



SCHOOL OF COMPUTING & INFORMATION TECHNOLOGY

A Project Report
on
“Genome Sequencing and Analysis on Cancer Codons”

Submitted in fulfillment of the requirements for the award of the Degree of

Bachelor of Technology

Submitted by

SHAILESH.D	R14CS182
SWETHA SIVAKUMAR	R14CS264
MANJUNATH.C	R14CS255
HARSHITHA.K. B	R13CS003

Under the guidance of

PROF. ANOOJA ALI
Assistant Professor,
School Of C&IT,
REVA UNIVERSITY

May 2019

Rukmini Knowledge Park, Kattigenahalli, Yelahanka,
Bengaluru-560064

www.reva.edu.in

DECLARATION

I, Mr./Ms. **Shailesh.D, Swetha Sivakumar, Manjunath.C, Harshitha.K.B**, student of B.Tech, belong in to School of C & IT, REVA University, declare that this Project Report / Dissertation entitled “Genome Sequencing and Analysis on Cancer Codons” is the result the of project / dissertation work done by me under the supervision of Prof. Anooja Ali at School of Computing and Information Technology, REVA University.

I am submitting this Project Report / Dissertation in partial fulfillment of the requirements for the award of the degree of Master of Bachelor of Technology in Computer Science and Engineering by the REVA University, Bangalore during the academic year 2017-18.

I declare that this project report has been tested for plagiarism and has passed the plagiarism test with the similarity score less than 25% and it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

I further declare that this project / dissertation report or any part of it has not been submitted for award of any other Degree / Diploma of this University or any other University/ Institution.

(Signature of the candidate)

Signed by me on

*Certified that this project work submitted by **Shailesh.D, Swetha Sivakumar, Manjunath.C, Harshitha.K. B** has been carried out under our guidance and the declaration made by the candidate is true to the best of my knowledge.*

Signature of Guide

Date:

Signature of Director of School

Date :.....

Official Seal of the School



SCHOOL OF COMPUTING & INFORMATION TECHNOLOGY

CERTIFICATE

Certified that the project work entitled **Genome Sequencing and Analysis on Cancer Codons** carried out under our guidance by **Shailesh.D(R14CS182), Swetha Sivakumar(R14CS264), Manjunath.C(R14CS255), Harshitha.K.B(R13CS003)**, bonafide students of REVA University during the academic year 2018-19, is submitting the project report in partial fulfillment for the award of **Bachelor of Technology** in Computer Science and Engineering during the academic year **2018-19**. The project report has been tested for plagiarism and has passed the plagiarism test with the similarity score less than 25%. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Signature with date

Prof. Anooja Ali
Guide

Signature with date

Dr. Sunil Kumar S. Manvi
Director

Sl. No.	Name of Examiner & Affiliation	Signature & Date
1		
2		

ACKNOWLEDGEMENT

Success of every candid effort perpetually needs the dedicated support from several people without them it would have been nearly impossible and such a comprehensive work eventually wouldn't be complete without express our gratitude and appreciation to those who stood by me and made it possible.

I would like to express my sincere gratitude to the esteemed REVA University, Bangalore for the wonderful opportunity given to me in carrying out the research work.

I express my deepest gratitude to **Dr. P. Shyama Raju**, Chancellor, REVA University, Bangalore, for the environment and infrastructure provided to carry out my research activities under one roof in REVA University campus, Bangalore.

I owe my deepest gratitude to **Dr. S. Y. Kulkarni**, Vice-Chancellor REVA University, Bangalore, for his continued support, encouragement and for making it possible to complete the submission of thesis in time.

It is pleasure to express my gratitude whole heartily thanks to **Dr. Sunil Kumar S. Manvi**, Director, School of Computing and Information Technology, REVA University, Bangalore, to timely process the research related tasks.

I wish to express heartfelt thanks to my research supervisor **Anooja Ali, Assistant Professor**, REVA University, for accepting me as a research scholar and for his valuable guidance, motivation during my research work. He has been a source of inspiration and his tireless guidance and incredible effort has enabled to complete the thesis successfully. His Simplicity, honesty, patience, generosity, and perfectness are qualities that have inspired me a lot.

Finally, I thank Almighty for his unlimited blessings without which I would not have reached this stage.

SHAILESH.D

SWETHA SIVAKUMAR

MANJUNATH.C

HARSHITHA.K.B

ABSTRACT

Sequences are common, occurring in any set that facilitates either total or partial ordering of the existence of sequences might even be very important.

A sequence alignment could be a method of arrangement of the sequences of DNA, RNA, or macromolecule to spot regions of similarity which will be a consequence of practical, structural, or organic process relationships between the sequences.

String matching is the technique of finding the occurrences of a character pattern in a given string.

This project provides an overview of different string-matching algorithms and comparative study of these algorithms.

In this project, We have evaluated several algorithms, such as Naive string-matching algorithm, Boyer-Moore algorithm, and their runtimes are compared, which leads to their comparative analysis of the algorithms.

The Cancer Mutated Sequences are evaluated under these algorithms where the accuracy of the mutation detection is exceptionally high.

TABLE OF CONTENTS

1 INTRODUCTION.....	1-8
1.1 INTRODUCTION.....	1
1.2 STATEMENT OF THE PROBLEM.....	2
1.3 OBJECTIVE OF THE PROBLEM.....	2
1.4 SCOPE OF THE PROJECT.....	2-3
1.4.1 EXISTING SYSTEM AND ITS DRAWBACKS.....	2
1.4.2 PROPOSED SYSTEM AND ITS ADVANTAGES.....	3
1.5 METHODOLOGY.....	3-7
1.5.1 BIOPYTHON.....	3
1.5.2RELATED WORK.....	4
1.5.2.1 INCOMPATIBILITY AMONG PROTEIN AND DNA.....	4
1.5.2.2 UNIQUE PROPERTY OF PROTEINS.....	4
1.5.2.3 WORKING ON FASTA FILES.....	5
1.5.2.4 REVERSE COMPLEMENT OF GENE SEQUENCES.....	6
1.5.2.5 REVERSE COMPLEMENT ON FASTA.....	6
1.5.2.6 I/O OPERATIONS DOCUMENTATION.....	7
1.5.2.7TRANSLATION.....	7
1.6 ORGANISATION OF THE REPORT.....	8
2 LITERATURE SURVEY.....	9-10
3.SYSTEM REQUIREMENTS.....	11
4 DESIGN.....	12-14
4.1DATA FLOW DIAGRAM.....	12
4.2USE CASE DIAGRAM.....	12
4.3SEQUENCE DIAGRAM.....	13
4.4CLASS DIAGRAM.....	13
4.5 ER DIAGRAM.....	14

5 MODULES DESCRIPTION.....	15-18
5.1 DATA PREPROCESSING.....	15
5.2 PATTERN MATCHING.....	16
5.3 SEQUENCE ALIGNMENT.....	17
5.4 MUTATION DETECTION.....	18
5.5 FASTA VISUALIZATION.....	18
6 IMPLEMENTATION.....	19-20
6.1 DATASET.....	19
6.2VISUALISATION TOOLS-FLUENT DNA.....	20
7.SNAPSHOTS.....	21-25
8.ALGORITHMS.....	26-29
8.1 K-MER INDEXING.....	26
8.2 BOYER MOORE ALGORITHM.....	27
8.3SMITH-WATERMAN ALGORITHM.....	28
8.4NEEDLEMAN-WUNSCH ALGORITHM.....	29
9.CONCLUSION AND FUTURE ENHANCEMENT.....	30
9.1 CONCLUSION.....	30
9.2 FUTURE ENHANCEMENT.....	30
REFERENCES.....	31
BIBLIOGRAPHY.....	32
APPENDIX	33-34
PLAGIARISM REPORT.....	33
PAPER PUBLICATION.....	34

Chapter - 1

INTRODUCTION

1.1 INTRODUCTION

Genome Sequencing in the field of Bioinformatics is quite an evolving application as all the information of an organism is carried on its DNA, It can accumulate mutation and it can be transferred from one generation to other.

