ANALISIS DE COMPLEJIDAD PARA EL ALGORITMO MERGESORT

El algoritmo `MERGESORT` funciona en dos fases principales:

1. **División recursiva**: El array se divide repetidamente en subarrays hasta que cada subarray tiene un solo elemento.
2. **Mezcla (merge)**: Luego, se combinan los subarrays de manera ordenada.

Complejidad del Algoritmo:

1. **División recursiva**:
   - En cada nivel de recursión, el array se divide a la mitad. Esto genera un árbol de recursión cuya altura es logarítmica, es decir, $\log_2 n$
   - Cada nivel del árbol de recursión procesa n elementos en total (considerando todas las divisiones en ese nivel).
2. **Mezcla (merge)**:
   - El proceso de mezcla compara y combina elementos de subarrays. En cada nivel de recursión, la operación de mezcla toma tiempo proporcional a n porque todos los elementos deben ser procesados.
   - Esto se repite a lo largo de los $\log_2 n$ niveles.

Complejidades:

- **O(n log n)**: `MERGESORT` tiene una complejidad temporal en el peor caso $O(n \log n)$. Esto se debe a que el proceso de dividir el array toma $\log_2 n$ niveles, y en cada nivel se realizan n operaciones para la mezcla. Así que, en total, el peor caso es $O(n \log n)$.
- **Ω(n log n)**: `MERGESORT` también tiene una complejidad temporal en el mejor caso $\Omega(n \log n)$. Incluso si el array ya está ordenado, el algoritmo realiza las mismas divisiones y combinaciones. No hay un atajo que lo haga más rápido, por lo que el mejor caso también es $n \log n$.
- **Θ(n log n)**: `MERGESORT` es un algoritmo con complejidad $\Theta(n \log n)$, lo que significa que tanto el mejor como el peor caso tienen la misma complejidad temporal. Esto indica que la función de tiempo está limitada de manera asintótica tanto superior como inferiormente por $n \log n$.