Introduction to Bioinformatics

BY

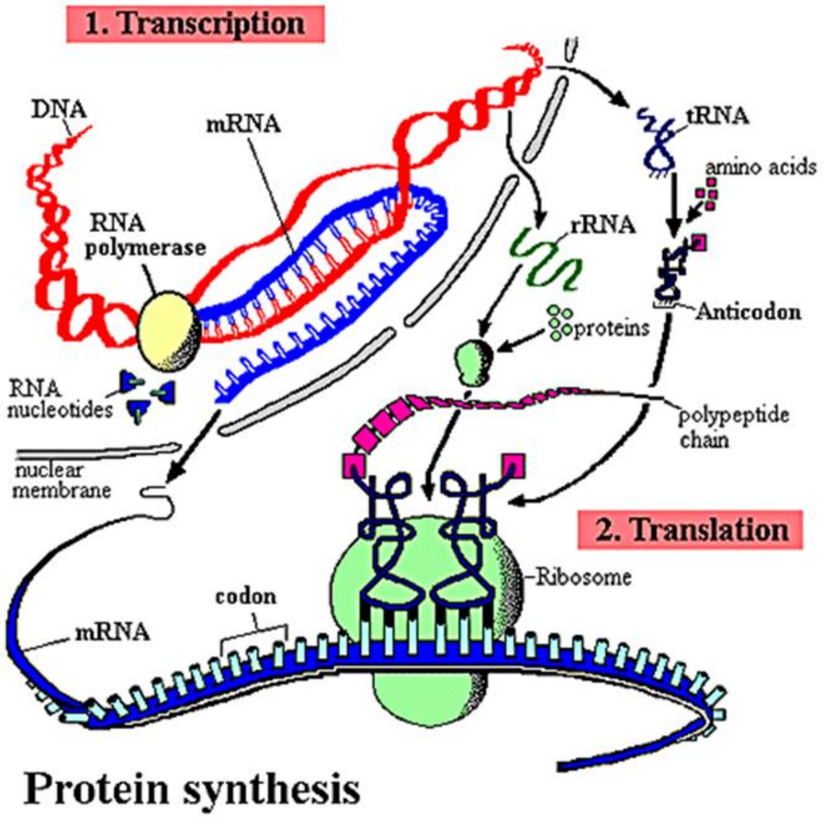
DR. MOHAMMAD HASHIM

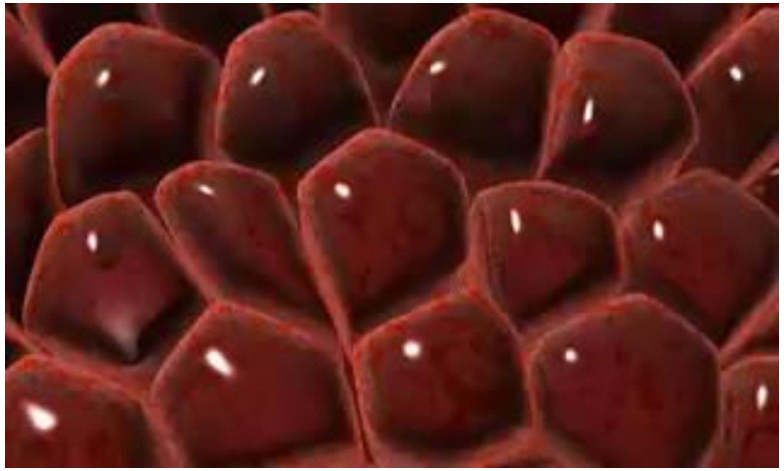
Lect. 2

Our First Bioinformatics Problem

Genome Replication Problem

Gene Expression





