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

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

aux esTop10 $(h : Imagen, g : Galeria) : Bool = |[im|im \leftarrow imagenes(g), votos(g,im) < votos(g,h)]| < 10$
aux ordenadaDecreciente$(l : [\mathbb{Z}]) : Bool = (\forall i,j \leftarrow [0..|l|), i \geq j) l_j \geq l_i$
}

2.3. laMasChiquitaConPuntoBlanco

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

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

asegura : $tieneBlanco(result)$;

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

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

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

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

}

2.5. votar

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

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

modifica: g;

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

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

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

}

2.6. eliminarMasVotada

problema eliminarMasVotada $(g : Galeria)$ {

requiere : $|imagenes(g)| > 0$;

modifica: g;

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

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

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

}

3. Implementacion

4. Demostraciones

4.1. Invariante de Representación

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

4.2. Función de Abstracción

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

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

votos . pop_back ();

//estado 3; //vale $|votos| == |votos@2| - 1$;
//vale $(\forall i \leftarrow [0..|votos@1| - 1))votos[i] == votos@2[i]$;
//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 $(\forall i \leftarrow [0..|votos@2| - 1))votos[i] == votos@2[i]$;
//implica $(\forall i \leftarrow [0..|votos@1| - 1)votos[i] == votos@1[i]$;
//implica $(\forall v \leftarrow votos)v \geq 0(puesesunasublista)$; (sublista de votos@1)

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

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

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

//1;
// vale $mismos(imagenes, imagenes(g))$;
//vale $(\forall i \leftarrow [0..|imagenes|))votos(g, imagenes[i]) == votos[i]$;
//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 $|imagenes(g)| == |imagenes(pre(g)| - 1 \land |votos(g)| == |votos(pre(g))| - 1$;
//vale $(\forall i \leftarrow [0..|imagenes| - 1) imagenes[i] == imagenes@1[i] \land (\forall i \leftarrow [0..|votos| - 1) votos[i] == votos@1[i]$;
//implica $imagenes(pre(g)) == imagenes(g) + + imagenes(pre(g))[imagenes(pre(g)) - 1] \land votos(pre(g)) == votos(g) + + votos(pre(g))[]$; (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;