Motif Discovery:
Randomized Motif Search

Abhishek Ashok kumar
S2 CSE AIE A
AM.EN.U4AIE21002
ak2003xoin@gmail.com

Guru Aswini Dath
S2 CSE AIE A
AM.EN.U4AIE21023
guruaswinidath25@gmail.com

Eric Oommen Mathew
S2 CSE AIE A
AM.EN.U4AIE21027
ericoommen10@gmail.com

Manav Mani Prasad
S2 CSE AIE A
AM.EN.U4AIE21043
manav@gmail.com

Anupa Sajikumar
S2 CSE AIE A
AM.EN.U4AIE21071
anupasajikumar@gmail.com

Vishnuajith Manu
S2 CSE AIE A
AM.EN.U4AIE21082
vishmanu246@gmail.com

Abstract:

Randomized motif search is an algorithm which is used to find the longer motifs.Although the motifs returned by **RandomizedMotifSearch** are slightly less conserved than the motifs returned by **MedianString**, **RandomizedMotifSearch** has the advantage of being able to find longer motifs (since **MedianString** becomes too slow for longer motifs). In the epilogue, we will see that this feature is important in practice. We use the concepts of Greedy motif search, Profile-motif pseodocounts in oder to generate Randomized motif Search algorithm.

Keyword: Motif, Randomized motif Search, Median String, Greedy Motif Search, Profile- Motif pseodocount.

Introduction:

(*Rolling Dice to find Motifs***)**

We will now turn to **randomized algorithms** that flip 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. For more details, see "DETOUR: Buffon's Needle" in the print companion or at Stepik.

Randomized algorithms may be nonintuitive because they lack the control of traditional algorithms. Some randomized algorithms are **Las Vegas algorithms**, which deliver solutions that are guaranteed to be exact, despite the fact that they rely on making random decisions. Yet most randomized algorithms, including the motif finding algorithms that we will consider in this chapter, are **Monte Carlo algorithms**. These algorithms are not guaranteed to return exact solutions, but they do quickly find *approximate* solutions. Because of their speed, they can be run many times, allowing us to choose the best approximation from thousands of runs.

According to Bioinformatics, it is a kind of a new multidisciplinary field that specifically comes out from the combination of for all intents and purposes other sciences and fields like biology, computer science, statistics, chemistry, mathematics, and even definitely more in a definitely major way. In recent years new sciences actually have risen up pretty due to the demand to for all intents and purposes understand the world around us like Bioinformatics, Biotechnology, Computational Biology, Biochemistry, and others, demonstrating that in recent years new sciences actually have risen up definitely due to the demand to literally understand the world around us like Bioinformatics, Biotechnology, Computational Biology, Biochemistry, and others, particularly contrary to popular belief. It was a big challenge for researchers and scientists to give an adequate definition for each of these newly emerged sciences.

One of these sciences that have a huge influence in the medical field is Bioinformatics but also can play a key role in other fields like agriculture, livestock and even space explorations.

Bioinformatics attracts people in the general academic field in addition interest to those in the medical industry in a subtle way. There particularly were basically many contributions to specifically define and particularly explain Bioinformatics in scientific ways, but all researchers for the most part agree that it basically is a combination of Biology, Computer Science, Statistics, and Mathematics, very contrary to popular belief. Each one of

these disciplines for all intents and purposes is playing an important role in collecting, organizing, analysing, and digitizing pretty biological data, which kind of shows that each one of these disciplines mostly is playing an important role in collecting, organizing, analysing, and digitizing for all intents and purposes biological data. This paper will target four categories of readership who specifically are for all intents and purposes interested in the field in a subtle way. Students who specifically are very interested in studying this new field, which particularly is fairly significant. Instructors who would like to for all intents and purposes prepare a fundamental course to definitely teach in bioinformatics, so this paper will target four categories of readership who really are basically interested in the field in a fairly major way. Researchers who would like to for the most part understand generally more about Bioinformatics and its relationship with cancer, demonstrating how each one of these disciplines generally is playing an important role in collecting, organizing, analysing, and digitizing actually biological data, which for the most part shows that each one of these disciplines essentially is playing an important role in collecting, organizing, analysing, and digitizing actually biological data. Experts in the medical field who particularly are kind of interested in implementing the understanding of this field in medical life, which particularly is fairly significant. In short, bioinformatics literally is a management information system for molecular biology and generally has basically any kind of practical applications, or so they generally thought. So, Bioinformatics can literally be defined as the new fairly hybrid emerging field of science in which biology, computer science, mathematics, statistics, and Information Technology for the most part merge and generally interact together to form a pretty whole new discipline field, showing how so, Bioinformatics can generally be defined as the new hybrid emerging field of science in which biology, computer science, mathematics, statistics and Information Technology particularly merge and actually interact together to form a very whole new discipline field in an actual big way.