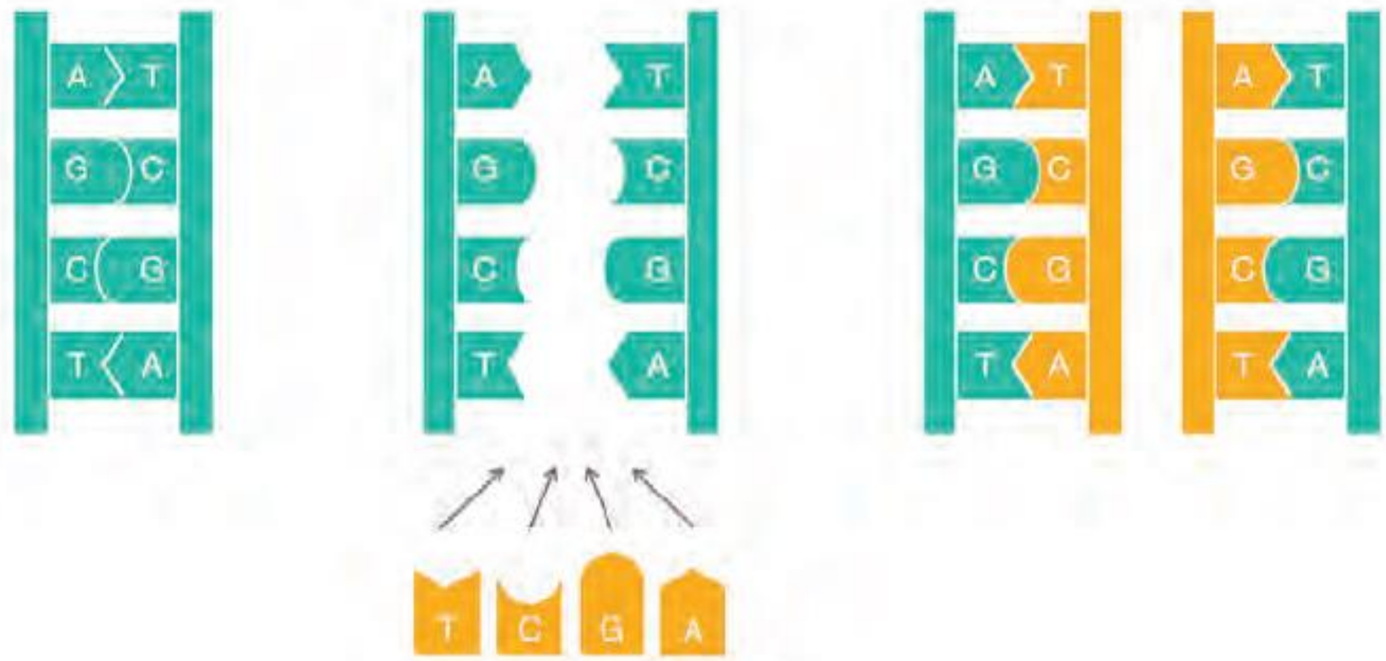
Genome Replication Problem

"It has not escaped our notice that the specific pairing we have postulated immediately suggests a possible copying mechanism for the genetic material."

