



**DEPARTAMENTO
DE COMPUTACION**

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico Número 2

Algoritmos y Estructuras de Datos I

Grupo: 07

Integrante	LU	Correo electrónico
Demartino, Francisco	348/14	demartino.francisco@gmail.com
Frachtenberg Goldsmit, Kevin	247/14	kevinfra94@gmail.com
Gomez, Horacio	756/13	horaciogomez.1993@gmail.com
Pondal, Iván	078/14	ivan.pondal@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. Especificación

Ejercicio 1. problema posicionesMasOscuras($i : Imagen$) = $res : [(\mathbb{Z}, \mathbb{Z})]\{$
 asegura : $mismos(res, [(x, y) \mid y \leftarrow [0..alto(i)], x \leftarrow [0..ancho(i)],$
 $sumaCanalesPixel(color(i, x, y)) == menorSumaCanales(i)]);$
 $\}$

Ejercicio 2. problema top10($g : Galeria$) = $result : [Imagen]\{$
 asegura topDiezOMenos: $|result| == \min(|imagenes(g)|, 10);$
 asegura estanEnGaleria: $(\forall im \leftarrow result) im \in imagenes(g);$
 asegura sinRepetidos: $(\forall i, j \leftarrow [0..|result|], i \neq j) result[i] \neq result[j];$
 asegura ordenPorVotos: $(\forall i \leftarrow [0..|result|])$
 $votos(g, result[i]) == listaVotosOrdenados(g, imagenes(g))[i];$
 $\}$

Ejercicio 3. problema laMasChiquitaConPuntoBlanco($g : Galeria$) = $result : Imagen\{$
 requiere : $|listaImagenesConPixelBlanco(imagenes(g))| > 0;$
 asegura tienePuntoBlanco: $result \in listaImagenesConPixelBlanco(imagenes(g));$
 asegura esChiquita: $area(result) == \minimo($
 $[area(i) \mid i \leftarrow listaImagenesConPixelBlanco(imagenes(g))]);$
 $\}$

Ejercicio 4. problema agregarImagen($g : Galeria, i : Imagen$) $\{$
 modifica $g;$
 asegura lasDeAntesEstanConSusVotos: $(\forall j \leftarrow imagenes(pre(g)))$
 $j \in imagenes(g) \wedge votos(g, j) == votos(pre(g), j);$
 asegura lasQueEstanSalvoLaQueAgregoVienenDeAntes: $(\forall j \leftarrow imagenes(g), j \neq i)$
 $j \in imagenes(pre(g));$
 asegura siAgregoNuevaEntraConCeroVotos: $i \notin imagenes(pre(g)) \rightarrow$
 $(i \in imagenes(g) \wedge votos(g, i) == 0);$
 $\}$

Ejercicio 5. problema votar($g : Galeria, i : Imagen$) $\{$
 requiere noSeVotaCualquierCosa: $i \in imagenes(pre(g));$
 modifica $g;$
 asegura noCambianLasImagenes: $mismos(imagenes(g), imagenes(pre(g)));$
 asegura noSeTocanLosVotosDeLosOtros: $(\forall m \leftarrow imagenes(g), m \neq i) votos(g, m) ==$
 $votos(pre(g), m);$
 asegura elQueSeVotaSumaUno: $votos(g, i) == votos(pre(g), i) + 1;$
 $\}$

Ejercicio 6. problema eliminarMasVotada($g : Galeria$) $\{$
 requiere noVacía: $|imagenes(pre(g))| > 0;$
 modifica $g;$
 asegura seVaUnaSola: $|imagenes(g)| == |imagenes(pre(g))| - 1;$
 asegura laQueSeFueEraGrosa: $(\forall i \leftarrow imagenes(pre(g)), i \notin imagenes(g))$
 $votos(pre(g), i) == maximo(todosLosVotos(pre(g)));$
 asegura lasQueEstanVienenDeAntesConSusVotos: $(\forall i \leftarrow imagenes(g))$
 $i \in imagenes(pre(g)) \wedge votos(g, i) == votos(pre(g), i);$
 $\}$