Sequencing and Analysis of genome helps in rapid clinical diagnosis and treatment of diseases and advance scientific research.

The accuracy of the Genome Analysis is quite unsubstantial and the incorrect diagnosis can lead to differing complications.

The solution for this drawback is the illustrative process in modules where each Algorithm analysis the mutated sequences and their comparative analysis is carried out.

During the Analysis of Genome Sequences, a given sequence is selected and, indexing of nucleotide, pattern matching of sequences, alignment of sequences and giving the accuracy of finding mutations to perfection.

1.2 STATEMENT OF THE PROBLEM:

- Genome Analysis using different approaches is beneficial to the biotechnology aspirants and genetic researches which helps in further discoveries and approaches.
- This project is mainly intended to provide accuracy in the rate of detection of mutations and the comparative analysis of Algorithms in each module with their runtimes.

1.3 OBJECTIVE OF THE PROBLEM

In Genetics, the problem statement is mainly to provide accuracy and analysis of each algorithms.

1.4 SCOPE OF THE PROJECT: -

1.4.1 EXISTING SYSTEM AND ITS DRAWBACKS

Existing System deals with Genome Sequencing with aberrated precisions and misconceptions about Genome Analysis Results is received by the general patients.

The Accuracy Aberrations are mainly due to:

1. Redundant Input Genome Sequence.
2. Chromosomal Variations lead to false results

DISADVANTAGES:

- Novelty & Expensive
- Gene Editing is Controversial
- Analytical validity
- Structural variants
- prenatal tests-insufficient fetal DNA
- Chromosome Abnormality leads to false results

1.4.2 PROPOSED SYSTEM AND ITS ADVANTAGES

Proposed System deals with DNA Sequencing which includes parsing,

ADVANTAGES:

- Scientific information with medical outcomes is accurately depicted.
- Technical accuracy
- Single test in a lifetime
- Staying ahead of non-genetic healthcare providers

1.5 METHODOLOGY

1.5.1 BIOPYTHON

BioPython requires very less code and is used to provide simple, standard and extensive access to bioinformatics through python language.

The specific goals of the BioPython are listed below

- Provides microarray data type used in clustering.
- Supports structure data used for PDB parsing, representation, and analysis.
- Supports journal data used in Medline applications.
- Supports BioSQL database, which is widely used standard database amongst all bioinformatics projects.
- Providing standardized access to bioinformatics resources.
- High-quality, reusable modules, and scripts.
- Fast array manipulation that can be used in Cluster code, PDB, NaiveBayes and Markov Model.
- Genomic data analysis.

1.5.2 RELATED WORK: -

1.5.2.1 INCOMPATIBILITY AMONG PROTEIN AND DNA: -

```
In [12]: p_seq = Seq("EVRNAK", IUPAC.protein)
         d_seq = Seq('TACACT', IUPAC.unambiguous_dna)
         d_seq + my_seq

Out[12]: Seq('TACACTAGTACACTGGT', IUPACUnambiguousDNA())

In [13]: p_seq + my_seq

-----
TypeError                                 Traceback (most recent call last)
<ipython-input-13-ed678bbfe19e> in <module>
----> 1 p_seq + my_seq

C:\ProgramData\Anaconda3\lib\site-packages\Bio\Seq.py in __add__(self, other)
    309         raise TypeError(
    310             "Incompatible alphabets {0!r} and {1!r}".format(
--> 311                 self.alphabet, other.alphabet))
    312         # They should be the same sequence type (or one of them is generic)
    313         a = Alphabet._consensus_alphabet([self.alphabet, other.alphabet])

TypeError: Incompatible alphabets IUPACProtein() and IUPACUnambiguousDNA()
```

1.5.2.2 UNIQUE PROPERTY OF PROTEINS:-

```
In [16]: p_seq = Seq("EVRNAK", IUPAC.protein)
         p_seq.reverse_complement()

-----
ValueError                                 Traceback (most recent call last)
<ipython-input-16-2828b73a4b37> in <module>
      1 p_seq = Seq("EVRNAK", IUPAC.protein)
----> 2 p_seq.reverse_complement()

C:\ProgramData\Anaconda3\lib\site-packages\Bio\Seq.py in reverse_complement(self)
    946         """
    947         # Use -1 stride/step to reverse the complement
--> 948         return self.complement()[::-1]
    949
    950     def transcribe(self):

C:\ProgramData\Anaconda3\lib\site-packages\Bio\Seq.py in complement(self)
    896         base = Alphabet._get_base_alphabet(self.alphabet)
    897         if isinstance(base, Alphabet.ProteinAlphabet):
--> 898             raise ValueError("Proteins do not have complements!")
    899         if isinstance(base, Alphabet.DNAAlphabet):
    900             ttable = _dna_complement_table

ValueError: Proteins do not have complements!
```


1.5.2.4 REVERSE COMPLEMENT OF GENE SEQUENCES: -

```
dna_seq
dna_seq.lower()
from Bio.Seq import Seq
from Bio.Alphabet import IUPAC
my_seq = Seq("GATCGATGGGCCTATATAGGATCGAAAATCGC", IUPAC.unambiguous_dna)
my_seq
my_seq.complement()
my_seq.reverse_complement()
my_seq[::-1]
from Bio.Seq import Seq
from Bio.Alphabet import IUPAC
protein_seq = Seq("EVRNAK", IUPAC.protein)
#protein_seq.complement()
from Bio.Seq import Seq
from Bio.Alphabet import IUPAC
coding_dna = Seq("ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG", IUPAC.unambiguous_dna)
coding_dna
```

```
41]: Seq('ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG', IUPACUnambiguousDNA())
```

1.5.2.5 REVERSE COMPLEMENT ON FASTA: -

```
In [1]: from Bio import SeqIO
from io import StringIO
records = SeqIO.parse("DNA.fasta", "fasta")
out_handle = StringIO()
SeqIO.write(records, out_handle, "fasta")
fasta_data = out_handle.getvalue()
print(fasta_data)

def reverseComplement(s):
    complement = {'A': 'T', 'C': 'G', 'G': 'C', 'T': 'A'}
    t = ''
    for base in s:
        t = complement[base] + t
    return t

def match(s1, s2):
    if not len(s1) == len(s2):
        return False
    for i in range(0, len(s1)):
        if not s1[i] == s2[i]:
            return False
    return True

>NP_001273979.1 stabilizer of axonemal microtubules 1 isoform c [Homo sapiens]
ATGAAAACCAAATGCATTTGCGAACTGTGCAGCTGCGGCCGCCATCATTGCCCGCATCTG
CCGACCAAATTTATGATAAAACCGAAAAACCGTGCTGAGCGAATATACCGAAAAAC
TATCCGTTTTATCATAGCTATCTGCCGCGCGAAAGCTTTAAACCGCGCCGCGAATATCAG
AAAGGCCGATTCCGATGGAAGGCCTGACCAACGAGCAGC
>NP_001017978.1 kita-kyushu lung cancer antigen 1 [Homo sapiens]
ATGAACCTTTATCTGCTGTGGCGAGCAGCATTCTGTGCGCGCTGATTGTGTTTTGGAAA
TATCGCCGCTTTCAGCGCAACACCGGCGAAATGAGCAGCAACAGCACCGCGCTGGCGCTG
GTGCGCCGAGCAGCAGCGGCTGATTAACAGCAACACCGATAACAACCTGGCGGTGTAT
GATCTGAGCCGCGATATTCTGAACAACTTCCGCATAGCATTGCGCGCCAGAAACGCATT
CTGGTGAACTGAGCATGGTGGAACAACTGGTGAACATACCTGCTGAGC
AAAGGCTTTCGCGGCGCGAGCCCGCATCGCAAAAGCAC
>sp|Q5H943.1|KKLC1_HUMAN RecName: Full=Kita-kyushu lung cancer antigen 1; Short=KK-LC-1; AltName: I
83
```

