CSF302 Design and Analysis of Algorithm, ODD Semester 2023, Project

Project Chosen: **Project-4: Design a GUI based solution for longest common subsequence**

For this assignment, you will solve problems based on what you have learned in **this course**.

Instructions

- The project must be developed in Java, and appropriate use of Java Swing must be done wherever required, as mentioned in each program.
- Mention names and IDs of all group members on the top of this document.
- There are 4 questions in this assignment. Attempt any one of them, and mention which project you have chosen on the top of this page.
- **Project submitted after due date will not be evaluated and a score of zero will be awarded.**
- Upload a **word version** of the document.
- The group size should not be more than 3, and the group must be formed from the same section. Inter section groups will not be accepted.
- Plagiarism may lead to award of zero marks.
- Marks distribution of this project is as follows: Appropriate GUI for input and output as mentioned in the question: 6 marks, Expected input and Output as mentioned in the question: 10 marks, Well documented code/Presentation = 4 marks. Total = 20 marks. Overall Weightage of Project in the course is 10% of total marks.
- If you have doubts about any project, please contact the course coordinator.

Due Date: **10 pm, November 24, 2023.**

Submitting this Assignment

You will submit (upload) this assignment in MS Teams. Email/paper submissions will not be accepted.

- Write your answer (codes and snapshots of input and output) after the question in this document.
- Name this document as Project_DAA2023_SAP1_SAP2_SAP3.doc in case group members IDs are SAP1, SAP2 and SAP3 respectively.

CSF302 Design and Analysis of Algorithm, ODD Semester 2023, Project

Project-4: Design a GUI based solution for longest common subsequence Algorithm:

Objective: Develop a code using dynamic programming to generate longest common subsequence of two given strings.

Inputs: Two strings as an input. Use appropriate swing components.

Outputs:

- Two tables, Table 1 and Table 2 as shown in Fig. 4.1. and 4.2.
- Table 1 (as shown in Fig. 4.1) shows length of longest common sequence.
- Table 2 (as shown in Fig. 4.2) the path that is taken to identify the longest common subsequence.
- The Longest common subsequence of the two strings.

X = ABCB
Y = BDCAB

i	j	0	1	2	3	4	5
		Yj	B	D	C	A	B
0	Xi	0	0	0	0	0	0
1	A	0	0	0	0	1	1
2	B	0	1	1	1	1	2
3	C	0	1	1	2	2	2
4	B	0	1	1	2	2	3

Fig.: 4.1

i	j	0	1	2	3	4	5
		Yj	B	D	C	A	B
0	Xi	0	0	0	0	0	0
1	A	0	0	0	0	1	1
2	B	0	1	1	1	1	2
3	C	0	1	1	2	2	2
4	B	0	1	1	2	2	3

