Segundo Parcial de Estructura de Datos y Algoritmos

Ejer 1	Ejer 2	Ejer 3	Nota
/5	/3	/2	/10

Duración: 2 horas 30 minutos

Condición Mínima de Aprobación. Deben cumplir estas 2 condiciones:

- Sumar **no menos de cuatro** puntos
- Sumar por lo menos 3 puntos entre el ejercicio 1 y 2.

Muy Importante

Al terminar el examen deberían subir los siguientes 2 archivos, según lo explicado en los ejercicios:

1) Solo las siguientes clases Java: **Quadtree.java** y **AdjacencyListGraph.java** según lo pedido. **Cualquier código auxiliar debe estar dentro de esos 2 archivos.**

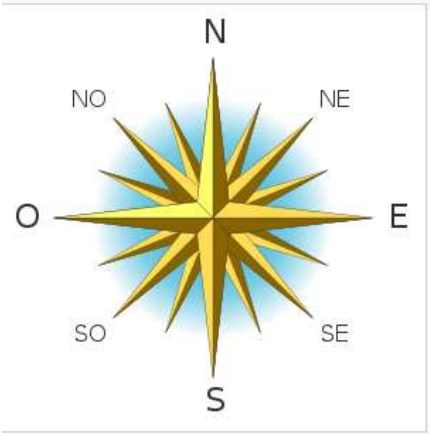
2) Para todos los ejercicios que no consistan en implementar código Java y pidan calcular complejidades, dibujar matrices, completar cuadros, hacer seguimientos, etc. pueden optar por alguna de estas estrategias:
   a. O completar este documento y subirlo también
   b. O directamente resolverlo en hojas de papel y sacarle fotos (formato jpg, png o pdf) y subir todas las imágenes.

Ejercicio 1

Las imágenes satelitales pueden representarse por medio de matrices bidimensionales. Sin embargo, suelen ocupar mucho espacio, por lo que para compactar su información se utiliza un Quadtree.

Un **Quadtree es un árbol no balanceado** que representa puntos en un espacio bidimensional. Sus **nodos internos tiene exactamente 4 hijos** (cada hijo representa uno de los 4 cuadrantes del espacio bidimensional: NO, NE, SO y SE) y no contiene datos (su data es null). Por el contrario, los **nodos hoja** no tiene hijos y posee data diferente a null.

Para construir un Quadtree a partir de una matriz bidimensional 2D x 2D (porque su dimensión debe ser cuadrada y par) se procede analizando sus celdas. Si todos sus valores son iguales entonces se genera una hoja. Caso contrario, se genera un nodo interno con 4 hijos correspondientes a dividir la matriz en 4 cuadrantes. Y se procede con el mismo análisis para decidir si cada una de esas 4 subcuadrantes generan un nodo hoja o un nodo interno con 4 hijos.



Cuadrantes:
NO
NE
SO
SE

(por convención, no se recorren en sentido horario u antihorario sino como una "Z")

LEER TODOS LOS CASOS DE USO ANTES DE EMPEZAR A IMPLEMENTAR!

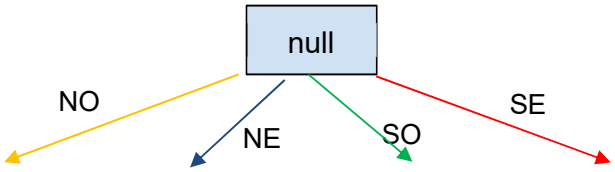
Caso de Uso A

Si la imagen satelital contiene

1	2
3	4

Como no todos los valores son iguales se construye un nodo interno con 4 hijos, cada uno representando los 4 cuadrantes **NO, NE, SO y SE.**

1	2
3	1



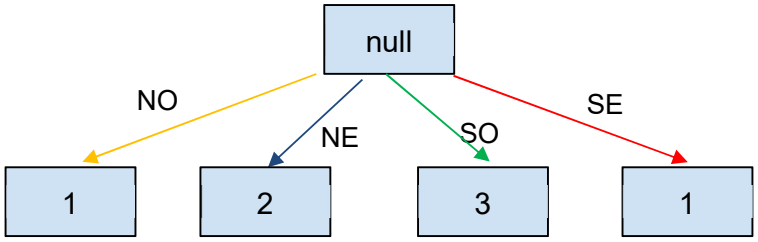
El cuadrante NO (naranja) tiene todas las celdas con el mismo valor, o sea es un nodo hoja con valor 1.

