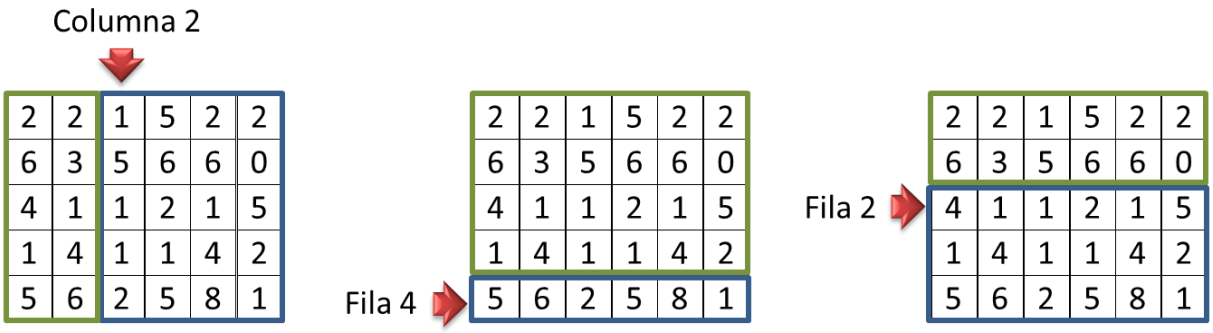
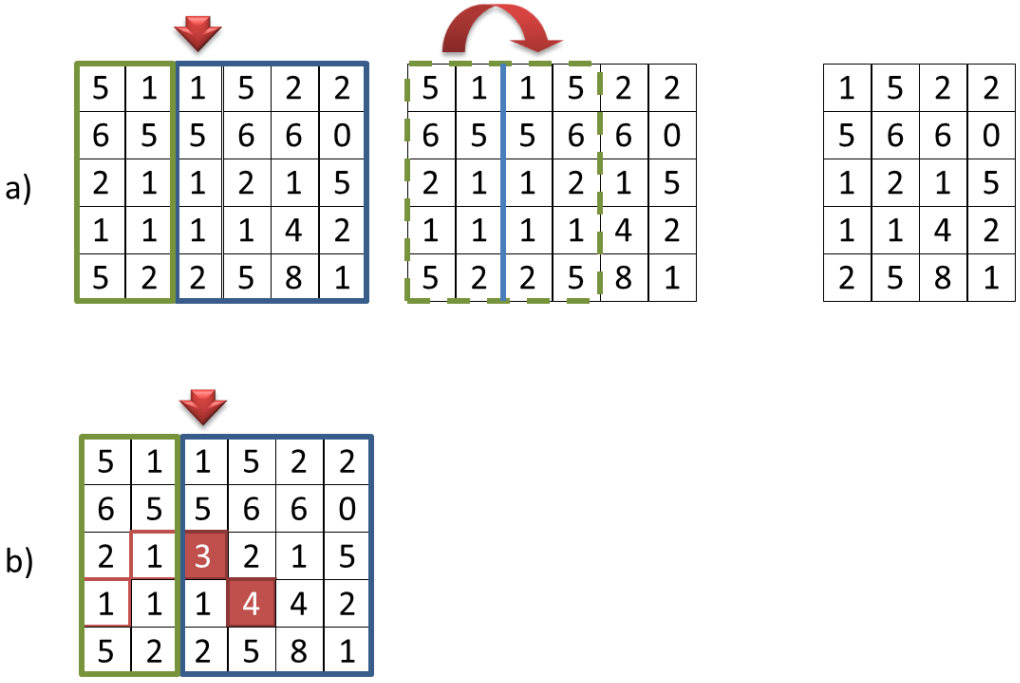
"Envolviendo una matriz"

Sobre una matriz bidimensional de números enteros se define la operación "doblar" de la siguiente forma:

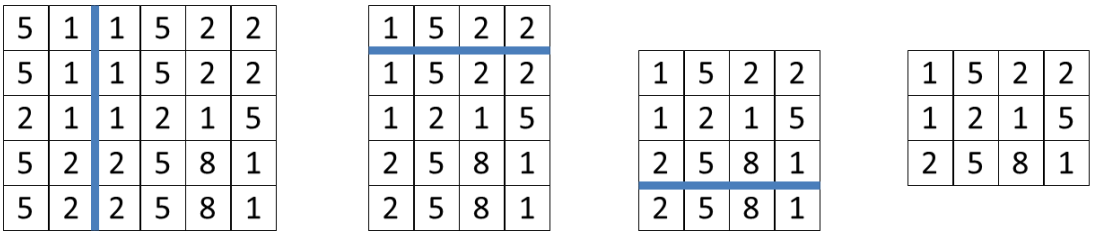
Se puede doblar tanto por una fila como por una columna. La selección de una fila/columna i divide la matriz en dos sub-matrices; la sub-matriz formada por las filas/columnas que están antes de la seleccionada y la sub-matriz formada por las demás. La figura muestra algunas posibles divisiones y la selección de la fila/columna en cada caso.



