Arboles Generales

Ejercicio de parcial - Quadtree

Un **quadtree** es una representación usada para cubrir un espacio cuadrado en dos dimensiones y posteriormente utilizado para determinar ciertas condiciones entre objetos en el mismo.

Un artista moderno trabaja con imágenes codificadas en **quadtree's**. El **quadtree** es un árbol 4-ario que codifica a una imagen con el siguiente criterio:

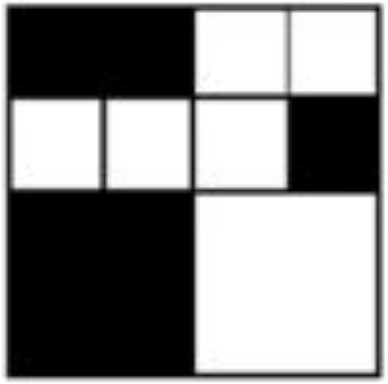
-Si toda la imagen tiene un mismo color, la misma es representada por un único nodo que almacene un dato que represente a ese color.

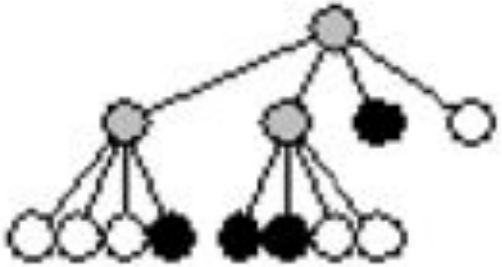
-En caso contrario, se divide la imagen en cuatro cuadrantes que se representan en el árbol como un nodo con 4 hijos, y cada hijo es la conversión de cada una de las partes de la imagen.

El artista desea saber cuántos pixeles de color negro posee una imagen dada. Usted debe implementar un método, que dado un **quadtree** y una cantidad total de pixeles, cuente cuantos pixeles de color negro contiene la imagen codificada en él.

Arboles Generales

Ejercicio de parcial - Quadtree

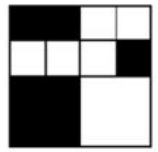


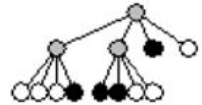


Para el quadtree de la Figura, la salida del método sería 448

Arboles Generales

Ejercicio de parcial – Quadtree

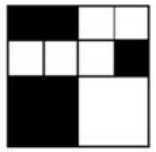


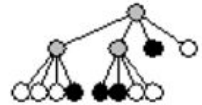


```
package parcial.quadtree;

public class Pixel {

 private String color; // Blanco, Negro, Mixto.

 public Pixel(String color) {
   this.color = color;
 }

 public String getColor() {
  return color;
 }

 public void setColor(String color) {
  this.color = color;
 }

 public boolean esNegro() {
  return this.color.equals("Negro");
 }

 public boolean esBlanco() {
  return this.color.equals("Blanco");
 }
}
```

Arboles Generales

Ejercicio de parcial – Quadtree

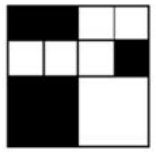


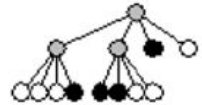


```
package parcial.quadtree;

public class CuentaPixels {
 . . . .
  public static void main(String[] args) {

    Pixel p0 = new Pixel("Blanco");
    Pixel p1 = new Pixel("Blanco");
    Pixel p2 = new Pixel("Blanco");
    Pixel p3 = new Pixel("Negro");

    Pixel p4 = new Pixel("Negro");
    Pixel p5 = new Pixel("Negro");
    Pixel p6 = new Pixel("Blanco");
    Pixel p7 = new Pixel("Blanco");

    Pixel p8 = new Pixel("Negro");
    Pixel p9 = new Pixel("Blanco");

    Pixel p10 = new Pixel("Mixto");
    Pixel p11 = new Pixel("Mixto");
    Pixel p12 = new Pixel("Mixto");

    Quadtree h1 = new Quadtree(p0);
    Quadtree h2 = new Quadtree(p1);
    Quadtree h3 = new Quadtree(p2);
    Quadtree h4 = new Quadtree(p3);

    Quadtree h5 = new Quadtree(p4);
    Quadtree h6 = new Quadtree(p5);
    Quadtree h7 = new Quadtree(p6);
    Quadtree h8 = new Quadtree(p7);
    . . .
```

Arboles Generales

Ejercicio de parcial - Quadtree

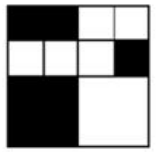


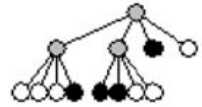


```
        . . . .

        ArbolGeneral<Pixel> h9 = new ArbolGeneral<Pixel>(p8);
        ArbolGeneral<Pixel> h10 = new ArbolGeneral<Pixel>(p9);

        ListaGenerica<ArbolGeneral<Pixel>> hijosII = new ListaEnlazadaGenerica<ArbolGeneral<Pixel>>();
        hijosII.agregarFinal(h1);
        hijosII.agregarFinal(h2);
        hijosII.agregarFinal(h3);
        hijosII.agregarFinal(h4);
        ArbolGeneral<Pixel> hizqizq = new ArbolGeneral<Pixel>(p10, hijosII);

        ListaGenerica<ArbolGeneral<Pixel>> hijosCI = new ListaEnlazadaGenerica<ArbolGeneral<Pixel>>();
        hijosCI.agregarFinal(h5);
        hijosCI.agregarFinal(h6);
        hijosCI.agregarFinal(h7);
        hijosCI.agregarFinal(h8);
        ArbolGeneral<Pixel> hcentroizq = new ArbolGeneral<Pixel>(p11, hijosCI);

        ListaGenerica<ArbolGeneral<Pixel>> hijos = new ListaEnlazadaGenerica<ArbolGeneral<Pixel>>();
        hijos.agregarFinal(hizqizq);
        hijos.agregarFinal(hcentroizq);
        hijos.agregarFinal(h9);
        hijos.agregarFinal(h10);

        ArbolGeneral<Pixel> raiz = new ArbolGeneral<Pixel>(p12, hijos);

        System.out.println("Los pixeles de color negro que posee la imagen es " + contar(raiz, 1024));

    }
}
```

Arboles Generales

Ejercicio de parcial - Quadtree





```
package parcial.quadtree;


public class CuentaPixels {

    public static int contar(ArbolGeneral<Pixel> arbol, int cantidadTotal) {
     int cantidad = 0;
     if (arbol.esHoja() && arbol.getDato().esNegro()) {
         return cantidadTotal;
     } else if (!arbol.esHoja()) {
         int cantidadHijos = cantidadTotal / 4;
         for (int i = 1; i <= 4; i++) {
             cantidad += contar(arbol.getHijos().elemento(i), cantidadHijos);
         }
     }
     return cantidad;
    }

  . . . .

}
```