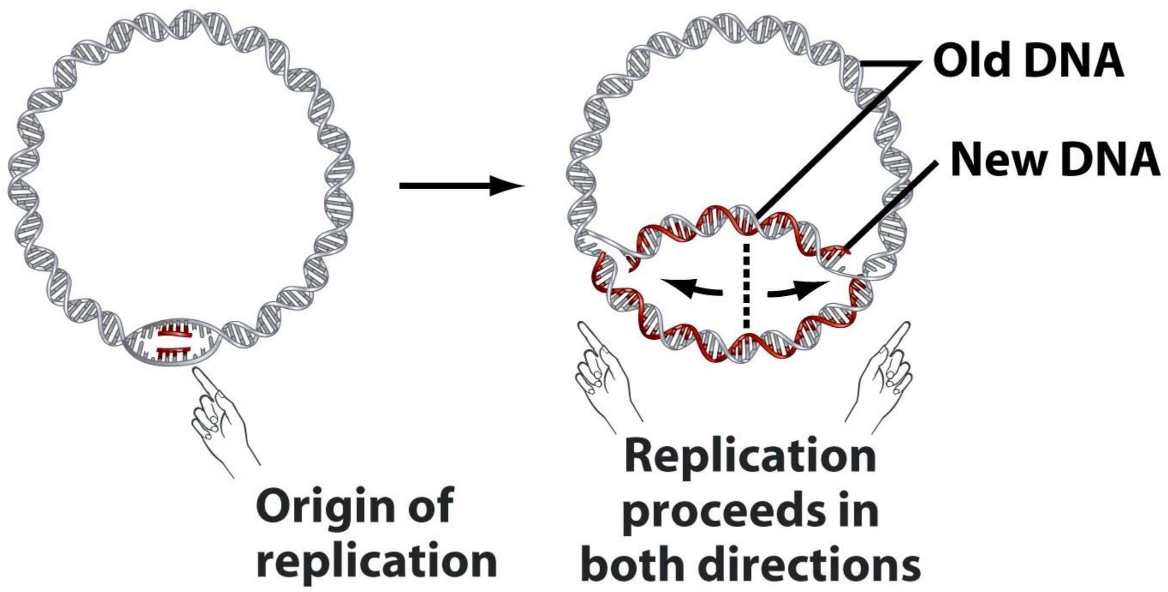
James Watson and Francis Crick, 1953

- **Genome Replication** is one of the most important tasks carried out in the cell.
- Before a cell can divide, it must first replicate its genome so that each of the two daughter cells inherits its own copy.
- The two strands of the parent DNA molecule unwind during replication, and then each parent strand acts as a template for the synthesis of a new strand.
- As a result, the replication process begins with a pair of complementary strands of DNA and ends with two pairs of complementary strands.

Genome replication



But Where in a genome does it all begin?



Replication begins in a region called the **replication origin** (***oriC***)

Search for Hidden Messages in Replication Origin

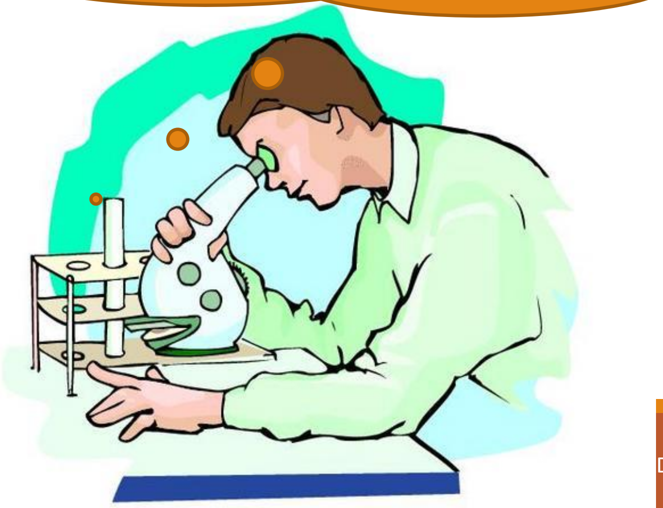
- **What is a Hidden Message in Replication Origin?**
- Some Hidden Messages are More Surprising than Others