A partir de la selección de una fila/columna se puede realizar un doblez válido si se cumple que los elementos de la sub-matriz más pequeña que se superponen encima de los de la mayor coinciden. La siguiente figura muestra este proceso, en a) se muestra un doblez válido mientras que en b) se muestra uno que no puede ser realizado por haber uno o más elementos que se superponen y no coinciden. Note que como resultado de "doblar" una matriz se obtiene una nueva sub-matriz igual a la sub-matriz mayor en el proceso de "doblez".



Dada una matriz de enteros se desea determinar la matriz más pequeña (entendiendo por tamaño de la matriz el producto de la cantidad de filas por la cantidad de columnas) que puede ser obtenida aplicándole una serie de "dobleces".



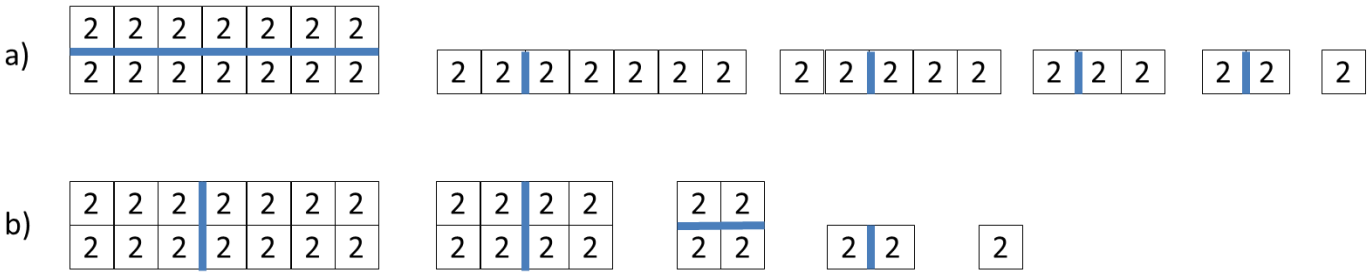
De todas las formas en que se puede envolver la matriz para obtener dicha sub-matriz se desea conocer la forma más “rápida”, es decir la que menos dobleces realiza.

Implemente una DLL de nombre `EnvolviendoMatrices` que tenga la siguiente clase:

```
namespace EnvolviendoMatrices
{
  public class Prueba
  {
    public static Doblez[] Envolver(int[,] matriz)
    {
      // ... Aquí va su implementación
    }
  }
}
```

El tipo `Doblez` está incluido en el ensamblado `Matrices.dll`. Este tipo permite expresar una operación de doblez a partir de un `Indice` (el indice de la fila/columna seleccionada para el doblez) y un valor `Orientacion` (para determinar si la selección fue una fila o una columna).

El siguiente ejemplo muestra una serie de dobleces válidos para la misma matriz. El b) resulta la secuencia óptima.



Aclaraciones

- Se asume como filas de la matriz la dimensión 0 del array bidimensional y como columnas la dimensión 1. Es decir: una matriz tiene `GetLength(0)` cantidad de filas y `GetLength(1)` cantidad de columnas.
- Si se dobla una matriz exactamente por la mitad, se asume que la sub-matriz de abajo/derecha es la que se superpone encima de la de arriba/izquierda. Es decir, queda como resultado la sub-matriz de arriba/izquierda.
- No tiene sentido doblar a partir de la primera fila/columna (no habría nada para superponer).
- Se garantiza que el parámetro `matriz` no será `null`.

En la biblioteca de clases `Matrices` también está presente el tipo Matriz. En este tipo se brindan las funcionalidades que se enuncian a continuación y que usted puede utilizar.

`SubMatriz`: Recibe una matriz, fila inicial, columna inicial, cantidad de filas y cantidad de columnas. Devuelve una nueva matriz con dimensión (cantidad de filas, cantidad de columnas) y los elementos que están en la matriz original a partir de la posición (fila inicial, columna inicial).

`Iguales`: Determina si dos matrices son iguales comparando elemento con elemento.

`InvierteHorizontal`: Obtiene una nueva matriz con todos los elementos de la matriz original pero invirtiendo sus columnas. Es decir, los que aparecían en la columna i de la matriz original aparecen en la columna GetLength(1)-1-i de la resultante.

`InvierteVertical`: Obtiene una nueva matriz con todos los elementos de la matriz original pero invirtiendo sus filas. Es decir, los que aparecían en la fila i de la matriz original aparecen en la fila GetLength(0)-1-i de la resultante.

Le recomendamos tratar de compilar su proyecto lo más frecuentemente posible desde que empiece a trabajar. Entre otros beneficios, esto disminuirá la probabilidad de que pierda su trabajo si ocurre algún fallo eléctrico o de hardware.