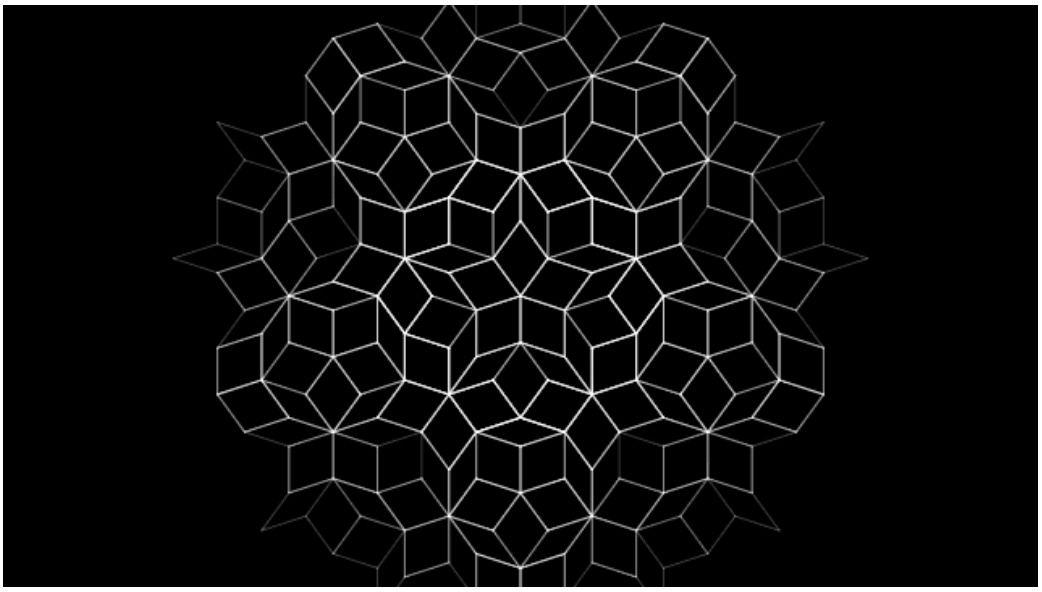
19BIO112 – INTELLIGENCE OF BIOLOGICAL SYSTEMS 2

L-SYSTEMS AND NCBI BLAST SEARCH



DEEPTHI SUDHARSAN

CB.EN.U4AIE19022

B-TECH CSE – AI

AMRITA COLLEGE OF ENGINEERING, COIMBATORE

ACKNOWLEDGEMENT

This project would not have been possible without the support and guidance of Dr. K P Soman, Mr. Premjith B, Mr. Paul and Mr. Adityan who have provided me with sufficient knowledge and guidelines even during these unprecedented and trying times.

I would also like to thank our Computer Science and Engineering (Artificial Intelligence) department for giving me this opportunity to nurture and hone my skills.

Furthermore, I would like to thank the Amrita Vishwa Vidyapeetham management for ample resources to avail my project needs and a platform for online lectures during this lockdown.

Table of Contents

PART I (L - System using Processing): 4

ABSTRACT : 4

OBJECTIVE 5

L – SYSTEM EXAMPLE 1: 5

PROCESSING CODE AND EXPLANATION : 5

OUTPUT: 10

L – SYSTEM EXAMPLE 2: 11

PROCESSING CODE AND EXPLANATION : 11

OUTPUT 14

APPLICATIONS OF L - SYSTEMS: 15

REFERENCES: 15

PART II (NCBI Blast Search): 16

OBJECTIVE : 16

FASTA FILE SEQUENCES GIVEN: 16

RESULT: 17

RATIONALE: 18

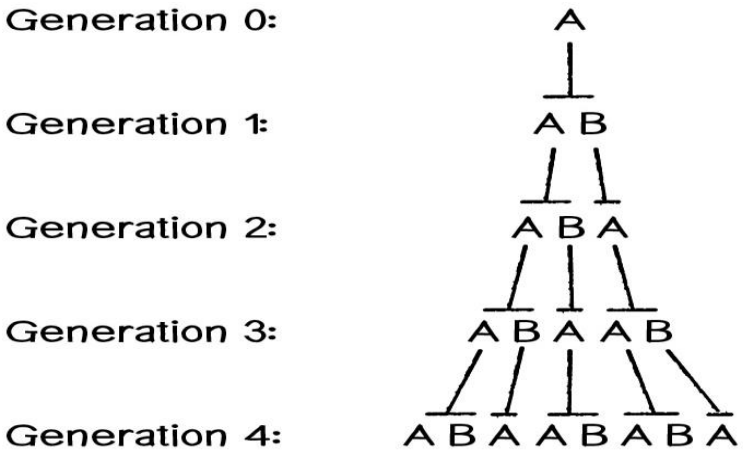
OBSERVATIONS: 20

REFERENCES: 21

PART I (L - System using Processing):

ABSTRACT :

An L - System is a parallel string rewriting system. A string rewriting system consists of an initial string, called the seed/axiom, and a set of rules for specifying how the symbols in a string are rewritten as (replaced by) strings.



