Raimundo Herrera Sufán - 14632152

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2133 — Estructuras de Datos y Algoritmos — 1' 2017

Tarea 0

Análisis de complejidad

Tiempo de ejecución

El algoritmo empleado para la convolución de matrices utiliza la imagen en cuestión y el kernel a aplicar. Como ambos elementos tienen dimensiones variables, la complejidad en cuanto a tiempo de ejecución depende fundamentalmente de ellos.

El código relevante para el algoritmo corresponde a la parte donde se itera sobre las matrices. La estructura se muestra a continuación, omitiendo aquellas partes que no afectan el análisis:

```
for (int row = 0; row < height; row++) {
        for (int col = 0; col < width; col++) {
                // declaraciones
                for (int i = row - margin_rows; i < row + margin_rows + 1; i++){
                        for (int j = col - margin_cols; j < col + margin_cols + 1; j++){
                                // multiplicaciones, sumas, declaraciones y asignaciones
                        }
                }
                // asignaciones
        }
}
```

Así, como se ve, se cuenta con 4 for anidados, los cuales dependen de distintas variables. Los primeros dos del alto y ancho de la imagen, respectivamente. Los últimos, del alto y ancho del kernel, respectivamente. Para mayor entendimiento del código, cabe señalar que $margin\_rows$ y $margin\_cols$ corresponden a la mitad del alto y ancho del kernel respectivamente, y es por eso que se introduce dos veces en los márgenes del for, para que este realice las iteraciones correspondientes.

Sea A el alto de la imagen, B el ancho de la imagen, C el alto del kernel y D el ancho del kernel, entonces, la cantidad de iteraciones que se efectúan corresponde a $A * B * C * D$. De este modo, es necesario también considerar todas las declaraciones, operaciones y asignaciones. Estas se ejecutan tomando tiempo constante, llamémoslo k. Así, la cantidad de pasos que se ejecutan, o tiempo de ejecución en función de los términos A, B, C y D definidos anteriormente, es $Tiempo = k * A * B * C * D$.

En cuanto a la complejidad del algoritmo, como k es una constante, y estas no se consideran en el análisis, se obtiene que el tiempo es $\mathcal{O}(A * B * C * D)$, siendo A, B, C y D los anchos y altos de la imagen y el kernel respectivamente.

Uso de memoria

Para el análisis del uso de memoria corresponde considerar aquellos elementos que utilizan espacio del *stack* y del *heap*. En el *heap* se crean dos imagenes a partir del mismo archivo, pero con usos diferentes, una de *input* y otra de *output*, además se crea la matriz del kernel. En el *stack*, por otro lado, se genera una cantidad constante de variables fuera de los for, y cierta cantidad de variables dentro de ellos, pero, no se crean nuevas variables, sino que cada vez que se ejecuta una iteración, se sobreescriben, por lo que también se puede considerar como constante.

Las imagenes se representan como arreglos de pixeles, los cuales a su vez son elementos con tres números, que representan la intensidad de cada color (RGB), que van de 0 a 255, por lo tanto es necesario como mínimo 1 byte por color. Así, la cantidad de memoria por imagen correspondería a $3 * alto\_imagen * ancho\_imagen$, y son dos imagenes. Pero consideremos la cantidad de memoria que ocupa cada pixel como una constante k, entonces se obtiene que la memoria utilizada por las imagenes es $2 * k * alto\_imagen * ancho\_imagen$. Luego, el kernel guarda un $float$ por cada elemento, teniendo que si el tamaño de un $float$ es una constante c la memoria utilizada es $c * alto\_kernel * ancho\_kernel$. Finalmente, llamemos s a la memoria utilizada por el *stack*.

En resumen, la memoria utilizada por el programa corresponde a

$$Memoria = 2 * k * alto\_imagen * ancho\_imagen + c * alto\_kernel * ancho\_kernel + s$$

En términos de complejidad, ignorando constantes, la complejidad es:

$$\mathcal{O}(alto\_imagen * ancho\_imagen + alto\_kernel * ancho\_kernel)$$