Análisis de eficiencia

Teorema Maestro

Segundo semestre 2022

IIC2283

Prof. Nicolás Van Sint Jan

Recordatorio: Búsqueda binaria

El siguiente es un posible pseudo-código para el algoritmo de **búsqueda binaria**:

BúsquedaBinaria(a, L, i, j)
  if $i > j$ **then return** no
  else if $i = j$ **then**
    if $L[i] = a$ **then return** i
    else return no
  else
    $p := \lfloor \frac{i+j}{2} \rfloor$
    if $L[p] < a$ **then return BúsquedaBinaria**$(a, L, p+1, j)$
    else if $L[p] > a$ **then return BúsquedaBinaria**$(a, L, i, p-1)$
    else return p

Llamada inicial al algoritmo: **BúsquedaBinaria**$(a, L, 1, n)$

Recordatorio: Tiempo de ejecución de búsqueda binaria

Si contamos sólo las **comparaciones**, entonces la siguiente expresión define la complejidad del algoritmo **BusquedaBinaria**:

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

donde $c \in \mathbb{N}$ y $d \in \mathbb{N}$ son constantes tales que $c \geq 1$ y $d \geq 1$.

Esto se conoce como una **ecuación de recurrencia**.

¿Cómo podemos solucionar la ecuación anterior?

Recordatorio: Sustitución de variables

Si asumimos que $n = 2^k$:

$$\begin{aligned} T(2^k) &= T(2^{k-1}) + d \\ &= (T(2^{k-2}) + d) + d \\ &= T(2^{k-2}) + 2d \\ &= (T(2^{k-3}) + d) + 2d \\ &= T(2^{k-3}) + 3d \\ &= \cdots \end{aligned}$$

Deducimos la expresión general para $k - i \geq 0$:

$$T(2^k) = T(2^{k-i}) + i \cdot d$$

Recordatorio: Sustitución de variables

Considerando $i = k$ obtenemos:

$$\begin{aligned} T(2^k) &= T(1) + k \cdot d \\ &= c + k \cdot d \end{aligned}$$

Dado que $k = \log_2(n)$, obtenemos que

$$T(n) = c + d \cdot \log_2(n)$$

para n potencia de 2.

Usando **inducción constructiva** vamos a extender esta solución y demostrar que $T(n) \in O(\log_2(n))$.

Recordatorio: Demostración

Vamos a demostrar:

$$\forall n \geq 2.\ T(n) \leq e \cdot \log_2(n)$$

Demostración

Casos base:

$$\begin{array}{rcccl} T(2) & = & c + d & = & e \cdot \log_2(2) \\ T(3) & = & c + d & < & e \cdot \log_2(3) \end{array}$$

Caso inductivo:

Suponemos que $n \geq 4$ y para todo $k \in \{2, \ldots, n-1\}$ se tiene que $T(k) \leq e \cdot \log_2(k)$

Recordatorio: Demostración

Demostración

Usando la definición de $T(n)$ y la hipótesis de inducción concluimos que:

$$\begin{aligned}
T(n) &= T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + d \\
&\leq e \cdot \log_2\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + d \\
&\leq e \cdot \log_2\left(\frac{n}{2}\right) + d \\
&= e \cdot \log_2(n) - e \cdot \log_2(2) + d \\
&= e \cdot \log_2(n) - (c + d) + d \\
&= e \cdot \log_2(n) - c \\
&< e \cdot \log_2(n)
\end{aligned}$$

$\square$

Outline

Ecuaciones de recurrencia (cont.)

Teorema maestro

Outline

Un segundo ejemplo de inducción constructiva

Considere la siguiente ecuación de recurrencia:

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

Queremos determinar una función $f(n)$ para la cual se tiene que $T(n) \in O(f(n))$.

¿ Alguna conjetura sobre que función podría ser $f(n)$?

Una posible solución para la ecuación de recurrencia

Dada la forma de la ecuación de recurrencia, podríamos intentar primero con

$$f(n) = n!$$

Tenemos entonces que determinar $c \in \mathbb{R}^+$ y $n_0 \in \mathbb{N}$ tales que

$$T(n) \leq c \cdot n!$$

para todo $n \geq n_0$.

Pero nos vamos a encontrar con un problema al tratar de usar la **hipótesis de inducción**.

Una posible solución para la ecuación de recurrencia

Supongamos que la propiedad se cumple para n:

$$T(n) \quad \leq \quad c \cdot n!$$

Tenemos que:

$$\begin{aligned} T(n+1) &= (n+1)^2 + (n+1) \cdot T(n) \\ &\leq (n+1)^2 + (n+1) \cdot (c \cdot n!) \\ &= (n+1)^2 + c \cdot (n+1)! \end{aligned}$$

¿ Cómo continuamos ?

No podemos continuar ya que no existe una constante c para la cual

$$(n+1)^2 + c \cdot (n+1)! \leq c \cdot (n+1)!$$

dado que $n \in \mathbb{N}$.

¿Cómo solucionamos el problema con la demostración?

Idea

Una demostración por inducción puede hacerse más simple considerando una **propiedad más fuerte**.

- Dado que la hipótesis de inducción se va a volver más fuerte

Vamos a seguir tratando de demostrar que $T(n) \in O(n!)$ pero ahora considerando una propiedad más fuerte.

Vamos a demostrar lo siguiente:

$$\exists c \in \mathbb{R}^+.\ \exists d \in \mathbb{R}^+.\ \exists n_0 \in \mathbb{N}.\ \forall n \geq n_0.\ T(n) \leq c \cdot n! - d \cdot n$$

¿ Por qué esta propiedad es **más fuerte** que la anterior ?

Inducción constructiva sobre una propiedad más fuerte

Para tener una mejor idea de los posible valores para c, d y n_0 vamos a considerar primero el paso inductivo en la demostración.

Supongamos que la propiedad se cumple para n:

$$T(n) \quad \leq \quad c \cdot n! - d \cdot n$$

Tenemos que:

$$\begin{aligned}
T(n+1) \quad &= \quad (n+1)^2 + (n+1) \cdot T(n) \\
&\leq \quad (n+1)^2 + (n+1) \cdot (c \cdot n! - d \cdot n) \\
&= \quad c \cdot (n+1)! + (n+1)^2 - d \cdot n \cdot (n+1) \\
&= \quad c \cdot (n+1)! + ((n+1) - d \cdot n) \cdot (n+1)
\end{aligned}$$

Inducción constructiva sobre una propiedad más fuerte

Para poder demostrar que la propiedad se cumple para $n+1$ necesitamos que lo siguiente sea cierto:

$$(n+1) - d \cdot n \quad \leq \quad -d$$

De lo cual concluimos la siguiente restricción para d:

$$\boxed{\frac{n+1}{n-1} \ \leq \ d}$$

Dado un n_0 apropiado ¿Qué d nos podría servir para $n \geq n_0$?

Si consideramos $n \geq 2$ concluimos que $d \geq 3$.

Consideramos entonces $n_0 = 2$ y $d = 3$

Inducción constructiva sobre una propiedad más fuerte

Para concluir la demostración debemos considerar el caso base $n_0 = 2$.
Tenemos que:

$$\begin{array}{lcccc} T(0) & = & 0 & & \\ T(1) & = & 1^2 + 1 \cdot T(0) & = & 1 \\ T(2) & = & 2^2 + 2 \cdot T(1) & = & 6 \end{array}$$

Entonces se debe cumplir que $T(2) \leq c \cdot 2! - 3 \cdot 2$, vale decir,

$$6 \ \leq \ c \cdot 2 - 6$$

Concluimos que $c \geq 6$, por lo que consideramos $c = 6$

- Tenemos entonces que

$$\forall n \geq 2.\ T(n) \leq 6 \cdot n! - 3 \cdot n$$

de lo cual concluimos que $T(n) \in O(n!)$.

Outline

El Teorema Maestro

Muchas de las **ecuaciones de recurrencia** que vamos a usar en este curso tienen la siguiente forma:

$$T(n) = \begin{cases} c & n = 0 \\ a \cdot T\left(\left\lfloor \frac{n}{b} \right\rfloor\right) + f(n) & n \geq 1 \end{cases}$$

donde a, b y c son constantes, y $f(n)$ es una función arbitraria.

El **Teorema Maestro** nos dirá cuál es el **orden** de $T(n)$ dependiendo de ciertas condiciones sobre a, b y $f(n)$.

¿ Qué pasa si cambiamos $\left\lfloor \frac{n}{b} \right\rfloor$ por $\left\lceil \frac{n}{b} \right\rceil$?

R: El Teorema Maestro también se puede utilizar cuando $\left\lfloor \frac{n}{b} \right\rfloor$ es reemplazado por $\left\lceil \frac{n}{b} \right\rceil$.

Una condición de regularidad sobre funciones

Antes de dar el enunciado del Teorema Maestro necesitamos definir una **condición de regularidad** sobre la función $f(n)$.

