

Algoritmos y estructuras de datos

Recursión

Francisco Javier Zaragoza Martínez

Universidad Autónoma Metropolitana Unidad Azcapotzalco
Departamento de Sistemas



15 de noviembre de 2023

Tony Hoare

Demos crédito a los diseñadores de Algol 60 que incluyeron la **recursión** en su lenguaje y me permitieron describir mi invento elegantemente al mundo.

Scott Alexander

“Lo sé”, dije yo. “Yo sé que sabes”, dijo ella, “pero no sabía que tú sabías que yo sabía que tú sabías y ahora lo sé”.

Real Academia Española

Recursión: Recursividad. **Recursividad**: Cualidad de recursivo. **Recursivo**: (1) Sujeto a reglas o pautas recurrentes. (2) Proceso que se aplica de nuevo al resultado de haberlo aplicado previamente. (3) Estructura que puede contener como constituyente otra del mismo tipo. **Recurrente**: Que recurre. **Recurrir**: Volver al lugar de donde salió.

Recursión

La **recursión** es un método en el cual la solución a un problema se alcanza resolviendo primero instancias más pequeñas del mismo (**caso recursivo**) hasta que sean tan pequeñas que se puedan resolver directamente (**caso base**).

Ejemplo

- 1 Cálculo del factorial.
- 2 Cálculo de potencias.
- 3 Máximo común divisor.
- 4 Números de Fibonacci.
- 5 El juego de Nim.

La recursión es una de las ideas centrales de la computación.

Cálculo del factorial

Definiciones

Iterativa

Usualmente se define $f(n) = n!$ como el producto $1 \cdot 2 \cdots (n - 1) \cdot n$.

Recursiva

Si nos damos cuenta que la primera parte de ese producto es $f(n - 1) = 1 \cdot 2 \cdots (n - 1)$, entonces también podemos definir el factorial **recursivamente** de esta manera:

$$f(n) = \begin{cases} 1 & \text{si } n = 0 \text{ (caso base)} \\ n \cdot f(n - 1) & \text{si } n > 0 \text{ (caso recursivo)} \end{cases}$$

Cálculo del factorial

Ejemplo y árbol de recursión

$$\begin{aligned}f(3) &= 3 \cdot f(2) \\&= 3 \cdot 2 \cdot f(1) \\&= 3 \cdot 2 \cdot 1 \cdot f(0) \\&= 3 \cdot 2 \cdot 1 \cdot 1 \\&= 3 \cdot 2 \cdot 1 \\&= 3 \cdot 2 \\&= 6\end{aligned}$$

$$f(3) = 3 \cdot f(2)$$

3

Cálculo del factorial

Ejemplo y árbol de recursión

$$\begin{aligned}f(3) &= 3 \cdot f(2) \\&= 3 \cdot 2 \cdot f(1) \\&= 3 \cdot 2 \cdot 1 \cdot f(0) \\&= 3 \cdot 2 \cdot 1 \cdot 1 \\&= 3 \cdot 2 \cdot 1 \\&= 3 \cdot 2 \\&= 6\end{aligned}$$

$$\begin{aligned}f(3) &= 3 \cdot f(2) \\&\quad \downarrow \\f(2) &= 2 \cdot f(1)\end{aligned}$$

$$\begin{aligned}&3 \\&\quad \downarrow \\&2\end{aligned}$$

Cálculo del factorial

Ejemplo y árbol de recursión

$$\begin{aligned}f(3) &= 3 \cdot f(2) \\&= 3 \cdot 2 \cdot f(1) \\&= 3 \cdot 2 \cdot 1 \cdot f(0) \\&= 3 \cdot 2 \cdot 1 \cdot 1 \\&= 3 \cdot 2 \cdot 1 \\&= 3 \cdot 2 \\&= 6\end{aligned}$$

$$\begin{aligned}f(3) &= 3 \cdot f(2) \\&\downarrow \\f(2) &= 2 \cdot f(1) \\&\downarrow \\f(1) &= 1 \cdot f(0)\end{aligned}$$

$$\begin{aligned}&3 \\&\downarrow \\&2 \\&\downarrow \\&1\end{aligned}$$

Cálculo del factorial

Ejemplo y árbol de recursión

$$\begin{aligned}f(3) &= 3 \cdot f(2) \\&= 3 \cdot 2 \cdot f(1) \\&= 3 \cdot 2 \cdot 1 \cdot f(0) \\&= 3 \cdot 2 \cdot 1 \cdot 1 \\&= 3 \cdot 2 \cdot 1 \\&= 3 \cdot 2 \\&= 6\end{aligned}$$

$$\begin{aligned}f(3) &= 3 \cdot f(2) \\&\downarrow \\f(2) &= 2 \cdot f(1) \\&\downarrow \\f(1) &= 1 \cdot f(0) \\&\downarrow \\f(0) &= 1\end{aligned}$$

$$\begin{aligned}3 \\&\downarrow \\2 \\&\downarrow \\1 \\&\downarrow \\0\end{aligned}$$

Cálculo del factorial

Implementación

Usando la definición recursiva

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

Llegamos a esta implementación:

```
int factorial(int n) {  
    if (n == 0)  
        return 1;  
    else  
        return n*factorial(n-1);  
}
```

Cálculo del factorial

Ejercicios

- 1 ¿Cuántas llamadas recursivas hace `factorial(n)` en términos de n ?
- 2 ¿Cuántas multiplicaciones hace `factorial(n)` en términos de n ?
- 3 Escribe una función iterativa `int factorial(int n)`.
- 4 ¿Cuántas multiplicaciones hace tu función en términos de n ?
- 5 Escribe funciones iterativa y recursiva `int suma(int n)` que calculen la suma $t(n) = 1 + 2 + \dots + n$. Para la versión recursiva observa que si $n > 0$ entonces $t(n) = n + t(n - 1)$.
- 6 Escribe funciones iterativa y recursiva `int suma(int n, int a[])` que calculen la suma $s(n) = a_0 + a_1 + \dots + a_{n-1}$.

Cálculo de potencias

Definiciones

Iterativa

Usualmente se define $f(x, n) = x^n$ como el producto $\overbrace{x \cdot x \cdots x \cdot x}^{n \text{ veces}}$.

Recursiva

Si nos damos cuenta que la primera parte de ese producto es $f(x, n - 1) = \overbrace{x \cdot x \cdots x}^{n - 1 \text{ veces}}$, entonces también podemos definir la exponenciación recursivamente de esta manera:

$$f(x, n) = \begin{cases} 1 & \text{si } n = 0 \text{ (caso base)} \\ x \cdot f(x, n - 1) & \text{si } n > 0 \text{ (caso recursivo)} \end{cases}$$

Cálculo de potencias

Ejemplo y árbol de recursión

$$\begin{aligned} f(2, 3) &= 2 \cdot f(2, 2) \\ &= 2 \cdot 2 \cdot f(2, 1) \\ &= 2 \cdot 2 \cdot 2 \cdot f(2, 0) \\ &= 2 \cdot 2 \cdot 2 \cdot 1 \\ &= 2 \cdot 2 \cdot 2 \\ &= 2 \cdot 4 \\ &= 8 \end{aligned}$$

$$f(2, 3) = 2 \cdot f(2, 2) \quad (2, 3)$$

Cálculo de potencias

Ejemplo y árbol de recursión

$$\begin{aligned}f(2, 3) &= 2 \cdot f(2, 2) \\&= 2 \cdot 2 \cdot f(2, 1) \\&= 2 \cdot 2 \cdot 2 \cdot f(2, 0) \\&= 2 \cdot 2 \cdot 2 \cdot 1 \\&= 2 \cdot 2 \cdot 2 \\&= 2 \cdot 4 \\&= 8\end{aligned}$$

$$\begin{aligned}f(2, 3) &= 2 \cdot f(2, 2) \\&\quad \downarrow \\f(2, 2) &= 2 \cdot f(2, 1)\end{aligned}$$

$$\begin{aligned}(2, 3) \\&\quad \downarrow \\(2, 2)\end{aligned}$$

Cálculo de potencias

Ejemplo y árbol de recursión

$$\begin{aligned}f(2, 3) &= 2 \cdot f(2, 2) \\&= 2 \cdot 2 \cdot f(2, 1) \\&= 2 \cdot 2 \cdot 2 \cdot f(2, 0) \\&= 2 \cdot 2 \cdot 2 \cdot 1 \\&= 2 \cdot 2 \cdot 2 \\&= 2 \cdot 4 \\&= 8\end{aligned}$$

$$\begin{aligned}f(2, 3) &= 2 \cdot f(2, 2) \\&\quad \downarrow \\f(2, 2) &= 2 \cdot f(2, 1) \\&\quad \downarrow \\f(2, 1) &= 2 \cdot f(2, 0)\end{aligned}$$

$$\begin{aligned}&(2, 3) \\&\quad \downarrow \\&(2, 2) \\&\quad \downarrow \\&(2, 1)\end{aligned}$$

Cálculo de potencias

Ejemplo y árbol de recursión

$$\begin{aligned} f(2, 3) &= 2 \cdot f(2, 2) \\ &= 2 \cdot 2 \cdot f(2, 1) \\ &= 2 \cdot 2 \cdot 2 \cdot f(2, 0) \\ &= 2 \cdot 2 \cdot 2 \cdot 1 \\ &= 2 \cdot 2 \cdot 2 \\ &= 2 \cdot 4 \\ &= 8 \end{aligned}$$

$$\begin{aligned} f(2, 3) &= 2 \cdot f(2, 2) \\ &\downarrow \\ f(2, 2) &= 2 \cdot f(2, 1) \\ &\downarrow \\ f(2, 1) &= 2 \cdot f(2, 0) \\ &\downarrow \\ f(2, 0) &= 1 \end{aligned}$$

$$\begin{aligned} (2, 3) \\ &\downarrow \\ (2, 2) \\ &\downarrow \\ (2, 1) \\ &\downarrow \\ (2, 0) \end{aligned}$$

Cálculo de potencias

Implementación

Usando la definición recursiva

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

Llegamos a esta implementación:

```
double potencia(double x, int n) {  
    if (n == 0)  
        return 1.0;  
    else  
        return x*potencia(x, n-1);  
}
```


Cálculo de potencias

Ejercicios

- 1 ¿Cuántas llamadas recursivas hace `potencia(x,n)` en términos de n ?
- 2 ¿Cuántas multiplicaciones hace `potencia(x,n)` en términos de n ?
- 3 Escribe una función iterativa `double potencia(double x, int n)`.
- 4 ¿Cuántas multiplicaciones hace tu función en términos de n ?
- 5 Escribe una nueva función recursiva que calcule x^n basada en la siguiente idea: Si $n = 0$, entonces $x^n = 1$. Si $n = 1$, entonces $x^n = x$. Si $n \geq 2$ es par, entonces $x^n = (x^{n/2})^2$. Finalmente, si $n \geq 3$ es impar, entonces $x^n = x \cdot (x^{(n-1)/2})^2$. ¿Cuántas multiplicaciones y llamadas recursivas se hacen en términos de n ?

Máximo común divisor

Definiciones

Dados dos enteros positivos a y b , el **máximo común divisor** de a y b es el entero positivo más grande d que es **divisor** tanto de a como de b .

Ejemplo

Los divisores positivos de 54 son 1, 2, 3, 6, 9, 18, 27, 54 y los divisores positivos de 42 son 1, 2, 3, 6, 7, 14, 21, 42. El máximo común divisor de 54 y 42 es 6.

Algoritmo de Euclides

Podemos calcular **recursivamente** el máximo común divisor de esta manera:

$$f(a, b) = \begin{cases} a & \text{si } b = 0 \text{ (caso base)} \\ f(b, a \bmod b) & \text{si } b > 0 \text{ (caso recursivo)} \end{cases}$$

Máximo común divisor

Ejemplo y árbol de recursión

$$\begin{aligned} f(54, 42) &= f(42, 54 \bmod 42) & f(54, 42) &= f(42, 12) & (54, 42) \\ &= f(42, 12) \\ &= f(12, 42 \bmod 12) \\ &= f(12, 6) \\ &= f(6, 12 \bmod 6) \\ &= f(6, 0) \\ &= 6 \end{aligned}$$

Máximo común divisor

Ejemplo y árbol de recursión

$$\begin{aligned} f(54, 42) &= f(42, 54 \bmod 42) \\ &= f(42, 12) \\ &= f(12, 42 \bmod 12) \\ &= f(12, 6) \\ &= f(6, 12 \bmod 6) \\ &= f(6, 0) \\ &= 6 \end{aligned}$$

$$\begin{aligned} f(54, 42) &= f(42, 12) \\ &\downarrow \\ f(42, 12) &= f(12, 6) \end{aligned}$$

$$\begin{aligned} (54, 42) \\ \downarrow \\ (42, 12) \end{aligned}$$

Máximo común divisor

Ejemplo y árbol de recursión

$$\begin{aligned} f(54, 42) &= f(42, 54 \bmod 42) \\ &= f(42, 12) \\ &= f(12, 42 \bmod 12) \\ &= f(12, 6) \\ &= f(6, 12 \bmod 6) \\ &= f(6, 0) \\ &= 6 \end{aligned}$$

$$\begin{aligned} f(54, 42) &= f(42, 12) \\ &\downarrow \\ f(42, 12) &= f(12, 6) \\ &\downarrow \\ f(12, 6) &= f(6, 0) \end{aligned}$$

$$\begin{aligned} (54, 42) \\ &\downarrow \\ (42, 12) \\ &\downarrow \\ (12, 6) \end{aligned}$$

Máximo común divisor

Ejemplo y árbol de recursión

$$\begin{aligned}f(54, 42) &= f(42, 54 \bmod 42) \\&= f(42, 12) \\&= f(12, 42 \bmod 12) \\&= f(12, 6) \\&= f(6, 12 \bmod 6) \\&= f(6, 0) \\&= 6\end{aligned}$$

$$\begin{aligned}f(54, 42) &= f(42, 12) \\&\downarrow \\f(42, 12) &= f(12, 6) \\&\downarrow \\f(12, 6) &= f(6, 0) \\&\downarrow \\f(6, 0) &= 6\end{aligned}$$

$$\begin{aligned}(54, 42) \\&\downarrow \\(42, 12) \\&\downarrow \\(12, 6) \\&\downarrow \\(6, 0)\end{aligned}$$

Máximo común divisor

Implementación

Usando la definición recursiva

$$f(a, b) = \begin{cases} a & \text{si } b = 0 \\ f(b, a \bmod b) & \text{si } b > 0 \end{cases}$$

Llegamos a esta implementación:

```
int euclides(int a, int b) {  
    if (b == 0)  
        return a;  
    else  
        return euclides(b, a % b);  
}
```

Ejercicios

- 1 Los coeficientes combinatorios $C(n, m)$ se definen recursivamente de la siguiente manera: Si $m = 0$ o $m = n$, entonces $C(n, m) = 1$. Si $0 < m < n$, entonces $C(n, m) = C(n - 1, m - 1) + C(n - 1, m)$. Escribe una función recursiva `int combina(int n, int m)` que calcule $C(n, m)$. ¿Cuántas sumas hace la llamada `combina(n, m)`?
- 2 También hay una fórmula para los coeficientes combinatorios:

$$C(n, m) = \frac{n!}{m!(n - m)!}.$$

Escribe una función iterativa que calcule $C(n, m)$. ¿Cuántos productos hace tu función?

Números de Fibonacci

Definición

Los números de Fibonacci se definen recursivamente de esta manera:

$$f(n) = \begin{cases} 0 & \text{si } n = 0 \text{ (caso base)} \\ 1 & \text{si } n = 1 \text{ (caso base)} \\ f(n-1) + f(n-2) & \text{si } n > 1 \text{ (caso recursivo)} \end{cases}$$

Ejemplo

$$\begin{aligned} f(5) &= f(4) + f(3) \\ &= (f(3) + f(2)) + (f(2) + f(1)) \\ &= ((f(2) + f(1)) + (f(1) + f(0))) + ((f(1) + f(0)) + 1) \\ &= (((f(1) + f(0)) + 1) + (1 + 0)) + ((1 + 0) + 1) \\ &= (((1 + 0) + 1) + (1 + 0)) + ((1 + 0) + 1) \\ &= 5 \end{aligned}$$

