Análise da complexidade de tempo (Big O) para cada algoritmo

---

1. Bubble Sort

Caso	Complexidade	Explicação
Melhor	**O(n)**	Ocorre quando o vetor já está ordenado; só é feita 1 passagem sem trocas.
Médio	**O(n²)**	Compara cada elemento com todos à frente.
Pior	**O(n²)**	Vetor invertido gera máximo de trocas.

---

2. Insertion Sort

Caso	Complexidade	Explicação
Melhor	**O(n)**	Vetor já ordenado: cada inserção faz apenas 1 comparação.
Médio	**O(n²)**	Elementos deslocados em média metade do vetor.
Pior	**O(n²)**	Vetor invertido: cada elemento deslocado até o início.

---

3. Selection Sort

Caso	Complexidade	Explicação
Melhor	**O(n²)**	Sempre procura o menor elemento ⇒ número fixo de comparações.
Médio	**O(n²)**	Independente da ordem inicial.
Pior	**O(n²)**	Também igual; não depende de trocas.

---

4. Merge Sort

Caso	Complexidade	Explicação
Melhor	**O(n log n)**	Divide o vetor ao meio e combina ordenadamente.
Médio	**O(n log n)**	Recursão equilibrada sempre.
Pior	**O(n log n)**	Mesmo no pior caso, o processo de merge é estável.

---

5. Quick Sort

Caso	Complexidade	Explicação
Melhor	**O(n log n)**	Partições bem equilibradas (pivô central).

Médio	**O(n log n)**	Em média, particiona o problema em partes razoavelmente iguais.
Pior	**O(n²)**	Pivô sempre escolhido como menor ou maior → vetores já ordenados ou invertidos.

---

6. Shell Sort

Shell Sort não tem uma análise exata para todos os tipos de sequência de gaps, mas para a sequência tradicional de gaps (n/2, n/4, ..., 1), temos:

Caso	**Complexidade aproximada**	**Explicação**
Melhor	**O(n log n)**	Para algumas sequências, o comportamento se aproxima de inserção em sublistas.
Médio	**O(n^(3/2))** ≈ $O(n^{1.5})$	Aproximação clássica da sequência de Shell.
Pior	**O(n²)**	Com certas sequências ruins, pode deteriorar.