Análisis de la eficiencia de un algoritmo

IIC2283

Una noción general de algoritmo

Sea Σ un alfabeto.

Vamos a pensar en un algoritmo $\mathcal{A}$ como una función $\mathcal{A} : \Sigma^* \to \Sigma^*$

Una noción general de algoritmo

Sea Σ un alfabeto.

Vamos a pensar en un algoritmo $\mathcal{A}$ como una función $\mathcal{A} : \Sigma^* \to \Sigma^*$

Esta es una representación general que incluye tanto a problemas de decisión como a los de computación.

- ¿Cómo se representa un problema de decisión?

Tiempo de ejecución de un algoritmo

A cada algoritmo $\mathcal{A}$ asociamos una función $\text{tiempo}_{\mathcal{A}} : \Sigma^* \to \mathbb{N}$ tal que:

$\text{tiempo}_{\mathcal{A}}(w)$: número de pasos realizados por $\mathcal{A}$ con entrada $w \in \Sigma^*$

Para definir esta función tenemos que definir qué operaciones vamos a contar, y qué costo les asignamos.

Tiempo de ejecución de un algoritmo en el peor caso

El primer tipo de análisis que vamos a realizar de la complejidad de un algoritmo va a estar basado en el peor caso.

Tiempo de ejecución de un algoritmo en el peor caso

El primer tipo de análisis que vamos a realizar de la complejidad de un algoritmo va a estar basado en el peor caso.

Para cada algoritmo $\mathcal{A}$ asociamos una función $t_{\mathcal{A}} : \mathbb{N} \to \mathbb{N}$ tal que

$$t_{\mathcal{A}}(n) \quad = \quad \text{máx}\{\text{tiempo}_{\mathcal{A}}(w) \mid w \in \Sigma^* \text{ y } |w| = n\},$$

donde $|w|$ es el largo de w

Notación asintótica

En muchos casos, nos interesa conocer el *orden* de un algoritmo en lugar de su complejidad exacta.

- Queremos decir que un algoritmo es lineal o cuadrático, en lugar de decir que su complejidad es $3n^2 + 17n + 22$

Vamos a desarrollar notación para hablar del orden de un algoritmo.

Notación asintótica

Supuesto

La complejidad de un algoritmo va a ser medida en términos de funciones de la forma $f : \mathbb{N} \to \mathbb{R}_0^+$, donde $\mathbb{R}^+ = \{r \in \mathbb{R} \mid r > 0\}$ y $\mathbb{R}_0^+ = \mathbb{R}^+ \cup \{0\}$

Notación asintótica

Supuesto

La complejidad de un algoritmo va a ser medida en términos de funciones de la forma $f : \mathbb{N} \to \mathbb{R}_0^+$, donde $\mathbb{R}^+ = \{r \in \mathbb{R} \mid r > 0\}$ y $\mathbb{R}_0^+ = \mathbb{R}^+ \cup \{0\}$

Estas funciones incluyen a las funciones definidas en las transparencias anteriores, y también sirven para modelar el tiempo de ejecución de un algoritmo

La notación $O(f)$

Sea $f : \mathbb{N} \to \mathbb{R}_0^+$

Definición

$$O(f) = \{g : \mathbb{N} \to \mathbb{R}_0^+ \mid (\exists c \in \mathbb{R}^+)(\exists n_0 \in \mathbb{N}) \\ (\forall n \geq n_0)\,(g(n) \leq c \cdot f(n))\}$$

La notación $O(f)$

Sea $f : \mathbb{N} \to \mathbb{R}_0^+$

Definición

$$O(f) = \{g : \mathbb{N} \to \mathbb{R}_0^+ \mid (\exists c \in \mathbb{R}^+)(\exists n_0 \in \mathbb{N}) \\ (\forall n \geq n_0)\,(g(n) \leq c \cdot f(n))\}$$

Decimos entonces que $g \in O(f)$

- También usamos la notación g es $O(f)$, lo cual es formalizado como $g \in O(f)$