1.5.2.6 I/O OPERATIONS DOCUMENTATION: -

```
In [9]: from Bio import SeqIO
help(SeqIO.convert)

Help on function convert in module Bio.SeqIO:

convert(in_file, in_format, out_file, out_format, alphabet=None)
    Convert between two sequence file formats, return number of records.

Arguments:
- in_file - an input handle or filename
- in_format - input file format, lower case string
- out_file - an output handle or filename
- out_format - output file format, lower case string
- alphabet - optional alphabet to assume

**NOTE** - If you provide an output filename, it will be opened which will
overwrite any existing file without warning. This may happen if even
the conversion is aborted (e.g. an invalid out_format name is given).

For example, going from a filename to a handle:

>>> from Bio import SeqIO
>>> try:
...     from StringIO import StringIO # Python 2
... except ImportError:
...     from io import StringIO # Python 3
...
>>> handle = StringIO("")
>>> SeqIO.convert("Quality/example.fastq", "fastq", handle, "fasta")
3
>>> print(handle.getvalue())
>EAS54_6_R1_2_1_413_324
CCCTTCTTGTCTTCAGCGTTTCTCC
```

1.5.2.7 TRANSLATION: -

2.DNA TO PROTEIN TRANSLATION

```
In [178]: #Translates an DNA sequence from a fasta file to an amino acid sequence.
from Bio.Seq import Seq
from Bio import SeqIO

def pad_seq(sequence):
    """ Pad sequence to multiple of 3 with N """
    remainder = len(sequence) % 3
    return sequence if remainder == 0 else sequence + Seq('N' * (3 - remainder))

seq_records = SeqIO.parse('DNA.fasta', 'fasta')

amino_acids1 = []
amino_acids2 = []
amino_acids3 = []

for record in seq_records:
    # starting from nucleotide 1
    amino_acids1.append(pad_seq(record).translate())
    print("FIRST")
    print(amino_acids1)
    # ...

    # starting from nucleotide 2
    record2 = record[1:]
    amino_acids2.append(pad_seq(record2).translate())
    print("SECOND")
    print(amino_acids2)
    # ...

    # starting from nucleotide 3
    record3 = record[2:]
    amino_acids3.append(pad_seq(record3).translate())
    print("THIRD")
```

1.6 ORGANISATION OF THE REPORT

- **Literature Survey (Chapter 2) –**
 - Includes a thorough study of the existing system's limitations that was overcome in the proposed system and also includes matter from papers/books and other sources which we referred for carrying out our project work.
- **Software Requirements (Chapter 3) -**
 - Includes a general description of the proposed system, hardware, software, and functional requirements and feasibility report.
- **Design (Chapter 4) -**
 - Includes details about the modeling of the proposed system. Data Flow Diagrams for structured design. Use Case Diagrams and Sequence Diagrams for Object-Oriented Design.
- **Modules Description (Chapter 5)-**
 - Includes details about the working of algorithms under each module and their relational importance to the main project
- **Implementation (Chapter 6) -**
 - contains description and details about pseudocode, program modules and sub modules etc.
- **Snapshots (Chapter 7)-**
 - contains the various results of the system developed.
- **Conclusion and Future Scope (Chapter 8)-**
 - includes concluding remarks of the project work, future enhancements, and limitations.
- **References –**
 - Includes list of names of the textbooks, papers, and websites referred for the project work.

Chapter 2

LITERATURE SURVEY

Genomics research often requires gathering data about genomic assembly, mutation, visualization, and parsing

DNA sequence is representation of genetic code contained with an organism. In this paper, we begin by presenting background information on the project and focus on describing datasets, indexing, matching, alignment & their differences between mutated and DNA sequences. String Searching is carried out by two prominent algorithms.

One is the K-mer Indexing where k-mers are produced based on the pattern required and are typically used during sequence assembly, but can also be used in the sequence alignment.

K-mer Indexing can also be used as a first stage analysis before alignment. It is used to find DNA barcoding of species, de novo sequence assembly, detect genome mis-assembly, and estimate the genome size

Another Algorithm is Boyer-Moore [BM] Algorithm which is the most efficient Algorithm for general matching, and its primary function is to gain more information by matching the pattern from right to left, which makes faster run-time.

The next Part of Sequencing is through Pattern Matching. There are mainly two types of Pattern Matching. One of them is Approximate Pattern matching [4]. It is primarily useful in matching extensive DNA sequences data as there is a need to find occurrences of a pattern matching inside large text or find matching patterns.

Distance-based Hamming [4] method can accept characters mismatches in the arrangement, gives different performance results depending upon several compared patterns.

Another type of Pattern Matching is Exact Pattern matching [3], searches all occurrences of a pattern in the text. The Exact-Pattern Matching or Naive Pattern Matching is the most efficient matching algorithm in terms of computational efficiency.

DNA Sequencing Alignment is a representation of similarity between 2 or more sections of the genetic code. There are mainly two types of Sequence Alignment namely, Global and Local Sequence Alignment.

Needleman-Wunsch Algorithm [2] is one of the Global Alignment Technique which is primarily based on Dynamic Programming, to find Alignment between two Sequences which are similar in length as well as similar across the width.

The Smith-Waterman algorithm is part of the Local Alignment Technique that employs dynamic programming to determine optimal local arrangements for sequence similarity between the pattern sequence and text sequence.

Finally, we provide mutational differences between two different Gene Sequences to determine the mutated cancer gene. We compare the efficiency of each algorithm in each module to ensure accuracy in finding mutations. We increase the efficiency of finding mutations [7] with primary techniques namely, indexing, pattern matching, sequence alignments and comparison of FASTA files to find out variation, leading to the increase in accuracy of finding cancer genome mutation.

Chapter 3

SYSTEM REQUIREMENTS

3.1 Hardware Requirements:

- Processor : Pentium IV 2.4 GHz
- RAM : 2 GB
- Hard Disk : 160 GB

3.2 Software Requirements:

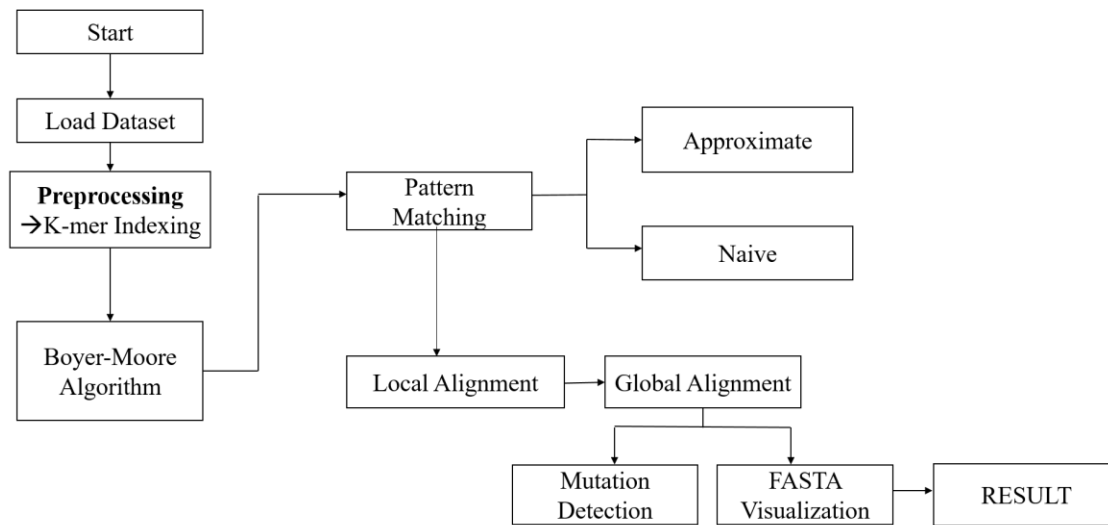
- **Language** : PYTHON
- **IDE** : Jupyter Notebook
- **Tools (Libraries)** : BioPython

