ALGORITMIA

Tema 2:

Diseño de Algoritmos Recursivos

Solución de los ejercicios:

Suma cifras
Semifactorial
Fibonacci
Rayuela
Plano
Tableros

Curso 2021-2022

Grado en Ingeniería Informática en Tecnologías de la Información
Escuela Politécnica de Ingeniería – Campus de Gijón
Universidad de Oviedo

SUMA CIFRAS

Enunciado del problema.-

Dado un número n, siendo n≥0, diseñar una función recursiva que retorne la suma de las cifras de dicho número n.

SOLUCIÓN.-

- **Especificación formal:**

  Q ≡ { n ≥ 0 }
  Función SUMA_CIFRAS (n:entero) retorna (p:entero)

- **Tipo de inducción: El dominio de nuestra función es IN.**

  Aplicaremos inducción fuerte. Nuestro objetivo es calcular SUMA_CIFRAS(n) y para ello, suponemos conocida la solución del subproblema que supone dividir el número n entre 10, de ese modo, "eliminamos" la última cifra y nos quedamos con las cifras más significativas. Este subproblema es menor que el problema original en base al preorden establecido por los naturales.

- **Análisis por casos:**

  Tenemos que resolver SUMA_CIFRAS(n), esto es, tenemos que solucionar el problema que calcula la suma de las cifras del número n. Lo denominaremos p.

  Supongamos conocida la solución del subproblema SUMA_CIFRAS(n/10), esto es, suponemos conocido la suma de las cifras del número n/10. A su valor asociado lo denominamos p´. Una vez conocido p´ o SUMA_CIFRAS(n/10), únicamente es necesario sumar la última cifra del número n, esto es, sumar n%10.

  En conclusión, el resultado del problema inicial sería:

  p = p´ + n%10 o dicho de otro modo

  SUMA_CIFRAS (n) = SUMA_CIFRAS (n/10) + (n%10)

  El caso base corresponde a los números que tienen una única cifra, esto es, que son menores de 10. En este caso la función retorna el propio número.

- **Algoritmo en pseudocódigo:**

```
Q ≡ { n ≥ 0 }
Función SUMA_CIFRAS ( n:entero ) retorna ( p:entero )
    si n < 10 entonces retorna n
      sino retorna SUMA_CIFRAS (n/10) + (n%10)
    fsi
ffunción
```

SEMIFACTORIAL

Enunciado del problema.-

Dado un número n, siendo n≥1, diseñar un función recursiva que retorne el **semifactorial**, o **doble factorial**, del entero n.

Nota.- el producto de todos los enteros desde 1 hasta un entero no-negativo n que tienen la misma paridad (pares o impares) que n se llama **doble factorial** o **semifactorial** de n y se representa como **n!!**

SOLUCIÓN.-

- **Especificación formal:**

Q ≡ { n ≥ 1 }
Función SEMIFACTORIAL (n:entero) retorna (p:entero)
R ≡ { p = n!! }

- **Tipo de inducción: El dominio de nuestra función es IN.**

Aplicaremos inducción fuerte. Nuestro objetivo es calcular SEMIFACTORIALI(n) y para ello, suponemos conocida la solución del subproblema que supone decrementar n en 2 unidades, esto es, SEMIFACTORIAL(n-2). Este subproblema es menor que el problema original en base al preorden establecido por los naturales.

- **Análisis por casos:**

Tenemos que resolver SEMIFACTORIALI(n), esto es, tenemos que solucionar el problema que calcula el semifactorial de n. Lo denominaremos p.

Supongamos conocida la solución del subproblema SEMIFACTORIAL(n-2), esto es, suponemos conocido el semifactorial de n-2. A su valor asociado lo denominamos p´. Una vez conocido (n-2)!!, únicamente es necesario multiplicar por n para calcular n!!.

En conclusión, el resultado del problema inicial sería:

$$p = p' * n \quad \text{o dicho de otro modo}$$

$$\text{SEMIFACTORIAL}(n) = \text{SEMIFACTORIAL}(n-2) * n$$

Existen dos casos base que corresponden a los subproblemas n=1 y n=2 al no disponer éstos de un problema sucesor. En ambos casos, lo que ha de devolver la función como solución es el propio número n.

