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

IIC2283 Diseño y análisis de algoritmos
2° semestre 2019

Ayudantía 2

Dudas a Antonio López (alopez7@uc.cl)

Teorema maestro

Sea $f : \mathbb{N} \to \mathbb{R}_0^+$, a, b, $c \in \mathbb{R}_0^+$ tales que $a \geq 1$, $b > 1$ y $T(n)$ definida como la siguiente ecuación de recurrencia:

$$T(n) = \begin{cases} c & n = 0 \\ a \cdot T(\lceil n/b \rceil) + f(n) & n \geq 1 \end{cases}$$

Se tiene que:

1. Si $f(n) \in O(n^{\log_b(a)-\epsilon})$ para $\epsilon > 0$, entonces $T(n) \in \Theta(n^{\log_b(a)})$
2. Si $f(n) \in \Theta(n^{\log_b(a)})$, entonces $T(n) \in \Theta(n^{\log_b(a)} \cdot \log_2(n))$
3. Si $f(n) \in \Omega(n^{\log_b(a)+\epsilon})$ para $\epsilon > 0$ y f es $(a, b)-$regular, entonces $T(n) \in \Theta(f(n))$

f es $(a, b)-$regular si $\exists c \in \mathbb{R}^+$, $\exists n_0 \in \mathbb{N} : c < 1 \wedge (\forall n > n_0)(a \cdot f(\lfloor n/b \rfloor \leq c \cdot f(n))$

Sección 1: Dividir para conquistar y teorema maestro

1. El siguiente algoritmo sirve para encontrar el valor que minimiza en una curva convexa:

Algorithm 1 ternary_search$(g(\cdot), s, e, \epsilon)$

while $e - s > epsilon$ **do**
  $i = (e - s)/3 + s$
  $j = 2 \cdot (e - s)/3 + s$
  $g_i = g(i)$
  $g_j = g(j)$
  if $g_i \leq g_j$ **then**
    $e = j$
  if $g_j \leq g_i$ **then**
    $s = i$
end
return $(s - e)/2$

Expresa el tiempo tomado por el algoritmo usando una recurrencia y calcula su complejidad usando el teorema maestro

Solución: Expresaremos la recurrencia del algoritmo como $T(n)$ donde n es el tamaño del intervalo $(e - s)$ de la siguiente forma:

$$T(n) = \begin{cases} c & n \leq \epsilon \\ T(\lceil \frac{2 \cdot n}{3} \rceil) + c & n > \epsilon \end{cases}$$

c corresponde a las operaciones de evaluar g y comparar los valores.

Usamos el teorema maestro:

$$a = 1$$
$$b = \frac{3}{2}$$
$$\log_{\frac{3}{2}} 1 = 0$$
$$f(n) = c$$

- **Caso 1:** Comprobamos si existe $k > 0$, $e > 0$ y n_0 tal que $c \leq k \cdot n^{0-e} \forall n > n_0$

  $$\frac{c}{k} \leq n^{-e} \forall n > n_0$$

  Sabemos que n^{-e} es una función decreciente y que tiende a 0 cuando n tiende a ∞ por lo que no se cumple la relación para todo $n > n_0$. Por lo tanto no se cumple el primer caso del teorema maestro.
- **Caso 2:** Comprobamos si $c \in \Theta(n^0) = \Theta(1)$. Efectivamente se cumple por lo que:

  $$T(n) \in \Theta(\log_2 n)$$

2. Un KD-Tree es una estructura de datos que particiona el espacio K dimensional. Uno de sus usos es almacenar puntos de K dimensiones. Para esto se construye la estructura dividiendo los datos en 2 mitades según algún eje a elección según si los elementos son mayores o no a la mediana en ese eje. Luego se construye un KD-Tree recursivamente para ambas mitades. Cuando solo hay un elemento entonces se crea una hoja del árbol que contiene el punto.

   a) Expresa la recurrencia del algoritmo de construcción de KD-Tree. Considera que calcular la mediana de n puntos puede ser calculada en $c \cdot n$ pasos con $c \in \mathbb{R}^+$.