Chapter – 4

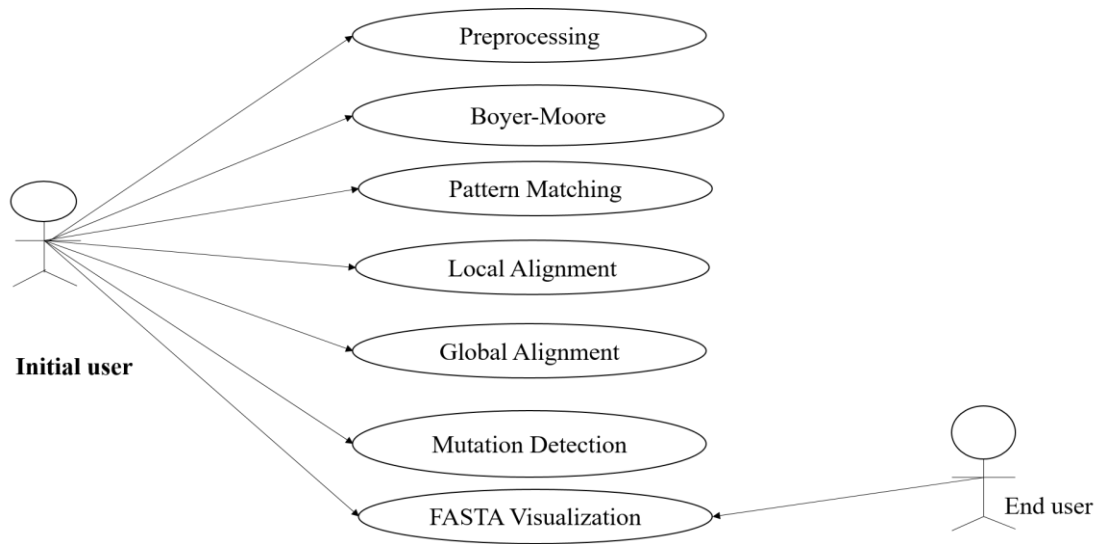
DESIGN

This chapter specifies the design description of “GENOME ANALYSIS”.

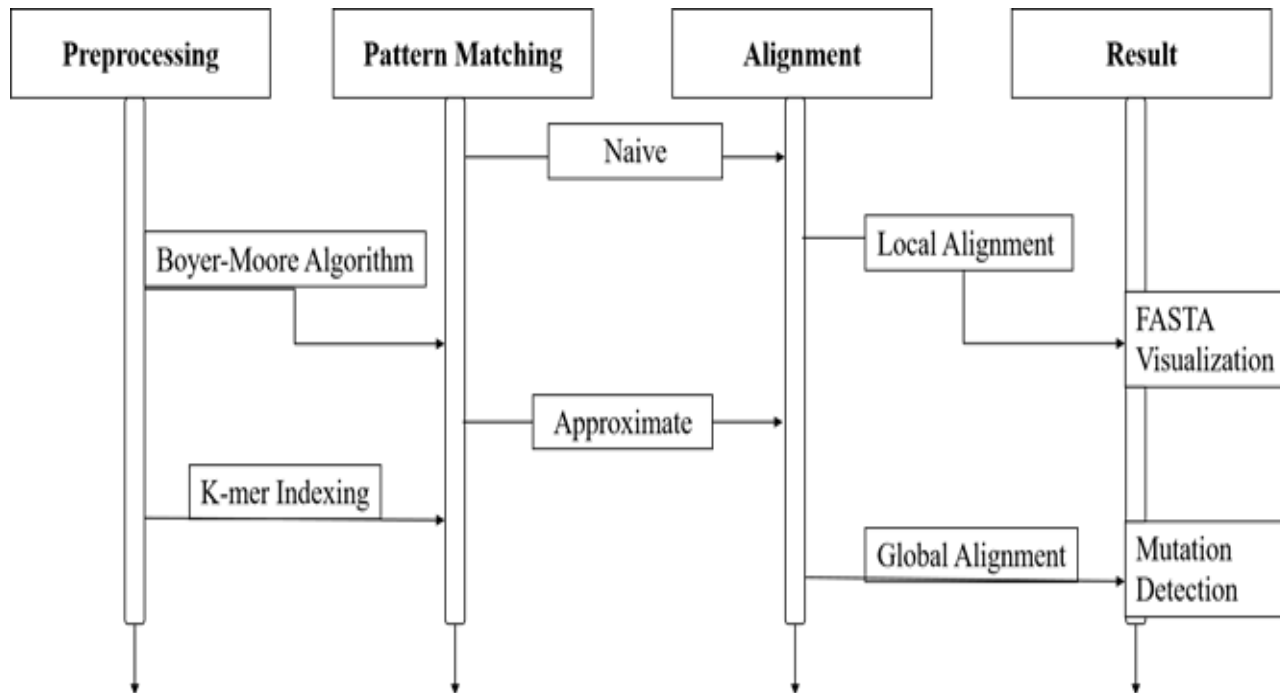
4.1 Data Flow diagrams



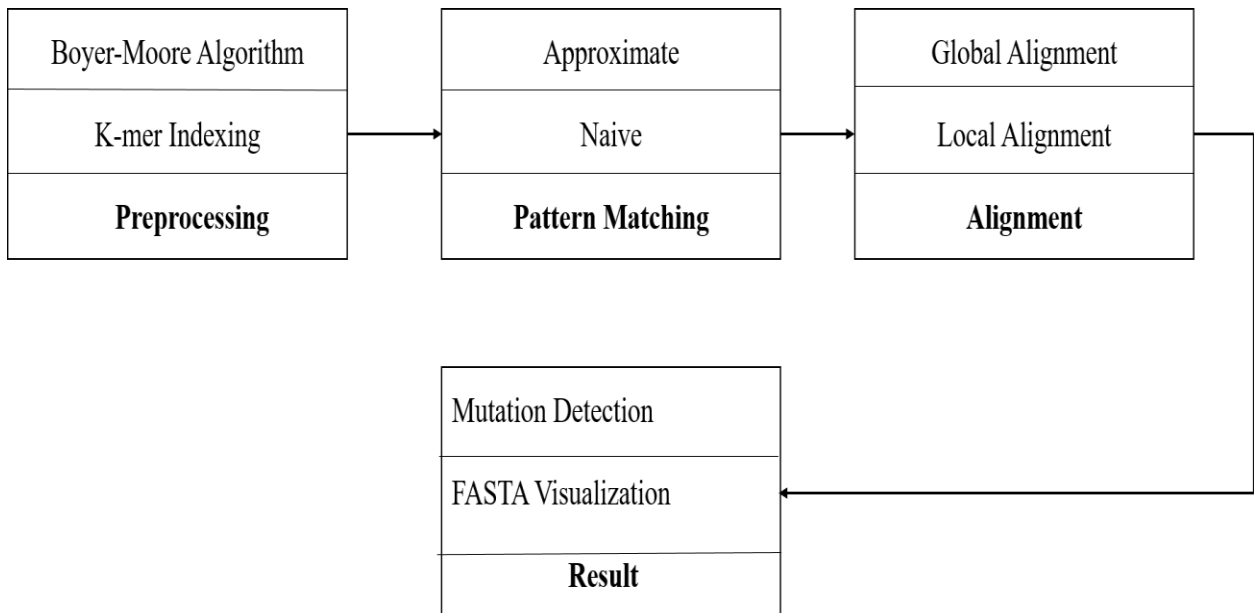
4.2 Use case Diagram.



