Relatório sobre o Projeto de Algoritmo

Flávio leone Ramos: 4809

October 6, 2018

1 Introdução e Objetivos

Neste relatório contém análises do tempo de execução dos algoritmos para ordenação Insertion Sort, Selection Sort, Shell Sort, Quick Sort, Merge Sort e Heap Sort, a fim de compará-los computacionalmente em diferentes casos de uso e entradas. Juntamente são listados resultados obtidos em relação ao tempo de busca em TAD's sequencial, binária e busca em árvore binária, com direrentes casos estudados. Semelhante calculamos o tempo a busca pelo maior e menor valor dentro dessa TAD a partir de um arquivo. Foram realizados vários casos de uso para obtenção da média e desvio padrão com entradas diferentes para cada caso.

2 Materiais e Métodos

Os algoritmos estudados neste trabalho foram implementados na linguagem de programação C/C++ e avaliados em um *hardware* composto por um processador Intel Core i5, 1T de HDD e 8G de RAM. Para avaliar os algoritmos, foram utilizados vários casos de teste (entradas de tamanhos diversos) de inteiros gerados aleatoriamente e sem repetição.

GitHub. https://github.com/Flavioleone83/projetoAlgoritmo

3 Resultados

Na figura 1 podemos observar 3 casos de testes com entradas para possibilitar o melhor caso, caso médio e pior caso, extraindo assim informações quanto ao tempo de resposta de certos algoritmos de ordenação, dependendo do tamanho da entrada de dados. Os algoritmos que tiveram as melhores respostas foram os Merge Sort, Shell Sort, Quick Sort e Heap Sort, com tempo nlogn, independente de suas entradas. Os algoritmos que apresentaram um bom desempenho com entradas de até 10.000 foram os Insertion sort, Selection Sort e Bubble Sort, porém con entradas superiores se mostraram ineficientes quanto ao tempo de execução.

4 Conclusões

Foi observado que, no pior caso, os algoritmos Shell Sort, Heap Sort, Quick Sort e Merge sorte obtiveram um tempo de resposta muito baixo com entradas de até 10^6. No entando ,no pior caso o algoritmo insertion Sorte demostrou um grande atraso no tempo de ordenação. Com entradas em ordem crescente(melhor caso), o algoritmo Insertion Sort mostrou um tempo de resposta bem abaixo dos demais casos de teste. Em outros casos, o tempo de resposta se manteve semelhante no pior caso, caso médio e melhor caso.

Estudo de caso 1 (Elementos aleatórios)				
Algoritmo	**Tamanho da entrada**			
	1,000	**10,000**	**100,000**	**1,000,000**
Insertion Sort	0,001331	0,096154	9,724286	1260,515991
Selection Sort	0,002268	0,129394	12,517659	1085,909180
Shell Sort	0,000210	0,002078	0,026366	0,363266
Merge Sort	0,000200	0,001923	0,023624	0,277313
Quick Sort	0,000128	0,001125	0,013880	0,160816
Heap Sort	0,000142	0,001467	0,018754	0,263522
Bubble Sort	0,002895	0,275866	31,246124	3189,958469

Estudo de caso 2 (Elementos em ordem crescente)				
Algoritmo	**Tamanho da entrada**			
	1,000	**10,000**	**100,000**	**1,000,000**
Insertion Sort	0,000006	0,000070	0,000492	2,1638281
Selection Sort	0,002405	0,137486	11,396810	1193,412964
Shell Sort	0,000028	0,000592	0,005573	0,055210
Merge Sort	0,000132	0,001332	0,016836	0,190508
Quick Sort	0,000035	0,000380	0,004878	0,057493
Heap Sort	0,000115	0,001228	0,015162	0,177439
Bubble Sort	0,002192	0,155750	16,128735	1708,851440

Estudo de caso 3 (Elementos em ordem decrecente)				
Algoritmo	**Tamanho da entrada**			
	1,000	**10,000**	**100,000**	**1,000,000**
Insertion Sort	0,003820	0,198347	20,137939	x
Selection Sort	0,002478	0,128816	12,212448	1520,694702
Shell Sort	0,000085	0,000628	0,007630	0,077626
Merge Sort	0,000228	0,001276	0,015136	0,175772
Quick Sort	0,000063	0,000369	0,00442	0,057069
Heap Sort	0,000191	0,001135	0,014220	0,164736
Bubble Sort	0,004961	0,242452	24,444935	x

Figure 1: Tabela com os tempos de ordenação com elementos não repetidos e aleatórios.

