	2/8	2/8	2/8	3/8

After adding pseudocounts, the probabilities of each 4-mer in the deleted string `ccgGCGTtag` are recomputed as follows:

ccgG	cgGC	gGCG	GCGT	CGTt	GTta	Ttag
$4/8^4$	$8/8^4$	$8/8^4$	$24/8^4$	$12/8^4$	$16/8^4$	$16/8^4$

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

- 35. Do We Have a "Clock" Gene?
- 36. Motif Finding Is More Difficult Than You Think
- 37. Scoring Motifs
- 38. From Motif Finding to Finding a Median String
- 39. Greedy Motif Search
- 40. Motif Finding Meets Oliver Cromwell
- 41. Randomized Motif Search
- 42. How Can a Randomized Algorithm Perform So Well?
- 43. Gibbs Sampling
- 44. Gibbs Sampling in Action**



Let's assume that after rolling the hypothetical 7-sided die reflecting these probabilities, we have arrived at the *Profile*-randomly generated 4-mer **GCGT** (the fourth 4-mer in the deleted sequence). The deleted string **ccgGCGTtag** is now added back to the collection of motifs, and **GCGT** substitutes the previously chosen **ccgG** in the 3rd sequence:

```
ttACCTaac
gATGTctgtc
ccgGCGTtag
cactaACGA
cgtagAGGT
```

We again roll a fair 5-sided die and delete a randomly chosen string (let's say the first string):

```
.....
gATGTctgtc
ccgGCGTtag
cactaACGA
cgtagAGGT
```

After constructing the motif and profile matrices, we obtain the following:

		G	T	c	t
<i>Motifs</i>		G	C	G	T
		a	c	t	a
		A	G	G	T
<i>Profile</i>	A:	2/4	0	0	1/4
	C:	0	2/4	1/4	0
	G:	2/4	1/4	2/4	0
	T:	0	1/4	1/4	3/4

Note that the profile matrix looks more biased toward the implanted motif than the previous profile matrix did.

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?
36. Motif Finding Is More Difficult Than You Think
37. Scoring Motifs
38. From Motif Finding to Finding a Median String
39. Greedy Motif Search
40. Motif Finding Meets Oliver Cromwell
41. Randomized Motif Search
42. How Can a Randomized Algorithm Perform So Well?
43. Gibbs Sampling
- 44. Gibbs Sampling in Action**



We update the count and profile matrices with pseudocounts:

Count	A:	3	1	1	2
	C:	1	3	2	1
	G:	3	2	3	1
	T:	1	2	2	4
Profile	A:	3/8	1/8	1/8	2/8
	C:	1/8	3/8	2/8	1/8
	G:	3/8	2/8	3/8	1/8
	T:	1/8	2/8	2/8	4/8

We compute the probabilities of all 4-mers in the deleted string `ttACCTaac`:

ttAC	tACC	ACCT	CCTt	CTta	Ttaa	taac
$2/8^4$	$2/8^4$	$72/8^4$	$24/8^4$	$8/8^4$	$4/8^4$	$1/8^4$

We then roll a 7-sided die to arrive at the *Profile*-randomly generated *k*-mer `ACCT`, which we add to the collection *Motifs* (shown in bold):

```

ttACCTtaac
gATGTctgtc
ccgGCGTtag
cactaACGAg
cgtcagAGGT

```

Bioinformatics Algorithms

Chapter 1: Where Does DNA

Replication Begin?: ...

Chapter 2: How Do We Sequence

Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

- 35. Do We Have a "Clock" Gene?
- 36. Motif Finding Is More Difficult Than You Think
- 37. Scoring Motifs
- 38. From Motif Finding to Finding a Median String
- 39. Greedy Motif Search
- 40. Motif Finding Meets Oliver Cromwell
- 41. Randomized Motif Search
- 42. How Can a Randomized Algorithm Perform So Well?
- 43. Gibbs Sampling

44. Gibbs Sampling in Action



After rolling the die once again, we remove the 4th sequence, add pseudocounts, and construct the resulting count and profile matrices:

<i>Dna</i>	ttACCTtaac gATGTctgtc ccgGCGTtag cgtcagAGGT			
<i>Motifs</i>	A	C	C	T
	G	T	c	t
	G	C	G	T
	A	G	G	T
<i>Count</i>	A: 3	1	1	1
	C: 1	3	3	1
	G: 3	2	3	1
	T: 1	2	1	5
<i>Profile</i>	A: 3/8	1/8	1/8	1/8
	C: 1/8	3/8	3/8	1/8
	G: 3/8	2/8	3/8	1/8
	T: 1/8	2/8	1/8	5/8

Bioinformatics Algorithms

Chapter 1: Where Does DNA

Replication Begin?: ...

Chapter 2: How Do We Sequence

Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

- 35. Do We Have a "Clock" Gene?
- 36. Motif Finding Is More Difficult Than You Think
- 37. Scoring Motifs
- 38. From Motif Finding to Finding a Median String
- 39. Greedy Motif Search
- 40. Motif Finding Meets Oliver Cromwell
- 41. Randomized Motif Search
- 42. How Can a Randomized Algorithm Perform So Well?
- 43. Gibbs Sampling

44. Gibbs Sampling in Action

Stepic



Full screen

We now compute the probabilities of all 4-mers in the deleted string cactaACGA_g:

cact	acta	ctaA	taAC	aACG	ACGA	CGA _g
$15/8^4$	$9/8^4$	$2/8^4$	$1/8^4$	$9/8^4$	$27/8^4$	$2/8^4$

We need to roll a 7-sided die to produce a *Profile*-randomly generated 4-mer. Assuming the most probable scenario is that we select ACGA, we update the selected 4-mers:

```

ttACCTaac
gATGTctgtc
ccgCGTtag
cactaACGAg
cgtcagAGGT

```

You can see that the algorithm is beginning to converge; rest assured that a subsequent iteration will produce all implanted motifs as soon as we select the 2nd sequence (when the incorrect 4-mer GTct will likely change into ATGT).

STOP and Think: Run GIBBSAMPLER on the Subtle Motif Problem. What do you find?

Share

< Back

Next step >

Discussions

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

- 35. Do We Have a "Clock" Gene?
- 36. Motif Finding Is More Difficult Than You Think
- 37. Scoring Motifs
- 38. From Motif Finding to Finding a Median String
- 39. Greedy Motif Search
- 40. Motif Finding Meets Oliver Cromwell
- 41. Randomized Motif Search
- 42. How Can a Randomized Algorithm Perform So Well?
- 43. Gibbs Sampling

44. Gibbs Sampling in Action



When we run `GIBBSAMPLER` 2000 times on the Subtle Motif Problem with implanted 15-mer `AAAAAAAAAGGGGGGG` (each time with new randomly selected k -mers for $N = 200$ iterations), it returns a collection *Motifs* with consensus string `AAAAAAGAGGGGGT` and $\text{Score}(\text{Motifs})$ equal to 38. This score is even lower than the score of 40 obtained by the implanted motifs!

Although `GIBBSAMPLER` performs well in many cases, it may converge to a suboptimal solution, particularly for difficult search problems with elusive motifs. A **local optimum** is a solution that is optimal within a small neighboring set of solutions, which is in contrast to a **global optimum**, or the optimal solution among all possible solutions. Since `GIBBSAMPLER` explores just a small subset of solutions, it may “get stuck” in a local optimum. For this reason, similarly to `RANDOMIZEDMOTIFSEARCH`, it should be run many times with the hope that one of these runs will produce the best-scoring motifs.