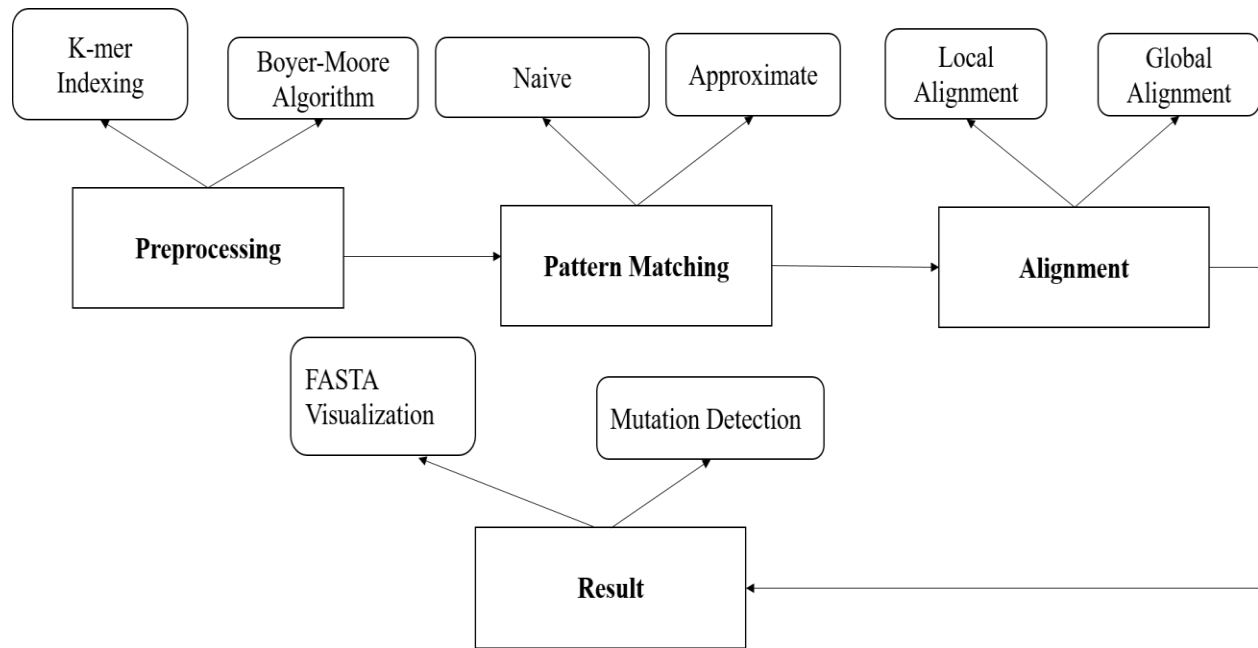
4.3 SEQUENCE DIAGRAM



4.4CLASS DIAGRAM



4.5ER DIAGRAM



Chapter – 5

MODULES DESCRIPTION

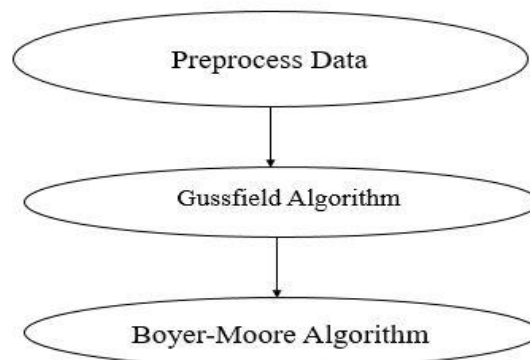
- Data Preprocessing
- Pattern Matching
- Sequence Alignment
- Mutation Detection
- FASTA-Visualization

5.1. DATA PREPROCESSING

In the preprocessing stage, we utilize two algorithms K-mer Indexing Algorithm and Boyer-Moyer Algorithm.

We use the term K-mer to refer to a substring of length k. Boyer-Moyer Algorithm for finding the good suffix and the bad character shift.

▪ FLOWCHART

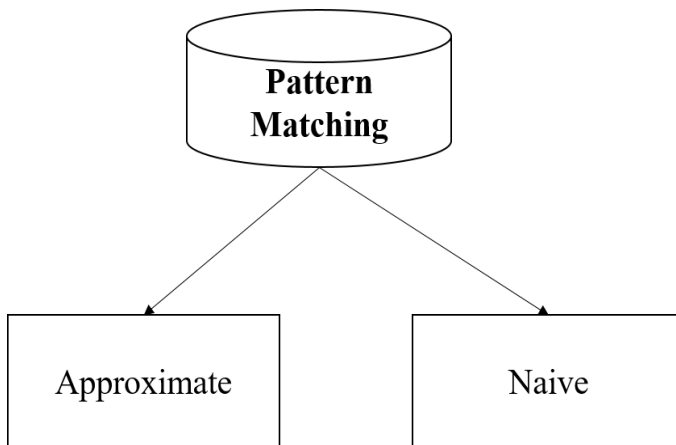


5.2. PATTERN MATCHING

In the Pattern Matching phase, we use Naïve String Search and approximate String matching to find the match or mismatch of the pattern.

We use Naïve String Search to check if one string (pattern string) matches with the master string (text string) whereas we use Approximate String matching to find the strings that match the pattern.

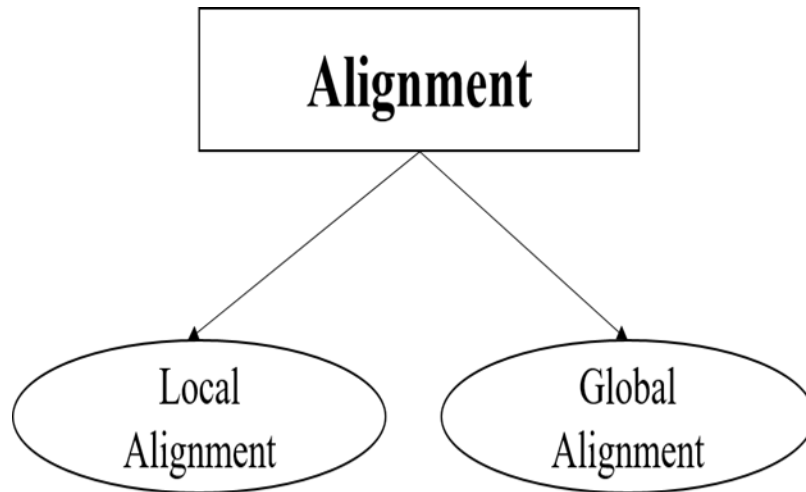
▪ FLOWCHART



5.3. SEQUENCE ALIGNMENT

The Sequence Alignment is done by Local Alignment and Global Alignment.

▪ **FLOW CHART**



5.4. MUTATION DETECTION

The mutations of the samples are found by parsing the normal and affected samples.

Zip command is used to compare the mismatch and its sum function is used to find the mutations of the samples.

