Recursión

Salvador López Mendoza

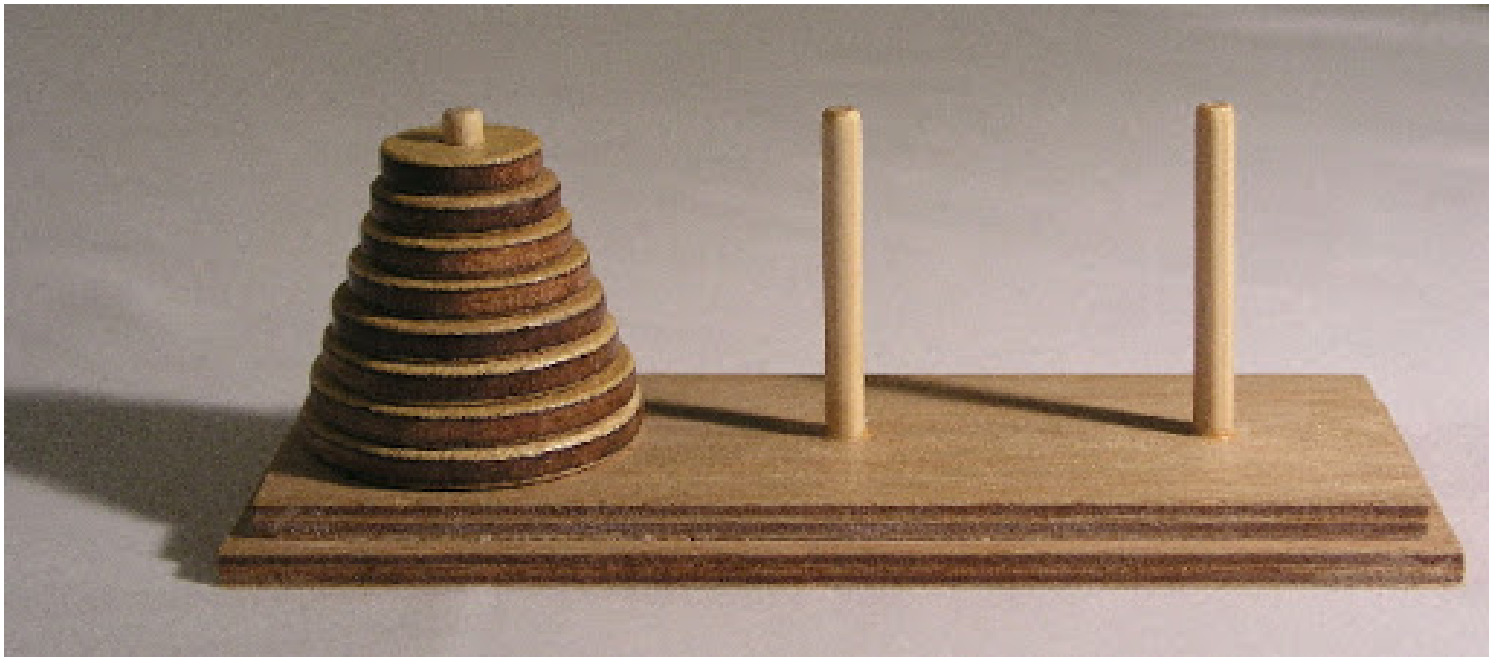
Noviembre de 2022

Leyenda de las torres de Hanoi

Cerca de la ciudad de Hanoi (Vietnam) existe un monasterio desde los principios de la humanidad.
Los monjes tienen una misión: mover todos los discos que se encuentran apilados en un poste a otro poste.
¡Cuando terminen su tarea se acabará la humanidad!

Las torres de Hanoi

Se muestra una imagen con ocho (8) discos en su configuración inicial.



La leyenda de las torres de Hanoi (II)

Leyenda de las torres de Hanoi

Cuando los Dioses crearon el mundo pusieron tres postes en un lugar cercano a lo que ahora se conoce como Hanoi.
En uno de los postes colocaron 64 discos de diferentes diámetros, de mayor a menor.
Se encomendó a un grupo de personas (los monjes) que pasaran todos los discos del primer poste al segundo poste.

Restricción: Solamente se puede pasar un disco (el que está en la parte superior) de un poste a otro poste. No se permite colocar un disco de diámetro mayor sobre uno de diámetro menor.

El mundo se acabará el día en que los monjes terminen de pasar todos los discos del primer poste al segundo.