Here, A is the seed or axiom.

Rules are,

A ->AB

B ->A

Processing is a graphics library and IDE based on
the Java programming language that helps to simplify computer programming, providing new classes and

mathematical functions for creating and processing computer graphics.

OBJECTIVE :

To create different L-systems using Processing software and UI (user interface).

L – SYSTEM EXAMPLE 1:

Axiom : F

Rules: F -> FF+[+F-F-F]-[-F+F+F]

PROCESSING CODE AND EXPLANATION :

Initialization and importing of Packages:

```
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.geom.Point2D;
import java.util.Dictionary;
import java.util.Hashtable;
import java.util.Vector;

import javax.swing.*;
```

```
static String lastString = "";
static int cwidth = 700;
static int cheight = 700;
static Point2D.Double current = new Point2D.Double(cwidth/2, 0);
static double currDir = 90.0;
static Vector positionStack = new Vector();
static Vector angleStack = new Vector();
static double deltaAngle = 25.0;
static double linelen = 200;
```

Here, we are importing the packages that we require for the UI (User Interface) and plotting the L – system.

The variable lastString is to store the current string and given as input to generate the next string.

cwidth and cheight are the dimensions of the canvas where the L-system is going to be plotted.

Current is the initial point or coordinates (here, on the x axis, at cwidth/2 location).

currDir is the direction it is facing at the moment.

deltaAngle is the rotation angle in radians.

linelen is the length of the line drawn (set to 200 in the beginning).

generate Function:

```
String generate(String sentence) {
        Dictionary rules = new Hashtable();

        rules.put("F", "FF+[+F-F-F]-[-F+F+F]");
        String nextSentence = "";

        // Re-initialize to begin from same place...
        linelen *= 0.5;
        currDir = 90;
        current = new Point2D.Double(cwidth / 2, 0);
        positionStack = new Vector();
        angleStack = new Vector();
```

```
        if (!lastString.equals("")) {

            for (int i = 0; i < sentence.length(); i++) {
                String currChar = sentence.substring(i, i + 1);

                String value = (String) rules.get(currChar);
                if (value != null) {
                    nextSentence += rules.get(currChar);
                } else {
                    nextSentence += currChar;
                }

            }

            lastString = nextSentence;
        } else {
            // This is special case ... we need to draw the first sequence as-is...
            nextSentence = "F";
            lastString = "F";
        }
```

A dictionary with the rules for the L-system is created. The size of line is made smaller in every iteration. The current coordinates and direction are re initialized to begin from the same place.

If the laststring is not empty, using a for loop, we split the string into individual substrings and based on the rules, we append the result to the nextstring. If the substring doesn't match any rule, the substring itself gets appended.

Else if laststring is empty that means we haven't begun yet or the laststring hasn't been set to the axiom, so we set nextSentence and laststring to the axiom.

rotatedLine and drawPaths Function:

```
Point2D.Double rotatedLine(double linelen) {
       double delx = linelen * Math.cos(Math.toRadians(currDir));
       double dely = linelen * Math.sin(Math.toRadians(currDir));

       return new Point2D.Double(delx, dely);
   }
```

rotatedLine function is used to set the change in the coordinates of the new point with respect to the origin.

```
Point2D.Double drawPaths(Graphics graphics, Point2D.Double start, String input) {
        // To cover coordinate mirror
        int yheight = cheight; // This comes from the original JPanel height that was set...
        Graphics2D g = (Graphics2D) graphics;
        g.setStroke(new BasicStroke(1));
        for (int i = 0; i < input.length(); i++) {
            String chr = input.substring(i, i + 1);
            Point2D.Double offset = null;
            switch (chr) {
                case "F":
                    g.setColor(Color.YELLOW);
                    offset = rotatedLine(linelen);
                    g.drawLine((int) current.x, yheight - (int) current.y, (int)( current.x + offset.x), yheight - (int) (current.y + offset.y));
                    current = new Point2D.Double(current.x + offset.x, current.y + offset.y);
                    break;
                case "+":
                    currDir += -deltaAngle;
                    break;
                case "-":
                    currDir += deltaAngle;
                    break;
                case "[":
                    positionStack.add(current);
                    angleStack.add(currDir);
                    break;
                case "]":
                    current = (Point2D.Double) positionStack.lastElement();
                    positionStack.remove(positionStack.size()-1);
                    currDir = (double) angleStack.lastElement();
                    angleStack.remove(angleStack.size()-1);
                    break;
                default:
                    System.out.println("Unimplemented command .... " + chr);
            }
            currDir %= 360.0;

        return current;
    }
```

