Técnicas Fundamentales

Teorema maestro y Dividir para conquistar

Segundo semestre 2022

IIC2283

Prof. Nicolás Van Sint Jan

Recordatorio: 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 $\left\lceil \frac{n}{b} \right\rceil$.

Recordatorio: 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.

Outline

Teorema maestro

Dividir para conquistar

Outline

Teorema maestro

Dividir para conquistar

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:

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

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:

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

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

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

Outline

Teorema maestro

Dividir para conquistar

Dividir para conquistar

Esta es la forma genérica de una algoritmo que utiliza la técnica de **dividir para conquistar**:

Algoritmo

DividirParaConquistar(w)
  if $|w| \leq k$ **then return InstanciasPequeñas**(w)
  else
    Dividir w en w_1, ..., w_ℓ
    for $i := 1$ **to** ℓ **do**
      $S_i :=$ **DividirParaConquistar**(w_i)
    return Combinar(S_1, ..., S_ℓ)

¿Cuál es la complejidad de un algoritmo de **dividir para conquistar**?

Dividir para conquistar

Considerar lo siguiente respecto al pseudo-código anterior:

- En **DividirParaConquistar** k es un constante que indica cuando el tamaño de una entrada w es considerado pequeño: $|w| \leq k$

- Las entradas pequeñas son solucionadas utilizado un algoritmo diseñado para ellas: **InstanciasPequeñas**
  - En general este algoritmo es sencillo y ejecuta un número pequeño de operaciones.

- Si el tamaño de una entrada w no es pequeño ($|w| > k$), entonces w es dividido es una secuencia $w_1, \ldots, w_\ell$ de entradas de menor tamaño para **DividirParaConquistar**

Dividir para conquistar

- Para cada $i \in \{1, \ldots, \ell\}$ la llamada **DividirParaConquistar**(w_i) resuelve el problema para la entrada w_i
  - El resultado de esta llamada es almacenado en S_i

- Finalmente la llamada **Combinar**(S_1, ..., S_ℓ) combina los resultados S_1, ..., S_ℓ para obtener el resultados para w.

Vimos un ejemplo de esta técnica en el algoritmo de **búsqueda binaria**.

Vamos a ver como utilizar esta técnica para obtener un algoritmo eficiente para la **multiplicación de dos números enteros**.

Antes de multiplicar: suma de números enteros

Sean $a, b \in \mathbb{Z}$ con $n \geq 1$ dígitos cada uno. Sea

$$c = a + b.$$

Considere **el algoritmo usual de la suma** para calcular c.

Consideramos la **suma de dos dígitos**, **comparación de dos dígitos** y **resta de un número con a lo más dos dígitos con uno de un dígito** como las operaciones a contar, cada una con costo 1.

Preguntas

1. ¿Tiene sentido suponer que todas tienen el mismo costo?
2. ¿Cuál es el peor caso para el algoritmo usual?
3. ¿Cuántas operaciones realiza el algoritmo en el peor caso?
4. ¿Cuántos dígitos puede tener c?

¿Se puede **sumar** más rápido que $\mathcal{O}(n)$?

Multiplicación de números enteros

Sean $a, b \in \mathbb{Z}$ con $n \geq 1$ dígitos cada uno. Sea

$$d = a \cdot b$$

Considere **el algoritmo usual de la multiplicación** para calcular d.

Esta vez tome **la suma** y **la multiplicación de dígitos** como las operaciones a contar, ambas con costo 1.

Preguntas

1. ¿Tiene sentido suponer que ambas operaciones tienen el mismo costo?
2. ¿Cuál es el peor caso para el algoritmo usual?
3. ¿Cuántas operaciones realiza el algoritmo en este caso?
4. ¿Cuántos dígitos puede tener d?

¿Se puede **multiplicar** más rápido que $\mathcal{O}(n^2)$?