LCS (reversed order): B C B

LCS (straight order): B C B

Fig.: 4.2

CSF302 Design and Analysis of Algorithm, ODD Semester 2023, Project

CODE

```
package com.mycompany.daaproject;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class Daaproject implements ActionListener {
   public static JTextField jtf1;
   public static JTextField jtf2;
   public static JLabel jl1;
   public static JLabel jl2;
   public static JLabel jl3;
   public static JLabel jl4;
   public static JLabel jl5;
   public static JButton jb1;
   public static JTextField jt1;
   public static JTextField jt2;
   public JTextArea outputTextArea1;
   public JTextArea outputTextArea2;// Added JTextArea for output
   private JFrame jf;
   Daaproject() {
      jf = new JFrame("DAA Project");
      jtf1 = new JTextField();
      jtf2 = new JTextField();
      jl1 = new JLabel("DAA PROJECT");
      jl2 = new JLabel("Enter the string 1 having alphabets :");
      jl3 = new JLabel("Enter the string 2 having alphabets :");
      jl4 = new JLabel("LCS in forward direction : ");
      jl5 = new JLabel("LCS in backward direction : ");
      jb1 = new JButton("Submit");
      jt1 = new JTextField();
      jt2 = new JTextField();
      jf.setLayout(null);
      jt1.setBounds(470,450,200,20);
      jt2.setBounds(470,500,200,20);
      jb1.setBounds(400, 150, 100, 30);
      jl1.setBounds(420, 5, 300, 50);
      jl2.setBounds(300, 50, 300, 50);
      jl3.setBounds(300, 90, 300, 50);
```

```
        jl4.setBounds(300,430,300,50);
        jl5.setBounds(300,480,300,50);
        jtf1.setBounds(530, 70, 100, 20);
        jtf2.setBounds(530, 110, 100, 20);
        outputTextArea1 = new JTextArea();
        outputTextArea1.setEditable(false); // Make it read-only
        outputTextArea1.setFont(new Font("Monospaced", Font.PLAIN, 12));
        outputTextArea1.setBorder(BorderFactory.createLineBorder(Color.BLACK));
        outputTextArea2 = new JTextArea();
        outputTextArea2.setEditable(false); // Make it read-only
        outputTextArea2.setFont(new Font("Monospaced", Font.PLAIN, 12));
        outputTextArea2.setBorder(BorderFactory.createLineBorder(Color.BLACK));
        JScrollPane scrollPane1 = new JScrollPane(outputTextArea1);
        scrollPane1.setBounds(30, 200, 400, 200);
        JScrollPane scrollPane2 = new JScrollPane(outputTextArea2);
        scrollPane2.setBounds(500, 200, 400, 200);
        jf.add(jl1);
        jf.add(jl2);
        jf.add(jl3);
        jf.add(jl4);
        jf.add(jl5);
        jf.add(jb1);
        jf.add(jtf1);
        jf.add(jtf2);
        jf.add(jt1);
        jf.add(jt2);
        jf.add(scrollPane1);
        jf.add(scrollPane2);
        jb1.addActionListener(this);
        jf.setSize(1000, 600);
        jf.setVisible(true);
        jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    public static void main(String[] args) {
        new Daaproject();
    }
    public void actionPerformed(ActionEvent e) {
        String s = e.getActionCommand();
        if (s.equals("Submit")) {
            String str1 = jtf1.getText();
            String str2 = jtf2.getText();
            String output1 = lcs1(str1, str2);
```

```
        String output2 = lcs2(str1, str2);
        String op1 = findLCS(str1,str2);
        String op2 = findLCSReverse(str1,str2);
        jt1.setText(op1);
        jt2.setText(op2);
        outputTextArea1.setText(output2);
        outputTextArea2.setText(output1);
    }
  }
  private static String findLCS(String str1, String str2) {
    int m = str1.length();
    int n = str2.length();
    int[][] lcsTable = new int[m + 1][n + 1];
    for (int i = 0; i <= m; i++) {
      for (int j = 0; j <= n; j++) {
        if (i == 0 || j == 0)
          lcsTable[i][j] = 0;
        else if (str1.charAt(i - 1) == str2.charAt(j - 1))
          lcsTable[i][j] = lcsTable[i - 1][j - 1] + 1;
        else
          lcsTable[i][j] = Math.max(lcsTable[i - 1][j], lcsTable[i][j - 1]);
      }
    }
    int lcsLength = lcsTable[m][n];
    char[] lcsChars = new char[lcsLength];
    int i = m, j = n;
    while (i > 0 && j > 0) {
      if (str1.charAt(i - 1) == str2.charAt(j - 1)) {
        lcsChars[lcsLength - 1] = str1.charAt(i - 1);
        i--;
        j--;
        lcsLength--;
      } else if (lcsTable[i - 1][j] > lcsTable[i][j - 1]) {
        i--;
      } else {
        j--;
      }
    }

    return new String(lcsChars);
  }
  private static String findLCS1(String str1, String str2) {
```

```
    int m = str1.length();
    int n = str2.length();

    int[][] lcsTable = new int[m + 1][n + 1];
    for (int i = 0; i <= m; i++) {
      for (int j = 0; j <= n; j++) {
        if (i == 0 || j == 0)
          lcsTable[i][j] = 0;
        else if (str1.charAt(i - 1) == str2.charAt(j - 1))
          lcsTable[i][j] = lcsTable[i - 1][j - 1] + 1;
        else
          lcsTable[i][j] = Math.max(lcsTable[i - 1][j], lcsTable[i][j - 1]);
      }
    }
    int lcsLength = lcsTable[m][n];
    char[] lcsChars = new char[lcsLength];
    int i = m, j = n;
    while (i > 0 && j > 0) {
      if (str1.charAt(i - 1) == str2.charAt(j - 1)) {
        lcsChars[lcsLength - 1] = str1.charAt(i - 1);
        i--;
        j--;
        lcsLength--;
      } else if (lcsTable[i - 1][j] > lcsTable[i][j - 1]) {
        i--;
      } else {
        j--;
      }
    }

    // Reverse the obtained LCS for backward direction
    StringBuilder reversedLCS = new StringBuilder();
    for (int k = lcsChars.length - 1; k >= 0; k--) {
      reversedLCS.append(lcsChars[k]);
    }
    return reversedLCS.toString();
  }

  private static String findLCSReverse(String str1, String str2) {
    return findLCS1(str1, str2);
  }
  private static String lcs2(String str1, String str2) {
```

