**Project Report : Largest Common Subsequence**

**Author: Nikhil Bhirud, Sachin Badgujar, Bhsuhan Suhas Deshpande**

**-----------------------------------------------------------------------------------------------------------------------------------------------------------**

**How does your demonstration algorithm improve the understanding of the algorithm?**

Following points make our demonstration better:

1. Our demonstration includes step by step procedure to execute the algorithm.
2. Every iteration and recursion is explained with the visual help of 2d-array structure.
3. We have made sure to mention the line in the code which is related to the results and the values of variables which outputs the results represented visually.
4. Also the code implementation is modular with proper names to functions and variables to clearly understand the purpose of statements.
5. In addition, we have added some pre-requisite information about LCS and its application

**What is Largest Common Subsequence?**

To tell the importance of this algorithm, we would like to start with some fields where it is used.

1. In every IT industry, every time when you need use the tools like Win-merge to find the difference between two files , you are using Longest Common Subsequence (LCS)
2. This probably be the best application of LCS, Scientists use LCS technique to find out similarity beween DNA’s of two species.

Since we have understood how important it is, let me explain what LCS is

It is a string consisting of common characters between two strings in a sequence.

e.g . in the following example, strings are 1)Algorithm 2)Program The possible LCS are

1) ORM

**A L G O R I T H M**

**P R O G R A M**

2) GRM

**A L G O R I T H M**

**P R O G R A M**

**Implementation:** -------------------------------------------------------------------------------------------------------------------------------------------------

//Main Class which invokes the LCS memorized method and prints result.

**public** **class** LCSManager {

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

LCS newLCS = **new** LCS(args[0], args[1]);

//LCS newLCS = new LCS("spanking","amputation");

newLCS.memoized\_lcs();

System.out.println("The Length of LCS is: "+newLCS.getLCSLength());

System.out.println("The LCS string is: "+newLCS.getlcsString());

System.out.println("Below is the LCS matrix: "); newLCS.printMatrix();

}

}

//LCS class which implements the LCS memorized algorithm

## public class LCS {

**private** String s1 = **new** String(); **private** String s2 = **new** String(); **private** **int**[][][] c; **private** **int** len1; **private** **int** len2; **private** **int** lcslength = 0;

**private** String lcsString = "";

**public** LCS(String s1, String s2) {

**this**.s1 = s1; **this**.s2 = s2;

**this**.len1 = s1.length(); **this**.len2 = s2.length();

**this**.c = **new** **int**[len1+1][len2+1][2];

//this.b = new int[l1][l2];

}

**public** **int**[][][] getMatrix() {

## return c;

}

**public** **void** printMatrix() {

System.out.println("c[][][0]"); **for** (**int** i = 0; i<= len1; i++) {

**for** (**int** j = 0; j<= len2; j++) {

System.out.print(c[i][j][0]);

}

System.out.println("");

}

System.out.println("");

System.out.println("c[][][1]");

**for** (**int** i = 0; i<= len1; i++) {

**for** (**int** j = 0; j<= len2; j++) {

System.out.print(c[i][j][1]);

}

System.out.println("");

}

}

**public** **void** memoized\_lcs() {

**for** (**int** i = 0; i<= len1; i++) { ……………………………(1)

**for** (**int** j = 0; j<= len2; j++) {

c[i][j][0] = 0; c[i][j][1] = 3;

//System.out.println("memoized\_lcs"+i+", "+j);

**if** ((i != 0) && (j != 0)) {

lookup\_lcs(i, j);

}

}

}

**this**.lcslength = c[len1][len2][0];

generateLCSString();

}

**private** **void** lookup\_lcs(**int** i, **int** j) { **if**( s1.charAt(i-1) == s2.charAt(j-1) ) {

c[i][j][0] = c[i-1][j-1][0] + 1; ……………………………(3) c[i][j][1] = 0; //diagonal arrow

}

## else {

c[i][j][0] = c[i-1][j-1][0]; ………………………….(2)

c[i][j][1] = 3; //none

}

**if**( c[i-1][j][0] >= c[i][j][0] ) {

c[i][j][0] = c[i-1][j][0];

c[i][j][1] = 1; //up arrow …………………………….(5)

}

**if**( c[i][j-1][0] >= c[i][j][0] ) {

c[i][j][0] = c[i][j-1][0];

c[i][j][1] = 2; //left arrow ………………………………(4)

}

//System.out.println("lookup\_lcs"+i+", "+j);

}

**private** **void** generateLCSString() { ……………………………..(6)

**int** i = len1; **int** j = len2;

**while**( i > 0 && j > 0 ) { **if**( c[i][j][1] == 0 ) { i--; j--;

lcsString = s1.charAt(i)+lcsString;

}

**else** **if**( c[i][j][1] == 1 ) {

i--;

}

**else** **if**( c[i][j][1] == 2 ) {

j--;

}

//System.out.println("generate\_lcs\_string"+i+", "+j);

}

}

**public** **int** getLCSLength() {

**return** lcslength;

}

**public** String getlcsString() {

**return** lcsString;

}

}

--------------------------------------------------------------------------------------

**Understanding the implementation:**

We will continue with the same example and go through the implementation to analyze it.