- **Algoritmo en pseudocódigo:**

```
Q ≡ { n ≥ 1 }
Función SEMIFACTORIAL ( n:entero ) retorna ( p:entero )
    si n ≤ 2 entonces retorna n
    sino retorna SEMIFACTORIAL (n-2) * n
    fsi
ffunción
R ≡ { p = n!! }
```

FIBONACCI

Enunciado del problema.-

La sucesión de Fibonacci es la siguiente sucesión infinita de números naturales: “La sucesión comienza con los números 0 y 1, y a partir de éstos, cada término es la suma de los dos anteriores.”

Dado un número n, siendo n≥0, diseñar una función recursiva que retorne el número que ocupa la posición n en la sucesión de Fibonacci. Se considerará que el primer número de la serie (el 0) ocupa la posición 0.

SOLUCIÓN.-

- **Especificación formal:**

  Q ≡ { n ≥ 0 }
  Función FIBONACCI (n:entero) retorna (p:entero)
  R ≡ { p = número que ocupa la posición n en la sucesión de Fibonacci }

- **Tipo de inducción: El dominio de nuestra función es IN.**

  Aplicaremos inducción fuerte. Nuestro objetivo es calcular FIBONACCI(n) y para ello, dada la descripción de la sucesión de Fibonacci que figura en el enunciado (“cada término es la suma de los dos anteriores”), suponemos conocida la solución de los subproblemas FIBONACCI(n-1) y FIBONACCI(n-2). Estos subproblemas son los dos problemas anteriores a FIBONACCI(n) y dichos subproblemas son menores que el problema original en base al preorden establecido por los naturales.

- **Análisis por casos:**

  Tenemos que resolver FIBONACCI(n), esto es, tenemos que solucionar el problema para el número que ocupa la posición n en la sucesión. Lo denominaremos p.
  Supongamos conocida la solución de los problemas sucesores:

  - FIBONACCI(n-1), esto es, suponemos conocido el número que ocupa la posición n-1 de la sucesión de Fibonacci. A su valor asociado lo denominamos p1´.
  - FIBONACCI(n-2), esto es, suponemos conocido el número que ocupa la posición n-2 de la sucesión de Fibonacci. A su valor asociado lo denominamos p2´.

  En conclusión, el resultado del problema inicial sería:

  $$p = p1' + p2' \text{ o dicho de otro modo}$$

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

  Existen dos casos base:

  - el número que ocupa la posición 0 de la sucesión de Fibonacci, que es el 0.
  - el número que ocupa la posición 1 de la sucesión de Fibonacci, que es el 1.

- **Algoritmo en pseudocódigo:**

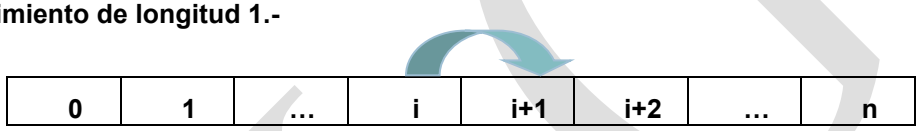
```
Q ≡ { n ≥ 0 }
Función FIBONACCI ( n:entero ) retorna ( p:entero )
    si n ≤ 1 entonces retorna n
    sino retorna FIBONACCI (n-1) + FIBONACCI (n-2)
    fsi
ffunción
R ≡ { p = número que ocupa la posición n en la sucesión de Fibonacci }
```

RAYUELA

Enunciado del problema.-

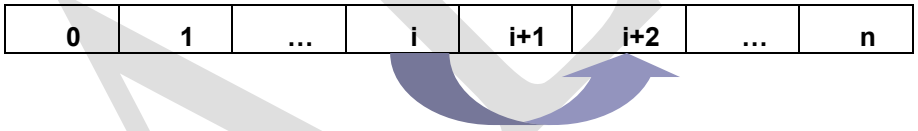
Dados unos cuadros alineados (como si se tratara del juego del cascayu o la rayuela) y numerados desde el 0, el juego consiste en ir saltando hasta el cuadro que ocupa la posición n, siendo n>0, permitiéndose para ello dos posibles movimientos:

Movimiento de longitud 1.-



y

Movimiento de longitud 2.-



Se desea diseñar una función recursiva que calcule de cuántas maneras diferentes se puede llegar al n-ésimo cuadro desde el cuadro de partida (numerado con el 0).

SOLUCIÓN.-

- **Especificación formal:**

```
Q ≡ { n ≥ 1 }
función RAYUELA ( n : entero ) retorna ( s : entero )
R ≡ { s = número de caminos diferentes desde la casilla 0 hasta la casilla n
      construidos éstos únicamente con saltos (movimientos) de longitud
      1 ó 2 }
```

- **Tipo de inducción: El dominio de nuestra función es IN.**

Aplicaremos inducción fuerte. Nuestro objetivo es calcular RAYUELA(n) y para ello suponemos conocida la solución de los subproblemas RAYUELA(n-1) y RAYUELA(n-2), siendo ambos problemas menores que el problema original en base al preorden establecido por los naturales.

