Algoritmos y notación $\mathcal{O}$

Clase 17

IIC 1253

Prof. Cristian Riveros

¿qué es un algoritmo?

Definición

Un **algoritmo** es una secuencia finita de instrucciones precisas para realizar una computación o resolver un problema.

¿qué es un algoritmo?

¿és esto un algoritmo?

Sea $a_1, \ldots, a_n$ una secuencia de números.

1. Asigne el máximo temporal de la secuencia como a_1.
2. Compare el siguiente elemento con el máximo temporal y, si es mayor que el máximo temporal, replace el máximo temporal por este elemento.
3. Repita el paso anterior si hay más elementos en la secuencia.
4. Para cuando no queden mas elementos y el máximo temporal es el máximo elemento de la secuencia.

¿qué es un algoritmo?

¿és esto un algoritmo?

input : Una secuencia $S = (a_1, \ldots, a_n)$ y $n \geq 1$.
output: El máximo de la secuencia S.

Function `Max` (S, n)
  $m := a_1$
  $k := 2$
  while $k \leq n$ **do**
    if $a_k > m$ **then**
      $m := a_k$
    $k := k + 1$
  return m

¿cuál es el "lenguaje de programación" de estos algoritmos?

¿qué es un algoritmo?

Definición

Un **algoritmo** es una secuencia finita de instrucciones precisas para realizar una computación o resolver un problema.

Un algoritmo puede estar dado por cualquier lenguaje:

- Lenguaje de programación.
  - Python, Java, C++, etc
- Lenguaje natural.
- Pseudo-código.

¿qué es **pseudo-código**?

¿qué es un algoritmo?

Algoritmo en pseudo-código

input : Dos número positivos n y m
output: Un número z

Function `M` (n, m)
  $z := n$
  while $m > 0$ **do**
    $z := z + n$
    $m := m - 1$
  return z

¿qué hace este algoritmo?

Propiedades de un algoritmo

1. **Input**: *"el input del algoritmo vendrá de un conjunto específico."*
2. **Output**: *"para cada input el algoritmo producirá un resultado de un conjunto específico."*
3. **Precisión**: *"los pasos del algoritmo deben estar definidos de manera precisa."*
4. **Correctitud**: *"el algoritmo debe producir el output correcto para cada input."*

Propiedades de un algoritmo

5. **Finitud**: *"el algoritmo debe producir el output en una cantidad finita de pasos."*

6. **Efectividad**: *"cada paso del algoritmo puede ser realizable y en una cantidad finita de tiempo."*

7. **Generalidad**: *"el algoritmo debe ser aplicable para todo contexto y no solo para un conjunto particular de inputs."*

¿cuál es la diferencia entre una **función** y un **algoritmo**?

¿qué nos gustaría analizar de un algoritmo?

1. **Finitud**.

   para cada input, ¿se detiene mi algoritmo
   en una cantidad finita de pasos?

2. **Correctitud**.

   ¿és mi algoritmo correcto?

3. **Eficiencia**.

   ¿cuál es la eficiencia de mi algoritmo?

Desde ahora supondremos la **finitud** y **correctitud**
de nuestros algoritmos y estudiaremos su **eficiencia**.

Outline

Outline

Eficiencia de algoritmos

Notación asintótica

¿eficiencia en terminos de qué?

1. Tiempo.
2. Espacio.
3. Comunicación.
4. Paralelización.
5. Lecturas a disco duro.
6. ...

Desde ahora en adelante nos preocuparemos solo del **tiempo**.

Eficiencia con respecto al tiempo

Definición

Para un **algoritmo** A sobre un conjunto de inputs $\mathcal{I}$ se define la función:

$$\text{tiempo}_A : \mathcal{I} \to \mathbb{N}$$

tal que para todo input $I \in \mathcal{I}$:

$$\text{tiempo}_A(I) \ = \ \text{número de } \textbf{pasos} \text{ realizados por } A \text{ con input } I$$

Eficiencia con respecto al tiempo

Ejemplo

Function `Max` *(S, n)*

$m := a_1$
$k := 2$
while $k \leq n$ **do**
  if $a_k > m$ **then**
    $m := a_k$
  $k := k + 1$
return m

- Para $S = 3, 5, 10, 30$ ¿cuánto es $\text{tiempo}_{\text{Max}}(S, 4)$?
- Para $S = 30, 10, 5, 3$ ¿cuánto es $\text{tiempo}_{\text{Max}}(S', 4)$?

Eficiencia con respecto al tiempo

Definición

Para un **algoritmo** A sobre un conjunto de inputs $\mathcal{I}$ se define la función:

$$\text{tiempo}_A : \mathcal{I} \to \mathbb{N}$$

tal que para todo input $I \in \mathcal{I}$:

$$\text{tiempo}_A(I) \;=\; \text{número de } \textbf{pasos} \text{ realizados por } A \text{ con input } I$$

¿cómo podemos comparar la eficiencia entre algoritmos?

Posible definición

Un algoritmo A es el "más eficiente" si para todo algoritmo B que calcula lo mismo que A se tiene que $\text{tiempo}_A(I) \leq \text{tiempo}_B(I)$ para todo $I \in \mathcal{I}$.