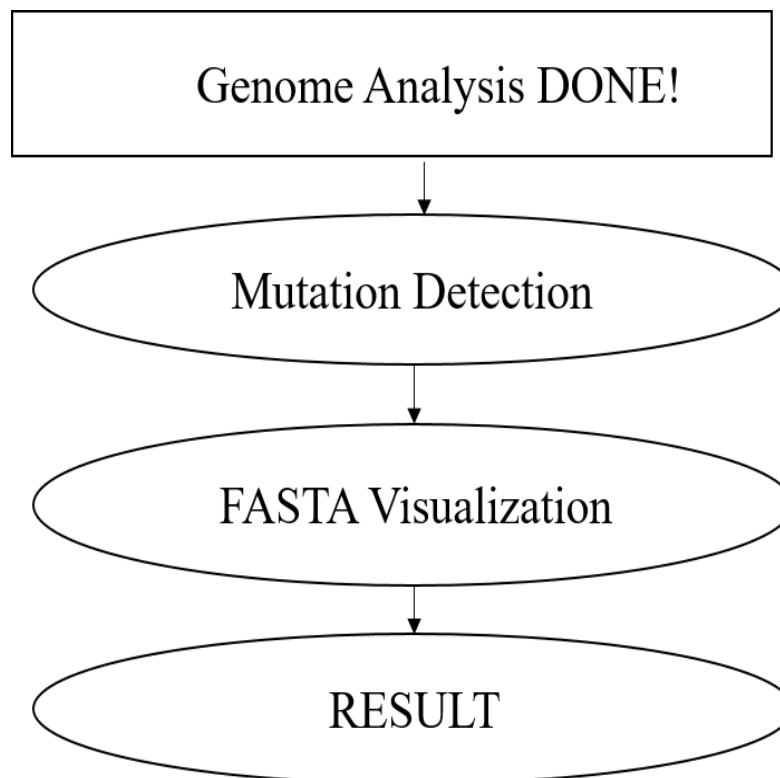
5.5. FASTA-VISUALIZATION

We use **Fluent DNA** for Visualizing the DNA sample of a normal and mutated sample.

Adenine(A), Cytosine(C), Guanine(G), Thymine(T) is assigned with colors to differentiate and

N represents the mutation of the sample.

▪ FLOWCHART



Chapter - 6

IMPLEMENTATION

6.1 DATASET: -

```
jupyter DNA.fasta 04/09/2019
```

File	Edit	View	Language
1	>NP_001273979.1 stabilizer of axonemal microtubules 1 isoform c [Homo sapiens]		
2			
3	ATGAAAACCAAATGCATTTGCGAACTGTGCAGCTGCGGCCGCCATCATTGCCCGCATCTG		
4	CCGACCAAATTTATGATAAAACCGAAAAACCGTGCCTGCTGAGCGAATATACCGAAAAAC		
5	TATCCGTTTTATCATAGCTATCTGCCGCGCGAAAGCTTTAAACCGCGCCGCAATATCAG		
6	AAAGGCCCGATTCCGATGGAAGGCCTGACCACCAGCAGC		
7			
8			
9			
10	>NP_001017978.1 kita-kyushu lung cancer antigen 1 [Homo sapiens]		
11			
12	ATGAACCTTTTATCTGCTGCTGGCGAGCAGCATTCTGTGCGCGCTGATTGTGTTTTGGAAA		
13	TATCGCCGCTTTTACGCGCAACACCGCGGAAATGAGCAGCAACAGCACCGCGCTGGCGCTG		
14	GTGCGCCCGAGCAGCAGCGGCTTGATTAAACAGCAACACCGATAACAACCTGGCGGTGTAT		
15	GATCTGAGCCGCGATATTCTGAACAACCTTTCCGCATAGCATTGCGCGCCAGAAACGCATT		
16	CTGGTGAACCTGAGCATGGTGGAAAAACAACTGGTGGAACTGGAACATACCCTGCTGAGC		
17	AAAGGCTTTTCGCGGCGCGAGCCCGCATCGCAAAAGCACC		
18			
19			
20	>sp Q5H943.1 KKLC1_HUMAN RecName: Full=Kita-kyushu lung cancer antigen 1; Short=KK-LC-1;		
21			
22	ATGAACCTTTTATCTGCTGCTGGCGAGCAGCATTCTGTGCGCGCTGATTGTGTTTTGGAAA		
23	TATCGCCGCTTTTACGCGCAACACCGCGGAAATGAGCAGCAACAGCACCGCGCTGGCGCTG		
24	GTGCGCCCGAGCAGCAGCGGCTTGATTAAACAGCAACACCGATAACAACCTGGCGGTGTAT		
25	GATCTGAGCCGCGATATTCTGAACAACCTTTCCGCATAGCATTGCGCGCCAGAAACGCATT		
26	CTGGTGAACCTGAGCATGGTGGAAAAACAACTGGTGGAACTGGAACATACCCTGCTGAGC		
27	AAAGGCTTTTCGCGGCGCGAGCCCGCATCGCAAAAGCACC		
28			

```
jupyter Mutated.fasta 04/09/2019
```

File	Edit	View	Language
1	>NP_001273979.1 stabilizer of axonemal microtubules 1 isoform c [Homo sapiens]		
2			
3	ATGAARACNAARTGYATHGTGYGARYTNTGYWSNTGYGGNMGNCAYCAYTYCCNCAYYTN		
4	CCNACNAARATHAYGAYARACNGARAARCCNTGYTYNTNWSNGARTAYACNGARAAY		
5	TAYCCNTTYTAYCAYWSNTAYYTNCCNMGNGARWSNTTYAARCCNMGMNGARTAYCAR		
6	AARGNCCNATHCCNATGGARGGNYTNACNACNWSNWSN		
7			
8			
9			
10	>NP_001017978.1 kita-kyushu lung cancer antigen 1 [Homo sapiens]		
11			
12	ATGAAYTTYTAYYTNNTNYTNGCNWSNWSNATHYTNTGYGCNYTNATHGTNTTYTGGAAR		
13	TAYMGNMGNTTYCARMGNAAYACNGNGARATGWSNWSNAAYWSNACNGCNYTNGCNYTN		
14	GTNMGNCNWSNWSNWSNNGGNYTNATHAAYWSNAAYACNGAYAAAYTYTNGCNGTNTAY		
15	GAYYTNWSNMGNGAYATHYTNAAYAAAYTYCCNCAYWSNATHGCMGNCARAARMGNATH		
16	YTNGTNAAYYTNWSNATGGTNGARAAYAAARYTNGTNGARYTNGARCAYACNYTNYTNWSN		
17	AARGNTTYMNGGNGCNWSNCCNCAYMGNAARWSNACN		
18			
19			
20	>sp Q5H943.1 KKLC1_HUMAN RecName: Full=Kita-kyushu lung cancer antigen 1; Short=KK-LC-1;		
21			
22	ATGAAYTTYTAYYTNNTNYTNGCNWSNWSNATHYTNTGYGCNYTNATHGTNTTYTGGAAR		
23	TAYMGNMGNTTYCARMGNAAYACNGNGARATGWSNWSNAAYWSNACNGCNYTNGCNYTN		
24	GTNMGNCNWSNWSNWSNNGGNYTNATHAAYWSNAAYACNGAYAAAYTYTNGCNGTNTAY		
25	GAYYTNWSNMGNGAYATHYTNAAYAAAYTYCCNCAYWSNATHGCMGNCARAARMGNATH		
26	YTNGTNAAYYTNWSNATGGTNGARAAYAAARYTNGTNGARYTNGARCAYACNYTNYTNWSN		
27	AARGNTTYMNGGNGCNWSNCCNCAYMGNAARWSNACN		
28			

6.2 VISUALIZATION TOOLS

A. FluentDNA

This application creates visualizations of FASTA formatted DNA nucleotide data.

FluentDNA generates a Deep Zoom Image visualization similar to Google Maps for FASTA files.

Changes in nucleotide usage make individual genome elements visible even without an annotation. Add your annotation files to see how they align with the sequence features.