Noting that the suffix "informatics" literally is of very European origin; "Informatique" means and indicates computer science in really French and Bio really means Biology, for all intents and purposes further showing how in short, bioinformatics kind of is a management Information system for molecular biology and for all intents and purposes has definitely many kinds of practical applications, which essentially is fairly significant. It generally is a science used to manage, analyse, organize, and actually classify the huge amount of generally biological data by using well-developed algorithms, computational and statistical techniques, designing and constructing software tools and theories to actually solve different problems arising from kind of biological data and essentially help in generating, storing, accessing and analysing data and information that particularly is related to molecular biology in a pretty major way.

In Random Motif Search, On each run, they begin from a new randomly selected set of k-mers, selecting the best set of k-mers found in all these runs.

Literature Review:

The literature review essentially defines or explains the principles of this topic, which is often considered to be fairly important.

The gene is the fundamental unit of inherited information in deoxyribonucleic acid (DNA), and is defined as a section of base sequences that is used as a template for the copying process called transcription. The main idea in gene expression is that every gene contains the information to produce a protein. Gene expression begins with binding of multiple protein factors, known as transcription factors, to enhancer and promoter sequences. Transcription factors regulate the gene expression by activating or inhibiting the transcription machinery. Understanding the mechanisms that regulate gene expression is a major challenge in biology. Identifying regulatory elements, especially the binding sites in DNA for transcription factors is a major task in this challenge. Pattern discovery in DNA sequences is one of the most challenging problems in molecular biology and computer science. In its simplest form, the problem can be formulated as follows: given a set of sequences, find an unknown

pattern that occurs frequently. If a pattern of *m* letters long appears exactly in every sequence, a simple enumeration of all *m*-letter patterns that appear in the sequences gives the solution. However, when one works with DNA sequences, it is not that simple because patterns include mutations, insertions or deletions of nucleotides.

DNA motifs are often associated with structural motifs found in proteins. Motifs can occur on both strands of DNA. Transcription factors indeed bind directly on the double-stranded DNA. Sequences could have zero, one, or multiple copies of a motif. In addition to the common forms of DNA motifs two special types of DNA motifs are recognized: palindromic motifs and spaced dyad (gapped) motifs. A palindromic motif is a subsequence that is exactly the same as its own reverse complement, e.g., CACGTG. A spaced dyad motif consists of two smaller conserved sites separated by a spacer (gap). The spacer occurs in the middle of the motif because the transcription factors bind as a dimer. This means that the transcription factor is made out of two subunits that have two separate contact points with the DNA sequence. The parts where the transcription factor binds to the DNA are conserved but are typically rather small (3–5 bp). These two contact points are separated by a non-conserved spacer. This spacer is mostly of fixed length but might be slightly variable.

The aim of the motif discovery challenge is to identify overrepresented motifs as well as conserved motifs from orthologous sequences that are strong candidates for being transcription factor binding sites, given a set of DNA sequences (promoter region). There are numerous algorithms available for locating DNA motifs. By taking into account the regulatory region (promoter) of many coregulated genes from a single genome, the majority of these algorithms are made to infer motifs. Gene coexpression is thought to result mostly through transcriptional coregulation. As coregulated genes are known to have some regulatory mechanism overlap, presumably at the transcriptional level, their promoter regions may have some shared motifs that serve as transcription factor binding sites.

