Recursividad

Problemas y programas

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

9 de noviembre de 2020

Definición

Temas

Recursividad

Definición (Recursividad)

Un elemento puede definirse en términos de sí mismo.

Una *definición recursiva* consta de dos partes:

1. *Caso base* el valor para el cual se conoce la respuesta directamente.
2. *Recursión* la definición de un elemento en términos de la misma definición, pero para un valor diferente.

Ejemplo: **factorial**

1. $0! = 1$
2. $n! = n \cdot (n-1)!$

$$n! = \begin{cases} 1 & \text{si } n = 0 \\ n \cdot (n-1)! & \text{si } n > 0 \end{cases}$$

Interpretación

- Para calcular el valor de la función para un valor concreto es necesario ir componiendo las llamadas a la función hasta caer en el caso base.

En el ejemplo:

$$
\begin{aligned}
5! &= 5 \cdot 4! \\
&= 5 \cdot (4 \cdot 3!) \\
&= 5 \cdot 4 \cdot (3 \cdot 2!) \\
&= 5 \cdot 4 \cdot 3 \cdot (2 \cdot 1!) \\
&= 5 \cdot 4 \cdot 3 \cdot 2 \cdot (1 \cdot 0!) \\
&= 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 \cdot 1 \\
&= 120
\end{aligned}
$$

Más casos base

A veces es conveniente definir más de un caso base para reducir el número de llamadas recursivas:

$$n! = \begin{cases} 1 & \text{si } n = 0 \\ 1 & \text{si } n = 1 \\ n \cdot (n-1)! & \text{si } n > 0 \end{cases}$$

$$\begin{aligned} 5! &= 5 \cdot 4! \\ &= 5 \cdot (4 \cdot 3!) \\ &= 5 \cdot 4 \cdot (3 \cdot 2!) \\ &= 5 \cdot 4 \cdot 3 \cdot (2 \cdot 1!) \\ &= 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 \\ &= 120 \end{aligned}$$

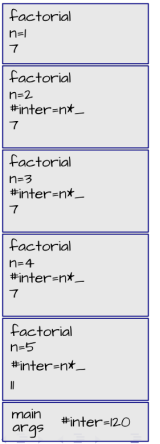
Recursión en Java

Temas

Código

- Las funciones recursivas se pueden calcular directamente en lenguajes de programación que las implementen.

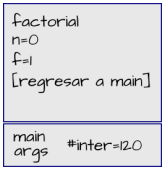
```
1  public class Recursión {
2
3    /** Calcula el factorial de un número recursivamente. */
4    public static long factorial(long n) {
5      if (n < 0) throw new IllegalArgumentException("Indefinido"
           ->);
6      if (n == 0 || n == 1) return 1;      // Casos base
7      else return n * factorial(n - 1);    // Llamada recursiva
8    }
9
10   public static void main(String[] args) {
11     System.out.println("El␣factorial␣de␣5␣es␣" + factorial(5))
           ->;
12   }
13 }
```



Iteración y recursión son equivalentes

- Cualquier función que se pueda programar iterativamente se puede programar recursivamente y visceversa.
- A veces es más sencillo programarla de un modo o del otro.

```
1  public class Iteración {
2    /** Calcula el factorial de un número recursivamente. */
3    public static long factorial(long n) {
4      if (n < 0) throw new IllegalArgumentException("Indefinido"
           ->);
5      long f = 1;
6      while(n > 1) {
7        f *= n;
8        n--;
9      }
10     return f;
11   }
12 }
```



Recursión doble

Temas

Recursión doble: Fibonacci

- Se dice que tenemos un caso de *recursión doble* cuando la función se manda llamar a sí misma dos veces.

Se dice que un criador de conejos comienza con una pareja de conejos bebés:

$$f(1) = 1$$

Al cabo de un mes los conejos han crecido, cada pareja adulta podrá tener una pareja de conejos y ahora hay dos parejas de conejos:

$$f(2) = 1$$
$$f(3) = f(2) + f(1) = 1 + 1 = 2$$

Mes con mes el número total de conejos son los que se tenían el mes anterior (nacieron o ya eran adultos) más los hijos de los que ya nacieron hace dos meses o más.

$$f(n) = f(n-1) + f(n-2)$$

Código

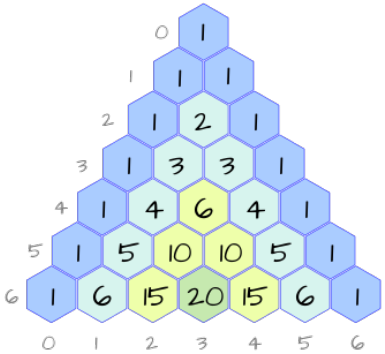
```java
public static long fibo(long n) {
  if (n < 1) throw new IllegalArgumentException("Indefinido");
  if (n == 1 || n == 2) return 1;
  else return fibo(n-1) + fibo(n-2);
}
```

- **Ojo:** ¿cuántas veces se manda llamar esta función?

Complejidad: https://www.ugr.es/~eaznar/fibo.htm

Temas

Triángulo de Pascal



$$P(i,j) = \begin{cases} 1 & \text{para } j = 0, j = i \\ P(i-1,j-1) + P(i-1,j) & \text{en otro caso} \end{cases} \quad (1)$$

Temas

Torres de Hanoi

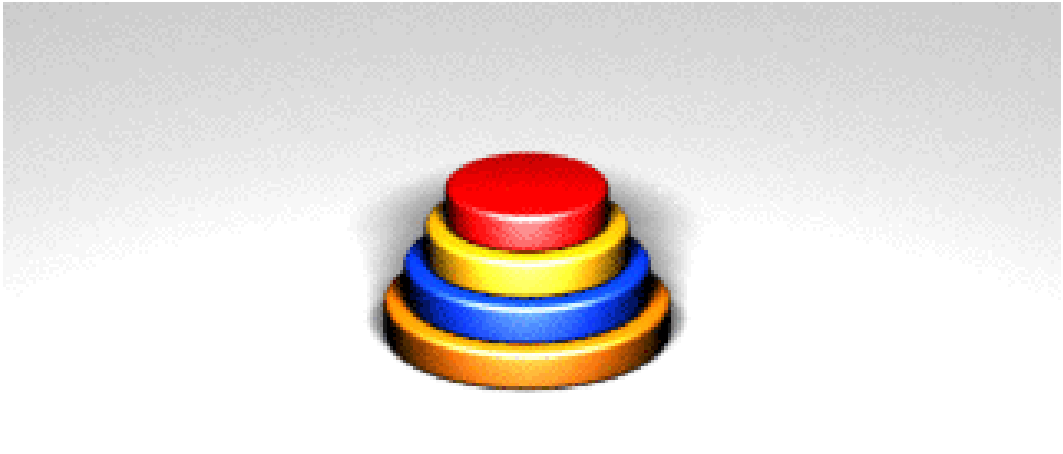


Figura: https://es.wikipedia.org/wiki/Torres_de_Han%C3%B3i#/media/Archivo:Tower_of_Hanoi_4.gif

Bibliografía I


Román, Leobardo López (2011). *Programación estructurada y orientada a objetos. Un enfoque algorítmico.* 3.ª ed. Alfaomega.

Sucesión de Fibonacci https://es.wikipedia.org/wiki/Sucesi%C3%B3n_de_Fibonacci

Licencia