Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
IIC1253 - Matemáticas Discretas

Ayudantía 9 - Algoritmos y complejidad

31 de mayo de 2024

Martín Atria, Paula Grune, Caetano Borges

Resumen

1 Algoritmos

Definimos un cerro como una lista consistiendo en una serie estrictamente creciente seguida de una serie estrictamente decreciente. Por ejemplo:

$$[1, 2, 4, 19, 8, 3]$$
$$[-5, 2, 7, 10, 15, 6, 5, 4, 1, 0]$$

son cerros.

1. Escriba un algoritmo que utilice una estrategia de dividir y conquistar que reciba como input un cerro, y entregue como output el valor máximo de este.
2. Calcule la complejidad de su algoritmo.

Intente crear un algoritmo que sea $O(\log(n))$.

Solución

a)
Input: un cerro $A = [a_0, \ldots, a_{n-1}]$.
Output: el valor máximo de A.

CerroSearch($A = [a_0, \dots, a_{n-1}]$)

$a \leftarrow 0$
$b \leftarrow n - 1$
$m \leftarrow \lfloor \frac{a+b}{2} \rfloor$
if $b == 0$ **then**
  return $A[0]$
else if $A[m] > A[m+1]$ **then**
  return CerroSearch($A[: m+1]$)
else if $A[m] < A[m+1]$ **then**
  return CerroSearch($A[m+1 :]$)
end if

b) Contaremos la cantidad de comparaciones de $T(n)$, donde n es el largo del input. Consideraremos el peor caso (que ocurre cuando el arreglo está ordenado). Consideremos la siguiente ecuación de recurrencia:

$$T(n) = \begin{cases} 1 & n = 1 \\ T(\lceil \frac{n}{2}) \rceil) + 3 & n > 1 \end{cases}$$

Supongamos que $n = 2^k$, con $k \in \mathbb{N}$. Tenemos que

$$\begin{aligned} T(n) = T(2^k) &= T\left(\frac{2^k}{2}\right) + 3 \\ &= T(2^{k-1}) + 3 \\ &= (T(2^{k-2}) + 3) + 3 \\ &= T(2^{k-2}) + 2 \cdot 3 \\ &\vdots \\ &= T(2^{k-i}) + i \cdot 3 \\ &\vdots \\ &= T(2^{k-k}) + k \cdot 3 \\ &= T(1) + k \cdot 3 \\ &= 3k + 1 \end{aligned}$$

Volviendo a términos de n, tenemos que $k = \log_2(n)$, por lo que

$$T(n) = 3\log_2(n) + 1$$

De lo que concluímos que $T(n) \in O(\log_2(n) \mid \text{POTENCIA}_2)$.
Además, dado que $\log_2(n)$ es asintóticamente no decreciente, 2-ármonica y $T(n)$ es asintóticamente no decreciente, concluímos que

$$T(n) \in O(\log_2(n)) = O(log(n))$$

2 Complejidad

Sean $f : \mathbb{N} \to \mathbb{R}^+$y $g : \mathbb{N} \to \mathbb{R}^+$dos funciones cualesquiera. Demuestre o entregue un contraejemplo para las siguientes afirmaciones:

1. Si $f(n) \in \Theta(g(n))$ entonces $\text{mín}\ \{f(n), g(n)\} \in \Theta(\ \text{máx}\ \{f(n), g(n)\})$.
2. Si $f(n) \in O(g(n))$ entonces $f(n)^{g(n)} \in O\left(g(n)^{f(n)}\right)$.

Solución

1. Definiendo $h(n) = \min\{f(n), g(n)\}$ y $H(n) = \max\{f(n), g(n)\}$ Como $f \in \Theta(g)$, existen constantes $c_1, c_2 \in R, n_o \in N$ tal que

$$c_1 g(n) \leq f(n) \leq c_2 g(n) \forall n \geq n_0 \tag{1}$$

Despejando de la parte derecha de la desigualdad

$$\frac{1}{c_2} f(n) \leq g(n) \forall n \geq n_0 \tag{2}$$

Tenemos dos escenarios desde $n \geq n_0$:
1. Cuando $f(n) \leq g(n), h(n) = f(n) \wedge H(n) = g(n)$. Usando (1) y la condición de este caso:

$$c_1 g(n) \leq f(n) \leq 1 \cdot g(n) \text{ para } n \geq n_0 \text{ tq } f(n) \leq 1 \cdot g(n)$$
$$\min\left\{c_1, \frac{1}{c_2}\right\} g(n) \leq c_1 g(n) \leq f(n) \leq 1 \cdot g(n)$$
$$\min\left\{c_1, \frac{1}{c_2}\right\} H(n) \leq h(n) \leq 1 \cdot H(n)$$