Using a for loop, we split the string into individual substrings. Using switch case, we check if the substring is equal to F, then we move forward, (note that here, in the canvas, the axes are inverted, so the x axis is on top and the y axis is on the left that is why while plotting, the y coordinate we subtract it from the

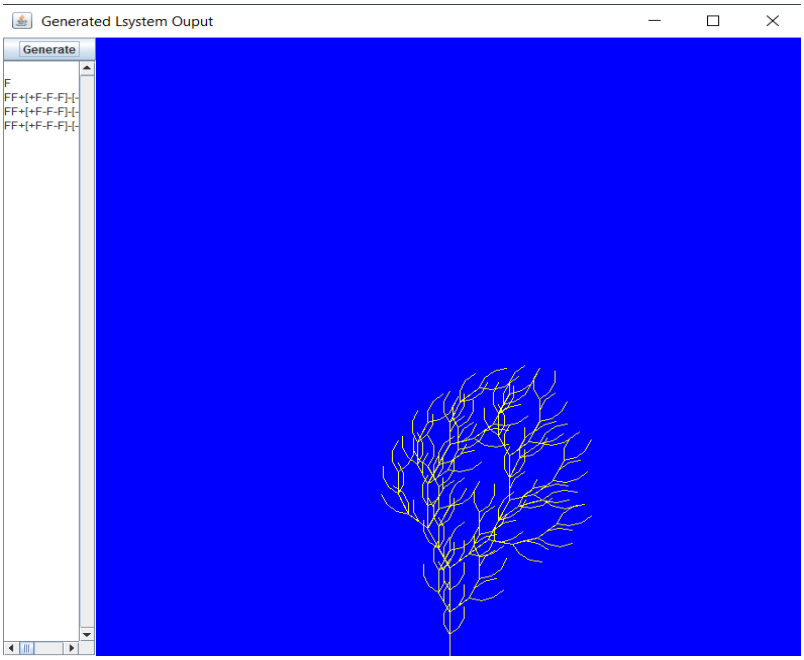
total height of the canvas). If it is “+” we rotate counter clockwise and if “-” we rotate clockwise. If the substring is equal to “[”, we push into stack, if “]”, we pop from the stack, else we print, “Unimplemented command”. Then we return current or the current coordinates.

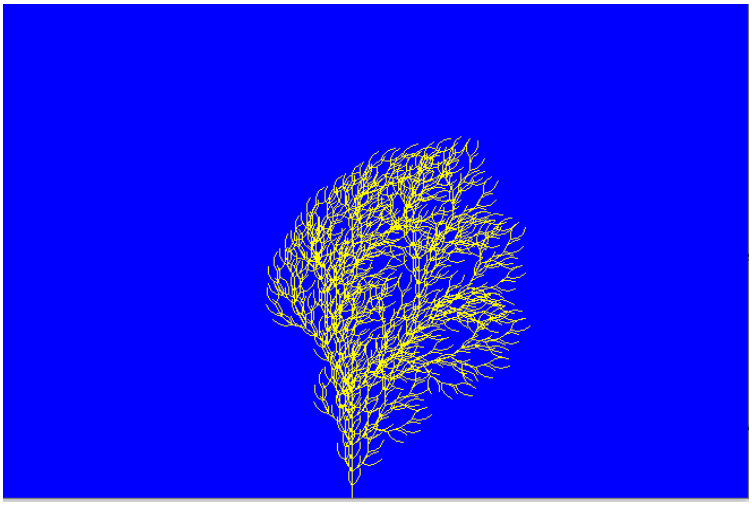
```
void setup() {
        JFrame jframe;
        jframe = new JFrame(" Generated Lsystem Ouput ");
        final JTextArea textarea = new JTextArea("");
        final JPanel dwg = new JPanel();

        Container container = jframe.getContentPane();
        GridBagLayout layout = new GridBagLayout();
        container.setLayout(layout);

        GridBagConstraints cons = new GridBagConstraints();
        cons.gridx = 0;
        cons.gridy = 0;
        cons.fill = GridBagConstraints.HORIZONTAL;
        cons.gridwidth = 1;
        cons.gridheight = 1;
        cons.weightx = 1.0;
        cons.weighty = 0.0;

        JButton button = new JButton(" Generate ");
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                textarea.setText(textarea.getText() + "\n" + generate(lastString));
                Point2D.Double newPoint = drawPaths(dwg.getGraphics(), current, lastString);
            }
        });
        button.setMinimumSize(new Dimension(40, 20));

        container.add(button, cons);

        JScrollPane scroll = new JScrollPane(textarea,
        JScrollPane.VERTICAL_SCROLLBAR_ALWAYS, JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);

        cons.fill = GridBagConstraints.BOTH;
        cons.gridx = 0;
        cons.gridy = 1;
        cons.gridheight = 9;
        cons.weighty = 1.0;
        container.add(scroll, cons);

        dwg.setPreferredSize(new Dimension(cwidth, cheight));
        cons.fill = GridBagConstraints.BOTH;
        cons.gridx = 1;
        cons.gridy = 0;
        cons.gridheight = 10;
        cons.gridwidth = 5;
        cons.weighty = cons.weightx = 0.0;
        dwg.setBackground(Color.BLUE);
        container.add(dwg, cons);

        container.doLayout();
        jframe.pack();
        jframe.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        jframe.setVisible(true);

    }
```

