Frequent words with mismatches and reverse compliment problem

21BIO112 Intelligence of biological systems

Group no - 13
Members
S HARI SANKAR
AM.EN.U4AIE21056
R RAKESH
AM.EN.U4AIE21052
SAI MANASA SOUMYA
AM.EN.U4AIE21057
K SUBASH
AM.EN.U4AIE21036
CHERISHMA AKSHAYA
AM.EN.U4AIE21052

***Abstract*—In the genome of every species, there exists an origin, known as the origin of replication (ORI), from where the genome starts to replicate itself during the process of cell division. Finding out this origin; is therefore a very prime and demanding problem in bioinformatics research, as this is the main responsible key-factor for the replication process of DNA. In this study we start off by choosing a benchmark dataset of a Frequent words with mismatch and reverse complement problem. We are using python as our programming language to find the solution to our problem. We are finding the solution to find the replication origin (Oric region).**

Keywords—Origin of replication (ORI), Bioinformatics, DNA replication, Frequent words with mismatches and reverse compliment

I. Introduction

The information stored in the DNA, plays a vital role in every life. When a cell dies or needs to get replaced, it is important to copy and reserve the information of that cell. Replication is the process that is responsible for this act. So, finding out the origin of replication is a problem, that needs to be solved with a smart computational approach.

A. DNA

DNA stands for Deoxyribonucleic Acid and it is the hereditary element in almost all living organisms. The information stored in DNA consists of four chemical bases: adenine (A), guanine (G), cytosine (C), and thymine (T). The sequence of these letters (or bases A, C, G, T) is the information available for making and maintaining an organism. The most important task of DNA is to replicate itself and help in cell division.

B. Replication

Replication is the process, where the DNA makes a same identical copy of itself. Whenever a cell gets divided, it makes a copy of its genome and the new cell contains the same information as the parent cell, so that it can work or behave as a template of its parent cell. Replication is the most important feature of every living organism. The replication starts at a point in the DNA, called the origin of chromosomal replication (ORIc), where the DNA double helix opens up.

C. DnaA and DnaA box

DnaA is the protein which is the replication initiation factor that promotes the unwinding of DNA at Oric. The onset of the initiation phase of DNA replication is determined by the concentration of DnaA. Replication begins with active DnaA binding to 9-mer (9-bp) repeats upstream of Oric called DnaA Box**.** Binding of DnaA-to-DnaA Box leads to strand separation.

D. Problem Statement

By using advanced methods like the minimum skew problem, we find an approximate location of Oric at position 3923620 in E. coli Genome. In an attempt to confirm this hypothesis, we will look for a hidden message representing a potential DnaA box near this location. After using the Frequent Words Problem to solve in a window of length 500 starting at position 3923620 of the genome reveals no 9-mers that appear three or more time. Even if we have managed to locate Oric in E. coli, it appears that we still have not found the DnaA boxes that jump-start replication in this bacterium.

E. Motivation

Upon examining the Oric of Vibrio cholerae one more time. We have noticed that in addition to the three occurrences of ATGATCAAG and three occurrences of its reverse complement CTTGATCAT, the Vibrio cholerae Oric contains additional occurrences of ATGATCAAC and CATGATCAT, which differ from ATGATCAAG and CTTGATCAT in only a single nucleotide:

Finding eight approximate occurrences of our target 9-mer and its reverse complement in a short region is even more statistically surprising than finding the six exact occurrences of ATGATCAAG and its reverse complement CTTGATCAT that we stumbled upon in the beginning of our investigation. Furthermore, the discovery of these approximate 9-mers makes sense biologically, since DnaA

can bind not only to “perfect” DnaA boxes but to their slight variations as well.

F. Objective

In this project, our goal is to modify our previous algorithm for the Frequent Words Problem in order to find DnaA boxes by identifying frequent k-mers, possibly with mismatches. Given strings Text and Pattern as well as an integer d, we define COUNT d (Text, Pattern) as the number of occurrences of Pattern in Text with at most d mismatches.