- **Análisis por casos:**

  Tenemos que resolver RAYUELA(n), esto es, tenemos que solucionar el problema para la casilla n, su valor asociado lo denominamos s.

  Supongamos conocida la solución de los problemas sucesores:

  - RAYUELA(n-1), esto es, suponemos conocida la resolución del subproblema: "número de caminos diferentes desde la casilla 0 hasta la casilla n-1, construidos éstos únicamente con saltos (movimientos) de longitud 1 ó 2". A su valor asociado lo denominamos s1´.
  - RAYUELA(n-2), esto es, suponemos conocida la resolución del subproblema: "número de caminos diferentes desde la casilla 0 hasta la casilla n-2, construidos éstos únicamente con saltos (movimientos) de longitud 1 ó 2". A su valor asociado lo denominamos s2´.

  El número de caminos diferentes que llevan a la casilla n se divide en dos conjuntos DISJUNTOS: se puede llegar a dicha casilla n viniendo **o bien de la casilla n-1 o bien de la casilla n-2**:

  - Todo camino que llega a la casilla n procedente de la casilla n-1, se crea a partir del camino que llega a n-1 añadiéndole el movimiento 1 que nos permite movernos de n-1 a n.

  - Todo camino que llega a la casilla n procedente de la casilla n-2, se crea a partir del camino que llega a n-2 añadiéndole el movimiento 2 que nos permite movernos de n-2 a n.

  El resultado del problema inicial sería:

  $$s = s1´ + s2´ \text{ o dicho de otro modo}$$

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

  Existen dos casos base:

  - que la casilla objetivo sea la casilla 1, para la que existe una única forma de llegar a ella, utilizando una vez el movimiento 1.

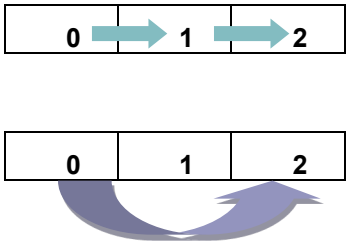
  - que la casilla objetivo sea la casilla 2, para la que existen dos formas de llegar a ella: o bien, dos movimientos del tipo 1; o bien, un movimiento del tipo 2.

Ejemplo ilustrativo.-

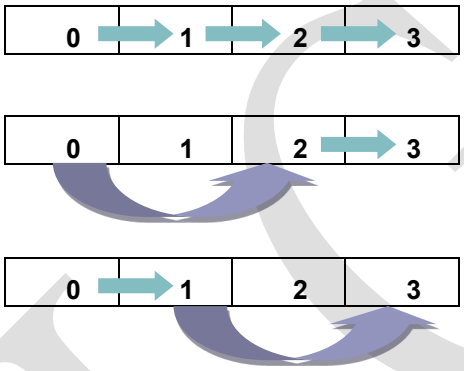
Queremos determinar la solución para RAYUELA(4).

Seguiremos el razonamiento recursivo explicado anteriormente, por tanto, supongamos conocido el número de caminos diferentes para llegar a las casillas 2 y 3, siendo sus valores asociados 2 y 3 respectivamente.

Vamos a mostrar a continuación los dos caminos que nos llevan de la casilla 0 a la casilla2.-



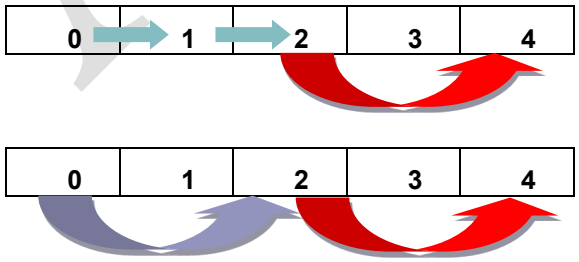
Vamos a mostrar a continuación los tres caminos que nos llevan de la casilla 0 a la casilla 3.-