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

- 35. Do We Have a "Clock" Gene?
- 36. Motif Finding Is More Difficult Than You Think
- 37. Scoring Motifs
- 38. From Motif Finding to Finding a Median String
- 39. Greedy Motif Search
- 40. Motif Finding Meets Oliver Cromwell
- 41. Randomized Motif Search
- 42. How Can a Randomized Algorithm Perform So Well?
- 43. Gibbs Sampling

44. Gibbs Sampling in Action

Motif finding becomes difficult if the **background nucleotide distribution** in the sample is skewed. In this case, searching for k -mers with the minimum score or entropy may lead to a biologically irrelevant motif composed from the most frequent nucleotides in the sample. For example, if A has frequency 80% and T, G, and C have frequencies of 5%, then k -mer AA...AA may represent a motif with minimum score/entropy, thus disguising biologically relevant motifs. For example, the relevant motif **GCCG** with score 5 in the example below loses to the 4-mer **aaaa** with score 1.

```

taaaaGtCGa
acGcTGaaaa
aaaaGCCtat
acCCGaataa
agaaaaGgCG

```

To find biologically relevant motifs in biased samples, you may want to use a generalization of entropy called **relative entropy** to highlight **GCCG** among the k -mers like **aaaa** composed from frequent nucleotides. Given a collection of strings Dna , the relative entropy of a $4 \times k$ profile matrix $P = (p_{r,j})$ is defined as

$$\sum_{j=1}^k \sum_{r \in \{A,C,G,T\}} p_{r,j} \cdot \log_2(p_{r,j}/b_r) = \sum_{j=1}^k \sum_{r \in \{A,C,G,T\}} p_{r,j} \cdot \log_2(p_{r,j}) - \sum_{j=1}^k \sum_{r \in \{A,C,G,T\}} p_{r,j} \cdot \log_2(b_r)$$

where b_r is the frequency of nucleotide r in Dna . Note that the sum in the entropy equation is preceded by a negative sign $(-\sum_{j=1}^k \sum_{r \in \{A,C,G,T\}} p_{r,j} \cdot \log_2(p_{r,j}))$ whereas the sum in the relative entropy equation is preceded by a positive sign $(+\sum_{j=1}^k \sum_{r \in \{A,C,G,T\}} p_{r,j} \cdot \log_2(p_{r,j}/b_r))$. Therefore, although we minimized the entropy of a motif matrix, we will now attempt to maximize the relative

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

35. Do We Have a "Clock" Gene?
36. Motif Finding Is More Difficult Than You Think
37. Scoring Motifs
38. From Motif Finding to Finding a Median String
39. Greedy Motif Search
40. Motif Finding Meets Oliver Cromwell
41. Randomized Motif Search
42. How Can a Randomized Algorithm Perform So Well?
43. Gibbs Sampling
44. Gibbs Sampling in Action
- 45. Complications in Motif Finding**

$$\sum_{j=1}^k \sum_{r \in \{A,C,G,T\}} p_{r,j} \cdot \log_2(p_{r,j}/b_r) =$$

$$\sum_{j=1}^k \sum_{r \in \{A,C,G,T\}} p_{r,j} \cdot \log_2(p_{r,j}) - \sum_{j=1}^k \sum_{r \in \{A,C,G,T\}} p_{r,j} \cdot \log_2(b_r)$$

In the relative entropy formula (reproduced above), the term $-\sum_{j=1}^k \sum_{r \in \{A,C,G,T\}} p_{r,j} \cdot \log_2(b_r)$ is called the **cross-entropy** of the profile matrix P ; note that the relative entropy of a profile matrix is simply the difference between the profile's cross-entropy and its entropy. For example, the relative entropy for the motif **GCCG** in the example above is equal to $10.12 - 3.53 = 6.59$, as shown below.

	G	t	C	G
	G	C	t	G
	G	C	C	t
	C	C	C	G
	G	g	C	G
A:	0.0	0.0	0.0	0.0
C:	0.2	0.6	0.8	0.0
G:	0.8	0.2	0.0	0.8
T:	0.0	0.2	0.2	0.2
entropy:	0.72 + 1.37 + 0.72 + 0.72 = 3.53			
cross-entropy:	2.47 + 2.59 + 2.47 + 2.59 = 10.12			

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

- 35. Do We Have a "Clock" Gene?
- 36. Motif Finding Is More Difficult Than You Think
- 37. Scoring Motifs
- 38. From Motif Finding to Finding a Median String
- 39. Greedy Motif Search
- 40. Motif Finding Meets Oliver Cromwell
- 41. Randomized Motif Search
- 42. How Can a Randomized Algorithm Perform So Well?
- 43. Gibbs Sampling
- 44. Gibbs Sampling in Action
- 45. Complications in Motif Finding**