La leyenda de las torres de Hanoi (III)

Detalles del problema:

- La cantidad total de discos es 64. Cada disco es de un diámetro distinto.
- Hay tres postes. Al iniciar, todos los discos se encuentran en un poste.
- Sólamente se permite mover un disco a la vez.
- Se puede colocar un disco sobre otro disco de diámetro mayor.

La cantidad mínima de movimientos para realizar la tarea es $2^{64} - 1$.

Si se realiza un movimiento cada segundo, el tiempo para terminar la tarea es de $585,000$ millones de años.

El Universo tiene cerca de $14,000$ millones de años de existencia.

Torres de hanoi (II)

Escribir un algoritmo que indique los pasos que se deben seguir para pasar todos los discos de un poste a otro.

En cada movimiento solo se puede tomar el disco que esté en la parte superior de un poste y se le puede colocar en otro poste cuyo disco superior sea de diámetro mayor (o que no tenga discos).

Para facilitar el trabajo se puede asumir que los discos están numerados del 1 al 64. El número del disco corresponde a su diámetro.

Algoritmo que tiene como parámetro la cantidad de discos (n).

Posible ejecución del programa (para $n = 2$):

```
1. Mueve disco 1 de la torre Origen a la torre Auxiliar.
2. Mueve disco 2 de la torre Origen a la torre Destino.
3. Mueve disco 1 de la torre Auxiliar a la torre Destino.
```

Recursión

Definición

La recursión (**recursividad**), consiste en realizar la definición de un concepto en términos de él mismo.

La defininición debe aportar algo.

Definiciones de diccionario escolar (primaria):

> **Liebre.** Especie de *conejo*.
>
> **Conejo.** Similar a la *liebre*.

Es un ejemplo de una definición recursiva mal hecha.

Recursión

La recursión se emplea en muchas situaciones, la más conocida es en las matemáticas.

Recursión en la programación

Cuando un método involucra, bajo ciertas circunstancias, llamadas a sí mismo se dice que éste es **recursivo**.

Recursión. Ejemplos

Las soluciones recursivas son muy utilizadas en matemáticas. Ejemplos:

- El factorial de un número n.

  Definición no recursiva: $n! = \prod_{i=1}^{n} i$

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

- El n-ésimo elemento de la sucesión de Fibonacci.
  1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

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

Otros ejemplos

- La suma de los primeros n enteros.

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

- Una función numérica.

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

Para el útlimo ejemplo, ¿cuánto vale $f(3)$?

Solución a f(3)

```
f(3) = 2*f(2) + 3*3

          f(2) = 2*f(1) + 2*2

                    f(1) = 2*f(0) + 1*1

                              f(0) = 0          (Solución conocida)

                    f(1) = 2*f(0) + 1*1 = 2*0 + 1*1 = 1

          f(2) = 2*f(1) + 2*2 = 2*1 + 4 = 6

f(3) = 2*f(2) + 3*3 = 2*6 + 3*3 = 21
```

Ejemplo de recursión

Implementación de la función recursiva $f(x)$ en Java:

```
public long f(int x) {
  if (x <= 0)
    return 0 ;
  else
    return 2*f(x-1) + x*x ;
}
```

La primera parte del `if` se denomina **caso base** y es aquel en que no se hace llamada a sí mismo y por tanto es el que permite que termine la recursión, *siempre y cuando el resto del método trabaje para llegar al caso base.*

Recursión. Terminación

Se debe garantizar que las llamdas recursivas terminan.

En el caso del ejemplo anterior, el parámetro de la función recursiva es un valor entero.

- Si al invocar a la funcion el valor es menor o igual a 0, el método termina.
  Devuelve el valor 0 como resultado.
- Si al invocar a la función el valor es positivo, se descadenan las llamadas recursivas. Cada una con un valor menor en una unidad.
  En algún momento (después de n llamadas recursivas) el valor del parámetro es 0 y termina la recursión.
  Se pueden realizar los cálculos que estaban pendientes (en orden inverso), hasta entregar el resultado final.

Recursión. Terminación (II)

```
public int duda (int n) {
  if (n == 0) {
    return 0 ;
  } else {
    return duda(n/3 + 1) + n - 1 ;
  }
}
```

¿Cuánto vale `duda(3)`?

```
duda(3) = duda(3/3 + 1) + 3 - 1 ;
          duda(2) = duda (2/3 + 1)+ 2 - 1 ;
                    duda(1) = duda(1/3 + 1) + 1 - 1 ;
                              duda(1) = duda(1/3 + 1) + 0 ;
                                        ...
```

¡Cuidado! Es una función recursiva que nunca termina. No se llega al caso base.

