NAME : SOUMYA GAWANDE

CLASS : A1 - B1

ROLL NUMBER : 17

DAA PRACTICAL : 5

TASK1:

Find the similarity between the given X and Y sequence.

X=AGCCCTAAGGGCTACCTAGCTT

Y= GACAGCCTACAAGCGTTAGCTTG

public class Main

{

static void print(Output[][] c , char[] x , int i , int j)

{

if(i == 0 || j == 0)

{

return;

}

if(c[i][j].dirn.equals("d"))

{

print(c , x , i - 1 , j - 1);

System.out.print(x[i - 1]);

}

else

{

if(c[i][j].dirn.equals("u"))

{

print(c , x , i - 1 , j);

}

else

{

print(c , x , i , j - 1);

}

}

}

public static void main(String[] args)

{

String X = "AGCCCTAAGGGCTACCTAGCTT";

String Y = "GACAGCCTACAAGCGTTAGCTTG";

char[] x = X.toCharArray();

char[] y = Y.toCharArray();

Output[][] c = new Output[X.length() + 1][Y.length() + 1];

for (int i = 0; i<= X.length(); i++)

{

for (int j = 0; j <= Y.length(); j++)

{

c[i][j] = new Output();

c[i][j].val = 0;

c[i][j].dirn = "";

}

}

for(int i = 1; i<= X.length(); i++)

{

for(int j = 1; j <= Y.length(); j++)

{

if(x[i - 1] == y[j - 1])

{

c[i][j].val = c[i - 1][j - 1].val + 1;

c[i][j].dirn = "d";

}

else

{

if(c[i - 1][j].val>= c[i][j - 1].val)

{

c[i][j].val = c[i - 1][j].val;

c[i][j].dirn = "u";

}

else

{

c[i][j].val = c[i][j - 1].val;

c[i][j].dirn = "l";

}

}

}

}

print(c , x , X.length() , Y.length());

System.out.println();

}

}

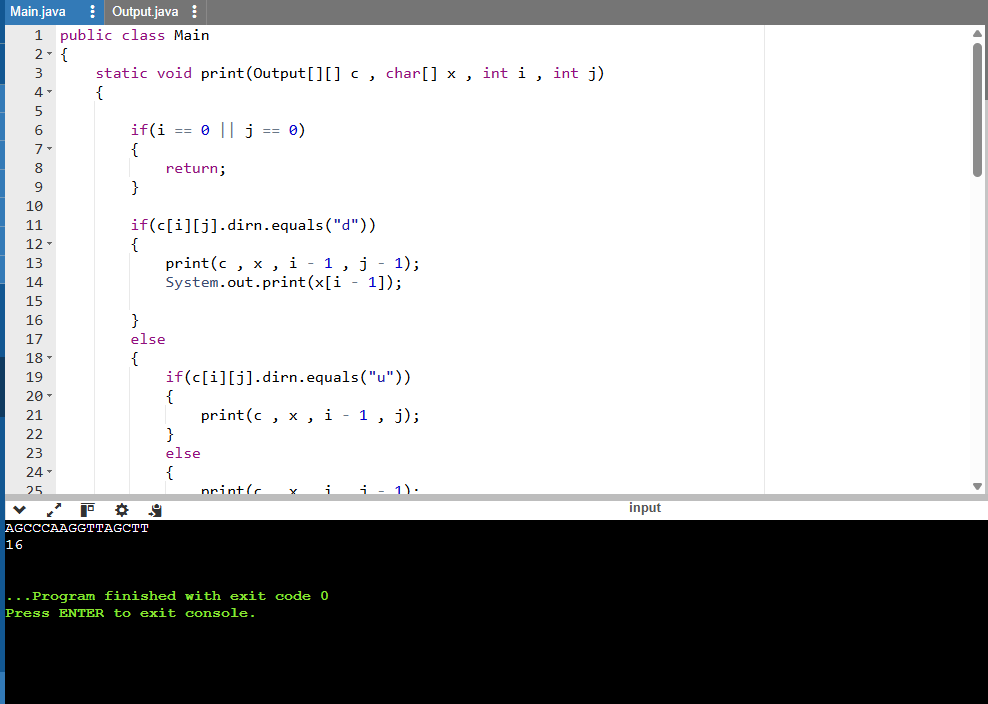
public class Output

{

int val;

String dirn;

}



TASK2:

Find the longest repeating subsequence (LRS). Consider it as a variation of the

longest common subsequence (LCS) problem.