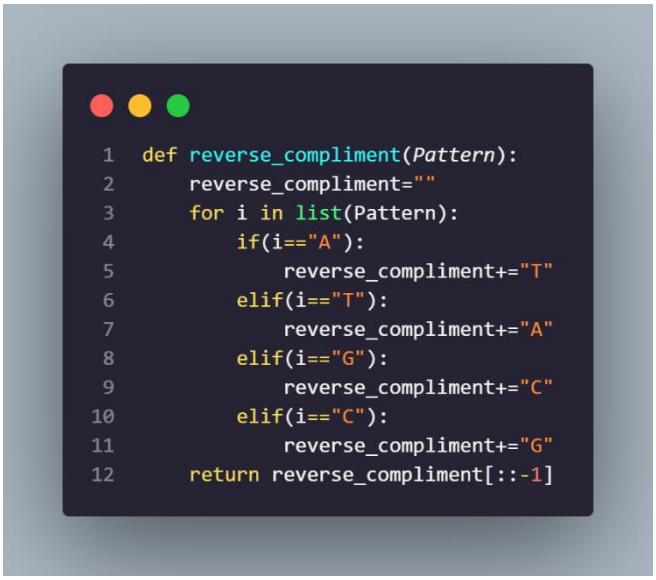
II. METHODOLOGY

A. Reverse compliment problem

In the first part of our algorithm, we will be dealing with the solution of finding the reverse compliment of a given DNA string. Reverse Complement converts a DNA sequence into its reverse, complement, or reverse-complement counterpart. For finding out the reverse compliment we use the following algorithm

- Replace every occurrence of adenine(A) with thymine(T) and vice versa.
- Replace every occurrence of guanine(G) with cytosine(C) and vice versa
- Reverse the resulting DNA string formed

The above algorithm implemented in python is shown below:



```
def reverse_compliment(Pattern):
    reverse_compliment=""
    for i in list(Pattern):
        if(i=="A"):
            reverse_compliment+="T"
        elif(i=="T"):
            reverse_compliment+="A"
        elif(i=="G"):
            reverse_compliment+="C"
        elif(i=="C"):
            reverse_compliment+="G"
    return reverse_compliment[::-1]
```

B. Hamming distance problem

The Hamming distance between two equal-length strings of symbols is the number of positions at which the corresponding symbols are different. The hamming distance also gives the number of mismatches between two DNA strings of a given length. It is one of the essential components of the algorithm which we are going to develop for frequent words with mismatches and reverse compliment. The algorithm for finding out the hamming distance between to Strings is given below-:

- Iterate through each character in the first string
- While iterating, check if the character at any given point in the first string is equal to the character, if not it corresponds to a mismatch
- Find out the total number of mismatches

The above algorithm implemented in python is shown below:

```
def hamming_distance(String1,String2):
    return sum(1 for i in range(len(String1)) if String1[i]!=String2[i])
```

C. Approximate pattern count

In the algorithm for frequent words problem, we were using the pattern count algorithm in which we were counting the number of terms a particular pattern appeared in the DNA string. But in the algorithm for approximate pattern count, we will be using the hamming distance method and counting the number of times a pattern appears in a string with at most d mismatches. The algorithm for approximate pattern count is given below-:

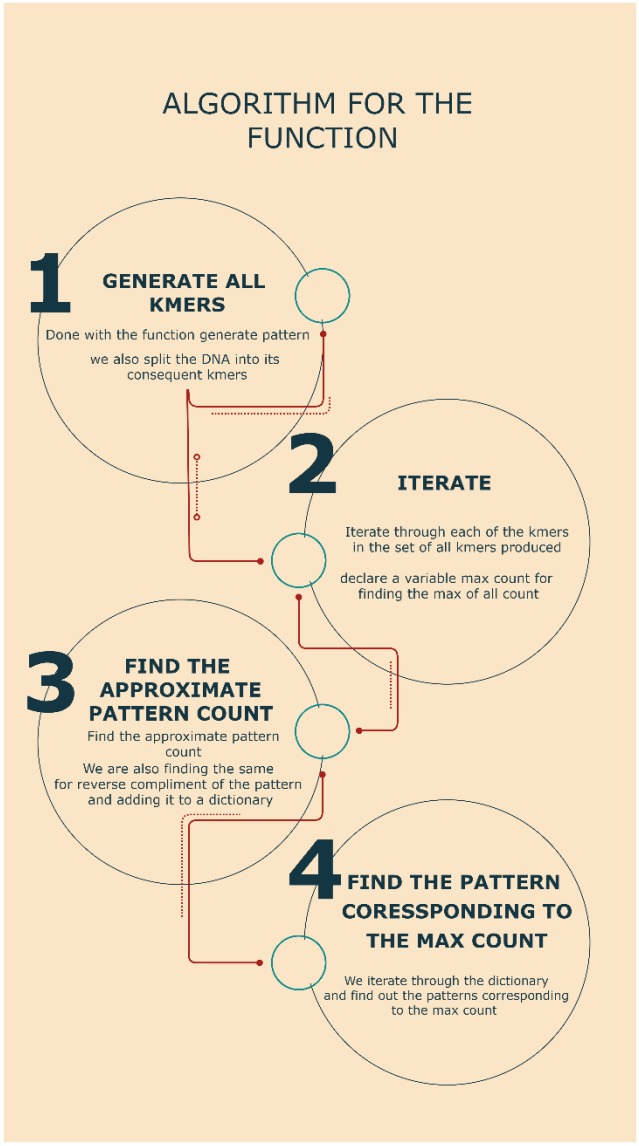
- Take each kmer from a particulate String and apply hamming distance to it with the pattern we are currently considering
- If the hamming distance between the string and pattern is less than or equal to d, the count increases by 1
- Return the total count at the end of function

