Dr. Babasaheb Ambedkar Marathwada University, Aurangabad

MAHATMA GHANDHI MISSION

Institute of Biosciences and Technology

N-6, CIDCO, Aurangabad.

YEAR: 2019-2020.

PROJECT REPORT

ON

"Design and Development of Human Genome Analysis Tool "

SUBMITTED BY

Mr. Saurabh Sunilrao Bobade (BBIS172007)

UNDER THE GUIDENCE OF

Ms. Archana Panche

“Design and Development of Human Genome Analysis tool”

A Thesis

submitted to

Mahatma Gandhi Mission’s

Institute of Biosciences and Technology,

Aurangabad.

FOR AWARD OF THE DEGREE OF BACHELOR OF SCIENCE

By

MR. SAURABH SUNILAO BOBADE

UNDER THE GUIDANCE OF

MS. ARCHANA PANCHE

CANDIDATES DECLARATION

I wish to state that the work embodied in this project titled, **"Design and Development of Human Genome Analysis tool"** forms my own contribution to the research work carried out under guidance of **Ms. Archana Panche** at the MGM's Institute of Biosciences and Technology, affiliated to **Dr.Babasaheb Ambedkar Marathwada University, Aurangabad**. This work has not been submitted for any other degree of this or any other University. Wherever references have been made to previous works of others, it has been clearly indicated as such and included in the References.

Signature of the Candidate

1. **Saurabh Sunilrao Bobade** (BBIS172007)

Certified by

Guide name: **Ms. Archana Panche**

Affiliation: **MGM'S Institute of Biosciences & Technology**

Date:

CERTIFICATE

The project work presented in this project has been carried out by **Saurabh Sunilrao Bobade (BBIS1720)** under my guidance and have completed as per the requirements of Dr.Babasaheb Ambedkar Marathwada University, Aurangabad in partial fulfilment of the degree of Bachelor of Science **(Bioinformatics)** for the academic year 2019-20.This constitutes their bonafide work. The project work done is original and has not been submitted for any other degree for this or other University. Further that they were regular students and has/have worked under my guidance at the Department of Biotechnology Engineering until the submission of the thesis to the University.

Date:
Place:

Guide	**Examiner**	**Director**
Ms. Archana Panche		**Dr. Sanjay Harke**
Project Head		**Academic co-ordinator**

ACKNOWLEDGEMENT

I would first like to thank my thesis Guide **Ms. Archana Panche** from the department of Bioinformatics at MGM institute of Bioscience and Technology. The door of Professor Office was always open whenever I ran into a trouble spot or had a question about my research or writing .She consistently allowed this paper to be my own work, but steered me in right the direction whenever she thought I needed it.

I would also like to thank the expert and guide who are involved in validation survey for this research project . Research Guide,**Ms. Archana Panche**. Without their passionate participation and input, the validation survey could not have been successfully conducted. Specially thank to **Ms. Mansi Dhawalekar** she always support to the related project

Finally, I must express my very profound gratitude to my parents and to for providing me with unfailing support and continues encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

1. Mr. Saurabh Sunilrao Bobade (BBIS172007)

INDEX

List of Abbreviation:

Sr. No	Abbreviation	Full form
1.	HGP	Human Gen0me Project
2.	DNA	Deoxyribonucleic acid
3.	ORF	Open Reading Frames
4.	BLAST	Basic Local Alignment Search Tool
5.	GUI	Graphical User Interface
6.	PHP	Personal Home Page
7.	html	Hyper Text Markup Language
8.	SQL	Structure Query Language

List of figures:

ABSTRACT

"***A picture is worth a thousand words***"

A tool to analyse the genome is developed with the help of two programming languages i. Java and ii. Python . This tool gives the alignment for the sequence and the alignment score by the Smith and Waterman’s algorithm . This tool is implementation of Smith and Waterman’s algorithm into computer language programs . This tool helps in the recognition of the presence of query sequence in the reference sequence .

Chapter 1: Introduction

1.Introduction:

The Human Genome Project [HGP], which was launched in 1990 and declared on 14th April 2003 . This project determines the sequence of the 3 billion chemical base pairs that makes up human DNA [7]. Different marker genes can be find out in the Human Genome with the help of this tool . With the help of three programming languages which are:

- JAVA
- PYTHON

The tool can be developed for the analysis of the genome sequence in the Android operating system. Java is a general purpose programming language, used for a back end programming a android apps. Python is a interpreted, high level language. Python can be also use for the development of algorithm in an application or tool. The tool will be useful to finding out the scores of alignment, different gene sequences , ORFs , etc.