Finding Origin of Replication

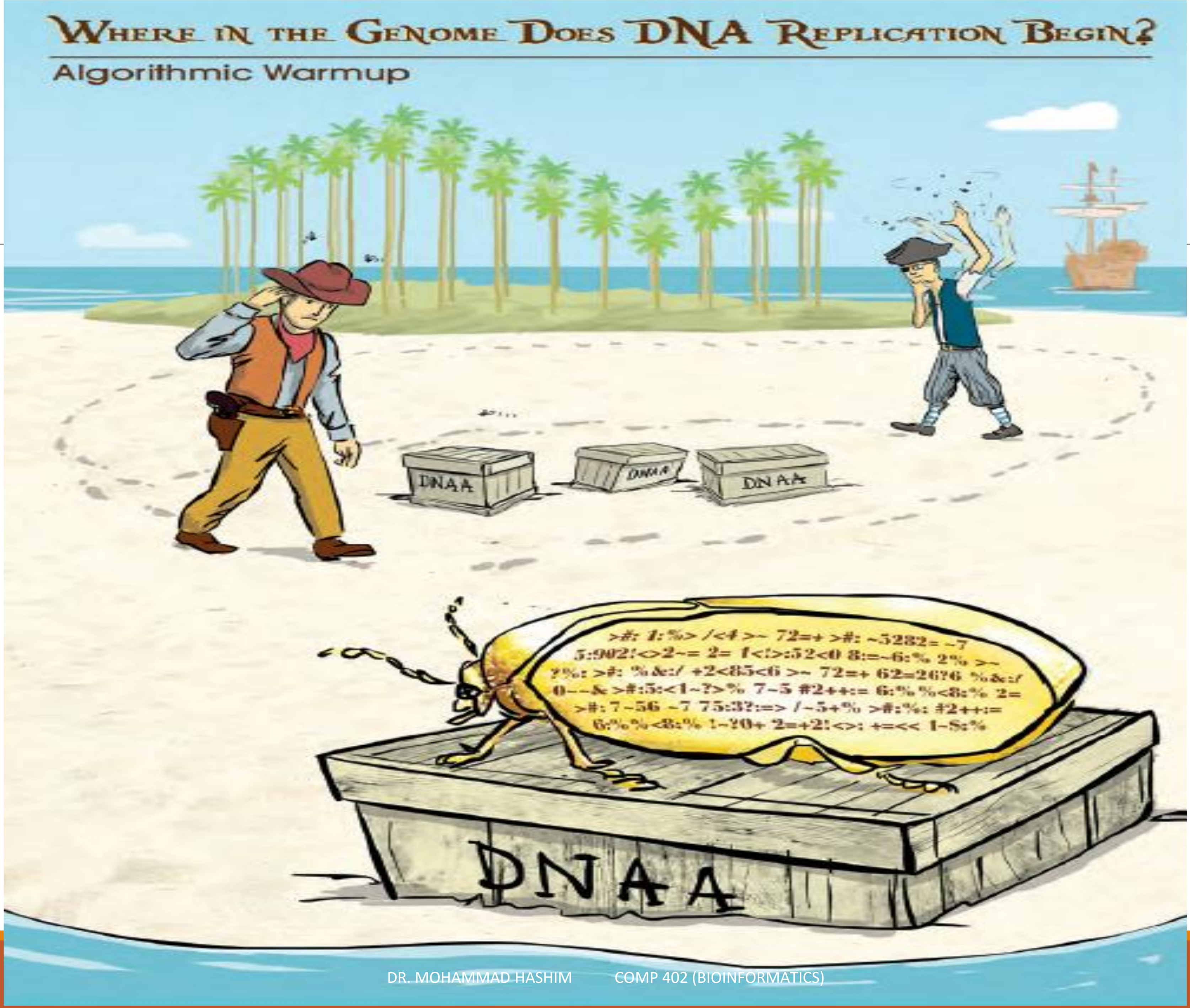
Finding *oriC* Problem: Finding *oriC* in a genome.

- **Input.** A genome (a string of nucleotides from alphabet {A, C, G, T}).
- **Output.** The location of *oriC* in the genome.

OK – let's cut out this DNA fragment.
Can the genome replicate without it?



This is not a computational problem!


Where in the Genome Does DNA Replication Begin?
Algorithmic Warmup
DNAA
DNAA
DNAA
DNAA

How Does the Cell Know to Begin Replication in Short *oriC*?

