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. Dado que en el ejercicio de la demostracion nos era dificultoso implicar los aseguras del problema, los docentes nos sugierieron cambiar la especificacion de pop_back por una mas fuerte, que es la que sigue:

   problema pop_back $(a : [T])$ {

   requiere : $|a| > 0$;

   modifica: a;

   asegura sacaElUltimo: $pre(a) == a + +[pre(a)[|pre(a)| - 1]]$;

   }

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 = $(\forall i \leftarrow [0..ancho(im)), \forall j \leftarrow [0..alto(im))) \ s \leq sumaColor(color(im, i, j))$;
}

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

4.2. Función de Abstracción

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

4.3. Correctitud del Código

void GaleriaImagenes :: eliminarMasVotada () {

//estado 1;
//vale $InvRep(this)$;
//implica $abs(this, pre(g)$;
//vale $|this.imagenes| > 0$;
//vale $this == pre(this)$;

imagenes . pop_back ();

//estado 2;
//vale $this@1.imagenes == this.imagenes + +[this@1.imagenes[|this@1.imagenes| - 1]]$; pues se cumplen los requieres de pop_back, dado que $|this@1.imagenes| > 0$
//vale $votos == votos@1$;

votos . pop_back ();

//estado 3;
//vale $this@2.votos == this.votos + +[this@2.votos[|this@2.votos| - 1]]$; pues se cumplen los requieres de pop_back, dado que $|this@1.votos == |this@1.imagenes|$ por $invRep(this@1)$ y a su vez $|this@1.imagenes| > 0$
//vale $imagenes == imagenes@2$;

}

//Queremos ver que vale InvRep(this);
// (1);
//vale $imagenes == imagenes@2 \land\ this@2.votos == this.votos + +[this@2.votos[|this@2.votos| - 1]]$;
//implica $pre(this).imagenes == this.imagenes + +[pre(this).imagenes[|pre(this).imagenes| - 1]] \land$
$pre(this).votos == this.votos + +[pre(this).votos[|pre(this).votos| - 1]]$; por las dos proposiciones anteriores y transformacion de estados
//implica $|pre(this).imagenes| == |this.imagenes| + 1 \land\ |pre(this).votos| == |this.votos| + 1$; por propiedad de longitud
//implica $|imagenes| == |votos|$ pues $|pre(this).imagenes| == |pre(this).votos|$ por $InvRep(pre(this))$

// (2);
//vale $this@2.votos == this.votos + +[this@2.votos[|this@2.votos| - 1]]$;
//implica $pre(this).votos == this.votos + +[pre(this).votos[|pre(this).votos| - 1]]$; por transformacion de estados
//implica $(\forall v \leftarrow this.votos)v \geq 0$; pues es una sublista de $pre(this).votos$ y vale para esta lista por $InvRep(pre(this))$

// (3);
//vale $this@1.imagenes == this.imagenes + +[this@1.imagenes[|this@1.imagenes| - 1]]$;
//implica $pre(this).imagenes == this.imagenes + +[pre(this).imagenes[|pre(this).imagenes| - 1]]$; por transformacion de estados
//implica $(\forall i, j \leftarrow [0..|this.imagenes|])i \neq j)this.imagenes[i] \neq this.imagenes[j]$; pues es una sublista de $pre(this).imagenes$ y vale para esta lista por $InvRep(pre(this))$

// (4);
//vale $this@2.votos == this.votos + +[this@2.votos[|this@2.votos| - 1]]$;
//implica $pre(this).votos == this.votos + +[pre(this).votos[|pre(this).votos| - 1]]$; por transformacion de estados
// implica $(\forall i \leftarrow [0..|this.votos| - 1))this.votos[i+1] \geq this.votos[i]$; pues this.votos es sublista de $pre(this).votos$, que esta ordenada por $InvRep(pre(this))$

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

//Ahora queremos ver que valen los aseguras del problema
//(1);
//vale $mismos(this.imagenes, imagenes(g))$; por abs(this, g)
//vale $mismos(pre(this).imagenes, imagenes(g))$; por abs(pre(this), pre(g))
//implica $|imagenes(g)| == |this.imagenes| \land |imagenes(pre(g))| == |pre(this).imagenes|$;
//vale $|pre(this).imagenes| == |this.imagenes| + 1$; por transformacion de estados y propiedades de longitud, justificado antes
//implica $|imagenes(g)| + 1 == |imagenes(pre(g))|$; por las dos proposiciones anteriores

//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;