The main purpose of the setup() function is to create a UI to allow user to generate each iteration of the L-system with the press of the button and generate the L-system plot simultaneously.

This function creates the pop up canvas and output string along with the generate button. Every time the generate button is clicked, the string is rewritten and the plot is updated.

OUTPUT:





L – SYSTEM EXAMPLE 2:

Axiom : F-F-F-F

Rules: F -> F[F]-F+F[--F]+F-F

PROCESSING CODE:

```
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.geom.Point2D;
import java.util.Dictionary;
import java.util.Hashtable;
import java.util.Vector;

import javax.swing.*;

    static String lastString = "";
    static int cwidth = 1000;
    static int cheight = 1000;
    static Point2D.Double current = new Point2D.Double(cwidth/2, 0);
    static double currDir = 90.0;
    static Vector positionStack = new Vector();
    static Vector angleStack = new Vector();
    static double deltaAngle = 90.0;
    static double linelen = 200;
```

```
String generate(String sentence) {
        Dictionary rules = new Hashtable();
        rules.put("F", "F[F]-F+F[--F]+F-F");
        String nextSentence = "";
        // Re-initialize to begin from same place...
        linelen *= 0.5;
        currDir = 90;
        current = new Point2D.Double(cwidth / 2, 0);
        positionStack = new Vector();
        angleStack = new Vector();
```

```
        if (!lastString.equals("")) {

            for (int i = 0; i < sentence.length(); i++) {
                String currChar = sentence.substring(i, i + 1);

                String value = (String) rules.get(currChar);
                if (value != null) {
                    nextSentence += rules.get(currChar);
                } else {
                    nextSentence += currChar;
                }
            }

            lastString = nextSentence;
        } else {
            // This is special case ... we need to draw the first sequence as-is...
            nextSentence = "F-F-F-F";
            lastString = "F-F-F-F";
        }

        return nextSentence;
    }
```