The above algorithm implemented in python is shown below:

```
def Approximate_pattern_count(Pattern,kmers,d):
    return sum(1 for i in kmers if hamming_distance(Pattern,i)<=d)
```

D. Frequent words problem with mismatches and reverse compliments

The frequent words problem with mismatches and reverse compliments is an extension of frequent words problem, in which we are integrating all the above discussed functions and producing the final functions. Frequent words with mismatches unlike normal frequent words problem does not check for pattern count, but checks for the number of mismatches between the pattern and kmer to be at most the given d value. It also checks the same for the reverse compliments of the pattern. Then it will sum up the approximate pattern count values of both the pattern and its reverse compliment. The pattern with the maximum count value may be the DnaA box pattern which we are looking for. The input for the frequent words with mismatches and reverse compliment includes a DNA string, the k value and number of mismatches d. The algorithm used for the problem is given below -:



The above algorithm implemented in python code is given below-:

```
def Frequent_words_with_mismatches_and_reverse_complement(Text,k,d):
    count_pattern={}
    output=[]
    Patterns = generate_Pattern(k)
    kmers = split_kmer(Text,k)
    max_Count=0
    for Pattern in Patterns:
        reverse = reverse_compliment(Pattern)
        count_pattern[Pattern]=Approximate_pattern_count(Pattern,kmers,d)+Approximate_pattern_count(reverse,kmers,d)
        max_Count = count_pattern[Pattern] if count_pattern[Pattern]>max_Count else max_Count
    output = [Pattern for Pattern in count_pattern if count_pattern[Pattern]==max_Count]
    print(*output,end=' ')
```

4) AGTGGGGGAGTGGGGGCACTCCCGATTTACATC
CACTCCCGACACTCCCGATTTACATCAGTGGGGGTTTA
CATCAAGGTTGGCTGATCCAAGGTTGAAGGTTGTTTAC
ATCGCTGATCCTTTACATCGCTGATCCTTTACATCAGT
GGGGGTTTACATCAGTGGGGGGCTGATCCTTTACATC
TTTACATCAGTGGGGGCACTCCCGAGCTGATCCGCTG
ATCCAGTGGGGGAGTGGGGGAAGGTTGGCTGATCCA
GTGGGGGGCTGATCCAGTGGGGGCACTCCCGAAGTG
GGGGGCTGATCCGCTGATCCGCTGATCCCACTCCCGA

III. RESULTS

A. Datasets

The following data sets from rosalind.info/problems/ba1j/ were used for testing the model we have created.

1) DNA-
ACGTTGCATGTCGCATGATGCATGAGAGCT
K value = 4 d value = 1
OUTPUT FROM THE MODEL

ATGT ACAT

2) DNA-
TGTCTGAGTGTCTGAGTGTCTGAGACGTATGGCAACC
ATTCGCACGTATGGCATCAGAGAGTGTCTGAGACGTAT
GGCAAGCCGATCGTGTCTGAGACGTATGGCAACCATT
CGCACGTATGGCAACCATTCGCTCAGAGAGACCATTC
GCACGTATGGCATCAGAGAGACCATTCGCACCATTCG
CTCAGAGAGTGTCTGAGAGCCGATCGACCATTCGCTG
TCTGAGACCATTCGCAGCCGATCGAGCCGATCGACCA
TTCGCACGTATGGCATCAGAGAGAGCCGATCGACCAT
TCGCTGTCTGAGACGTATGGCATCAGAGAGTGTCTGA
GACGTATGGCATCAGAGAGACGTATGGCATCAGAGAG
ACCATTCGCACGTATGGCAAGCCGATCGTCAGAGAGA
CGTATGGCAACCATTCGCACGTATGGCAACGTATGGC
AAGCCGATCGTCAGAGAGACGTATGGCATGTCTGAGA
CGTATGGCAACGTATGGCAAGCCGATCGAGCCGATCG
ACGTATGGCATGTCTGAGTCAGAGAGAGCCGATCGTC
AGAGAGACCATTCGCTCAGAGAGACCATTCGCAGCCG
ATCGTCAGAGAGTCAGAGAGTGTCTGAGTCAGAGAGA
CGTATGGCAACCATTCGCTCAGAGAGACCATTCGCAC
GTATGGCAACCATTCGCTCAGAGAGAGCCGATCGTGT
CTGAGAGCCGATCGACGTATGGCAAGCCGATCGACGT
ATGGCAAGCCGATCGACCATTCGCTCAGAGAGACGTA
TGGCAAGCCGATCGTCAGAGAGACCATTCGCAGCCGA
TCGAGCCGATCGAGCCGATCGACCATTCGCACGTATG
GCAACCATTCGCAGCCGATCGAGCCGATCGACCATTC
GCTCAGAGAGAGCCGATCGTGTCTGAGTCAGAGAGAC
CATTCGCTCAGAGAGAGCCGATCGTGTCTGAGAGCCG
ATCGACCATTCGCACGTATGGCA
K Value – 6 d value – 2

