Aluno: Daniel Fróes

n° usp: 10255956

Análise experimental dos métodos mais comuns de ordenação

Introdução

Com uma grande variedade de métodos de ordenação disponíveis ao programador, torna-se interessante uma comparação empírica entre eles a fim de analisar de, maneira experimental, qual método é mais eficiente para cada situação.

Desse modo, no experimento, foram analisados oito métodos de ordenação diferentes: bubble sort, bubble sort com sentinela, cocktail sort, selection sort, insertion sort, merge sort, heap sort e quick sort. Para cada um deles, foram analisados quatro casos distintos de inicialização do vetor, sendo estes, aleatório, semi ordenado, semi ordenado inversamente e com vários valores repetidos. Além disso, para cada caso de inicialização, foram ordenados cinco vetores de tamanhos diferentes (tamanho : 100, 1000, 10000, 100000, 1000000).

Os dados computados no experimento, foram a quantidade de atribuições e comparações de elementos do vetor em cada caso. A fim de garantir uma maior veracidade dos dados coletados, para cada tamanho de vetor, foram feitas 5 ordenações e, delas, foram tiradas as médias de comparações e atribuições de cada algoritmo.

A fim de comparar os resultados, será usado o conceito de ordem de grandeza, este que aproxima o valor para a potência de dez mais próxima, assim, facilitando a comparação de números muitos grandes, como os que serão analisados.

Instrução para a compilação

Abrindo o diretório contendo os códigos fontes no terminal, digite a linha a seguir:

$ make all

Digitando-a, o código será compilado e executado automaticamente.

Para selecionar o método de ordenação, basta seguir a tabela a seguir. Os resultados estarão na mesma pasta que os códigos fonte em um arquivo texto gerado chamado “results.txt”.

Método de ordenação	**Palavra chave**
Bubble sort	“bubble”
Bubble sort com sentinela	“sentinel”
Cocktail sort	“cocktail”
Selection sort	“selection”
Insert sort	“insert”
Merge sort	“merge”
Quick sort	“quick”
Heap sort	“heap”

Resultados

Vetor aleatório:

Atribuições:

Tam. vetor	Bubble	Bubble c/ sentinela	Cocktail	Insertion	Selection	Merge	Heapsort	Quicksort
10^2	20168	20168	24138	7316	5049	4032	3118	2349
10^3	2236566	2236566	3025998	751516	500499	59856	42092	30247
10^4	225242745	225242745	320618344	75140909	50004999	801696	522376	371593
10^5	22537780142	22537780142	32145668606	7513193374	5000049999	10013568	6224522	4394360
10^6	2249871186363	2249871186363	3218149889185	749963062115	500000499999	119708544	72148231	50928693

Comparações:

Tam. vetor	Bubble	Bubble c/ sentinela	Cocktail	Insertion	Selection	Merge	Heapsort	Quicksort
10^2	14850	14652	14850	7615	5236	1611	3099	1332
10^3	1498500	1488571	1498500	755498	502478	26151	50536	22779
10^4	149985000	149933623	149985000	74383537	50024977	361308	706197	319666
10^5	14999850000	14998513909	14999850000	7491739689	5000249963	4609472	9058687	4287399
10^6	1499998500000	1499974160216	1499998500000	750016292389	500002499956	56021458	110383534	49901020

Para o vetor aleatório, temos que os que mais se destacaram no número de atribuições foram o Heap sort, Merge sort e o Quick sort, todos apresentando ordens de grandeza de 10^8 para o maior vetor, tendo, desse modo uma diferença de 5 casas decimais com os piores casos, Bubble, Bubble com sentinela e o Cocktail.

Já para as comparações, o Merge sort se mostrou o mais eficaz, sendo o único com um valor na ordem de grandeza (para o maior vetor) de 10^8. Os piores caso entretanto, continuou ficando com o Bubble, Bubble com sentinela e o Cocktail, todos ficando na ordem de grandeza dos 10^{12}.

Vetor quase ordenado:

Atribuições:

Tam. vetor	Bubble	Bubble c/ sentinela	Cocktail	Insertion	Selection	Merge	Heapsort	Quicksort
10^2	1264	1264	14454	1015	5049	4032	3368	1166
10^3	12667	12667	1414416	10216	500499	59856	43975	12136
10^4	126760	126760	142147809	102247	50004999	801696	543858	122152
10^5	1263364	1263364	14143654785	1021115	5000160798	10013568	6447892	1219236
10^6	12619819	12619819	1415874674818	10206600	500000499999	119708544	74329994	12185440

Comparações:

Tam. vetor	Bubble	Bubble c/ sentinela	Cocktail	Insertion	Selection	Merge	Heapsort	Quicksort
10^2	14810	1950	14810	421	5145	1211	3208	1490
10^3	1498489	22741	1498489	4222	501606	17028	52456	24562
10^4	149984999	254602	149984999	42253	50016135	222298	732567	345132
10^5	14999849998	2843156	14999849998	421121	5000049999	2711893	9343950	4446272
10^6	1499998499998	29786804	1499998499998	4206606	500001609198	32016162	113162293	54436743

Em geral, pode-se perceber que houve uma diminuição geral nas ordens de grandeza dos contadores.