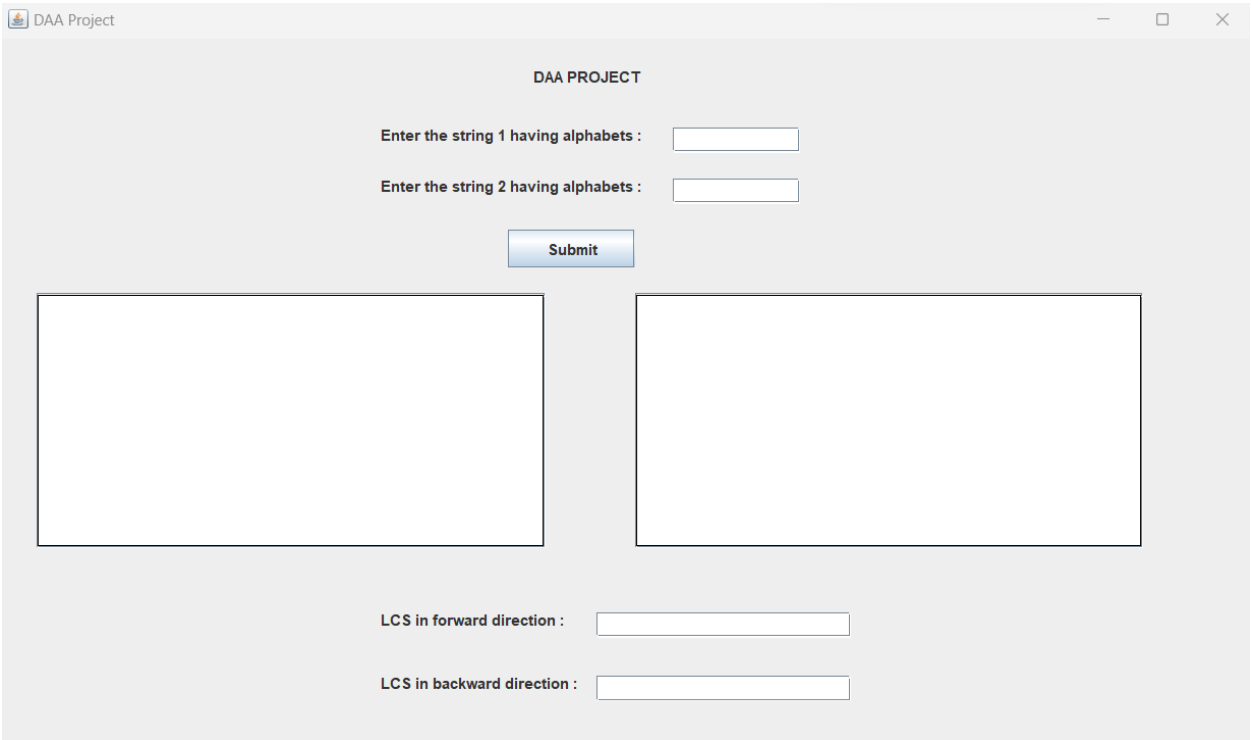
```
    int m = str1.length();
    int n = str2.length();
    int[][] lcsTable = new int[m + 1][n + 1];
    for (int i = 0; i <= m; i++) {
        for (int j = 0; j <= n; j++) {
            if (i == 0 || j == 0)
                lcsTable[i][j] = 0;
            else if (str1.charAt(i - 1) == str2.charAt(j - 1))
                lcsTable[i][j] = lcsTable[i - 1][j - 1] + 1;
            else
                lcsTable[i][j] = Math.max(lcsTable[i - 1][j], lcsTable[i][j - 1]);
        }
    }
    StringBuilder result = new StringBuilder();
    int colHeaderSpace = 9;
    result.append(" ".repeat(colHeaderSpace)); // Initial space before the column headings
    for (int j = 0; j < n; j++) {
        result.append(String.format("%-" + colHeaderSpace + "s", str2.charAt(j)));
    }
    result.append("\n");
    for (int i = 0; i <= m; i++) {
        if (i == 0) result.append(" ".repeat(colHeaderSpace)); // Adjust the space for the row
heading
        else result.append(String.format("%-" + colHeaderSpace + "s", str1.charAt(i - 1)));

        for (int j = 0; j <= n; j++) {
            result.append(String.format("| %2d   ", lcsTable[i][j]));
        }
        result.append("|\n");
        if (i < m) {
            result.append(" ".repeat(colHeaderSpace)); // Adjust the space for the row heading
            for (int j = 0; j <= n; j++) {
                result.append("+------");
                if (j == n) {
                    result.append("+");
                }
            }
            result.append("\n");
        }
    }
    return result.toString();
}
```

