DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico Número 2

Algoritmos y Estructuras de Datos I

Grupo: 1

Integrante	LU	Correo electrónico
Ciruelos Rodríguez, Gonzalo	063/14	gciruelos@dc.uba.ar
Gatti, Mathias	477/14	mathigatti@gmail.com
Rabinowicz, Lucía	105/14	lu.rabinowicz@gmail.com
Weber, Andres	923/13	herr.andyweber@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

1. Observaciones

1. a

2. Especificación

2.1. posicionesMasOscuras

problema posicionesMasOscuras $(i : Imagen) = result : [\langle \mathbb{Z}, \mathbb{Z} \rangle]$ {

asegura : $mismos(result, [(x, y) | x \leftarrow [0..ancho(i)), y \leftarrow [0..alto(i)), sumaColor(color(i, x, y)) == colorMinimo(i)])$;

aux sumaColor(p:Pixel):$\mathbb{Z}$ = red(p)+green(p)+blue(p);
aux colorMinimo(i:Imagen):$\mathbb{Z} = min([sumaColor(color(i, x, y)) | x \leftarrow [0..ancho(i)), i \leftarrow [0..alto(i))])$;
aux min $(l : [\mathbb{Z}]) : \mathbb{Z} = [l_i | (\forall i, j \leftarrow [0..|l|))l_i \leq l_j]_0$;
}

2.2. top10

problema top10 $(g : Galeria)$=result:$[Imagen]$

asegura : $0 \leq |Result| \leq 10$;

asegura : $if\ |imagenes(g)| \leq 10\ then\ mismos(result, imagenes(g))\ else\ (\forall i \leftarrow sacar(imagenes(g), result))(j \leftarrow result), votos(g, i) \leq votos(g, j)$;

asegura : $estaOrdenadaDecreciente([votos(g, result_i) | i \leftarrow [0..|result|)])$;

aux sacar $(L : [T], A : [T]) : [T] = [l | (l \leftarrow L), (\forall a \leftarrow A), l \neq a]$
aux estaOrdenadaDecreciente$(l : [\mathbb{Z}]) : Bool = (\forall i, j \leftarrow [0..|l|), i \geq j), l_j \geq l_i$
}

2.3. laMasChiquitaConPuntoBlanco

problema laMasChiquitaConPuntoBlanco $(g : Galeria) = result : Imagen$ {

requiere existeImagenConPuntoBlanco: $(\exists h \leftarrow imagenes(g))tieneBlanco(h)$;

asegura : $tieneBlanco(result)$;

asegura : $ancho(Result) * alto(Result) \leq ancho(j) * alto(j)(\forall j \leftarrow imagenesConBlanco(g))$;

aux imagenesConBlanco $(g : Galeria) : [Imagen] = [H | H \leftarrow imagenes(g), tieneBlanco(H)]$
aux tieneBlanco$(i : Imagen) : Bool = (\exists x \leftarrow [0..ancho(i)), y \leftarrow [0..alto(i)))sumaColor(color(i, x, y)) == 765$
aux sumaColor$(p : Pixel) : \mathbb{Z} = red(p) + green(p) + blue(p)$
}

2.4. agregarImagen

problema agergarImagen $(g : Galeria, i : Imagen)$ {

requiere : $i \notin imagenes(g)$;

modifica: g;

asegura : $mismos(imagenes(g), imagenes(pre(g)) + +[i])$;

asegura : $votos(g, i) == 0$;

asegura : $(\forall h \leftarrow imagenes(g), j \leftarrow imagenes(pre(g)), h == j)votos(g, h) == votos(pre(g)), j)$;

}

2.5. votar

problema votar $(g : Galeria, i : Imagen)$ {

requiere : $i \in imagenes(g)$;

modifica: g;

asegura : $mismos(imagenes(g), imagenes(pre(g)))$;

asegura : $(\forall h \leftarrow imagenes(g), j \leftarrow imagenes(pre(g)), h == j, h \neq i) votos(g, h) == votos(pre(g), j)$;

asegura : $votos(g, i) == votos(pre(g), i) + 1$;

}

2.6. eliminarMasVotada

problema eliminarMasVotada $(g : Galeria)$ {

requiere : $|imagenes(g)| \neq 0$;

modifica: g;

asegura : $|imagenes(pre(g))| == |imagenes(g)| + 1$;

asegura : $(\exists h \in imagenes(pre(g)))(\forall j \leftarrow imagenes(g)), h \notin imagenes(g)) votos(pre(g), h) \geq votos(g, j)$;

asegura : $(\forall j \leftarrow imagenes(pre(g)), not(esMasVotada(pre(g), j))), j \in imagenes(g)$;

asegura : $(\forall i \leftarrow imagenes(g) not(esMasVotada(pre(g), i)) votos(g, i) == votos(pre(g), i)$;

asegura : $(\forall i \leftarrow imagenes(g), esMasVotada(i)) votos(g, i) == votos(pre(g), i)$;

aux esMasVotada $(g : Galeria, i : Imagen) : Bool = (\forall j \leftarrow imagenes(g)) votos(g, i) \geq votos(g, j)$

}

3. Implementacion

Código fuente 1: pixel.cpp

```
1  #include "pixel.h"
2
3  Pixel::Pixel(int red, int green, int blue) {
4      if(red < 0 || red > 255) red = 0;
5      if(green < 0 || green > 255) green = 0;
6      if(blue < 0 || blue > 255) blue = 0;
7
8      intensidades[0] = red;
9      intensidades[1] = green;
10     intensidades[2] = blue;
11 }
12
13 void Pixel::cambiarPixel(int red, int green, int blue) {
14     if(red < 0 || red > 255) red = 0;
15     if(green < 0 || green > 255) green = 0;
16     if(blue < 0 || blue > 255) blue = 0;
17
18     intensidades[0] = red;
19     intensidades[1] = green;
20     intensidades[2] = blue;
21 }
22
23 int Pixel::red() const {
```

```
24      return intensidades[0];
25  }
26
27  int Pixel::green() const {
28      return intensidades[1];
29  }
30
31  int Pixel::blue() const {
32      return intensidades[2];
33  }
34
35  void Pixel::guardar(std::ostream& os) const {
36      os << "(" << intensidades[0] << ";" << intensidades[1] << ";" << intensidades[2] << ")";
37  }
38
39  void Pixel::cargar(std::istream& is) {
40      char parentesis, puntoYComaOParentesis;
41      is >> parentesis;
42
43      int colores = 0;
44      while(colores < 3) {
45          int este_color;
46          is >> este_color;
47          intensidades[colores] = este_color;
48
49          is >> puntoYComaOParentesis;
50          colores++;
51      }
52  }
```

4. Demostraciones