El cuadrante NE (azul) tiene todas las celdas con el mismo valor, o sea es un nodo hoja con valor 2.

El cuadrante SO (verde) tiene todas las celdas con el mismo valor, o sea es un nodo hoja con valor 3.

El cuadrante SE (rojo) tiene todas las celdas con el mismo valor, o sea es un nodo hoja con valor 1.

El Quadtree resultante es el siguiente



Caso de Uso B

Si la imagen satelital contiene

1	**1**
1	**1**

Como todos los valores son iguales se construye un nodo hoja y este es el Quadtree resultante

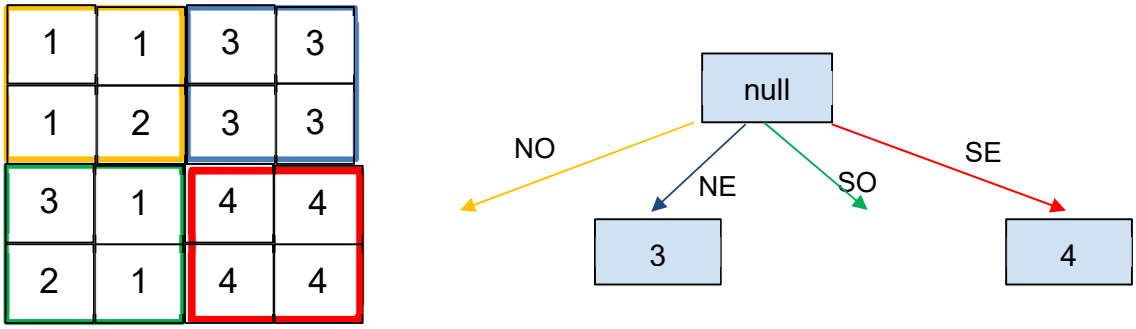
1

Caso de Uso C

Si la imagen satelital contiene

1	**1**	**3**	**3**
1	**2**	**3**	**3**
3	**1**	**4**	**4**
2	**1**	**4**	**4**

Como no todos los valores son iguales se construye un nodo interno con 4 hijos, cada uno representando los 4 cuadrantes **NO, NE, SO y SE.**



El cuadrante “NO” (naranja) no tiene todos los valores iguales en sus celdas, entonces apuntará a un nodo interno (que se subdividirá en 4 subcuadrantes).

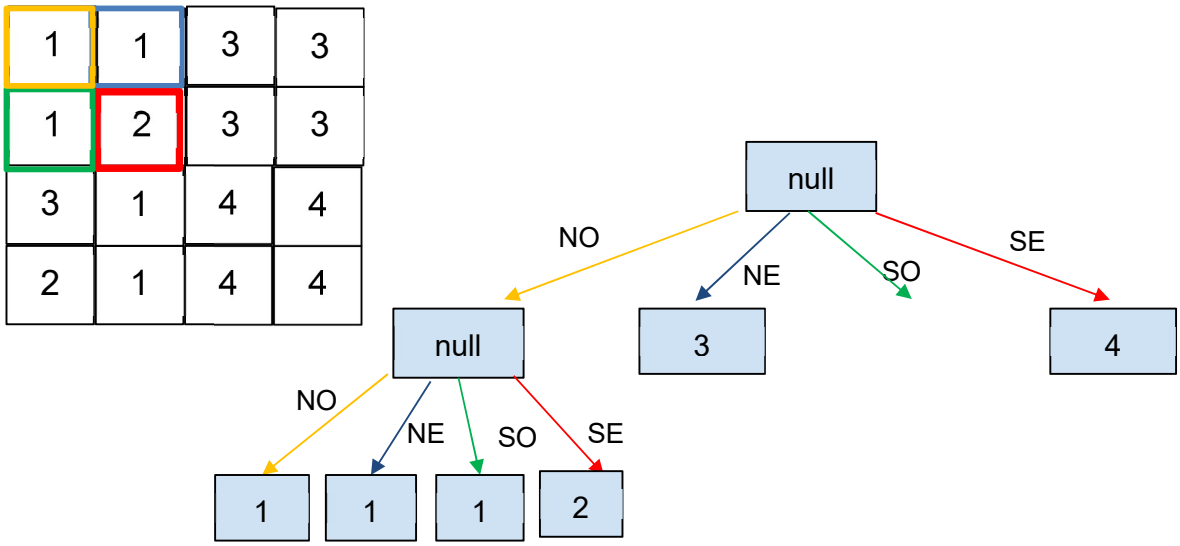
El cuadrante NE (azul) tiene todas las celdas con el mismo valor, o sea es un nodo hoja con valor 3.

