## *Ejemplo: Sucesión de Fibonacci*

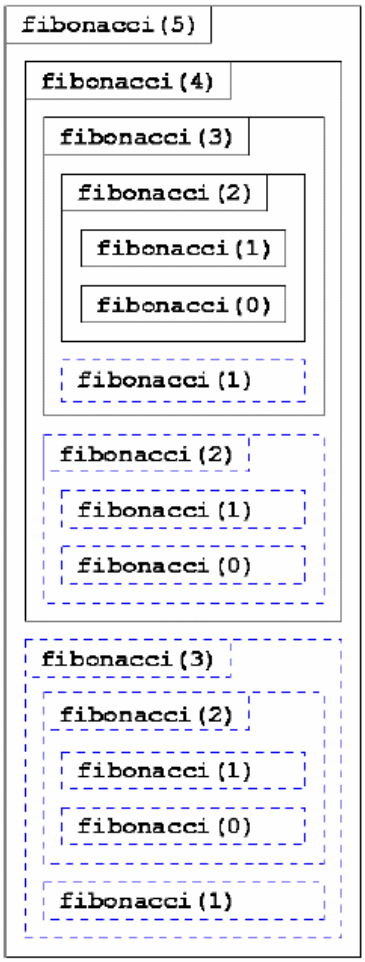
$$Fib(0) = Fib(1) = 1$$
$$Fib(n) = Fib(n-1) + Fib(n-2)$$

*Solución recursiva*

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

*Solución iterativa*

```
static int fibonacci (int n)
{
  int actual, ant1, ant2;

  ant1 = ant2 = 1;

  if ((n == 0) || (n == 1)) {
     actual = 1;
  } else
     for (i=2; i<=n; i++) {
         actual = ant1 + ant2;
         ant2 = ant1;
         ant1 = actual;
     }
  }

  return actual;
}
```




*Cálculo recursivo de* `fibonacci(5)`

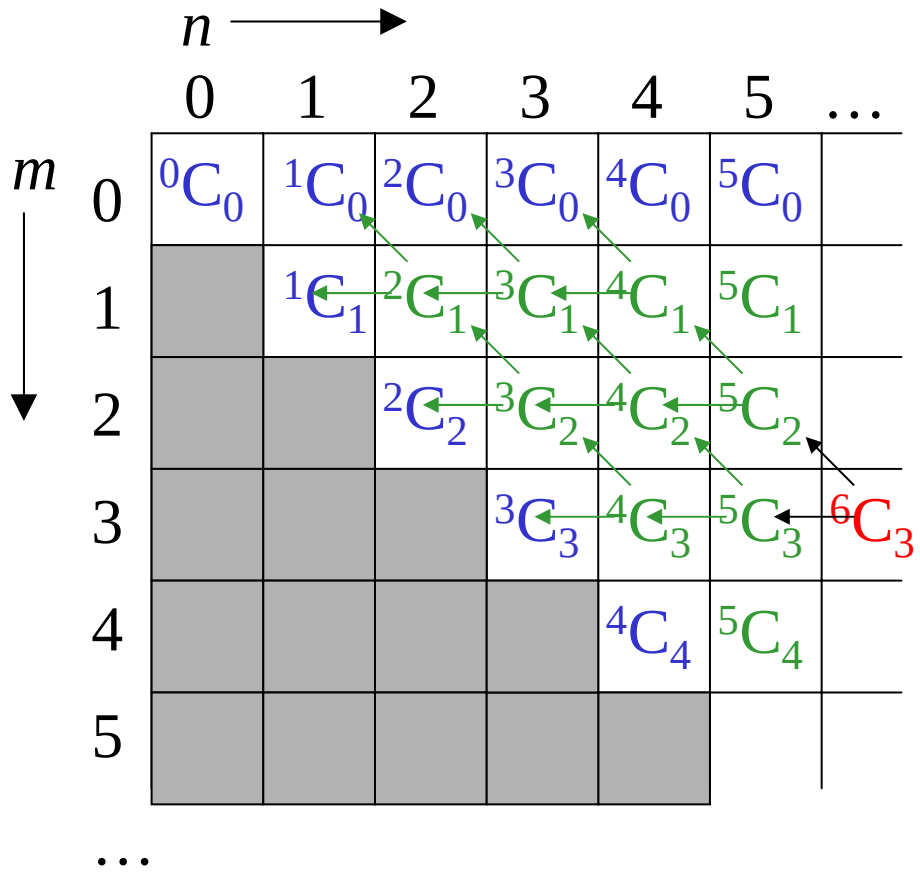
## *Ejemplo: Combinaciones*

¿De cuántas formas se pueden seleccionar
m elementos de un conjunto de n?

Si cogemos dos letras del conjunto {A,B,C,D,E}
podemos obtener 10 parejas diferentes:

| | | | |
|---|---|---|---|
| {A, B} | {A, C} | {A, D} | {A, E} |
| | {B, C} | {B, D} | {B, E} |
| | | {C, D} | {C, E} |
| | | | {D, E} |

$$\binom{n}{m} = \binom{n-1}{m} + \binom{n-1}{m-1}$$



*Implementación recursiva:*

```
static int combinaciones (int n, int m)
{
  if ((m == 0) || (m == n))
     return 1;
  else
     return combinaciones(n-1, m)
          + combinaciones(n-1, m-1);
}
```

El algoritmo recursivo anterior es muy ineficiente porque realiza muchas veces los mismos cálculos.

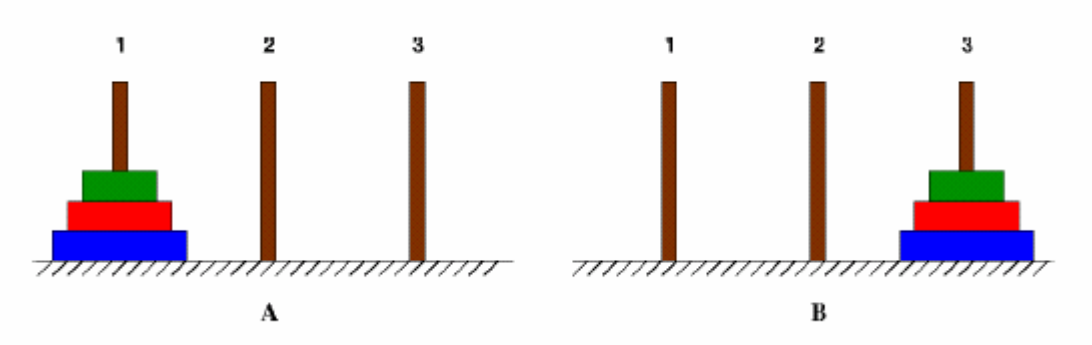
De hecho, se realizan $\binom{n}{m}$ llamadas recursivas.

Para mejorar la eficiencia del algoritmo recursivo podemos buscar un algoritmo iterativo equivalente:

- Usando una matriz en la que iremos almacenando los valores que vayamos calculando para no repetir dos veces el mismo trabajo.