1.1. Auxiliares

- $\text{aux cuenta}(x : T, a : [T]) : \mathbb{Z} = |[1 \mid y \leftarrow a, y == x]|;$
- $\text{aux mismos}(a, b : [T]) : \text{Bool} = |a| == |b| \wedge (\forall c \leftarrow a) \text{cuenta}(c, a) == \text{cuenta}(c, b);$
- $\text{aux minimo}(l : [\mathbb{Z}]) : \mathbb{Z} = [x \mid x \leftarrow l, (\forall y \leftarrow l) x \leq y][0];$
- $\text{aux maximo}(l : [\mathbb{Z}]) : \mathbb{Z} = [x \mid x \leftarrow l, (\forall y \leftarrow l) x \geq y][0];$
- $\text{aux sumaCanalesPixel}(p : \text{Pixel}) : \mathbb{Z} = \text{red}(p) + \text{green}(p) + \text{blue}(p);$
- $\text{aux listaSumaCanalesPixeles}(i : \text{Imagen}) : [\mathbb{Z}] = [\text{sumaCanalesPixel}(\text{color}(i, x, y)) \mid y \leftarrow [0..\text{alto}(i)], x \leftarrow [0..\text{ancho}(i)]];$
- $\text{aux area}(i : \text{Imagen}) : \mathbb{Z} = \text{ancho}(i) * \text{alto}(i);$
- $\text{aux menorSumaCanales}(i : \text{Imagen}) : \mathbb{Z} = \text{minimo}(\text{listaSumaCanalesPixeles}(i));$
- $\text{aux todosLosVotos}(g : \text{Galeria}) : [\mathbb{Z}] = [\text{votos}(g, i) \mid i \leftarrow \text{imagenes}(g)];$
- $\text{aux esPixelBlanco}(px : \text{Pixel}) : \text{Bool} = \text{red}(px) == \text{green}(px) == \text{blue}(px) == 255;$
- $\text{aux tienePixelBlanco}(i : \text{Imagen}) : \text{Bool} = \text{alguno}([\text{esPixelBlanco}(\text{color}(i, x, y)) \mid y \leftarrow [0..\text{alto}(i)], x \leftarrow [0..\text{ancho}(i)]]);$
- $\text{aux listaImagenesConPixelBlanco}(imgs : [\text{Imagen}]) : [\text{Imagen}] = [im \mid im \leftarrow imgs, \text{tienePixelBlanco}(im)];$
- $\text{aux cuentaMasVotos}(g : \text{Galeria}, imgs : [\text{Imagen}], img : \text{Imagen}) : \mathbb{Z} = |[1 \mid im \leftarrow imgs, \text{votos}(g, im) > \text{votos}(g, img)]|;$
- $\text{aux listaVotosOrdenados}(g : \text{Galeria}, imgs : [\text{Imagen}]) : [\mathbb{Z}] = [\text{votos}(g, im) \mid i \leftarrow [0..|imgs|], im \leftarrow imgs, \text{cuentaMasVotos}(g, imgs, im) == i];$

2. Demostraciones

2.1. Predicado de abstracción e invariante de representación

```

class GaleriaImagenes {
    public:
        void dividirYAgregar(const Imagen &imagen, int n, int m);
        Imagen laMasChiquitaConPuntoBlanco() const;
        void agregarImagen(const Imagen &imagen);
        void votar(const Imagen &imagen);
        void eliminarMasVotada();
        vector<Imagen> top10() const;
        void guardar(std::ostream& os) const;
        void cargar(std::istream& is);

    private:
        void acomodar();
        bool existeImagen(const Imagen &imagen);
        std::vector<Imagen> imagenes;
        std::vector<int> votos;

        // InvRep(imp : GaleriaImagenes):
        // |imp.imagenes| == |imp.votos| ∧
        // (∀v ← imp.votos) v ≥ 0 ∧
        // (∀i, j ← [0..|imp.imagenes|], i < j)
        // imp.imagenes[i] ≠ imp.imagenes[j] ∧ imp.votos[i] ≤ imp.votos[j]

        // abs(imp : GaleriaImagenes, esp : Galeria):
        // mismos(imp.imagenes, imagenes(esp)) ∧
        // (∀i ← [0..|imp.votos|]) imp.votos[i] == votos(esp, imp.imagenes[i])
};

