Resolución por Expansión de Merge Sort

La recurrencia del algoritmo es:

$$T(n) = 2T\left(\frac{n}{2}\right) + cn$$

donde c es una constante positiva.

Expandimos varias veces:

$$\begin{aligned}
T(n) &= 2T\left(\frac{n}{2}\right) + cn \\
&= 2\left[2T\left(\frac{n}{4}\right) + c\frac{n}{2}\right] + cn \\
&= 4T\left(\frac{n}{4}\right) + 2c\frac{n}{2} + cn \\
&= 4T\left(\frac{n}{4}\right) + cn + cn \\
&= 4T\left(\frac{n}{4}\right) + 2cn \\
&= 4\left[2T\left(\frac{n}{8}\right) + c\frac{n}{4}\right] + 2cn \\
&= 8T\left(\frac{n}{8}\right) + 4c\frac{n}{4} + 2cn \\
&= 8T\left(\frac{n}{8}\right) + cn + 2cn \\
&= 8T\left(\frac{n}{8}\right) + 3cn \\
&= 8\left[2T\left(\frac{n}{16}\right) + c\frac{n}{8}\right] + 3cn \\
&= 16T\left(\frac{n}{16}\right) + 8c\frac{n}{8} + 3cn \\
&= 16T\left(\frac{n}{16}\right) + cn + 3cn \\
&= 16T\left(\frac{n}{16}\right) + 4cn \\
&\vdots
\end{aligned}$$

Después de k iteraciones:

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + kcn$$

El proceso termina cuando $\frac{n}{2^k} = 1 \implies k = \log_2 n$:

$$T(n) = nT(1) + cn \log_2 n$$

$$T(n) = \Theta(n \log n)$$

Análisis de Casos

- **Mejor caso:**
  Aunque el arreglo esté ya ordenado, Merge Sort siempre divide y fusiona todas las sublistas, por lo que el costo es igual:

  $$T_{\text{mejor}}(n) = \Theta(n \log n)$$

- **Caso promedio:**
  El trabajo realizado es el mismo para cualquier orden de entrada, por lo que:

  $$T_{\text{promedio}}(n) = \Theta(n \log n)$$

- **Peor caso:**
  Incluso si el arreglo está en el peor orden posible, el algoritmo sigue haciendo el mismo número de divisiones y fusiones:

  $$T_{\text{peor}}(n) = \Theta(n \log n)$$