Universidad Central de Venezuela.
Facultad de Ciencias.
Algoritmos y Estructura de Datos.
Jiménez Yunelis CI 20.653.996, sección C1
Marval Julyamnis CI 19.255.842, sección C1

Informe Proyecto 3

En éste proyecto de requería comprimir imágenes binarias cargadas al programa. Se debía validar si la imagen era binaria (potencia de 2), lo cual fue resuelto usando una sencilla comparación según su tamaño y sus colores, si solo hay una o dos colores en la imagen la acepta sino no la procesa. Se usó una estructura de Quadtree (árbol de 4 grados) el cual consiste en dividir la imagen en 4 cuadrantes para tratarlos por separado y, así, lograr una mejor compresión.

Se desarrolló un segmento de código donde se genera el árbol de grado 4 y donde se validan sus nodos según su color. Esto se puede verificar en QuadTree.cpp donde se desarrolló todo lo relacionado al árbol (desde su creación usando la información de la imagen y la información de su pre orden, creación de la imagen binaria según la información que se almacena en el árbol, invertir los colores de la imagen, hacer el flip horizontal, toda la información referente a la imagen (cantidad de nodos de cada color, tamaños, etc.), la recursión para ir creando el árbol según su representación de pre orden, la recursión tanto para dividir la imagen binaria en cuadrantes y luego unirlos como para generar una matriz según los datos almacenados en el árbol, y la destrucción del árbol).

En el segmento de código Branch Leaf se realizan las operaciones relacionadas a los nodos del árbol y hace las cargas de los diferentes cuadrantes que se general al hacer la compresión de la imagen y esta el destructor de cada cuadrante luego de haber trabajado en ellos.

En el segmento de BinarieImage se trabaja en la información de la imagen cargada. Su tamaño, el color predominante de la imagen, valida que la imagen sea binaria o monocromática e imprime en la consola una matriz formada por 0 (ceros) y 1 (unos) los cuales representan los colores de la imagen donde 0 es el color base y 1 el color menos predominante.

De acuerdo a la cantidad de nodos presentes en un árbol, es eficiente usar una matriz NxN, a nivel de complejidad de espacio, siempre que la cantidad de nodos del árbol sea baja, ya que al comprimirla y separarla e cuadrantes se debe cargar más cosas a memoria para ser tratadas.