Replication origin of *Vibrio cholerae* (≈500 nucleotides):

```
atcaatgatcaacgtaagcttctaagcatgatcaaggtgctcacacagtttatccacaac
ctgagtggatgacatcaagataggtcgttgtatctccttcctctcgtactctcatgacca
cggaaagatgatcaagagaggatgatttcttggccatatcgcaatgaatacttgtgactt
gtgcttccaattgacatcttcagcgccatattgcgctggccaaggtgacggagcgggatt
acgaaagcatgatcatggctgttgttctgtttatcttgttttgactgagacttgttagga
tagacggtttttcatcactgactagccaaagccttactctgcctgacatcgaccgtaaat
tgataatgaatttacatgcttccgcgacgatttacctcttgatcatcgatccgattgaag
atcttcaattgttaattctcttgcctcgactcatagccatgatgagctcttgatcatgtt
tccttaaccctctattttttacggaagaatgatcaagctgctgctcttgatcatcgtttc
```

There must be a **hidden message** telling the cell to start replication here.

The Hidden Message Problem

Hidden Message Problem. Finding a hidden message in a string.

- **Input.** A string *Text* (representing replication origin).
- **Output.** A hidden message in *Text*.

This is not a computational problem either!

The notion of "**hidden message**" is not precisely defined.

“The Gold-Bug” Problem


```
53++!305))6*;4826)4+.)4+);806*;48!8`60))85;]
8*:+*8!83(88)5*!;46(;88*96*?;8)*+(;485);5*!2
:*+(;4956*2(5*4)8`8*;4069285);)6!8)4++;1(+9;
48081;8:8+1;48!85;4)485!528806*81(+9;48;(88;
4(+?34;48)4+;161;:188;+?;
```

A secret message left by pirates

(“The Gold-Bug” by Edgar Allan Poe)

Why is “;48” so Frequent?

Hint: The message is in English

```
53++!305))6*;4826)4+.)4+);806*;48!8`60))85;]
8*:+*8!83(88)5*!46(88*96*?;8)*+(;485);5*!2:*
+(;4956*2(5*4)8`8*;4069285););)6!8)4++;1(+9;48
081;8:8+1;48!85;4)485!528806*81(+9;48;(88;4(
+?34;48)4+;161;:188;+?;
```

“THE” is the Most Frequent English Word

```
53++!305))6*THE26)4+.)4+)806*THE!8`60))85;]8
*:+*8!83(88)5*!;46(;88*96*?;8)*+(THE5);5*!2:
*+(;4956*2(5*4)8`8*;4069285);)6!8)4++;1(+9TH
E081;8:8+1THE!85;4)485!528806*81(+9THE;(88;4
(+?34THE)4+;161;:188;+?;
```

Could you Complete Decoding the Message?

53++!305))6*THE26)H+.)H+)806*THE
!E`60))E5;]E*:+*E!E3(EE)5*!TH6(T
EE*96*?;E)*+(THE5)T5*!2:*+(TH956
*2(5*H)E`E*TH0692E5)T)6!E)H++T1(
+9THE0E1TE:E+1THE!E5T4)HE5!52880
6*E1(+9THET(EETH(+?34THE)H+T161T
:1EET+?T

The Hidden Message Problem Revisited

Hidden Message Problem. Finding a hidden message in a string.

- **Input.** A string *Text* (representing *oriC*).
- **Output.** A hidden message in *Text*.

The notion of "**hidden message**" is not precisely defined.

Hint: For various biological signals, certain words appear surprisingly frequently in small regions of the genome.
AATTT is a surprisingly frequent 5-mer in:

ACAAATTTGCATAATTTCGGGAAATTTCCT

The Frequent Words Problem

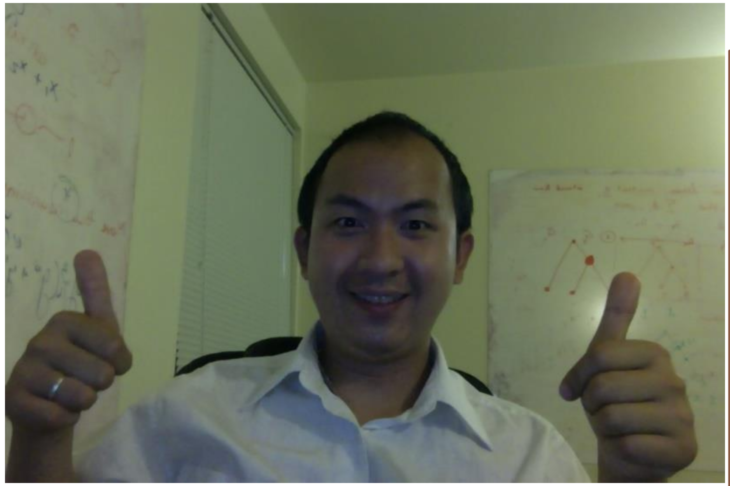
Frequent Words Problem. Finding most frequent *k*-mers in a string.