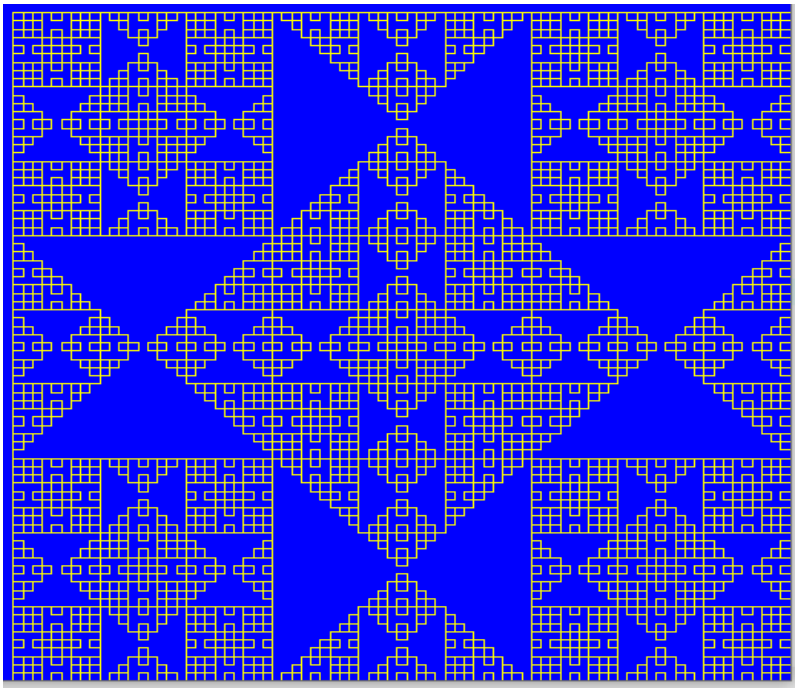
```
Point2D.Double rotatedLine(double linelen) {
        double delx = linelen * Math.cos(Math.toRadians(currDir));
        double dely = linelen * Math.sin(Math.toRadians(currDir));

        return new Point2D.Double(delx, dely);
    }
```

```
Point2D.Double drawPaths(Graphics graphics, Point2D.Double start, String input) {
       // To cover coordinate mirror
       int yheight = cheight; // This comes from the original JPanel height that was set...
       Graphics2D g = (Graphics2D) graphics;
       g.setStroke(new BasicStroke(1));
       g.setColor(Color.BLUE);
       g.fillRect(0, 0, cwidth, cheight);

       for (int i = 0; i < input.length(); i++) {
           String chr = input.substring(i, i + 1);
           Point2D.Double offset = null;
           switch (chr) {
              case "F":
                  g.setColor(Color.YELLOW);
                  offset = rotatedLine(linelen);
                  g.drawLine((int) current.x, yheight - (int) current.y, (int)( current.x + offset.x), yheight - (int) (current.y + offset.y));
                  current = new Point2D.Double(current.x + offset.x, current.y + offset.y);
                  break;
```

```
                case "+":
                    currDir += -deltaAngle;
                    break;
                case "-":
                    currDir += deltaAngle;
                    break;
                case "[":
                    positionStack.add(current);
                    angleStack.add(currDir);
                    break;
                case "]":
                    current = (Point2D.Double) positionStack.lastElement();
                    positionStack.remove(positionStack.size()-1);
                    currDir = (double) angleStack.lastElement();

                    angleStack.remove(angleStack.size()-1);
                    break;
                default:
                    System.out.println("Unimplemented command .... " + chr);
            }
            currDir %= 360.0;
        }
        return current;
    }

void setup() {
        JFrame jframe;
        jframe = new JFrame(" Generated Lsystem Ouput ");
        final JTextArea textarea = new JTextArea("");
        final JPanel dwg = new JPanel();
        dwg.setBackground(Color.BLUE);
        Container container = jframe.getContentPane();
        GridBagLayout layout = new GridBagLayout();
        container.setLayout(layout);

        GridBagConstraints cons = new GridBagConstraints();
        cons.gridx = 0;
        cons.gridy = 0;
        cons.fill = GridBagConstraints.HORIZONTAL;
        cons.gridwidth = 1;
        cons.gridheight = 1;
        cons.weightx = 1.0;
        cons.weighty = 0.0;

        JButton button = new JButton(" Generate ");
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {

                textarea.setText(textarea.getText() + "\n" + generate(lastString));
                dwg.setBackground(Color.BLUE);
                Point2D.Double newPoint = drawPaths(dwg.getGraphics(), current, lastString);
            }
        });
        button.setMinimumSize(new Dimension(40, 20));

        container.add(button, cons);
```

```
        JScrollPane scroll = new JScrollPane(textarea,
        JScrollPane.VERTICAL_SCROLLBAR_ALWAYS, JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);

        cons.fill = GridBagConstraints.BOTH;
        cons.gridx = 0;
        cons.gridy = 1;
        cons.gridheight = 9;
        cons.weighty = 1.0;
        container.add(scroll, cons);

        dwg.setPreferredSize(new Dimension(cwidth, cheight));
        cons.fill = GridBagConstraints.BOTH;
        cons.gridx = 1;
        cons.gridy = 0;
        cons.gridheight = 10;
        cons.gridwidth = 5;
        cons.weighty = cons.weightx = 0.0;
        dwg.setBackground(Color.BLUE);
        container.add(dwg, cons);

        container.doLayout();
        jframe.pack();
        jframe.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    jframe.setVisible(true);

}
```

OUTPUT:



By changing and playing with the values, rules, axioms and angles, we can make different L systems. For example, the same code as the first example has been modified by giving a different axiom and set of rules and the deltaAngle to get the above new L system. Also , another modification in this code is we are resetting the canvas and redrawing each and every iteration step to get the output.

