Taller 4 (Árboles de partición)

GRUPO 1

Sophia Aristizábal

(Ingeniería de Sistemas y economía)

Juan Felipe Bríñez

(Ingeniería de Sistemas)

Iván Darío Orozco Ibáñez

(Ingeniería de Sistemas e Ingeniería Electrónica)

Facultad de Ingeniería, Pontificia Universidad Javeriana

SISTEMAS 4800-1 (2891): Estructuras de Datos (Teo-Prac)

Andrea del Pilar Rueda Olarte

21 de Febrero del 2023

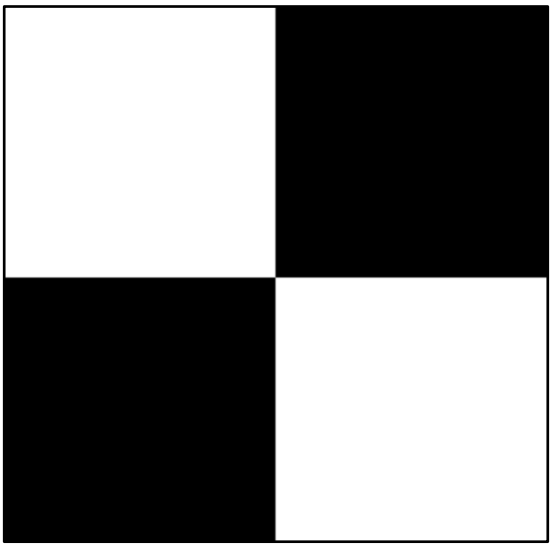
Índice

Desarrollo del taller

1. Análisis diferencia de tamaños

La diferencia de tamaño entre los archivos pbm y el preorden proporcionado es evidente para cada una de las imágenes. Por ejemplo, para la imagen 0, su tamaño es de 8x8 pixeles (64) para la imagen pmb, pero el tamaño de su preorden es de 5 nodos. Así mismo, para la imagen 1 su tamaño es de 256x256 (65,536) pixeles para la imagen pmb, pero el tamaño de su preorden es de 2833 nodos. Por su parte la imagen 2 es de 512X512 (262,144) y su preorden es de 43273 nodos; el de la 3 es de 512X512 (262,144) y su preorden es de 25205; el de la 4 es de 512X512 (262,144) y su preorden es de 31997; el de la 5 es de 1024X1024 (1,048,576) y su preorden es de 40169; el de la 6 es de 512X512 (262,144) y su preorden es de 16381 ; y el de la 7 es de 8X8 (262,144) y su preorden es de 21. De este modo, se puede evidenciar que efectivamente el árbol es lo suficientemente capaz de guardar la imagen con un tamaño menor que su representación en el archivo pmb. Sin embargo, es importante aclarar que esto sucede, porque la imagen tiene suficientes sectores homogéneos para dividirse en menor cantidad de áreas, que la cantidad de píxeles que tiene.