The two strings are

1) Algorithm 2) Program

The program utilizes a 3d array. The 3d array can be imagined as two 2d arrays stacked one upon other.

In the implementation,

**C [i] [j] [0] is the Island Matrix** and

**C [i] [j] [1] is the Bridge Matrix**

Let’s first understand how to fill these matrices.

**Iteration 0:**

For all values where i=0 or j=0 or both set. (marked by (1) in the implementation)

**C [i] [j] [0]=0 (Island of 0)**

**C [i] [j] [1]=3 (cell to be left empty)**

**Island Matrix: Bridge Matrix:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  | **P** | **R** | **O** | **G** | **R** | **A** | **M** | |  | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | | **A** | **0** |  |  |  |  |  |  |  | | **L** | **0** |  |  |  |  |  |  |  | | **G** | **0** |  |  |  |  |  |  |  | | **O** | **0** |  |  |  |  |  |  |  | | **R** | **0** |  |  |  |  |  |  |  | | **I** | **0** |  |  |  |  |  |  |  | | **T** | **0** |  |  |  |  |  |  |  | | **H** | **0** |  |  |  |  |  |  |  | | **M** | **0** |  |  |  |  |  |  |  | | |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  | **P** | **R** | **O** | **G** | **R** | **A** | **M** | |  | **3** | **3** | **3** | **3** | **3** | **3** | **3** | **3** | | **A** | **3** |  |  |  |  |  |  |  | | **L** | **3** |  |  |  |  |  |  |  | | **G** | **3** |  |  |  |  |  |  |  | | **O** | **3** |  |  |  |  |  |  |  | | **R** | **3** |  |  |  |  |  |  |  | | **I** | **3** |  |  |  |  |  |  |  | | **T** | **3** |  |  |  |  |  |  |  | | **H** | **3** |  |  |  |  |  |  |  | | **M** | **3** |  |  |  |  |  |  |  | |

For i=1, and j=6, Match found.

Hence update (refer implementation (3))

**C [i] [j] [0]= C [i-1] [j-1] [0]+1; (start new island of (1))**

**C [i] [j] [1]=0 (create bridge)**

For all other values of j,

**C [i] [j] [0]=set same value as previous element in the row. (**refer (4) in imp**)**  **C [i] [j] [1]=2 (go left)**

**Island Matrix:** Match found. **Bridge Matrix:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  | **P** | **R** | **O** | **G** | **R** | **A** | **M** | |  | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | | **A** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | | **L** | **0** |  |  |  |  |  |  |  | | **G** | **0** |  |  |  |  |  |  |  | | **O** | **0** |  |  |  |  |  |  |  | | **R** | **0** |  |  |  |  |  |  |  | | **I** | **0** |  |  |  |  |  |  |  | | **T** | **0** |  |  |  |  |  |  |  | | **H** | **0** |  |  |  |  |  |  |  | | **M** | **0** |  |  |  |  |  |  |  | |  |

For i=2, and j=6,

Prev. column value > prev. row value

Hence update

**C [i] [j] [0]= set same value as previous element in the column.** (refer implementation (5)) **C [i] [j] [1]=1 (go up)**

For all other values of j,

**C [i] [j] [0]=set same value as previous element in the row. (**refer (4) in imp**)**  **C [i] [j] [1]=2 (go left)**

**Island Matrix:**   **Bridge Matrix:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  | **P** | **R** | **O** | **G** | **R** | **A** | **M** | |  | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | | **A** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | | **L** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | | **G** | **0** |  |  |  |  |  |  |  | | **O** | **0** |  |  |  |  |  |  |  | | **R** | **0** |  |  |  |  |  |  |  | | **I** | **0** |  |  |  |  |  |  |  | | **T** | **0** |  |  |  |  |  |  |  | | **H** | **0** |  |  |  |  |  |  |  | | **M** | **0** |  |  |  |  |  |  |  | |  |

For i=3, and j=4, Match found.

Hence update (refer implementation (3))

**C [i] [j] [0]= C [i-1] [j-1] [0]+1; (start island of (1))**

**C [i] [j] [1]=0 (create bridge)**

For all other values of j,

**C [i] [j] [0]=set same value as previous element in the row. (**refer (4) in imp**)**  **C [i] [j] [1]=2 (go left)**

**Island Matrix:** Match found. **Bridge Matrix:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  | **P** | **R** | **O** | **G** | **R** | **A** | **M** | |  | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | | **A** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | | **L** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | | **G** | **0** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | | **O** | **0** |  |  |  |  |  |  |  | | **R** | **0** |  |  |  |  |  |  |  | | **I** | **0** |  |  |  |  |  |  |  | | **T** | **0** |  |  |  |  |  |  |  | | **H** | **0** |  |  |  |  |  |  |  | | **M** | **0** |  |  |  |  |  |  |  | |  |

For i=4, and j=3, Match found.