- **Input.** A string *Text* and an integer *k*.
- **Output.** All **most frequent *k*-mers** in *Text*.

This is better, but where is the definition of “a most frequent *k*-mer?”

The Frequent Words Problem

Frequent Words Problem. Finding most frequent *k*-mers in a string.
- **Input.** A string *Text* and an integer *k*.
- **Output.** All **most frequent *k*-mers** in *Text*.



A *k*-mer ***Pattern*** is a **most frequent *k*-mer** in a text if no other *k*-mer is more frequent than *Pattern*.

AATTT is a most frequent 5-mer in:
ACA**AATTT**GCAT**AATTT**CGGGA**AATTT**CCT

Son Pham, Ph.D., kindly gave us permission to use his photographs and greatly helped with preparing this presentation. **Thank you Son!**

Does the Frequent Words Problem Make Sense to Biologists?

Frequent Words Problem. Finding most frequent *k*-mers in a string.
- **Input.** A string *Text* and an integer *k*.
- **Output.** All **most frequent *k*-mers** in *Text*.

Replication is performed by **DNA polymerase** and the initiation of replication is mediated by a protein called ***DnaA***.

DnaA binds to short (typically 9 nucleotides long) segments within the replication origin known as a ***DnaA* box**.

A *DnaA* box is a hidden message telling *DnaA*: "**bind here!**" And *DnaA* wants to see multiple *DnaA* boxes.

Outline

Search for Hidden Messages in Replication Origin

- What is a Hidden Message in Replication Origin?
- **Some Hidden Messages are More Surprising than Others**

oriC of *Vibrio cholerae*

```
atcaatgatcaacgtaagcttctaagcatgatcaaggtgctcacacagtttatccacaacct
gagtggatgacatcaagataggtcgttgtatctccttcctctcgtactctcatgaccacgga
aagatgatcaagagaggatgatttcttggccatatcgcaatgaatacttgtgacttgtgctt
ccaattgacatcttcagcgccatattgcgctggccaaggtgacggagcgggattacgaaagc
atgatcatggctgttgttctgtttatcttgttttgactgagacttgttaggatagacggttt
ttcatcactgactagccaaagccttactctgcctgacatcgaccgtaaattgataatgaatt
tacatgcttccgcgacgatttacctcttgatcatcgatccgattgaagatcttcaattgtta
attctcttgcctcgactcatagccatgatgagctcttgatcatgtttccttaaccctctatt
ttttacggaagaatgatcaagctgctgctcttgatcatcgtttc
```

Too Many Frequent Words – Which One is a Hidden Message?

atcaatgatcaacgtaagcttctaagc**ATGATCAAG**gtgctcacacagtttatccacaacctgagtggatgacatcaagatag
gtcgttgtatctccttcctctcgtactctcatgaccacggaaag**ATGATCAAG**agaggatgatttcttggccatatcgcaatg
aatacttgtgacttgtgcttccaattgacatcttcagcgccatattgcgctggccaaggtgacggagcgggattacgaaagca
tgatcatggctgttgttctgtttatcttgttttgactgagacttgttaggatagacggtttttcatcactgactagccaaagc
cttactctgcctgacatcgaccgtaaattgataatgaatttacatgcttccgcgacgatttacct**CTTGATCAT**cgatccgat
tgaagatcttcaattgttaattctcttgcctcgactcatagccatgatgagct**CTTGATCAT**gtttccttaaccctctatttt
ttacggaaga**ATGATCAAG**ctgctgct**CTTGATCAT**cgtttc

Most frequent 9-mers in this *oriC* (all appear 3 times):
ATGATCAAG, **CTTGATCAT**, **TCTTGGATCA**, **CTCTTGATC**

Is it **STATISTICALLY** surprising to find a 9-mer appearing **3 or more** times within ≈ 500 nucleotides?

