



**DEPARTAMENTO
DE COMPUTACION**

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 1

Una especificación vale más que mil imágenes

Algoritmos y Estructuras de Datos I

Grupo: 7

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. Observaciones

1. A lo largo de la confección de este trabajo práctico, nos surgió la duda acerca de si considerar una imagen vacía de 0x0 como válida. Ante las respuestas obtenidas de los docentes, quienes nos indicaron que podíamos o no considerarla como válida, decidimos dejarla como imagen no válida a raíz de que, por ejemplo, en el caso del filtro “Dividir”, podrían generarse infinitas imágenes vacías.

2. Resolución

Ejercicio 1. Especificación Blur:

```
problema Blur(imagenOriginal : Imagen, k :  $\mathbb{Z}$ ) = imagenNueva : Imagen{
  requiere : k > 0  $\wedge$  imagenVálida(imagenOriginal);
  asegura : imagenVálida(imagenNueva);
  asegura : mismoTamaño(imagenOriginal, imagenNueva);
  asegura : todos([imagenNueva[y][x] == colorPromedioEnPosición(imagenOriginal, x, y, k)
    | x  $\leftarrow$  [0..ancho(im)], y  $\leftarrow$  [0..alto(im)]));
}
```

Ejercicio 2. Especificación Acuarela:

```
problema Acuarela(imgOriginal : Imagen, k :  $\mathbb{Z}$ ) = imgFinal : Imagen{
  requiere : k > 0  $\wedge$  imagenVálida(imgOriginal);
  asegura : imagenVálida(imgFinal);
  asegura : mismoTamaño(imgOriginal, imgFinal);
  asegura : filtroAcuarela(imgOriginal, imgFinal, k);
}
```

Ejercicio 3. Especificación Dividir:

```
problema Dividir(im : Imagen, m, n :  $\mathbb{Z}$ ) = listaPartes : [Imagen]{
  requiere : m > 0  $\wedge$  n > 0;
  requiere : imagenVálida(im);
  requiere : ancho(im) mod n == 0;
  requiere : alto(im) mod m == 0;
  asegura : mismos(listaPartes, imCortadas(im, m, n));
}
```

Ejercicio 4. Especificación Pegar:

```
problema Pegar(destino : Imagen, col : Pixel, origen : Imagen){
  modifica destino;
  requiere : pixelVálido(col);
  requiere : imagenVálida(origen)  $\wedge$  imagenVálida(pre(destino));
  requiere : máscaraVálida(pre(destino), col);
  asegura :  $\neg$ imagenTieneHuecoPegar(pre(destino), origen, col)  $\rightarrow$ 
    destino == pre(destino);
  asegura : imagenTieneHuecoPegar(pre(destino), origen, col)  $\rightarrow$ 
    aplicaPegar(pre(destino), destino, huecoPegar(origen, col), origen);
}
```

