Fun with 2D Arrays

Special notes: This problem is worth 11 points.

In this problem you are to complete three methods in the `FunWith2Darrays` class. The three methods are `isArrowHeadArray`, `isIntegerGeneralizedPermutationArray` and the `isMongeArray` method.

The `isArrowHeadArray` method has a single `int[] [] isArrow` parameter and returns a `boolean`. `isArrowHeadArray` returns `true` is the parameter `isArrow` is an Arrowhead matrix and `false` if `isArrow` is not an Arrowhead matrix.

According to Wikipedia: In the mathematical field of linear algebra, an arrowhead matrix is a square matrix containing zeros in all entries except for the first row, first column, and main diagonal. In other words, the matrix has the form

$$A = \begin{bmatrix} * & * & * & * \\ * & * & 0 & 0 \\ * & 0 & * & 0 \\ * & 0 & 0 & * \end{bmatrix}$$

Where * is any non-zero value.

In writing the `isArrowHeadArray` method, you may assume:

- `isArrow[i].length == isArrow[k].length`
  for all `i, k,` with `0 <= i, j < isArrow.length`
- `isArrow.length > 0 && isArrow[0].length > 0`

The following table shows the results of two calls of the `isArrowHeadArray` method.

The following code	Returns

`FunWith2DArrays ma = new FunWith2DArrays();`
`int[][] isArrow = { {0, 0, 0, 0},`
`{0, 0, 16, 29},`
`{0, 28, 0, 34},`
`{0, 13, 6, 0} };`
`ma.isArrowHeadArray( isArrow );`
	`false`

`FunWith2DArrays ma = new FunWith2DArrays();`
`int[][] isArrow1 = { {10, 11, 12, -3},`
`{20, -1, 0, 0},`
`{30, 0, 33, 0},`
`{40, 0, 0, 9} };`
`ma.isArrowHeadArray( isArrow1 );`
	`true`

The `isIntegerGeneralizedPermutationArray` method has a single `int[] [] gpa` parameter and returns a `boolean`. `isIntegerGeneralizedPermutationArray` returns `true` if the parameter `gpa` is a Generalized Permutation Matrix (with integer elements) and false if `gpa` is not a Generalized Permutation Matrix.

According to Wikipedia: In mathematics, a generalized permutation matrix is a matrix with the same nonzero pattern as a permutation matrix, i.e. there is exactly one nonzero entry in each row and each column. Unlike a permutation matrix, where the nonzero entry must be 1, in a generalized permutation matrix the nonzero entry can be any nonzero value. An example of a generalized permutation matrix is

$$\begin{bmatrix} 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -5 \\ 10 & 0 & 0 & 0 \\ 0 & 5 & 0 & 1 \end{bmatrix}$$

In writing the `isIntegerGeneralizedPermutationArray` method, you may assume:

- `gpa[i].length == gpa{k].length`
  for all `i, k,` with `0 <= i, j < gpa.length`
- `gpa.length > 0 && gpa[0].length > 0`

The following code shows the results of a call to the `isIntegerGeneralizedPermutationArray` method.

The following code	Returns

`FunWith2DArrays ma = new FunWith2DArrays();`
 	
`int[][] isgpm1 = { { 0, 0, -2, 0},`
`{ 0, 0, 0, -5},`
`{10, 0, 0, 0},`
`{ 0, 5, 0, 0} };`
`ma.isIntegerGeneralizedPermutationArray( isgpm1 ) );`
	`true`

The `isMongeArray` method has a single `int[] [] ma` parameter and returns a `boolean`. `isMongeArray` returns `true` if the parameter `ma` is a Monge Matrix (with integer elements) and false if `ma` is not a Monge Matrix.

According to Wikipedia: Monge arrays, or Monge matrices, are mathematical objects named for their discoverer, the French mathematician Gaspard Monge.

An m-by-n matrix is said to be a Monge array if, for all i, j, k, l such that $0 \le i < k < m$ and $0 \le j < l < n$ one obtains

$$A[i, j] + A[k, \ell] \le A[i, \ell] + A[k, j].$$

So whenever we pick two rows and two columns of a Monge array (a 2 × 2 sub-matrix) and consider the four elements at the intersection points, the sum of the upper-left and lower right elements (across the main diagonal) is less than or equal to the sum of the lower-left and upper-right elements (across the antidiagonal).

This matrix is a Monge array:

$$\begin{bmatrix} 10 & 17 & 13 & 28 & 23 \\ 17 & 22 & 16 & 29 & 23 \\ 24 & 28 & 22 & 34 & 24 \\ 11 & 13 & 6 & 17 & 7 \\ 45 & 44 & 32 & 37 & 23 \\ 36 & 33 & 19 & 21 & 6 \\ 75 & 66 & 51 & 53 & 34 \end{bmatrix}$$

For example, take the intersection of second and fourth row with the first and fifth column. The four elements are:

$$\begin{bmatrix} 17 & 23 \\ 11 & 7 \end{bmatrix}$$

17 + 7 = 24
23 + 11 = 34

The sum of the upper-left and lower right elements is less than or equal to the sum of the lower-left and upper-right elements.

In writing the `isMongeArray` method, you may assume:

- `ma[i].length == ma[k].length`
  for all `i, k,` with `0 <= i, j < ma.length`
- `ma.length > 0  && ma[0].length > 0`

Do NOT assume the 2d array is a square array

The following code shows the results of the `isMongeArray` method.

The following code	Returns

`FunWith2DArrays ma = new FunWith2DArrays();`
`int[][] values = { {10, 17, 13, 28, 23},`
`{17, 22, 16, 29, 23},`
`{24, 28, 22, 34, 24},`
`{11, 13,  6, 17,  7},`
`{45, 44, 32, 37, 23},`
`{36, 33, 19, 21,  6},`
`{75, 66, 51, 53, 34} };`
 	
`ma.isMongeArray( values );`
	`true`