Creando miniaturas

Contribución de Facundo Gutiérrez y Agustín Santiago Gutiérrez

Descripción del problema

En computación, una imagen se suele almacenar utilizando lo que se denomina un mapa de bits: una matriz de píxeles, donde cada uno se representa con tres números enteros entre 0 y 255 inclusive. El primer número indica la intensidad de rojo, el segundo la intensidad de verde, y el tercero la intensidad de azul. Combinando las intensidades de los 3 colores primarios en diferentes cantidades, se pueden formar los más de 16 millones de colores que muestran las pantallas habituales.

Se tiene un mapa de bits de $n \times m$ (es decir, n filas de m píxeles cada una). Las filas se numeran de 0 a $n-1$. Las columnas de 0 a $m-1$.

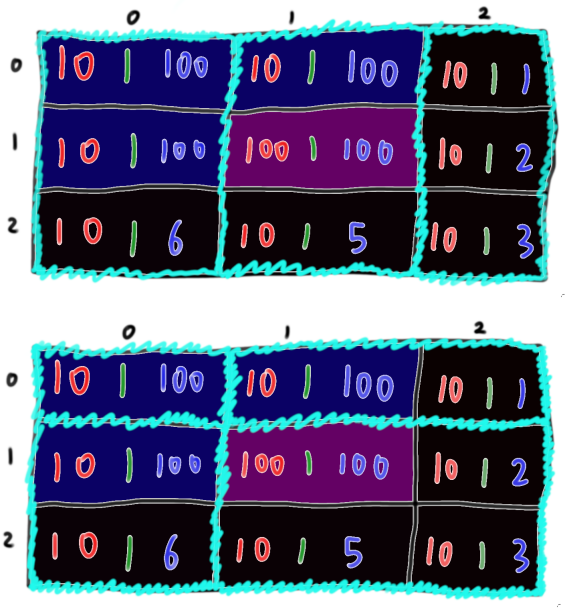
Una tarea muy común en computación gráfica consiste en mostrar una *miniatura* de una imagen. Una miniatura de $k \times l$, con $1 \leq k \leq n$ y $1 \leq l \leq m$, es un nuevo mapa de bits que aproxima al original, aunque típicamente es más pequeño. Esto se usa por ejemplo en los navegadores de archivos, para mostrar muchas miniaturas de las fotos en una carpeta a la vez, permitiendo al usuario encontrar más fácilmente la imagen deseada para abrirla.

Una estrategia muy básica para rápidamente crear una miniatura de $k \times l$ del mapa de bits dado, consiste en cuadricular y promediar. Más específicamente, se realizan los siguientes pasos:

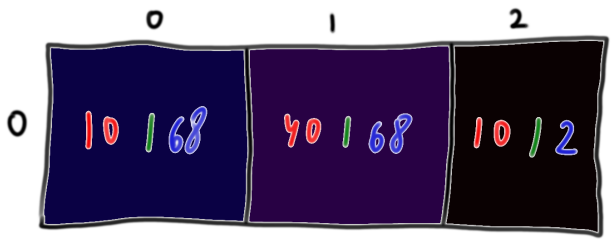
1. Se particionan las n filas en k rangos, numerados de 0 a $k-1$. El 0-ésimo va desde la fila 0 hasta la $\lfloor \frac{n}{k} \rfloor - 1$ inclusive. El 1-ésimo va desde la fila $\lfloor \frac{n}{k} \rfloor$ hasta la $\lfloor \frac{2n}{k} \rfloor - 1$ inclusive. En general, el i-ésimo rango abarca desde la fila $\lfloor \frac{in}{k} \rfloor$ hasta la $\lfloor \frac{(i+1)n}{k} \rfloor - 1$ inclusive.
2. De manera análoga a las filas, se particionan las m columnas en l rangos, numerados de 0 a $l-1$. El j-ésimo rango abarca desde la columna $\lfloor \frac{jm}{l} \rfloor$ hasta la $\lfloor \frac{(j+1)m}{l} \rfloor - 1$ inclusive.
3. Se crea la miniatura dando a cada color (rojo, verde, azul) del píxel (i, j), la **intensidad promedio** en la imagen original de ese color, entre todos los píxeles cuyas filas están en el rango i de filas y sus columnas en el rango j de columnas.
4. Como el valor de intensidad debe ser siempre entero, si el promedio resulta ser p, la intensidad final será $\lfloor p \rfloor$

