Análise de Algoritmos

Docente: Warley Gramacho
Discentes: Neudison Nonato Maia Filho, Carlos Eduardo, Pablo Henrique Silva Ribeiro e Ângelo Gabriel
Universidade Federal do Tocantins - UFT

Sumário

1. Introdução

Este foi um trabalho realizado de Projeto e Análise de Algoritmos com o intuito de estudar os algoritmos de ordenação. Para facilitar o estudo e ter um pouco mais de precisão nas comparações foi utilizado o mesmo ambiente em todas. Os códigos foram feitos em Python, no compilador de versão 3.9. O computador utilizado tinha as seguintes características: Processador Intel Core i7-8700 (9º geração), Gtx 1660 Ti, 16 GB RAM, SSD mvme 1 TB.

2. Relatório

2.1. Bubble Sort (Iterativo)

2.1.1. Desordenado:
100 (Cem): 0.0000 segundos
1.000 (Mil): 0.0569 segundos
10.000 (Dez Mil): 6.1613 segundos
20.000 (Vinte Mil): 25.0750 segundos
50.000 (Cinquenta Mil): 157.9858 segundos
100.000 (Cem Mil): 645.0928 segundos
500.000 (Quinhentos Mil): Execução interrompida pelo alto tempo gasto
1.000.000 (Um Milhão): Execução interrompida pelo alto tempo gasto

2.1.2.Crescente:
100 (Cem): 0.0000 segundos
1.000 (Mil): 0.0782 segundos
10.000 (Dez Mil): 8.2028 segundos
20.000 (Vinte Mil): 33.9896 segundos
50.000 (Cinquenta Mil): 221.4234 segundos
100.000 (Cem Mil): 892.3955 segundos
500.000 (Quinhentos Mil): Execução interrompida pelo alto tempo gasto
1.000.000 (Um Milhão): Execução interrompida pelo alto tempo gasto

2.1.3. Decrescente:
100 (Cem): 0.0009 segundos
1.000 (Mil): 0.0290 segundos
10.000 (Dez Mil): 3.3008 segundos
20.000 (Vinte Mil): 13.2929 segundos
50.000 (Cinquenta Mil): 83.3639 segundos
100.000 (Cem Mil): 333.0735 segundos
500.000 (Quinhentos Mil): Execução interrompida pelo alto tempo gasto
1.000.000 (Um Milhão): Execução interrompida pelo alto tempo gasto

2.2. Insertion Sort (Iterativo)

2.2.1. Desordenado:
100 (Cem): 0.0000 segundos

1.000 (Mil): 0.0305 segundos
10.000 (Dez Mil): 2.5315 segundos
20.000 (Vinte Mil): 10.3266 segundos
50.000 (Cinquenta Mil): 66.3173 segundos
100.000 (Cem Mil): 269.6857 segundos
500.000 (Quinhentos Mil): Execução interrompida pelo alto tempo gasto
1.000.000 (Um Milhão): Execução interrompida pelo alto tempo gasto

2.2.2. Crescente:
100 (Cem): 0.0000 segundos
1.000 (Mil): 0.0000 segundos
10.000 (Dez Mil): 0.0000 segundos
20.000 (Vinte Mil): 0.0000 segundos
50.000 (Cinquenta Mil): 0.0000 segundos
100.000 (Cem Mil): 0.0000 segundos
500.000 (Quinhentos Mil): 0.0624 segundos
1.000.000 (Um Milhão): 0.1093 segundos

2.2.3. Decrescente:
100 (Cem): 0.0000 segundos
1.000 (Mil): 0.0462 segundos
10.000 (Dez Mil): 4.9845 segundos
20.000 (Vinte Mil): 19.8342 segundos
50.000 (Cinquenta Mil): 132.0181 segundos
100.000 (Cem Mil): 525.2938 segundos
500.000 (Quinhentos Mil): Execução interrompida pelo alto tempo gasto
1.000.000 (Um Milhão): Execução interrompida pelo alto tempo gasto

