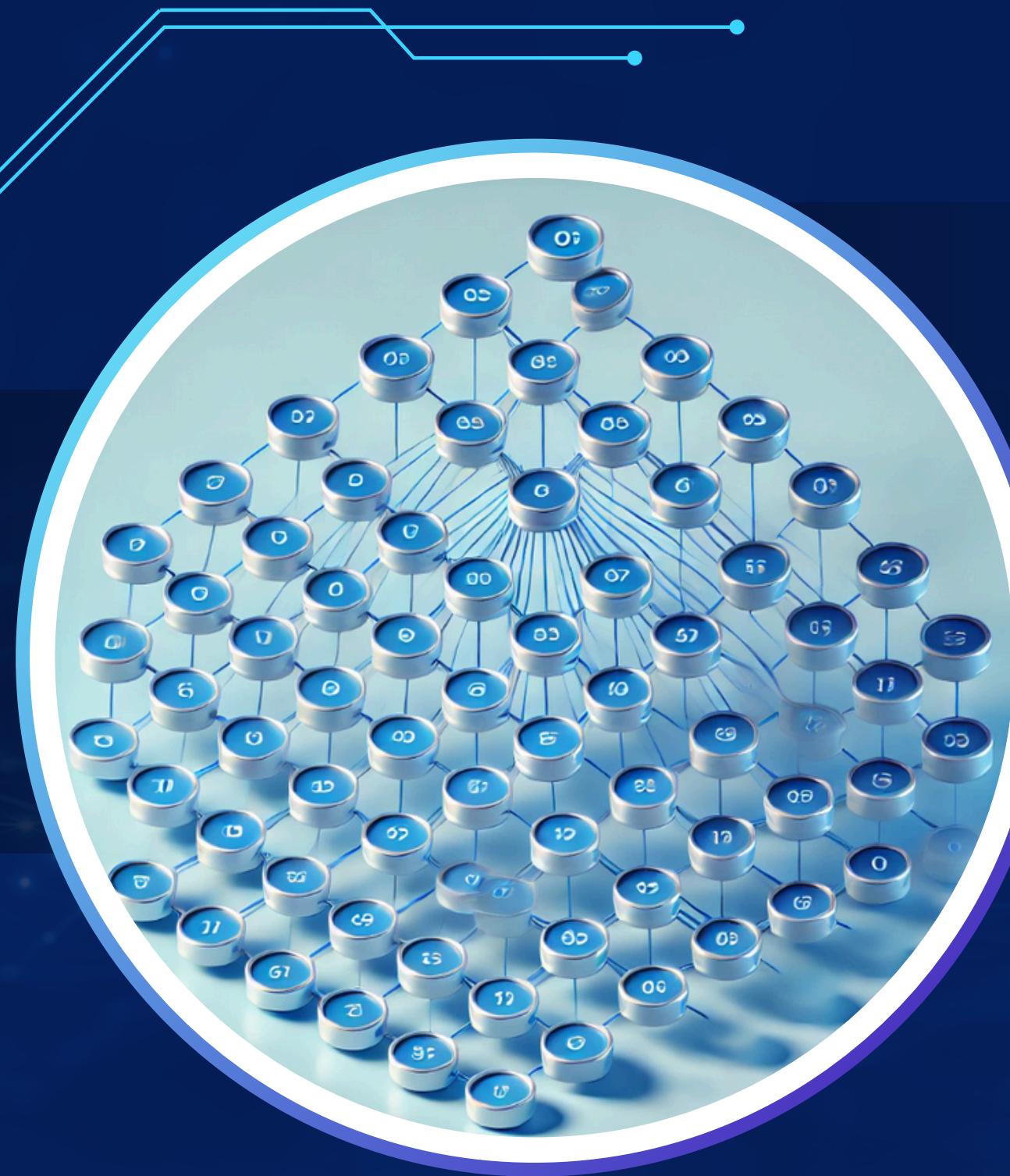




HEAPSORT



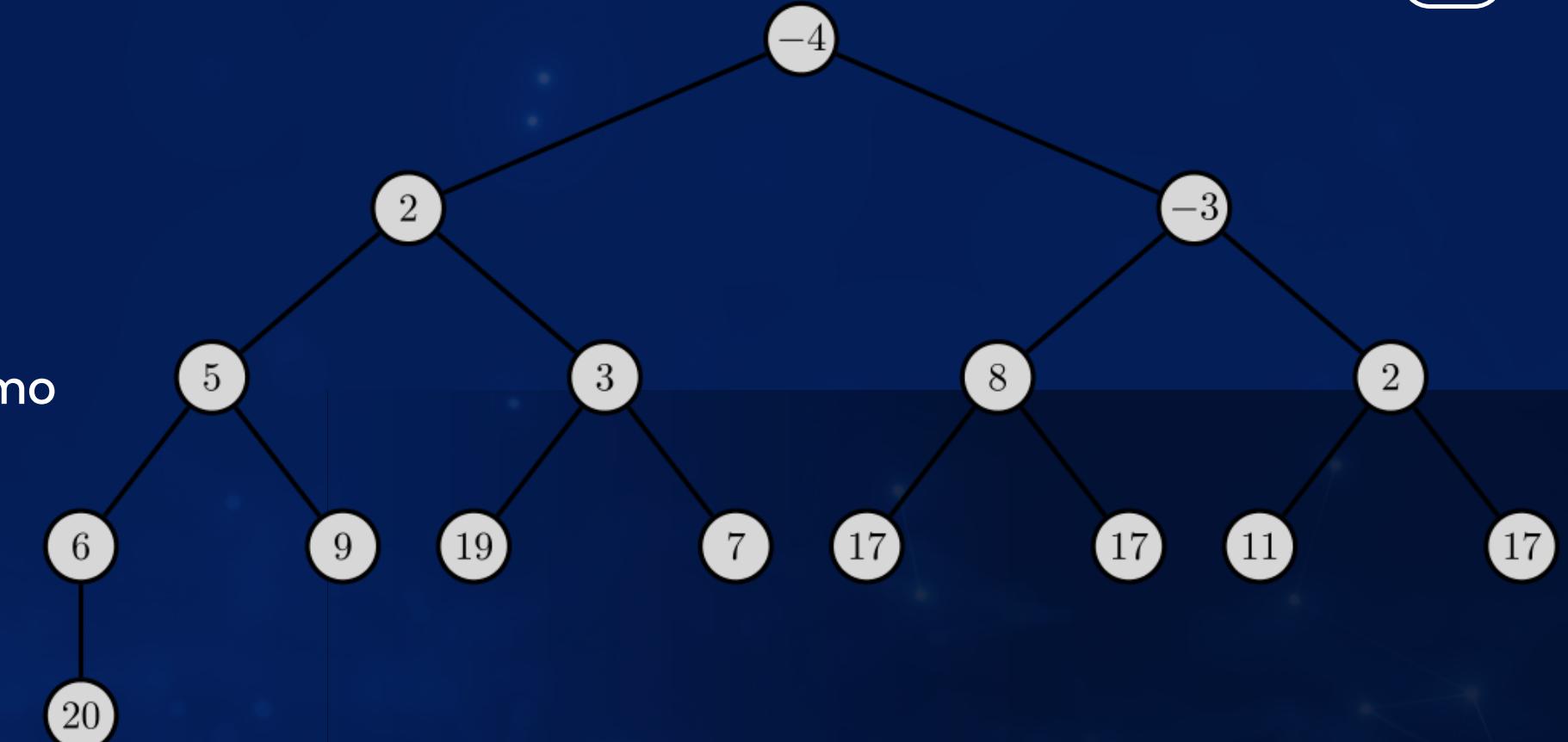


BINARY HEAP

Estrutura de dados que organiza elementos em uma árvore binária, o Binary Heap pode ser configurado como Min-Heap ou Max-Heap. Esse conceito é fundamental para o algoritmo Heap Sort, pois permite acesso rápido ao maior (ou menor) elemento, reorganizando a estrutura de maneira eficiente e acelerando o processo de ordenação.