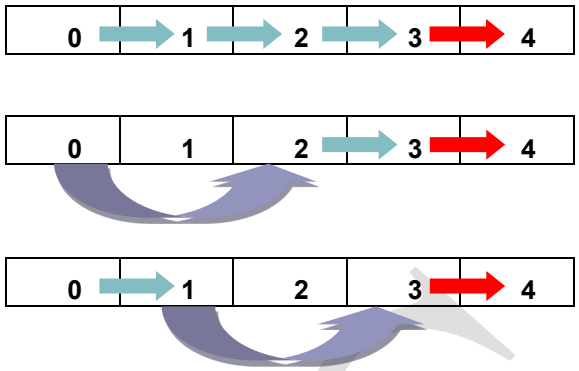
Por tanto, si a cada uno de los 5 caminos mostrados anteriormente le añadimos el movimiento correspondiente que nos sitúa en la casilla objetivo, la casilla 4, obtendremos los caminos que nos llevan de la casilla 0 a la casilla 4.

En el caso de los caminos que desembocan en la casilla 2 debemos incorporarles el movimiento tipo 2 para situarnos directamente en la casilla 4. Mientras que a los movimientos que acaban en la casilla 3 debemos incorporarles el movimiento tipo 1 para situarnos directamente en la casilla 4.

En primer lugar, mostramos los caminos que proceden de la casilla 2 y que nos conducen a la casilla 4. En rojo aparece señalado el último movimiento, el que nos conduce de la casilla 2 a la casilla 4.



En segundo lugar, mostramos los movimientos que proceden de la casilla 3 y que nos conducen a la casilla 4. En rojo aparece señalado el último movimiento, el que nos conduce de la casilla 3 a la casilla 4.



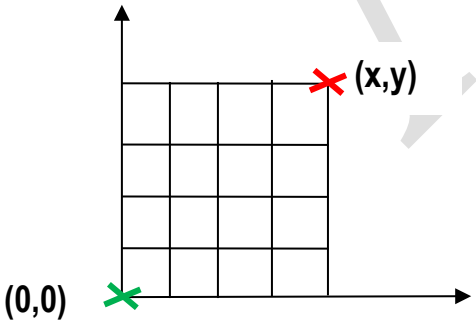
- **Algoritmo en pseudocódigo:**

```
Q ≡ { n ≥ 1 }
función RAYUELA ( n : entero) retorna ( s : entero )
        si n ≤ 2 entonces retorna n
        sino retorna RAYUELA (n-1) + RAYUELA (n-2)
        fsi
ffunción
R ≡ { s = número de caminos diferentes desde la casilla 0 hasta la casilla n,
          construidos éstos únicamente con saltos (movimientos) de longitud 1 ó 2 }
```

PLANO

Enunciado del problema.-

Dado un plano n x n y dado un punto (x,y), con x>0 ∨ y>0, en el plano, se desea saber cuántos caminos diferentes pueden llevarnos desde el origen (0,0) al punto (x,y) teniendo en cuenta que no se admiten retrocesos en el desplazamiento. Los únicos movimientos permitidos son a la derecha y hacia arriba en vertical, siendo dichos movimientos de longitud 1.



SOLUCIÓN.-

Dada la semejanza de este ejercicio con el anterior, indicamos brevemente el análisis por casos antes de mostrar el algoritmo en pseudocódigo.

- **Análisis por casos:**

  El número de caminos diferentes que llevan al punto (x,y) se divide en dos conjuntos DISJUNTOS: se puede llegar al punto (x,y) viniendo **o bien del punto (x-1,y) o bien del punto (x,y-1):**

  - Todo camino que llega al punto (x,y) procedente del punto (x-1,y), se crea a partir del camino que llega a (x-1,y) añadiéndole el movimiento hacia la derecha de (x-1,y) a (x,y).

  - Todo camino que llega al punto (x,y) procedente del punto (x,y-1), se crea a partir del camino que llega a (x,y-1) añadiéndole el movimiento hacia arriba de (x,y-1) a (x,y).

  El resultado del problema inicial sería:

  $$PLANO(x,y) = PLANO(x-1,y) + PLANO(x,y-1)$$

  Existen dos casos base:

  - que el punto objetivo sea del tipo (x,0) donde x>0, en este caso existe un único camino para alcanzar el punto (x,0) que es haciendo uso de movimientos hacia la derecha.
  - que el punto objetivo sea del tipo (0,y) donde y>0, en este caso existe un único camino para alcanzar el punto (0,y) que es haciendo uso de movimientos hacia arriba.