1.1 The Human Genome Project:

The Human Genome Project [HGP], which was launched in 1990 and declared on 14th April 2003 . The idea of the HGP was first publicly advocated by Renato Dulbecco in an article published in 1984 .[10] This project determine the sequence of the 3 billion chemical base pairs that makes up human DNA . The goal for this project was :

- to identify all the approximately 20,500 genes in human DNA
- to store this information in Database
- to improve tools for data analysis.

The project exemplifies the power, necessity and success of large, integrated, cross disciplinary efforts - so-called ‘big science’ - directed towards complex major objectives.[5]

1.2 Marker Genes :

Molecular markers serve to assign individual samples to specific groups. Such markers should be easily identified and have a high discrimination power, being highly conserved within groups while showing sufficient variability between the groups that are to be distinguished.[3]

1.3 Smith and Waterman's Algorithm :

The **Smith–Waterman algorithm** performs local sequence alignment; that is, for determining similar regions between two strings of nucleic acid sequences or protein sequences. The algorithm was first proposed by Temple F. Smith and Michael S. Waterman in 1981.

Let $A = a_1,a_2,\ldots a_n$ and $B = b_1,b_2,\ldots b_m$ be the sequences to be aligned, where n and m are the lengths of A and B respectively.

1. Determine the substitution matrix and the gap penalty scheme.
   - s(a,b) - Similarity score of the elements that constituted the two sequences
   - W_k - The penalty of a gap that has length k
2. Construct a scoring matrix H and initialize its first row and first column. The size of the scoring matrix is (n+1)*(m+1). The matrix uses 0-based indexing.

   Hk0 = H0l = for 0<k<n and 0<l<m
3. Fill the scoring matrix using the equation below.

$$H_{ij} = \max \left\{ \begin{array}{l} H_{i-1,j-1}+s(a_i+b_j), \\ \max_{k>1} \{H_{i-k,j}+W_k\}, \\ \mathrm{Max}_{l>1} \{H_{i,j-1}-W_l\}, \\ 0 \end{array} \right.$$

where

$H_{i-1,j-1}+s(a_i+b_j)$ is the score of aligning a_i and b_j,

$H_{i-k,j}+W_k$ is the score if a_i is at the end of a gap of length k,

$H_{i,j-1}-W_l$ is the score if b_j is at the end of a gap of length l,

0 means there is no similarity up to a_j and b_j.[7]

4. Traceback. Starting at the highest score in the scoring matrix H and ending at a matrix cell that has a score of 0, traceback based on the source of each score recursively to generate the best local alignment.

1.4 Java :

Java was originally developed by James Gosling at Sun Microsystems and released in 1995 as a core component of Sun Microsystems's Java platform , There were five primary goals in the creation of the Java language:

1. It must be simple, object-oriented, and familiar.
2. It must be robust and secure.
3. It must be architecture-neutral and portable.
4. It must execute with high performance.
5. It must be interpreted, threaded, and dynamic.[6]

1.5 Python :

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL), capable of exception handling and interfacing with the Amoeba operating system. Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by meta programming and meta objects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming . Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution .[4]

Chapter 2: Review of Literature

.

2.Review of Literature:

2.1 Tools or Software for genome analysis :

- Needleman and Wunsch proposed a heuristic homology algorithm for sequence alignment, also referred to as the Needleman–Wunsch algorithm in 1970 . It is a global alignment algorithm that requires calculation steps (and are the lengths of the two sequences being aligned). It uses the iterative calculation of a matrix for the purpose of showing global alignment.[9]

- Lipman and Pearson develop FASTA in 1985 , which is DNA and Protein sequence alignment software package . Its legacy is the FASTA format which is now ubiquitous in bioinformatics.[10]

- Altschul , Gish , Miller , Myers and Lipman develop a tool called BLAST (Basic Local Alignment Search Tool) in 1990 . BLAST is an algorithm for comparing primary biological sequence information , such as the amino-acid sequences of proteins or the nucleotides of DNA and/or RNA sequences.[11]

2.2 The use of tools in genome analysis :

- Gasperskaja and Kučinskas in 2017 states that how the tools are used for the genome analysis in the paper of “The most common technologies and tools for functional genome analysis”.As changes in the human genome are highly likely to cause pathological conditions, functional analysis is vitally important for human health. This paper aims to give a brief overview of the most common technologies and tools that could be applied for functional genome analysis .
- Lee , Cho , Kim and Cho (2 may 2013), they described that Synthetic biology is an emerging discipline for designing and synthesizing predictable, measurable, controllable, and transformable biological systems . The paper published was “Emerging tools for Synthetic genome design” .