   Solución:

   $$T(n) = \begin{cases} c & n = 1 \\ T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + c \cdot n & n > 1 \end{cases}$$

   b) Usa el teorema maestro para resolver la recurrencia anterior.

   Solución: Dado que tenemos un techo y un piso podemos acotar la recurrencia por arriba y por abajo con otras 2 recurrencias: $T_1(n) \leq T(n) \leq T_2(n)$ donde:

   $$T_1(n) = \begin{cases} c & n = 1 \\ 2 \cdot T_1(\lfloor \frac{n}{2} \rfloor) + c \cdot n & n > 1 \end{cases}$$

$$T_2(n) = \begin{cases} c & n = 1 \\ 2 \cdot T_2(\lceil \frac{n}{2} \rceil) + c \cdot n & n > 1 \end{cases}$$

En ambas recurrencias podemos usar directamente el teorema maestro y van a cumplir los mismos casos por lo que usamos el teorema maestro en T_1 y se cumple lo mismo para T_2:

$$a = 2$$
$$b = 2$$
$$\log_b a = 1$$
$$f(n) = c \cdot n$$

- **Caso 1:** Comprobamos si existe $k > 0$, $e > 0$ y n_0 tal que $c \cdot n \leq k \cdot n^{1-e} \forall n > n_0$

  $$\frac{c}{k} \leq n^{-e} \forall n > n_0$$

  Sabemos que n^{-e} es una función decreciente y que tiende a 0 cuando n tiende a ∞ por lo que no se cumple la relación para todo $n > n_0$. Por lo tanto no se cumple el primer caso del teorema maestro.
- **Caso 2:** Comprobamos si $c \cdot n \in \Theta(n)$. Efectivamente se cumple por lo que:

  $$T(n) \in \Theta(n \cdot \log_2 n)$$

c) ¿Qué pasa al usar el teorema maestro si en la construcción del KD-Tree se calcula la mediana con un algoritmo que toma $c \cdot n \cdot \log(n)$ pasos?

Solución: En este caso la recurrencia queda como:

$$T(n) = \begin{cases} c & n = 1 \\ T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + c \cdot n \cdot \log n & n > 1 \end{cases}$$

Al igual que antes podemos construir T_1 y T_2 de manera de que $T_1(n) \leq T(n) \leq T_2(n)$ y poder usar el teorema maestro:

$$T_1(n) = \begin{cases} c & n = 1 \\ 2 \cdot T_1(\lfloor \frac{n}{2} \rfloor) + c \cdot n \cdot \log n & n > 1 \end{cases}$$

$$T_2(n) = \begin{cases} c & n = 1 \\ 2 \cdot T_2(\lceil \frac{n}{2} \rceil) + c \cdot n \cdot \log n & n > 1 \end{cases}$$

Usamos el teorema maestro para T_1:

$$a = 2$$
$$b = 2$$
$$\log_b a = 1$$
$$f(n) = c \cdot n \cdot \log n$$

- **Caso 1:** Comprobamos si existe $k > 0$, $e > 0$ y n_0 tal que $c \cdot n \cdot \log n \leq k \cdot n^{1-e} \forall n > n_0$

  $$\frac{c}{k} \cdot \log n \leq n^{-e} \forall n > n_0$$

  Siendo $\log n$ creciente y n^{-e} decreciente no se puede cumplir la relación.

- **Caso 2:** Comprobamos si $c \cdot n \cdot \log n \in \Theta(n)$.
  Sabemos que no se cumple ya que $n \cdot \log(n) \notin O(n)$
- **Caso 3:** Comprobamos si existe $k > 0$, $e > 0$, n_0 tal que $c \cdot n \cdot \log n \geq k \cdot n^{1+e} \forall n > n_0$:

  $$c \cdot \log n \geq k \cdot n^e \forall n > n_0$$

  Sabemos que n^e crece siempre más rápido que $\log n$ por lo que no se puede cumplir la relación. Por lo tanto no se cumple el tercer caso.

