Sistemas Operativos
Departamento de Ingeniería en Informática
LAB2 : Programación multihebra

1 Objetivo

El objetivo de este laboratorio es conocer, usar y apreciar las características de paralelismo de un programa multi hebreado. Aunque no es necesario disponer de un multiprocesador para desarrollar y ejecutar este lab, sería provechoso que la ejecución lo realizaran en un computador con al menos dos núcleos.

En este laboratorio usted implementará un programa multi-hebreado que calcula el histograma de una imagen.

2 El Histograma

Sea A una imagen de $N \times N$ píxeles, tal que $0 \leq A_{ij} \leq 255$. El histograma de A es una función discreta unidimensional H, tal que

$$H_k = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} I(A_{ij} - k), \quad k = 0, 1, \ldots, 255 \tag{1}$$

donde I es la función indicador definida de la siguiente manera

$$I(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Note que la función indicador en H_k retorna el valor 1 cuando el pixel en la posición (i, j) tiene valor k. Es decir, H_k es el número de veces que el valor k aparece en la imagen.

Cada k representa un bin del histograma y en la equación (3) un bin corresponde a solo un valor de pixel. Pero esto no siempre es el caso. Es posible construir histogramas donde cada bin represente un subconjunto de valores. Por ejemplo, podríamos construir un histograma de las edades de los estudiantes del Diinf, usando los siguientes bins: [1-18], [19-21], [22-25], [26-31], [32,-]. Asi, H_0 sería el número de estudiantes con edades menores o igual a 18, H_1 el número de estudiantes con edades entre 19 y 21 años, etc.

El histograma puede ser normalizado tal que H_k represente la frecuencia (porcentaje) que un valor aparece en un conjunto de datos, es decir

$$H_k = \frac{1}{N \times N} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} I(A_{ij} - k), \quad k = 0, 1, \ldots, 255 \tag{3}$$

La siguiente figura muestra una imagen en niveles de gris y su respectivo histograma con 256 bins.

En cambio, la siguiente figura muestra el histograma para la misma imagen, pero con solo 16 bins. Los límites de los bins son: [0,15], [16,31], [32,47], [48,63], [64,79], [80,95], [96,11], [112,127], [128,143],

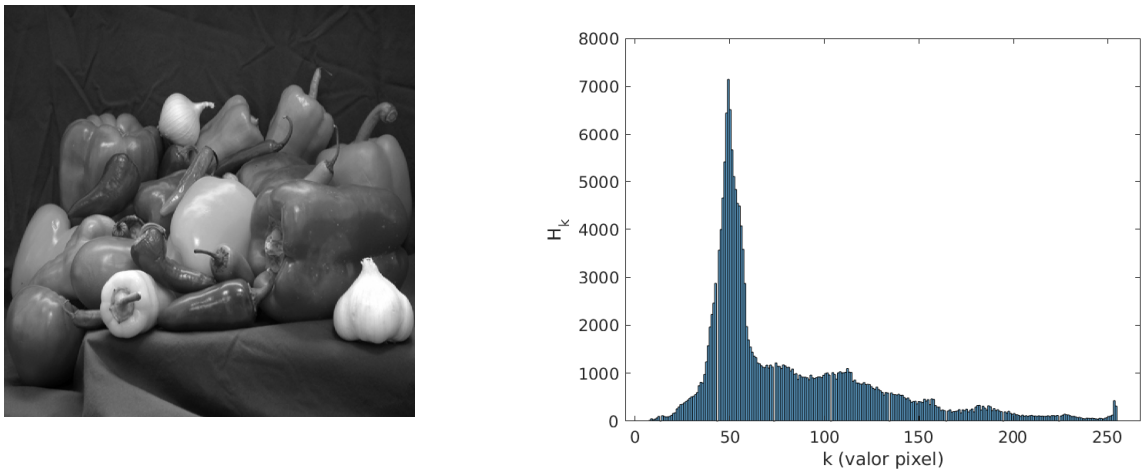

Figure 1: Imagen con valores de píxeles entre 0 y 255, y su correspondiente histograma con 256 bins.