Hence update (refer implementation (3))

**C [i] [j] [0]= C [i-1] [j-1] [0]+1; (start island of (1))**

**C [i] [j] [1]=0 (create bridge)**

For all other values of j,

**C [i] [j] [0]=set same value as previous element in the row. (**refer (4) in imp**)**  **C [i] [j] [1]=2 (go left)**

**Island Matrix:** Match found. **Bridge Matrix:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  | **P** | **R** | **O** | **G** | **R** | **A** | **M** | |  | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | | **A** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | | **L** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | | **G** | **0** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | | **O** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | | **R** | **0** |  |  |  |  |  |  |  | | **I** | **0** |  |  |  |  |  |  |  | | **T** | **0** |  |  |  |  |  |  |  | | **H** | **0** |  |  |  |  |  |  |  | | **M** | **0** |  |  |  |  |  |  |  | |  |

**Iteration 5:**

For i=5, and j=2,5, Match found.

Hence update (refer implementation (3))

**C [i] [j] [0]= C [i-1] [j-1] [0]+1; (start island of (1),(2))**

**C [i] [j] [1]=0 (create bridge)**

For all other values of j,

**C [i] [j] [0]=set same value as previous element in the row. (**refer (4) in imp**)**  **C [i] [j] [1]=2 (go left)**

**Island Matrix:** Match found. **Bridge Matrix:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  | **P** | **R** | **O** | **G** | **R** | **A** | **M** | |  | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | | **A** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | | **L** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | | **G** | **0** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | | **O** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | | **R** | **0** | **0** | **1** | **1** | **1** | **2** | **2** | **2** | | **I** | **0** |  |  |  |  |  |  |  | | **T** | **0** |  |  |  |  |  |  |  | | **H** | **0** |  |  |  |  |  |  |  | | **M** | **0** |  |  |  |  |  |  |  | |  |

**Iterations 6, 7, 8:**

For i=6, 7, 8 and j=2, 5,

Prev. column value > prev. row value

Hence update

**C [i] [j] [0]= set same value as previous element in the column.** (refer implementation (5)) **C [i] [j] [1]=1 (go up)**

For all other values of j,

**C [i] [j] [0]=set same value as previous element in the row. (**refer (4) in imp**)**  **C [i] [j] [1]=2 (go left)**

**Island Matrix: Bridge Matrix:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  | **P** | **R** | **O** | **G** | **R** | **A** | **M** | |  | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | | **A** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | | **L** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | | **G** | **0** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | | **O** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | | **R** | **0** | **0** | **1** | **1** | **1** | **2** | **2** | **2** | | **I** | **0** | **0** | **1** | **1** | **1** | **2** | **2** | **2** | | **T** | **0** | **0** | **1** | **1** | **1** | **2** | **2** | **2** | | **H** | **0** | **0** | **1** | **1** | **1** | **2** | **2** | **2** | | **M** | **0** |  |  |  |  |  |  |  | |  |

**Iterations 9:**

For i=9, and j=9, Match found.

Hence update (refer implementation (3))

**C [i] [j] [0]= C [i-1] [j-1] [0]+1; (start island of (3))**

**C [i] [j] [1]=0 (create bridge)**

For i=9 and j=2, 5,

Prev. column value > prev. row value

Hence update

**C [i] [j] [0]= set same value as previous element in the column.** (refer implementation (5)) **C [i] [j] [1]=1 (go up)**

For all other values of j,

**C [i] [j] [0]=set same value as previous element in the row. (**refer (4) in imp**)**  **C [i] [j] [1]=2 (go left)**

**Island Matrix:** Match found. **Bridge Matrix:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  | **P** | **R** | **O** | **G** | **R** | **A** | **M** | |  | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | | **A** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | | **L** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | | **G** | **0** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | | **O** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | | **R** | **0** | **0** | **1** | **1** | **1** | **2** | **2** | **2** | | **I** | **0** | **0** | **1** | **1** | **1** | **2** | **2** | **2** | | **T** | **0** | **0** | **1** | **1** | **1** | **2** | **2** | **2** | | **H** | **0** | **0** | **1** | **1** | **1** | **2** | **2** | **2** | | **M** | **0** | **0** | **1** | **1** | **1** | **2** | **2** | **3** | |  |

**Result:**

**Now our matrices are ready. So let’s retrace it to get the result. (refer (6 ) in the implementation)**

**Island Matrix: Bridge Matrix:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  | **P** | **R** | **O** | **G** | **R** | **A** | **M** | |  | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | | **A** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | | **L** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | | **G** | **0** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | | **O** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | | **R** | **0** | **0** | **1** | **1** | **1** | **2** | **2** | **2** | | **I** | **0** | **0** | **1** | **1** | **1** | **2** | **2** | **2** | | **T** | **0** | **0** | **1** | **1** | **1** | **2** | **2** | **2** | | **H** | **0** | **0** | **1** | **1** | **1** | **2** | **2** | **2** | | **M** | **0** | **0** | **1** | **1** | **1** | **2** | **2** | **3** | |  |

**Write down character’s at each bridge.**

**Stop when you reach island 0.**

**Answer: ORM**