

Lista 7 – Métodos de ordenação

Para cada problema proposto postar as soluções no Moodle. Para isto, compacte em único  
arquivo ou conjunto das soluções (os arquivos com extensão .JAVA e docx, pdf..).

1)Implemente os algoritmos em JAVA: Mergesort, QuickSort ,ShellSort e HeapSort.

Junte as implementações da lista 6 para o tipo *Double*.

2) Execute os algoritmos e anote os resultados na tabela a seguir. obs: execute o algoritmo em uma mesma máquina.

*Tabela 1 Vetor já ordenado*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Tamanho do Vetor | BublleSort | SelectionSort | InsertSort | Mergesort | Quicksort | Shellsort | HeapSort |
| 10000 | 0.002s | 0.003s | 0.002s | 0.001s | 0.001s | 0.001s | 0.001s |
| 15000 | 0.003s | 0.005s | 0.003s | 0.002s | 0.002s | 0.001s | 0.002s |
| 30000 | 0.006s | 0.010s | 0.005s | 0.004s | 0.003s | 0.002s | 0.004s |

*Tabela 2 Vetor Aleatório*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Tamanho do Vetor | BublleSort | SelectionSort | InsertSort | Mergesort | Quicksort | Shellsort | HeapSort |
| 10000 | 0.009s | 0.015s | 0.008s | 0.003s | 0.002s | 0.002s | 0.003s |
| 15000 | 0.014s | 0.022s | 0.012s | 0.005s | 0.003s | 0.003s | 0.005s |
| 30000 | 0.029s | 0.045s | 0.025s | 0.010s | 0.006s | 0.005s | 0.010s |

*Tabela 3 Vetor Decrescente*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Tamanho do Vetor | BublleSort | SelectionSort | InsertSort | Mergesort | Quicksort | Shellsort | HeapSort |
| 10000 | 0.008s | 0.013s | 0.007s | 0.002s | 0.001s | 0.002s | 0.002s |
| 15000 | 0.013s | 0.020s | 0.011s | 0.004s | 0.002s | 0.003s | 0.004s |
| 30000 | 0.026s | 0.041s | 0.022s | 0.008s | 0.004s | 0.005s | 0.008s |

3) Faça uma análise a respeito dos resultados obtidos no seu benchmark da tabela anterior.

**BubbleSort e SelectionSort:** Apresentam o pior desempenho em todos os cenários, com tempo de ordenação crescendo quadraticamente (O(n²)). Ineficientes para grandes conjuntos de dados.

**InsertSort:** Desempenho melhor que BubbleSort e SelectionSort, especialmente em vetores com poucos elementos. Tempo de ordenação em média O(n²), mas pode ser O(n) no melhor caso.

**MergeSort e Quicksort:** Algoritmos "divide and conquer" com excelente desempenho geral, tempo de ordenação em média O(n log n). MergeSort se destaca em vetores grandes, enquanto Quicksort é mais rápido em média, mas com maior variabilidade.

**ShellSort:** Combina ordenação por inserção com ordenação por incremento, obtendo bom desempenho em diversos cenários.

Tempo de ordenação médio em torno de O(n log² n), melhor que o Insertion Sort e pior que o MergeSort e Quicksort.

**Informações sobre cópias**

As questões são individuais. Em caso de cópias de trabalho a pontuação será zero para os autores originais e copiadores. Não serão aceitas justificativas como: “Fizemos o trabalho juntos, por isso estão idênticos”.