UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación

Algoritmos y Estructura de Datos I

Segundo cuatrimestre de 2014

18 de septiembre de 2014

Una especificación vale más que mil imágenes

Aclaraciones

- Para aprobar la totalidad del TP es necesario tener aprobado cada uno de sus módulos.
- **No** está permitido el uso de **acum** para la resolución.
- Agregar las precondiciones necesarias a los problemas para asegurar que no se rompan los invariantes de tipo.
- **Fecha de entrega: 3 de Octubre de 2014**

El objetivo del TP es especificar algunas funciones aplicadas sobre imágenes. Describimos a continuación una forma de representar imágenes y algunas operaciones. Estas funciones deberán ser especificadas para erradicar la ambigüedad natural del lenguaje.

¿Qué es una imagen? Decimos que una imagen es una representación visual en dos dimensiones. Denominamos pixel a la unidad más pequeña de la imagen. Por lo tanto una imagen es una matriz de pixeles. Cada pixel es de un color, por lo tanto necesitaremos una forma de representar colores.

Representaremos un pixel usando el modelo RGB. Es un modelo de color basado en la síntesis aditiva, con el que es posible representar un color mediante la mezcla por adición de los tres colores de luz primarios. La intensidad de cada una de las componentes se mide según una escala que, usualmente, va del 0 al 255[1] y cada color es definido por un conjunto de valores escritos entre paréntesis (correspondientes a valores "R", "G" y "B") y separados por comas. Por ejemplo, el rojo se representa con (255,0,0), el verde con (0,255,0) y el azul con (0,0,255). En tanto, el negro es (0,0,0) y el blanco es (255,255,255), el amarillo (255,255,0), cian (0,255,255) y magenta (255,0,255).

Definimos *k-vecinos* de una posición de una matriz a todas las posiciones que estén a menos de distancia k en todas las dimensiones. Por ejemplo los *2-vecinos* de (3,5) son (2,4),(3,4)(4,4),(2,5),(3,5),(4,5),(2,6),(3,6) y (4,6). Los *2-vecinos* de (0,0) son el (0,0),(0,1),(1,0) y (1,1), ya que las posiciones son pares de enteros positivos. Cuando una posición no tiene la máxima cantidad de vecinos diremos que sus *k-vecinos* están *incompletos*, por lo tanto los *2-vecinos* de (0,0) están *incompletos*.

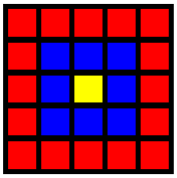


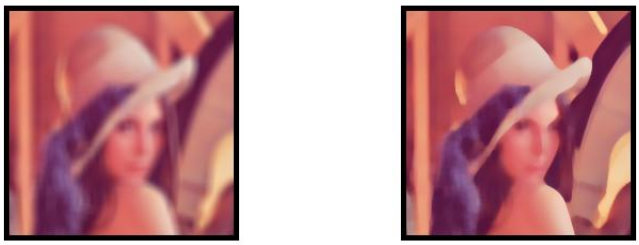
Figura 1: Los cuadrados azules y el amarillo, son los *2-vecinos* del cuadrado amarillo.
Los cuadrados rojos, azules y el amarillo son los *3-vecinos* del cuadrado amarillo.

Para especificar vamos a tener un tipo Pixel, que es una tripla de enteros (uno por cada componente), y un tipo Imagen que es una lista de listas de Pixel. Cada lista de Pixel es una fila de la imagen.

[1] El número 255 es arbitrario para modelar la intensidad, pero lo usaremos así por cuestiones de uso

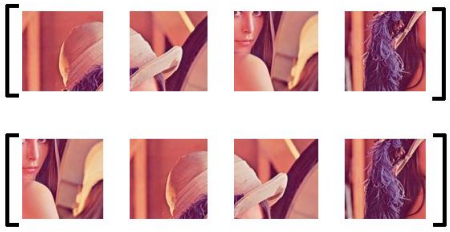
Las funciones que deben especificar como problemas son:

- **blur**: Toma de parámetros una imagen y un entero k. Devuelve una imagen donde la intensidad de color de cada pixel corresponde al promedio (redondeado para abajo[2]) de las intensidades respectivas de los k-*vecinos* del pixel en la imagen original. Aquellas posiciones que tengan sus $k - vecinos\ incompletos$ en la imagen original quedarán en negro.



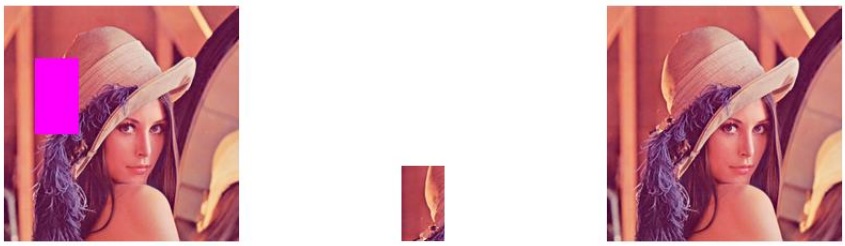
A la izquierda un ejemplo de **blur**.
A la derecha un ejemplo de **acuarela**.

- **acuarela**: Toma de parámetros una imagen y un entero k. Devuelve una imagen que cada posición tiene, en cada componente, la mediana de los valores en esa componente de $k - vecinos$ en la imagen original. Aquellas posiciones que tengan sus $k - vecinos\ incompletos$ en la imagen original quedarán en negro.
- **dividir**: Toma de parámetros una imagen y dos enteros m y n. Divide la imagen en m filas y n columnas resultando en $m \times n$ rectángulos de igual tamaño. Devuelve una lista (en cualquier orden) de imágenes correspondientes a dicha partición.



Dos listas de imagnes que son solución para **dividir** tomando a $m = 2$ y $n = 2$.

- **pegar**: Toma de parámetros una imagen, un pixel y una segunda imagen. La primer imagen tiene que tener exactamente un rectángulo del color del pixel tomado como parámetro. Modifica la primer imagen de forma tal que la segunda imagen este contenida una vez en el rectángulo, el resto de la imagen permanece igual. Si no cabe, no se modifica.



Ejemplo de **pegar**

- **<u>transicion</u>**: Toma dos imágenes *inicial* y $final$ del mismo tamaño y un entero n. Devuelve una secuencia de n imágenes tal que en la $i - esima$ imagen cada pixel esta en transición. El pixel en transición de la posición (x,y) se define como $inicial_{xy} + \frac{i \times (final_{xy} - inicial_{xy})}{n-1}$ en cada componente (tomando parte entera). La primer imagen debe ser la *inicial* y la última debe ser $final$.

[2]Por ejemplo: 3,4 redondea a 3. 3 redondea a 3 y 3.9 redondea a 3