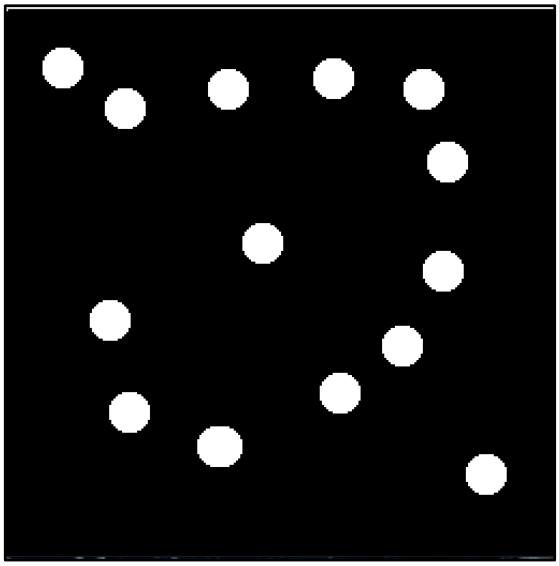
2. Descripción de las imágenes



img 0

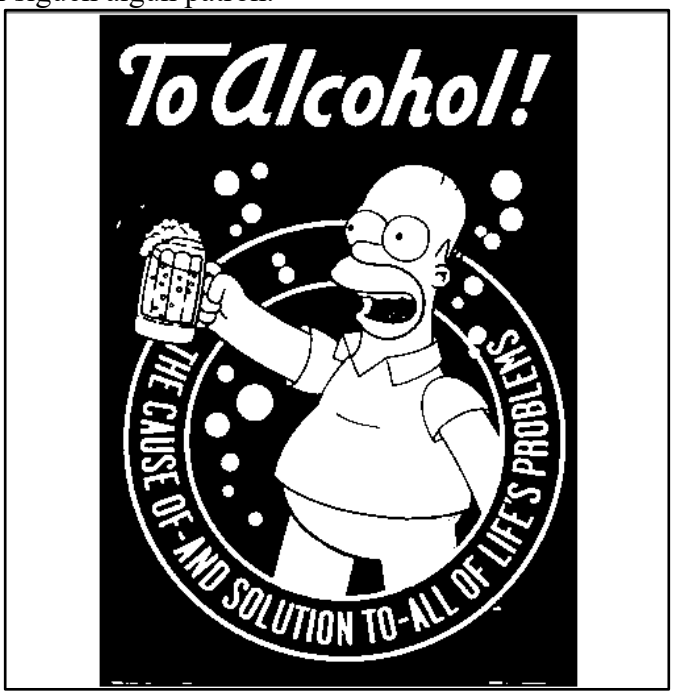
ÁRBOLES DE PARTICIÓN

En esta imagen se puede ver como existe una división de 4 zonas cuadradas, donde dos de ellas son completamente de color negro y dos blancas. Cabe resaltar que cada una de estas zonas tiene la misma cantidad de píxeles respecto a las otras.



img 1

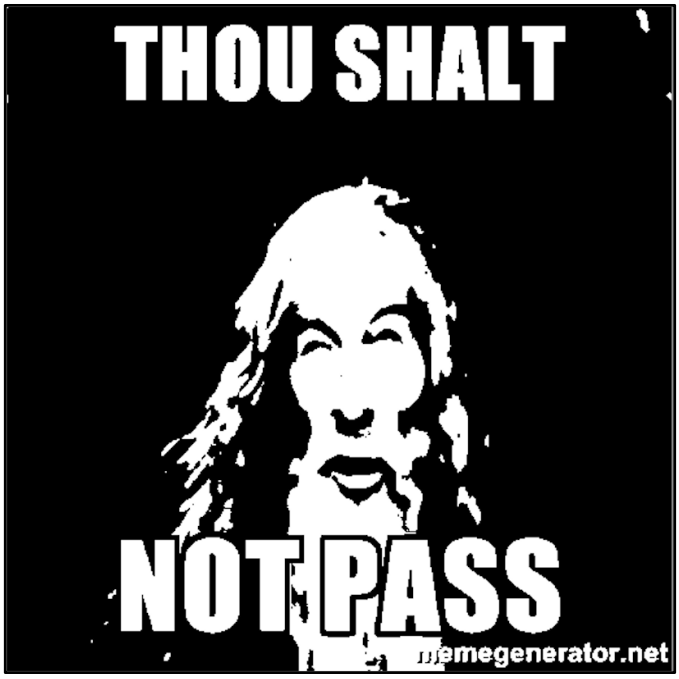
Esta imagen tiene 14 puntos blancos notorios del mismo tamaño, los cuales no forman ninguna figura en específico, ni siguen algún patrón.



img 2

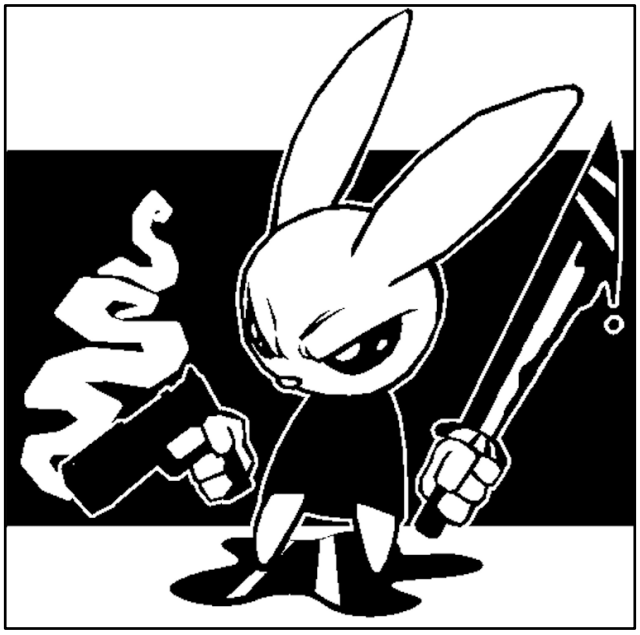
En la imagen anterior se aprecia a el personaje Homero de la caricatura Los Simpson sosteniendo una vaso de cerveza, rodeado de un círculo que tiene un mensaje en inglés el cual es “The cause

of-and solution to-all of life's problems". Adicionalmente arriba del todo hay una frase que dice "To Alcohol!"



img 3

Esta imagen hace alusión a un meme en donde aparece la cara de un señor con pelo y barba larga (Gandalf), junto con un texto que dice "Thou Shalt Not Pass". Así mismo, en la parte inferior derecha hay un hipervínculo.