REQUIREMENTS: -

- pip==9.0.1
- setuptools==38.2.4
- wheel==0.24.0
- cx_Freeze==5.1.1
- Pillow>=3.2.0
- six==1.10.0
- psutil==4.3.1
- blist==1.3.6
- natsort==5.1.1
- numpy==1.13.3

Chapter - 7

SNAPSHOTS

7.1PREPROCESSING

```
In [121]: def boyer_moore(p, p_bm, t):
          """ Do Boyer-Moore matching """
          i = 0
          occurrences = []
          while i < len(t) - len(p) + 1:
              shift = 1
              mismatched = False
              for j in range(len(p)-1, -1, -1):
                  if p[j] != t[i+j]:
                      skip_bc = p_bm.bad_character_rule(j, t[i+j])
                      skip_gs = p_bm.good_suffix_rule(j)
                      shift = max(shift, skip_bc, skip_gs)
                      mismatched = True
                      break
              if not mismatched:
                  occurrences.append(i)
                  skip_gs = p_bm.match_skip()
                  shift = max(shift, skip_gs)
              i += shift
          return occurrences

In [122]: t = 'GCTAGCTCTACGAGTCTA'
          p = 'TCTA'
          p_bm = BoyerMoore(p, alphabet='ACGT')

In [123]: boyer_moore(p, p_bm, t)

Out[123]: [6, 14]
```

7.2 PATTERN MATCHING

```
In [131]: def approximate_match(p, t, n):
    segment_length = int(round(len(p) / (n+1)))
    all_matches = set()
    for i in range(n+1):
        start = i*segment_length
        end = min((i+1)*segment_length, len(p))
        p_bm = BoyerMoore(p[start:end], alphabet='ACGT')
        matches = boyer_moore(p[start:end], p_bm, t)
        # Extend matching segments to see if whole p matches
        for m in matches:
            if m < start or m-start+len(p) > len(t):
                continue
            mismatches = 0
            for j in range(0, start):
                if not p[j] == t[m-start+j]:
                    mismatches += 1
                    if mismatches > n:
                        break
            for j in range(end, len(p)):
                if not p[j] == t[m-start+j]:
                    mismatches += 1
                    if mismatches > n:
                        break
            if mismatches <= n:
                all_matches.add(m - start)
    return list(all_matches)
```

```
In [132]: p = 'AACTTG'
    t = 'CACTTAATTTG'
    print(approximate_match(p, t, 2))

[0, 5]
```

7.3 SEQUENCE ALIGNMENT

```

----- local alignment -----
< case 1 >
score: 17
T-CACACTG
TGCACAC-G

```

In [168]: data

Out[168]:

	-	A	A	C	T	C	A	C	A	C	T	G	T	T
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T	0	0	0	0	3	1	0	0	0	0	3	1	3	3
G	0	0	0	0	1	2	0	0	0	0	1	6	4	2
C	0	0	0	3	1	4	2	3	1	3	1	4	5	3
A	0	3	3	1	2	2	7	5	6	4	2	2	3	4
C	0	1	2	6	4	5	5	10	8	9	7	5	3	2
A	0	3	4	4	5	3	8	8	13	11	9	7	5	3
C	0	1	2	7	5	8	6	11	11	16	14	12	10	8
G	0	0	0	5	6	6	7	9	10	14	15	17	15	13

```

----- global alignment -----
< case 1 >
score: 14
AGTCCT-A
AG-CCTTA
< case 2 >
score: 14
AGTCC-TA
AG-CCTTA

```

In [171]: data

Out[171]:

	-	A	G	T	C	C	T	A
-	0	-2	-4	-6	-8	-10	-12	-14
A	-2	3	1	-1	-3	-5	-7	-9
G	-4	1	6	4	2	0	-2	-4
C	-6	-1	4	5	7	5	3	1
C	-8	-3	2	3	8	10	8	6
T	-10	-5	0	5	6	8	13	11
T	-12	-7	-2	3	4	6	11	12
A	-14	-9	-4	1	2	4	9	14

7.4 MUTATION DETECTION





MUTATION-DETECTION:

```
In [175]: from Bio import SeqIO

normal_samples = SeqIO.parse("DNA.fasta", "fasta")
treated_samples = SeqIO.parse("Mutated.fasta", "fasta")

for normal, treated in zip(normal_samples, treated_samples):
    if normal.id == treated.id:
        mutations = sum(1 for n, t in zip(str(normal.seq), str(treated.seq)) if n != t)
        print(f"Found {mutations} mutation(s) for id {normal.id}")

Found 93 mutation(s) for id NP_001273979.1
Found 165 mutation(s) for id NP_001017978.1
Found 165 mutation(s) for id sp|Q5H943.1|KKLC1_HUMAN
Found 134 mutation(s) for id AAC35497.1
Found 174 mutation(s) for id XP_002699503.1
Found 161 mutation(s) for id sp|Q4R717.1|KKLC1_MACFA
Found 138 mutation(s) for id AAN12271.1
Found 161 mutation(s) for id NP_001270680.1
Found 400 mutation(s) for id Sp|P63244.3|Rack1_Human
Found 174 mutation(s) for id XP_027389897.1
Found 169 mutation(s) for id XP_007647427.1
Found 187 mutation(s) for id XP_014918346.1
Found 174 mutation(s) for id XP_0026997503.1
Found 89 mutation(s) for id XP_002699603.1
Found 66 mutation(s) for id XP_004699503.1
```

Sp P63244.3 Rack1
Human Recname: F1

XP_027389897.1 k
a-kyushu lung can

XP_007647427.1 k
a-kyushu lung can

XP_014918346.1 k
a-kyushu lung can


DNA.fasta

```

sp P63244.3 Rack-
Human RecName: F1
xp 027389897.1 k-
a-kyushu lung can
xp 007647427.1 k-
a-kyushu lung can
xp 014918346.1 k-
a-kyushu lung can
xp 0026997503.1 l
ta-kyushu lung c:

```

Mutated.fasta

Genome Analysis of any Organism can be detected with the help of the program at a very fast rate and under high accuracy. Information provided by the genomic sequencing can be a value to the future generation for mutation detection. The genome sequencing compares the genetic DNA with the Normal DNA to detect the anomalies in the sequencing pattern and deduces the mutation if found through Cascade Testing.

Cascade testing with other family members provides crucial evidence required for Genome Sequencing. The DNA can be analyzed using FASTA Visualization format helping to produce accurate results.

Chapter - 8

ALGORITHMS

8.1. K-MER INDEXING ALGORITHM

The first step in any k-mer analysis is the generation of a profile, which is constructed by the indexing algorithm. It's generally used for sequence alignment.

K-mer refers to all the potential substrings of a string of length k.

The efficiency of the algorithm is enhanced by encoding the DNA string in binary. Following, the binary encoded k-mers are used as the index of a count table.

This can be achieved by the concatenation of the binary code for each nucleotide in a given DNA string.

This procedure eliminates the need to store the actual k-mer sequences since they can be retrieved from decoding the offset in the count table.

The binary code for each nucleotide is chosen in such a way that the complement of the nucleotide can be intended using the binary NOT operator.

The indexing algorithm returns a profile that holds observed counts for all possible substrings of length k that can be stored for other analyses.

8.2 BOYER MOORE ALGORITHM

Boyer Moore Algorithm actually learns from character comparisons to skip pointless alignments

It performs both alignment in left-to-right order and character comparisons in right-to-left order.

The complexity of the preprocessing usually depends on the size of the alphabet of the string.

The working of character comparisons during mismatches is Alignments are skipped until mismatch becomes a match, or Pattern P moves past mismatched character.

Alignments are performed only during a character match among Pattern P and Text T.

8.3. SMITH-WATERMAN ALGORITHM

The Algorithm works as Local Alignment where matches, mismatches, and gaps are considered as binary. The matches are considered as Positive 1, mismatches and gaps are considered as Zero.

The Upward, Side-ward, and Diagonal Values are added accordingly with respect to their directions.

The Maximum value among the three values are assigned in to-be-filled blocks and the process continues till the end of the sequence in Score Matrix.

The Final Step is Traceback Method, where the maximum values are traced back from end to the begin of the sequence in Score Matrix.

