



TALLER 4: ÁRBOLES DE PARTICIÓN

VALENTINA GARCIA ALFONSO

NICOLÁS PADILLA MEDINA

ESTRUCTURAS DE DATOS

Presentado a:

JOHN CORREDOR FRANCO

PONTIFICIA UNIVERSIDAD JAVERIANA, MAYO DEL 2023

1. INTRODUCCIÓN

El enunciado del presente taller indica que es necesario leer archivos que contienen el tamaño de la imagen que se debe decodificar, junto con el recorrido preorden, el cual hay que leerlo para poder generar el árbol de partición correspondiente. Una vez generado el árbol, se procede a generar un archivo PBM (*Portable BitMap format*) que contiene la imagen a blanco y negro que es posible visibilizar. El diseño y resultados de la implementación son presentados a lo largo del documento.

2. TADS IMPLEMENTADOS

TAD Nodo

Datos mínimos:

- pInicial: Coordenada, identifica la coordenada en la que será posicionado ese nodo.
- pFinal: Coordenada, identifica un puntero que apunta a un objeto de tipo Nodo (hijo izquierdo).
- color: Entero, identifica el color que tiene ese nodo (2 para gris, 0 para blanco y 1 para negro).
- hijos: Vector de Nodos, vector de nodos que identifican los hijos de cada nodo (a excepción de las hojas).

Operaciones:

- Nodo(): crea un nodo por defecto (valor en 0, todos sus atributos en 0, y color en 2).
- Nodo(puntoInicial, puntoFinal, color): crea un nodo con los valores pasados como parámetro.
- setSupIzq(color): crea el hijo que se va a ubicar arriba a la izquierda del nodo padre con su respectivo color.
- setSupDer(color): crea el hijo que se va a ubicar arriba a la derecha del nodo padre con su respectivo color.
- setInfIzq(color): crea el hijo que se va a ubicar abajo a la izquierda del nodo padre con su respectivo color.
- setInfDer(color): crea el hijo que se va a ubicar abajo a la derecha del nodo padre con su respectivo color.
- encontrarCoordenada(punto): busca la coordenada o el nodo dentro del árbol, llamando recursivamente si tiene hijos, también se guía de las coordenadas para saber a que hijo corresponde.
- getColor(): retorna el color que tiene ese nodo.
- setColor(color): define el color del nodo correspondiente.
- getHijoSupIzq(): devuelve un puntero que apunta hacia la dirección del hijo superior izquierdo.

- getHijoSupDer(): devuelve un puntero que apunta hacia la dirección del hijo superior derecho.
- getHijoInfIzq(): devuelve un puntero que apunta hacia la dirección del hijo inferior izquierdo.
- getHijoInfDer(): devuelve un puntero que apunta hacia la dirección del hijo inferior derecho.
- getPInicial(): retorna el punto inicial(en coordenadas).
- getPFinal(): retorna el punto final (en coordenadas).
- imprimir(): imprime el número de los nodos hijos,(si su número es 2) si es otro los imprime directamente sin pasar a los hijos.

TAD QuadTree

Datos mínimos:

- raiz: nodo de tipo nodoQuad, representa la raíz del árbol.

Operaciones:

- QuadTree(): crea un árbol con atributos vacíos (puntos iniciales y finales en 0 y el color en 2).
- QuadTree(puntoInicial, puntoFinal, color): crea un árbol dándole como raíz los valores dados.
- imprimirArbol(cadena de caracteres): se crea la imagen con el nombre dado y con el formato PBM.
- llenarArbol(cadena de caracteres): se recibe el nombre del archivo a leer y con base a la lectura se genera el árbol de partición.