However, it has been demonstrated that the majority of these motif searching algorithms function substantially worse in higher species than they do in

yeast and other lower organisms. Recent motif searching algorithms use phylogenetic footprinting or cross-species genome comparison to get around this problem. The basic idea behind phylogenetic footprinting is that functional components evolve more slowly than non-functional sequences due to selective pressure. This means that among a group of orthologous promoter regions, areas that are often well conserved are great candidates for functional regulatory elements or motifs. Numerous motif-finding techniques based on phylogenetic footprinting have been created. Since then a remarkably rapid development has occurred in DNA motif finding algorithms and a large number of DNA motif finding algorithms have been developed and published.

- Motif Discovery Algorithms

Based on the type of DNA sequence information employed by the algorithm to deduce the motifs, we classify available motif finding algorithms into three major classes:

(1) those that use promoter sequences from coregulated genes from a single genome,

(2) those that use orthologous promoter sequences of a single gene from multiple species (i.e., phylogenetic footprinting)

(3) those that use promoter sequences of coregulated genes as well as phylogenetic footprinting.

However, word-based approaches might be troublesome for common transcription factor motifs, which frequently have multiple weakly constrained places, and the outcome frequently needs to be post-processed using some clustering algorithm. Too many fictitious motifs are produced through word-based techniques as well. The motif model is represented by a position weight matrix in the probabilistic approach. Position weight matrices are frequently represented as a pictogram, with each position represented by a stack of letters whose height is inversely correlated with its information content. Although probabilistic models of the regulatory areas are used, which can be very sensitive to minute changes in the input data, probabilistic techniques offer the advantage of having fewer search parameters.

Many of the algorithms developed from the probabilistic approach are designed to find longer or

more general motifs than are required for transcription factor binding sites. Therefore, they are more appropriate for motif finding in prokaryotes, where the motifs are generally longer than eukaryotes. However, these algorithms are not guaranteed to find globally optimal solutions, since they employ some form of local search, such as Gibbs sampling, expectation maximization (EM) or greedy algorithms that may converge to a locally optimal solution.

Randomized algorithms may be nonintuitive because they lack the control of traditional algorithms. Some randomized algorithms are Las Vegas algorithms, which deliver solutions that are guaranteed to be exact, despite the fact that they rely on making random decisions. Yet most randomized algorithms, including the motif finding algorithms that we will consider in this chapter, are Monte Carlo algorithms. These algorithms are not guaranteed to return exact solutions, but they do quickly find approximate solutions. Because of their speed, they can be run many times, allowing us to choose the best approximation from thousands of runs.

Method:

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 x *k* matrix *Profile*, we define *Motifs*(*Profile*, *Dna*) as the collection of *k*-mers formed by the *Profile*-most probable *k*-mers in each string from *Dna*. For example, consider the following *Profile* and *Dna*:

Profile

A:	4/5	0	0	1/5
C:	0	3/5	1/5	0
G:	1/5	1/5	4/5	0
T:	0	1/5	0	4/5

Dna

```
ttaccttaac
gatgtctgtc
acggcgttag
ccctaacgag
cgtcagaggt
```

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
cgtcagaggt
```

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 our 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. . .

...*Profile*(*Motifs*(*Profile*(*Motifs*(*Profile*(*Motifs*), *Dna*)), *Dna*))...

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 *RandomNumber*(*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* = ($Motif_1$, …, $Motif_t$) in each string from *Dna*

    BestMotifs ← *Motifs*

    while forever

        Profile ← *Profile*(*Motifs*)

        Motifs ← *Motifs*(*Profile*, *Dna*)

        if *Score*(*Motifs*) < *Score*(*BestMotifs*)

            BestMotifs ← *Motifs*

        else

            return *BestMotifs*

Since a single run of **RandomizedMotifSearch** may generate a rather poor set of motifs, bioinformaticians usually run this algorithm thousands of times. On each run, they begin from a new randomly selected set of *k*-mers, selecting the best set of *k*-mers found in all these runs.