Chapter 3: Objectives

3. Objectives

1. Retrieval of Human Genome from ensemble database .
2. Development of a tool to perform alignment in python language.
3. Creating a GUI for the tool.

Chapter 4: Materials and Methods

4.Materials and Methodology:

`

4.1 Requirements :

Software required :

- Visual Studio code

Programming languages required :

- Java
- Python

Special plug-in required :

- Tkinter

Databases required :

- Human Genome Database(ensembl)

4.2 Methadology :

- Retrieval of reference Human Genome sequence from the Database (The Genome Database)[6].
- Developing a program on the basis of Smith and Waterman's Algorithm for the alignment of sequence .
- Developed programs in two languages

4.3 Development of code

4.3.1 Code in Java :

```
import java.util.Stack;


public class SmithWaterman {
    private String one, two;
    private int matrix[][];
    private int gap;
    private int match;
    private int o;
    private int l;
    private int e;

    public SmithWaterman(String one, String two) {
        this.one = "-" + one.toLowerCase();
        this.two = "-" + two.toLowerCase();
        this.match = 2;

        // Define affine gap starting values
        o = -2;
        l = 0;
        e = -1;

        // initialize matrix to 0
        matrix = new int[one.length() + 1][two.length() + 1];
        for (int i = 0; i < one.length(); i++)
            for (int j = 0; j < two.length(); j++)
                matrix[i][j] = 0;

    }

    // returns the alignment score
    public int computeSmithWaterman() {
        for (int i = 0; i < one.length(); i++) {
            for (int j = 0; j < two.length(); j++) {
                gap = o + (l - 1) * e;
                if (i != 0 && j != 0) {
                    if (one.charAt(i) == two.charAt(j)) {
                        // match
                        // reset l
                        l = 0;
                        matrix[i][j] = Math.max(0, Math.max(
                                matrix[i - 1][j - 1] + match, Math.max(
                                        matrix[i - 1][j] + gap,
                                        matrix[i][j - 1] + gap)));
                    } else {
                        // gap
                        l++;
                        matrix[i][j] = Math.max(0, Math.max(
                                matrix[i - 1][j - 1] + gap, Math.max(
                                        matrix[i - 1][j] + gap,
                                        matrix[i][j - 1] + gap)));
                    }
                }
        }
    }

    // find the highest value
    int longest = 0;
    int iL = 0, jL = 0;
    for (int i = 0; i < one.length(); i++) {
        for (int j = 0; j < two.length(); j++) {
            if (matrix[i][j] > longest) {
                longest = matrix[i][j];
                iL = i;
                jL = j;
            }
        }
    }

    // Backtrack to reconstruct the path
    int i = iL;
    int j = jL;
    Stack<String> actions = new Stack<String>();

    while (i != 0 && j != 0) {
        // diag case
        if (Math.max(matrix[i - 1][j - 1],
                Math.max(matrix[i - 1][j], matrix[i][j - 1])) == matrix[i - 1][j - 1]) {
            actions.push("align");
            i = i - 1;
            j = j - 1;
            // left case
        } else if (Math.max(matrix[i - 1][j - 1],
                Math.max(matrix[i - 1][j], matrix[i][j - 1])) == matrix[i][j - 1]) {
            actions.push("insert");
            j = j - 1;
            // up case
        } else {
            actions.push("delete");
            i = i - 1;
        }
    }

    String alignOne = new String();
    String alignTwo = new String();

    Stack<String> backActions = (Stack<String>) actions.clone();
    for (int z = 0; z < one.length(); z++) {
        alignOne = alignOne + one.charAt(z);
        if (!actions.empty()) {
            String curAction = actions.pop();
            // System.out.println(curAction);
            if (curAction.equals("insert")) {
                alignOne = alignOne + "-";
                while (actions.peek().equals("insert")) {
                    alignOne = alignOne + "-";
                    actions.pop();
                }
            }
        }
    }

    for (int z = 0; z < two.length(); z++) {
        alignTwo = alignTwo + two.charAt(z);
        if (!backActions.empty()) {
            String curAction = backActions.pop();
            if (curAction.equals("delete")) {
                alignTwo = alignTwo + "-";
                while (backActions.peek().equals("delete")) {
                    alignTwo = alignTwo + "-";
                    backActions.pop();
                }
            }
        }
    }

    // print alignment
    System.out.println(alignOne + "\n" + alignTwo);
    return longest;
}

public void printMatrix() {
    for (int i = 0; i < one.length(); i++) {
        if (i == 0) {
            for (int z = 0; z < two.length(); z++) {
                if (z == 0)
                    System.out.print("    ");
                System.out.print(two.charAt(z) + "  ");


                if (z == two.length() - 1)
                    System.out.println();
            }
        }


        for (int j = 0; j < two.length(); j++) {
            if (j == 0) {
                System.out.print(one.charAt(i) + "  ");
            }
            System.out.print(matrix[i][j] + "  ");
        }
        System.out.println();
    }
    System.out.println();
}
```

```
  public static void main(String[] args) {

      // DNA sequence Test:

      SmithWaterman sw = new SmithWaterman("TAACCCTAACCCTAACATGCAG",
"GCTGAGACTTCC");

      System.out.println("Alignment Score: " + sw.computeSmithWaterman());

      sw.printMatrix();



   }

}
```

