Pontificia Universidad Católica de Chile
Facultad de Ingeniería
Departamento de Ciencias de la computación
Curso: Matemáticas discretas
Ayudantes: Francisca Caprile, Catalina Ortega, Matías Fernández e Ignacio Vergara

Ayudantía 10

3 de noviembre de 2023

2º semestre 2020 - Profesores G. Diéguez - S. Bugedo - N. Alvarado- B. Barías

---

Resumen

- **Algoritmo** Un algoritmo es un método para convertir un input válido en un output. Para efectos de este curso, se utilizará pseudocódigo para escribir algoritmos.

  → **Precondiciones:** representan el input del programa.

  → **Postcondiciones:** representan el output del programa, lo que hace el algoritmo con el input.

- **Correctitud de algoritmos** Un algoritmo es correcto si para todo input válido, el algoritmo se detiene y produce un output correcto. Por ende, hay que demostrar dos propiedades.

  → **Corrección parcial:** si el algoritmo se detiene, se cumplen las postcondiciones.

  → **Terminación:** el algoritmo se detiene

  Existen distintos tipos de algoritmos, con distintas técnicas de demostración de correctitud.

- **Correctitud de algoritmos iterativos:** Primero se encuentra un invariante (una propiedad verdadera en cada paso de la iteración), y luego se debe demostrar la corrección del loop inductivamente:

  - Base: las precondiciones deben implicar que $I(0)$ es verdadero.
  - Inducción: para todo natural $k > 0$, si G e $I(k)$ son verdaderos antes de la iteración, entonces $I(k+1)$ es verdadero después de la iteración.
  - Corrección: inmediatamente después de terminado el loop (es decir, cuando G es falso), si $k = N$ e $I(N)$ es verdadero, entonces las postcondiciones se cumplen.

  Y para demostrar terminación, debemos mostrar que existe un k para el cual G es falso.

- **Correctitud de algoritmos recursivos:** En el caso de los algoritmos recursivos, no necesitamos dividir la demostración en corrección parcial y terminación.
  Basta demostrar por inducción la propiedad (corrección) deseada.
  En general, la inducción se realiza sobre el tamaño del input.

- **Notación asintótica**
  Sea $f : \mathbb{N} \to \mathbb{R}^+$.

  - **La notación $O(f)$ se define como:**

$$O(f) = \left\{ g : \mathbb{N} \to \mathbb{R}^+ \;\middle|\; \exists c \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, g(n) \leq c \cdot f(n) \right\}$$

Diremos que $g \in O(f)$ significa que g es a lo sumo de orden f o que pertenece a $O(f)$.

- **La notación $\Omega(f)$ se define como:**

$$\Omega(f) = \left\{ g : \mathbb{N} \to \mathbb{R}^+ \;\middle|\; \exists c \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, g(n) \geq c \cdot f(n) \right\}$$

Diremos que $g \in \Omega(f)$ significa que g es al menos de orden f o que pertenece a $\Omega(f)$.

- **La notación $\Theta(f)$ se define como:**

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

Diremos que $g \in \Theta(f)$ significa que g es exactamente de orden f o que pertenece a $\Theta(f)$.

- **Complejidad de algoritmos**
Para un algoritmo dado, se puede estimar el tiempo para el peor caso (cuando el input hace que el algoritmo se demore la mayor cantidad de tiempo posible) y el mejor caso (lo contrario) para un tamaño de input n.
Se encuentra la complejidad del algoritmo.
Para algunos casos, basta con sumar o multiplicar las complejidades de secciones de código.
Existen otras técnicas para casos más complejos.

→ **Ecuación de recurrencia y teorema maestro:**

Si $a_1, a_2, b, c, c_0, d \in \mathbb{R}^+$ y $b > 1$, entonces para una recurrencia de la forma:

$$T(n) = \begin{cases} c_0 & \text{si } 0 \leq n < n_0 \\ a_1 \cdot T(\lceil \frac{n}{b} \rceil) + a_2 \cdot T(\lfloor \frac{n}{b} \rfloor) + c \cdot n^d & \text{en otro caso} \end{cases}$$

Se cumple que $T(n) \in$:

$$\begin{cases} \Theta(n^d) & \text{si } a_1 + a_2 < b^d \\ \Theta(n^d \cdot \log(n)) & \text{si } a_1 + a_2 = b^d \\ \Theta(n^{\log_b(a_1+a_2)}) & \text{si } a_1 + a_2 > b^d \end{cases}$$

Ejercicio 1 | Correctitud

a) Escriba un algoritmo iterativo que resuelva el problema del Mínimo Común Múltiplo. Su algoritmo debe recibir como input dos números y devolver como output el número natural que corresponda al mínimo común múltiplo del input.

b) Demuestre que su algoritmo es correcto.

Ejercicio 2 | Notación asintótica

Dado $f(n) = \sqrt{n}$ y $g(n) = n^{sin(n)}$

Decida si (1) $f \in \Theta(g)$, (2) $f \in O(g)$, (3) $f \in \Omega(g)$ o (4) ninguna de las anteriores. Demuestre su afirmación usando la definición formal de la notación Θ, O, o Ω.

Ejercicio 3 | Complejidad y ecuaciones de recursividad

Considere el siguiente algoritmo,

Algorithm 1: PowerAlgorithm

Data: x, n
Result: x^n

```
if n = 1 then
    return x;
n_half ← ⌊n/2⌋;
Powhalf ← PowerAlgorithm(x, n_half);
Finalpow ← Powhalf * Powhalf;
if n mód 2 = 1 then
    return Finalpow * x;
else
    return Finalpow;
```

Determine su ecuación de recurrencia y complejidad.

Ejercicio 4 | Notación asintótica

Demuestre según la definicion de Notacion O que

$$log(a_k \cdot n^k + a_{k-1}n^{k-1} + ... + a_1 \cdot n + a_0) \in O(log(n))$$

Con $k \geq 0$ y $a_i \geq 0$ para todo $i \leq k$.