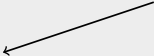
Números de Fibonacci

Árbol de recursión

$$f(5) = f(4) + f(3)$$

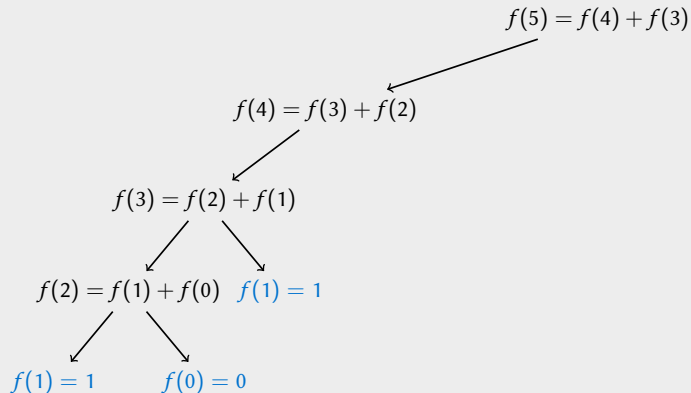
Números de Fibonacci

Árbol de recursión

$$f(5) = f(4) + f(3)$$

$$f(4) = f(3) + f(2)$$

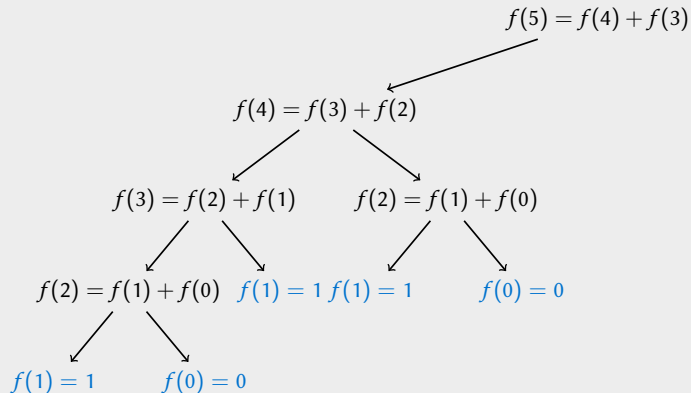
Números de Fibonacci

Árbol de recursión



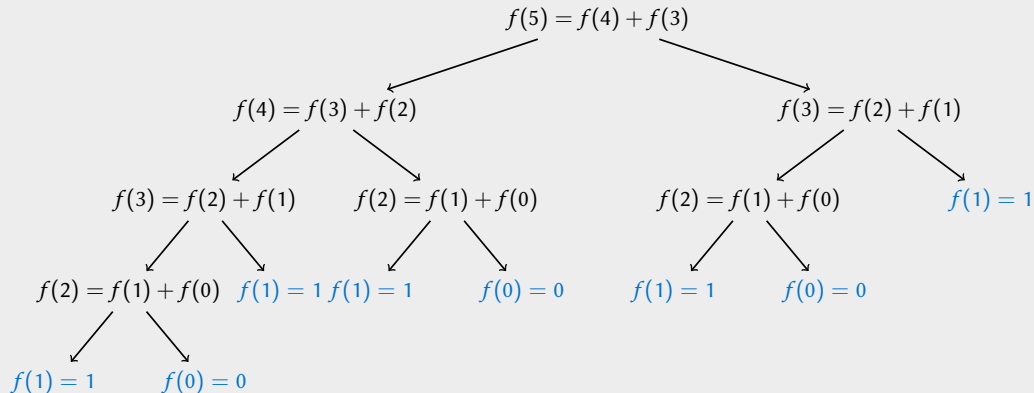
Números de Fibonacci

Árbol de recursión



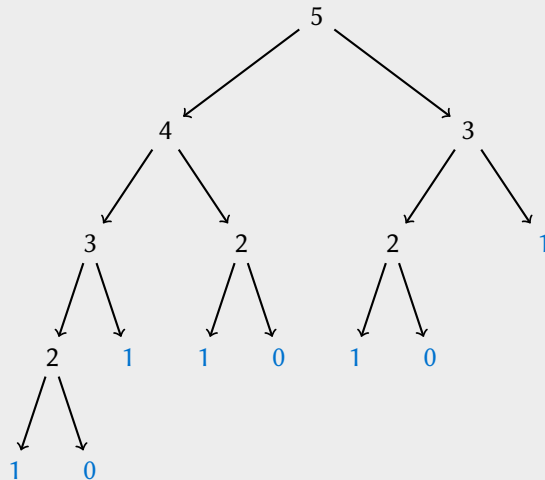
Números de Fibonacci

Árbol de recursión



Números de Fibonacci

Árbol de recursión (simplificado)



Números de Fibonacci

Implementación

Usando la definición recursiva

$$f(n) = \begin{cases} 0 & \text{si } n = 0 \\ 1 & \text{si } n = 1 \\ f(n-1) + f(n-2) & \text{si } n > 1 \end{cases}$$