For the more conserved but irrelevant motif `aaaa`, the relative entropy is equal to $4.18 - 0.72 = 3.46$ ($b_A = 0.52$, $b_C = 0.18$, $b_G = 0.18$, $b_T = 0.12$), as shown below. Thus, `GCCG` loses to `aaaa` with respect to entropy but wins with respect to relative entropy.

	a	a	a	a	
	a	a	a	a	
	a	a	a	a	
	a	t	a	a	
	a	a	a	a	
A:	1.0	0.8	1.0	1.0	
C:	0.0	0.0	0.0	0.0	
G:	0.0	0.0	0.0	0.0	
T:	0.0	0.2	0.0	0.0	
entropy:	0.0	+ 0.72	+ 0.0	+ 0.0	= 0.72
cross-entropy:	0.94	+ 1.36	+ 0.94	+ 0.94	= 4.18

Another complication in motif finding is that many motifs are best represented in a different alphabet than the alphabet of 4 nucleotides. Let `W` denote either `A` or `T`, `S` denote either `G` or `C`, `K` denote either `G` or `T`, and `Y` denote either `C` or `T`. Now, consider the motif `CSKWYWWATKWATYYK`, which represents the CSRE motif in yeast. This strong motif in a hybrid alphabet corresponds to 2^{11} different motifs in the standard 4-letter alphabet of nucleotides. However, each of these 2^{11} motifs is too weak to be found by algorithms we have considered in this chapter.

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

- 35. Do We Have a "Clock" Gene?
- 36. Motif Finding Is More Difficult Than You Think
- 37. Scoring Motifs
- 38. From Motif Finding to Finding a Median String
- 39. Greedy Motif Search
- 40. Motif Finding Meets Oliver Cromwell
- 41. Randomized Motif Search
- 42. How Can a Randomized Algorithm Perform So Well?
- 43. Gibbs Sampling
- 44. Gibbs Sampling in Action
- 45. Complications in Motif Finding**

Tuberculosis Hibernate to Hide from Antibiotics?

Stepic



Full screen

Tuberculosis (TB) is an infectious disease caused by the *Mycobacterium tuberculosis* bacterium (MTB) and is responsible for over a million deaths each year, mainly in low-income countries. Although the spread of TB has been greatly reduced due to antibiotics, strains that resist all available treatments are now emerging. MTB is successful as a pathogen because it can persist in humans for decades without causing disease: one-third of the world population has **latent MTB infections**, in which MTB lies dormant within the host's body and may or may not reactivate at a later time. The widespread prevalence of latent infections makes it difficult to control TB epidemics. Biologists are therefore interested in finding out what makes the disease latent and how MTB activates itself within a host.

It remains unclear why MTB can stay latent for so long and how it survives during latency. The resistance of latent TB to antibiotics implies that MTB may have an ability to shut down expression of most genes and stay dormant, not unlike bears hibernating in the winter. Hibernation in bacteria is called **sporulation** because many bacteria form protective and metabolically dormant **spores** that can survive in tough conditions (e.g., heat shock or oxygen shortage), allowing the bacteria to persist in the environment until conditions improve.

Hypoxia, or oxygen shortage, is often associated with latent forms of TB. Biologists have found that the tuberculosis bacterium becomes dormant in low-oxygen environments, presumably with the idea that the host's lungs will recover enough to potentially spread the disease in the future. Since MTB shows a remarkable ability to survive for years without oxygen, it is important to identify MTB genes responsible for the development of the latent state under hypoxic conditions. Biologists are interested to find a **master regulator** (transcription factor) that "senses" the shortage of oxygen and starts a genetic program that affects the expression of many genes, allowing MTB to adapt to hypoxia.

Bioinformatics Algorithms

Chapter 1: Where Does DNA

Replication Begin?: ...

