Figure 1: Enter Caption

Relatório Ordenações

Igor Terplak Gutierrez e Patrick Froes

Outubro de 2024

Contents

1 Introdução

Neste projeto, foram analisados quatro algoritmos de ordenação: Insert Sort, Quick Sort, Shell Sort e Gnome Sort. A implementação foi realizada em Java, com o objetivo de observar o desempenho de cada algoritmo em diferentes conjuntos de dados. A análise incluiu a comparação de complexidade de tempo, eficiência prática e comportamento com diferentes tamanhos de conjunto de dados.

2 Insert Sort

O Insert Sort é um algoritmo baseado na inserção de elementos em uma lista ordenada à medida que percorre uma lista desordenada. Ele divide a lista em duas partes: uma parte já ordenada e outra ainda desordenada. Para cada elemento da parte desordenada, o algoritmo busca a posição correta na parte ordenada e o insere ali.

2.1 Caracteristicas

A complexidade de tempo varia entre $O(n)$ e $O(n^2)$, é estável e ideal para pequenas listas, porém é ineficiente em listas grandes e desordenadas, pois seu custo é muito alto para as mesmas.

3 Quick Sort

O Quick Sort é um algoritmo recursivo que segue o conceito de dividir e conquistar, ele seleciona um elemento pivô e organiza a lista para que os elementos menores que o pivô fiquem a esquerda e os maiores a direita, dividindo a lista em duas e aplicando o processo novamente em ambas recursivamente.

3.1 Caracteristicas

A complexidade de tempo varia entre $O(nlog(n))$ e $O(n^2)$, não possui estabilidade, porém é geralmente mais rápido em listas grandes e não ordenadas ou aleatórias, o maior determinante de seu desempenho é a escolha do elemento pivô.

4 Shell Sort

O Shell Sort se apresenta como uma versão melhorada do Insert Sort. O metodo de ordenação se baseia em comparar primeiro elementos distantes entre si e reduzir a distancia gradualmente, realizando assim a ordenação.

4.1 Caracteristicas

A complexidade de tempo varia entre $O(nlog(n))$ e $O(n^2)$, não possui estabilidade, mais utilizado em listas médias e se apresenta de maneira mais eficiente que o Insert Sort para listas grandes.

5 Gnome Sort

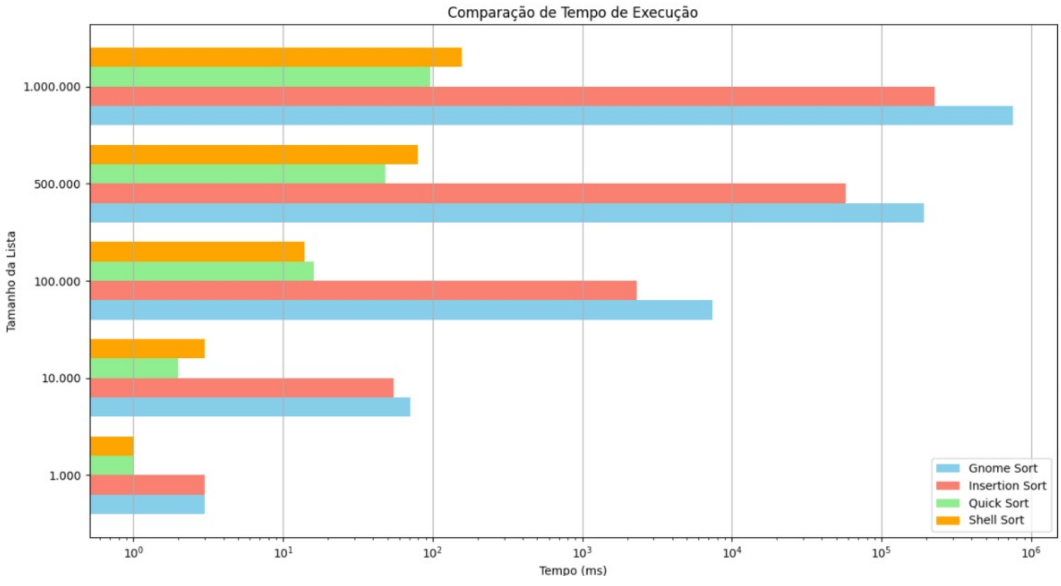
O Gnome Sort é um algoritmo de ordenação simples que funciona percorrendo a lista e, sempre que encontra um par de elementos fora de ordem, realiza uma troca e volta para verificar os elementos anteriores. Quando não há mais trocas a necessárias, a lista está ordenada.

5.1 Caracteristicas

A complexidade de tempo varia entre $O(n)$ e $O(n^2)$, é um algoritmo estavel, é um método que se destaca por sua simplicidade, porém é pouco eficiente quando a lista cresce de tamanho, pode ser útil para listas que estejam quase ordenadas.