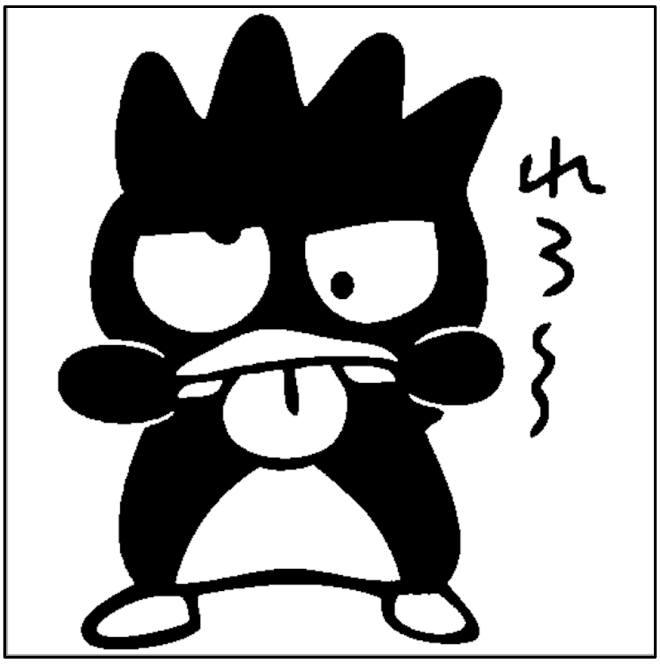
img 4

ÁRBOLES DE PARTICIÓN

Esta imagen expone a un conejo con ceño fruncido que tiene una pistola en una de sus manos y una espada en la otra, la cual tiene rastros de sangre dándonos a saber que posiblemente mató.

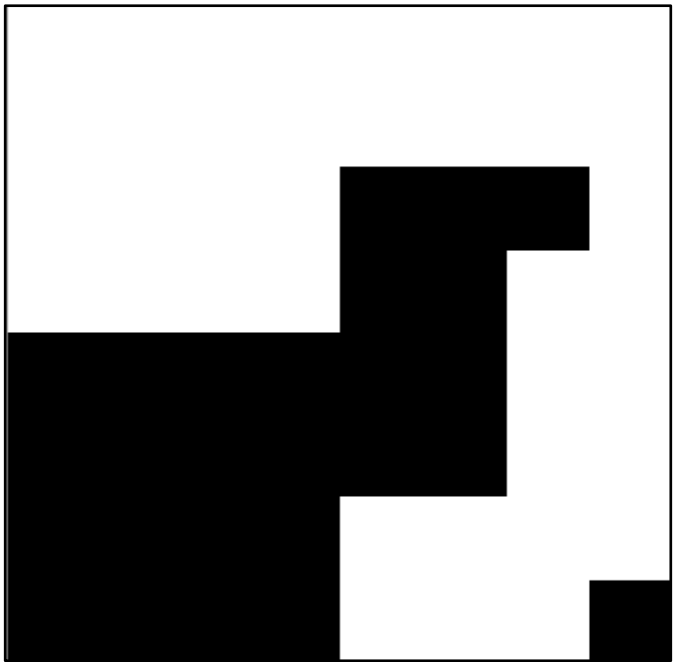
img 5

En esta imagen se expone una pantera de color negro con un circulo a su alrededor de este mismo color.



img 6

La imagen anterior ilustra a un pingüino animado (Badtz Maru) sacando la lengua, junto con unos símbolos a la derecha de él.



img 7

Se puede apreciar en esta imagen cuadrados de diferentes tamaños de color negro, los cuales estan ubicados solo en tres zonas de la imagen, esto si se diviera en cuatro áreas iguales, es decir trazando una linea por la mitad vertical y otra por la horizontal. Una de estas áreas es completamente negra, mientras que las otras dos cuentan con dos cuadros, cada uno de diferente tamaño.

3. **Diseño del árbol QuadTree**

TAD NodoQ

Datos mínimos:

- **`dato`**, tipo plantilla, representa el valor actual del nodo.
- **`hijoSupIzq`**, apuntador a un **NodoQ**, representa el hijo superior izquierdo del nodo.
- **`hijoSupDer`**, apuntador a un **NodoQ**, representa el hijo superior derecho del nodo.
- **`hijoInfIzq`**, apuntador a un **NodoQ**, representa el hijo inferior izquierdo del nodo.
- **`hijoInfDer`**, apuntador a un **NodoQ**, representa el hijo inferior derecho del nodo.