Fig 4.1 . Code in Java

4.3.2 Code in Python :

4.3.2.1 Front End code :

```
import swaln as swa

from tkinter import *


root=Tk()


root.title("Tool")

thelabel = Label(root,text="Application for alignment").grid(row=1,column=2)

#thelabel.pack()


seqA = StringVar()

seqB = StringVar()


Label(root,text="Select Genome = ").grid(row=7,column=1)

e1 = Entry(root,textvariable = seqA)

e1.grid(row=7,column=2)

Label(root, text='Enter a Sequence').grid(row=8,column=1)

e2 = Entry(root,textvariable = seqB)

e2.grid(row=8, column=2)


button = Button(root,text="Align Sequence",command=lambda:swa.s_w(seqA.get(),seq
B.get())).grid(row=9,column=2)


"""menu = Menu(root)

root.config(menu=menu)
```

```
filemenu = Menu(menu)
menu.add_cascade(label='File', menu=filemenu)
filemenu.add_command(label='New')
filemenu.add_command(label='Open...')
filemenu.add_command(label='Refresh')
filemenu.add_separator()
filemenu.add_command(label='Exit', command=root.quit)
helpmenu = Menu(menu)
menu.add_cascade(label='Help', menu=helpmenu)
helpmenu.add_command(label='About') """
root.mainloop()
```