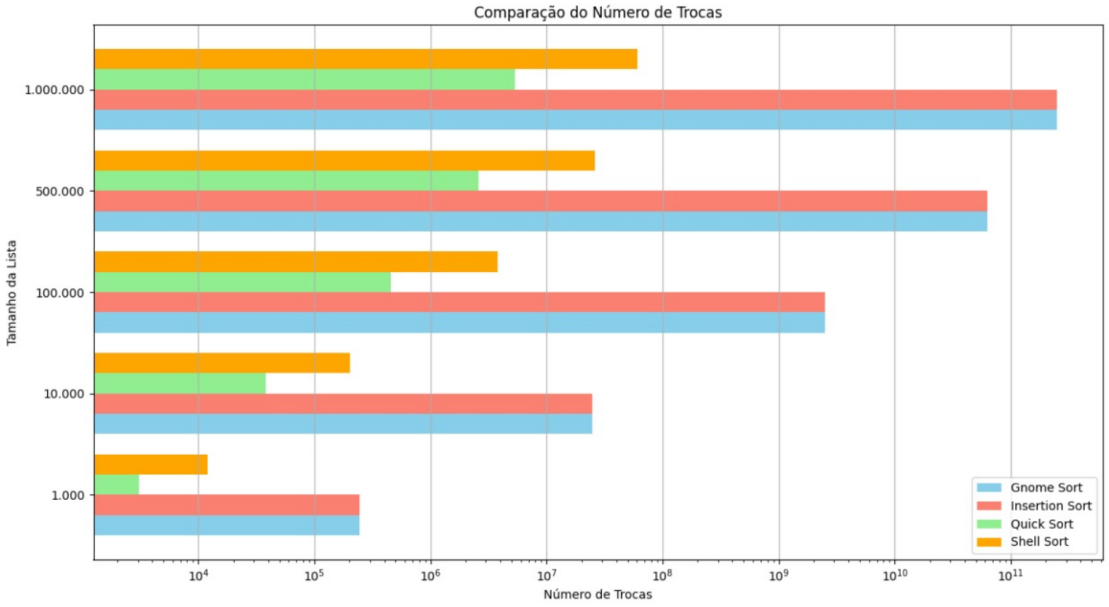
6 Resultados dos Testes

6.1 Número de interações



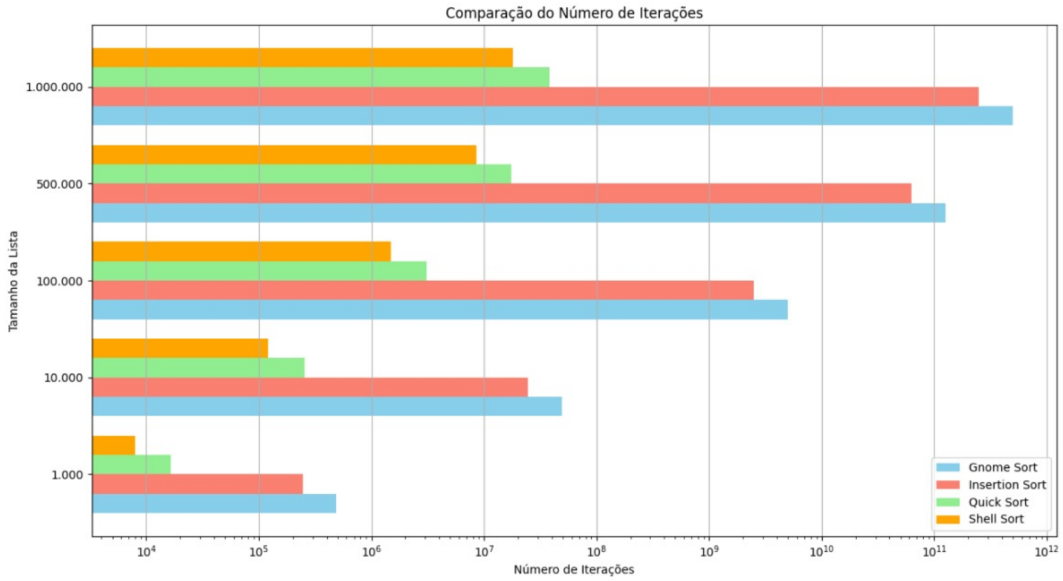
Algoritmo	1.000	10.000	100.000	500.000	1.000.000
Gnome Sort	3	71	7442	192662	758371
Insertion Sort	3	55	2314	57950	227581
Quick Sort	1	2	16	48	97
Shell Sort	1	3	14	80	157

6.2 Numero de Trocas



Algoritmo	1.000	10.000	100.000	500.000	1.000.000
Gnome Sort	243028	24678380	2504314643	62539271703	249991466783
Insertion Sort	243028	24678380	2504314643	62539271703	249991466783
Quick Sort	3094	38039	458036	2582606	5347449
Shell Sort	12011	203716	3813634	26197460	60768133

6.3 Tempo de Execução dos Algoritmos



Algoritmo	1.000	10.000	100.000	500.000	1.000.000
Gnome Sort	3	71	7442	192662	758371
Insertion Sort	3	55	2314	57950	227581
Quick Sort	1	2	16	48	97
Shell Sort	1	3	14	80	157

7 Conclusão

Neste projeto, fizemos as implementações de quatro diferentes metodos de ordenação, as análises realizadas mostraram as seguintes caracteristicas para cada metodo:

- **Insert Sort**: Possui uma eficiencia baixa, tendo sua utilidade apenas em listas pequenas e quase ordenadas, devido a sua baixa complexidade.
- **Quick Sort**:Possui alta velocidade e capacidade de lidar com grandes volumes de dados desordenados, porém é necessário cuidado na escolha do elemento pivo, sendo muito útil em qualquer tamanho de lista.
- **Shell Sort**: Sendo uma versão aprimorada do Insert Sort apresenta uma eficiencia média em listas pequenas e medianas, porém ainda não sendo recomendado para grandes volumes de dados, sua utilidade está focada especialmente em listas de tamanho médio.

- **Gnome Sort**: Sendo um algoritmo simples e ineficiente sua útilidade se limita a listas pequenas ou quase ordenadas, não sendo capaz de lidar com grandes volumes de dados.

Cada algoritmo de ordenação tem seu próprio comportamento específico quanto à complexidade de tempo e implantação. A seleção do algoritmo perfeito é determinada pelas especificações do conjunto de dados e do problema a ser resolvido. A seleção do algoritmo certo é baseada no tamanho da lista, no grau de desordem de seus valores e na estabilidade ou instabilidade da lista necessária.