RPEC - Relatório TDE3 RA04

Daniela Lima & Rafael Galo

October 2023

1 Introduction

Este trabalho tem como objetivo a implementação e comparação do desempenho de diferentes algoritmos de ordenação. Os algoritmos escolhidos foram: Insertion Sort, Bubble Sort, Merge Sort e Quick Sort. A avaliação do desempenho será feita considerando três métricas principais: a) Tempo de execução, b) Número de trocas e c) Número de iterações.

Para realizar a comparação de desempenho, utilizaremos vetores de inteiros preenchidos aleatoriamente com tamanhos de 50, 500, 1000, 5000 e 10000 elementos. Para garantir a robustez dos resultados, serão realizadas, no mínimo, 5 rodadas de execução para cada tamanho de vetor. A média dos resultados obtidos em múltiplas execuções será apresentada e analisada, permitindo uma avaliação mais precisa do comportamento de cada algoritmo de ordenação em diferentes cenários.

A escolha dos algoritmos de ordenação e a abordagem de comparação detalhada visam proporcionar insights valiosos sobre o desempenho relativo desses algoritmos em situações do mundo real. Esperamos que este estudo contribua para uma melhor compreensão das vantagens e desvantagens de cada algoritmo e auxilie na seleção do mais apropriado para diferentes necessidades de ordenação.

GitHub

2 Quick Sort

O Quick Sort é outro algoritmo de ordenação de divisão e conquista que seleciona um elemento como "pivô" e reorganiza os elementos de forma que todos os elementos menores que o pivô estejam à esquerda e todos os elementos maiores à direita. Esse processo é repetido para as sub-listas resultantes até que todo o conjunto de dados esteja ordenado. O Quick Sort é eficiente em média, com complexidade O(n log n), mas pode ter um desempenho ruim no pior caso. No entanto, é amplamente utilizado devido à sua eficiência média e facilidade de implementação.

2.1 Teste com 50 elementos

Tempo de execução	8 ms
Número de trocas	1.195.548
Número de iterações	7.795.344

2.2 Teste com 500 elementos

Tempo de execução	8 ms
Número de trocas	3.189.943
Número de iterações	20.808.644

2.3 Teste com 1000 elementos

Tempo de execução	8 ms
Número de trocas	5.170.566
Número de iterações	34.142.473

2.4 Teste com 5000 elementos

Tempo de execução	8 ms
Número de trocas	7.150.330
Número de iterações	47.559.410

2.5 Teste com 10000 elementos

Tempo de execução	8 ms
Número de trocas	9.143.545
Número de iterações	60.816.314

3 Merge Sort

O Merge Sort é um algoritmo de ordenação eficiente que segue uma abordagem de divisão e conquista. Ele divide o conjunto de dados em pequenos subconjuntos, os ordena e, em seguida, mescla esses subconjuntos ordenados para obter a lista final ordenada. O Merge Sort possui uma complexidade média de O(n log n) e é eficiente para grandes conjuntos de dados.

3.1 Teste com 50 elementos

Tempo de execução	18 ms
Número de trocas	10.232.052
Número de iterações	10.232.052

3.2 Teste com 500 elementos

Tempo de execução	18 ms
Número de trocas	27.285.472
Número de iterações	27.285.472

3.3 Teste com 1000 elementos

Tempo de execução	13 ms
Número de trocas	44.338.892
Número de iterações	44.338.892

3.4 Teste com 5000 elementos

Tempo de execução	13 ms
Número de trocas	61.392.312
Número de iterações	61.392.312

3.5 Teste com 10000 elementos

Tempo de execução	11 ms
Número de trocas	78.445.732
Número de iterações	78.445.732

4 Bubble Sort

O Bubble Sort é outro algoritmo de ordenação simples que compara repetidamente pares de elementos adjacentes e os troca se estiverem na ordem errada. Esse processo de "bolha" continua até que todo o conjunto de dados esteja ordenado. O Bubble Sort é fácil de entender, mas não é eficiente para grandes conjuntos de dados, com uma complexidade média de O(n^2).

4.1 Teste com 50 elementos

Tempo de execução	14.051 ms
Número de trocas	213.503.340
Número de iterações	181.852.760

4.2 Teste com 500 elementos

Tempo de execução	15.206 ms
Número de trocas	228.617.746
Número de iterações	232.152.440

4.3 Teste com 1000 elementos

Tempo de execução	11.275 ms
Número de trocas	254.607.420
Número de iterações	212.762.500

4.4 Teste com 5000 elementos

Tempo de execução	12.274 ms
Número de trocas	263.794.687
Número de iterações	201.278.373

4.5 Teste com 10000 elementos

Tempo de execução	17.222 ms
Número de trocas	234.117.438
Número de iterações	251.514.233

5 Insertion Sort

O Insertion Sort é um algoritmo de ordenação simples e eficiente que funciona percorrendo uma lista de elementos e inserindo cada elemento na posição correta, deslocando os elementos maiores conforme necessário. Ele é especialmente eficaz para pequenos conjuntos de dados e já parcialmente ordenados. A complexidade média do caso é $O(n^2)$, tornando-o menos eficiente para grandes conjuntos de dados.

5.1 Teste com 50 elementos

Tempo de execução	1.834 ms
Número de trocas	213.646.170
Número de iterações	213.646.170

5.2 Teste com 500 elementos

Tempo de execução	1.278 ms
Número de trocas	228.862.599
Número de iterações	228.862.599

5.3 Teste com 1000 elementos

Tempo de execução	1.322 ms
Número de trocas	254.954.295
Número de iterações	254.954.295

5.4 Teste com 5000 elementos

Tempo de execução	2.815 ms
Número de trocas	263.468.216
Número de iterações	263.468.216

5.5 Teste com 10000 elementos

Tempo de execução	3.026 ms
Número de trocas	233.688.946
Número de iterações	233.688.946

6 Conclusão

A tabela a seguir resume os resultados dos testes de desempenho dos algoritmos de ordenação para diferentes quantidades de elementos:

Algoritmo	Qtd. Elementos	Tempo Exe. (ms)	Num. de Trocas	Num. de Iterações
Quick Sort	50	8	1.195.548	7.795.344
	500	8	3.189.943	20.808.644
	1000	8	5.170.566	34.142.473
	5000	8	7.150.330	47.559.410
	10000	8	9.143.545	60.816.314
Merge Sort	50	18	10.232.052	10.232.052
	500	18	27.285.472	27.285.472
	1000	13	44.338.892	44.338.892
	5000	13	61.392.312	61.392.312
	10000	11	78.445.732	78.445.732
Bubble Sort	50	14.051	213.503.340	181.852.760
	500	15.206	228.617.746	232.152.440
	1000	11.275	254.607.420	212.762.500
	5000	12.274	263.794.687	201.278.373
	10000	17.222	234.117.438	251.514.233
Insertion Sort	50	1.834	213.646.170	213.646.170
	500	1.278	228.862.599	228.862.599
	1000	1.322	254.954.295	254.954.295
	5000	2.815	263.468.216	263.468.216
	10000	3.026	233.688.946	233.688.946

Com base nos resultados, podemos concluir que os algoritmos de ordenação Quick Sort e Merge Sort geralmente apresentam melhor desempenho em termos de tempo de execução, número de trocas e número de iterações em comparação com Bubble Sort e Insertion Sort. No entanto, a escolha do algoritmo de ordenação mais adequado depende das características específicas do problema e do tamanho dos dados a serem ordenados.