Fundamentos matemáticos

En los sistemas matemáticos se definen axiomas y reglas de inferencia. Se construyen sistemas formales basados en enunciados que se pueden demostrar.

¿Qué es computable?

- Respuesta informal:
  Aquello (algoritmos) que se puede demostrar que se puede resolver en una computadora.
- Otra forma (más formal) de decirlo:
  Aquello que se puede representar mediante una ***Máquina de Turing***.

  Teoría de la computabilidad.

Inducción matemática

Una forma de demostrar la veracidad de ciertos enunciados es mediante la **inducción matemática**.

Se utiliza cuando hay una gran cantidad de casos, relacionados de una forma especial.

Esquema de demostración:

1. Se demuestra el *caso base*.
2. Se supone que el enunciado es válido para el i-ésimo caso y se demuestra que es válido para el $(i+1)$-ésimo caso.

Demostrar que

$$\sum_{i=1}^{i=n} i = \frac{n(n+1)}{2}$$

Las torres de Hanoi, Análisis del problema

Afirmación

Se requieren $2^n - 1$ movimientos para resolver el problema de las torres de Hanoi con n discos.

Se demuestra por **inducción**:

Las torres de Hanoi (II)

1. Para $n = 1$.
   Se pasa el disco 1 de la torre origen a la torre destino.
   Problema resuelto. Se realizó 1 movimiento. $1 = 2^1 - 1$.
2. Se supone que el resultado es válido para cualquier valor $k \geq 1$. Por demostrar, es válido para $k + 1$.
   Se tienen $k + 1$ discos.
   Se mueven los primeros k discos de la torre origen a la torre auxiliar.
   Se realizan $2^k - 1$ movimientos.
   Se mueve el disco $k + 1$ de la torre origen a la torre destino. Se realiza 1 movimiento.
   Se mueven los primeros k discos de la torre auxiliar a la torre destino.
   Se realizan $2^k - 1$ movimientos.
   Problema resuelto. Se realizan $2^k - 1$ movimientos, más 1 movimiento, más $2^k - 1$ movimientos. El total de movimientos es $(2^k - 1) + 1 + (2^k - 1) = 2^{k+1} - 1$.

Las torres de Hanoi (III)

¿Cómo se programa la solución?

La demostración por inducción da la respuesta:

- Si $n = 1$ se mueve el disco del poste origen al poste destino.
- Si $n > 1$ se mueven los primeros $n - 1$ discos del poste origen al poste auxiliar. Se mueve el disco n del poste origen al poste destino. Se mueven los primeros $n - 1$ discos del poste auxiliar al poste destino.

Las torres de Hanoi en Java

```
private static void torresHanoi(int n, String inicial,
                    String fin, String auxiliar) {
  // Revisa si es el caso base
  if (n == 1) {
    System.out.println("Mueve el disco " + n + " del poste "
                       + inicial + " al poste " + fin) ;
  } else {
    // Mueve los primeros n-1 discos al poste auxiliar
    torresHanoi(n-1, inicial, auxiliar, fin) ;
    // Mueve el disco n al poste destino
    System.out.println("Mueve el disco " + n + " del poste "
                       + inicial + " al poste " + fin) ;
    // Mueve los primeros n-1 discos del poste auxiliar al
        poste final
    torresHanoi(n-1, auxiliar, fin, inicial) ;
  }

} // Fin de torresHanoi
```

Solución a problemas recursivos

- Dividir el problema original en uno o varios problemas del mismo tipo que el original pero más pequeños.
- Resolver estos problemas más sencillos.
- Con las soluciones a estos problemas sencillos se construye la solución de los problemas más complejos.

Otro ejemplo de recursión

Imprimir cualquier número entero al revés.

```
public static void invertirEntero (int n) {
  escribirDigito(n % 10) ; // Último dígito por la derecha
  if (n >= 10) {
    invertirEntero(n / 10) ;//Resto del número (invertido)
  } else {
    System.out.println() ;
  }
}

public void escribirDigito(int n) {
  System.out.print(n) ;
}
```

Si se invoca a este método con el valor 4567, imprimira el valor 7654.