Chapter 2: How Do We Sequence

Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

- 35. Do We Have a "Clock" Gene?
- 36. Motif Finding Is More Difficult Than You Think
- 37. Scoring Motifs
- 38. From Motif Finding to Finding a Median String
- 39. Greedy Motif Search
- 40. Motif Finding Meets Oliver Cromwell
- 41. Randomized Motif Search
- 42. How Can a Randomized Algorithm Perform So Well?

Tuberculosis Hibernate to Hide from Antibiotics?

Stepic



Full screen

A decade ago, biologists found the dormancy survival regulator (DosR), a transcription factor that regulates many genes whose expression dramatically changes under hypoxic conditions. However, it remained unclear how DosR regulates these genes, and its transcription factor binding site remained unknown. In an attempt to resolve this puzzle, biologists performed a DNA array experiment and found 25 genes whose expression levels significantly change in hypoxic conditions. Given the upstream regions of these genes (each 250 nucleotides long), we would like to discover the "hidden message" that DosR uses to control the expression of these genes.

To simplify the problem a bit, we have selected just 10 of the 25 genes, resulting in a **DosR dataset**. We will try to identify motifs in this dataset using the arsenal of motif finding tools that we have developed. But we will not give you a hint about the DosR motif or its length.

[Download DosR Dataset](#)

What k -mer size should we choose to analyze the DosR dataset using **MEDIANSTRING** and **RANDOMIZEDMOTIFSEARCH**? Taking a wild guess and running these algorithms for k from 8 to 12 returns the consensus strings shown below.

MEDIANSTRING			RANDOMIZEDMOTIFSEARCH		
k	consensus	score	k	consensus	score
8	CATCGGCC	11	8	CCGACGGG	13
9	GGCGGGAC	16	9	CCATCGGCC	16
10	GGTGGCCACC	19	10	CCATCGGCC	21
11	GGACTTCCGGC	20	11	ACCTTCGGCCC	25
12	GGACTTCCGGCC	23	12	GGACCAACGGCC	28

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

- 35. Do We Have a "Clock" Gene?
- 36. Motif Finding Is More Difficult Than You Think
- 37. Scoring Motifs
- 38. From Motif Finding to Finding a Median String
- 39. Greedy Motif Search
- 40. Motif Finding Meets Oliver Cromwell
- 41. Randomized Motif Search
- 42. How Can a Randomized Algorithm Perform So Well?

Tuberculosis Hibernate to Hide from Antibiotics?

Stepic



Full screen

Note that although the consensus strings returned by **RANDOMIZEDMOTIFSEARCH** generally deviate from the median strings, the consensus string of length 12 (GGACCAACGGCC, with score 28) is very similar to the median string (GGACTTCCGGCC, with score 23).

While the motifs returned by **RANDOMIZEDMOTIFSEARCH** are slightly less conserved than the motifs returned by **MEDIANSTRING**, the former algorithm has the advantage of being able to find longer motifs (since **MEDIANSTRING** becomes too slow for longer motifs). Here is the motif of length 20 returned by **RANDOMIZEDMOTIFSEARCH**:

CGGGACCTACGTCCTAGCC (score: 57)

As you can see, the consensus strings of length 12 found by **RANDOMIZEDMOTIFSEARCH** and **MEDIANSTRING** are “embedded” (with small variations) in the longer motif of length 20:

```
GGACCAACGGCC
CGGGACCTACGTCCTAGCC
GGACTTCCGGCC
```

Finally, in 2000 runs with $N = 200$, **GIBBSAMPLER** returned the same consensus string for DosR dataset as **RANDOMIZEDMOTIFSEARCH**, but generated a different collection of motifs with a smaller score of 55.

As you have seen in this chapter, different motif finding algorithms generate somewhat different results, and the question of what the DosR motif in *Mycobacterium tuberculosis* looks like remains unclear. Try to answer this question and find all putative DosR motifs in *Mycobacterium tuberculosis* as well as all genes that they regulate. We will provide you with the upstream regions of all 25 genes identified in the DosR study by [Park et al., 2003](#) to help you address the following problem:

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

- 35. Do We Have a "Clock" Gene?
- 36. Motif Finding Is More Difficult Than You Think
- 37. Scoring Motifs
- 38. From Motif Finding to Finding a Median String
- 39. Greedy Motif Search
- 40. Motif Finding Meets Oliver Cromwell
- 41. Randomized Motif Search
- 42. How Can a Randomized Algorithm Perform So Well?

