EXAMEN MUNDIAL DE PROGRAMACIÓN

CURSO 2011-2012

MATRICES ESPARCIDAS

Una matriz de números enteros se dice esparcida si la cantidad de elementos iguales a cero que contiene es mucho mayor que la cantidad de distintos de cero. Por ejemplo, la siguiente es una matriz esparcida:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

El problema consiste en implementar la clase `MatrizEsparcida`, sin usar un array de dos dimensiones el cual podría ocupar mucha memoria innecesaria, que permita representar este tipo de matrices esparcidas y hacer operaciones con ellas.

Para ello debe implementar los miembros que se relacionan a continuación.

Constructor

```
public MatrizEsparcida(int filas, int columnas){…}
```

el constructor de la clase. Como parámetros se pasan la cantidad de filas y columnas que debe tener la matriz. Inicialmente todos los elementos son iguales a cero. Los valores de los parámetros `filas` y `columnas` deben ser mayores que cero, de lo contrario se debe lanzar como excepción una instancia de `ArgumentException`. Note que en principio no debe haber restricciones sobre el valor máximo de estas cantidades lo cual no sería posible en un array tradicional, la definición de un array en la forma

```
int[,] matriz = new int[100000, 100000];
```

debería dar excepción por falta de memoria. Sin embargo, el siguiente fragmento de código no debe provocar ningún error por falta de memoria

```
MatrizEsparcida matriz = new MatrizEsparcida(100000, 100000);
```

Propiedades

```
public int Filas{get {…}}
```

```
public int Columnas{get {…}}
```

retornan la cantidad de filas y columnas respectivamente que tiene la matriz.

```
public int this[int fila, int columna]{get{…} set{…}}
```

es un *indexer* de la clase mediante el que se puede obtener el valor o asignar elementos en determinadas posiciones de la matriz. Al igual que un array los índices de filas y columnas comienzan en cero y llegan hasta los valores Filas-1 y Columnas-1. Si se accede a una posición incorrecta se debe lanzar `IndexOutOfRangeException`.

La propiedad

```
public int CeldasNoCero{get {…}}
```

retorna como valor la cantidad de elementos distintos de cero que tiene la matriz.

Por otra parte, la propiedad

```
public MatrizEsparcida Transpuesta{get{…}}
```

devuelve una nueva instancia del tipo `MatrizEsparcida` que se corresponda con la matriz transpuesta. Recuerde que $T^{m\times n}$ es la transpuesta de la matriz $M^{n\times m}$ si y solo si $T_{i,j} = M_{j,i}$.

El método:

```
public MatrizEsparcida Suma(MatrizEsparcida m){…}
```

retorna una nueva instancia de `MatrizEsparcida` que representa la suma de la instancia con la que se invoca el método y la que se pasa como parámetro. Recuerde que dos matrices se pueden sumar si tienen la misma dimensión y que si $A^{m\times n} + B^{m\times n} = C^{m\times n}$ entonces $C_{i,j} = A_{i,j} + B_{i,j}$. Si la matriz que se pasa como parámetro no tiene las dimensiones correctas entonces debe lanzar `ArgumentException`. Asuma que el valor del parámetro nunca será `null`.

Considere que dispone del tipo que permite caracterizar a los elementos de una matriz esparcida que sean distintos de 0

```
public struct Celda
{
	public Celda(int fila, int columna, int valor){…}
	public int Fila{get;}
	public int Columna{get;}
	public int Valor{get;}
}
```

la clase `MatrizEsparcida` cuenta con los métodos:

```
public IEnumerable<Celda> ElementosEnColumna(int columna){…}
```

```
public IEnumerable<Celda> ElementosEnFila(int fila){…}
```

que retornan un enumerable con los elementos distintos de cero en la columna o fila respectivamente. Las instancias de Celda deben tener la información correcta en cuanto a la posición que ocupa cada elemento en la matriz y deben estar ordenados por fila, en el resultado del método `ElementosEnColumna` y por columna en `ElementosEnFila`. En ambos casos si el valor de la columna o fila no es un índice válido se debe lanzar `IndexOutOfRangeException`.

Además tenga en cuenta que la clase `MatrizEsparcida` implementa la interfaz `IEnumerable<Celda>` con lo cual debe implementar el método:

```
public IEnumerator<Celda> GetEnumerator(){…}
```

el cual debe retornar un enumerador que devuelva por filas todos los elementos distintos de cero. Los elementos de una misma fila deben estar ordenados por columna..

Tenga presente que las dimensiones de las matrices esparcidas pueden ser muy grandes, en el orden de los millones y que aun así, las implementaciones que brinde de los miembros de esta clase deben ser eficientes y no demorar un tiempo excesivo. Para completar el ejercicio debe utilizar la plantilla que se proporciona junto con esta orientación.

El siguiente ejemplo muestra alguna de las operaciones con las matrices:

```
MatrizEsparcida m1 = new MatrizEsparcida(10000000, 500000);
m1[100, 230] = 10;
m1[110, 100000] = -3;
m1[10000, 500] = 45;
m1[999999, 345676] = 89;

MatrizEsparcida m2 = new MatrizEsparcida(10000000, 500000);
m2[100, 230] = 56;
m2[80, 90] = -78;
m2[10000, 500] = -45;
m2[880, 123456] = 68;

//Calculando la suma de m1 y m2 e imprimiendo las celdas distintas de cero
Console.WriteLine("Suma");
MatrizEsparcida suma = m1.Suma(m2);
foreach (Celda c in suma)
      Console.WriteLine(“suma[{0}, {1}] = {2}”, c.Fila, c.Columna, c.Valor);

suma[100, 10] = 34;
suma[100, 1000] = 48;


Console.WriteLine();
//Imprimiendo los elementos distintos de cero en la fila 100
Console.WriteLine("Elementos en la fila 100");
foreach (Celda c in suma.ElementosEnFila(100))
      Console.WriteLine(“suma[{0}, {1}] = {2}”, c.Fila, c.Columna, c.Valor);

//Se asigna valor cero a un elemento de la matriz
suma[880, 123456] = 0;

Console.WriteLine();
//Imprimiendo la transpuesta de la matriz suma modificada
Console.WriteLine("Transpuesta");
foreach (Celda c in suma.Transpuesta)
      Console.WriteLine(“t[{0}, {1}] = {2}”, c.Fila, c.Columna, c.Valor);
```

Las operaciones anteriores tienen como salida:

```
Suma
suma[80, 90] = -78
suma[100, 230] = 66
suma[110, 100000] = -3
suma[880, 123456] = 68
suma[999999, 345676] = 89

Elementos en la fila 100
suma[100, 10] = 34
suma[100, 230] = 66
suma[100, 1000] = 48

Transpuesta
t[10, 100] = 34
t[90, 80] = -78
t[230, 100] = 66
t[1000, 100] = 48
t[100000, 110] = -3
t[345676, 999999] = 89
```