

# 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 i5-6200U 2.3 GHz de 1000 GB de HDD e 8GB de RAM. 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
<b>Selection Sort Aleatório</b>	0.000000	0.001000	0.121000	12.324000	1217.406982
<b>Selection Sort Crescente</b>	0.000000	0.001000	0.121000	12.202000	1329.373047
<b>Selection Sort Decrescente</b>	0.000000	0.001000	0.116000	11.937000	1211.635010
<b>Insertion Sort Aleatório</b>	0.000000	0.000000	0.000000	0.001000	891.0908997
<b>Insertion Sort Crescente</b>	0.000000	0.000000	0.000000	0.000000	0.004000
<b>Insertion Sort Decrescente</b>	0.000000	0.000000	0.000000	0.000000	1760.741943
<b>Shell Sort Aleatório</b>	0.000000	0.000000	0.000000	0.004000	0.048000
<b>Shell Sort Crescente</b>	0.000000	0.000000	0.000000	0.004000	0.044000
<b>Shell Sort Decrescente</b>	0.000000	0.000000	0.000000	0.004000	0.044000
<b>Merge Sort</b>	0.000000	0.001000	0.002000	0.016000	0.194000

<b>Aleatório</b>					
<b>Merge Sort Crescente</b>	0.000000	0.000000	0.002000	0.017000	0.185000
<b>Merge Sort Decrescente</b>	0.000000	0.000000	0.002000	0.017000	0.183000
<b>Quick Sort Aleatório</b>	0.000000	0.000000	0.000000	0.005000	0.055000
<b>Quick Sort Crescente</b>	0.000000	0.000000	0.000000	0.004000	0.049000
<b>Quick Sort Decrescente</b>	0.000000	0.000000	0.000000	0.004000	0.047000
<b>Heap Sort Aleatório</b>	0.000000	0.000000	0.001000	0.014000	0.151000
<b>Heap Sort Crescente</b>	0.000000	0.000000	0.001000	0.013000	0.148000
<b>Heap Sort Decrescente</b>	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 entradas, 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.