No se cumple ninguno de los 3 casos del teorema maestro por lo que no podemos encontrar la complejidad usando este teorema.

d) Calcula una cota para la recurrencia anterior desarrollándola con $n = 2^k$.

Solución: Dado que no pusimos usar el teorema maestro calcularesmos la recurrencia de $T(n)$ usando $n = 2^k$:

$$T(n) = T(2^k) \begin{cases} c & k = 0 \\ 2 \cdot T(2^{k-1}) + c \cdot 2^k \cdot k & k > 0 \end{cases}$$

$$T(2^k) = 2 \cdot T(2^{k-1}) + c \cdot 2^k \cdot k$$
$$T(2^k) = 2 \cdot [2 \cdot T(2^{k-2}) + c \cdot 2^{k-1} \cdot (k-1)] + c \cdot 2^k \cdot k$$
$$T(2^k) = 2^2 \cdot T(2^{k-2}) + c \cdot 2^k \cdot (k-1) + c \cdot 2^k \cdot k$$
$$T(2^k) = 2^2 \cdot [2 \cdot T(2^{k-3}) + c \cdot 2^{k-2} \cdot (k-2)] + c \cdot 2^k \cdot (k-1) + c \cdot 2^k \cdot k$$
$$T(2^k) = 2^3 \cdot T(2^{k-3}) + c \cdot 2^k \cdot (k-2) + c \cdot 2^k \cdot (k-1) + c \cdot 2^k \cdot k$$

$$T(2^k) = 2^i \cdot T(2^{k-i}) + c \cdot 2^k \cdot \sum_{j=0}^{i-1} (k-j)$$

Cuando $i = k$, $T(2^{k-i}) = c$ por lo que:

$$T(2^k) = c \cdot 2^k + c \cdot 2^k \cdot \sum_{j=0}^{k-1} (k-j)$$

La sumatoria se puede escribir como:

$$\sum_{j=0}^{k-1} (k-j) = \sum_{j=1}^{k} j = \frac{k \cdot (k-1)}{2}$$

Por lo tanto:

$$T(2^k) = c \cdot 2^k + c \cdot 2^k \cdot \frac{k \cdot (k-1)}{2}$$

Usando: $2^k = n$ y $k = \log_2 n$:

$$T(n) = T(2^k) = c \cdot n + c \cdot n \cdot \frac{\log_2 n \cdot (\log_2 n - 1)}{2}$$

$$T(n) = c \cdot n + \frac{c}{2} \cdot n \cdot \log_2^2 n - c \cdot n \cdot \frac{log_2 n}{2}$$

Podemos probar que $T(n) \in O(n \log_2^2 n)$:

$$c \cdot n + \frac{c}{2} \cdot n \cdot \log_2^2 n - c \cdot n \cdot \frac{log_2 n}{2} < k \cdot n \log_2^2 n$$

ya que

$$c \cdot n - c \cdot n \cdot \frac{log_2 n}{2} < (k - \frac{c}{2}) \cdot n \log_2^2 n$$

$$c \cdot (1 - \frac{log_2 n}{2}) < (k - \frac{c}{2}) \cdot \log_2^2$$

$$c - \frac{log_2 n}{2} < (k - \frac{c}{2}) \cdot \log_2^2$$

Dado que la parte izquierda es una función decreciente, basta con que $k > \frac{c}{2}$ para que se cumpla la relación con n suficientemente grande. Por lo tanto $T(n) \in O(n \cdot \log_2^2 n)$.