```

2.2. Preservación del invariante de representación

```

void GaleriaImágenes::eliminarMasVotada() {
    // E0:
    // implica InvRep(this)
    // por ser un metodo publico, InvRep se asume verdadero
    // implica abs(this,g)
    // porque InvRep(this) es verdadero
    // vale |imagenes(g)| > 0
    // por requiere
    // implica |this.imagenes| > 0
    // por abs(this, g)
    // ya que vale mismos(this.imagenes, imagenes(g))
    // implica |this.votos| > 0
    // porque InvRep(this) es verdadero con lo cual
    // |this.imagenes| == |this.votos|
    // implica  $P_{Aux1}: |imagenes(g)| > 0$ 
    this→imagenes.pop_back();
    // E1:
    // vale  $Q_{Aux1}$ :
    // |this.imagenes| == |this.imagenes@E0| - 1  $\wedge$ 
    //  $(\forall i \leftarrow [0..|this.imagenes@E0| - 1]) this.imagenes[i] == this.imagenes@E0[i]$ 
    // vale this.votos == this.votos@E0
    // implica  $P_{Aux2}: |this.votos| > 0$ 
    this→votos.pop_back();
    // E2:
    // vale  $Q_{Aux2}$ :
    // |this.votos| == |this.votos@E1| - 1  $\wedge$ 
    //  $(\forall i \leftarrow [0..|this.votos@E1| - 1]) this.votos[i] == this.votos@E1[i]$ 
    // vale this.imagenes == this.imagenes@E1

    // implica |this.imagenes| == |this.imagenes@E0| - 1  $\wedge$  |this.votos| == |this.votos@E0| - 1
    // por transformacion de estados
    // implica |this.imagenes| == |this.votos|
    // por implica anterior e InvRep@E0

    // implica  $(\forall i \leftarrow [0..|this.votos@E0| - 1]) this.votos[i] == this.votos@E0[i]$ 
    // por transformacion de estados
    // implica  $(\forall v \leftarrow this.votos) v == v@E0$ 
    // por implica anterior, donde se estaba recorriendo todo el listado
    // implica  $(\forall v \leftarrow this.votos) v \geq 0$ 
    // por InvRep@E0

    // implica  $(\forall i \leftarrow [0..|this.votos@E0| - 1]) this.votos[i] == this.votos@E0[i] \wedge$ 
    //  $(\forall j \leftarrow [0..|this.imagenes@E0| - 1]) this.imagenes[j] == this.imagenes@E0[j]$ 
    // por transformacion de estados
    // implica  $(\forall i, j \leftarrow [0..|this.imagenes@E0| - 1], i < j)$ 
    //  $this.imagenes[i] \neq this.imagenes[j] \wedge this.votos[i] \leq this.votos[j]$ 
    // por InvRep@E0
    // implica  $(\forall i, j \leftarrow [0..|this.imagenes|), i < j)$ 
    //  $this.imagenes[i] \neq this.imagenes[j] \wedge this.votos[i] \leq this.votos[j]$ 
    // por transformacion de estados

    // implica InvRep(this)
}

```

2.3. Demostración de correctitud

Ahora que demostramos que se restaura el invariante de representación a la salida del método podemos usar el predicado de abstracción para demostrar que también se cumple la postcondición.

```
// implica  $|this.imagenes| == |this.imagenes@E0| - 1$ 
// por transformacion de estados
// implica  $|imagenes(g)| == |imagenes(pre(g))| - 1$ 
// por abs(this, g)
// mismos(this.imagenes, imagenes(g))
// por abs(pre(this), pre(g))
// mismos(this.imagenes@E0, imagenes(pre(g)))

// implica  $(\forall i \leftarrow imagenes(pre(g)), i \notin imagenes(g))$ 
// votos(pre(g), i) == maximo(todosLosVotos(pre(g)))
// por InvRep(this), ya que nos aseguraba que las imagenes
// se encontrasen ordenadas por votos en orden creciente
// removiendo la ultima en el listado, se removio la mas votada

// implica  $(\forall i \leftarrow [0..|this.imagenes@E0| - 1])$ 
//  $this.imagenes[i] == this.imagenes@E0[i] \wedge this.votos[i] == this.votos@E0[i]$ 
// por transformacion de estados e InvRep@E0
// (la longitud de imagenes que es la misma que la de votos)
// implica  $(\forall i \leftarrow [0..|imagenes(pre(g))| - 1])$ 
//  $imagenes(g)[i] == imagenes(pre(g))[i] \wedge$ 
//  $votos(g, imagenes(g)[i]) == votos(pre(g), imagenes(pre(g))[i])$ 
// por abs(this, g)
// mismos(this.imagenes, imagenes(g))
//  $(\forall i \leftarrow [0..|this.votos|]) this.votos[i] == votos(g, this.imagenes[i])$ 
// por abs(pre(this), pre(g))
// mismos(this.imagenes@E0, imagenes(pre(g)))
//  $(\forall i \leftarrow [0..|this.votos@E0|]) this.votos@E0[i] == votos(pre(g), this.imagenes@E0[i])$ 
// implica  $(\forall i \leftarrow [0..|imagenes(g)|])$ 
//  $imagenes(g)[i] == imagenes(pre(g))[i] \wedge$ 
//  $votos(g, imagenes(g)[i]) == votos(pre(g), imagenes(pre(g))[i])$ 
// por primer asegura  $|imagenes(g)| == |imagenes(pre(g))| - 1$ 
// implica  $(\forall i \leftarrow imagenes(g)) i \in imagenes(pre(g)) \wedge votos(g, i) == votos(pre(g), i)$ 
// por implica anterior, se recorrian todos los valores de imagenes
// con lo cual se puede escribir con un paratodo
```