# PUZZLE

# TROMINO

## Práctica "Divide y Vencerás"

Un tromino es una figura geométrica compuesta por 3 cuadros de tamaño $1 \times 1$ en forma de $L$. Sobre una retícula cuadrada de $n \times n$ (siendo $n = 2^k$) se dispone un cuadrado marcado (en negro, por ejemplo) de tamaño $1 \times 1$ y el resto vacío. El problema consiste en rellenar el resto de la cuadrícula de trominos sin solaparlos y cubriéndola totalmente, exceptuando el cuadrado mencionado. Aunque en una primera aproximación puede pensarse que se trata de un problema resoluble únicamente mediante exploración ciega y la técnica de vuelta atrás, hay una manera de realizarlo mediante *divide y vencerás* si se cumple la condición de que $n$ sea potencia de 2.

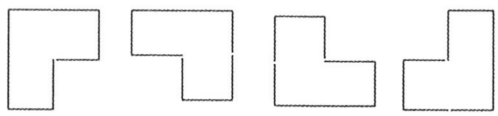

Figura 4.1: *Rotaciones de un tromino*

De esta manera, supongamos que tenemos que resolver el problema para un tamaño $8 \times 8$. Partimos de una cuadrícula vacía y con una de las casillas señalada en negro. Esta casilla no podrá usarse ni solaparse con trominos. El resto de las casillas deben rellenarse con trominos en cualquiera de las orientaciones posibles, como se muestra en la figura 4.1. Se supone que se dispone de un numero ilimitado de ellos (no hay limitaciones respecto al número de veces que se usa cada orientación).

Para abordar el problema partimos de la demostración de que es posible dividir el resto de las cuadrículas en grupos de 3, es decir, que $n^2 - 1$ es múltiplo de 3, con $n = 2^k$. Vemos que para $k = 1$ se cumple. Por inducción, suponemos cierto que $2^{2k} - 1$ es múltiplo de 3, es decir que$2^{2k} - 1 = 3p$ con $p$ un valor natural. Veamos ahora el paso de inducción:

$$2^{2(k+1)} - 1 = 2^{2k+2} - 1 = 2^2 2^{2k} - 1 = 4(3p+1) - 1 = 12p + 3 = 3(4p-1)$$

Pasamos a razonar ahora sobre el algoritmo. La manera de resolver el problema es dividir el cuadrado en 4 cuadrantes. Aquel que tiene la casilla negra (casilla marcada) queda

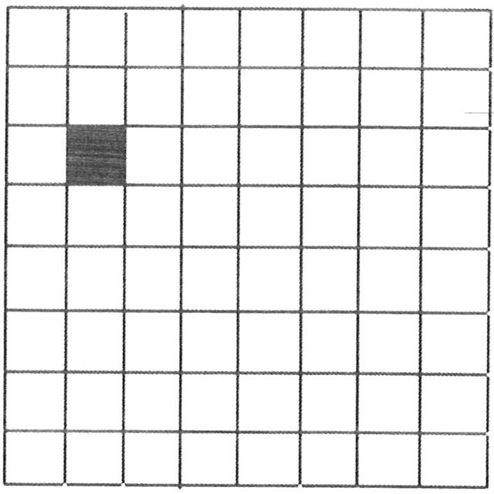

Figura 4.2: *Retículo con casilla en negro*

como está. A continuación ubicamos un tromino en el centro del tablero de manera que esté orientado hacia el cuadrante con la casilla negra. Cada uno de los 3 cuadrantes que intersecan con el tromino usarán como casilla marcada la intersección con el mismo (figura 4.3). Seguidamente para cada uno de los cuadrantes, se llama recursivamente a la función.

