



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 (im : Imagen) = result : [ $\mathbb{Z}$ ,  $\mathbb{Z}$ ] {
  asegura : mismos(result, [(i, j) | i  $\leftarrow$  [0..ancho(im)], j  $\leftarrow$  [0..alto(im)], esMinimo(sumaColor(color(im, i, j), im))]);
  aux sumaColor(p:Pixel): $\mathbb{Z}$  = red(p)+green(p)+blue(p);
  aux esMinimo(s :  $\mathbb{Z}$ , im : Imagen) : Bool = ( $\forall i \leftarrow$  [0..ancho(im)],  $\forall j \leftarrow$  [0..alto(im)]) s  $\geq$  sumaColor(color(im, i, j))
}
```

2.2. top10

```
problema top10 (g : Galeria)=result:[Imagen]
  asegura : 0  $\leq$  |result|  $\leq$  10;
  asegura : |imagenes(g)|  $\leq$  10  $\implies$  mismos(result, imagenes(g));
  asegura : ( $\forall h \leftarrow$  result) h  $\in$  imagenes(g)  $\wedge$  esTop10(h, g);
  asegura : ordenadaDecreciente([votos(g, resulti) | i  $\leftarrow$  [0..|result|]]);
  aux esTop10 (h : Imagen, g : Galeria) : Bool = |[im | im  $\leftarrow$  imagenes(g), votos(g, im) < votos(g, h)]| < 10
  aux ordenadaDecreciente(l : [ $\mathbb{Z}$ ]) : Bool = ( $\forall i, j \leftarrow$  [0..|l|], i  $\geq$  j) lj  $\geq$  li
}
```

2.3. laMasChiquitaConPuntoBlanco

```
problema laMasChiquitaConPuntoBlanco (g : Galeria)= result : Imagen {
  requiere existeImagenConPuntoBlanco: ( $\exists h \leftarrow$  imagenes(g)) tieneBlanco(h);
  asegura : tieneBlanco(result);
  asegura : ( $\forall j \leftarrow$  imagenesConBlanco(g)) ancho(result) * alto(result)  $\leq$  ancho(j) * alto(j);
  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)) == 255 * 3
  aux sumaColor(p : Pixel) :  $\mathbb{Z}$  = red(p) + green(p) + blue(p)
}
```

2.4. agregarImagen

```
problema agregarImagen (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(pre(g))) votos(g, h) == votos(pre(g), h);
}
```

}

2.5. votar

```
problema votar ( $g : \text{Galeria}, i : \text{Imagen}$ ) {
  requiere :  $i \in \text{imagenes}(g)$ ;
  modifica:  $g$ ;
  asegura :  $\text{mismos}(\text{imagenes}(g), \text{imagenes}(\text{pre}(g)))$ ;
  asegura :  $(\forall h \leftarrow \text{imagenes}(g), h \neq i) \text{votos}(g, h) == \text{votos}(\text{pre}(g), h)$ ;
  asegura :  $\text{votos}(g, i) == \text{votos}(\text{pre}(g), i) + 1$ ;
}
```

2.6. eliminarMasVotada

```
problema eliminarMasVotada ( $g : \text{Galeria}$ ) {
  requiere :  $|\text{imagenes}(g)| > 0$ ;
  modifica:  $g$ ;
  asegura :  $|\text{imagenes}(\text{pre}(g))| == |\text{imagenes}(g)| + 1$ ;
  asegura :  $(\forall h \leftarrow \text{imagenes}(\text{pre}(g))) h \notin \text{imagenes}(g) \implies (\forall j \leftarrow \text{imagenes}(\text{pre}(g))) \text{votos}(\text{pre}(g), h) \geq \text{votos}(\text{pre}(g), j)$ ;
  asegura :  $(\forall h \leftarrow \text{imagenes}(g), h \in \text{imagenes}(\text{pre}(g))) \text{votos}(\text{pre}(g), h) == \text{votos}(g, h)$ ;
}
```

3. Implementacion

4. Demostraciones

4.1. Invariante de Representación

```
InvRep (imp: ClaseGaleriaImagen):
   $|\text{imp.imagenes}| == |\text{imp.votos}| \wedge$ 
   $(\forall v \leftarrow \text{votos}) v \geq 0 \wedge$ 
   $(\forall i, j \leftarrow [0..|\text{imp.imagenes}|], i \neq j) \text{imp.imagenes}[i] \neq \text{imp.imagenes}[j] \wedge$ 
   $(\forall i \leftarrow [0..|\text{Imp.votos}| - 1]) \text{votos}[i + 1] \geq \text{votos}[i]$ 
```

4.2. Función de Abstracción

```
abs (imp: ClaseGaleriaImagen, esp: Galeria):
   $\text{mismos}(\text{imagenes}(\text{esp}), \text{imp.imagenes}) \wedge$ 
   $(\forall h \leftarrow \text{imagenes}(\text{esp}), \forall i \leftarrow [0..|\text{imp.imagenes}|], \text{imp.imagenes}[i] == h) \text{imp.votos}[i] == \text{votos}(\text{esp}, h)$ 
```

4.3. Correctitud del Código

