Desempenho dos Algoritmos de Ordenação de Vetores

Michel T. S. Brito NUSP: 11257755

Introdução

O intuito deste relatório é analisar o desempenho de diferentes algoritmos de ordenação com diferentes tamanhos de listas (tamanho da forma 10^p em que p varia de 1 a 6).

Os algoritmos usados neste programa foram BubbleSort, IsertionSort, MergeSort e QuickSort.

Foram usadas tabelas ao invés de gráficos, pois a partir de p = 5 obteve-se números muito discrepantes entre si, assim com um gráfico não seria possível ter uma melhor noção da variação entre os tempos de ordenação dos algoritmos.

Testes

Foram feitos um total de 20 testes nas listas em que p varia de 1 a 4, 10 testes nas listas em que p vale 5 e 3 testes nas listas em que p vale 6 (aqui o número de testes e reduzido por causa da demora da execução dos testes, tornando-se inviável a realização de muitos). Após esses testes foi tirada uma média, tal valor é usado nas tabelas abaixo.

Obs.: O algoritmo QuickSort apresentou problemas nos casos de vetores quase ordenados e vetores em ordem inversa, no primeiro caso a partir de p = 5 ocorreu Stack Overflow, já no segundo caso quando p = 5 algumas das simulações foram bem sucedidas, porém na enorme maioria delas ouve o mesmo problema e pôr fim quando p = 6, nenhuma simulação foi bem sucedida.

Lista Quase Ordenada

Eficiência dos Algoritmos Lista Int com 10^p Números (1 ≤ p ≤ 6) (Tempo em ms)

	1	2	3	4	5	6
Bubble	0	0	4	30	2159	380152
Insertion	0	0	0	0	3	19
Merge	0	0	0	2	43	121
Quick	0	0	2	49	Stack Overflow	Stack Overflow

Eficiência dos Algoritmos Lista Float com 10^p Números (1 ≤ p ≤ 6) (Tempo em ms)

	1	2	3	4	5	6
Bubble	0	0	5	40	3620	414421
Insertion	0	0	0	1	3	20
Merge	0	0	0	2	40	110
Quick	0	0	1	31	Stack Overflow	Stack Overflow

Eficiência dos Algoritmos Lista Double com 10^p Números (1 ≤ p ≤ 6) (Tempo em ms)

	1	2	3	4	5	6
Bubble	0	0	4	40	3766	603141
Insertion	0	0	0	1	7	28
Merge	0	0	0	2	44	131
Quick	0	0	1	21	Stack Overflow	Stack Overflow

Lista Aleatória

Eficiência dos Algoritmos Lista Int com 10^p Números (1 ≤ p ≤ 6) (Tempo em ms)

	1	2	3	4	5	6
Bubble	0	0	6	140	15705	1671182
Insertion	0	0	2	309	2958	323671
Merge	0	0	0	3	45	227
Quick	0	0	0	2	14	106

Eficiência dos Algoritmos Lista Float com 10^p Números (1 ≤ p ≤ 6) (Tempo em ms)

	1	2	3	4	5	6
Bubble	0	0	6	165	17332	1818664
Insertion	0	0	2	57	5216	554612
Merge	0	0	0	3	42	199
Quick	0	0	0	2	13	117

Eficiência dos Algoritmos Lista Double com 10^p Números (1 ≤ p ≤ 6) (Tempo em ms)

	1	2	3	4	5	6
Bubble	0	0	6	164	17556	2009551
Insertion	0	0	3	55	5126	323671
Merge	0	0	0	4	50	221
Quick	0	0	0	2	14	113

Lista Ordem Inversa

Eficiência dos Algoritmos Lista Int com 10^p Números (1 ≤ p ≤ 6) (Tempo em ms)

	1	2	3	4	5	6
Bubble	0	0	4	38	3161	354770
Insertion	0	0	0	1	7	14
Merge	0	0	0	3	40	126
Quick	0	0	1	40	Stack Overflow	Stack Overflow

Eficiência dos Algoritmos Lista Float com 10^p Números (1 ≤ p ≤ 6) (Tempo em ms)

	1	2	3	4	5	6
Bubble	0	0	4	41	3694	392129
Insertion	0	0	0	2	7	21
Merge	0	0	4	3	46	114
Quick	0	0	1	29	Stack Overflow	Stack Overflow

Eficiência dos Algoritmos Lista Double com 10^p Números (1 ≤ p ≤ 6) (Tempo em ms)

	1	2	3	4	5	6
Bubble	0	0	4	45	3822	584773
Insertion	0	0	0	2	8	22
Merge	0	0	0	4	54	120
Quick	0	0	1	22	Stack Overflow	Stack Overflow

Conclusão

Conclui-se que a partir de p = 5 BubbleSort e QuickSort se tornam muito ineficientes, um passa a demorar muito tempo para ordenar a lista e o outro ocorre Stack Overflow, para listas até 1000 números qualquer algoritmo escolhido é eficiente, destacando-se o InsertionSort e MergeSort.

Para listas de menos de 1000 elementos, independentemente do tipo de elemento (Int, Float, Double), todos os algoritmos apresentaram resultados bons, assim qualquer um deles é indicado para ordenar os elementos.

Em listas de mais de 10000 elementos obteve-se resultados interessantes, para uma sequência de números quase ordenados e em ordem inversa, o algoritmo InsertionSort se destacou pelos menores tempos de ordenação, independentemente do tipo de elemento.

Porém em para uma sequência aleatória o algoritmo QuickSort, que obteve resultados negativos em outras sequencias, desempenhou-se muito bem, obtendo os menores tempos de execução.

Conclui-se que para listas menos de 10000 elementos, qualquer algoritmo ordenará com baixos tempos. Acima disso, para listas quase ordenadas e inversas recomenda-se o uso do InsertionSort e para uma lista aleatória é preferível o uso do QuickSort.

Fontes

https://www.devmedia.com.br/algoritmos-de-ordenacao-em-java/32693

https://www.baeldung.com/java-merge-sort