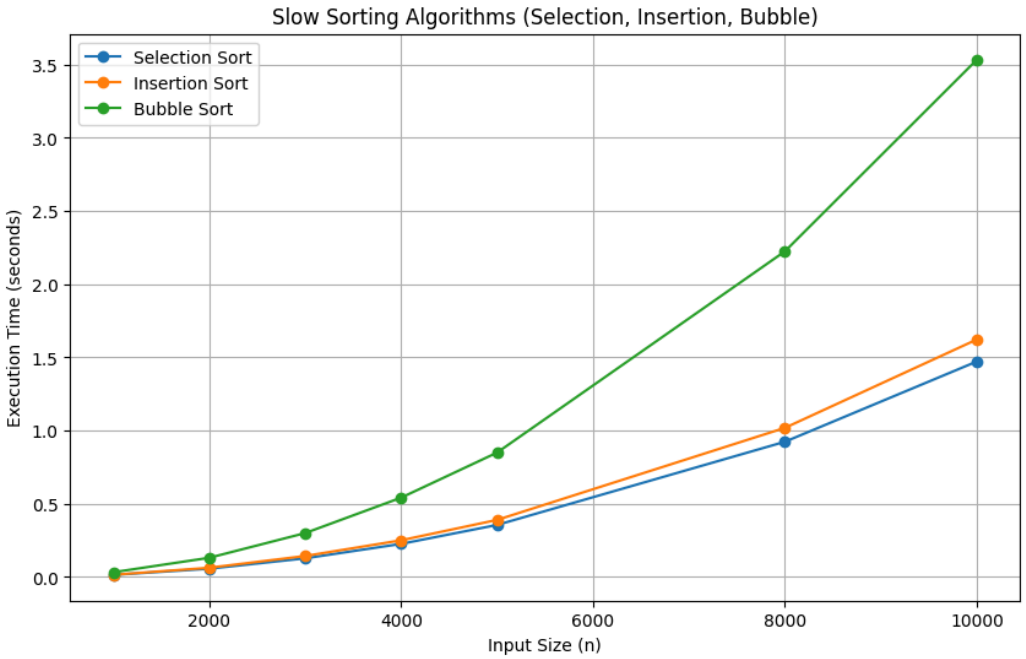
Relatório de Benchmark de Algoritmos de Ordenação

Autor: Gabriel S. N. Neto
Universidade: Universidade Federal do Rio Grande do Norte
Curso: Engenharia de Computação
Matéria: Estrutura de Dados 1
Data: 04/06/2025

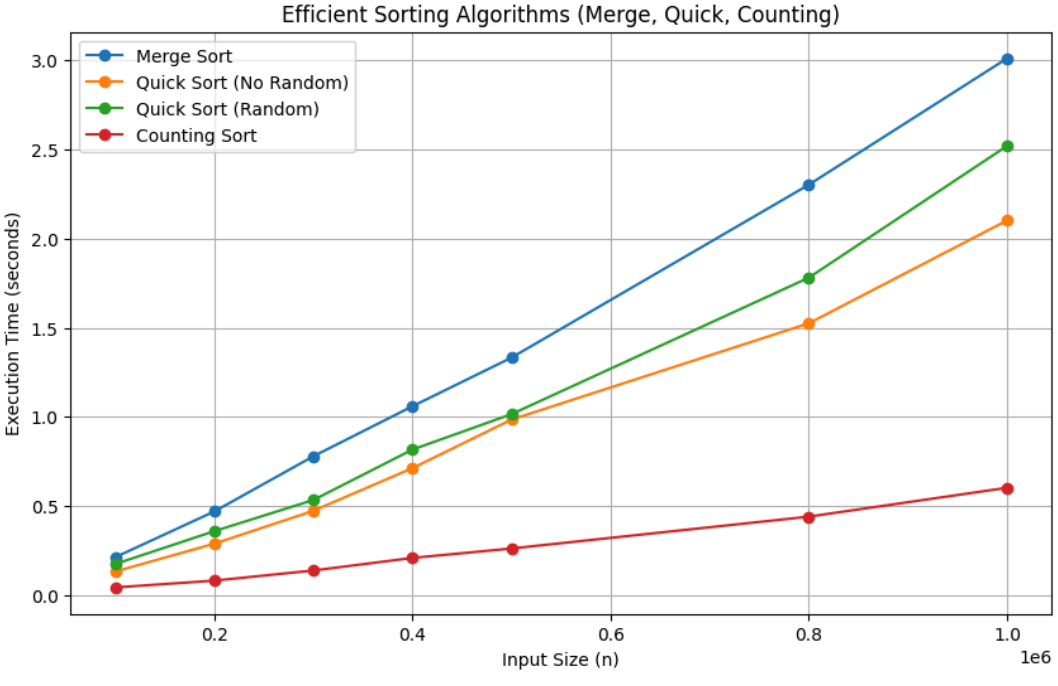
Este relatório tem como objetivo apresentar um benchmark comparativo entre diversos algoritmos de ordenação, incluindo Selection Sort, Bubble Sort, Insertion Sort, Merge Sort, Quick Sort (sem randomização), Quick Sort (com randomização) e Counting Sort. Além disso, serão apresentadas análises assintóticas dos algoritmos e uma "tier list" para ranqueamento com base nos testes realizados.