Sea $f : \mathbb{N} \to \mathbb{R}_0^+$ una función y $a, b \in \mathbb{R}$ constantes tales que $a \geq 1$ y $b > 1$.

Definición

La función f es **(a, b)-regular** si existen constantes $c \in \mathbb{R}^+$ y $n_0 \in \mathbb{N}$ tales que $c < 1$ y

$$\forall n \geq n_0.\ a \cdot f\left(\left\lfloor \frac{n}{b} \right\rfloor\right) \leq c \cdot f(n)$$

Ejercicio

1. Demuestre que las funciones n, n^2 y 2^n son (a, b)-regulares si $a < b$.
2. Demuestre que la función $\log_2(n)$ no es (1,2)-regular.

Una solución al segundo problema

Solución al ejercicio 2

Por contradicción, supongamos que $\log_2(n)$ es (1,2)-regular.

Entonces existen constantes $c \in \mathbb{R}^+$ y $n_0 \in \mathbb{N}$ tales que $c < 1$ y

$$\forall n \geq n_0. \log_2 \left\lfloor \frac{n}{2} \right\rfloor \leq c \cdot \log_2(n)$$

En particular, podemos concluir que para todo $k \geq n_0$:

$$\log_2 \left\lfloor \frac{2 \cdot k}{2} \right\rfloor \leq c \cdot \log_2(2 \cdot k)$$

Vale decir:

$$\log_2(k) \leq c \cdot (\log_2(k) + 1)$$

Una solución al segundo problema

Solución al ejercicio 2 (continuación)

$$\log_2(k) \leq c \cdot (\log_2(k) + 1)$$

Dado que $0 < c < 1$, concluimos que:

$$\log_2(k) \leq \frac{c}{1 - c}$$

Lo cual nos lleva a una contradicción (**¿Por qué?**). $\square$

El enunciado del Teorema Maestro

Teorema Maestro

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

$$T(n) \quad = \quad \begin{cases} c & n = 0 \\ a \cdot T\left(\left\lfloor \frac{n}{b} \right\rfloor\right) + f(n) & n \geq 1 \end{cases}$$

Se tiene que:

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

Usando el Teorema Maestro

Ejemplo

Considere la siguiente ecuación de recurrencia:

$$T(n) \quad = \quad \begin{cases} 1 & n = 0 \\ 3 \cdot T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + c \cdot n & n \geq 1 \end{cases}$$

Dado que $\log_2(3) > 1.5$, tenemos que $\log_2(3) - 0.5 > 1$

Deducimos que $c \cdot n \in \mathcal{O}\left(n^{\log_2(3) - 0.5}\right)$, por lo que usando el Teorema Maestro concluimos que $T(n) \in \Theta\left(n^{\log_2(3)}\right)$

El Teorema Maestro y la función $\lceil x \rceil$

Suponga que cambiamos $\left\lfloor \frac{n}{b} \right\rfloor$ por $\left\lceil \frac{n}{b} \right\rceil$ en la definición de (a, b)-regularidad.

El Teorema Maestro sigue siendo válido pero con $T\left(\left\lfloor \frac{n}{b} \right\rfloor\right) + f(n)$ reemplazado por $T\left(\left\lceil \frac{n}{b} \right\rceil\right) + f(n)$.

Ahora considere la siguiente ecuación:

$$T(n) = \begin{cases} 1 & n = 0 \\ T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + c \cdot n & n \geq 1 \end{cases}$$

¿Se puede utilizar el Teorema Maestro en la ecuación anterior?

Analizando la complejidad de un algoritmo

Sea $\mathcal{A} : \Sigma^* \to \Sigma^*$ un algoritmo.

Definición

Decimos que $\mathcal{A}$ **en el peor caso es** $\mathcal{O}(f(n))$ si

$$t_{\mathcal{A}}(n) \in \mathcal{O}(f(n))$$

Recuerde que $t_{\mathcal{A}}(n)$ es el mayor número de pasos realizados por $\mathcal{A}$ sobre las entradas $w \in \Sigma^*$ de largo n

Analizando la complejidad de un algoritmo

Notación

Las definición de peor caso puede ser modificada para considerar las notaciones Θ y Ω

- Simplemente reemplazando $\mathcal{O}(f(n))$ por $\Theta(f(n))$ u $\Omega(f(n))$, respectivamente

Por ejemplo, decimos que **$\mathcal{A}$ en peor caso es $\Omega(f(n))$** si

$$t_{\mathcal{A}}(n) \in \Omega(f(n)).$$