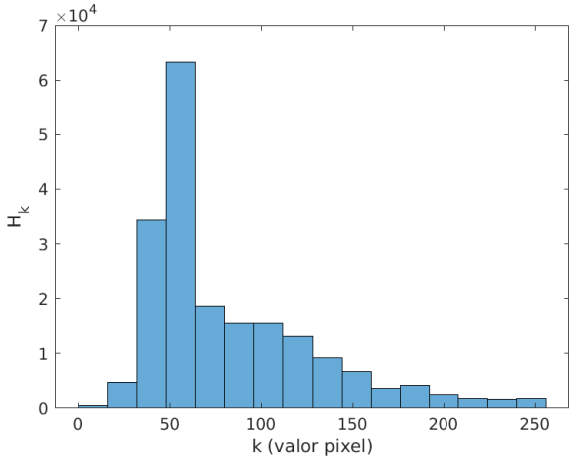

Figure 2: Histograma para la imagen anterior con 16 bins.

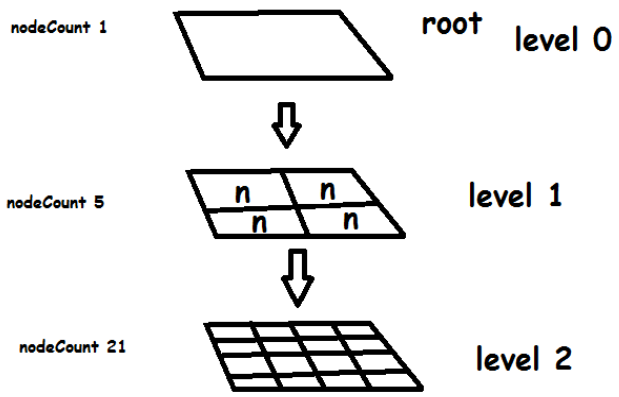

Figure 3: Descomposición recursiva de un domino 2D con un quadtree de 3 niveles.

[144,159], [160,175], [176,191], [192,207], [208,223], [224,239], [240,256]. Es decir, los valores de los píxeles son acumulados en bins de igual tamaño, que en este caso son de 16 valores.

Note que para un histograma discreto, se satisface la siguiente propiedad

$$\sum_k H_k = N^2$$

que es independiente del número de bins. Esta propiedad puede ser usada para verificar correctitud de cálculo del histograma.

3 Quadtree

Un quadtree o árbol cuaternario es una estructura de datos que almacena eficientemente datos bidimensionales. Un quadtree también se usa para particionar un espacio de datos bidimensionales, recursivamente subdividiéndolo en 4 cuadrantes. La subdivisión de algún cuadrante puede condicionarse según alguna propiedad de los píxeles del cuadrante, pero en nuestro caso siempre un cuadrante se subdividirá en cuatro subcuadrantes de igual tamaño. El número de niveles de descomposición estará dado por un argumento de línea de comando.

En este lab usaremos la lógica de descomposición del quadtree para dividir el trabajo de las hebras por cuadrantes.

La siguiente figura ejemplifica el particionamiento de una imagen con un quadtree de 3 niveles. Si la imagen del nivel 0 (imagen inicial) es una image cuadrada de $N \times N$, entonces, los cuadrantes del nivel 1, son de $N/2 \times N/2$, y los del nivel 2 sonde $N/4 \times N/4$. Y en general, el tamaño de los cuadrantes en el nivel i son de $N/2^i \times N/2^i$.

4 Cálculo paralelo del histograma

El objetivo de este lab es diseñar y construir un programa multihebreado que calcule el histograma de una imagen cuadrada, dividiendo el cálculo entre las hebras usando la descomposición quadtree.

La lógica del algoritmo es la siguiente:

1. La hebra inicial (main) crea una hebra en nivel 0.
2. La hebra nivel 0 crea 4 hebras, cada una de las cuales está encargada de calcular el histograma de sus respectivos cuadrantes.
3. Estas hebras son de nivel 1.
4. La hebra de nivel 0 espera que las 4 hebras de nivel 1 terminen su trabajo.
5. Si no se ha alcanzado el máximo nivel de descomposición, cada hebra de nivel 1, crea otras 4 hebras de nivel 2. Es decir las hebras no calculan el histograma sino que crean hebras hijas las cuales calculan el histograma de subcuadrantes de dicha hebra.
6. Cada hebra de nivel 1 espera a sus 4 hebras de nivel 2 a que finalicen su trabajo.
7. Este procedimiento se repite hasta que se alcanza el nivel máximo.
8. Cuando se alcanza el nivel máximo, cada hebra pasa a gris su porción de la imagen (ver ecuación 4) y además calcula el histograma de su subcuadrante. Por ejemplo, en la Figura 3, existen en total 13 hebras credas, pero solo las 8 hebras de nivel 2 calculan el histograma de sus cuadrantes.
9. El cálculo del histograma de cada subcuadrante se realiza en un histograma local a la hebra, es decir, cada hebra del último nivel procura memoria dinámica para su histograma. De esta forma no existe problemas de acceso concurrente a una misma área de memoria.
10. Note que el tamaño del histograma siempre es el mismo, independiente del nivel de la hebra.
11. Cuando 4 hebras de un nivel i han calculado cada una un histograma, le corresponde a la hebra madre, de nivel $i-1$, sumar dichos histogramas.
12. Recuerde que cuando una hebra termina su trabajo, ya sea calcula histograma o suma histogramas, finaliza su ejecución y es eliminada del sistema, por lo que es necesario disponer de un método para que su histograma no se destruya y lo reciba la hebra madre.
13. Así, la hebra madre suma los cuatro histogramas y crea uno consolidado.
14. Este procedimiento continua hasta alcanzar el nivel 0, donde se suman los útlimos 4 histrogramas y se obtien el histograma final.

$$Y = R * 0.3 + G * 0.59 + B * 0.11 \tag{4}$$

donde R, G, y B son los componentes rojos, verdes y azul, respectivamente.

5 Ejecución

El programa debe ser capaz de trabajar con imágenes de distinto tamaño y con distintos niveles de descomposición. Para simplificar, el tamaño de la imagen siempre será potencia de 2, es decir 2^k. De esta forma, siempre se podrá subdividir un cuadrante en 4 subcuadrantes de igual tamaño.

La siguiente es la línea de comandos

```
$ ./lab2 -i imagen.bmp -o histograma.txt -L niveles -B numerodebins
```

donde `-i` especifica la imagen de entrada que estará en formato BMP, `-o` indica el nombre del archivo de salida que contiene el histograma, `-L` es el nivel de descomposición. La opción `-B` especifica el número de bins del histograma, el cual siempre será también potencia de dos. Por ejemplo 256, 128, 64, etc.

El archivo de salida está en formato texto plano (ASCII), donde cada línea especifica el rango del bin y su frecuencia, separados por un tab. Por ejemplo, si el número de bins es 256, el archivo tendría el sigueinte formato

```
[0,     0]      12
[1,     1]      0
[2,     2]      21
[3,     3]      94
...
```

Note que hay un tab entre la coma y el valor del límte superior.

Para el ejemplo de 16 bins, el archivo de salida se vería de esta forma

```
[0,     15]     204
[16,    31]     188
[32,    47]     1023
...
```

El archivo de salida NO DEBE contener nada más que lo indicado.

El programa debe verificar que el archivo de entrada exista, que el número de niveles es un número entre 0 y 8 y sea potencia de dos, y que el número de bins sea un número positivo mayor a 1 y sea potencia de dos.

Cabe destacar

6 Entregables

Subir en uvirtual al menos los siguientes archivos:

1. `Makefile`: archivo para make que compila los programas
2. `hist.c`: archivo con el código. Puede incluir otros archivos fuentes. algoritmos.
3. Clara documentación, es decir, entregar códigos debidamente comentados, en donde a lo menos se incluya le descripción de cada función, sus parámetros de entrada y salida. Los comentarios deben seguir la siguiente estructura:

```
//Entradas: Explicar entradas, qué representan y su tipo de dato
//Funcionamiento: Explicación breve del funcionamiento
//Salidas: Explicar el tipo de dato de la salida y lo que representa
funcion( ... ){
//comentario especifico 1

//comentario específico 2

//...

}
```

Además, también se evalúan buenas prácticas de programación, respecto al nombramiento de variables con nombres significativos, etc.

El archivo comprimido debe llamarse: RUTESTUDIANTE1_RUTESTUDIANTE2.zip

Ejemplo: 19689333k_186593220.zip

NOTA1: los laboratorios son en parejas, las cuales deben ser de la misma sección. De lo contrario no se revisarán laboratorios.

NOTA2: se descuenta un punto por cada día de retraso.

Fecha de entrega: domingo 06 de diciembre antes de las 23:59 hrs.