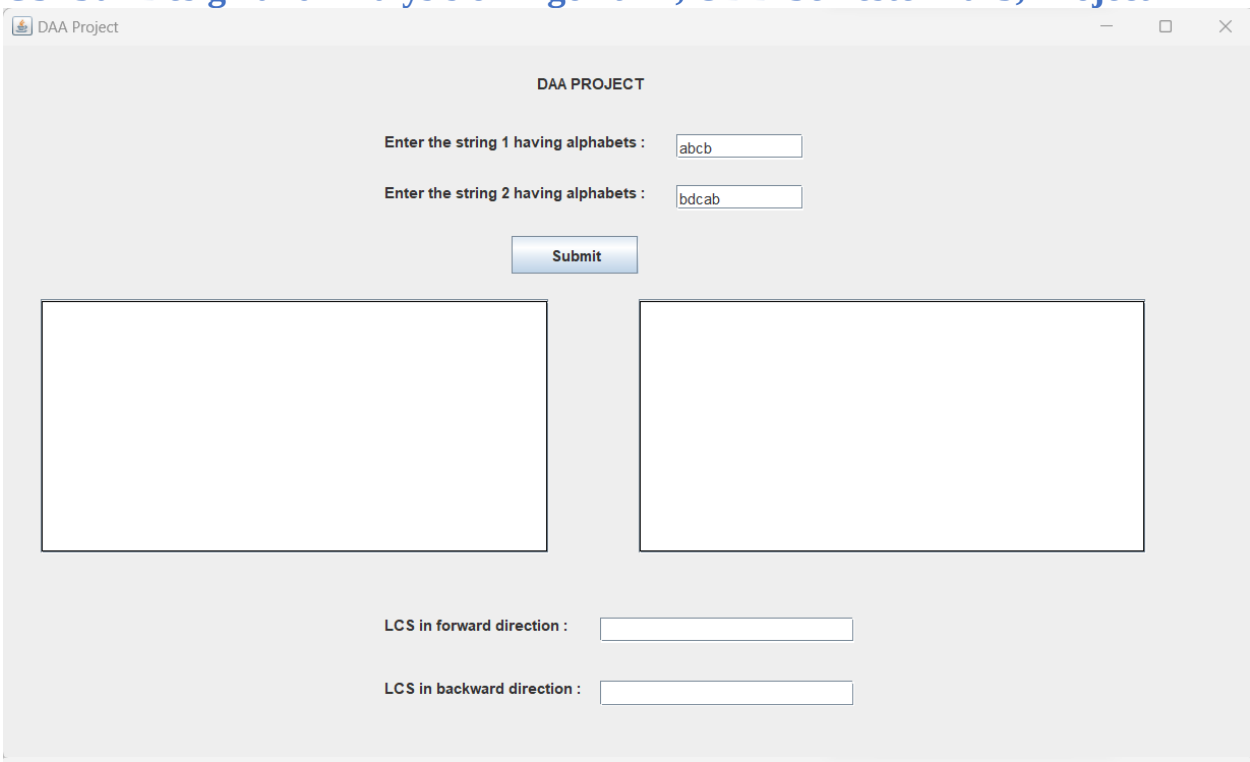
```
private static String lcs1(String str1, String str2) {
 int m = str1.length();
 int n = str2.length();
 int[][] lcsTable = new int[m + 1][n + 1];
 for (int i = 0; i <= m; i++) {
    for (int j = 0; j <= n; j++) {
       if (i == 0 || j == 0)
          lcsTable[i][j] = 0;
       else if (str1.charAt(i - 1) == str2.charAt(j - 1))
          lcsTable[i][j] = lcsTable[i - 1][j - 1] + 1;
       else
          lcsTable[i][j] = Math.max(lcsTable[i - 1][j], lcsTable[i][j - 1]);
    }
 }
 StringBuilder result = new StringBuilder();
 int colHeaderSpace = 9;
 result.append(" ".repeat(colHeaderSpace));
 for (int j = 0; j < n; j++) {
    result.append(String.format("%-" + colHeaderSpace + "s", str2.charAt(j)));
 }
 result.append("\n");
 for (int i = 0; i <= m; i++) {
    if (i == 0) result.append(" ".repeat(colHeaderSpace));
    else result.append(String.format("%-" + colHeaderSpace + "s", str1.charAt(i - 1)));
    for (int j = 0; j <= n; j++) {
       if (i == 0 || j == 0) {
          result.append(String.format("| %2d   ", lcsTable[i][j]));
       } else {
          String arrow = getArrow(i, j, lcsTable);
          result.append(String.format("| %2d%s ", lcsTable[i][j], arrow));
       }
    }
    result.append("|\n");
    if (i < m) {
       result.append(" ".repeat(colHeaderSpace));
       for (int j = 0; j <= n; j++) {
          result.append("+------");
          if (j == n) {
             result.append("+");
          }
       }
       result.append("\n");
```

