Relatório de Análise de Algoritmos de Ordenação

Giovani Yamaguchi Tortato e Matheus Garozi

Introdução

Esse relatório apresenta uma análise comparativa de desempenho entre três algoritmos de ordenação fundamentais: Bubble Sort, Insertion Sort e Quick Sort. A comparação foi feita comparando o tempo de execução (ms), usando os conjuntos de dados fornecidos que tem diferentes tamanhos e diferentes estados de ordenação, aleatório, crescente e decrescente..

Condições de teste, 100, 1.000, 10.000 elementos.

Resultados do Desempenho (ms)

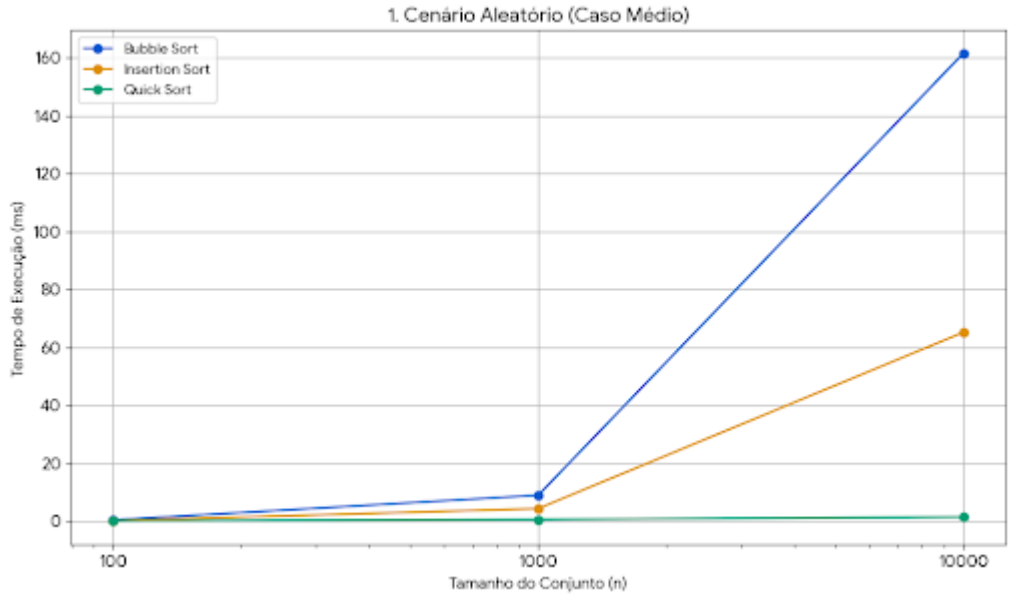
A tabela a seguir, foi o resultado dos algoritmos com as três condições dos conjuntos de dados.

Algoritmo	**Tipo**	**Tamanho (n)**	**Tempo (ms)**
Bubble Sort	Aleatório	100	0,3528
Insertion Sort	Aleatório	100	0,1023
Quick Sort	Aleatório	100	0,0466
Bubble Sort	Aleatório	1000	8,9295
Insertion Sort	Aleatório	1000	4,3274
Quick Sort	Aleatório	1000	0,5725
Bubble Sort	**Aleatório**	**10000**	**161,5234** (pior tempo de execução

			para dados Aleatórios.)
Insertion Sort	Aleatório	10000	65,2055
Quick Sort	**Aleatório**	**10000**	**1,4216** Melhor tempo de execução para dados Aleatórios.
Bubble Sort	Crescente	100	0,0021
Insertion Sort	Crescente	100	0,0603
Quick Sort	Crescente	100	0,0323
Bubble Sort	Crescente	1000	0,0010
Insertion Sort	Crescente	1000	0,0581
Quick Sort	Crescente	1000	2,2246
Bubble Sort	**Crescente**	**10000**	**0,0056** melhor tempo de execução de todos os testes.
Insertion Sort	Crescente	10000	0,1804