¿es esta definición correcta? ¿es "robusta"?

Eficiencia con respecto al tiempo

Function `Max` (S, n)
- $m := a_1$
- $k := 2$
- **while** $k \leq n$ **do**
  - **if** $a_k > m$ **then**
    - $m := a_k$
  - $k := k + 1$
- **return** m

Function `Max2` (S, n)
- $m := a_n$
- **for** $k = n - 1$ **to** 1 **do**
  - **if** $a_k > m$ **then**
    - $m := a_k$
- **return** m

Function `Max3` (S, n)
- **if** $n = 2 \ \wedge \ a_1 \leq a_2$ **then**
  - **return** a_2
- $m := a_1$
- $k := 2$
- **while** $k \leq n$ **do**
  - **if** $a_k > m$ **then**
    - $m := a_k$
  - $k := k + 1$
- **return** m

¿cuál de los algoritmos es más "eficiente"?

Outline

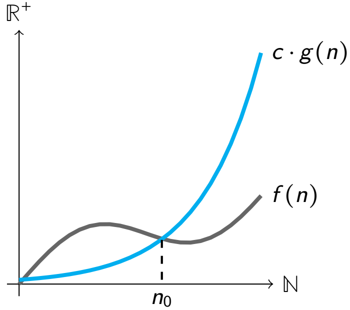
Notación $\mathcal{O}$

Sea $\quad f : \mathbb{N} \to \mathbb{R}^+ \quad$ y $\quad g : \mathbb{N} \to \mathbb{R}^+$.

Definición

Se define el conjunto $\mathcal{O}(g)$ de todas las funciones $f : \mathbb{N} \to \mathbb{R}^+$ tal que **existe** $c \in \mathbb{R}^+$ y $n_0 \in \mathbb{N}$ tal que:

$$f(n) \;\leq\; c \cdot g(n) \quad \textbf{para todo } n \geq n_0$$



Notación $\mathcal{O}$

Sea $f : \mathbb{N} \to \mathbb{R}^+$ y $g : \mathbb{N} \to \mathbb{R}^+$.

Definición

Se define el conjunto $\mathcal{O}(g)$ de todas las funciones $f : \mathbb{N} \to \mathbb{R}^+$ tal que **existe** $c \in \mathbb{R}^+$ y $n_0 \in \mathbb{N}$ tal que:

$$f(n) \;\leq\; c \cdot g(n) \quad \textbf{para todo } n \geq n_0$$

En notación lógica:

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

Pensar en $f \in \mathcal{O}(g)$ como decir que f "**crece más lento o igual**" que g.

Notación $\mathcal{O}$

Definición

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

Ejemplo

Considere las funciones $f(x) = x^3 + 2x + 1$ y $g(x) = x^3$.

$$¿f \in \mathcal{O}(g)?$$

Para $n \geq 1$ tenemos que:

$$n^3 + 2n + 1 \; \leq \; n^3 + 2n^3 + n^3 \; = \; 4n^3$$

Si tomamos $c = 4$ y $n_0 = 1$ entonces para todo $n \geq n_0$:

$$f(n) \; = \; n^3 + 2n + 1 \; \leq \; 4n^3 \; = \; c \cdot g(n)$$

Por lo tanto, $f \in \mathcal{O}(g)$.

Notación $\mathcal{O}$

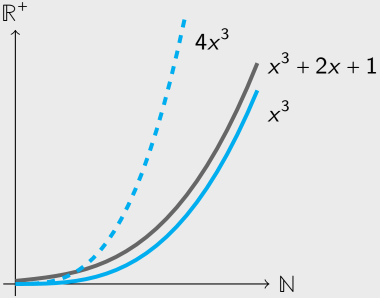
Definición

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

Ejemplo

Considere las funciones $f(x) = x^3 + 2x + 1$ y $g(x) = x^3$.

$$¿f \in \mathcal{O}(g)?$$



Notación $\mathcal{O}$

Sea $f : \mathbb{N} \to \mathbb{R}^+$ y $g : \mathbb{N} \to \mathbb{R}^+$.

Definición

Se define el conjunto $\mathcal{O}(g)$ de todas las funciones $f : \mathbb{N} \to \mathbb{R}^+$ tal que **existe** $c \in \mathbb{R}^+$ y $n_0 \in \mathbb{N}$ tal que:

$$f(n) \ \leq \ c \cdot g(n) \quad \textbf{para todo } n \geq n_0$$

Notación

Cuando $f \in \mathcal{O}(g)$ diremos alternativamente que:

- f es $\mathcal{O}(g)$.
- f es de **orden** g.
- $f = \mathcal{O}(g)$ (ojo, esto es solo notación!)