The Gaps between the Traceback are considered as Mutated Nucleotide among the considered two sequences.

8.3. NEEDLEMAN WUNSCH ALGORITHM

The Algorithm works as Local Alignment where matches, mismatches, and gaps are considered as integer values. The matches are considered as Positive 1, mismatches as Negative 1 and gaps are considered as -2.

The Upward, Side-ward, and Diagonal Values are added accordingly with respect to their directions.

The Maximum value among the three values are assigned in to-be-filled blocks and the process continues till the end of the sequence in Score Matrix.

The Final Step is Traceback Method, where the maximum values are traced back from end to the begin of the sequence in Score Matrix.

The Gaps between the Traceback are considered as Mutated Nucleotide among the considered two sequences

Among the alignment algorithms, This Algorithm is best for finding mutations as it is a Global Alignment.

Chapter - 9

CONCLUSION AND FUTURE ENHANCEMENT

9.1 CONCLUSION:

- To obtain information of genetics value for next generations is beneficiary
- Genomic sequencing can provide information on mutations, sequence aberrations.

9.2 FUTURE ENHANCEMENT

- Genome Assembly and Sequencing .is emerging technology and incorporates new approaches frequently
- We will be researching and working further on Multiple Sequence Alignments and Chromosomal Aberrations.

REFERENCES

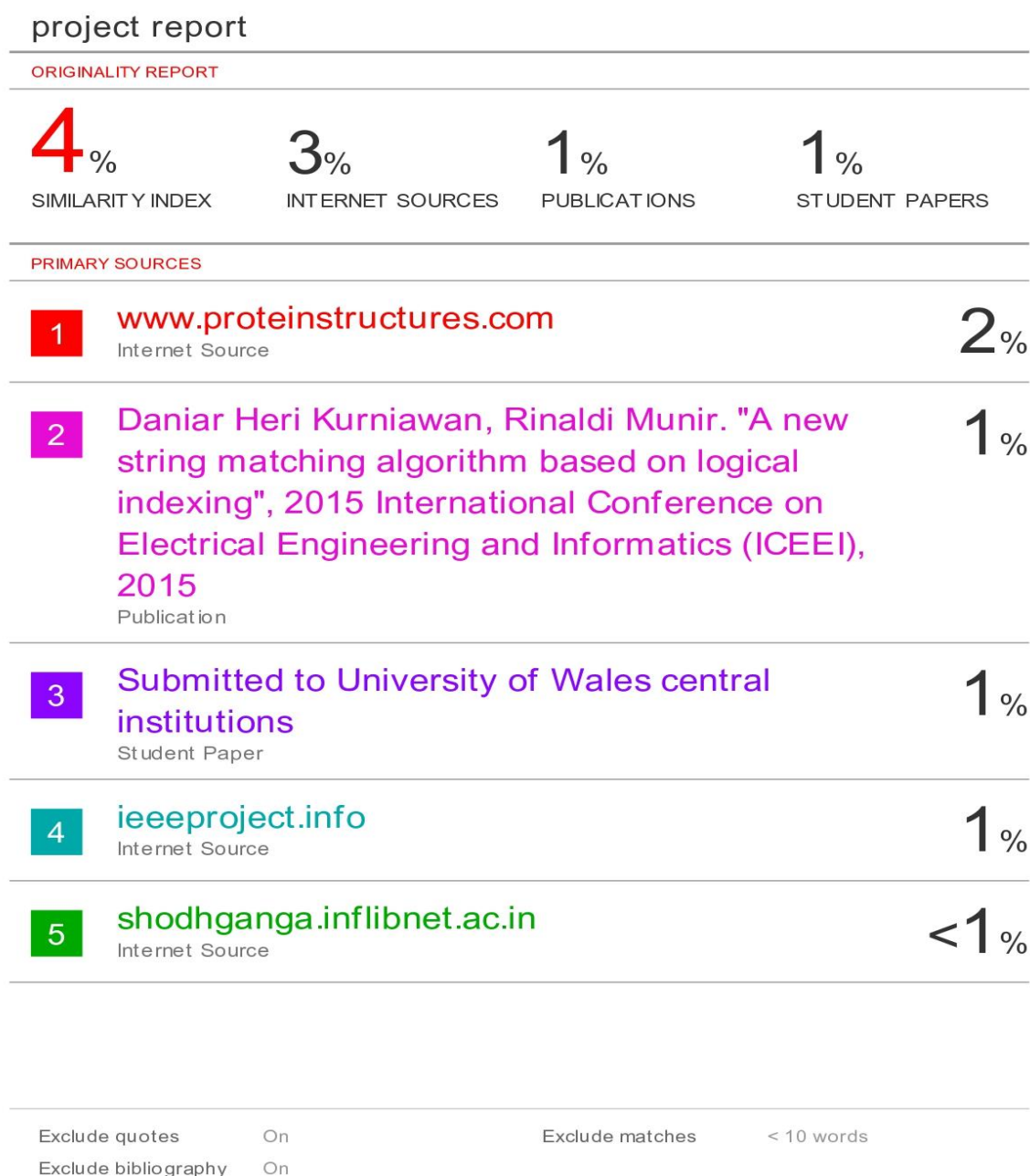
- [1] P. Fournier-Viger, J. C. W. Lin, R. U. Kiran, Y. S. Koh, and R. Thomas, "A survey of sequential pattern mining," *Data Science and Pattern Recognition*, vol. 1(1), pp. 54-77, 2017.
- [2.] M. S. Chen, J. S. Park, and P. S. Yu, "Efficient data mining for path traversal patterns," *IEEE Transactions on Knowledge and Data Engineering*, vol. 10, no. 2, pp. 209-221, 1998.
- [3] C. Creighton and S. Hanash, "Mining gene expression databases for association rules," *Bioinformatics*, vol. 19, no. 1, pp. 79-86, 2003.
- [4] Pritchard L, White JA, Birch PR, and Toth IK (2006) Genome Diagram: a Python package for the visualization of large-scale genomic data. *Bioinformatics*, 22, 616-617
- [5] Cock PJ, Fields CJ, Goto N, Heuer ML and Rice PM (2009) The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Res.*, 38, 1767-1771
- [6]. Baeza-Yates, R.; Navarro, G. (June 1996). "A faster algorithm for approximate string matching." In Dan Hirschberg; Gene Myers (eds.). *Combinatorial Pattern Matching (CPM'96)*, LNCS 1075. Irvine, CA. Pp. 1–23.
- [7]. Smith, Temple F. & Waterman, Michael S. (1981). "Identification of Common Molecular Subsequences." *Journal of Molecular Biology*. 147 (1): 195–197. CiteSeerX 10.1.1.63.2897. doi:10.1016/0022-2836(81)90087-5. PMID 7265238.
- [8]. Needleman, Saul B. & Wunsch, Christian D. (1970). "A general method applicable to the search for similarities in the amino acid sequence of two proteins." *Journal of Molecular Biology*. 48 (3): 443–53. doi:10.1016/0022-2836(70)90057-4. PMID 5420325

BIBLIOGRAPHY

- **DATA MINING FOR GENOMICS AND PROTEOMICS** (Analysis of Gene and Protein Expression Data) by DARIUS M. DZIUDA.
- <http://prody.csb.pitt.edu/> Protein Dynamics & Sequence Analysis
- <https://biopython.org/> BioPython Documentation
- **DATASET OBTAINED FROM THIS WEBSITE** → <https://www.ncbi.nlm.nih.gov/-->
- <https://www.bioinformatics.org/> Bioinformatics

• APPENDIX

I. PLAGIARISM REPORT:-



II. PAPER PUBLICATION:-



1. **INDEXING**: -ELSEVIER'S EL COMPENDEX
2. The Paper will be published and will be available in **IEEE Xplore Digital Library**