Let the given string be S. You need to find the LRS within S. To use the LCS framework, you

effectively compare S with itself. So, consider string1 = S and string2 = S.

Example:

AABCBDC

LRS= ABC or ABD

public class Main

{

static void print(Output[][] c , char[] x , int i , int j)

{

if(i == 0 || j == 0)

{

return;

}

if(c[i][j].dirn.equals("d"))

{

print(c , x , i - 1 , j - 1);

System.out.print(x[i - 1]);

}

else

{

if(c[i][j].dirn.equals("u"))

{

print(c , x , i - 1 , j);

}

else

{

print(c , x , i , j - 1);

}

}

}

public static void main(String[] args)

{

String S = "AABCBDC";

char[] x = S.toCharArray();

int n = x.length;

Output[][] c = new Output[n + 1][n + 1];

for(int i = 0; i<= n; i++)

{

for(int j = 0; j <= n; j++)

{

c[i][j] = new Output();

c[i][j].val = 0;

c[i][j].dirn = "";

}

}

for(int i = 1; i<= n; i++)

{

for(int j = 1; j <= n; j++)

{

if(x[i - 1] == x[j - 1] &&i != j)

{

c[i][j].val = c[i - 1][j - 1].val + 1;

c[i][j].dirn = "d";

}

else

{

if(c[i - 1][j].val>= c[i][j - 1].val)

{

c[i][j].val = c[i - 1][j].val;

c[i][j].dirn = "u";

}

else

{

c[i][j].val = c[i][j - 1].val;

c[i][j].dirn = "l";

}

}

}

}

print(c, x, n, n);

System.out.println();

}

}

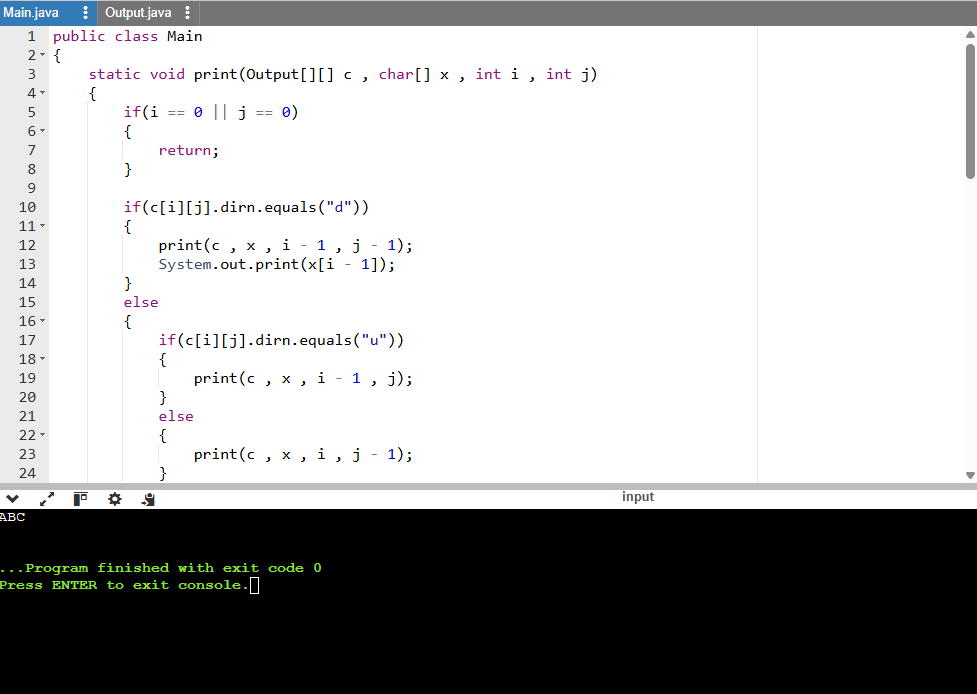
class Output

{

int val;

String dirn;

}



LeetCodeAssesment:

<https://leetcode.com/problems/longest-common-subsequence/description/>

public class Solution

{

public int longestCommonSubsequence(String text1, String text2)

{

int n = text1.length();

int m = text2.length();

int[][] dp = new int[n + 1][m + 1];

for (int i = 1; i <= n; i++)

{

for (int j = 1; j <= m; j++)

{

if (text1.charAt(i - 1) == text2.charAt(j - 1))

{

dp[i][j] = dp[i - 1][j - 1] + 1;

}

else

{

dp[i][j] = Math.max(dp[i - 1][j], dp[i][j - 1]);

}

}

}

return dp[n][m];

}

}