```

void GaleriaImagenes :: eliminarMasVotada () {

    //estado 1;
    //vale InvRep(this);
    //vale  $|imagenes| > 0$ ;

    imagenes . pop_back ();

    //estado 2;
    //vale  $|imagenes| == |imagenes@1| - 1$ ;
    //vale  $(\forall i \leftarrow [0..|imagenes@1| - 1])\text{imagenes}[i] == \text{imagenes@1}[i]$ ;
    //vale  $votos == votos@1$ ;

    votos . pop_back ();

    //estado 3;    //vale  $|votos| == |votos@2| - 1$ ;
    //vale  $(\forall i \leftarrow [0..|votos@1| - 1])\text{votos}[i] == \text{votos@2}[i]$ ;
    //vale  $imagenes == imagenes@2$ ;

}

//Queremos ver que vale InvRep(this);
// 1;
//vale  $imagenes == imagenes@2$ ;
//implica  $|imagenes| = |imagenes@1| - 1 \wedge |votos| == |votos@1| - 1$ ;
//implica  $|imagenes| == |votos|(pues|imagenes@1| == |votos@1|porInvRep)$ ;

// 2;
//vale  $(\forall v \leftarrow votos@1)v \geq 0$ ;
//vale  $votos@2 == votos@1$ ;
//vale  $(\forall i \leftarrow [0..|votos@2| - 1])\text{votos}[i] == \text{votos@2}[i]$ ;
//implica  $(\forall i \leftarrow [0..|votos@1| - 1])\text{votos}[i] == \text{votos@1}[i]$ ;
//implica  $(\forall v \leftarrow votos)v \geq 0(puesesunasublista)$ ; (sublista de votos@1)

// 3;
//vale  $imagenes == imagenes@2$ ;
//vale  $(\forall i \leftarrow [0..|imagenes@1| - 1])\text{imagenes@2}[i] == \text{imagenes@1}[i]$ ;
//implica  $imagenes@1[0..|imagenes@1| - 1] == imagenes$ ;
//vale  $(\forall i, j \leftarrow [0..|imagenes@1|])i \neq j \Rightarrow \text{imagenes@1}[i] \neq \text{imagenes@1}[j]$ ;
//implica  $(\forall i, j \leftarrow [0..|imagenes|])i \neq j \Rightarrow \text{imagenes}[i] \neq \text{imagenes}[j]$ ; (sublista de imagenes@1)

// 4;
//vale  $(\forall i \leftarrow [0..|votos@1| - 1])\text{votos@1}[i + 1] \geq \text{votos@1}[i]$ ;
//vale  $votos@2 == votos@1$ ;
//vale  $(\forall i \leftarrow [0..|votos@2| - 1])\text{votos}[i] == \text{votos@2}[i]$ ;
//implica  $(\forall i \leftarrow [0..|votos@1| - 1])\text{votos}[i] == \text{votos@1}[i]$ ;
// implica  $(\forall i \leftarrow [0..|votos| - 1])\text{votos}[i + 1] \geq \text{votos}[i]$ ; (por ser sublista de votoss@1)
//vale InvRep(this);

//vale InvRep(this);
//vale abs(this,g);

//1;
// vale mismos(imagenes, imagenes(g));
//vale  $(\forall i \leftarrow [0..|imagenes|])\text{votos}(g, \text{imagenes}[i]) == \text{votos}[i]$ ;
//vale  $|imagenes| == |imagenes| - 1$ ; (ya que  $imagenes == imagenes@2$ )
//implica  $|imagenes| + 1 == |imagenes@1|$ ;

```

```

//vale  $|imagenes(g)| + 1 == |imagenes(pre(g))|$ ; (pues son mismos)

//2;
//vale  $|imagenes(g)| == |imagenes(pre(g))| - 1 \wedge |votos(g)| == |votos(pre(g))| - 1$ ;
//vale  $(\forall i \leftarrow [0..|imagenes| - 1])imagenes[i] == imagenes@1[i] \wedge (\forall i \leftarrow [0..|votos| - 1])votos[i] == votos@1[i]$ ;
//implica  $imagenes(pre(g)) == imagenes(g) + +imagenes(pre(g))[imagenes(pre(g)) - 1] \wedge votos(pre(g)) ==$ 
 $votos(g) + +votos(pre(g))$ ; (por mismos entre g y this)
//vale  $(\forall i \leftarrow [0..|votos@1| - 1])votos@1[i + 1] \geq votos@1[i]$ ;
//vale  $(\forall i \leftarrow [0..|votos| - 1])votos[i + 1] \geq votos[i]$ ;
//implica  $(\forall i \leftarrow [0..|votos| - 1])votos@1[|votos@1| - 1] \geq votos[i]$ ;
//implica  $(\forall h \leftarrow imagenes(pre(g)))h \neq imagenes(g) \implies (\forall j \leftarrow imagenes(pre(g)))votos(pre(g), h) \geq votos(pre(g), j)$ ;

//3;

```