2.1. Auxiliares

- $\text{aux pixelNegro} : \text{Pixel} = (0, 0, 0);$
- $\text{aux canalesRojos}(ps : [\text{Pixel}]) : [\mathbb{Z}] = [\text{prm}(p) \mid p \leftarrow ps];$
- $\text{aux canalesVerdes}(ps : [\text{Pixel}]) : [\mathbb{Z}] = [\text{sgd}(p) \mid p \leftarrow ps];$
- $\text{aux canalesAzules}(ps : [\text{Pixel}]) : [\mathbb{Z}] = [\text{ter}(p) \mid p \leftarrow ps];$
- $\text{aux cuenta}(x : T, a : [T]) : \mathbb{Z} = |[y \mid y \leftarrow a, y == x]|;$
- $\text{aux mismos}(a, b : [T]) : \text{Bool} = |a| == |b| \wedge (\forall c \leftarrow a) \text{cuenta}(c, a) == \text{cuenta}(c, b);$
- $\text{aux concat}(ts : [[T]]) : [T] = [ts[i][j] \mid i \leftarrow [0..|ts|], j \leftarrow [0..|ts[i]|]];$
- $\text{aux todosIguales}(ts : [T]) : \text{Bool} = (|ts| == 0) \vee \text{cuenta}(ts[0], ts) == |ts|;$
- $\text{aux alto}(im : \text{Imagen}) : \mathbb{Z} = |im|;$
- $\text{aux ancho}(im : \text{Imagen}) : \mathbb{Z} = \text{if } (\text{alto}(im) == 0) \text{ Then } 0 \text{ Else } |im[0]|;$
- $\text{aux listaAnchos}(im : \text{Imagen}) : \mathbb{Z} = [|im[i]| \mid i \leftarrow [0..|im|]];$
- $\text{aux pixelVálido}(p : \text{Pixel}) : \text{Bool} =$
 $(\text{prm}(p) \geq 0) \wedge (\text{prm}(p) < 256) \wedge (\text{sgd}(p) \geq 0) \wedge (\text{sgd}(p) < 256) \wedge (\text{ter}(p) \geq 0) \wedge (\text{ter}(p) < 256);$
- $\text{aux imagenVálida}(im : \text{Imagen}) : \text{Bool} = \text{ancho}(im) > 0 \wedge \text{alto}(im) > 0 \wedge$
 $\text{todosIguales}(\text{listaAnchos}(im)) \wedge \text{todos}([\text{pixelVálido}(p) \mid p \leftarrow \text{concat}(im)]);$
- $\text{aux mismoTamaño}(a, b : \text{Imagen}) : \text{Bool} = \text{ancho}(a) == \text{ancho}(b) \wedge \text{alto}(a) == \text{alto}(b);$
- $\text{aux kVecinosCompletos}(k, x, y : \mathbb{Z}, im : \text{Imagen}) : \text{Bool} = (x \geq k) \wedge (y \geq k) \wedge$
 $(x < \text{ancho}(im) - k) \wedge (y < \text{alto}(im) - k);$
- $\text{aux subImagen}(x0, x1, y0, y1 : \mathbb{Z}, im : \text{Imagen}) : \text{Imagen} =$
 $[|im[i][j] \mid j \leftarrow [x0...x1] \mid i \leftarrow [y0...y1]|];$
- $\text{aux kVecinos}(k, x, y : \mathbb{Z}, im : \text{Imagen}) : [\text{Pixel}] =$
 $\text{concat}(\text{subImagen}(x - k, x + k, y - k, y + k, im));$
- $\text{aux promedio}(ns : [\mathbb{Z}]) : \mathbb{Z} = \text{if } (|ns| == 0) \text{ then } 0 \text{ else } (\text{sum}(ns) \text{ div } |ns|);$
- $\text{aux pixelPromedio}(ps : [\text{Pixel}]) : \text{Pixel} =$
 $(\text{promedio}(\text{canalesRojos}(ps)), \text{promedio}(\text{canalesVerdes}(ps)), \text{promedio}(\text{canalesAzules}(ps)));$
- $\text{aux colorPromedioEnPosición}(im : \text{Imagen}, x, y, k : \mathbb{Z}) : \text{Pixel} =$
 $\text{if } k\text{VecinosCompletos}(k, x, y, im) \text{ then } \text{pixelPromedio}(k\text{Vecinos}(k, x, y, im)) \text{ else } \text{pixelNegro};$
- $\text{aux menores}(xs : [\mathbb{Z}], x : \mathbb{Z}) : \mathbb{Z} = |[y \mid y \leftarrow xs, y < x]|;$
- $\text{aux menoresIguales}(xs : [\mathbb{Z}], x : \mathbb{Z}) : \mathbb{Z} = |[y \mid y \leftarrow xs, y \leq x]|;$
- $\text{aux esMediana}(xs : [\mathbb{Z}], x : \mathbb{Z}) : \text{Bool} =$
 $\text{menores}(xs, x) < (|xs| \text{ div } 2) \wedge \text{menoresIguales}(xs, x) \geq (|xs| \text{ div } 2);$
- $\text{aux pixelEsMedianaLista}(px : \text{Pixel}, \text{listaPixeles} : [\text{Pixel}]) : \text{Bool} =$
 $\text{esMediana}(\text{canalesRojos}(\text{listaPixeles}), \text{prm}(px)) \wedge \text{esMediana}(\text{canalesVerdes}(\text{listaPixeles}), \text{sgd}(px)) \wedge$
 $\text{esMediana}(\text{canalesAzules}(\text{listaPixeles}), \text{ter}(px));$
- $\text{aux pixelCumpleFiltroAcuarela}(imgOriginal, imgFinal : \text{Imagen}, k, x, y : \mathbb{Z}) : \text{Bool} =$
 $\text{if } k\text{VecinosCompletos}(k, x, y, imgOriginal) \text{ then}$
 $\text{pixelEsMedianaLista}(imgFinal[y][x], k\text{Vecinos}(k, x, y, imgOriginal)) \text{ else}$
 $imgFinal[y][x] == \text{pixelNegro};$
- $\text{aux filtroAcuarela}(imgOriginal, imgFinal : \text{Imagen}, k : \mathbb{Z}) : \text{Bool} =$
 $\text{todos}([\text{pixelCumpleFiltroAcuarela}(imgOriginal, imgFinal, k, x, y)$
 $\mid y \leftarrow [0..\text{alto}(imgOriginal)], x \leftarrow [0..\text{ancho}(imgOriginal)]]);$