- **Algoritmo en pseudocódigo:**

$Q \equiv \{ x > 0 \vee y > 0 \} \equiv \{ (x \geq 0 \wedge y > 0) \vee (x > 0 \wedge y \geq 0) \}$
funcion PLANO (x, y : enteros) retorna (e : entero)
si ($x > 0 \wedge y = 0) \vee (x = 0 \wedge y > 0)$ entonces retorna 1
sino retorna PLANO(x-1, y) + PLANO(x, y-1)
fsi
ffunción
R ≡ { e = número de caminos diferentes en el plano desde el punto (0,0) hasta el punto (x,y) construidos únicamente con arcos de longitud 1 orientados hacia la derecha o hacia arriba en vertical, nunca en diagonal }

TABLEROS

Enunciado del problema.-

Calcular el número de formas diferentes de colocar un tablero de m x m sobre otro de n x n donde $n \geq m$, $n > 0$ y $m > 0$.

SOLUCIÓN 1.-

- **Especificación formal:**

  $Q \equiv \{ n \geq m \wedge n > 0 \wedge m > 0 \}$
  función TABLEROS (n, m:entero) retorna (f:entero)
  $R \equiv \{ f = (n-m+1)^2 \}$

- **Tipo de inducción: El dominio de nuestra función es IN x IN.**

  Queremos convertir el conjunto IN x IN de las parejas de naturales en un preorden. Hay muchas formas, tantas como aplicaciones f : IN x IN →IN podamos construir. Ejemplos:

  $(n', m') \leq_1 (n,m) \stackrel{def}{=} n' \leq n$

  $(n', m') \leq_2 (n,m) \stackrel{def}{=} m' \leq m$

  $(n', m') \leq_3 (n,m) \stackrel{def}{=} (n' + m') \leq (n + m)$

  En nuestro caso vamos a optar por la primera, esto es, desde un punto de vista formal, a cada pareja de valores (n,m) le asignaremos el natural n y aplicaremos inducción débil en base al orden establecido por los naturales asignados. De ese modo, nuestro objetivo es calcular TABLEROS(n,m) y para ello suponemos conocida la solución del subproblema TABLEROS(n-1,m), siendo este último problema menor que el problema original en base al preorden establecido. Aplicaremos por tanto la inducción sobre el primer parámetro, es decir, sobre n.

- **Análisis por casos:**

  Tenemos que resolver TABLEROS(n,m), esto es, tenemos que solucionar el problema para el par de valores (n,m). Su valor asociado lo denominamos f.

  Supongamos conocida la solución del problema sucesor TABLEROS(n-1,m), esto es, suponemos conocida la resolución del subproblema: “numero de formas de colocar un tablero m x m sobre uno de (n-1) x (n-1)”. A su valor asociado lo denominamos f´.

La solución del problema inicial TABLEROS(n,m) requiere sumar a f´ el número de colocaciones diferentes que nos proporcionan la fila y la columna n. Este número es n-m+1 para el caso de la fila n y el mismo valor para el caso de la columna n-ésima. Si bien hay que tener en consideración que hay una ubicación que se repite en ambos casos, siendo ésta la de la esquina inferior derecha. Por tanto el número total de colocaciones diferentes que nos proporcionan la fila y la columna n es: 2(n-m+1)-1.

$$f = f' + 2(n-m+1)-1, \text{ o dicho de otro modo}$$

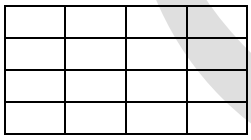
$$TABLEROS(n,m) = TABLEROS(n-1,m) + 2(n-m+1)-1$$

El caso base corresponde al subproblema en el que n=m, en ese caso sólo existe una forma de colocar el tablero m x m sobre el de n x n.

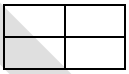
Ejemplo ilustrativo.-

Queremos determinar la solución para TABLEROS(4,2).

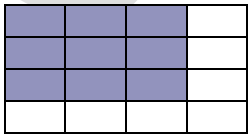
Sea el tablero 4 x 4



Y este es el tablero 2 x 2



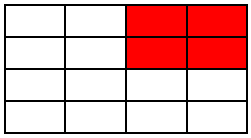
Seguiremos el razonamiento recursivo explicado anteriormente. Por tanto, supongamos conocido el número de formas diferentes de colocar el tablero 2 x 2 sobre un tablero 3 x 3. Esto quiere decir que conocemos el número de formas diferentes de colocar el tablero 2 x 2 sobre la zona coloreada del tablero 4 x 4 que figura a continuación:



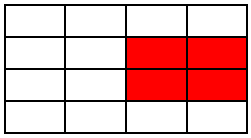
Por lo que para calcular TABLEROS(4,2) sabiendo la solución para TABLEROS(3,2), únicamente nos quedar sumar el número de formas diferentes que tenemos de colocar el tablero 2 x 2 a lo largo de la fila 4 y de la columna 4. Tanto en un caso como en otro se ha de tener en cuenta tanto el largo de la fila/columna (4) como la longitud del tablero que se coloca encima (2) ya que la relación entre dichos números determina el número de colocaciones, siendo éstas, 4-2+1. En términos generales, n-m+1.

