DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Recuperatorio 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 imágen no valida 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 : $k > 0 \land imagenVálida(imagenOriginal)$;
    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 : $k > 0 \land imagenVálida(imgOriginal)$;
    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 : $m > 0 \land n > 0$;
    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($origen, destino : Imagen, col : Pixel)\{$
    modifica $destino$;
    requiere : $pixelVálido(col)$;
    requiere : $imagenVálida(origen) \land imagenVálida(pre(destino))$;
    requiere : $máscaraVálida(pre(destino), col)$;
    asegura : $\neg imagenTieneHuecoPegar(pre(destino), origen, col) \rightarrow$
    $destino == pre(destino)$;
    asegura : $imagenTieneHuecoPegar(pre(destino), origen, col) \rightarrow$
    $aplicaPegar(pre(destino), destino, huecoPegar(origen, col), origen)$;
}

Ejercicio 5. Especificación Transición:

problema Transición$(inicial, final : Imagen, n : \mathbb{Z}) = secuencia : [Imagen]\{$
requiere : $n > 1$;
requiere : $imagenVálida(inicial) \wedge imagenVálida(final)$;
requiere : $mismoTamaño(inicial,\ final)$;
asegura : $|secuencia| == n$;
asegura : $todos[imagenEnTransición(inicial, final, n, i) == secuencia[i] \mid i \leftarrow [0..n)]$;
$\}$

2.1. Auxiliares

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

- aux $pixelEsMedianaLista(px : Pixel, listaPixeles : [Pixel]) : Bool =$
  $esMediana(canalesRojos(listaPixeles), prm(px)) \wedge esMediana(canalesVerdes(listaPixeles), sgd(px)) \wedge$
  $esMediana(canalesAzules(listaPixeles), ter(px))$;
- aux $pixelCumpleFiltroAcuarela(imgOriginal, imgFinal : Imagen, k, x, y : \mathbb{Z}) : Bool =$
  $if\ kVecinosCompletos(k, x, y, imgOriginal)\ then$
  $pixelEsMedianaLista(imgFinal[y][x], kVecinos(k, x, y, imgOriginal))\ else$
  $imgFinal[y][x] == pixelNegro$;
- aux $filtroAcuarela(imgOriginal, imgFinal : Imagen, k : \mathbb{Z}) : Bool =$
  $todos([pixelCumpleFiltroAcuarela(imgOriginal, ImgFinal, k, x, y)$
  $| y \leftarrow [0..alto(imgOriginal)), x \leftarrow [0..ancho(imgOriginal))])$;
- 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 $máscaraVálida(img : Imagen, col : Pixel) : Bool =$
  $|listaColorEnImagen(img, col)| > 0\ \wedge$
  $|listaColorEnImagen(img, col)| == área(rectánguloMáscara(img, col))\ \wedge$
  $rectánguloMáscara(img, col) == huecoPegar(rectánguloMáscara(img, col),\ col)$;

- aux $imagenEnTransición(inicial, final : Imagen, n, i : \mathbb{Z}) : Imagen =$
  $[[pixelEnTransición(inicial[y][x], final[y][x], n, i)$
  $\mid x \leftarrow [0..ancho(inicial))\ ] \mid y \leftarrow [0..alto(inicial))\ ];$
- aux $pixelEnTransición(a, b : Pixel, n, i : \mathbb{Z}) : Pixel =$
  $[canalEnTransición(prm(a), prm(b), n, i),$
  $canalEnTransición(sgd(a), sgd(b), n, i),$
  $canalEnTransición(ter(a), ter(b), n, i)];$
- aux $canalEnTransición(a, b, n, i : \mathbb{Z}) : \mathbb{Z} = a + (i * (b - a))\ div\ (n - 1);$