- **aux** *imCortadas*(*im* : *Imagen*, *m*, *n* : \mathbb{Z}) : *Imagen* =
 $[subImagen((i * ancho(im) \div n), ((i + 1) * ancho(im) \div n) - 1, (j * alto(im) \div m),$
 $((j + 1) * alto(im) \div m) - 1, im) | i \leftarrow [0..m), j \leftarrow [0..n)];$
- **aux** *contieneSubImagen*(*grande*, *chica* : *Imagen*) : *Bool* =
 $alguno([chica == subImagen(x0, x0 + ancho(chica) - 1, y0, y0 + ancho(chica) - 1, grande)$
 $| x0 \leftarrow [0..ancho(grande) - ancho(chica)], y0 \leftarrow [0..alto(grande) - alto(chica)])];$
- **aux** *huecoGrande*(*col* : *Pixel*, *alto*, *ancho* : \mathbb{Z}) : *Imagen* = $[[col | j \leftarrow [0..ancho)]] | i \leftarrow [0..alto)];$
- **aux** *huecoPegar*(*im* : *Imagen*, *col* : *Pixel*) : *Imagen* = *huecoGrande*(*col*, *alto*(*im*), *ancho*(*im*));
- **aux** *imagenTieneHuecoPegar*(*destino*, *origen* : *Imagen*, *col* : *Pixel*) : *Bool* =
contieneSubImagen(*destino*, *huecoPegar*(*origen*, *col*));
- **aux** *enRango*(*a*, *inicio*, *fin* : \mathbb{Z}) : *Bool* = $inicio \leq a \wedge a \leq fin;$
- **aux** *sonIgualesSalvoRectangulo*(*im1*, *im2* : *Imagen*, *x0*, *x1*, *y0*, *y1* : \mathbb{Z}) : *Bool* =
 $todos([im1[y][x] == im2[y][x] \vee (enRango(y, y0, y1) \wedge enRango(x, x0, x1))$
 $| x \leftarrow [x0..x1], y \leftarrow [y0..y1]]);$
- **aux** *aplicaPegar*(*grande1*, *grande2*, *chica1*, *chica2* : *Imagen*) : *Bool* =
 $mismoTamaño(grande1, grande2) \wedge$
 $mismoTamaño(chica1, chica2) \wedge$
 $alguno([aplicaPegarEnPos(grande1, grande2, chica1, chica2,$
 $x0, x0 + ancho(chica1) - 1, y0, y0 + alto(chica1) - 1)$
 $| x0 \leftarrow [0..ancho(grande1) - ancho(chica1)], y0 \leftarrow [0..alto(grande1) - alto(chica1)]$
 $]);$
- **aux** *aplicaPegarEnPos*(*g1*, *g2*, *c1*, *c2* : *Imagen*, *x0*, *x1*, *y0*, *y1* : \mathbb{Z}) : *Bool* =
 $c1 == subImagen(x0, x1, y0, y1, g1) \wedge$
 $c2 == subImagen(x0, x1, y0, y1, g2) \wedge$
 $sonIgualesSalvoRectangulo(g1, g2, x0, x1, y0, y1);$
- **aux** *listaColorEnImagen*(*img* : *Imagen*, *col* : *Pixel*) : $[(x, y)] =$
 $[(x, y) | y \leftarrow [0..alto(img)], x \leftarrow [0..ancho(img)], img[y][x] == col];$
- **aux** *recorte*(*esquinaSupIzq*, *esquinaInfDer* : (\mathbb{Z}, \mathbb{Z}) , *img* : *Imagen*) : *Imagen* =
subImagen(*prm*(*esquinaSupIzq*), *prm*(*esquinaInfDer*), *sgd*(*esquinaSupIzq*), *sgd*(*esquinaInfDer*), *img*);
- **aux** *rectánguloMáscara*(*img* : *Imagen*, *col* : *Pixel*) : *Imagen* =
recorte(
listaColorEnImagen(*img*, *col*)[0],
listaColorEnImagen(*img*, *col*)[*listaColorEnImagen*(*img*, *col*)| - 1],
img);
- **aux** *área*(*img* : *Imagen*) : $\mathbb{Z} = ancho(img) * alto(img);$
- **aux** *máscaraVálida*(*img* : *Imagen*, *col* : *Pixel*) : *Bool* =
 $|listaColorEnImagen(img, col)| > 0 \wedge$
 $|listaColorEnImagen(img, col)| == área(rectánguloMáscara(img, col)) \wedge$
 $rectánguloMáscara(img, col) == huecoPegar(rectánguloMáscara(img, col), col);$