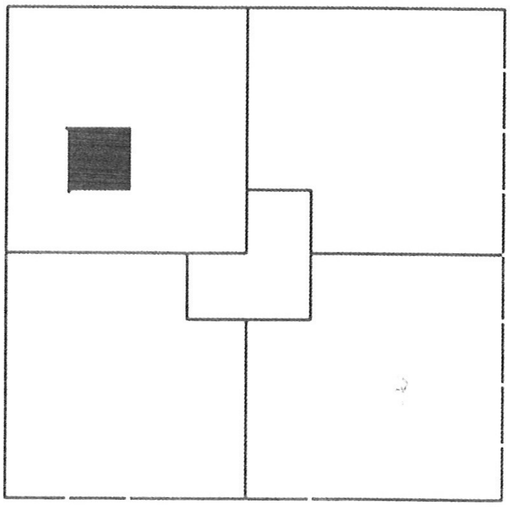

Figura 4.3: *Proceso de resolución en primera recursión*

Los elementos del esquema aplicados al problema son los siguientes:

**Caso trivial:** para el caso de que $n = 2$ podremos dar una solución si una de las casillas es la casilla en negro o está *marcada*. En este caso elegimos una de las rotaciones posibles del tromino y lo colocamos completando la retícula de $2 \times 2$.

**Descomponer:** los cuadrados de tamaño $2^k \times 2^k$ se descomponen en 4 cuadrículas $2^{k-1} \times 2^{k-1}$. Cada uno de ellos tendrá una casilla *marcada*. Una de estas casillas será la casilla en negro inicial. Las otras 3 casillas marcadas serán las 3 casillas del tromino colocado en el centro del tablero, como indica la figura 4.3

**Combinar:** se recogen en el cuadrado solución los triminos ubicados en cada uno de los cuadrantes.

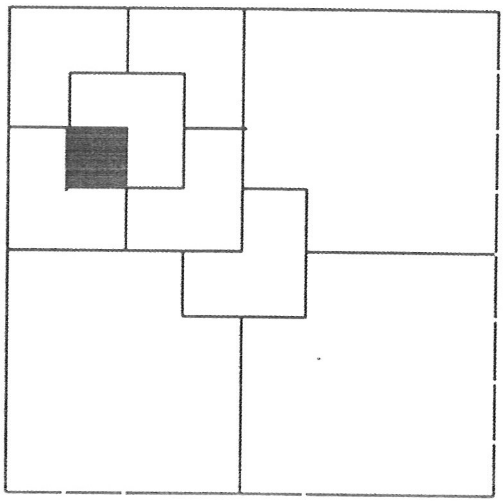

Figura 4.4: *Proceso de resolución en segunda recursión*

Consideremos los parámetros del algoritmo, que debe contener un tablero $T$, el tamaño del mismo, $n$ y la posición de la casilla *marcada*, que denominaremos $m$. El algoritmo queda como sigue:

**fun** Tromino(T:matriz [1..n,1..n] de natural,n,m:natural)
  **var**
    $T_1, T_2, T_3, T_4$ :matriz [1..n,1..n] de natural
  **fvar**

  **si** $n = 2$ **entonces**
    T $\leftarrow$ colocaTromino(T,m)

```
        dev T
    sino
        m' ← esquina cuadrante con casilla negra
        colocaTromino(T, m')
        T1, T2, T3, T4 ← dividir T en 4 cuadrantes
        tromino(T1, n/2, m1)
        tromino(T2, n/2, m2)
        tromino(T3, n/2, m3)
        tromino(T4, n/2, m4 )
    fsi
    T ← combinar(T1, T2, T3, T4)
    dev T
ffun
```

El coste del algoritmo se calcula a partir de la expresión de $T(n) = 4T(n/2) + c$, con $b = 2$, $a = 4$ y $k = 0$. Se suponen unitarias la descomposición de $T$ en subcuadrantes y la superposición de éstos en $T$. De no ser así, podría formularse el algoritmo parametrizando cuadrantes directamente en la variable $T$. El resultado de la ecuación cuando $a > b^k$ es $\Theta(n^{\log_b a})$, es decir $\Theta(n^2)$ u orden cuadrático.

textualmente:

"parametrizando cuadrantes en la variable T"

$\Downarrow$

<u>una sola matriz cuadrada</u>