4.3.2.2 Back end Code :

```
import sys

# open the file
#Human_Genome = open("Humanch1.fasta")

# read the contents
#seq1 = Human_Genome.read()
#print ("Enter the first sequence. ")
#seq1=sys.stdin.readline()
#seq1=seq1.replace("\n","")
#print ("Enter the second sequence. ")
#seq2=sys.stdin.readline()
#seq2=seq2.replace("\n","")
```

```
#valid = 'ATCG'
#if any(i not in valid for i in seq1 + seq2):
#    print ('Please provide 2 valid DNA sequences as input.\n', 'Exit!\n')
#    sys.exit()

match    =  2
mismatch = -2
gap      = -1

def s_w(seqA, seqB):

    cols      = len(seqA)
    rows      = len(seqB)
    matrix    = [[0 for row in range(rows+1)] for col in range(cols+1)]
    paths     = [[0 for row in range(rows+1)] for col in range(cols+1)]
    max_score = 0
    s1, s2 = [], []

    for i in range(cols):
        for j in range(rows):
            if seqA[i] == seqB[j]:
                diag = matrix[i][j] + match
            else:
                diag = matrix[i][j] + mismatch
            up    = matrix[i + 1][j] + gap
            left  = matrix[i][j + 1] + gap
            score = max(0,diag, up, left)
            matrix[i+1][j+1] = score
            if score > max_score:
                max_score = score
                start_pos = [i+1, j+1]

            if matrix[i+1][j+1]   == diag and matrix[i+1][j+1] != 0:
                paths[i+1][j+1] = 'diag'
            elif matrix[i+1][j+1] == up   and matrix[i+1][j+1] != 0:
                paths[i+1][j+1] = 'up'
            elif matrix[i+1][j+1] == left and matrix[i+1][j+1] != 0:
                paths[i+1][j+1] = 'left'

    i, j = start_pos
    start_path = paths[i][j]
    while start_path != 0:
        if start_path == 'diag':
            s1.append(seqA[i-1])
            s2.append(seqB[j-1])
            i, j = i-1, j-1
        elif start_path == 'up':
            s1.append('-')
            s2.append(seqB[j-1])
            j = j-1
        else:
            s1.append(seqA[i-1])
    import sys
```

```
# open the file
#Human_Genome = open("Humanch1.fasta")

# read the contents
#seq1 = Human_Genome.read()
#print ("Enter the first sequence. ")
#seq1=sys.stdin.readline()
#seq1=seq1.replace("\n","")
#print ("Enter the second sequence. ")
#seq2=sys.stdin.readline()
#seq2=seq2.replace("\n","")

#valid = 'ATCG'
#if any(i not in valid for i in seq1 + seq2):
#    print ('Please provide 2 valid DNA sequences as input.\n', 'Exit!\n')
#    sys.exit()

match    =  2
mismatch = -2
gap      = -1

def s_w(seqA, seqB):

    cols      = len(seqA)
    rows      = len(seqB)
    matrix    = [[0 for row in range(rows+1)] for col in range(cols+1)]
```

```
paths     = [[0 for row in range(rows+1)] for col in range(cols+1)]
max_score = 0
s1, s2 = [], []

for i in range(cols):
    for j in range(rows):
        if seqA[i] == seqB[j]:
            diag = matrix[i][j] + match
        else:
            diag = matrix[i][j] + mismatch
        up    = matrix[i + 1][j] + gap
        left  = matrix[i][j + 1] + gap
        score = max(0,diag, up, left)
        matrix[i+1][j+1] = score
        if score > max_score:
            max_score = score
            start_pos = [i+1, j+1]

        if matrix[i+1][j+1]   == diag and matrix[i+1][j+1] != 0:
            paths[i+1][j+1] = 'diag'
        elif matrix[i+1][j+1] == up   and matrix[i+1][j+1] != 0:
            paths[i+1][j+1] = 'up'
        elif matrix[i+1][j+1] == left and matrix[i+1][j+1] != 0:
            paths[i+1][j+1] = 'left'

i, j = start_pos
start_path = paths[i][j]
```

```
    while start_path != 0:

        if start_path == 'diag':

            s1.append(seqA[i-1])

            s2.append(seqB[j-1])

            i, j = i-1, j-1

        elif start_path == 'up':

            s1.append('-')

            s2.append(seqB[j-1])

            j = j-1

        else:

            s1.append(seqA[i-1])

            s2.append('-')

            i = i-1

        start_path = paths[i][j]


        print(s1,s2)

    #s1.reverse()

    #s2.reverse()

    seq1 = "".join(s1)

    seq2 = ''.join(s2)

    print("Score for the alignment is = " ,max_score)

    print (' Alignment\n', seq1, '\n', seq2, '\n')




#aln1, aln2 = s_w(seqA, seqB)

#print("Score for the alignment is = " ,max_score)
```

```
#print (' Alignment\n', seqA, '\n', seqB, '\n')
        s2.append('-')
            i = i-1
        start_path = paths[i][j]


        print(s1,s2)
    #s1.reverse()
    #s2.reverse()
    seq1 = "".join(s1)
    seq2 = ''.join(s2)
    print("Score for the alignment is = " ,max_score)
    print (' Alignment\n', seq1, '\n', seq2, '\n')
#aln1, aln2 = s_w(seqA, seqB)
#print("Score for the alignment is = " ,max_score)
#print (' Alignment\n', seqA, '\n', seqB, '\n')
```

Fig 4.2 . Code in Python

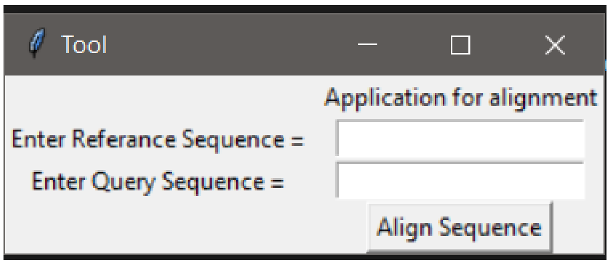
Chapter 5: Result and Discussion

Chapter 5: Result and Discussion

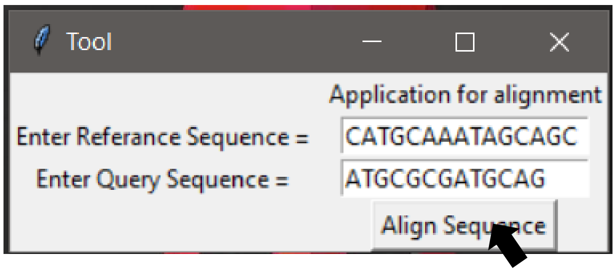
5.1 Result

After running this tool , we get the complete alignment of the given sequences and the score for the alignment of the sequence by the Smith and Waterman Algorithm .

5.1.1 Output for Python Code :



Fig(5.1) : Tool window(1)



Enter Referance Sequence = CATGCAAATAGCAGC

Enter Query Sequence = ATGCGCGATGCAG

Fig(5.2) : Tool window(2)

5.1.2 Output of alignment :

['G'] ['G']

['G', 'T'] ['G', '-']

['G', 'T', 'A'] ['G', '-', 'A']

```
['G', 'T', 'A', 'C'] ['G', '-', 'A', 'C']

['G', 'T', 'A', 'C', 'G'] ['G', '-', 'A', 'C', 'G']

['G', 'T', 'A', 'C', 'G', 'T'] ['G', '-', 'A', 'C', 'G', 'T']

['G', 'T', 'A', 'C', 'G', 'T', 'A'] ['G', '-', 'A', 'C', 'G', 'T', 'A']

['G', 'T', 'A', 'C', 'G', 'T', 'A', '-'] ['G', '-', 'A', 'C', 'G', 'T', 'A', 'G']

['G', 'T', 'A', 'C', 'G', 'T', 'A', '-', 'C'] ['G', '-', 'A', 'C', 'G', 'T', 'A', 'G', 'C']

['G', 'T', 'A', 'C', 'G', 'T', 'A', '-', 'C', 'G'] ['G', '-', 'A', 'C', 'G', 'T', 'A', 'G', 'C',
'G']

['G', 'T', 'A', 'C', 'G', 'T', 'A', '-', 'C', 'G', 'T'] ['G', '-', 'A', 'C', 'G', 'T', 'A', 'G', 'C',
'G', '-']

['G', 'T', 'A', 'C', 'G', 'T', 'A', '-', 'C', 'G', 'T', 'A'] ['G', '-', 'A', 'C', 'G', 'T', 'A', 'G',
'C', 'G', '-', '-']

['G', 'T', 'A', 'C', 'G', 'T', 'A', '-', 'C', 'G', 'T', 'A', 'C'] ['G', '-', 'A', 'C', 'G', 'T', 'A',
'G', 'C', 'G', '-', '-', 'C']

['G', 'T', 'A', 'C', 'G', 'T', 'A', '-', 'C', 'G', 'T', 'A', 'C', 'G'] ['G', '-', 'A', 'C', 'G', 'T',
'A', 'G', 'C', 'G', '-', '-', 'C', 'G']

['G', 'T', 'A', 'C', 'G', 'T', 'A', '-', 'C', 'G', 'T', 'A', 'C', 'G', 'T'] ['G', '-', 'A', 'C', 'G',
'T', 'A', 'G', 'C', 'G', '-', '-', 'C', 'G', 'T']

['G', 'T', 'A', 'C', 'G', 'T', 'A', '-', 'C', 'G', 'T', 'A', 'C', 'G', 'T', 'A'] ['G', '-', 'A', 'C',
'G', 'T', 'A', 'G', 'C', 'G', '-', '-', 'C', 'G', 'T', 'A']

Score for the alignment is =  20

 Alignment

 GTACGTA-CGTACGTA

 G-ACGTAGCG—CGTA
```

