The programs aim to solve time series problem in Data Mining without doing the math. All it need is input values straight and the program produces the output for it.

These are 2 programs namely Distancematrix.java and Gamma.java written in java.

1. Program Distancematrix takes input as the row elements first and column elements later and outputs the matrix displaying the square of difference of numbers it stands adjacent to for Distance Matrix. The matrix values are displayed for 2 decimal points which can be changed by altering the values in print statements. Also, the number of rows and columns in the program are hard coded as java dynamic arrays do not exist in java. So before executing the program one needs to edit the number of rows and columns in the line number 9, 10, indicated by int r, int c.
2. Program Gamma also takes input as the row elements first and column elements later and outputs the matrix displaying the sum of square of difference of numbers it stands adjacent to and minimum of number at position [i-1][j], [i][j-1], [i-1][j-1]. The matrix values are displayed for 2 decimal points which can be changed by altering the values in print statements. The program also displays the path from the formed matrix following minimum value from one diagonal end to another. The first row and column are hardcoded to 99.99 in line 54 and 58 in code which is to represent infinity. It needs to be changed if the any of value input in the rows and columns are greater than 99.99. This value needs to be greater than each of the input values from rows and columns. Here also, the number of rows and columns in the program are hard coded as java dynamic arrays do not exist in java. So before executing the program one needs to edit the number of rows and columns in the line number 27,28, indicated by int r, int c.

Input Instructions:

For a table of this format, before executing check the number of rows which are 6 and columns which are 5 against the values of int r and int c in program.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *d*(*i*, *j*) | 0.3 | 0.4 | 0.3 | 0.1 | 0.6 |
| 0.2 |  |  |  |  |  |
| 0.3 |  |  |  |  |  |
| 0.2 |  |  |  |  |  |
| 0.4 |  |  |  |  |  |
| 0.1 |  |  |  |  |  |
| 0.9 |  |  |  |  |  |

The input values would be

0.2 0.3 0.2 0.4 0.1 0.9

0.3 0.4 0.1 0.6

Distancematrix.java

**import** java.util.Scanner;

**public** **class** Distancematrix {

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

System.***out***.println("Change the value of r->row and c->column in line number 9,10 to required input values");

Scanner sc=**new** Scanner(System.***in***);

**int** r=6; // number of rows

**int** c=5; // number of columns

System.***out***.println("Enter all the "+r+" rows elements and then enter all the "+c+" column elements");

**float**[] row=**new** **float**[r];

**float**[] col=**new** **float**[c];

//to store the row element

**for**(**int** i=0;i<r;i++)

{

row[i]=sc.nextFloat();

}

//to store the column element

**for**(**int** j=0;j<c;j++)

{

col[j]=sc.nextFloat();

}

//2D array mat[][] to do the computations

**float**[][] mat=**new** **float**[r][c];

//computations and storing them in array mat[][]

**for**(**int** i=0;i<r;i++)

{

**for**(**int** j=0;j<c;j++)

{

mat[i][j]=(row[i]-col[j])\*(row[i]-col[j]);

}

}

//printing back the result from array mat[][]

System.***out***.printf("\t");

**for**(**int** i=0;i<c;i++)

{

System.***out***.printf("%.2f\t",col[i]);

}

System.***out***.println();

**for**(**int** i=0;i<r;i++)

{

System.***out***.printf("%.2f\t",row[i]);

**for**(**int** j=0;j<c;j++)

{

System.***out***.printf("%.2f\t",mat[i][j]);

}

System.***out***.println();

}

sc.close();

}

}

Snapshot of output

A screenshot of a cell phone

Description automatically generated

Gamma.java

**import** java.util.Scanner;

**public** **class** Gamma {

//function to find minimum of 3 numbers

**static** **float** min(**float** a,**float** b, **float** c)

{

**if**(a<b && a<c)

{

**return** a;

}

**else** **if** (b<c && b<a)

{

**return** b;

}

**else**

{

**return** c;

}

}

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

System.***out***.println("Change the value of r->row and c->column in program in line number 27,28 to required input values");

Scanner sc=**new** Scanner(System.***in***);

**int** r=7; // number of rows

**int** c=8; // number of columns

System.***out***.println("Enter all the "+r+" rows elements and then enter all the "+c+" column elements");

// incrementing the number of to form matrix of 1 extra row and column to have infinity in first row and column

r++;

c++;

**float**[] row=**new** **float**[r];

**float**[] col=**new** **float**[c];

//to store the row element

**for**(**int** i=1;i<r;i++)

{

row[i]=sc.nextFloat();

}

//to store the column element

**for**(**int** j=1;j<c;j++)

{

col[j]=sc.nextFloat();

}

//2D array mat[][] to do the computations

**float**[][] mat=**new** **float**[r][c];

//storing 99.99 which represents infinity in the first row and first column of mat[][]

//this value needs to be changed if the number in the input are higher than this

**for**(**int** i=0;i<r;i++)

{

mat[i][0]=99.99f;

}

**for**(**int** j=0;j<c;j++)

{

mat[0][j]=99.99f;

}

//storing 0 at mat[0][0]

mat[0][0]=0;

//computations and storing them in array mat[][]

**for**(**int** i=1;i<r;i++)

{

**for**(**int** j=1;j<c;j++)

{

mat[i][j]=((row[i]-col[j])\*(row[i]-col[j])) + *min*(mat[i-1][j],mat[i-1][j-1],mat[i][j-1]);

}

}

//printing the array[][]

System.***out***.printf("\t\t");

**for**(**int** i=1;i<c;i++)

{

System.***out***.printf("%.2f\t",col[i]);

}

System.***out***.println();

**for**(**int** i=0;i<7;i++)

{

System.***out***.printf("\_\_\_\_\_\_\_\_\_\_\_\_");

}

System.***out***.println();

**for**(**int** i=0;i<r;i++)

{

**if**(i==0)

{

System.***out***.printf("\t|");

}

**else**

{

System.***out***.printf("%.2f\t|",row[i]);

}

**for**(**int** j=0;j<c;j++)

{

System.***out***.printf("%.2f\t",mat[i][j]);

}

System.***out***.println();

}

**for**(**int** i=0;i<7;i++)

{

System.***out***.printf("-------------");

}

// i and r to point to starting point

**int** i=r-1,j=c-1;

System.***out***.println("\n"+"The path is ");

//finding minimum value from left, top and left top diagonal elements to form the path

**while**(i!=1 || j!=1)

{

System.***out***.printf("( %d,%d ),",i,j);

**if**(mat[i-1][j]<mat[i-1][j-1] && mat[i-1][j]<mat[i][j-1])

{

i=i-1;

}

**else** **if** (mat[i-1][j-1]<mat[i][j-1] && mat[i-1][j-1]<mat[i-1][j])

{

i=i-1;

j=j-1;

}

**else**

{

j=j-1;

}

}

System.***out***.println("( "+i+","+j+" ),");

sc.close();

}

}

Snapshot of output

A close up of a piece of paper

Description automatically generated

A screenshot of a cell phone

Description automatically generated