La notación $O(f)$

Sea $f : \mathbb{N} \to \mathbb{R}_0^+$

Definición

$$O(f) = \{g : \mathbb{N} \to \mathbb{R}_0^+ \mid (\exists c \in \mathbb{R}^+)(\exists n_0 \in \mathbb{N}) \\ (\forall n \geq n_0)\,(g(n) \leq c \cdot f(n))\}$$

Decimos entonces que $g \in O(f)$

- También usamos la notación g es $O(f)$, lo cual es formalizado como $g \in O(f)$

Ejercicio

Demuestre que $3n^2 + 17n + 22 \in O(n^2)$

Las notaciones $\Omega(f)$ y $\Theta(f)$

Definición

$$\Omega(f) = \{g : \mathbb{N} \to \mathbb{R}_0^+ \mid (\exists c \in \mathbb{R}^+)(\exists n_0 \in \mathbb{N}) \\ (\forall n \geq n_0)\,(c \cdot f(n) \leq g(n))\}$$

$$\Theta(f) = O(f) \cap \Omega(f)$$

Las notaciones $\Omega(f)$ y $\Theta(f)$

Definición

$$\Omega(f) = \{g : \mathbb{N} \to \mathbb{R}_0^+ \mid (\exists c \in \mathbb{R}^+)(\exists n_0 \in \mathbb{N}) \\ (\forall n \geq n_0)\,(c \cdot f(n) \leq g(n))\}$$

$$\Theta(f) = O(f) \cap \Omega(f)$$

Ejercicios

1. Demuestre que $3n^2 + 17n + 22 \in \Theta(n^2)$
2. Demuestre que $g \in \Theta(f)$ si y sólo si existen $c, d \in \mathbb{R}^+$ y $n_0 \in \mathbb{N}$ tal que para todo $n \geq n_0$: $c \cdot f(n) \leq g(n) \leq d \cdot f(n)$

Ejercicios

1. Sea $p(n)$ un polinomio de grado $k \geq 0$ con coeficientes en los números enteros. Demuestre que $p(n) \in O(n^k)$.

2. ¿Cuáles de las siguientes afirmaciones son ciertas?
   - $n^2 \in O(n)$
   - Si $f(n) \in O(n)$, entonces $f(n)^2 \in O(n^2)$
   - Si $f(n) \in O(n)$, entonces $2^{f(n)} \in O(2^n)$

3. Suponga que $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ existe y es igual a ℓ. Demuestre lo siguiente:
   - Si $\ell = 0$, entonces $f \in O(g)$ y $g \notin O(f)$
   - Si $\ell = \infty$, entonces $g \in O(f)$ y $f \notin O(g)$
   - Si $\ell \in \mathbb{R}^+$, entonces $f \in \Theta(g)$

4. Encuentre funciones f y g tales que $f \in \Theta(g)$ y $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ no existe

Búsqueda binaria

Suponga que tiene una lista ordenada (de menor a mayor) $L[1 \ldots n]$ de números enteros con $n \geq 1$

¿Cómo podemos verificar si un número a está en L?

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)

Tiempo de ejecución de búsqueda binaria

¿Cuál es la complejidad del algoritmo?

- ¿Qué operaciones vamos a considerar?
- ¿Cuál es el peor caso?

Tiempo de ejecución de búsqueda binaria

¿Cuál es la complejidad del algoritmo?

- ¿Qué operaciones vamos a considerar?
- ¿Cuál es el peor caso?

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

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

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

Tiempo de ejecución de búsqueda binaria

¿Cuál es la complejidad del algoritmo?

- ¿Qué operaciones vamos a considerar?
- ¿Cuál es el peor caso?

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

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

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

Esta es una ecuación de recurrencia

Solucionando una ecuación de recurrencia

¿Cómo podemos solucionar una ecuación de recurrencia?

- Técnica básica: sustitución de variables

Solucionando una ecuación de recurrencia

¿Cómo podemos solucionar una ecuación de recurrencia?