Llegamos a esta implementación:

```
int fibonacci(int n) {  
    if (n == 0)  
        return 0;  
    else if (n == 1)  
        return 1;  
    else  
        return fibonacci(n-1)+fibonacci(n-2);  
}
```


Números de Fibonacci

Ejercicios

- 1 Escribe una función iterativa que encuentre $f(n)$ calculando de forma sucesiva $f(0), f(1), f(2), \dots$. ¿Cuántas sumas hace en términos de n ?
- 2 Escribe una nueva función recursiva que calcule $f(n)$ basada en la siguiente idea: Si $n = 0$, entonces $f(n) = 0$. Si $n = 1$, entonces $f(n) = 1$. Si $n \geq 2$ es par, entonces $f(n) = f(\frac{n}{2} + 1)^2 - f(\frac{n}{2} - 1)^2$. Finalmente, si $n \geq 3$ es impar, entonces $f(n) = f(\frac{n+1}{2})^2 + f(\frac{n-1}{2})^2$. ¿Cuántas llamadas recursivas se hacen en términos de n ?
- 3 Los números de Lucas se definen como $L(0) = 2, L(1) = 1$ y $L(n) = L(n-1) + L(n-2)$ para $n \geq 2$. Escribe funciones iterativa y recursiva `int lucas(int n)` que calculen $L(n)$.
- 4 Los números de Pell se definen como $P(0) = 0, P(1) = 1$ y $P(n) = 2P(n-1) + P(n-2)$ para $n \geq 2$. Escribe funciones iterativa y recursiva `int pell(int n)` que calculen $P(n)$.

El juego de Nim

Definición

En este juego se comienza con $n \geq 1$ piedras. Por turnos, dos jugadores quitan 1, 2 o 3 piedras. Gana el jugador que quita la última piedra.

Análisis del juego de Nim

Si $n \in \{1, 2, 3\}$ el jugador que tiene el turno **puede ganar**, pues puede quitar todas las piedras. Si $n > 3$ el jugador que tiene el turno **puede ganar** si y sólo si el otro jugador **no puede ganar** con alguna de $n - 1$, $n - 2$ o $n - 3$ piedras.

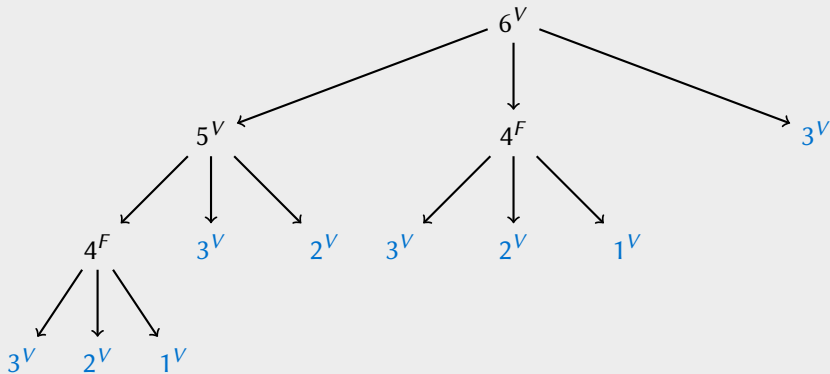
Función de Nim

Sea $f(n)$ verdadero si el jugador que tiene el turno **puede ganar** con n piedras:

$$f(n) = \begin{cases} \text{verdadero} & \text{si } 1 \leq n \leq 3 \text{ (casos base)} \\ \neg f(n-1) \vee \neg f(n-2) \vee \neg f(n-3) & \text{si } n > 3 \text{ (caso recursivo)} \end{cases}$$

El juego de Nim

Árbol de recursión



Juego de Nim

Implementación

Usando la definición recursiva

$$f(n) = \begin{cases} \text{verdadero} & \text{si } 1 \leq n \leq 3 \\ \neg f(n-1) \vee \neg f(n-2) \vee \neg f(n-3) & \text{si } n > 3 \end{cases}$$

Llegamos a esta implementación:

```
int nim(int n) {  
    if (n == 1 || n == 2 || n == 3)  
        return 1;  
    else  
        return !nim(n-1) || !nim(n-2) || !nim(n-3);  
}
```