2. Cuando $g(n) < f(n), h(n) = g(n) \wedge H(n) = f(n)$ Con (2) queda:

$$\frac{1}{c_2} f(n) \leq g(n) \leq 1 \cdot f(n) \text{ para } n \geq n_0 \text{ tq } f(n) > g(n)$$
$$\min\left\{c_1, \frac{1}{c_2}\right\} f(n) \leq \frac{1}{c_2} f(n) \leq g(n) \leq 1 \cdot f(n)$$
$$\min\left\{c_1, \frac{1}{c_2}\right\} H(n) \leq \frac{1}{c_2} f(n) \leq h(n) \leq 1 \cdot H(n)$$

Combinando ambos casos:

$$\min\left\{c_1, \frac{1}{c_2}\right\} H(n) \leq h(n) \leq 1 \cdot H(n) \quad \forall n \geq n_0$$
$$c_1' H(n) \leq h(n) \leq c_2' H(n) \quad \forall n \geq n_0$$

Con lo cual, $h(n) \in \Theta(H(n))$

2. La afirmación anterior es falsa, se puede demostrar a través de un contraejemplo: Sea $f(n) = 2$ y $g(n) = n$. De acuerdo a la jerarquía en notación $\mathcal{O}$ vista en clases se cumple que $f(n) \in \mathcal{O}(g(n)$. Se debe demostrar ahora que $f(n)^{g(n)} \notin \mathcal{O}\left(g(n)^{f(n)}\right)$: Nuevamente si consideramos la jerarquía en notación $\mathcal{O}$ vista en clases, si $f(n)^{g(n)} = 2^n$ y $g(n)^{f(n)} = n^2$. Entonces no se cumple que $f(n)^{g(n)} \in \mathcal{O}\left(g(n)^{f(n)}\right)$, lo que es equivalente a $f(n)^{g(n)} \notin \mathcal{O}\left(g(n)^{f(n)}\right)$. Por demostración a través de contraejemplo queda demostrado que la afirmación no se cumple para dos funciones arbitrarias.

3 Complejidad + inducción

Considere la siguiente ecuación de recurrencia:

$$T(n) = \begin{cases} 1 & \text{si } n = 1 \\ 4 \cdot T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n^2 \log_2(n) & \text{si } n > 1 \end{cases}$$

Demuestre usando inducción que $T(n) \in O\left(n^2 (\log n)^2\right)$. Puede que los siguientes valores le resulten útiles:

$$\log_2(3) \approx 1,6 \quad \log_2(5) \approx 2,3 \quad \log_2(6) \approx 2,6 \quad \log_2(7) \approx 2,8$$

Solución

Por definición de O asintótica, tenemos que $g \in O(f)$ si y solo si

$$\left(\exists c \in \mathbb{R}^+\right) (\exists n_0 \in \mathbb{N}) (\forall n \geq n_0) (g(n) \leq c \cdot f(n))$$

Demostraremos por inducción que $T(n) \in O\left(n^2 (\log n)^2\right)$. Usaremos logaritmo en base 2, pues es el que aparece en la ecuación de recurrencia. Debemos encontrar $c \in \mathbb{R}^+$y $n_0 \in \mathbb{N}$ tales que para todo $n \geq n_0$ se cumpla que

$$T(n) \leq c \cdot n^2 \left(\log_2(n)\right)^2$$

Para esto, vamos a inspeccionar los primeros valores de $T(\cdot)$, y de esta forma trataremos de inferir ambas constantes. Reemplazando en la ecuación de recurrencia, y comparando con los valores de $n^2 \left(\log_2(n)\right)^2$, tenemos que:

$$\begin{aligned}
&T(1) = 1 && \not\leq 1^2(\log_2(1))^2 = 0 \\
&T(2) = 4 \cdot T(1) + 2^2 \cdot \log_2(2) = 4 + 4 \cdot 1 = 8 && \not\leq 2^2(\log_2(2))^2 = 4 \\
&T(3) = 4 \cdot T(1) + 3^2 \cdot \log_2(3) \approx 4 + 9 \cdot 1.6 = 18.4 && \leq 3^2(\log_2(3))^2 \approx 22.6 \\
&T(4) = 4 \cdot T(2) + 4^2 \cdot \log_2(4) = 4 \cdot 8 + 16 \cdot 2 = 64 && \leq 4^2(\log_2(4))^2 = 64 \\
&T(5) = 4 \cdot T(2) + 5^2 \cdot \log_2(5) \approx 4 \cdot 8 + 25 \cdot 2.3 = 89.5 && \leq 5^2(\log_2(5))^2 \approx 134 \\
&T(6) = 4 \cdot T(3) + 6^2 \cdot \log_2(6) \approx 4 \cdot 18.4 + 36 \cdot 2.6 = 167.2 && \leq 6^2(\log_2(6))^2 \approx 240
\end{aligned}$$

Teniendo estos valores en cuenta, podemos observar que para $n \geq 3$ se cumple que si $c = 1$ entonces $T(n) \leq c \cdot n^2 \left(\log_2(n)\right)^2$. Demostraremos entonces, por inducción fuerte, que

$$T(n) \leq n^2 \left(\log_2(n)\right)^2 \quad \forall n \geq 3$$

BI: Como la propiedad no se cumple para $T(1)$ y $T(2)$, todos los casos que involucren estos subcasos en la ecuación de recurrencia serán casos base. Dado lo anterior, los casos bases son $n \in \{3, 4, 5\}$ (cuando $n = 6$ se usa $T(3)$, que ya sería un caso base; por lo tanto, $n = 6$ no es un caso base). Como vimos antes, para $n \in \{3, 4, 5\}$ se cumple la propiedad.

HI: Suponemos que para todo $k \in \{3, \ldots, n-1\}$ se cumple la propiedad.

TI: Por demostrar que $T(n) \leq n^2 \left(\log_2(n)\right)^2$, para $n \geq 6$.

$$\begin{aligned}
T(n) &= 4 \cdot T(\lfloor \frac{n}{2} \rfloor) + n^2 \log_2(n) && \text{como } 3 \leq \lfloor \frac{n}{2} \rfloor < n \text{ aplicamos la HI} \\
&\leq 4 \cdot \lfloor \frac{n}{2} \rfloor^2 \cdot (\log_2 \lfloor \frac{n}{2} \rfloor)^2 + n^2 \log_2(n) \\
&\leq 4 \cdot (\frac{n}{2})^2 \cdot (\log_2(\frac{n}{2})^2 + n^2 \log_2(n) \\
&= n^2 (\log_2(\frac{n}{2}))^2 + n^2 \log_2(n) \\
&= n^2 ((\log_2(\frac{n}{2}))^2 + \log_2(n)) \\
&= n^2 ((\log_2(n) - \log_2(2))^2 + \log_2(n)) \\
&= n^2 ((\log_2(n) - 1)^2 + \log_2(n)) \\
&= n^2 ((\log_2(n))^2 - 2\log_2(n) + 1 + \log_2(n)) \\
&= n^2 ((\log_2(n))^2 - \log_2(n) + 1) && \text{como } n \geq 3,\ -\log_2(n) + 1 \leq 0 \\
&\leq n^2 (\log_2(n))^2
\end{aligned}$$

Con esto hemos demostrado que para todo $n \geq 3$ se cumple que $T(n) \leq n^2 (\log_2(n))^2$, por lo que $T(n) \in O(n^2 (\log_2(n))^2)$.

Observación: este ejercicio también se podía demostrar utilizando otro c y su n_0 correspondiente, pero el procedimiento de la inducción es análogo al caso anterior. Lo importante es fijar el c y el n_0 antes de empezar la inducción.