Veamos a través de representaciones gráficas las 3 colocaciones diferentes del tablero 2 x 2 a lo largo de la columna 4:

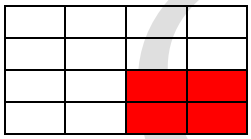
Colocación 1 en la Columna 4



Colocación 2 en la Columna 4



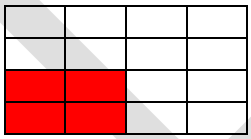
Colocación 3 en la Columna 4



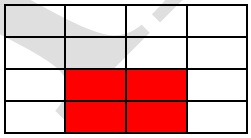
En conclusión, 4-2+1=3 colocaciones diferentes a lo largo de la Columna 4.

Ahora veamos a través de representaciones gráficas las 3 colocaciones diferentes del tablero 2 x 2 a lo largo de la fila 4:

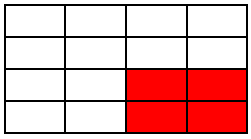
Colocación 1 en la Fila 4



Colocación 2 en la Fila 4



Colocación 3 en la Fila 4



En conclusión, 4-2+1=3 colocaciones diferentes a lo largo de la Fila 4.

Podemos comprobar cómo las colocaciones 3 de la Fila 4 y de la Columna 4 coinciden, por lo que debemos contabilizar esta colocación una única vez. Por tanto,

$$TABLEROS(4,2) = TABLEROS(3,2) + 2\ (4-2+1) - 1$$

- **Algoritmo en pseudocódigo:**

```
Q ≡ {  n ≥ m  ∧  n > 0  ∧  m > 0  }
función TABLEROS ( n, m:entero ) retorna ( f:entero )
        caso
                n = m → 1
                n > m → TABLEROS (n-1, m) + 2(n-m+1)-1
        fcaso
ffunción
R ≡ { f = (n-m+1)^2 }
```

SOLUCIÓN 2.-

- **Especificación formal:**

```
Q ≡ {  n ≥ m  ∧  n > 0  ∧  m > 0  }
función TABLEROS ( n, m:entero ) retorna ( f:entero )
R ≡ { f = (n-m+1)^2 }
```

- **Tipo de inducción:**

Nuestro objetivo es calcular TABLEROS(n,m) y para ello suponemos conocida la solución del subproblema TABLEROS(n-1,m-1), siendo este último problema menor que el problema original en base al preorden establecido. Aplicaremos por tanto la inducción sobre los parámetros de la función, es decir, sobre n y m.

- **Análisis por casos:**

Tenemos que resolver TABLEROS(n,m), esto es, tenemos que solucionar el problema para el par de valores (n,m). Su valor asociado lo denominamos f.

Supongamos conocida la solución del problema sucesor TABLEROS(n-1,m-1), esto es, suponemos conocida la resolución del subproblema: "numero de formas de colocar un tablero (m-1) x (m-1) sobre uno de (n-1) x (n-1)". A su valor asociado lo denominamos f´.

La solución del problema inicial TABLEROS(n,m) no requiere sumar un número adicional de colocaciones a f´ ya que la distancia entre n-1 y m-1 es la misma que entre n y m, por lo que el número de colocaciones es la misma en ambas situaciones. Por tanto,

$$f = f' \quad \text{o dicho de otro modo}$$

$$\text{TABLEROS}(n,m) = \text{TABLEROS}(n-1,m-1)$$

El caso base corresponde al subproblema en el que m=1, en ese caso el número de formas de colocar un tablero 1x1 sobre uno de nxn es el número de casillas de tamaño 1x1 del tablero nxn, es decir, n^2.

- **Algoritmo en pseudocódigo:**

```
Q ≡ { n ≥ m ∧ n > 0 ∧ m > 0 }
función TABLEROS ( n, m:entero ) retorna ( f : entero )
 caso
        m = 1 → n^2
        m > 1 → TABLEROS (n-1, m-1)
 fcaso
ffunción
R ≡ { f = (n-m+1)^2 }
```