Moto Academy Android

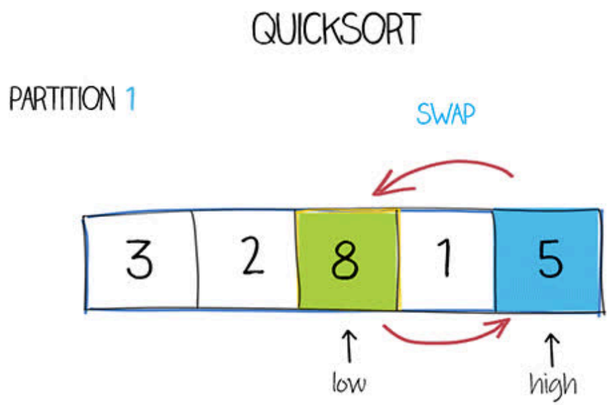
Algoritmos de Ordenação

Fabrício, Fernanda, Jeovana, Lucas e Rômulo

Quick Sort

é um algoritmo de ordenação eficiente e baseado no paradigma "divide e conquista". Ee funciona assim:

1. Escolha de um pivô
2. Particionamento
3. Recursão



Tempo:

- Pior caso: $O(n^2)$
- Melhor e médio caso: O(n log n)

Espaço:

- Pior caso: O(n)
- Melhor e médio caso: O(log n)

MergeSort

Objetivo: Apresentar brevemente o que o Merge Sort faz.

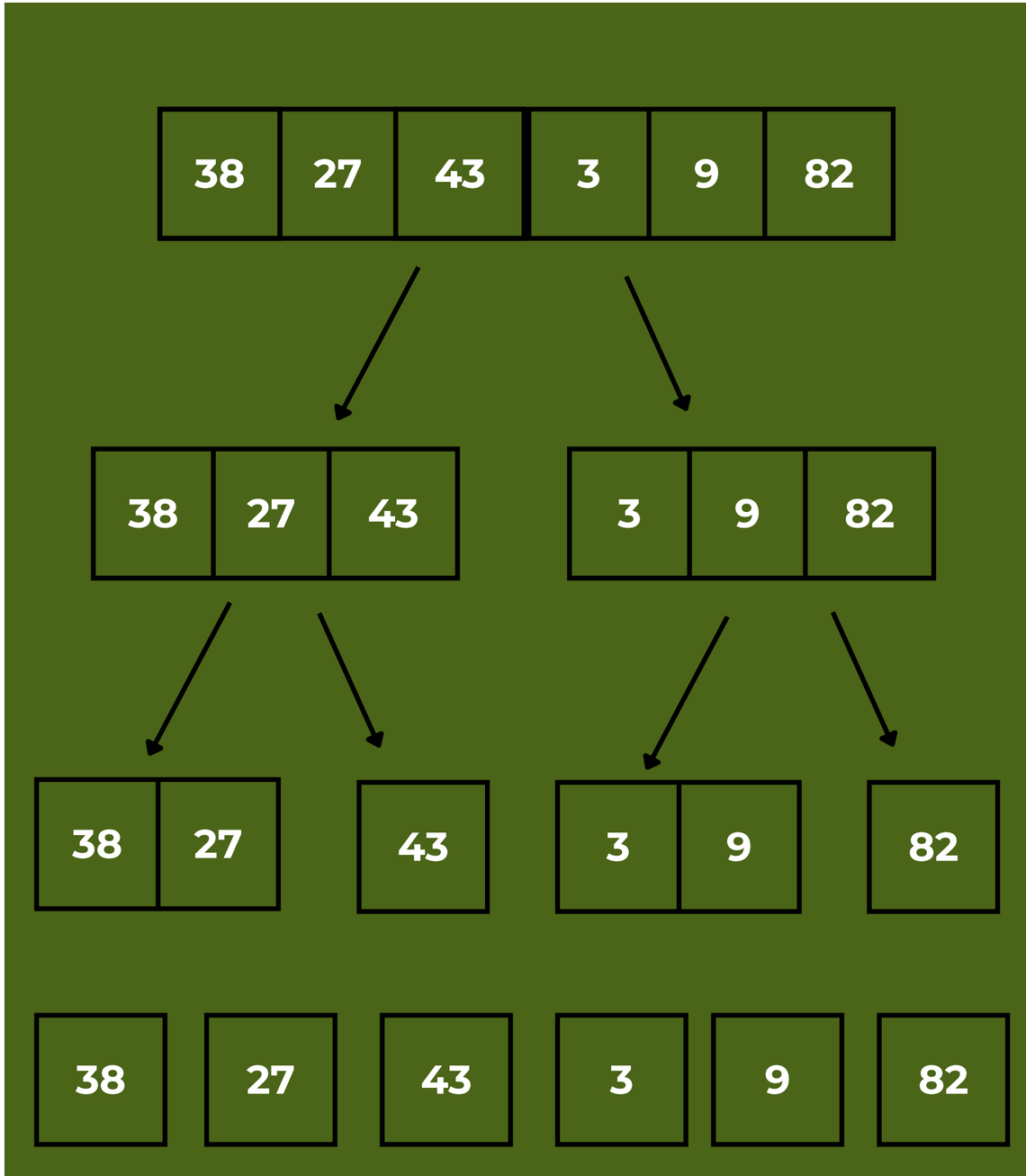
Merge Sort é um algoritmo de ordenação baseado na técnica de Dividir e Conquistar. Ele divide o vetor em partes menores até que cada parte tenha um único elemento e depois as une de forma ordenada.

Diagrama

Divisão

Ordenação

Mesclagem

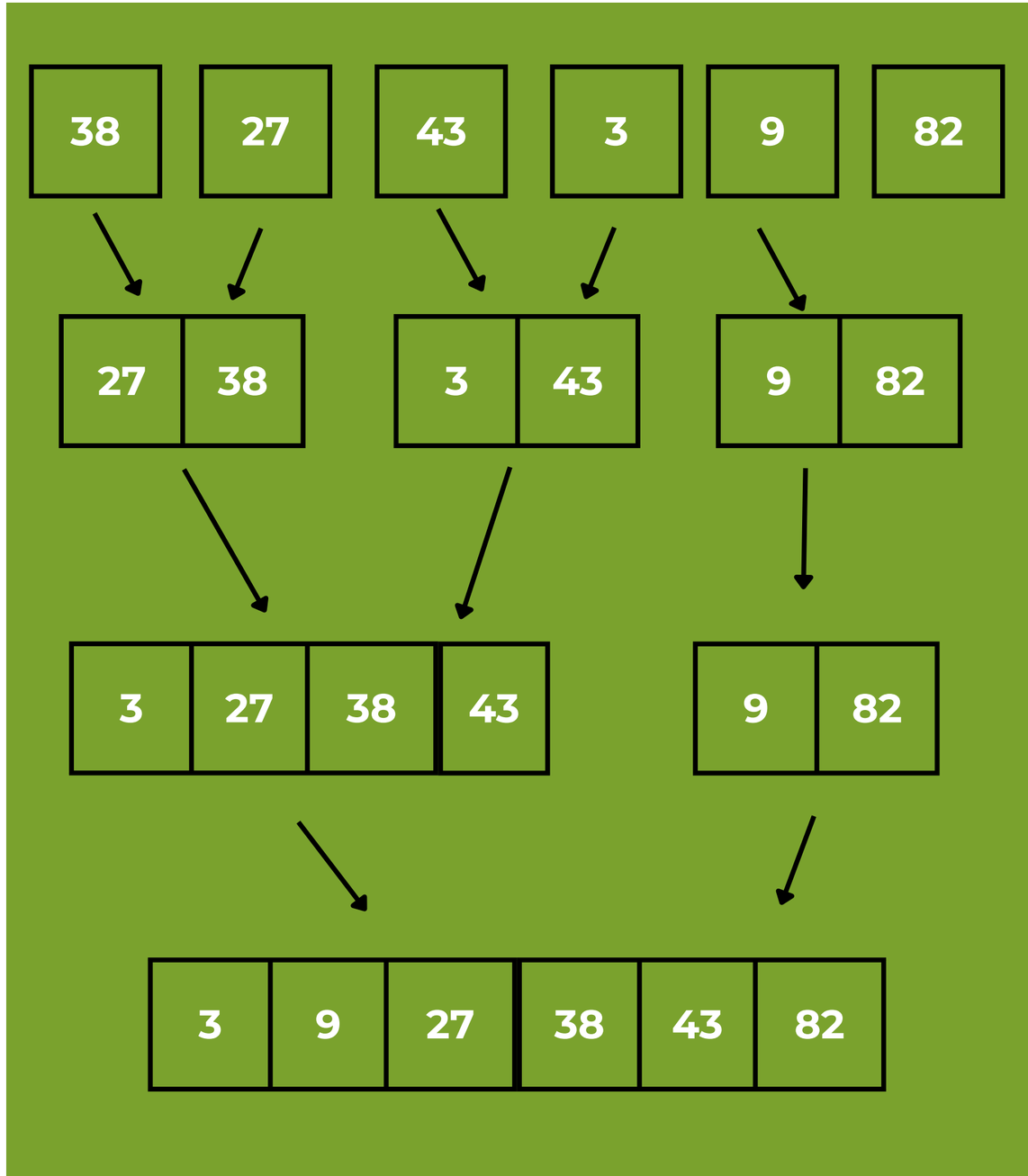


Diagrama

Divisão

Ordenação

Mesclagem



Merge Sort

01

Tempo de execução

O(n log n): no pior caso, médio e melhor caso.

02

Espaço

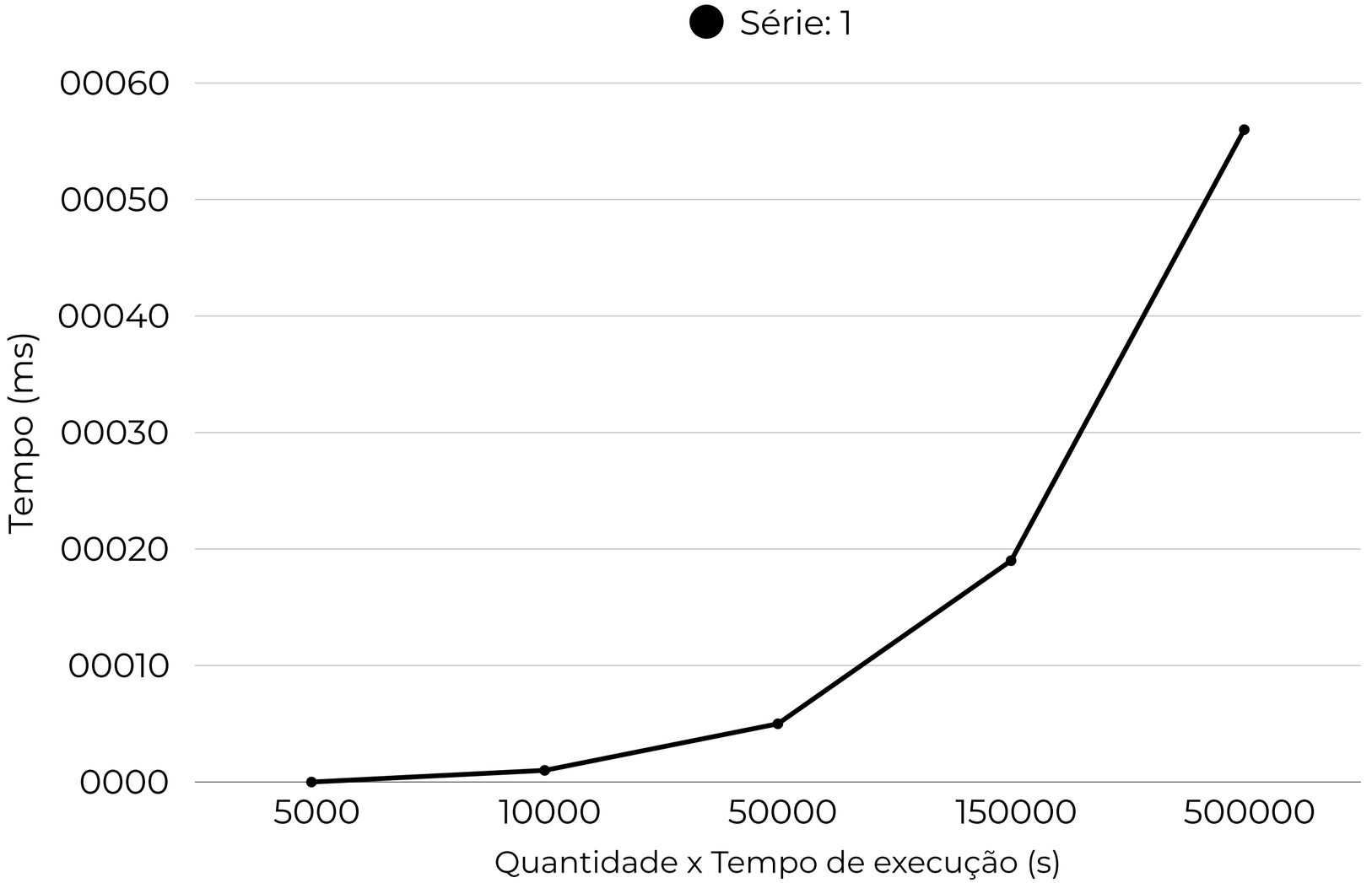
O (n): devido a necessidade de espaço adicional aos subarrays temporário.

03

Vantagens

- Estabilidade
- Eficiência

Gráfico do Merge Sort



BinaryInsertionSort

É uma versão otimizada do algoritmo clássico InsertionSort. Usando a busca binária reduzindo o número de comparações.

01

Funcionamento

Busca binária;
InsertionSort;

02

Complexidade

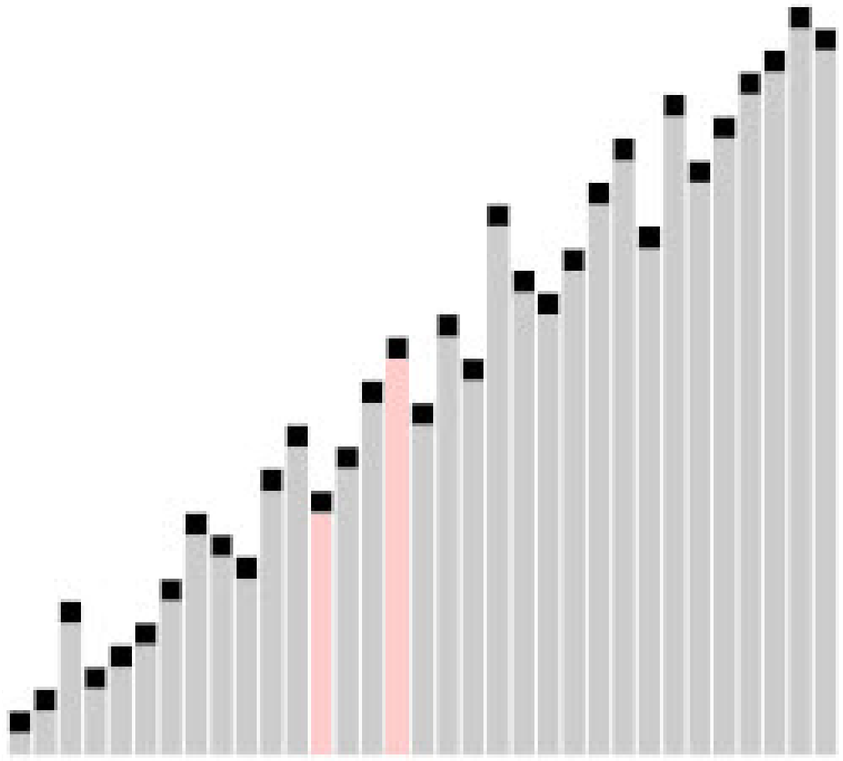
Melhor caso: O(log n)
Média: $O(n^2)$
Pior:$O(n^2)$

03

Vantagens

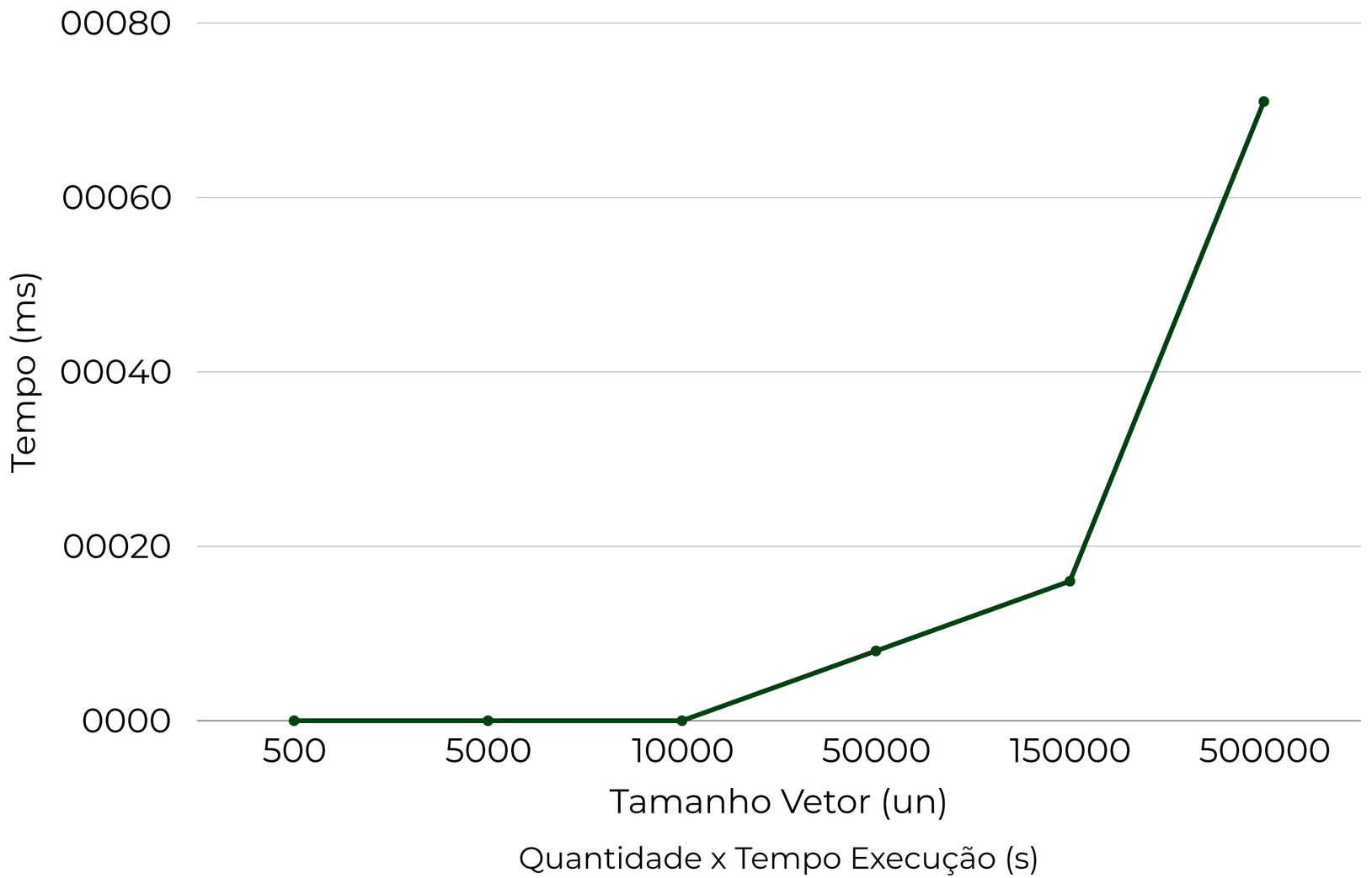
Reduz comparações;
Bom para listas pequenas
Simples;

CombSort



O ***CombSort*** é um algoritmo de ordenação de troca que aprimora o Bubble Sort ao usar um **gap** inicial igual ao tamanho do vetor, reduzido em cada iteração pelo fator de encolhimento (geralmente **~1,3**), para eliminar rapidamente elementos muito deslocados antes de uma última fase de trocas adjacentes. Em seu **pior** e no **caso médio**, sua complexidade é quadrática (**$O(n^2)$**), mas, graças à redução progressiva do gap, o desempenho prático é bem superior ao ***Bubble Sort*** puro, e no **melhor caso** — com o vetor já quase ordenado — atinge comportamento **O(n log n)**.

Gráfico Comparativo do CombSort



Quantidade x Tempo Execução (s)

ShellSort

O Shell Sort é uma versão melhorada do Insertion Sort e consegue ser muito mais rápido em **certos casos.**

O algoritmo não compara com o próximo, e sim a partir da escolha do **Gap.**

- **Gap: Distancia entre as comparações.**

O melhor uso, quanto maior a distancia do gap e maior é a entrada.

Não funciona tão bem em entradas pequenas.

Tempo:

- Pior caso: $O(n^2)$
- Melhor caso: $O(n \log n)$

O caso depende da escolha do Gap.

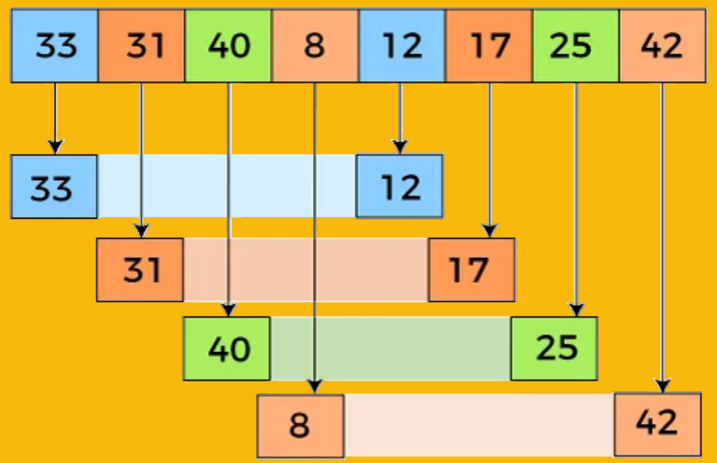
Espaço: O(1)

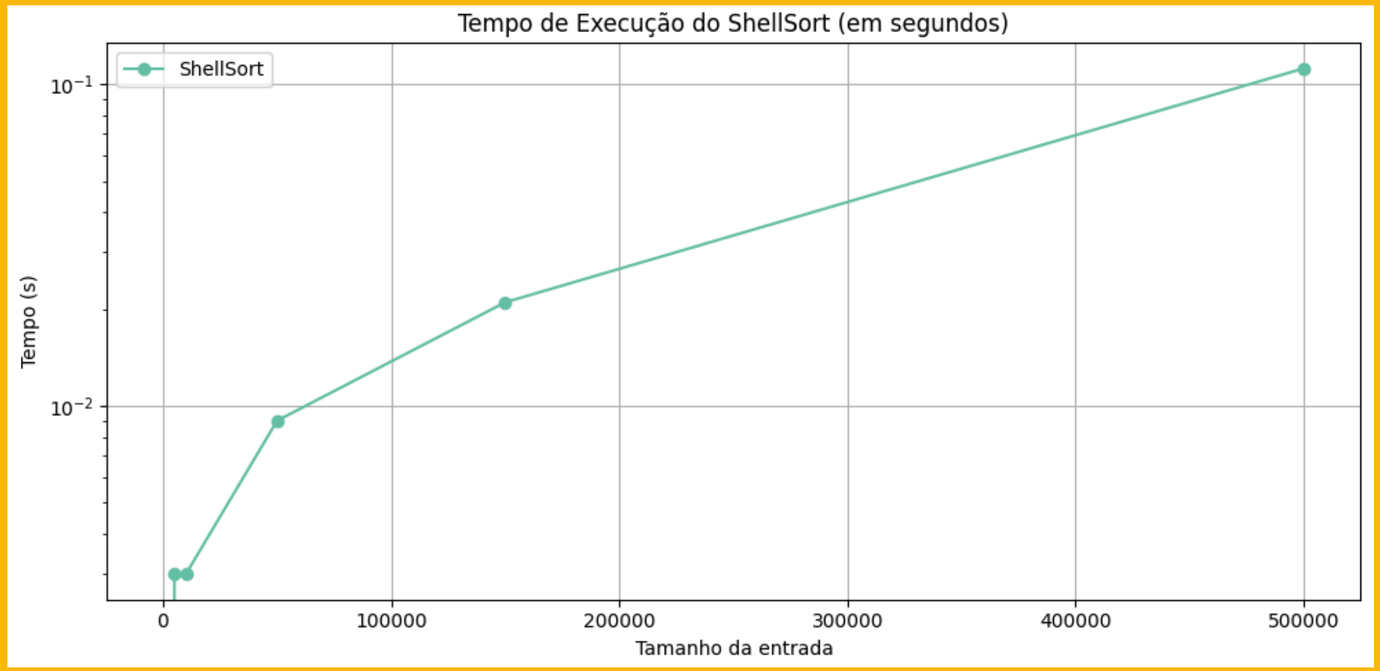
ShellSort

O funciona a partir do tamanho do vetor.

Gap = N/2.