Análise da média em 5 casos de uso(Entrada de dados aleatório)				
Algoritmo	**Tamanho da entrada**			
	1,000	**10,000**	**100,000**	**1,000,000**
Insertion Sort	0,001728	0,31810775	10,3514885	x
Selection Sort	0,0024525	0,4130105	12,09176175	x
Shell Sort	0,00014375	0,005852	0,0249015	0,3692815
Merge Sort	0,000175	0,0019655	0,023434	0,27815175
Quick Sort	0,000413	0,0011335	0,044155	0,16408175
Heap Sort	0,0004735	0,00150325	0,0191755	0,248494
Bubble Sort	0,01447975	0,283905	31,257449	x

Obtendo o desvio padrão em 5 casos de uso(Entrada de dados aleatório)				
Algoritmo	**Tamanho da entrada**			
	1,000	**10,000**	**100,000**	**1,000,000**
Insertion Sort	0,000153651	0,106530601	0,230790978	x
Selection Sort	0,000043008	0,143526983	0,128646	x
Shell Sort	0,000029914	0,002013161	0,000746893	0,091439
Merge Sort	0,00036569	0,000142686	0,001186567	0,015205081
Quick Sort	0,000131385	0,000107491	0,017603915	0,000859089
Heap Sort	0,000469215	0,00010145	0,000290097	0,007338436
Bubble Sort	0,037492292	0,009339011	0,068363722	x

Figure 2: Tabela com os tempos de ordenação com elementos não repetidos e aleatórios.

Estudo de caso 1				
Algoritmo	**Tamanho da entrada**			
	1,000	**10,000**	**100,000**	**1,000,000**
minMax1	0,000028	0,000052	0,000333	0,002523
minMax2	0,000012	0,000042	0,000318	0,002934
minMax3	0,000015	0,000039	0,000286	0,002832

Estudo de caso 2				
Algoritmo	**Tamanho da entrada**			
	1,000	**10,000**	**100,000**	**1,000,000**
minMax1	0,000022	0,000044	0,000299	0,002875
minMax2	0,000012	0,000037	0,000297	0,002755
minMax3	0,000014	0,000036	0,000274	0,002437

Estudo de caso 3				
Algoritmo	**Tamanho da entrada**			
	1,000	**10,000**	**100,000**	**1,000,000**
minMax1	0,000028	0,000044	0,000312	0,002552
minMax2	0,000012	0,000037	0,000364	0,002657
minMax3	0,000014	0,000037	0,000290	0,002764

Figure 3: Tabela com os tempos de busca pelo maior e menor valor dentro de uma TAD.

Estudo de caso 1 (Elemento encontrado)				
Algoritmo	**Tamanho da entrada**			
	1,000	**10,000**	**100,000**	**1,000,000**
Pesquisa Sequencial	0,000005	0,000005	0,000012	0,000003
Pesquisa Binária	0,000189	0,003321	0,032279	0,414655
Árvore Binária	0,000060	0,000605	0,005993	0,081499

Estudo de caso 2 (Elemento não encontrado)				
Algoritmo	**Tamanho da entrada**			
	1,000	**10,000**	**100,000**	**1,000,000**
Pesquisa Sequencial	0,000006	0,000014	0,000007	0,000007
Pesquisa Binária	0,000190	0,002707	0,030663	0,417519
Árvore Binária	0,000071	0,000470	0,006418	0,081969

Figure 4: Tabela com os tempos de pesquisa por um valor aleatório dentro de uma TAD.