- Aplicando directamente la expresión $\binom{n}{m} = \frac{n!}{m!(n-m)!}$

## *Ejemplo: Las torres de Hanoi*



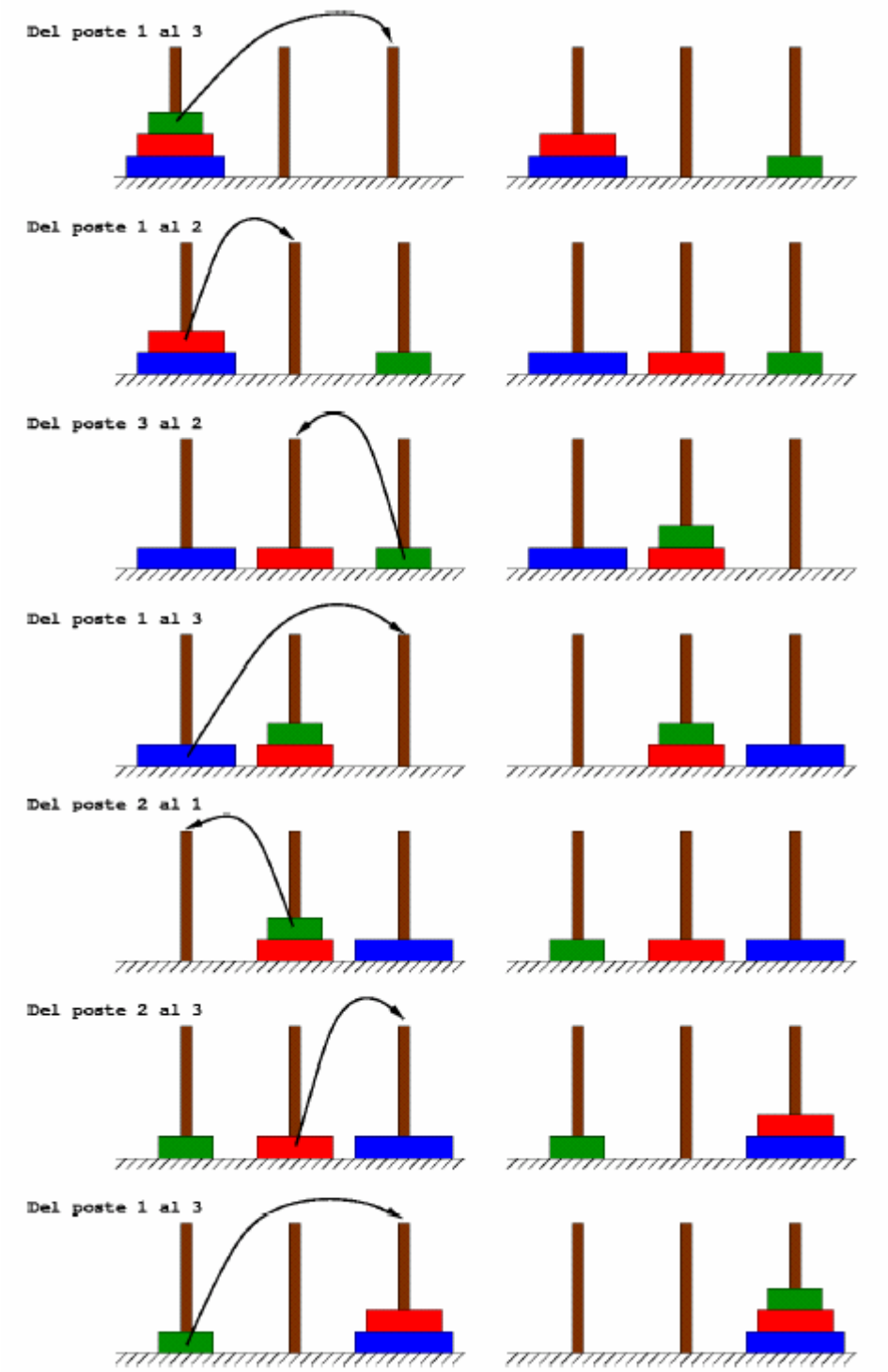

Mover n discos del poste 1 al poste 3
(utilizando el poste 2 como auxiliar):

```
    hanoi (n, 1, 2, 3);
```

Solución recursiva:

```
static void hanoi
       (int n, int inic, int tmp, int fin)
{
  if (n > 0) {

     // Mover n-1 discos de "inic" a "tmp".
     // El temporal es "fin".

     hanoi (n-1, inic, fin, tmp);

     // Mover el que queda en "inic" a "fin"

     System.out.println(inic+“->”+fin);

     // Mover n-1 discos de "tmp" a "fin".
     // El temporal es "inic".

     hanoi (n-1, tmp, inic, fin);
  }
}
```

*Solución para 3 discos*




Según la leyenda, los monjes de un templo tenían que mover una pila de 64 discos sagrados de un sitio a otro. Sólo podían mover un disco al día y, en el templo, sólo había otro sitio en el que podían dejarlos, siempre ordenados de forma que los mayores quedasen en la base.

El día en que los monjes realizasen el último movimiento, el final del mundo habría llegado… ¿en cuántos días?