2.3. Merge Sort (Recursivo)

2.3.1. Desordenado:
100 (Cem): 0.000 segundos
1.000 (Mil): 0.0005 segundos
10.000 (Dez Mil): 0.0211 segundos
20.000 (Vinte Mil): 0.0598 segundos
50.000 (Cinquenta Mil): 0.1636 segundos
100.000 (Cem Mil): 0.3466 segundos
500.000 (Quinhentos Mil): 2.0715 segundos
1.000.000 (Um Milhão): 4.3688 segundos

2.3.2. Crescente:
100 (Cem): 0.0000 segundos
1.000 (Mil): 0.0009 segundos
10.000 (Dez Mil): 0.0280 segundos
20.000 (Vinte Mil): 0.0463 segundos
50.000 (Cinquenta Mil): 0.1313 segundos
100.000 (Cem Mil): 0.2712 segundos
500.000 (Quinhentos Mil): 1.5352 segundos

1.000.000 (Um Milhão): 3.1953 segundos

2.3.3. Decrescente:
100 (Cem): 0.0000 segundos
1.000 (Mil): 0.0020 segundos
10.000 (Dez Mil): 0.0307 segundos
20.000 (Vinte Mil): 0.0535 segundos
50.000 (Cinquenta Mil): 0.1333 segundos
100.000 (Cem Mil): 0.2796 segundos
500.000 (Quinhentos Mil): 1.5433 segundos
1.000.000 (Um Milhão): 3.1898 segundos

2.4. Quick Sort (Recursivo)

2.4.1. Desordenado:
100 (Cem): 0.0010 segundos
1.000 (Mil): 0.0009 segundos
10.000 (Dez Mil): 0.0160 segundos
20.000 (Vinte Mil): 0.0349 segundos
50.000 (Cinquenta Mil): 0.1039 segundos
100.000 (Cem Mil): 0.2190 segundos
500.000 (Quinhentos Mil): 1.3723 segundos
1.000.000 (Um Milhão): 2.9016 segundos

2.4.2. Crescente:
100 (Cem): 0.0000 segundos
1.000 (Mil): 0.0379 segundos
10.000 (Dez Mil): 3.9003 segundos
20.000 (Vinte Mil): 15.5900 segundos
50.000 (Cinquenta Mil): 97.4611 segundos
100.000 (Cem Mil): 390.5776 segundos
500.000 (Quinhentos Mil): Execução interrompida pelo alto tempo gasto
1.000.000 (Um Milhão): Execução interrompida pelo alto tempo gasto

2.4.3. Decrescente:
100 (Cem): 0.0009 segundos
1.000 (Mil): 0.0570 segundos
10.000 (Dez Mil): 5.8619 segundos
20.000 (Vinte Mil): 23.5166 segundos
50.000 (Cinquenta Mil): 149.2663 segundos
100.000 (Cem Mil): 603.0414 segundos
500.000 (Quinhentos Mil): Execução interrompida pelo alto tempo gasto
1.000.000 (Um Milhão): Execução interrompida pelo alto tempo gasto

