Práctica 2:

Explicación código

El código adjunto para la práctica 2 consiste en 6 clases, de las cuales explicaremos las más importantes algo más en profundidad que las clases que están de apoyo. Podríamos considerar como las clases más importantes complejidad, arbolNArio (la cual ya venía predefinida y hemos aplicado cambios para el toString) y compresiónImagen.

CompresiónImagen:

Podríamos decir que es la clave realmente de toda esta práctica, en esta clase se encuentran los métodos para crear el árbol que almacenará la compresión de la matriz que compone una imagen binaria, cuyo método de creación se encuentra también en esta clase.

El primer método de esta clase a destacar sería el siguiente:

Texto

Descripción generada automáticamente

Éste simplemente crea a partir de un número una matriz de ese orden rellenada de forma aleatoria con ‘B’ y ‘N’, representando el blanco y negro de una imagen binaria. Evidentemente devolverá esta matriz creada.

Antes de explicar el método que aplica divide y vencerás veremos otro del que hace uso:

Texto

Descripción generada automáticamente

Este método está encargado de comprobar si la zona (proporcionada por los atributos que se corresponden con inicio fila,iF, final fila, fF, inicio columna,iC, fin columna,fC) que le es designada de la matriz que se le proporciona. Devolverá true si todo el contenido de la matriz en esa zona es igual

El siguiente método a tener en cuenta sería el que propiamente implementa la estrategia divide y vencerás:

Imagen que contiene Gráfico de dispersión

Descripción generada automáticamente

Este método recibe como argumentos la matriz que antes hemos creado y que implementa la imagen que queremos comprobar; el árbol donde irá almacenada la compresión, y las correspondientes posiciones a comprobar de la matriz dependiendo del cuartil que le corresponda.

El caso base en este caso sería que compararMatriz, método anteriormente analizado, devolviera true, en cuyo caso el árbol se rellenaría con la primera posición de la zona designada pues serían todas las posiciones iguales de esa zona.

El caso general simplemente crea cuatro hijos para el nodo del árbol actual, los inicializa, y llama recursivamente al método con todos y cada uno de los cuatro hijos recién creados

Existe otro método en esta clase llamado complejidad, pero no requiere de un análisis tan extenso pues lo único que hacer es las llamadas a estos tres métodos para poder comprobar su correcto funcionamiento. Cierto es que cabe a destacar la comprobación de que el número proporcionado para crear la matriz sea potencia de 2:



**Complejidad:**

Esta clase cuenta con un único método que se encarga de calcular la complejidad empírica, para ello ejecutamos la parte de divide y vencerás con un incremento en los valores del orden de la matriz hasta 1024x1024, con lo que sacaremos el tiempo que tarda en ejecutarse en nanosegundos (calculamos a partir de este tiempo lo que tarde en ms para mayor exactitud) y lo pasamos como string a una cola que luego nos permitirá exportarlo a un documento en csv para su mejor visionado.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente