Literate Programming y Google Code Jam 2020

Álvaro González Sotillo

2020-04-04

Índice

1. CodeJam utilizando *emacs*

Estos son mis programas para la edición de 2020 de CodeJam. Como en mi anterior participación, están hechos con org-babel

2. Vestigium

2.1. Problem

Vestigium means "traceïn Latin. In this problem we work with Latin squares and matrix traces.

The trace of a square matrix is the sum of the values on the main diagonal (which runs from the upper left to the lower right).

An N-by-N square matrix is a Latin square if each cell contains one of N different values, and no value is repeated within a row or a column. In this problem, we will deal only with "natural Latin squaresïn which the N values are the integers between 1 and N.

Given a matrix that contains only integers between 1 and N, we want to compute its trace and check whether it is a natural Latin square. To give some additional information, instead of simply telling us whether the matrix is a natural Latin square or not, please compute the number of rows and the number of columns that contain repeated values. Input

The first line of the input gives the number of test cases, T. T test cases follow. Each starts with a line containing a single integer N: the size of the matrix to explore. Then, N lines follow. The i-th of these lines contains N integers Mi,1, Mi,2 ..., Mi,N. Mi,j is the integer in the i-th row and j-th column of the matrix. Output

For each test case, output one line containing Case #x: k r c, where x is the test case number (starting from 1), k is the trace of the matrix, r is the number of rows of the matrix that contain repeated elements, and c is the number of columns of the matrix that contain repeated elements. Limits Test set 1 (Visible Verdict)

Time limit: 20 seconds per test set. Memory limit: 1GB. 1 T 100. 2 N 100. 1 Mi,j N, for all i, j. Sample

2.2. Input

```
3
4
1 2 3 4
2 1 4 3
3 4 1 2
4 3 2 1
4
2 2 2 2
2 3 2 3
```

```
2 2 2 3
2 2 2 2
3
2 1 3
1 3 2
1 2 3
```

2.3. Output

```
Case #1: 4 0 0
Case #2: 9 4 4
Case #3: 8 0 2
```

In Sample Case #1, the input is a natural Latin square, which means no row or column has repeated elements. All four values in the main diagonal are 1, and so the trace (their sum) is 4.

In Sample Case #2, all rows and columns have repeated elements. Notice that each row or column with repeated elements is counted only once regardless of the number of elements that are repeated or how often they are repeated within the row or column. In addition, notice that some integers in the range 1 through N may be absent from the input.

In Sample Case #3, the leftmost and rightmost columns have repeated elements.

2.4. Mi solución

Importante: La clase que se envía debe llamarse `Solution`

```
object Solution extends App{

  val in = {
    var ret =       new java.util.Scanner(System.in);

    scala.util.Try{
      ret = new java.util.Scanner( new java.io.FileInputStream("vestigium.in"))
    }

    ret
  }

  def readInt = in.nextInt()

  val T = readInt;
  for( t <- 1 to T ){
    val N = readInt
    val rows = Array.fill(N)( new Array[Int](N) )
    val columns = Array.fill(N)( new Array[Int](N) )

    for( row <- 0 until N ; col <- 0 until N ){
      val v = readInt
      rows(row)(col) = v
      columns(col)(row) = v
    }

    def repeated(rowOrColumn: Array[Int]) = {
      val groups = rowOrColumn.groupBy(v=>v)
      val filtered = groups.filter{ case (k,v) => v.size > 1 }
      filtered.size > 0
    }

    val r = rows.filter( repeated ).size
    val c = columns.filter(repeated).size

    val k = (0 until N).map( i => rows(i)(i) ).sum

    println( s"Case␣#$t:␣$k␣$r␣$c" )
  }
}
```