El cuadrante SO (verde) no tiene todos los valores iguales en sus celdas, entonces apuntará a un nodo interno (que se subdividirá en 4 subcuadrantes).

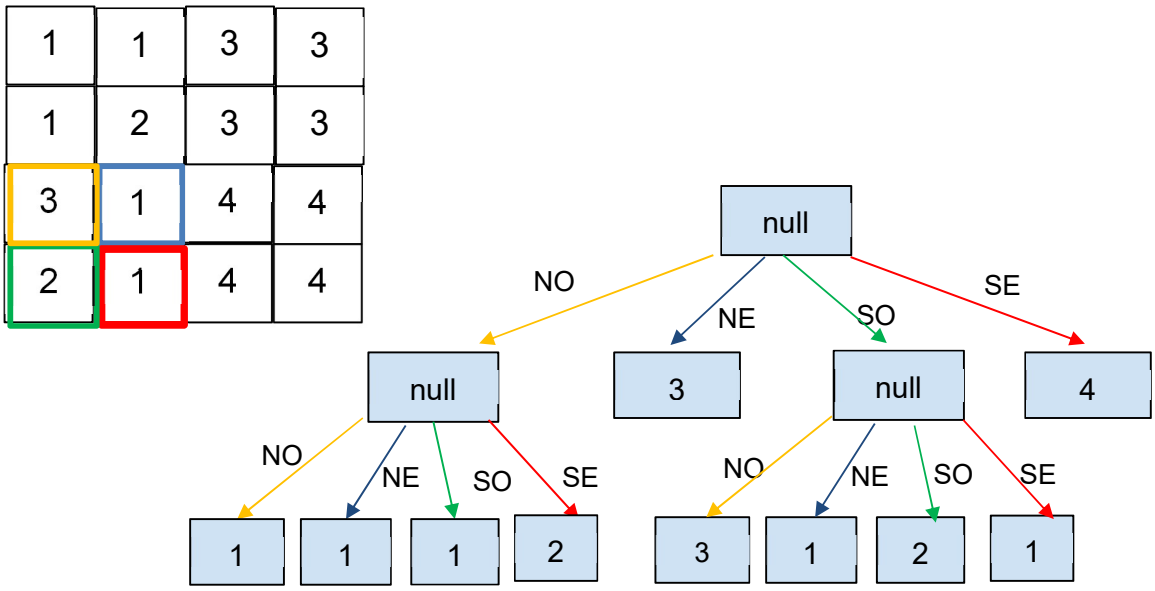
El cuadrante SE (rojo) tiene todas las celdas con el mismo valor, o sea es un nodo hoja con valor 4.

Avancemos entonces a analizar cómo sigue el Quadtree en esos nodos internos.

Respecto del cuadrante “NO”, dijimos que debe volver a particionarse en 4 subcuadrantes. En este caso, cada una de ellas tiene un único valor. Quedando así:



Respecto del cuadrante “SO”, dijimos que debe volver a particionarse en 4 subcuadrantes. En este caso, cada una de ellas tiene un único valor. Quedando así:



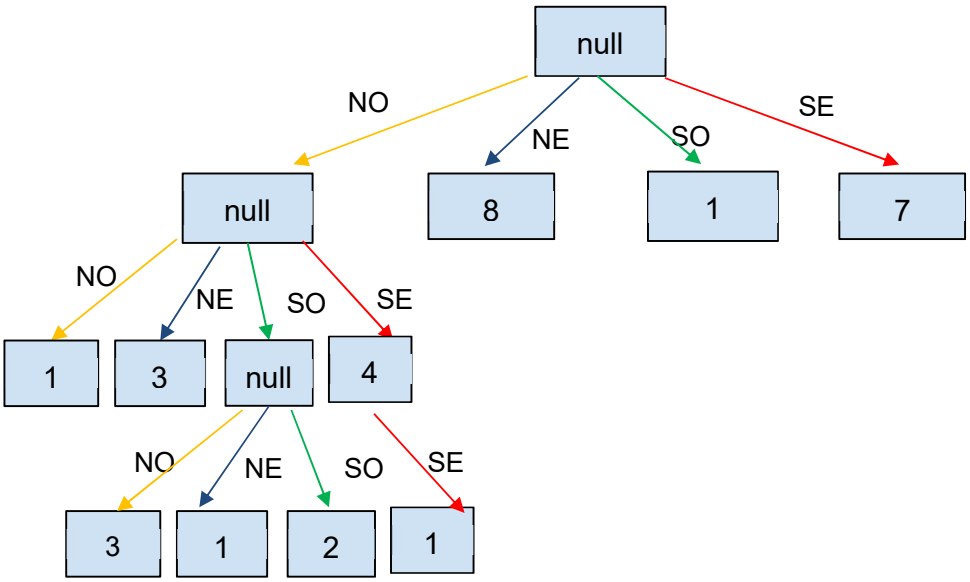
Entonces ese es el Quadtree correspondiente a la matriz dada.

Caso de Uso D

Si la imagen satelital contiene

1	**1**	**3**	**3**	**8**	**8**	**8**	**8**
1	**1**	**3**	**3**	**8**	**8**	**8**	**8**
3	**1**	**4**	**4**	**8**	**8**	**8**	**8**
2	**1**	**4**	**4**	**8**	**8**	**8**	**8**
1	**1**	**1**	**1**	**7**	**7**	**7**	**7**
1	**1**	**1**	**1**	**7**	**7**	**7**	**7**
1	**1**	**1**	**1**	**7**	**7**	**7**	**7**
1	**1**	**1**	**1**	**7**	**7**	**7**	**7**

El resultado final del Quadtree debería ser



Según lo explicado previamente, la siguiente representación del QuadTree que ya se ha definido es la siguiente:

```
public class Quadtree {

       private QTNode root;


      public Quadtree( Integer[][] matrix ){
            if (! checkDimIsSquareAndEven(matrix) )
                  throw new RuntimeException("Invalid Dim");

            // TODO: Completar !!
      }

      public Integer[][] toMatrix( ) {
             // TODO: Completar
      }
```

```
	private class QTNode{
		private Integer data;
		private int dimension;
		public int getDim(){ return dimension; }


		private QTNode upperLeft;
		private QTNode upperRight;
		private QTNode lowerLeft;
		private QTNode lowerRight;
	}
}
```

Adicionalmente a lo explicado anteriormente la clase QTNode incluye el atributo dimensión para guardar la dimensión de la matriz del cuadrante que representa para poder reconstruir correctamente la matriz original si fuera necesario.

Para verlo más claramente si no se guardara la dimensión las matrices:

2	2	2	2
2	2	2	2
2	2	2	2
2	2	2	2

y la matriz:

2	2
2	2

tendrían la misma representación de un solo QTNode con valor 2. Mientras que guardando la dimensión la primera tendría dimension = 4 mientras que la segunda tendría dimension = 2 .

Se pide:

1.1) Terminar de implementar el **constructor del QuadTree** que toma por parámetro una matriz de enteros, según lo explicado previamente. La matriz debe ser de dimensión 2N x 2N (o sea cuadrada y además par) y eso ya está chequeado con el método provisto checkDimIsSquareAndEven. **NO SE PUEDE GUARDAR LA MATRIZ RECIBIDA (ni directamente ni de a pedazos) para evitar reconstruirla en el item 1.2. La idea de usar un Quadtree es compactar y no es aceptable tener la información de la matriz y además el quadtree.**

1.2) Implementar el método **toMatrix** que construye la matriz en base a la información existente en un QuadTree. Asumir que el QuadTree está bien construido (no validar).

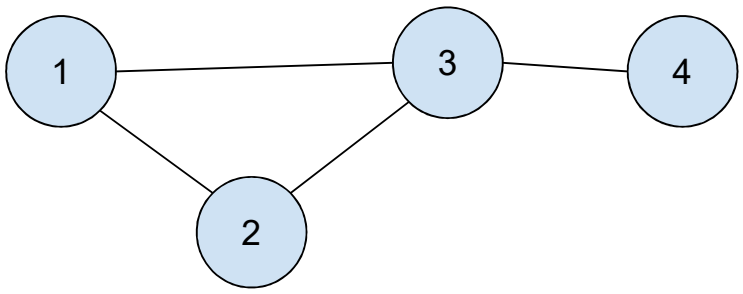
Ejercicio 2

Utilizar el código provisto en este examen, que es un SUBCONJUNTO del grafo con Lista de Adyacencia implementado en clase.

Dado el grafo **no dirigido implementado** con Lista de Adyacencia se pide implementar el método **int connectedComponentsQty()** que determina la cantidad de componentes conexas que contiene el grafo.

