

Recursividad

Introducción

→ C++

Mathematica

→ $P(x), (x)$

Concepto

Factorial

$$n! = \begin{cases} 1 & n=0 \vee n=1 \\ 1 \cdot 2 \cdot \dots \cdot n & \text{en otro caso} \end{cases}$$

En Mathematica:

Factoriales[n_] := If[Or[n==0, n==1], Return[1],
 $n \cdot \text{Factoriales}[n-1]$]
 else

$(5!) = 120$

Factoriales[5] = $5 \cdot \text{Factoriales}[4] = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$

↳ Factoriales[4] = $4 \cdot \text{Factoriales}[3] = 4 \cdot 3 \cdot 2 \cdot 1$

↳ Factoriales[3] = $3 \cdot \text{Factoriales}[2] = 3 \cdot 2 \cdot 1$

↳ Factoriales[2] = $2 \cdot \text{Factoriales}[1] = 2 \cdot 1$

↳ Factoriales[1] = 1

Sucesión de Fibonacci

cuántas parejas de conejos obtendremos al final de un cierto año, si empezando con una pareja, cada pareja produce cadames una nueva que puede reproducirse al segundo mes de existencia?

$f(a_n) = a_{n-1} + a_{n-2}$

$a_1 = 1$
 $a_2 = 1$

$a_5 = a_4 + a_3$, $a_4 = a_3 + a_2$, $a_3 = a_2 + a_1 = 2$
 ↓ ↓ ↓
 3 2 3
 ↓
 5

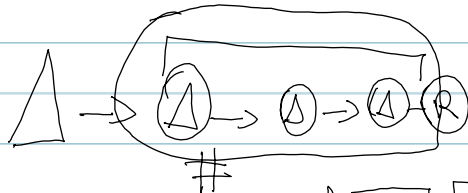
Mathematica y la ~~sucesión~~ ^{sucesión} de Fibonacci

\downarrow
Fibonacci [n]

$a_5 \rightarrow$ Fibonacci [5]
 $\rightarrow 5.$

Principios básicos de una recursividad

demonstración de una recursividad ✓
 \hookrightarrow inductiva ✓
 $\hookrightarrow \mathbb{N}$ ✓



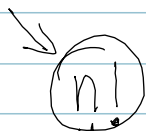
Por ejemplo: $\{100, 80, 10, 5, 1\} \rightarrow \mathbb{N}$
 $\{10, -2, -1\} \rightarrow \mathbb{Z}$
 $\{7/2, 5/2, 3/2, 1/2\} \rightarrow \mathbb{Q}$

↓
Un problema recursivo

dominio \rightarrow Programa

Propiedades de una recursividad

a) cíclica \rightarrow invoca #
definición



$$1! = 0! = 1$$

$$\checkmark 3! = 3 \cdot 2! = 3 \cdot 2 = 6$$

$$\checkmark 2! = 2 \cdot 1! = 2 \cdot 1 = 2$$

$$\checkmark a_n = a_{n-1} + a_{n-2} \checkmark$$

$$a_1 = a_2 = 1$$

$$a_4 = a_3 + a_2 = 2 + 1 = 3$$

$$a_3 = a_2 + a_1 = 2$$

1. Estructuras discretas

Algoritmos

Veamos el siguiente algoritmo de búsqueda de un dato "a".

$L = \{x_1, x_2, \dots, x_n\} \vdash a$

1. [Inicialización] sea v el último dato de la lista L , si $L = \emptyset$ "dato no encontrado", finalice.

2. [Compare] Tome a " v " y compare con " a ";

$V = a \rightarrow$ "dato encontrado" ✓
 $V \neq a \rightarrow$ continuar ,

3. [Continuar con la búsqueda] Tome a $(L) = L - \text{div}$ y llame a este algoritmo.

Principio de demostración por recursividad

→ Segundo → Inducción

Teorema

$P(x)$ es una proposición

$\forall x$, finaliza generando una solución si

1. Se selecciona un dominio bien fundado.

2. $P(x)$ se prueba siendo x un valor mínimo en dicho dominio
 ↳ Se demuestra para el caso raíz

3. Hipótesis recursiva

$P(y) \Rightarrow P(x)$
 siendo $y < x$

4. se prueba $P(x)$

Ejemplo 1

Demuestre la validez del programa

→ $\begin{cases} \rightarrow \text{Factoriales}[n-1] := \\ \text{If } [\text{Or } [n \leq 0, n = 1], \text{Return } [1]] \text{ then } n * \text{Factoriales}[n-1] \end{cases}$