Por otro lado podemos probar que $T(n) \in \Omega(n \cdot \log_2^2 n)$:

$$c \cdot n + \frac{c}{2} \cdot n \cdot \log_2^2 n - c \cdot n \cdot \frac{log_2 n}{2} > k \cdot n \log_2^2 n$$

Si elegimos $k = \frac{c}{4}$:

$$c \cdot n + \frac{c}{4} \cdot n \cdot \log_2^2 n - c \cdot n \cdot \frac{log_2 n}{2} > 0$$

Esto se cumple con n suficientemente grande por lo que $T(n) \in \Omega(n \cdot \log_2^2 n)$.
Por lo tanto $T(n) \in \Theta(n \cdot \log_2^2 n)$ cuando $n = 2^k$.

e) Usa inducción constructiva para demostrar que $T_1(n) \in O(n \log_2^2(n))$. (cambié el ejercicio porque tiene que tener piso y no techo para demostrarlo)

Solución: Primero lo demostraremos para los casos base y luego construiremos el caso inductivo.

$$T_1(n) = \begin{cases} c & n = 1 \\ 2 \cdot T_1(\lfloor \frac{n}{2} \rfloor) + c \cdot n \cdot \log_2 n & n > 1 \end{cases}$$

- **Caso base:** $T_1(1)$

  $$T_1(1) = c < k \cdot 1 \cdot log_2^2(1) = 0$$

  Esto no se cumple ya que $c > 0$. Por lo tanto debemos probar con un caso base mayor.
- **Caso base:** $T_1(2)$

  $$T_1(2) = 2 \cdot T_1(1) + c \cdot 2 \cdot log_2^2(2) = 4c < k \cdot 2 \cdot log_2^2(2) = 2k$$
  $$2c < k$$

- **Caso base:** $T_1(3)$. Ya que $T_1(3)$ depende de $T_1(1)$, también debemos demostrarlo para $T_1(3)$:

  $$T_1(3) = 2 \cdot T_1(1) + 3 \cdot c \cdot log_2^2(3) = 2c + c \cdot 3 \cdot log_2 3 < k \cdot 3 \cdot log_2^2(3)$$
  $$k > \frac{2c + 3 \cdot c \cdot log_2 3}{3 \cdot log_2^2(3)} \approx 1{,}66c$$

  Por lo tanto $k > 2c$ nos sirve para ambos casos base.
- **Caso inductivo:** $T_2(n)$
  Nuestra hipótesis de inducción dice que $T_1(i) < k \cdot i \cdot \log_2^2(i)$, para $1 < i < n$. Dado esto queremos demostrar que $T_1(n) < k \cdot n \log_2^2(n)$:

  $$T_1(n) = 2 \cdot T_1\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + c \cdot n \cdot \log_2(n)$$

$$< 2 \cdot k \cdot \left\lfloor \frac{n}{2} \right\rfloor \cdot \log_2^2 \left(\left\lfloor \frac{n}{2} \right\rfloor \right) + c \cdot n \cdot \log_2(n)$$

$$\leq 2 \cdot k \cdot \frac{n}{2} \cdot \log_2^2 \left(\frac{n}{2} \right) + c \cdot n \cdot \log_2(n)$$

$$= k \cdot n \cdot (\log_2(n) - 1)^2 + c \cdot n \cdot \log_2(n)$$

$$= k \cdot n \cdot (\log_2^2(n) - 2 \cdot \log_2(n) + 1) + c \cdot n \cdot \log_2(n)$$

$$= kn \log_2^2(n) - 2kn \log_2(n) + kn + cn \log_2(n)$$

Demostraremos que esto es menor a $kn \log_2^2(n)$:

$$kn \log_2^2(n) - 2kn \log_2(n) + kn + cn \log_2(n) < kn \log_2^2(n)$$

$$-2kn \log_2(n) + kn + cn \log_2(n) < 0$$

$$k(-2n \log_2(n) + n) + cn \log_2(n) < 0$$

$$k(-2n \log_2(n) + n) < -cn \log_2(n)$$

$$k > \frac{-cn \log_2(n)}{-2n \log_2(n) + n}$$

$$k > \frac{cn \log_2(n)}{2n \log_2(n) - n}$$

Este funcion (de la derecha) es decreciente, y cuando $n = 2$ es igual a c. Por lo tanto cuando $k > 2$ de cumple que $T_1(n) < k \cdot n \log_2^2(n) \forall n \geq 2$. Por lo tanto $T_1(n) \in O(n \cdot \log_2^2(n))$