Projeto 1 – Algoritmo de Ordenação

Douglas Patrick Barbosa Boaventura -- 5144
Jordi Henrique Marques da Silva -- 3927
Christian Rodrigues Moura - 3629

Outubro 08, 2018

1. Introdução e objetivos

Neste relatório são reportadas análises de tempo de execução dos algoritmos (Selection sort, Insertion sort, Shellsort, Mergesort, Quicksort e Heapsort) a fim de, compará-los computacionalmente, determinando qual algoritmo possui maior desempenho.

2. Materiais e Métodos

Os algoritmos estudados neste trabalho foram implementados sobre a linguagem de programação C e avaliados em um hardware composto com um processador Intel Core i5-7300HQ 2.5 GHz de 1000 GB de HDD e 8GB de RAM DDR4. Para avaliar os algoritmos, foram utilizados vários casos de teste com entradas de diversos tamanhos entre elas 10, 1000, 10000, 100000, 1000000, 1000000.

3. Resultados

Algoritmo	Tamanho da entrada (n)				
*	**10**	**1000**	**10000**	**100000**	**1000000**
Selection Sort Aleatório	0.000000	0.001000	0.121000	12.324000	1217.406982
Selection Sort Crescente	0.000000	0.001000	0.121000	12.202000	1329.373047
Selection Sort Decrescente	0.000000	0.001000	0.116000	11.937000	1211.635010
Insertion Sort Aleatório	0.000000	0.000000	0.000000	0.001000	891.0908997
Insertion Sort Crescente	0.000000	0.000000	0.000000	0.000000	0.004000
Insertion Sort Decrescente	0.000000	0.000000	0.000000	0.000000	1760.741943
Shell Sort Aleatório	0.000000	0.000000	0.000000	0.004000	0.048000
Shell Sort Crescente	0.000000	0.000000	0.000000	0.004000	0.044000
Shell Sort Decrescente	0.000000	0.000000	0.000000	0.004000	0.044000
Merge Sort Aleatório	0.000000	0.001000	0.002000	0.016000	0.194000

Merge Sort Crescente	0.000000	0.000000	0.002000	0.017000	0.185000
Merge Sort Decrescente	0.000000	0.000000	0.002000	0.017000	0.183000
Quick Sort Aleatório	0.000000	0.000000	0.000000	0.005000	0.055000
Quick Sort Crescente	0.000000	0.000000	0.000000	0.004000	0.049000
Quick Sort Decrescente	0.000000	0.000000	0.000000	0.004000	0.047000
Heap Sort Aleatório	0.000000	0.000000	0.001000	0.014000	0.151000
Heap Sort Crescente	0.000000	0.000000	0.001000	0.013000	0.148000
Heap Sort Decrescente	0.000000	0.000000	0.001000	0.013000	0.145000

4. Conclusões

Foi possível concluir com execução de cada algoritmo para as diversas quantidades de entras, que o algoritmo de pior desempenho em geral foi Selection sort, que independente da ordenação sempre terá o mesmo custo $O(n^2)$.

O algoritmo de melhor desempenho geral foi Shell sort que utiliza a quebra sucessiva para inserir em sequencia. Logo em seguida vem o Merge Sort por sua abordagem de divisão e conquista consegue subdividir a instancia em subproblemas e utilizar a recursividade para ordenar.

Para entradas de um milhão de dados o algoritmo com pior desempenho foi o Insertion Sort em ordem decrescente que teve o tempo de execução de 29 minutos, o de melhor desempenho foi Shell Sort.

Para a quantidade de 10 entradas não houve diferença de tempo entre os algoritmos. Já para entradas de 1000 o algoritmo Selection sort obteve o mesmo tempo de execução para todos os tipos de ordenação tanto crescente, decrescente e aleatório. O Merge sort teve um custo maior na ordenação aleatória.

Os algoritmos Selection Sort e Insertion Sort são os menos eficientes para grandes quantidades de dados, enquanto os todos os demais tem tempo de execução inferior a um minuto.

5. Anexos

Git Hub: https://github.com/christianxng/Projeto-de-Algoritmos---Projeto-1
Git Hub: https://github.com/JordiHOFC/Devv