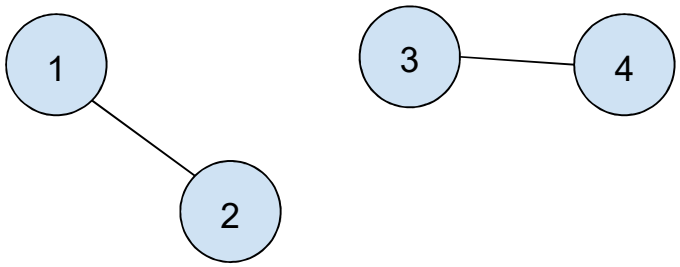
Si fuera dirigido lanzar excepción.

Caso de Uso A



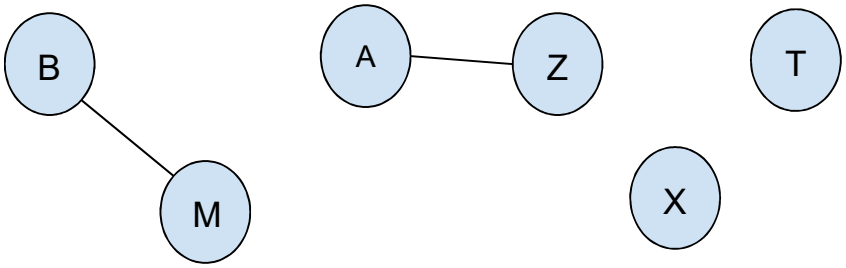
Debe devolver 1

Caso de Uso B



Debe devolver 2

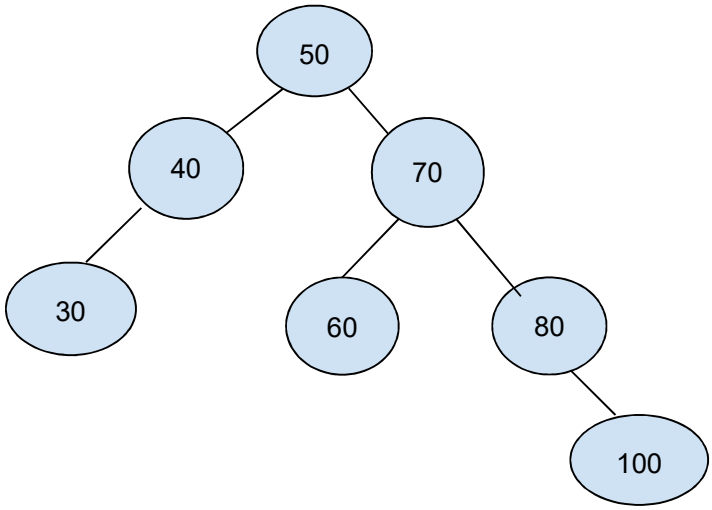
Caso de Uso C



Debe devolver 4

Ejercicio 3

Tomando como comienzo el siguiente árbol AVL que **no acepta repetidos** (los ignora).



Para cada operación se pide:

- Mostrar primero gráficamente dónde se inserta el valor.

- Luego, analizar si algún nodo queda desbalanceado. En caso de que haya que solucionar, indicar claramente de la siguiente manera:

    - Si fuera rotación simple:
        - Decir explícitamente EN PALABRAS que es rotación SIMPLE.
        - Decir explícitamente EN PALABRAS cuál es el pivote y hacia donde se rota (Izq o Der).
        - Mostrar cómo queda el gráfico luego de la rotación simple.

    - Si fuera rotación doble, **desdoblar el análisis en 2 casos sucesivos:**
        - Decir explícitamente EN PALABRAS que es Rotación Doble.
        - Para la primera rotación:
            - Decir explícitamente EN PALABRAS cuál es el pivote y hacia donde se rota (Izq o Der).
            - Mostrar cómo queda el gráfico luego de dicha primera rotación simple.
        - Para la segunda rotación:
            - Decir explícitamente EN PALABRAS cuál es el pivote y hacia donde se rota (Izq o Der).
            - Mostrar cómo queda el gráfico luego de dicha segunda rotación simple.

Si no se realiza todo el análisis anterior, no se considerará la respuesta.

Las operaciones solicitadas en secuencia son:

3.1) Inserción del valor 110

3.2) Inserción del valor 35

3.3) Inserción del valor 90

3.4) Inserción del valor 95