3. DIAGRAMA DE TADS

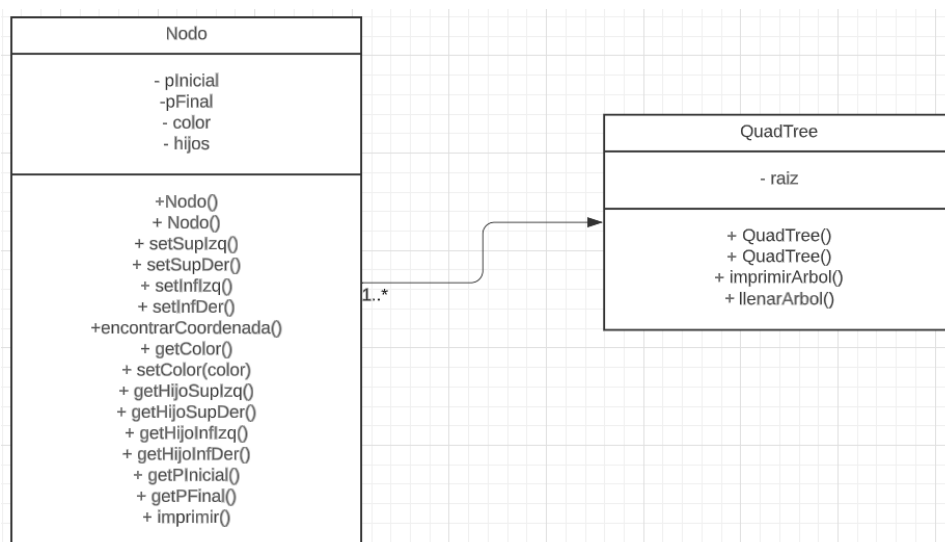


Figura 1. Diagrama de TADS. Fuente: Elaboración propia.

4. IMÁGENES GENERADAS

A. *imagen0*

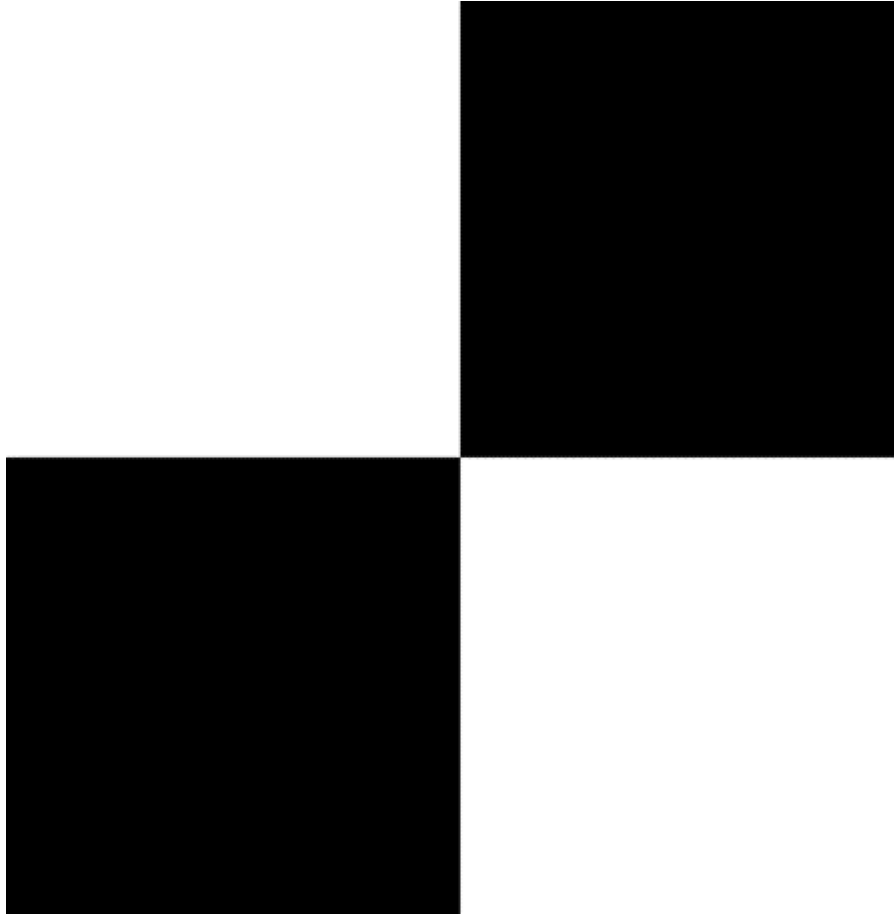


Figura 2. Mini-ajedrez. Fuente: salida del código.

CLa figura anterior representa un ajedrez pequeño, con los colores blanco y negros intercalados y concentrados en una zona determinada de la figura.