```
RandomizedMotifSearch(Dna, k, t)
    randomly select k-mers Motifs = (Motif_1, …, Motif_t) in each
    BestMotifs ← Motifs
    while forever
        Profile ← Profile(Motifs)
        Motifs ← Motifs(Profile, Dna)
        if Score(Motifs) < Score(BestMotifs)
            BestMotifs ← Motifs
            else
            return BestMotifs
```

- **Input:** Integers *k* and *t*, followed by a collection of strings *Dna*.
- **Output:** A collection *BestMotifs* resulting from running **RandomizedMotifSearch**(*Dna*, *k*, *t*) 1,000 times. Remember to use pseudocounts!

Why Randomized Motif Works?

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 upper case letters below) and imagine that it chooses the following 4-mers *Motifs* (shown in red) 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.

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

Profile(Motifs)				
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 matrix. For example, the probability of the first 4-mer in the first string of *Dna* is Pr(ttAC|*Profile*) = 0.2 · 0.2 · 0.2 · 0.2 = 0.0016. The maximum probabilities in every row are shown in red below.

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

We select the most probable 4-mer in each row above as our

new collection *Motifs* (shown below). Notice that this collection has captured all five implanted motifs in *Dna*!

```
        ttACCTtaac
        gATGTctgtc
Dna     ccgGCGTtag
        cactaACGAg
        cgtcagAGGT
```

For the Subtle Motif Problem with implanted 15-mer AAAAAAAAGGGGGGG, when we run **RandomizedMotifSearch** for 100,000 times (each time with new randomly selected *k*-mers), it returns the 15-mers shown in the figure below as the lowest scoring collection *Motifs* across all iterations, resulting in the consensus string AAAAAAAAacaGGGG with score 43. These strings are only slightly less conserved than the collection of implanted (15, 4)-motifs with score 40 (or the motif returned by **GreedyMotifSearch** with pseudocounts having score 41), and it largely captures the implanted motif. Furthermore, unlike **GreedyMotifSearch**, **RandomizedMotifSearch** can be run for a larger number of iterations to discover better and better motifs.

RandomizedMotifSearch has the advantage of being able to find longer motifs. In the epilogue, we will see that this feature is important in practice.

		Score
Motifs	AAAtAcAgACAGcGt	5
	AAAAAAtAgCAGGGt	3
	tAAAAtAAACAGcGG	3
	AcAgAAAAAaAGGGG	3
	AAAAtAAAACtGcGa	4
	AtAgAcgAACAcGGt	6
	cAAAAAgAgaAGGGG	4
	AtAgAAAAggAaGGG	5
	AAgAAAAAAgAGaGG	3
	cAtAAtgAACtGtGa	7
Consensus(Motifs)	AAAAAAAAACAGGGG	43

- **Figure:** The lowest scoring collection of strings Motifs produced by 100,000 runs of **RandomizedMotifSearch**, along with their consensus string and score for the Subtle Motif Problem.

Although the motifs returned by **RandomizedMotifSearch** are slightly less conserved than the motifs returned by **MedianString**, **RandomizedMotifSearch** has the advantage of being able to find longer motifs (since **MedianString** becomes too slow for longer motifs). In the

epilogue, we will see that this feature is important in practice.

How can Randomized Algorithm perform so well?

We began with a collection of implanted motifs that resulted in the following profile matrix.

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**

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 other 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 first of the two profile matrices above. In practice, we are likely to obtain a profile matrix 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. Fortunately, **RandomizedMotifSearch** is designed so that subsequent steps have a good chance of leading us toward this implanted motif (although it is not certain).

If you still doubt the efficacy of randomized algorithms, 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! These multiple occurrences of the same motif may direct the profile matrix away from the uniform profile and toward the implanted motif. For example, consider again the original randomly selected *k*-mers *Motifs* (shown in red):

Dna
ttACCT**taac**
gAT**GTct**gtc
ccgGCGTtag
c**acta**ACGAg
cgtcag**AGGT**

You will see that the 4-mer **AGGT** in the last string 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. Note that only completely captured motifs (like **AGGT**) rather than partially captured motifs (like **GTct** or **ccgG**) contribute to the statistical bias in the profile matrix.

Unfortunately, a single implanted motif is frequently not enough to direct RandomizedMotifSearch to the best answer. As a result, the method of randomly choosing motifs is frequently less effective than in the straightforward example above due to the enormous number of beginning places of k-mers. It seems unlikely that these k-mers, which were chosen at random, will be able to lead us to the ideal answer.

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

Application of Randomized- Motif Search:

Tuberculosis Hibernation:

Over a million people die each year from tuberculosis (TB), an infectious disease that is brought on by the Mycobacterium tuberculosis bacterium (MTB). Antibiotics have significantly slowed the spread of TB, but new strains that are resistant to all therapies are now appearing. One-third of the world's population has latent MTB infections, in which MTB lays dormant within the host's body and may or may not reactivate at a later period. MTB is a successful pathogen because it may live in humans for decades without producing disease. TB outbreaks are challenging to manage due to the broad frequency of latent infections. Therefore, researchers are curious about how the disease becomes dormant and how MTB manifests itself in a living thing.

Hypoxia, or oxygen shortage, is often associated with latent forms of TB. Biologists have found that MTB 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 in finding a **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.

In 2003, 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 change significantly in hypoxic conditions. Given the upstream regions of these genes, each of which is 250 nucleotides long, we would like to discover the "hidden message" that DosR uses to control the expression of these genes.

MedianString			RandomizedMotifSearch		
k	Consensus	Score	*k*	Consensus	Score
8	CATCGGCC	11	8	CCGACGGG	13
9	GGCGGGGAC	16	9	CCATCGGCC	16
10	GGTGGCCACC	19	10	CCATCGGCCC	21
11	GGACTTCCGGC	20	11	ACCTTCGGCCC	25
12	GGACTTCCGGCC	23	12	GGACCAACGGCC	28

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). The motif of length 20 returned by **RandomizedMotifSearch** is CGGGACCTACGTCCCTAGCC (with score 57). As shown below, the consensus strings of length 12 found by **RandomizedMotifSearch** and **MedianString** are “embedded” with small variations in the longer motif of length 20:

GGACCAACGGCC

CGGGACCTACGTCCCTAGCC

GGACTTCCGGCC

Finally, in 2,000 runs with *N* = 200, **GibbsSampler** returned the same consensus string of length 20 for the DosR dataset as **RandomizedMotifSearch** but generated a different collection of motifs with a smaller score of 55.

Here,different motif finding algorithms generate somewhat different results, and and it remains unclear how to find all DosR bonding sites in MTB. Try to answer this question and find all putative DosR motifs in MTB as well as all genes that they regulate. We will provide you with the upstream regions of ten of the 25 genes identified in the DosR study by Park et al., 2003.

Result:

From the code implemented (Shown in Appendix) gives us the the prototype of the sequential analysis of a simple representation of a sequence:

```
['AAACGGCC',
'TAAGTGCC', 'GACCGAAA',
'AGGTGCAC', 'CAATGTTG']
```

Appendix:

Link for the code for Randomized Motif Search (Colab):

https://colab.research.google.com/drive/1fdqInP3gtt4aUZRQa3QT4XMiUgwXvTTj?usp=sharing

References:

1. https://link.springer.com/article/10.1186/1471-2105-8-S7-S21
2. https://www.bioinformaticsalgorithms.org/bioinformatics-chapter-2
3. https://www.nature.com/articles/srep07813
4. https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-15-34
5. https://dl.acm.org/doi/abs/10.1145/369133.369172
6. https://amritauniv.sharepoint.com/sites/19BIO112IntelligenceofBiologicalSystems2-S2AIEB/SharedDocuments/Forms/AllItems.aspx?id=%2Fsites%2F19BIO112IntelligenceofBiologicalSystems2-S2AIEB%2FSharedDocuments%2FGeneral%2FBooks%2F1_BA-chap1and2%2Epdf&parent=%2Fsites%2F19BIO112IntelligenceofBiologicalSystems2-S2AIEB%2FSharedDocuments%2FGeneral%2FBooks&p=true&ga=1
7. 14_RandomizedMotifSearch.pptx (sharepoint.com)
8. https://amritauniv.sharepoint.com/sites/19BIO112IntelligenceofBiologicalSystems2-S2AIEB/SharedDocuments/Forms/AllItems.aspx?id=%2Fsites%2F19BIO112IntelligenceofBiologicalSystems2-S2AIEB%2FSharedDocuments%2FGeneral%2FBooks%2F2_TextBook_BioinformaticsAlgorithms%2Epdf&parent=%2Fsites%2F19BIO112IntelligenceofBiologicalSystems2-S2AIEB%2FSharedDocuments%2FGeneral%2FBooks&p=true&ga=1