$\mathbb{N} \cup \{0\} \rightarrow$ bien fundado $(S) \subseteq \mathbb{N} \cup \{0\}$

Finaliza, $\forall m, m \in \mathbb{N} \cup \{0\}; m \leq n$

$\text{Factoriales}[n] = \begin{cases} n=1 \rightarrow 1! = 1 \\ \text{Factoriales}[n-1] \end{cases}$
 $n-1 < n$

Ejemplo 2

a^n ↓
 $\begin{cases} \text{Potencia}[a, n-1] := \\ \text{If } [\text{And } [a = 0, n \neq 0], \text{Return } ["Indefinido"]], \\ \text{If } [a \neq 0 \text{ And } n = 0, \text{Return } [1]] \text{ then } a * \text{Potencia}[a, n-1] \end{cases}$

Determine si la recursión es válida en $\mathbb{N} \cup \{0\}$
 ¿Qué ocurre si el dominio es \mathbb{Z} ?

$n=0 \rightarrow$ "Indefinido"

Hipótesis recursiva

$\forall m, m \in \mathbb{N} \vee 0 \neq m, m < n \checkmark$

Potencia $[a, n]$

$\hookrightarrow a * \text{Potencia}[a, n-1]$
 Finaliza

¿Qué pasa con (\mathbb{Z}) !?

No \$ No finita

Potencia $[a, n] \checkmark$
 $\downarrow \downarrow$
 2^{-5}

\$ Recursion Limit ...

Ejemplo 3

$\text{Dato}[a] := \text{If}[L \neq \{\}, v = \text{Last}[L];$
 $\rightarrow \text{If}[v == a, \text{Return}["\text{Dato encontrado}"]]$
 $L = \text{Delete}[L, \text{Length}[L]]; \text{Dato}[a]$
 $\text{Return}["\text{Dato no encontrado}"]]$

$(\mathbb{N}), n \geq 1$
 1. Encuentra
 2. Iteración \rightarrow no fue hallado

$\forall m, m \in \mathbb{N}, m < n$

$\text{Dato}[a] \rightarrow n$
 $\hookrightarrow n-1 \rightarrow n-1 < n$

Finaliza

Sí es válido.

Otros ejemplos de recursividades

En Mathematica:

$$\rightarrow \text{MCD}[n, m] := \text{If}[n == 0, \text{Return}[m], \text{MCD}[\text{Mod}[m, n], n]]$$

$$\text{MCD}[120, 144]$$

$$\rightarrow \text{Mod}[144, 120] = 24$$

$$\text{MCD}[24, 120]$$

$$\rightarrow \text{Mod}[120, 24] = 0$$

$$\text{MCD}[0, 24] = 24 \checkmark$$

$$n \geq m \vee m \geq n$$

De manera directa en el software Mathematica:

$$\text{GCD}[n, m] \checkmark$$

$$\text{LCM}[n, m] \rightarrow (n, m)$$

Algoritmo Quicksort

1. Se selecciona un elemento de una lista L , denominado pivote.
2. Se ordenan los elementos alrededor del pivote, quedando a su izquierda los datos menores a él y a su derecha los mayores.
3. La lista se divide en dos: la de la izquierda y la de la derecha.
4. Se llama a este algoritmo para ordenar la sublista izquierda.
5. Se llama a este algoritmo para ordenar la sublista derecha.

$$\begin{aligned} i &= 1 & L[i] > \text{pivote} \wedge L[j] < \text{pivote} \\ j &= n & \rightarrow L[i] \leftrightarrow L[j] \\ & & i++ \wedge j-- \end{aligned}$$

Una implementación en Mathematica es:

```

QuickSort [begin_, end_] := If [begin < end, i = begin;
j = end; piv = L [Floor [(begin+end)/2]];
while [i ≤ j, while [L [i]] < piv, i++];
while [L [j]] > piv, j--];
If [i ≤ j, aux = L [i]; L [i] = L [j];
L [j] = aux; i++; j--]];
Print [L];

```

If [begin < j, QuickSort [begin, j];

If [i < end, QuickSort [i, end];]

Por ejemplo: $L = \{10, -1, 3, 5, 7\}$; QuickSort [1, n];

$\{10, -1, 3, 5, 7\}$ $\frac{1+5}{2} = 3 \rightarrow 3$

#1 $\{3, -1, 10, 5, 7\}$

#2 $\{-1, 3, 10, 5, 7\}$

#3 $\{-1, 3, 7, 5, 10\}$

#4 $\{-1, 3, 5, 7, 10\}$

#5 $\{-1, 3, 5, 7, 10\}$

