NOMBRE: Rafael Fernández Sánchez

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

IIC2133 — Estructura de Datos y Algoritmos — 1' 2020

Informe Tarea 1 – Complejidad Creación QuadTree

Generamos el *QuadTree* a partir de una matriz de pixeles. Para esto usamos una función recursiva donde en cada paso dividimos la matriz de pixeles en 4, y creamos un nodo padre donde sus nodos hijos serán los 4 árboles resultantes de la llamada recursiva sobre las divisiones recién creadas de los pixeles. El caso base ocurre cuando la matriz recibida tiene solo 4 pixeles.

Además, se implementó el cálculo de las **desviaciones estándar** de los componentes de LAB en el mismo proceso de creación del árbol. Cada nodo del árbol, luego de su creación, almacena los valores σ y μ de los tres componentes de color, calculados considerando los nodos hojas que se pueden alcanzar al descender por ese nodo. Para esto, se usaron **métodos incrementales**, de modo que en cada paso recursivo se toma una **cantidad constante** de pasos para realizar el cómputo de los σ y μ.

El proceso de creación del *QuadTree* puede ser descrito por el siguiento pseudocódigo.

Algorithm 1 CrearQuadTree(pixeles)

if $|pixeles| = 4$ **then**
    $nodo = NuevoNodo()$
    $nodo.\mu_L = PromedioL(pixeles)$
    $nodo.\mu_A = PromedioA(pixeles)$
    $nodo.\mu_B = PromedioB(pixeles)$
    $nodo.\sigma_L = nodo.\sigma_A = nodo.\sigma_B = 0$
    return $nodo$
end if
$nodo = NuevoNodo()$
$hijoSupIzq = CrearQuadTree(pixeles/4)$
$hijoSupDer = CrearQuadTree(pixeles/4)$
$hijoInfIzq = CrearQuadTree(pixeles/4)$
$hijoInfDer = CrearQuadTree(pixeles/4)$
$nodo.hijos =$ {hijoSupIzq, hijoSupDer, hijoInfIzq, hijoInfDer}
$ComputarEstadsticasIncremental(nodo)$
return $nodo$

Notar que aprovechamos la estructura recursiva del *QuadTree* para generarlo fácilmente mediante recursión.

$$T(n) = \begin{cases} c_1 & \text{Para } n = 4. \\ 4T(\frac{n}{4}) + c_2 & \text{Para } n > 4. \end{cases} \tag{1}$$

Podemos ver que esta recurrencia es el caso con $a = b = 4$ y $d = 0$ del teorema maestro, según la notación del material de clases del curso IIC1253 2019-1. Luego, según el teorema maestro, tenemos lo siguiente:

$$CrearQuadTree \in \Theta(n^{\log_4 4})$$

$$CrearQuadTree \in \Theta(n)$$

Donde n corresponde a los pixeles de la imagen, y debido a que el lado de las imágenes usadas serán potencias de 2, entonces n es una potencia de 4.

Luego, la complejidad de *CrearQuadTree* es **lineal**.

NOMBRE: Rafael Fernández Sánchez

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

IIC2133 — Estructura de Datos y Algoritmos — 1' 2020

Informe Tarea 1 – Complejidad Filtrar

Para filtrar un imágen, primero generaremos el *QuadTree* corresponiente y luego recorreremos el árbol en profundidad desde la raíz. En el momento que encontremos un nodo con un $\gamma < \alpha$, terminaremos la búsqueda para ese nodo y sus descendientes. Recordar que el γ para cada nodo fue calculado cuando se generó el *QuadTree*. El filtrado de un imágen esta dado por los siguientes pseudocódigos.