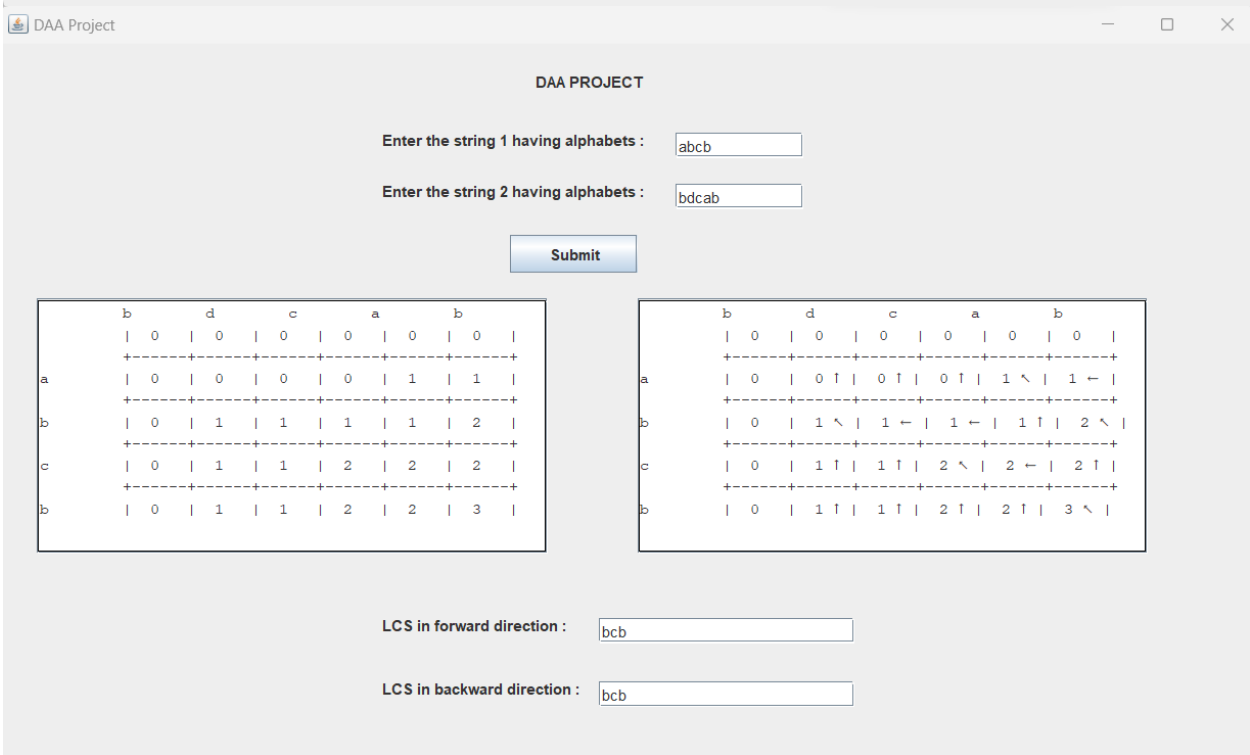
```
        }
    }
    return result.toString();
}
    private static String getArrow(int i, int j, int[][] lcsTable) {
        if (i == 0 || j == 0) {
            return " ";
        } else if (lcsTable[i][j] == lcsTable[i - 1][j]) {
            return " ↑";
        } else if (lcsTable[i][j] == lcsTable[i][j - 1]) {
            return " ←";
        } else {
            return " ↖";
        }
    }
}
```

OUTPUT SCREENSHOT



CSF302 Design and Analysis of Algorithm, ODD Semester 2023, Project





Kartik Gupta