link para o código no github
ou acesse em: github.com/gabriel26077/sorting-algorithms-benchmark/

A imagem abaixo mostra uma comparação entre o Selection Sort, Insertion Sort e o Bubble Sort. É fácil perceber uma curvatura voltada para cima, o que indica que a relação não é linear e que deve vir a explodir conforme a entrada cresce indefinidamente.



A segunda imagem abaixo mostra um teste comparativo entre os algoritmos eficientes (Merge, Quick e Counting sort). Aqui já não é possível perceber uma curva, e sem a presença de maiores valores de ordenação não podemos dizer que esses gráficos não são lineares. Bom, ao menos conseguimos dizer que eles são bem melhores que os algoritmos mais lentos da imagem anterior. Mas para ter de fato a certeza que são melhores cabe uma análise assintótica separada para cada um.



1. Selection Sort

Análise Assintótica:

- Pior Caso: $O(n^2)$
- Melhor Caso: $\Omega(n^2)$
- Caso Médio: $\Theta(n^2)$
- link para o código

2. Bubble Sort

Análise Assintótica:

- Pior Caso: $O(n^2)$: array em ordem inversa.
- Melhor Caso: $\Omega(n)$ com uma pequena otimização de uma flag que indica se houve troca ou não.
- Caso Médio: $\Theta(n^2)$
- link para código

3. Insertion Sort

Análise Assintótica:

- Pior Caso: $O(n^2)$
- Melhor Caso: $\Omega(n)$
- Caso Médio: $\Theta(n^2)$
- link para o código

4. Merge Sort

Análise Assintótica:

- Pior Caso: O(n log n)
- Melhor Caso: Ω(n log n)
- Caso Médio: Θ(n log n)
- link para o código

Comentários: Algoritmo bom, estável e sempre constante.

5. Quick Sort (sem randomização)

Análise Assintótica:

- Pior Caso: $O(n^2)$
- Melhor Caso: $\Omega(n \log n)$
- Caso Médio: $\Theta(n \log n)$
- link para o código

Comentários: O pior caso ocorre quando o array está em ordem inversa.

6. Quick Sort (com randomização)

Análise Assintótica:

- Pior Caso: $O(n^2)$
- Melhor Caso: $\Omega(n \log n)$
- Caso Médio: $\Theta(n \log n)$
- link para o código

Comentários: Ocorre que aqui a randomização deixa extremamente improvável de ocorrer o pior caso.

7. Counting Sort

Análise Assintótica:

- Pior Caso: O(n + k) (onde k é o range dos números)
- Melhor Caso: Ω(n + k)
- Caso Médio: Θ(n + k)
- link para o código

Comentários: extremamente eficiente se o k for pequeno em relação a n. O melhor nos nossos testes.

Tier List (Ranqueamento Pessoal)

Após os testes realizados, segue a minha "tier list" para o ranqueamento dos algoritmos:

1. Counting Sort
2. Quick Sort (Com e sem randomização)
3. Merge sort
4. Selection sort
5. Insertion sort
6. Bubble sort

É importante ressaltar que apesar dos algoritmos 4,5,6 serem considerados lentos, eles possuem seu valor. Em aplicações específicas um simples insertion sort poderá ser muito eficiente, enquanto que o bubble e o selection possuem seu valor didático.