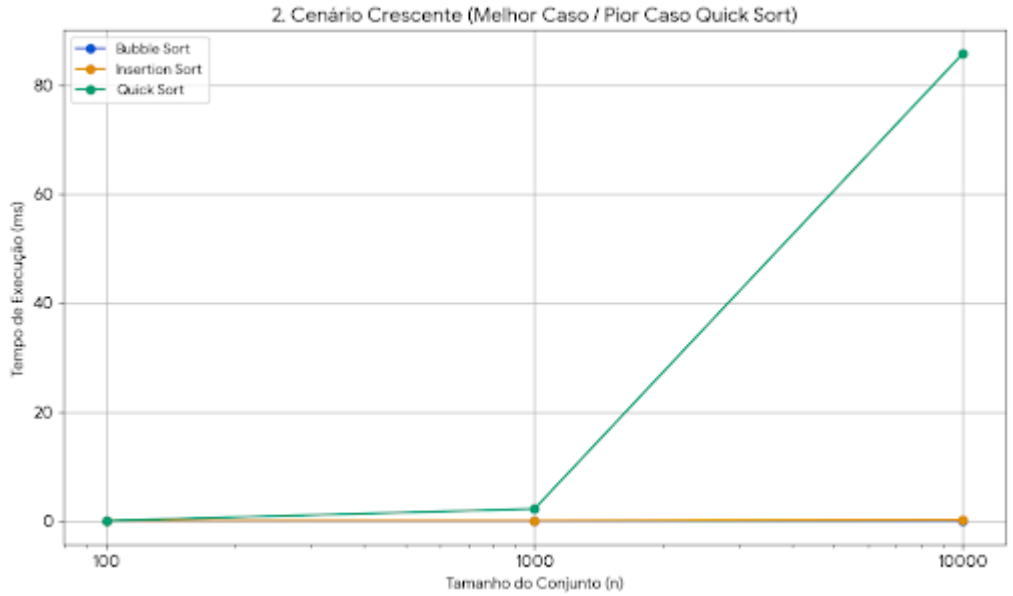
Quick Sort	**Crescente**	**10000**	**85,7250** pior tempo de execução do Quick Sort.
Bubble Sort	Decrescente	100	0,0797
Insertion Sort	Decrescente	100	0,0067
Quick Sort	Decrescente	100	0,0275
Bubble Sort	Decrescente	1000	4,0445
Insertion Sort	Decrescente	1000	0,2993
Quick Sort	Decrescente	1000	0,8542
Bubble Sort	Decrescente	10000	115,7311
Insertion Sort	Decrescente	10000	30,6906
Quick Sort	Decrescente	10000	75,6903

Cenário Aleatório



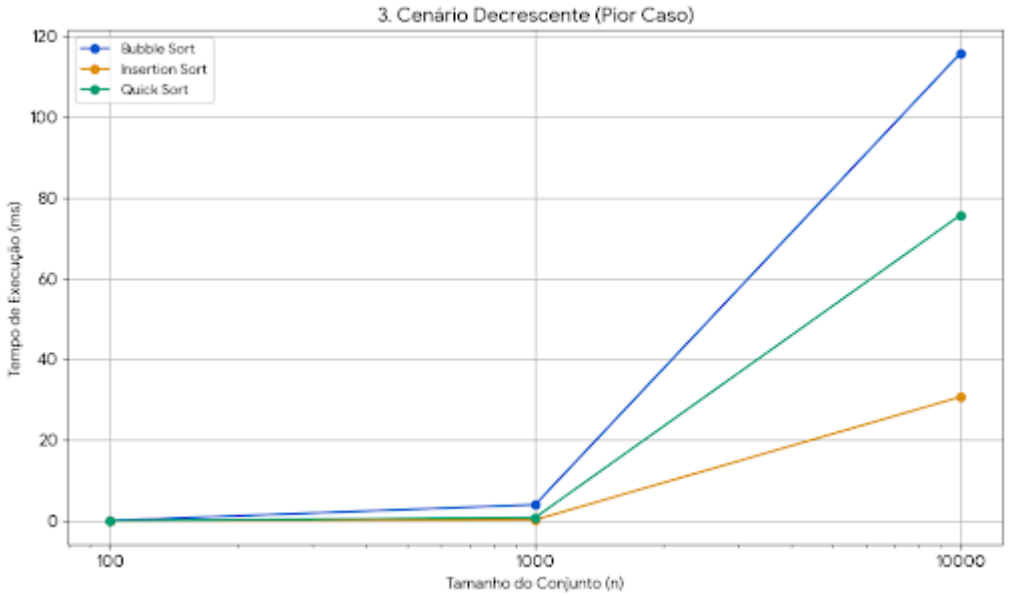
Com dados aleatórios o Quick Sort é superior em escalabilidade, manteve um tempo de execução baixo. Para n=10.000, o Quick Sort levou só 1,4216 ms, confirmando sua eficiência na média. Os algoritmos Bubble Sort com 161,5234 ms e Insertion Sort com 65,2055 ms) tiveram um crescimento de tempo maior.

Cenário Crescente



Esse cenário é o melhor caso para algoritmos com otimização de parada antecipada. O Bubble Sort, foi rápido para todos os tamanhos (0,0056 ms para n=10.000). O Quick Sort teve seu pior caso , para 10.000 elementos levou 85,7250 ms.

Cenário Decrescente



Pior cenário para os algoritmos, Bubble Sort (115,7311 ms), Insertion Sort (30,6906 ms) pois precisam realizar o número máximo de comparações e trocas. Quick Sort também sofreu uma piora de desempenho neste cenário, 75,6903 ms.

Conclusão

Quick Sort: É o algoritmo mais eficiente para conjuntos de dados grandes e desordenados, teve a melhor escalabilidade na média. Por causa da sua complexidade O(n \log n) acaba sendo a melhor escolha para a maioria das aplicações.

Insertion Sort: Muito bom para conjuntos de dados pequenos (n<100) ou para listas que já estão quase ordenadas. Como gasta pouco recurso computacional as vezes acaba sendo mais rápido que o Quick Sort.
Bubble Sort: Apesar de ter o melhor desempenho com a lista já ordenada, seu desempenho na média é pior, sendo muito complicado trabalhar com grandes conjuntos de dados.

Fontes: Utilizado IA (Gemini) para ajudar a gerar os gráficos e tirar algumas dúvidas programando o código.