Relatório de Análise de Algoritmos de Ordenação

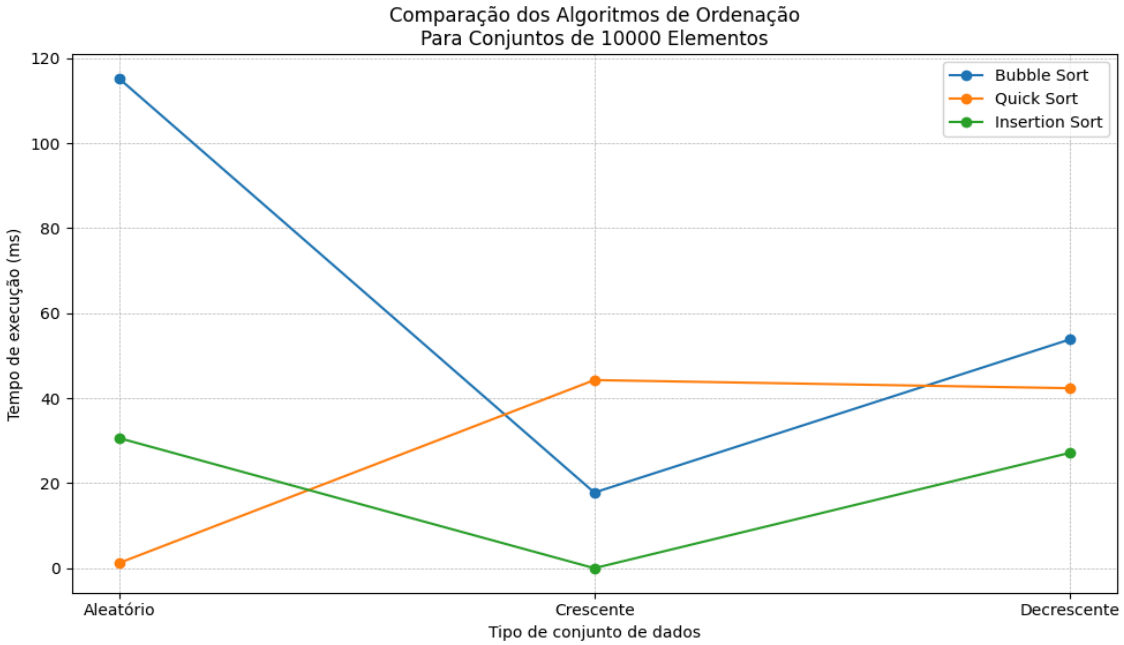
Alunos: Angelo Andrioli Netho, Eduardo Mendes Carbonera, Kaio Gonçalves Teles

Este relatório apresenta uma análise de desempenho dos algoritmos de ordenação Bubble Sort, Insertion Sort e Quick Sort aplicados a três tipos de conjuntos de dados: aleatório, ordenado crescente e ordenado decrescente. Os tempos de execução foram coletados em milissegundos (ms) e refletem o comportamento prático de cada algoritmo frente a diferentes padrões de entrada.

O objetivo é comparar o desempenho relativo entre os três algoritmos, destacando como a ordem inicial dos dados influencia diretamente o tempo total de processamento

RESULTADO DE TEMPOS DE EXECUÇÃO(10.000 ELEMENTOS)

Tipo de Conjunto de Dados	Bubble Sort	Insertion Sort	Quick Sort
Aleatório	115.20ms	30.61ms	1.24ms
Crescente	17.80ms	0.0079ms	44.27ms
Decrescente	53.80ms	27.127ms	42.36ms



Nos **dados aleatórios**, o Quick Sort foi o mais rápido porque seu funcionamento é baseado em escolher um pivô e dividir a lista em partes menores que são ordenadas separadamente. Esse processo tende a funcionar muito bem quando os dados estão misturados, pois as divisões internas costumam ficar equilibradas, permitindo que o algoritmo avance rapidamente. Já o Insertion Sort precisa inserir cada elemento na posição correta, o que demanda mais movimentos, e o Bubble Sort precisa comparar e trocar pares repetidamente, tornando-o bem mais lento nesse tipo de entrada.

Nos **dados ordenados de forma crescente**, o Insertion Sort foi o mais eficiente devido ao seu funcionamento: ele percorre a lista da esquerda para a direita, sempre comparando o elemento atual com o anterior. Se tudo já está no lugar certo, o algoritmo praticamente não faz trocas, apenas confirma a ordem e segue em frente. O Quick Sort, por outro lado, pode acabar escolhendo um pivô ruim nesse cenário, o que faz com que a divisão dos dados não seja tão eficiente. O Bubble Sort melhora um pouco nesse caso, já que as trocas são reduzidas, mas ainda assim passa repetidas vezes pela lista para garantir que tudo esteja ordenado.

Nos **dados decrescentes**, o Quick Sort novamente se destaca, pois mesmo com a ordem inversa ele consegue dividir a lista em partes menores rapidamente e reorganizá-las sem depender de deslocamentos longos. O Insertion Sort, por outro lado, sofre bastante: como a lista está totalmente invertida, cada novo elemento analisado precisa ser deslocado para o início, repetidamente. Já o Bubble Sort volta a ter desempenho ruim porque precisa realizar trocas em praticamente todos os pares da lista para colocá-los na ordem correta.