MergeSort

A complexidade é a mesma independentemente da distribuição dos dados porque o algoritmo divide o array e o mescla de forma eficiente e previsível. A análise da complexidade é a mesma para todos os casos.

A recorrência

$T(n)=2 \cdot T(n/2)+O(n)$

- $2 \cdot T(2/n)$ é o custo de ordenar duas sublistas de tamanho n/2.
- O(n)é o custo de mesclar duas sublistas de tamanho n/2

Explicando:

nível 0 (Raiz):

- Custo é O(n) pra mesclar e dividir o array.

Nível 1:

- O array é dividido em duas sublistas de tamanho n/2.
- O custo total neste nível é $2 \cdot O(n/2)=O(n)$

Nível 2:

- Cada sublista é dividida em duas partes de tamanho n/4
- O custo total neste nível é $4 \cdot O(n/4)=O(n)$

Nível k:

- Existem 2^k sublistas, cada uma com tamanho n/(2^k)
- O custo total neste nível é $2^k \cdot O(n/(2^k))=O(n)$

o número de níveis da árvore de recursão é log2n, porque o array é dividido pela metade a cada nível até chegar a sublistas de tamanho 1.

$$T(n) = \sum_{k=0}^{\log_2 n} O(n)$$

Como o custo é O(n) em cada nível e existe log2n+1 níveis:

$$T(n) = (\log_2 n + 1) \cdot O(n)$$

Resumo:

Recorrência: T(n)=2*T(2/n)+O(n)

Custo Total em cada Nível: O(n)

Número Total de Níveis: log2n

Somatória Total dos Custos: T(n)=(log2n+1)*O(n)

Complexidade:O(nlogn)

LEMBRANDO: O MELHOR, MÉDIO E PIOR CASO TEM CUSTO E COMPLEXIDADE IGUAIS

$O(n \log n)$