Os algoritmos mais efetivos para as atribuições foram o Insertion, o Bubble e o Bubble com sentinela , todos com ordem de grandeza de 10^7. Já para as comparações, os mais eficazes foram o Insertion e o Bubble, ambos com ordem de grandeza de 10^6.

Os piores casos para atribuições foram o Cocktail, este que ficou com ordem de grandeza de 10^{12}. Já para as comparações foram o Bubble e o Cocktail, ambos com ordem de grandeza de 10^{12}.

Vetor quase ordenado inversamente:

Atribuições:

Tam. vetor	Bubble	Bubble c/ sentinela	Cocktail	Insertion	Selection	Merge	Heapsort	Quicksort
10^2	33185	33185	30769	11655	5049	4032	3118	2019
10^3	4350342	4350342	3094085	1456108	500499	59856	40153	23946
10^4	448461958	448461958	352187717	149547313	50004999	801696	499578	244345
10^5	44984558689	44984558689	36025000873	14995452890	5000049999	10013568	5988370	2449172
10^6	4499845500592	4499845500592	3666292246411	1499954500191	500000499999	119708544	69918794	24504189

Comparações:

Tam. vetor	Bubble	Bubble c/ sentinela	Cocktail	Insertion	Selection	Merge	Heapsort	Quicksort
10^2	14850	14818	14850	11061	5238	1529	2981	1126
10^3	1493935	1498474	1493935	1450114	502457	21297	48191	20324
10^4	149811548	149984914	149811548	149487319	50024611	264214	678072	294420
10^5	14994685318	14999849909	14994685318	14994852896	5000246271	3141505	8766785	3949227
10^6	1499963347047	1499998499880	1499963347047	1499948500197	500002462100	36377964	107737327	49467014

O melhor caso para as atribuições foi o Quick sort, sendo ele o único com ordem de grandeza de 10^7. Já para as comparações, foram tanto o Quick sort, quanto o Merge sort, ambos, também, com ordem de grandeza de 10^7.

Os piores casos tanto em comparações quanto em atribuições ficou empatado entre o Bubble, Bubble com sentinela, Cocktail, Insertion e o Selection, todos eles com a ordem de grandeza de 10^{12} em ambas categorias.

Vetor com vários valores repetidos:

Atribuições:

Tam. vetor	Bubble	Bubble c/ sentinela	Cocktail	Insertion	Selection	Merge	Heapsort	Quicksort
10^2	20168	20168	24138	7316	5049	4032	3118	2349
10^3	2236566	2236566	3025998	751516	500499	59856	42092	30247
10^4	225242745	225242745	320618344	75140909	50004999	801696	522376	371593
10^5	22537780142	22537780142	32145668606	7513193374	5000049999	10013568	6224522	4394360
10^6	2249871186363	2249871186363	3218149889185	749963062115	500000499999	119708544	72148231	50928693

Comparações:

Tam. vetor	Bubble	Bubble c/ sentinela	Cocktail	Insertion	Selection	Merge	Heapsort	Quicksort
10^2	14774	14316	14774	6722	5205	1590	3011	708
10^3	1498500	1488526	1498500	745522	502454	26114	50488	16076
10^4	149985000	149824918	149985000	75080915	50024959	361484	706051	249144
10^5	14999850000	14997231281	14999850000	7512593380	5000249942	4608955	9058617	3566817
10^6	1499998500000	1499979912578	1499998500000	749957062121	500002499940	56022720	110386703	43039814

Os melhores algoritmo para as atribuições nesse caso foram o Merge sort, heapsort e Quick sort, todos com ordem de grandeza de 10^8. Já para as comparações, o mais eficiente foi o Quick sort, este sendo o único com ordem de grandeza de 10^7.

Os piores casos, novamente, tanto em atribuições como em comparações, novamente, ficou em empatado entre o Bubble, Bubble com sentinela, Cocktail, insertion e o Selection, todos eles com a ordem de grandeza de 10^{12} em ambas categorias.

Conclusão

Apesar de cada algoritmo ter sua serventia em determinada situação, é possível inferir a partir dos resultados os algoritmos mais indicados para cada um dos casos analisados, principalmente para vetores muito grandes (visto que para tamanhos pequenos os algoritmos se mantiveram, em geral, equiparados um ao outro), tendo como parâmetro apenas as atribuições e comparações dos elementos dos vetores.

Para o vetor aleatório, o algoritmos mais eficaz foi o Merge sort, este que tem apresenta somente o problema de alocar memória adicional. Se memória não for um problema, portanto, recomenda-se seu uso ao invés dos outros. Entretanto, se uso da memória extra não seja possível, o uso do Quick sort ou do Heap sort, apesar de um pouco menos eficiente na quantidade de comparações, são as melhores opções dentre as demais.

Para o caso do vetor quase ordenado, temos que o mais eficaz foi o Insertion, este que se mostrou o mais eficiente em ambos parâmetros, assim, sendo o mais recomendado para esta situação.

Para o caso do vetor quase ordenado inversamente, temos que o mais eficaz foi o Quick sort, superando o Merge sort apenas no contador de comparações (além de não necessitar da memória extra).

Por fim, o vencedor para o caso do vetor com valores repetidos foi novamente o Quick sort, superando, novamente, por pouco o Merge sort na quantidade de comparações.