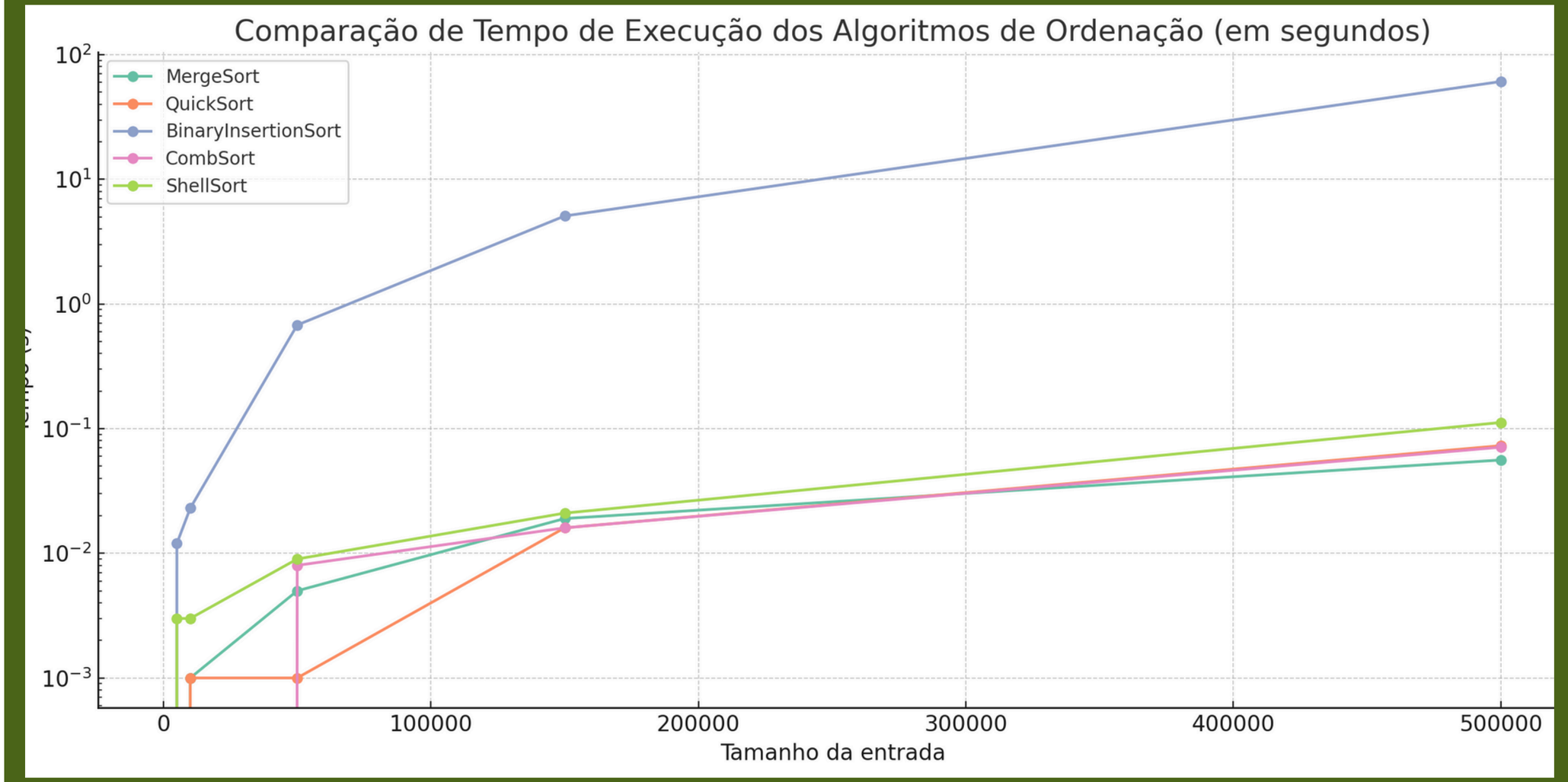
h = 8/2 = 4 → h= 4/2 = 2 → h = 2/2 = 1



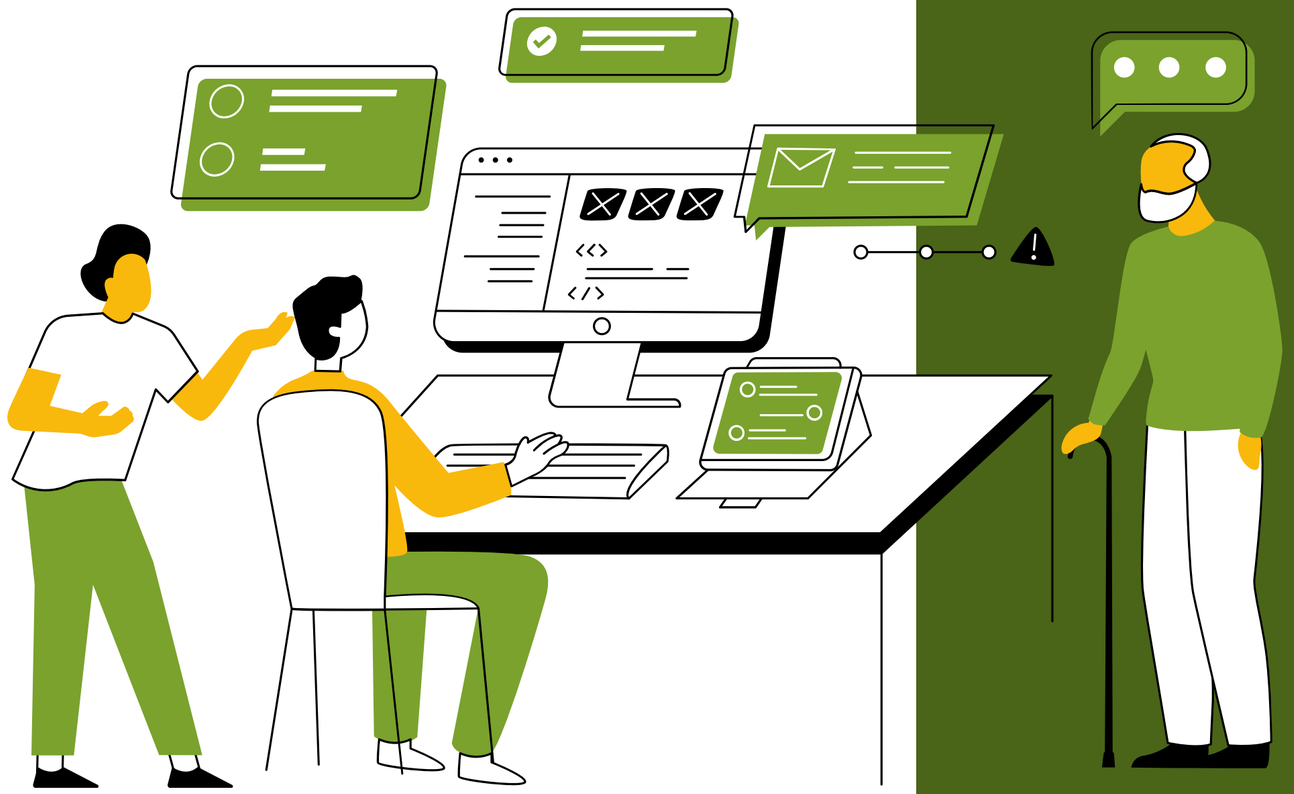


Dados coletados

(ms)



Obrigado!



https://github.com/abriciof/motoacademy