Comparação de Complexidades de Algoritmos de Ordenação

Algoritmo	Melhor Caso	Pior Caso
Selection Sort	$O(n^2)$	$O(n^2)$
Bubble Sort	$O(n)$ (vetor já ordenado)	$O(n^2)$
Merge Sort	$O(n \log n)$	$O(n \log n)$
Quick Sort	$O(n \log n)$	$O(n^2)$ (partição desbalanceada)
Heap Sort	$O(n \log n)$	$O(n \log n)$

Table 1: Complexidade de Tempo para Algoritmos de Ordenação (Melhor e Pior Caso)

Comparação: Quick Sort vs Heap Sort

Critério	Quick Sort	Heap Sort
Complexidade (Melhor Caso)	$O(n \log n)$	$O(n \log n)$
Complexidade (Pior Caso)	$O(n^2)$ (partição desbalanceada)	$O(n \log n)$
Constante Oculta	Menor, devido ao menor número de operações por comparação	Maior, devido à manutenção do heap
Uso de Memória	Excelente localidade de cache (acesso sequencial)	Menor localidade de cache (acesso disperso)
Estabilidade	Não estável (em implementações padrão)	Não estável
Robustez	Sensível à escolha de pivô	Robusto para qualquer entrada
Praticidade	Melhor desempenho na prática para a maioria das entradas	Utilizado em casos específicos
Aplicações	Geralmente preferido em bibliotecas padrão de ordenação	Útil em sistemas embarcados e situações críticas

Table 2: Comparação entre Quick Sort e Heap Sort