2.5. Selection Sort ()

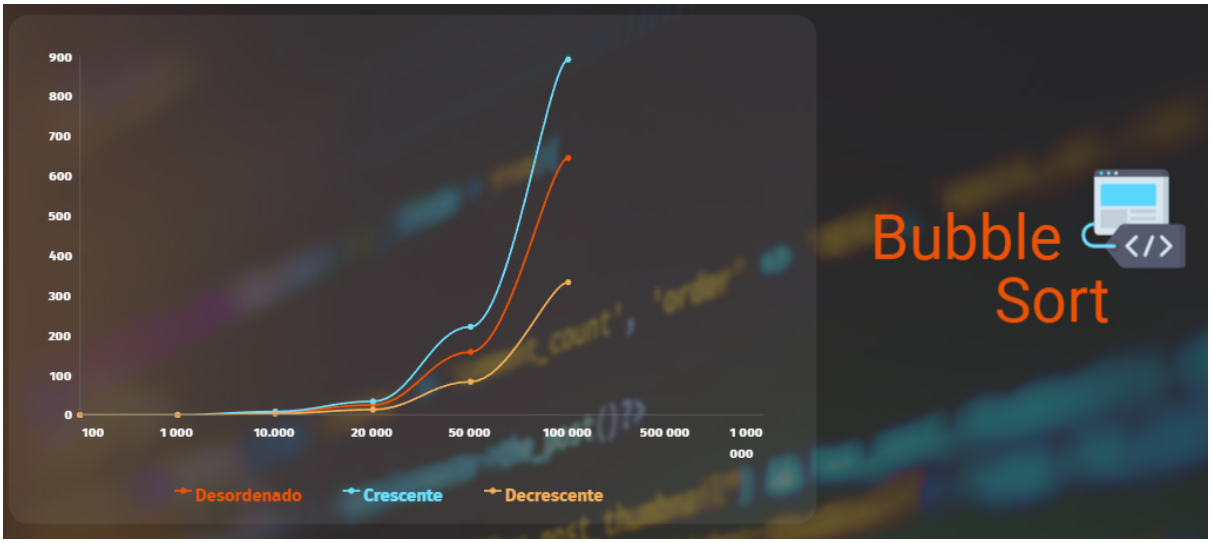
2.5.1. Desordenado:
100 (Cem): 0.0000 segundos
1.000 (Mil): 0.0150 segundos
10.000 (Dez Mil): 2.3155 segundos
20.000 (Vinte Mil): 9.6039 segundos
50.000 (Cinquenta Mil): 61.6360 segundos
100.000 (Cem Mil): 259.4773 segundos
500.000 (Quinhentos Mil): Execução interrompida pelo alto tempo gasto
1.000.000 (Um Milhão): Execução interrompida pelo alto tempo gasto

2.5.2. Crescente:
100 (Cem): 0.0000 segundos
1.000 (Mil): 0.0196 segundos
10.000 (Dez Mil): 2.3298 segundos
20.000 (Vinte Mil): 9.2944 segundos
50.000 (Cinquenta Mil): 58.0034 segundos
100.000 (Cem Mil): 232.8707 segundos
500.000 (Quinhentos Mil): Execução interrompida pelo alto tempo gasto
1.000.000 (Um Milhão): Execução interrompida pelo alto tempo gasto

2.5.3. Decrescente:
100 (Cem): 0.0000 segundos
1.000 (Mil): 0.0260 segundos
10.000 (Dez Mil): 2.5559 segundos
20.000 (Vinte Mil): 10.1461 segundos
50.000 (Cinquenta Mil): 65.5413 segundos
100.000 (Cem Mil): 259.5651 segundos
500.000 (Quinhentos Mil): Execução interrompida pelo alto tempo gasto
1.000.000 (Um Milhão): Execução interrompida pelo alto tempo gasto

3. Gráficos

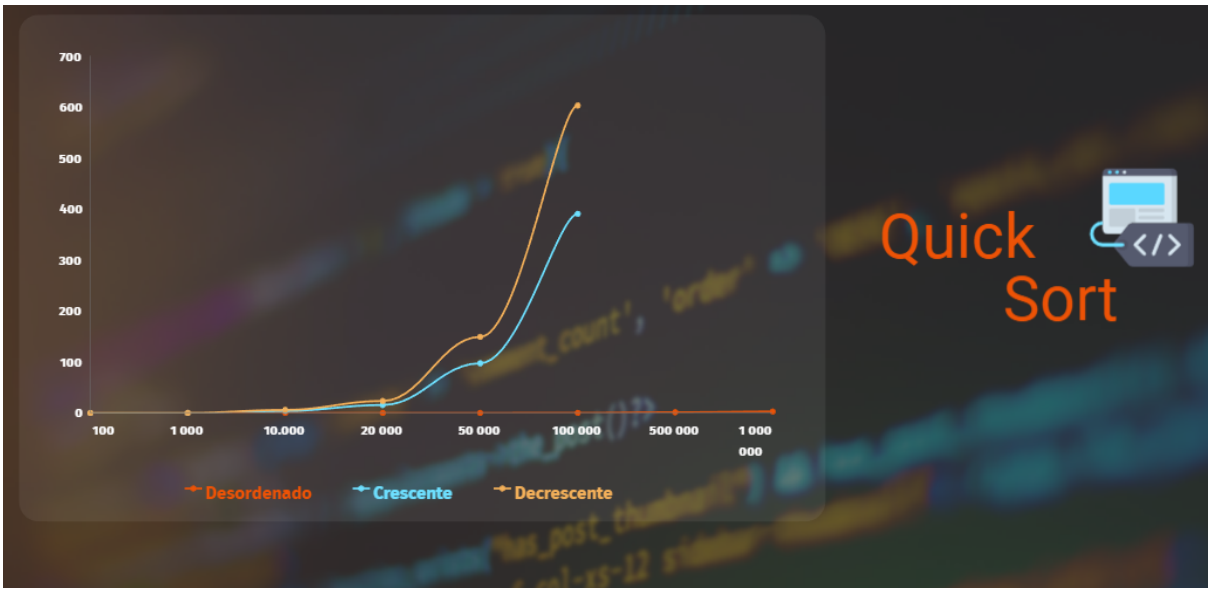
3.1. Bubble Sort (Iterativo)