Hidden Message Found!

atcaatgatcaacgtaagcttctaagc**ATGATCAAG**gtgctcacacagtttatccacaacctgagtggatgacatcaagatag
gtcgttgtatctccttcctctcgtactctcatgaccacggaaag**ATGATCAAG**agaggatgatttcttggccatatcgcaatg
aatacttgtgacttgtgcttccaattgacatcttcagcgccatattgcgctggccaaggtgacggagcgggattacgaaagca
tgatcatggctgttgttctgtttatcttgttttgactgagacttgttaggatagacggtttttcatcactgactagccaaagc
cttactctgcctgacatcgaccgtaaattgataatgaatttacatgcttccgcgacgatttacct**CTTGATCAT**cgatccgat
tgaagatcttcaattgttaattctcttgcctcgactcatagccatgatgagct**CTTGATCAT**gtttccttaaccctctatttt
ttacggaaga**ATGATCAAG**ctgctgct**CTTGATCAT**cgtttc

ATGATCAAG
| | | | | | | | | are **reverse complements** and likely ***DnaA*** boxes
TACTAGTTC (***DnaA*** does not care what strand to bind to)

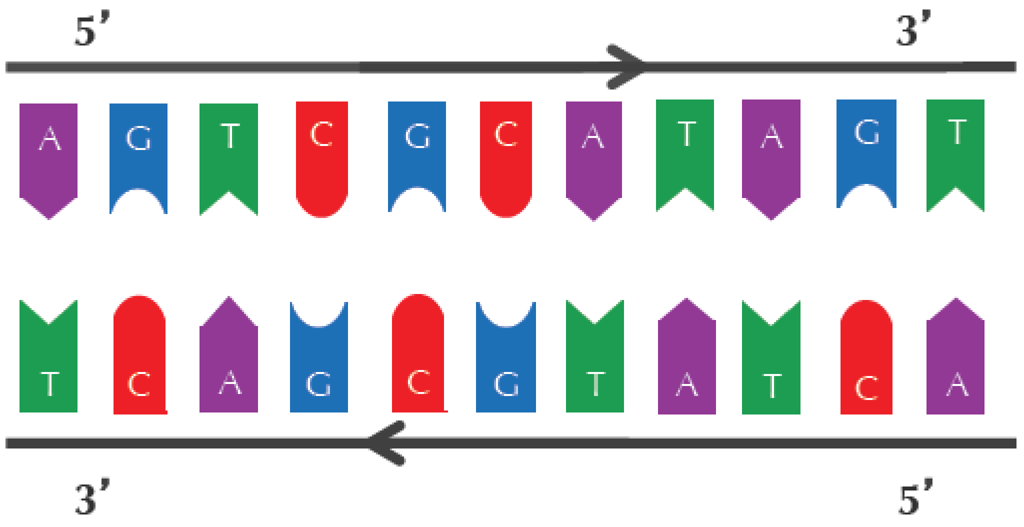
It is **VERY SURPRISING** to find a 9-mer appearing **6 or more** times (counting reverse complements) within a short ≈ 500 nucleotides.