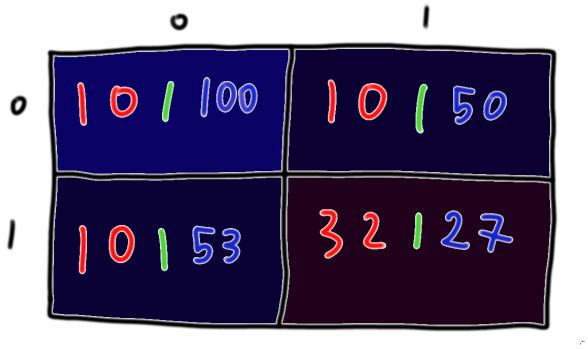
Tu tarea consiste en implementar un programa que calcule estas miniaturas rápidamente.

Las siguientes figuran muestran la imagen del ejemplo, con las regiones a promediar para cada miniatura:



Y las miniaturas resultantes del ejemplo son:





NOTA: La notación $\lfloor x \rfloor$ indica el mayor entero que es menor o igual que x. Por ejemplo, $\lfloor 3,5 \rfloor = \lfloor 3,1 \rfloor = \lfloor 3,99 \rfloor = \lfloor 3 \rfloor = 3$

Detalles de implementación

Se deben implementar dos funciones:

`inicializar(rojo, verde, azul)`, que será llamada al comenzar el programa para darte el mapa de bits original. `rojo`, `verde` y `azul` son matrices de $n \times m$ enteros.

`miniatura(k, l, rojo, verde, azul)`, que será llamada varias veces con distintos valores de k y l. La función debe escribir el resultado en los parámetros `rojo`, `verde` y `azul`, que serán matrices de $k \times l$ enteros inicializados en 0. En estas matrices se deben escribir las intensidades correspondientes de los píxeles de la miniatura de $k \times l$ generada por el procedimiento de cuadricular y promediar.

Evaluador local

El evaluador local lee de la entrada estándar:

- Una línea con tres enteros n, m y q
- Tres bloques de líneas, uno por cada color (rojo, verde, azul)
  - Cada bloque tiene n líneas
  - En cada bloque, cada una de las líneas tiene m enteros.
  - En cada bloque, el entero j de la línea i indica la intensidad de color correspondiente para el píxel (i, j)
- q líneas más. La i-ésima de estas líneas contiene dos enteros k_i, l_i, para una llamada a `miniatura`

El evaluador primero llama a `inicializar` con los datos provistos en los 3 bloques de la entrada, y luego realiza q llamadas a la función `miniatura`. En la i-ésima llamada utiliza los valores de la entrada k_i, l_i.

Por cada una de las q llamadas, escribe a la salida 3 bloques de k líneas, cada una con l enteros, en el mismo formato que la entrada para indicar las intensidades de rojo, verde y azul calculadas por la función en los parámetros `rojo`, `verde` y `azul`.

Restricciones

- $1 \leq k_i \leq n \leq 2000$
- $1 \leq l_i \leq m \leq 2000$
- La cantidad total de píxeles entre todas las q miniaturas será como mucho 1.000.000

Ejemplos

Si el evaluador local recibe la siguiente entrada:

```
3 3 2
10 10 10
10 100 10
10 10 10
1 1 1
1 1 1
1 1 1
100 100 1
100 100 2
6 5 3
1 3
2 2
```

Una implementación correcta debe devolver:

```
10 40 10
1 1 1
68 68 2
10 10
10 32
1 1
1 1
100 50
53 27
```

Subtareas

1. $n, m \leq 10$ (7 ptos.)
2. $k_i = 1$ para todo i (15 ptos.)
3. $k_i = k_j$ para todo i, j (25 ptos.)
4. $q = 1$ (9 ptos.)
5. $n, m \leq 100$ (28 ptos.)
6. $n, m \leq 1000$ (8 ptos.)
7. Sin más restricción (8 ptos.)