Genes encode proteins, and proteins dictate cell function. To respond to changes in their environment, cells must therefore control their protein levels. The flow of information from DNA to RNA to protein means that the cell can adjust the amount of proteins that it produces during both transcription (DNA to RNA) and translation (RNA to protein).

Transcription begins when an RNA polymerase binds to a **promoter sequence** on the DNA molecule, which is often located just upstream from the starting point for transcription. The initiation of transcription is a convenient control point for the cell to regulate gene expression since it is at the very beginning of the protein production process. The genes transcribed in a cell are being controlled by various transcription regulators that may increase or suppress transcription.

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

- 35. [Do We Have a "Clock" Gene?](#)
- 36. [Motif Finding Is More Difficult Than You Think](#)
- 37. [Scoring Motifs](#)
- 38. [From Motif Finding to Finding a Median String](#)
- 39. [Greedy Motif Search](#)
- 40. [Motif Finding Meets Oliver Cromwell](#)
- 41. [Randomized Motif Search](#)
- 42. [How Can a Randomized Algorithm Perform So Well?](#)
- 43. [Gibbs Sampling](#)
- 44. [Gibbs Sampling in Action](#)
- 45. [Complications in Motif Finding](#)



Stepic

Full screen

A **DNA array** (also known as a **microarray**) is a collection of DNA molecules attached to a solid surface. Each spot on the microarray is assigned a unique DNA sequence called a **probe** that measures the expression level of a specific gene, known as a **target**. In most arrays, probes are synthesized and then attached to a glass or silicon chip (figure below). Fluorescently labeled targets then bind to their corresponding probe (e.g., when their sequences are complementary), generating a fluorescent signal. The strength of this signal depends upon the amount of target sample that binds to the probe at a given spot. Thus, the higher the expression level of a gene, the higher the intensity of its fluorescent signal on the array. Since an array may contain thousands of probes (modern microarrays may contain over a million probes), biologists can measure the expression of thousands of genes in a single array experiment. For example, the microarray experiment that identified the evening element in *Arabidopsis thaliana* measured the expression of 8,000 genes.

Bioinformatics Algorithms

Chapter 1: Where Does DNA Replication Begin?: ...

Chapter 2: How Do We Sequence Antibiotics?: ...

Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:

- 35. Do We Have a "Clock" Gene?
- 36. Motif Finding Is More Difficult Than You Think
- 37. Scoring Motifs
- 38. From Motif Finding to Finding a Median String
- 39. Greedy Motif Search
- 40. Motif Finding Meets Oliver Cromwell
- 41. Randomized Motif Search

[Stepic](#)[Full screen](#)

In his landmark work *Histoire naturelle*, Comte de Buffon questioned the usefulness of mathematics and outlined a history of the Earth having little relation to the Biblical account. It was translated into many languages, making Buffon one of the most widely read French authors of the 18th century. Despite his doubts about the usefulness of mathematics, Buffon is now viewed as the father of randomized algorithms.

He answered the following question:

Suppose we have a floor made of parallel strips of wood, each the same width d , and we drop a needle of length c onto the floor. What is the probability that the needle will lie across a line between two strips?

Buffon's needle led to the earliest Monte Carlo algorithm for approximating the number π .

Bioinformatics Algorithms

[Chapter 1: Where Does DNA Replication Begin?: ...](#)

[Chapter 2: How Do We Sequence Antibiotics?: ...](#)

[Chapter 3: Which DNA Patterns Play the Role of Molecular Clocks?:](#)

- [35. Do We Have a "Clock" Gene?](#)
- [36. Motif Finding Is More Difficult Than You Think](#)
- [37. Scoring Motifs](#)
- [38. From Motif Finding to Finding a Median String](#)
- [39. Greedy Motif Search](#)
- [40. Motif Finding Meets Oliver Cromwell](#)
- [41. Randomized Motif Search](#)