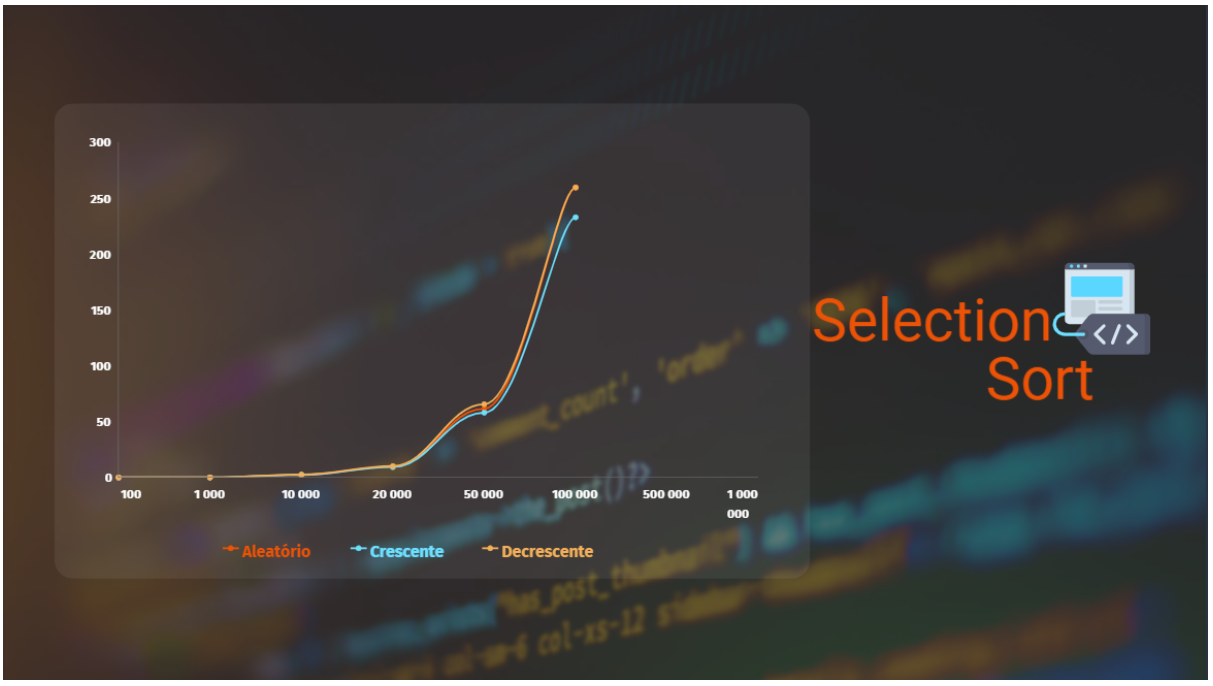
3.2. Insertion Sort (Iterativo)

3.3. Merge Sort (Recursivo):

3.4. Quick Sort (Recursivo):



3.5. Selection Sort (Iterativo):



4. Conclusão

Dos algoritmos testados, como esperado, o Quick Sort foi o mais rápido quando mexia com números desordenados. Indo na direção oposta o Bubble Sort foi o pior deles, onde até mesmo a demora para organizar não valia o tempo da espera passando de duas horas. De todos os algoritmos testados o Merge Sort foi o que mais apresentou promissor, o que já era esperado pois todos os casos dele são iguais O(n log n), e o que achamos ser o melhor dos algoritmos por causa de sua consistência.