Reverse Complement Sequence

- Recall that nucleotides **A** and **T** are complements of each other, as are **C** and **G**. Then, having one strand of DNA one can imagine the synthesis of a **complementary strand.**

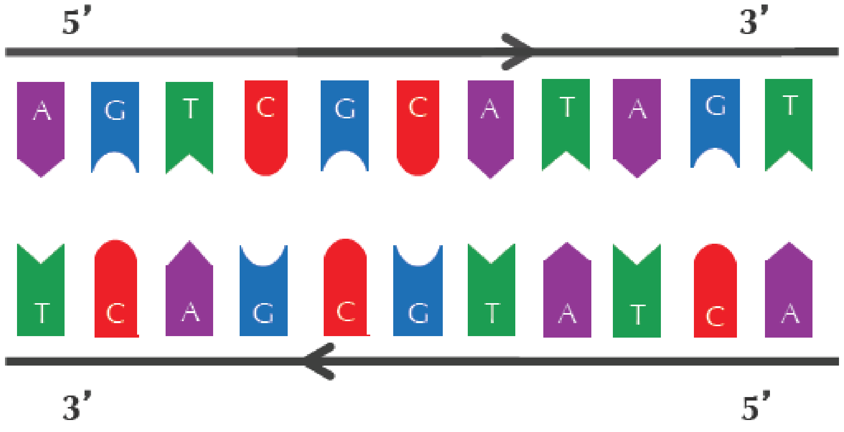
- For example, the strand **AGTCGCATAGT** has its complementary strand **ACTATGCGACT**. (not **TCAGCGTATCA**)

- This is because each strand is read in a specific direction denoted ($5' \longrightarrow 3'$) called 5 prime to 3 prime



Reverse Complement Sequence

- Given a nucleotide p, we denote its complementary nucleotide as $\bar{p}$.
- The **reverse complement** of a string $Pattern = p_1 \cdots p_n$ is the string $\overline{Pattern} = \overline{p_n} \cdots \overline{p_1}$ formed by
  - taking the complement of each nucleotide in Pattern,
  - then reversing the resulting string.



Reverse Complement Problem:
Find the reverse complement of a DNA string.

Input: A DNA string *Pattern*.
Output: $\overline{Pattern}$, the reverse complement of *Pattern*.

Important Notes

- We define COUNT(Text, Pattern) as the number of times that a *k*-mer Pattern appears as a substring of Text.
- For example:
  COUNT(ACA**ACTAT**GCAT**ACTAT**CGGGA**ACTAT**CCT, **ACTAT**) = 3.
- Note that COUNT(CG**ATATA**TCC**ATA**G, **ATA**) is equal to 3 (not 2) since we should account for overlapping occurrences of Pattern in Text.
- To compute COUNT(Text, Pattern), our plan is to “slide a window” down Text, checking whether each *k*-mer substring of Text matches Pattern.

Important Notes

- We will therefore refer to the *k*-mer starting at position *i* of Text as Text (*i*, *k*). We will often use **0-based indexing**.
- In this case, Text begins at position 0 and ends at position |Text|-1
- For example, if Text = GACCATACTG, then Text(4, 3) = ATA.
- Note that the last *k*-mer of Text begins at position |Text|- *k*
- For example: the last 3-mer of GACCATACTG starts at position 10 – 3= 7 (which is CTG)

Important Notes

- A straightforward algorithm for finding the most frequent k-mers in a string Text checks all k-mers appearing in this string (there are $|\text{Text}| - k + 1$ such k-mers) and then computes how many times each k-mer appears in Text.

- This algorithm, called **FREQUENTWORDS**, we will need to generate an array COUNT, where COUNT(i) stores the value COUNT(Text, Pattern) where Pattern = Text(i, k)

Text	**A**	**C**	**T**	G	**A**	**C**	**T**	C	C	C	A	C	C	C	C
COUNT	**2**	1	1	1	**2**	1	1	3	1	1	1	3	3		

```
FREQUENTWORDS(Text, k)
    FrequentPatterns ← an empty set
    for i ← 0 to |Text| − k
        Pattern ← the k-mer Text(i, k)
        COUNT(i) ← PATTERNCOUNT(Text, Pattern)
    maxCount ← maximum value in array COUNT
    for i ← 0 to |Text| − k
        if COUNT(i) = maxCount
            add Text(i, k) to FrequentPatterns
    remove duplicates from FrequentPatterns
    return FrequentPatterns
```

the **complexity** of this algorithm as $\mathcal{O}(|Text|^2 \cdot k)$

Explain