Análisis de la eficiencia de un algoritmo

IIC2283 – Diseño y Análisis de Algoritmos

Diego Arroyuelo, Juan P. Castillo
diego.arroyuelo@uc.cl

Departamento de Ciencia de la Computación
Pontificia Universidad Católica de Chile

2025-2

Ecuaciones de recurrencia

Aparecen naturalmente al analizar algoritmos recursivos, por ejemplo:

Algoritmo 1: $\textsc{Fact}(n)$

if $n = 0$ **then**
    return 1
else
    return $n \times \textsc{Fact}(n-1)$
end

El tiempo de ejecución de $\textsc{Fact}(n)$ puede expresarse como la ecuación de recurrencia:

$$T(n) = \begin{cases} T(n-1) + b, & n \geq 1; \\ a, & n = 0. \end{cases}$$

para constantes $a \in \mathbb{N}$, $b \in \mathbb{N}$, ambas > 0, que representan el tiempo de ejecución de una multiplicación y la instrucción **return**

Ecuaciones de recurrencia

Las ecuaciones de recurrencia son una forma válida de representar una función matemática

Indican cómo debe computarse la función correspondiente pero no dan pistas respecto a la función que están denotando, lo cual sería útil para poder analizarlas

Por ejemplo, la ecuación de recurrencia

$$T(n) = \begin{cases} 2T(n-1)+1, & n \geq 1; \\ 0, & n = 0. \end{cases}$$

es equivalente a la función $T(n) = 2^n - 1$, para $n \geq 0$.

Los algoritmos recursivos serán importantes en el curso, por lo que estudiaremos formas de encontrar la forma cerrada de ecuaciones de recurrencia

Ecuaciones de recurrencia

Analicemos ahora otro algoritmo recursivo típico:

Algoritmo 2: $\textsc{Fib}(n)$

if $n \leq 1$ **then**
    return n
else
    return $\textsc{Fib}(n-1) + \textsc{Fib}(n-2)$
end

El valor $\textsc{Fib}(n)$ puede expresarse con la ecuación de recurrencia:

$$T(n) = \begin{cases} T(n-1) + T(n-2), & n \geq 2; \\ 0, & n = 0; \\ 1, & n = 1. \end{cases}$$

Note que no estamos analizando el tiempo de ejecución en este caso, sino el valor de $\textsc{Fib}(\textsc{n})$

Ecuaciones de recurrencia

A pesar de que $T(n)$ puede definirse de forma simple y directa desde el algoritmo, no sabemos mucho respecto de la función que representa

Nos gustaría encontrar la forma cerrada de T para poder estudiarla (e.g., entender su velocidad de crecimiento)

Para el caso $\text{Fib}(n)$, ¿Cuál es la función que acota a $T(n)$?

A continuación demostraremos que $T(n) \in O(2^n)$

Eso significa que hay que demostrar que $\exists c \in \mathbb{R}^+$, $\exists n_0 \in \mathbb{N}$, tal que

$$T(n) \leq c \cdot 2^n, \ \forall n \geq n_0.$$

Vamos a demostrarlo por inducción

Ecuaciones de recurrencia

Casos base: Primero chequeamos que la propiedad se cumpla para los casos base

- Para $n = 0$, $T(0) = 0$, entonces se debe cumplir $0 \leq c \cdot 2^0$, lo cual es cierto para $c \geq 0$.
- Para $n = 1$, $T(1) = 1$, entonces se debe cumplir $1 \leq c \cdot 2^1$, lo cual es cierto para $c \geq 1/2$.

Esto significa que la propiedad es cierta para los casos base

Ecuaciones de recurrencia

Hipótesis inductiva: Asumimos $T(n') \leq c \cdot 2^{n'}, \ \forall n' \in \{0, 1, \ldots, n-1\}$

Estamos usando inducción fuerte, por lo que es necesario chequear que la propiedad se cumple para todos los casos base

Paso inductivo: Sea $n \geq 2$. Entonces $T(n)$ corresponde al caso recurrente:

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

Esto prueba la propiedad para todo $n \geq 0$.

Búsqueda binaria

Suponga que tiene una lista ordenada (de menor a mayor) $L[1 \dots n]$ 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?

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

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

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

Solucionando una ecuación de recurrencia

¿Cómo podemos solucionar una ecuación de recurrencia?

- Técnica básica: sustitución de variables

Para la ecuación anterior usamos la sustitución $n = 2^k$, por lo que $k = \log_2(n)$

- 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:

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

obtenemos:

$$T(2^k) = \begin{cases} b, & k = 0 \quad [2^k = 1] \\ T(2^{k-1}) + d, & k > 0 \quad [2^k > 1] \end{cases}$$

Ecuaciones de recurrencia: sustitución de variables

Extendiendo la expresión anterior obtenemos:

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

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:

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

Dado que $k = \log_2(n)$, obtenemos que $T(n) = b + d \cdot \log_2(n)$ 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:

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

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

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

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

- Por esto usamos el término inducción constructiva

Inducción constructiva

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

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

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

Tenemos entonces que demostrar lo siguiente:

$$\forall n \geq 2, \;\; T(n) \leq c \cdot \log_2(n)$$

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 $n' \in \{n_0, \ldots, n-1\}$

Inducción constructiva y fuerte

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

- 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 $n' \in \{2, \ldots, n-1\}$

La demostración por inducción fuerte

Casos base:

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

Caso inductivo:

Suponemos que $n \geq 4$ y para todo $n' \in \{2, \ldots, n-1\}$ se tiene que $T(n') \leq c \cdot \log_2(n')$

La demostración por inducción fuerte

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

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

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 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 $n \geq n_0$

- 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:

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

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

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.

Supongamos que la propiedad se cumple para n:

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

Tenemos que:

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

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 $n_0 = 2$

Tenemos que:

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

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:

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

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

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

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

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

De esto concluimos que:

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

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:

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

Lo cual nos lleva a una contradicción. $\square$

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:

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

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 $T(\lceil \frac{n}{b} \rceil) + f(n)$

El enunciado del Teorema Maestro

La siguiente es una versión más fuerte 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* $f(n) \in \Theta(n^{\log_b(a)} \cdot \log_2^k(n))$ *para* $k \geq 0$, *entonces* $T(n) \in \Theta(n^{\log_b(a)} \cdot \log_2^{k+1}(n))$
3. *Si* $f(n) \in \Omega(n^{\log_b(a)+\varepsilon})$ *para* $\varepsilon > 0$ *y* f *es* (a, b)*-regular, entonces* $T(n) \in \Theta(f(n))$