OUTPUT FROM THE MODEL

CGTCCG CGGACG

3) DNA-
GCTGATCCAAGGTTGAGTGGGGGCACTCCCGATTTAC
ATCCACTCCCGAAAGGTTGTTTACATCAGTGGGGGTTT
ACATCGCTGATCCGCTGATCCAAGGTTGCACTCCCGA
CACTCCCGATTTACATCCACTCCCGAGCTGATCCAAGG
TTGAAGGTTGGCTGATCCTTTACATCCACTCCCGAAAG
GTTGCACTCCCGAAAGGTTGAAGGTTGTTTACATCGCT
GATCCCACTCCCGAGCTGATCCAAGGTTGTTTACATCT
TTACATCCACTCCCGAAAGGTTGGCTGATCCAAGGTTG
AGTGGGGGTTTACATCAAGGTTGAAGGTTGTTTACATC

TTTACATCAGTGGGGGGCTGATCCCACTCCCGAAGTG
GGGGGCTGATCCAAGGTTGAAGGTTGCACTCCCGAAG
TGGGGGAGTGGGGGGCTGATCCAAGGTTGTTTACATC
AGTGGGGGAGTGGGGGAGTGGGGGTTTACATCAGTG
GGGGTTTACATCAAGGTTGGCTGATCCAAGGTTG
K value – 5, d value - 3

OUTPUT FROM THE MODEL

CGTGC GCACG

When the same problem was used for finding the Oric in a 500 size at starting position 3923620 E. coli genome, the resulting 9 mer found was the DnaA box was TTATCCACA and its reverse compliment TGTGGATAA. The pattern had only a mismatch of 1

We were very fortunate that the DnaA boxes of E. coli are captured in the window that we chose. Moreover, while TTATCCACA represents a most frequent 9-mer with 1 mismatch and reverse complements in this 500-nucleotide window, it is not the only one: GGATCCTGG, GATCCCAGC, GTTATCCAC, AGCTGGGAT, and CTGGGATCA also appear four times with 1 mismatch and reverse complements.

IV. Conclusions

Thus, we can conclude from this project that the frequent words with mismatches is an efficient algorithm for finding the replication origin in a genomic sequence. The dataset output when tested in the Rosalind website, showed positive results. Another thing which we can conclude that from the problem that we have completed in the project is that bioinformatics when combined with computational engineering techniques can help us to identify the basics of life. It will intern result in future efficient technologies which may even include stopping cureless diseases and conditions like cancers. The next question hints at how Oric predictions can be carried out using comparative genomics, a bioinformatics approach that uses evolutionary similarities to answer difficult questions about genomes.

Acknowledgment

We the members of group 13 would like to show our sincere gratitude towards Dr. Manjusha Nair, Assistant Professor (Sl Gd), for introducing us to the field of bio informatics and motivating us to put a lot of effort and master every possible experimental approach and research methodology in the field of Bio informatics. We would also like to thank our faculties and alumni for their unwavering support, consideration, and inspiration during the duration of this study.

References

- http://dspace.uiu.ac.bd/bitstream/handle/52243/2123/A_Novel_Approach_to_Predict_the_Origin_of_Replication.pdf?sequence=1&isAllowed=
- https://algorithmcode.wordpress.com/2014/09/20/dna-replication-frequent-words-reverse-
- https://rosalind.info/problems/ba1j/explanation/

  complement-pattern-matching-clump-finding-skewi-mismatches/
- Appendix A – source code - https://github.com/subhash137/Frequent-Words-Problem-With-Mismatches-and-Reverse-Complement-Problem.git

****