Algunas propiedades de la notación $\mathcal{O}$

Definición

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

Propiedades

1. Si $f(n) \leq g(n)$ para todo $n \in \mathbb{N}$, entonces $f \in \mathcal{O}(g)$.
2. Para todo $k \in \mathbb{N}$, si $f \in \mathcal{O}(g)$ entonces $k \cdot f \in \mathcal{O}(g)$.
3. Para todo g creciente y $k \in \mathbb{N}$, si $f \in \mathcal{O}(g)$ entonces $f + k \in \mathcal{O}(g)$.
4. Si $f \in \mathcal{O}(g)$ y $g \in \mathcal{O}(h)$, entonces $f \in \mathcal{O}(h)$.
5. Si $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$, entonces $f \in O(g)$.

Importante: esta última propiedad NO se puede usar en este curso.

Mas ejemplos de la notación $\mathcal{O}$

Definición

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

Ejemplo

Considere las funciones $f(x) = a_k x^k + \ldots + a_1 x + a_0$ y $g(x) = x^k$.

$$¿f \in \mathcal{O}(g)?$$

Para $n \geq 1$ tenemos que:

$$a_k n^k + \ldots + a_1 n + a_0 \leq a_k n^k + \ldots + a_1 n^k + a_0 n^k = \left(\sum_{i=0}^{k} a_i \right) \cdot n^k$$

Si tomamos $c = \sum_{i=0}^{k} a_i$ y $n_0 = 1$ entonces para todo $n \geq n_0$:

$$f(n) = a_k n^k + \ldots + a_1 n + a_0 \leq c \cdot n^k = c \cdot g(n)$$

Por lo tanto, $f \in \mathcal{O}(g)$.

Notación $\mathcal{O}$ para polinomios

Teorema

1. Sea $f(x) = a_k x^k + \ldots + a_1 x + a_0$ un polinomio sobre $\mathbb{N}$, entonces:

$$f \in \mathcal{O}(x^k)$$

2. $x^{k+1} \notin \mathcal{O}(x^k)$ para todo $k \in \mathbb{N}$.

Demostración 2.

Por contradicción, suponga que existe $c \in \mathbb{R}^+$ y $n_0 \in \mathbb{N}$ tal que:

$$n^{k+1} \leq c \cdot n^k \quad \text{para todo } n \geq n_0$$

Si consideramos $n \geq \max\{c+1, n_0\}$, entonces:

$$\begin{aligned} n^{k+1} &= n \cdot n^k \\ &\geq (c+1) \cdot n^k \\ &= c \cdot n^k + n^k > c \cdot n^k \quad \times \end{aligned}$$

Mas ejemplos de la notación $\mathcal{O}$

Definición

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

Ejemplo

Considere la función $f(n) = \log_a(n)$ y $g(n) = \log_b(n)$.

$$¿\log_a(n) \in \mathcal{O}(\log_b(n))?$$

Por propiedad de la función logaritmo sabemos:

$$\log_b(n) = \frac{\log_a(n)}{\log_a(b)}$$

Si consideramos $c = \log_a(b)$ y $n_0 = 1$, entonces:

$$\log_a(n) \leq c \cdot \log_b(n) \qquad \text{para todo } n \geq n_0$$

Por lo tanto, $\log_a(n) \in \mathcal{O}(\log_b(n))$.

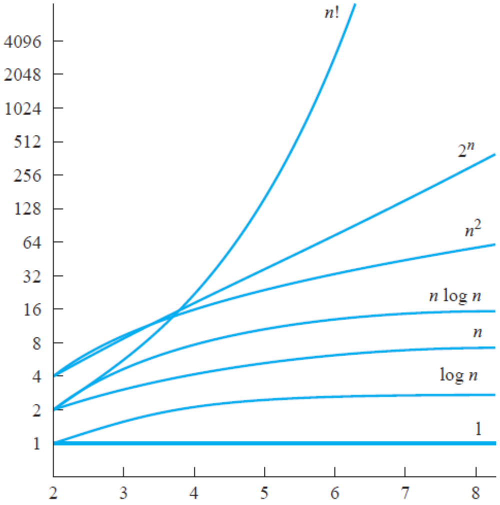
Logaritmos y exponenciales en notación $\mathcal{O}$

Teorema

1. Para todo $a, b > 1$, se tiene que $\log_a(n) \in \mathcal{O}(\log_b(n))$.
2. Para todo $a < b$ con $a, b \in \mathbb{N}$, se tiene que $a^n \in \mathcal{O}(b^n)$ y $b^n \notin \mathcal{O}(a^n)$.
3. Para todo $a \in \mathbb{N}$, se tiene que $a^n \in \mathcal{O}(n!)$ y $n! \notin \mathcal{O}(a^n)$.
4. $n! \in \mathcal{O}(2^{n \cdot \log(n)})$.

Demuestre 2., 3. y 4..

Jerarquía en notación $\mathcal{O}$



Jerarquía en notación $\mathcal{O}$

Notación	Nombre
$\mathcal{O}(1)$	Constante
$\mathcal{O}(\log n)$	Logarítmico
$\mathcal{O}(n)$	Lineal
$\mathcal{O}(n \log n)$	$n \log n$
$\mathcal{O}(n^2)$	Cuadrático
$\mathcal{O}(n^3)$	Cúbico
$\mathcal{O}(n^m)$	Polinomial
$\mathcal{O}(k^n)$	Exponencial
$\mathcal{O}(n!)$	Factorial