B. *imagen1*

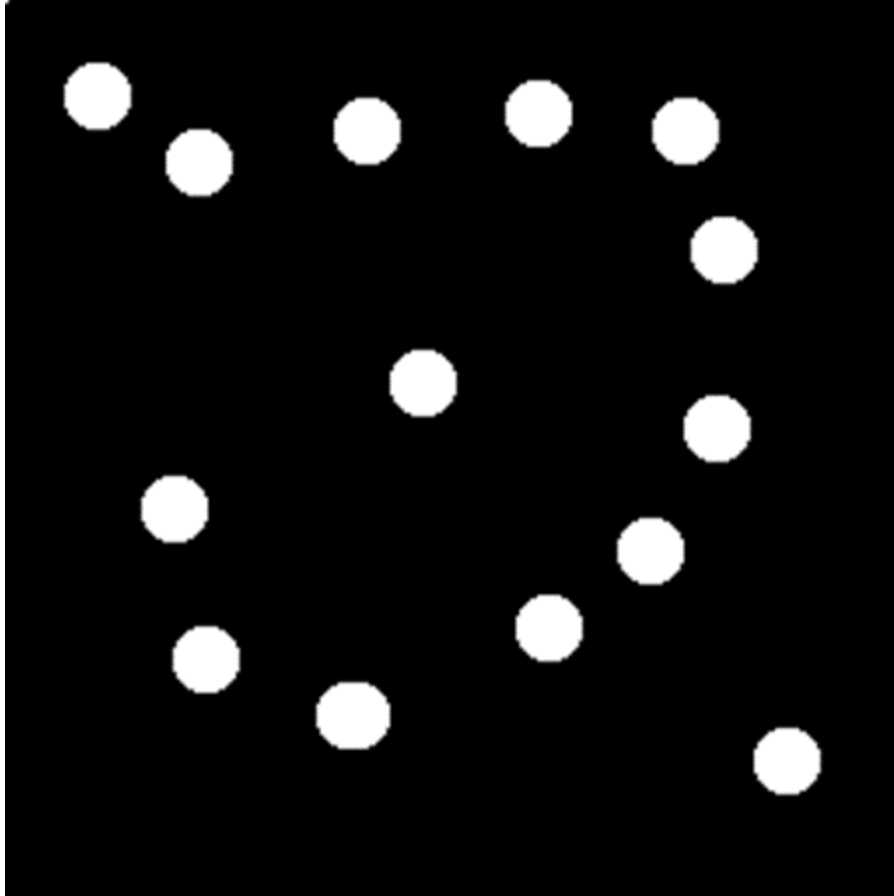


Figura 3. Puntos blancos en una superficie negra. Fuente: salida del código.

En la figura anterior es posible observar puntos blancos generados sobre un fondo negro.

C. imagen2



Figura 4. Homero de Los Simpson. Fuente: salida del código.

En esta figura se puede observar a “Homero” el personaje característico de la serie de Los Simpsons, el cual está sosteniendo una cerveza con la frase “The cause of-and solution to-all of life’s problems”.

D. imagen3

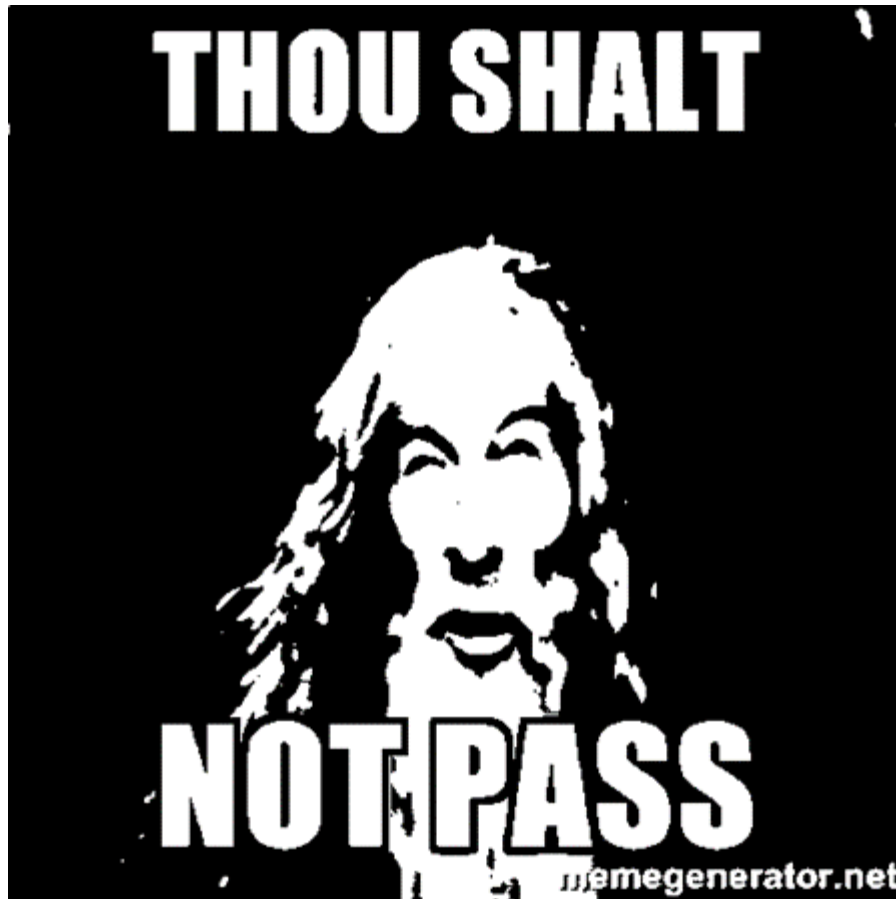


Figura 5. Meme de The Lord of the Rings. Fuente: salida del código.