APPLICATIONS OF L - SYSTEMS:

1. To model the development of simple multicellular organisms (for example, algae) in terms of division, growth, and death of individual cells and to describe the behaviour of plant cells.
2. In Textile Industry to create new patterns and designs.
3. In architecture, for designing and for generating architectural plans.
4. Used to carry out many mathematical computations like calculating factorial, summing series etc.
5. Writing musical compositions and playing it on a microcomputer
6. Cryptology (for coding and encoding secret messages)

REFERENCES:

1. https://www.youtube.com/watch?v=17WoOqgXsRM&list=PLRqwX-V7Uu6ZiZxtDDRCi6uhfTH4FilpH
2. http://algorithmicbotany.org/papers/l-sys.csiro96.pdf
3. https://stackoverflow.com/questions/29793849/how-to-create-a-simple-l-system-in-processing
4. https://www.computerhope.com/jargon/p/processi.htm
5. https://morphocode.com/intro-to-l-systems/

PART II (NCBI Blast Search):

OBJECTIVE :

To analyze and determine three DNA sequences given using NCBI Blast Search and to tabulate and justify the observations.

FASTA FILE SEQUENCES GIVEN:

Three sequences given have been provided to search, ponder upon and tabulate the observations.

>CB.EN.U4AIE.19022 QSP006657

AAGTYGAGCGGTAAGGCTCCTTCGGGAGTACACGAGCGGCGAACGGGTGAGTAACACGTGAGCAATCTGCCCTTCTCTTCGGAATAA
CCATTGGAAACGATGGCTAATGCCGAATACGACCTCGGATCACATGTTCTGAGGTGGAAAGCTCCGGCGGAGAAGGATGAGCTCGC
GGCCTATCAGCTAGTTGGTGAGGTAACGGCTCACCAAGGCATTGACGGGTAGCCGGCCTGAGAGGGTGACCGGCCACACTGGGACT
GAGACACGGCCCAGACTCCTACGGGAGGCAGCAGTGGGGAATATTGGACAATGGGCGAAACCTTGATCCAGCAACGCCGCGTGAG
GGATGACGGCCTTCGGGTTGTAAACCTCTTTCAGCGGGGACGAAGCGAAAGTGACGGTACCCGCAGAAGAAGGACCKGCCAACTAC
GTGCCAGCAGCCGCGGTAATACGTAGGGTCCGAGCGTTGTCCGGAATTATTGGGCGTAAAGGGCTCGTAGGCGGTTTGTCGCGTCG
GGAGTGAAAACTCAGGGCTCAACCCTGAGCGTGCTTCCGATACGGGCAGACTAGAGGTATGCAGGGGAGAACGGAATTCCTGGTGT
AGCGGTGGAATGCGCAGATATCAGGAGGAACACCGGTGGCGAAGGCGGTTCTCTGGGCATTACCTGACGCTGAGGAGCGAAAGCA
TGGGGAGCGAACAGGATTAGATACCCTGGTAGTCCATGCCGTAAACGTTGGGCGCTAGGTGTGGGGACCTTCCACGGTCTCCGTGCC
GCAGCTAACGCATTAAGCGCCCCGCCTGGGGAGTACGGCCGCAAGGCTAAAACTCAAAGGAATTGACGGGGGCCCGCACAAGCGGC
GGAGCATGCTGATTAATTCGATGCAACGCGAAGAACCTTACCTGGGTTTGACATATACCGGAAAGCTGCAGAGATGTAGCCCCCCTT
GTGGTCGGTATACAGGTGGTGCATGGCTGTCGTCAGCTCGTGTCGTGAGATGTTGGGTTAAGTCCCGCAACGAGCGCAACCCTCGTT
CTATGTTGCCAGCACGTAATGGTGGGGACTCATAGGAGACTGCCGGGGTCAACTCGGAGGAAGGTGGGGATGACGTCAAGTCTTCA
TGCCCCTTATGTCCAGGGCTTCAAGCATGCTACAATGGCCGGTACAAAGGGCTGCGAAACCGCAAGGTGGAGCGAATCCCAAAAAGC
CGGTCTCAGTTCGGATTGGGGTCTGCAACTCGACCCCATGAAGTCGGAGTCGCTAGTAATCGCAGATCAGCAACGCTGCGGTGAATA
CGTTCCCGGGCCTTGTACACACCGCCCGTCACGTCATGAAAGTCGCAACACCCGAAGCCAGTG