Algorithm 2 FiltrarRecursivo(*nodo*, α)

```
1:  if Gamma(nodo) ≤ α then
2:      PintarPixelesPromedio(nodo)
3:      return
4:  end if
5:  if nodo es hoja then
6:      return
7:  end if
8:  for hijo in nodo.hijos do
9:      FiltrarRecursivo(nodo, α)
10: end for
11:
12: return
```

Algorithm 3 Filtrar(Imagen, α)

```
1: pixeles = ObtenerPixeles(imagen)
2: arbol = CrearQuadTree(pixeles)
3: FiltrarRecursivo(arbol, α)
```

Ya que solo nos interesa el tiempo de ejecución del filtrado en sí, **ignoraremos** los pasos que toman la línea 2 de FiltrarRecursivo y la línea 2 de Filtrar, que corresponden a operaciones sobre la imágen.

El **peor caso** corresponde cuando todos los nodos tienen $\gamma > \alpha$, lo que corresponde a que la imagen no sufra cambios. Es fácil ver que este peor caso tendrá una complejidad que **no depende** de α.

Para este peor caso, la estructura de $FiltrarRecursivo$ es muy parecida a la de $CrearQuadTree$, ya que en escencia estamos recorriendo el árbol en los dos casos. La recurrencia de $FiltrarRecursivo$ está dada por:

$$T(n) = \begin{cases} c & \text{Para } n = 1. \\ 4T(\frac{n}{4}) & \text{Para } n > 1. \end{cases} \tag{2}$$

Podemos usar el mismo argumento de antes para determinar entonces que $FiltrarRecursivo \in \mathcal{O}(n)$.

Notemos que como analizamos el peor caso, usamos $\mathcal{O}$ en vez de Θ.

Finalmente, la función *Filtrar* primero genera el *QuadTree* y luego usa la función $FiltrarRecursivo$. Como vimos, $CrearQuadTree \in n$, por lo que quedamos con

$$Filtrar \in \Theta(n) + \mathcal{O}(n)$$
$$Filtrar \in \mathcal{O}(n)$$

Siendo n el número de pixeles.

NOMBRE: Rafael Fernández Sánchez

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2133 — Estructura de Datos y Algoritmos — 1' 2020

Informe Tarea 1 – Complejidad Comprimir

Usamos busqueda binaria para encontrar el mínimo α tal que la cantidad de nodos hoja es $\leq h$.

El siguiente pseudocódigo describe las funciones utilizadas.

Algorithm 4 BusquedaBinariaAlpha(arbol, min, max, h, mejorAlpha)

```
mitad = min + (max - min) / 2
FiltrarRecursivo(arbol, mitad)
hojas = ContarHojas(arbol)
if max - min = 2 then
  if hojas ≤ h then
    return mitad
  end if
  return mejorAalpha
end if
if hojas ≤ h then
  mejorAlpha = mitad
  return BusquedaBinariaAlpha(arbol, min, mitad, h, mejorAlpha)
else
  return BusquedaBinariaAlpha(arbol, mitad, max, h, mejorAlpha)
end if
```

Algorithm 5 Comprimir(imagen, h)

```
pixeles = ObtenerPixeles(imagen)
arbol = CrearQuadTree(pixeles)
alpha = BusquedaBinariaAlpha(arbol, 0, 128, h, 128)
FiltrarRecursivo(arbol, alpha)
```

Usaremos los mismos supuestos de la complejidad de filtrar. En este caso podemos ver que en la busqueda binaria se diminuye el rango de busqueda en 2 hasta llegar llegar a un rango de busqueda de la forma $[i, i+2]$. Dado que estamos buscando un α en el rango $[0, 128]$, realizaremos 7 iteraciones hasta llegar al α buscado. En cada iteración ejecutamos un Filtrado, el cual es $\mathcal{O}(n)$ así como tambien un recuento de los nodos hojas que también es $\mathcal{O}(n)$ ya que recorre el arbol de manera similar al filtrado en el peor caso. Finalmente ejecutamos el filtro definitivo a la imagen. Por lo tanto,

$$Comprimir \in 7(\mathcal{O}(n) + \mathcal{O}(n)) + \mathcal{O}(n)$$

$$Comprimir \in \mathcal{O}(n)$$