En la figura anterior, se observa una línea destacable de la saga de The Lord of the Rings, “You shall not pass” dicha por Gandalf.

E. imagen4



Figura 6. Conejo bravo. Fuente: salida del código.

En esta figura se puede observar un conejo con una espada y una pistola.

F. imagen5



Figura 7. Símbolo de los ThunderCats en blanco y negro. Fuente: salida del código.

En la figura 7 se puede evidenciar el símbolo característico de los ThunderCats.

G. imagen6



Figura 8. Badtz-Maru. Fuente: salida del código.

En la figura 8, se reconoce a Badtz-Maru quien es un pingüino y principal amigo de Hello Kitty.

H. imagen



Figura 9. Forma extraña. Fuente: salida del código.

La figura anterior es una imagen de prueba que ejemplifica el funcionamiento de los Quadtrees.

I.imagen8



Figura 10. Carita feliz. Fuente: salida del código.

La figura anterior representa el emoji de una cara feliz '😊'.

J.Imagen9



Figura 11. Casa. Fuente: salida del código.

La figura anterior representa la imagen de una casa.

K.Imagen10

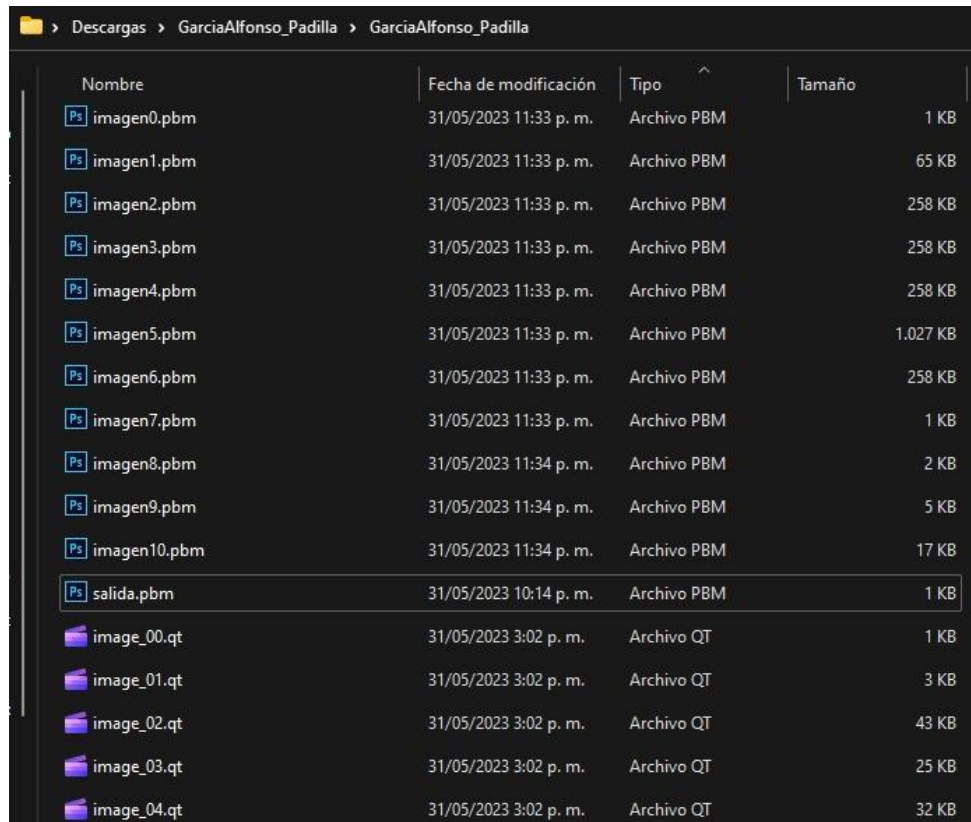


Figura 12. Flecha derecha. Fuente: salida del código.

La figura anterior representa la imagen de una flecha que apunta a la derecha.