>CB.EN.U4AIE.19022 GOH000007

AGTAGTCATATGCTTGTCTCAAAGATTAAGCCATGCATGTCTAAGTATAAGCAATTTATACAGTGAAACTGCGAATGGCTCATTAAAT
CAGTTATCGTTTATTTGATAGTACCTTGCTAATTGGGATAAGCCTGGTAATTCTAGAGCTAATACATGCTTCAAAGCCCAACCTCCGG
AAGGGCTGTATTTATTAGATAAAAAATCAACAACCTTGATGATTCATAATAACTTGTCGAATCGCATGGCCTCGGGCCGGCGATGGT
TCATTCAAATTTCTGCCCTATCAACTTTCGATGGTAGGATAGAGGCCTACCATGGTTTCAACGGGTAACGGGGAATAAGGGTTCGGT
TCCGGAGAGGGAGCCTGAGAAACGGCTACCACATCCAAGGAAGGCAGCAGGCGCGCAAATTACCCAATCCCGACACGGGGAGGTA
GTGACAATAAATAACGATGCAGGGCCTTTCGGGTCTTGTAATTGGAATGAGTACAATGTAAATACCTTAACGAGGAACAATTGGAG
GGCAAGTCTGGTGCCAGCAGCCGCGGTAATTCCAGCTCCAAGACCGTATATTAAAGTTGTTGCAGTTAAAAAGCTCGTAGTTGAACC
TTGGGGAGGCCGCCCGGGTCCGCCTTTGCGGCGAGCACTCGGCGCGACTTCCCTTTCCTCCCTCGGGCCTTCTGGTGCGAGGGACTA
TTACTTTGAGTAAATGAGAGTGTTCAAAGCAGGCGCACGCTTGAATCTGTTAGCATGGAATAATAGAATAGGACGCATGGTTCTATT
TTGTTGGTTTCTAGGACCATCGTAATGATTAATAGGGACGGTCGGGGGCATCAGTATTCAGTTGTCAGAGGTGAAATTCTTGGATTT
ACTGAAGACTAACTACTGCGAAAGCATTTGCCAAGGACGTTTTCATTAATCAAGAACGAAAGTTAGGGGATCGAAGATGATCAGATA
CCGTCGTAGTCTTAACCATAAACTATGCCGACTAGGGATCGGGCGGCGTTCATTTAGTGACGCGCTCGGCACCTTACGAGAAATCAA
AGTTTTTGGGTTCTGGGGGGAGTATGGTCGCAAGGCTGAAACTTAAAGGAATTGACGGAAGGGCACCACCAGGAGTGGAGCCTGC
GGCTTAATTTGACTCAACACGGGGAAACTCACCAGGTCCAGACACAATAAGGATTGACAGATTGAGAGCTCTTTCTTGATTTTGTGG
GTGGTGGTGCATGGCCGTTCTTAGTTGGTGGAGTGATTTGTCTGCTTAATTGCGATAACGAACGAGACCTTAACCTCTAAATAGTGC
CACTAGCAATTTGCTGGCGGGTGCTTCTTAGGGGGACTATTGACTTGAAGTCGATGGAAGTTTGAGGCAATAACAGGTCTGTGATGC
CCTTAGACGTTCTGGGCCGCACGCGCGCTACACTGACGGAGCCAGCGAGTTGACCTTGGCCGAGAGGTCTGGGAAATCTTGTGAAA
CTCCGTCGTGCTGGGGATAGAGCATTGCAATTTTTGCTCTTCAACGAGGAATTCCTAGTAAGCGCAAGTCATCAGCTTGCGTTGATTA
CGTCCCTGCCCTTTGTACACACCGCCCGTCGCTACTACCGATTGAATGGCTTAGTGAGGCCTCCGGATTGCTTTAGGGGAGAGGGCA
ACCTTTTCCTGGAGGCGAGAAGCTGGTCAAACTTGGTCATTTAGAGGAAGTAAAAGTCGTAACAAGGTTTCCGTAGGTGAACCTGCG
GAAGGATCATTA

>CB.EN.U4AIE.19022 QMU000540

AGTGTTGGGTTTAAAGCAGGGGTTAAAGATTACAAACTTACTTATTACACTCCTGAGTACGAAACCAAAGATACTGATATCTTGGCA
GCGTTCCGAGTAACTCCTCAACCCGGAGTCCCTCCTGAAGAAGCAGGAGCTGCAGTAGCTGCGGAATCTTCTACTGGTACATGGACA
ACTGTTTGGACTGATGGACTTACCAGTCTTGATCGCTACAAAGGACGATGCTATCATATCGAGCCTGTTGCTGGAGAAGAAAATCAA
TATATTGCCTATGTAGCTTATCCTTTAGACCTTTTTGAAGAAGGTTCTGTTACTAACATGTTTACTTCTATTGTAGGTAATGTATTTGGT
TTCAAAGCCCTACGAGCTCTACGCTTGGAAGACTTGCGAATTCCCCCTGCTTATTCAAAAACTTTCCAAGGCCCACCTCATGGTATCCA
ATCTGAAAGAGATAAGTTGAACAAATATGGTCGTCCTCTATTGGGATGTACTATTAAACCAAAATTGGGATTATCCGCAAAAAACTA
CGGTAGAGCATGTTATGAATGTCTA

RESULT:

The observations from the three sequences after searching have been analyzed and tabulated.

ROLL NO	9-DIGIT CODE	ACCESSION NO.	MAX SCORE	TOTAL SCORE	QUERY COVER	PERCENTAGE IDENTITY	E - VALUE	ORGANISM	BARCODE REGION
22	QSP006657	NR_025681.1	2499	2499	100%	100.00%	0	Aeromicrobium marinum	rRNA
22	GOH000007	NG_063392.1	3230	3230	100%	100.00%	0	Candida melibiosica	rRNA
22	QMU000540	JQ594519.1	1014	1014	100%	99.82%	0	Rhynchospora sp. MHG-12500	rbcL

RATIONALE:

The reason for selecting the following results out of the plethora of results obtained while searching and identifying it as the target sequences that closely resembles the sequences that have been provided to me is because out of all the results, the results that have been taken stood out with both the maximum percentage identity and maximum query cover and also with the least E-value.

The percentage identity describes how [many characters] identical the query sequence is to the target sequence. The query cover tells us how much of the sequences [query and target sequences] are relative to each other. The higher the percent identity and query cover, the more significant the match is.

Similarly, the lesser the expected value or E value, the more significant the match is. Expected value describes how many times you would expect a match by chance in the chosen database.

This was simple as one organism alone stood out from the crowd satisfying the parameters almost always completely.

Let us take a case where it is possible to have a 99% identity match, but only across 65% of our sequence (query cover). In this case we would have no information on how closely the other 35% of your sequence matches up. This means reviewing the query cover percentage of a BLAST search is just as important as the identity percentage.

The smaller is the query coverage, less data (nucleotides) are been compared and the chance or error (E) is higher. The lower percentage Query Coverage of your query sequence indicates that the overlap with the reference sequence is very low. However, higher percentage Identity indicates that whatever sequence which overlapped, matched very well.

Finding out the organism which relates more to the given sequence can be done efficiently by doing it the following way,

1. First step, is finding out the organisms with the maximum percentage identity (a high Identity value generally falls in the range of 98-100% sequence similarity).

2. Filter out the organisms gotten from the first step based on maximum query cover out of them all (A high Query Cover value for the initial triage is in the 70%+ range).

3. The organism with the highest Query Cover obtained in step 2 is the organism we were looking for.

4. If multiple organisms satisfy the requirements of step 3 (same highest query value and also highest percentage identity, we check for the minimum E value, usually less than 0.01).

So, we understand that if the query coverage is more, the overlap with the reference sequence is very high and when the percentage identity for the organisms with higher query coverage is high, that means whatever sequence overlapped, matched very well.

Easier way to select the organism this way is by setting the"Optimize for" option under "Program Selection" to "Highly similar sequences (megablast)" .

OBSERVATIONS:

SEQUENCE I:

The organism that matched the most with the given sequence is Aeromicrobium marinum [high GC Gram+]. From its barcode region, 16S ribosomal RNA (rRNA) gene, we can identify the organism as a prokaryote.

SEQUENCE II:

The organism that matched the most with the given sequence is Candida melibiosica [budding yeast]. From its barcode region, 18S ribosomal RNA (rRNA) gene, we can identify the organism as a fungi.

SEQUENCE III:

The organism that matched the most with the given sequence is Rhynchospora sp. MHG-12500 [monocots]. From its barcode region, Ribulose-1, 5-Bisphosphate Carboxylase/Oxygenase large subunit (rbcL) gene, found in the chloroplast DNA, we can identify the organism as a plant.

We can see that :

1. Given the barcode region, it is easy for us to classify the organism to its kingdom.

2. Lesser the E – value and greater the percentage identity and sequence cover, the more identical the query sequence is to the target sequence.

REFERENCES:

1. https://blast.ncbi.nlm.nih.gov/Blast.cgi
2. https://ase.tufts.edu/chemistry/walt/sepa/Activities/BLASTpractice.pdf
3. https://mycoflora.org/index.php/resources/faq/41-examining-your-blast-results
4. https://www.researchgate.net/post/What_makes_query_cover_low_in_BLAST-N_search