- Técnica básica: sustitución de variables

Para la ecuación anterior usamos la sustitución $n = 2^k$

- Vamos a resolver la ecuación suponiendo que n es una potencia de 2
- Vamos a utilizar inducción para demostrar que la solución obtenida nos da el orden del algoritmo

Ecuaciones de recurrencia: sustitución de variables

Si realizamos la sustitución $n = 2^k$ en la ecuación:

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

obtenemos:

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

Ecuaciones de recurrencia: sustitución de variables

Extendiendo la expresión anterior obtenemos:

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

Ecuaciones de recurrencia: sustitución de variables

Extendiendo la expresión anterior obtenemos:

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

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

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

Ecuaciones de recurrencia: sustitución de variables

Considerando $i = k$ obtenemos:

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

Ecuaciones de recurrencia: 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

Ecuaciones de recurrencia: 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 vamos a extender esta solución y vamos a demostrar que $T(n) \in O(\log_2(n))$

Inducción constructiva

Sea $T(n)$ definida como:

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

Queremos demostrar que $T(n) \in O(\log_2(n))$

- Vale decir, queremos demostrar que existen $e \in \mathbb{R}^+$ y $n_0 \in \mathbb{N}$ tales que $T(n) \leq e \cdot \log_2(n)$ para todo $n \geq n_0$

Inducción constructiva

Sea $T(n)$ definida como:

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

Queremos demostrar que $T(n) \in O(\log_2(n))$

- Vale decir, queremos demostrar que existen $e \in \mathbb{R}^+$ y $n_0 \in \mathbb{N}$ tales que $T(n) \leq e \cdot \log_2(n)$ para todo $n \geq n_0$

Inducción nos va servir tanto para demostrar la propiedad y como para determinar valores adecuados para e y n_0

- Por esto usamos el término inducción constructiva

Inducción constructiva

Dado que $T(1) = c$ y $\log_2(1) = 0$ no es posible encontrar una valor para e tal que $T(1) \leq e \cdot \log_2(1)$

Dado que $T(2) = (c + d)$, si consideramos $e = (c + d)$ tenemos que $T(2) \leq e \cdot \log_2(2)$

- Definimos entonces $e = (c + d)$ y $n_0 = 2$

Tenemos entonces que demostrar lo siguiente:

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

Inducción constructiva y fuerte

¿Cuál es el principio de inducción adecuado para el problema anterior?

- Tenemos n_0 como punto de partida
- n_0 es un caso base, pero podemos tener otros
- Dado $n > n_0$ tal que n no es un caso base, suponemos que la propiedad se cumple para todo $k \in \{n_0, \ldots, n-1\}$

Inducción constructiva y fuerte

Queremos demostrar que $(\forall n \geq 2)\,(T(n) \leq e \cdot \log_2(n))$

Inducción constructiva y fuerte

Queremos demostrar que $(\forall n \geq 2)\,(T(n) \leq e \cdot \log_2(n))$

- 2 es el punto de partida y el primer caso base

Inducción constructiva y fuerte

Queremos demostrar que $(\forall n \geq 2)\,(T(n) \leq e \cdot \log_2(n))$

- 2 es el punto de partida y el primer caso base
- También 3 es un caso base ya que $T(3) = T(1) + d$ y para $T(1)$ no se cumple la propiedad

Inducción constructiva y fuerte

Queremos demostrar que $(\forall n \geq 2)\,(T(n) \leq e \cdot \log_2(n))$

- 2 es el punto de partida y el primer caso base
- También 3 es un caso base ya que $T(3) = T(1) + d$ y para $T(1)$ no se cumple la propiedad
- Para $n \geq 4$ tenemos que $T(n) = T(\lfloor \frac{n}{2} \rfloor) + d$ y $\lfloor \frac{n}{2} \rfloor \geq 2$, por lo que resolvemos este caso de manera inductiva
  - Suponemos que la propiedad se cumple para todo $k \in \{2, \ldots, n-1\}$

La demostración por inducción fuerte

Casos base:

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

La demostración por inducción fuerte

Casos base:

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

Caso inductivo:

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

La demostración por inducción fuerte

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

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

Un segundo ejemplo de inducción constructiva

Considere la siguiente ecuación de recurrencia:

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

Un segundo ejemplo de inducción constructiva

Considere la siguiente ecuación de recurrencia:

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

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

Un segundo ejemplo de inducción constructiva

Considere la siguiente ecuación de recurrencia:

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

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

- ¿Alguna conjetura sobre quién 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 $n \geq n_0$

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:

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

Una posible solución para la ecuación de recurrencia

Supongamos que la propiedad se cumple para n:

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

Tenemos que:

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

Una posible solución para la ecuación de recurrencia

Supongamos que la propiedad se cumple para n:

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

Tenemos que:

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

Pero 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?

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

¿Cómo solucionamos el problema con la demostración?

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.

¿Cómo solucionamos el problema con la demostración?

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:

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

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.

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:

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

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:

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

Inducción constructiva sobre una propiedad más fuerte

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

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

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:

$$\frac{(n+1)}{(n-1)} \quad \leq \quad d$$

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:

$$\frac{(n+1)}{(n-1)} \quad \leq \quad d$$

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 $n_0 = 2$

Inducción constructiva sobre una propiedad más fuerte

Para concluir la demostración debemos considerar el caso base $n_0 = 2$

Tenemos que:

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

Inducción constructiva sobre una propiedad más fuerte

Para concluir la demostración debemos considerar el caso base $n_0 = 2$

Tenemos que:

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

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

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

Inducción constructiva sobre una propiedad más fuerte

Para concluir la demostración debemos considerar el caso base $n_0 = 2$

Tenemos que:

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

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

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

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 $T(n) \in O(n!)$

El Teorema Maestro

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

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

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

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

El Teorema Maestro

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

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

Una condición de regularidad sobre funciones

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

Definición

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

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

Una condición de regularidad sobre funciones

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

Definición

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

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

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

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

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

Una solución al segundo problema

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

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

De esto concluimos que:

$$(\forall k \geq n_0)(\log_2(\lfloor \frac{2 \cdot k}{2} \rfloor) \leq c \cdot \log_2(2 \cdot k))$$

Una solución al segundo problema

Vale decir:

$$(\forall k \geq n_0)(\log_2(k) \leq c \cdot (\log_2(k) + 1))$$

Una solución al segundo problema

Vale decir:

$$(\forall k \geq n_0)(\log_2(k) \leq c \cdot (\log_2(k) + 1))$$

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

$$(\forall k \geq n_0)(\log_2(k) \leq \frac{c}{1-c})$$

Lo cual nos lleva a una contradicción. □

El enunciado del Teorema Maestro

Teorema

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

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

Se tiene que:

El enunciado del Teorema Maestro

Teorema

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

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

Se tiene que:

1. *Si* $f(n) \in O(n^{\log_b(a)-\varepsilon})$ *para* $\varepsilon > 0$, *entonces* $T(n) \in \Theta(n^{\log_b(a)})$

El enunciado del Teorema Maestro

Teorema

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

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

Se tiene que:

1. *Si* $f(n) \in O(n^{\log_b(a)-\varepsilon})$ *para* $\varepsilon > 0$, *entonces* $T(n) \in \Theta(n^{\log_b(a)})$
2. *Si* $f(n) \in \Theta(n^{\log_b(a)})$, *entonces* $T(n) \in \Theta(n^{\log_b(a)} \cdot \log_2(n))$

El enunciado del Teorema Maestro

Teorema

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

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

Se tiene que:

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

Usando el Teorema Maestro

Considere la siguiente ecuación de recurrencia:

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

Usando el Teorema Maestro

Considere la siguiente ecuación de recurrencia:

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

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

Usando el Teorema Maestro

Considere la siguiente ecuación de recurrencia:

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

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

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

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

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

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

Analizando la complejidad de un algoritmo

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

- 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

Definición (Complejidad en el peor caso)

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

Analizando la complejidad de un algoritmo

Notación

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

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

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