5. CONCLUSIÓN

Finalmente, se puede evidenciar la diferencia de tamaño entre los archivos de entrada y de salida, como se muestra a continuación:



Nombre	Fecha de modificación	Tipo	Tamaño
imagen0.pbm	31/05/2023 11:33 p. m.	Archivo PBM	1 KB
imagen1.pbm	31/05/2023 11:33 p. m.	Archivo PBM	65 KB
imagen2.pbm	31/05/2023 11:33 p. m.	Archivo PBM	258 KB
imagen3.pbm	31/05/2023 11:33 p. m.	Archivo PBM	258 KB
imagen4.pbm	31/05/2023 11:33 p. m.	Archivo PBM	258 KB
imagen5.pbm	31/05/2023 11:33 p. m.	Archivo PBM	1.027 KB
imagen6.pbm	31/05/2023 11:33 p. m.	Archivo PBM	258 KB
imagen7.pbm	31/05/2023 11:33 p. m.	Archivo PBM	1 KB
imagen8.pbm	31/05/2023 11:34 p. m.	Archivo PBM	2 KB
imagen9.pbm	31/05/2023 11:34 p. m.	Archivo PBM	5 KB
imagen10.pbm	31/05/2023 11:34 p. m.	Archivo PBM	17 KB
salida.pbm	31/05/2023 10:14 p. m.	Archivo PBM	1 KB
image_00.qt	31/05/2023 3:02 p. m.	Archivo QT	1 KB
image_01.qt	31/05/2023 3:02 p. m.	Archivo QT	3 KB
image_02.qt	31/05/2023 3:02 p. m.	Archivo QT	43 KB
image_03.qt	31/05/2023 3:02 p. m.	Archivo QT	25 KB
image_04.qt	31/05/2023 3:02 p. m.	Archivo QT	32 KB

Figura 13. Archivos de entrada y salida del presente taller. Fuente: elaboración propia.


















Nombre	Fecha de modificación	Tipo	Tamaño
 imagen6.pbm	31/05/2023 11:33 p. m.	Archivo PBM	258 KB
 imagen7.pbm	31/05/2023 11:33 p. m.	Archivo PBM	1 KB
 imagen8.pbm	31/05/2023 11:34 p. m.	Archivo PBM	2 KB
 imagen9.pbm	31/05/2023 11:34 p. m.	Archivo PBM	5 KB
 imagen10.pbm	31/05/2023 11:34 p. m.	Archivo PBM	17 KB
 salida.pbm	31/05/2023 10:14 p. m.	Archivo PBM	1 KB
 image_00.qt	31/05/2023 3:02 p. m.	Archivo QT	1 KB
 image_01.qt	31/05/2023 3:02 p. m.	Archivo QT	3 KB
 image_02.qt	31/05/2023 3:02 p. m.	Archivo QT	43 KB
 image_03.qt	31/05/2023 3:02 p. m.	Archivo QT	25 KB
 image_04.qt	31/05/2023 3:02 p. m.	Archivo QT	32 KB
 image_05.qt	31/05/2023 3:02 p. m.	Archivo QT	40 KB
 image_06.qt	31/05/2023 3:02 p. m.	Archivo QT	17 KB
 image_07.qt	31/05/2023 3:02 p. m.	Archivo QT	1 KB
 image_08.qt	31/05/2023 3:02 p. m.	Archivo QT	1 KB
 image_09.qt	31/05/2023 3:02 p. m.	Archivo QT	2 KB
 image_10.qt	31/05/2023 3:02 p. m.	Archivo QT	3 KB

Figura 14. Archivos de entrada y salida del presente taller. Fuente: elaboración propia.

La diferencia de tamaños es destacable y radica principalmente en que en el archivo de salida se encuentran todos los números binarios que representan la imagen a blanco y negro, siendo considerablemente extensa a comparación de las pocas líneas de entrada del archivo. Una diferencia notoria no está directamente relacionada cuando el tamaño de la imagen es muy grande, esta también se puede observar cuando hay bastantes intercambios entre blanco y negro. De la misma forma cabe resaltar que a diferencia no cambia sustancialmente cuando los tamaños son pequeños, la diferencia de tamaño dependerá de la imagen en particular y de la complejidad de la representación requerida por el quadtree.

Finalmente, a partir de estos resultados podemos decir que los quadrees son lo suficientemente eficientes para decodificar imágenes en especial si son grandes y complejas, esto debido a que este comprime grandes cantidades de información en una estructura recurrente “pequeña”, mientras realiza un recorrido en profundidad del árbol renderizando la imagen, lo que permite una decodificación rápida y una imagen entendible.