Operaciones:

+ **`NodoQ()`**, crea un nodo con atributos vacíos (sin un valor actual del nodo).
+ **`NodoQ( val )`**, crea un nodo en donde el atributo **`dato_`** tiene un valor de **`val`** y los demás atributos un valor nulo.

+ **~NodoQ()**, elimina la rama que conecta con el nodo actual.
+ **fijarDato(val)**, cambia el valor del **dato_** por **val**.
+ **obtenerDato()**, retorna el valor actual que tiene **dato_** dentro del nodo.
+ **obtenerHijoSupIzq()**, retorna un **NodoQ** con el hijo superior izquierdo que tiene **hijoSupIzq**.
+ **obtenerHijoSupDer()**, retorna un **NodoQ** con el hijo superior derecho que tiene **hijoSupDer**.
+ **obtenerHijoInfIzq()**, retorna un **NodoQ** con el hijo inferior izquierdo que tiene **hijoInfIzq**.
+ **obtenerHijoInfDer()**, retorna un **NodoQ** con el hijo inferior derecho que tiene **hijoInfDer**.
+ **fijarHijoSupIzq(val)**, le crea un hijo superior izquierdo al nodo actual. Se le asigna por defecto al hijo superior izquierdo el valor del **dato_** por **val**.
+ **fijarHijoSupDer(val)**, le crea un hijo superior derecho al nodo actual. Se le asigna por defecto al hijo superior derecho el valor del **dato_** por **val**.
+ **fijarHijoInfIzq(val)**, le crea un hijo inferior izquierdo al nodo actual. Se le asigna por defecto al hijo inferior izquierdo el valor del **dato_** por **val**.
+ **fijarHijoInfDer(val)**, le crea un hijo inferior derecho al nodo actual. Se le asigna por defecto al hijo inferior derecho el valor del **dato_** por **val**.
+ **esHoja()**, retorna el estado actual del nodo con base al número de hijos. Se considera hoja si no tiene ningún hijo, de tener 1 o 2 hijos no se considera como hoja.
+ **estaCompletade1y0s(),** indica si el número de hijos del nodo está completo (alcanzó su tope de 4 hijos).

TAD ArbolQ

Datos mínimos:

- **raiz**, apuntador a un **NodoQ**, representa la raíz del árbol.

Operaciones:

+ **ArbolQ()**, crea un árbol Quad con atributos vacíos (sin raíz).
+ **ArbolQ(val)**, crea un árbol Quad con una raíz por defecto con el valor de **val**.

+ **~ArbolQ()**, elimina el actual árbol Quad.
+ **datoRaiz()**, retorna el dato de la **raiz**.
+ **obtenerRaiz()**, retorna el valor actual de la raíz.
+ **fijarRaiz(val)**, cambia el valor de la **raiz** por **val**.
+ **esVacio()**, retorna el estado actual de la raíz dado verdadero si existe la raiz, falso si no..
+ **insertarNodo(val)**, intenta agregar un nodo al árbol Quad con el valor de **val**. Retorna verdadero si el nodo fue insertado, falso sí no.
+ **insertarNodo(nval, nodo),**
+ **preOrden()**, evalúa si la raíz está vacía. En caso de qué no lo esté, se ingresa a **preOrden(nodo)** para imprimir los valores en preorden.
+ **preOrden(nodo)**, muestra el valor de los nodos mientras realiza un recorrido pre-orden sobre el árbol Quad.
+ **inOrden()**, evalúa si la raíz está vacía. En caso de qué no lo esté, se ingresa a **inOrden(nodo)** para imprimir los valores en inorden.
+ **inOrden(nodo)**, muestra el valor de los nodos mientras realiza un recorrido in-orden sobre el árbol Quad.
+ **posOrden()**, evalúa si la raíz está vacía. En caso de qué no lo esté, se ingresa a **posOrden(nodo)** para imprimir los valores en posorden.
+ **posOrden(nodo),** muestra el valor de los nodos mientras realiza un recorrido pos-orden sobre el árbol Quad.
+ **nivelOrden()**, muestra el valor de los nodos mientras realiza un recorrido nivel-orden sobre el árbol Quad.
+ **armarMatriz(imagen, tamaño),** evalúa si la raíz está vacía. En caso de que no lo esté, se ingresa a **armatMatriz(nodo, imagen, tamaño, inicioI, inicioJ)**
+ **armatMatriz(nodo, imagen, tamaño, inicioI, inicioJ)**, recorre frecuentemente en un recurrido de preorden los descendientes de la raíz para formar la matriz que mas adelante va a conformar la imagen.