5.1.3 Output of Java Code :

```
-taacc-ct-aac--cctaacatgcag

-gc--tgagacttcc

Alignment Score: 9

  - g c t g a g a c t t c c

- 0 0 0 0 0 0 0 0 0 0 0 0 0

t 0 0 0 2 1 0 0 0 0 2 2 1 0

a 0 0 0 0 0 3 2 2 1 0 0 0 0

a 0 0 0 0 0 2 2 4 3 1 0 0 0

c 0 0 2 1 0 0 0 0 6 5 3 2 2

c 0 0 2 1 0 0 0 0 2 5 3 5 4

c 0 0 2 1 0 0 0 0 2 4 3 5 7

t 0 0 0 4 3 1 0 0 0 4 6 5 5

a 0 0 0 0 0 5 4 2 1 2 3 2 0

a 0 0 0 0 0 2 4 6 5 3 0 0 0

c 0 0 2 1 0 0 0 1 8 7 5 2 2

c 0 0 2 1 0 0 0 0 3 7 5 7 6

c 0 0 2 1 0 0 0 0 2 6 5 7 9

t 0 0 0 4 3 1 0 0 0 4 8 7 7

a 0 0 0 0 0 5 4 2 1 2 5 4 2

a 0 0 0 0 0 2 4 6 5 3 2 1 0

c 0 0 2 1 0 0 0 1 8 7 5 4 3
```

```
a 0 0 0 0 0 2 1 2 7 6 4 1 0

t 0 0 0 2 1 0 0 0 2 9 8 7 5

g 0 2 1 0 4 3 2 1 0 6 5 3 1

c 0 0 4 3 2 1 0 0 3 5 4 7 6

a 0 0 2 1 0 4 3 2 2 3 2 3 2

g 0 2 1 0 3 3 6 5 3 0 0 0 0
```

5.2 Discussion :

This project helps to find out any query sequence is present in the reference sequence or not . With the help of this project we get the alignment of the query sequence and the reference sequence . This tools also give us the alignment score for the performed alignment , the method or algorithm used for it is Smith and Waterman’s Algorithm for global alignment.

Chapter 6: Conclusion and Future Scope

6. Conclusion and Future Scope

6.1 Conclusion

1. This tool provides the specific alignment for the given sequence.
2. The tool can be used to get the score for the alignment

6.2 Future Scope

1. The tool can be assign with the HTML and PHP code to give the web interface for the tool .
2. The database of variety of the genomes can be also stored in the database with the help of SQL so the tool can be further use for analysis of different genomes .

Chapter 7 : References

7. Referances :

1]. Bo-Rahm Lee, Suhyung Cho, Yoseb Song, Sun Chang Kim, and Byung-Kwan Cho (2013) “Emerging Tools for Synthetic Genome Design” (*Mol Cells*) 2013 , 35 : (1,2)

2]. Evelina Gasperskaja and Vaidutis Kučinskas(2017)” The most common technologies and tools for functional genome analysis ” (*Acta Med Litu*) 2017, 24 :(1,2)

3]. Harry M. Bohle and Toni Gabaldón (2012) “Selection of Marker Genes Using Whole-Genome DNA Polymorphism Analysis” (*Evol bioinform online*)2012,8 : (1,2)

4]. G Van Rossum -USENIX annual technical conference, (2007) [HTML] PythonProgramming Language.(Book)- colenak.ptkpt.net (1,2)

5]. Leroy Hood and Lee Rowen (2013) ,”The Human Genome Project: big science transforms biology and medicine” , (*Genome Med*)2013,5(9):(2)

6]. Ken Arnold, James Gosling and David Holmes (2005) THE Java™ Programming Language, Fourth Edition (Book) (1,35)

7]. T. F. SMITE AND M. S. WATERMAN (1981) “Identification of Common Molecular Subsequences” (*J. Mol. Biol.*) 147, 195-197

8]. https://web.ornl.gov/sci/techresources/Human_Genome/index.shtml (12/12/2019)

9]. https://en.wikipedia.org/wiki/Smith%E2%80%93Waterman_algorithm (13/12/2019)

10]. https://en.wikipedia.org/wiki/FASTA (13/12/2019)

11]. https://en.wikipedia.org/wiki/BLAST_(biotechnology) (13/12/2019)

12]. https://en.wikipedia.org/wiki/Human_Genome_Project#References/ (12/12/2019)

13]. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5467957/ (12/12/2019)

14]. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3887862/ (12/12/2019)