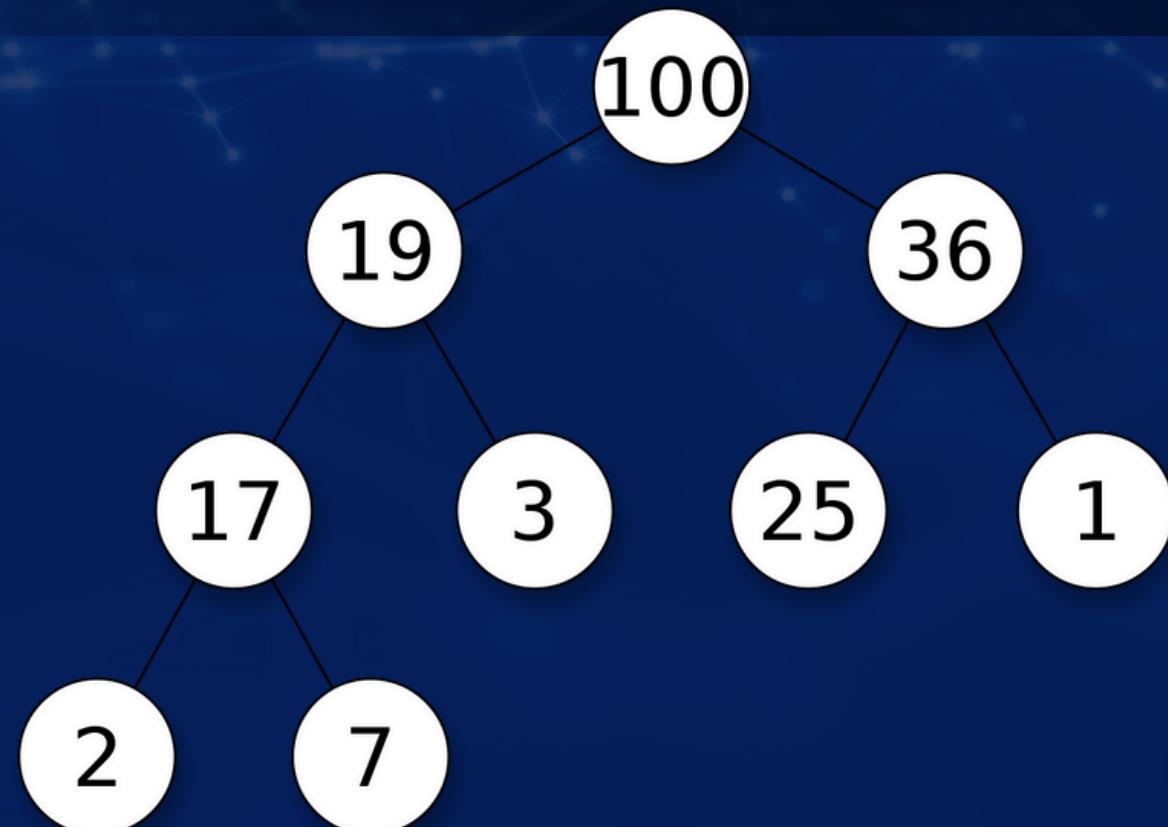
■ Min -heap

Em uma árvore para qualquer nó, os filhos sempre terão valor maior que o nó pai.. Quanto mais próximo da raiz, menor é o valor do elemento.



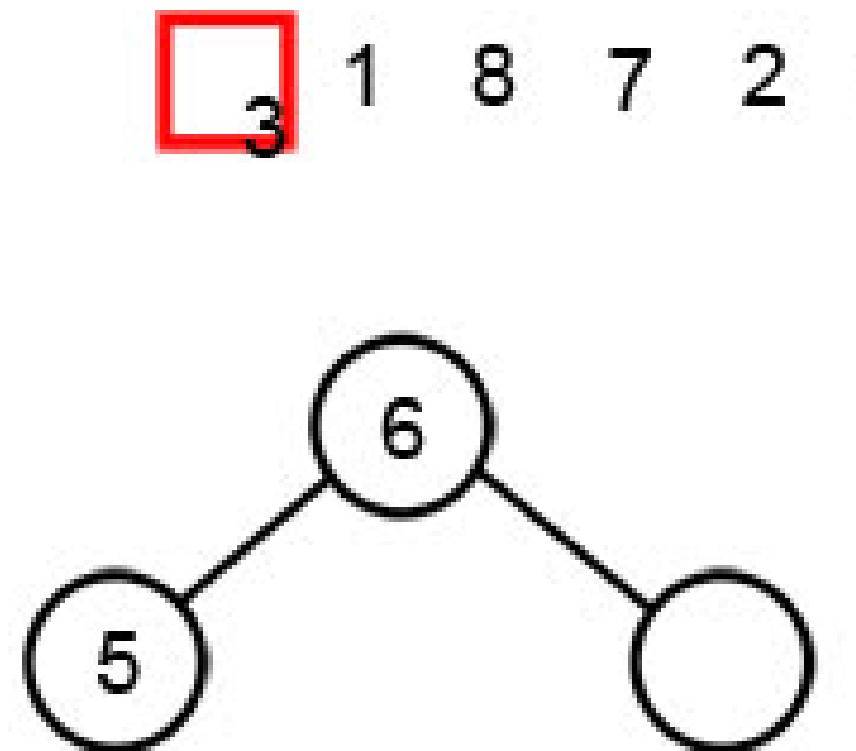
■ Max -heap

Em uma árvore para qualquer nó, os filhos sempre terão valor menor que o nó pai.. Quanto mais próximo da raiz, maior é o valor do elemento.



LÓGICA DE FUNCIONAMENTO

O heapsort é um método de ordenação que usa uma estrutura chamada "heap" (ou montinho), parecida com uma árvore. Ele organiza a lista para que o maior número fique no topo e, a partir daí, vai rearranjando os elementos para deixá-los em ordem.



ANÁLISE DE COMPLEXIDADE

■ Tempo

O heapsort organiza a lista em um montinho, removendo e reordenando os maiores números de forma eficiente, com um tempo consistente em qualquer situação.

■ Espaço

O heapsort é econômico, pois usa apenas a lista original e pouca memória extra para realizar a ordenação.



VANTAGENS

Tempo de ordenação consistente, uso eficiente de espaço, implementação simples.

Complexidade Consistente

O heapsort garante um tempo de ordenação bom em todos os casos (melhor, pior e médio), o que é útil em aplicações onde a previsibilidade é importante.

Uso de espaço

É um algoritmo in-place, ou seja, não requer muito espaço extra. Isso é vantajoso em ambientes com recursos limitados.

Simples de implementar

A lógica do heapsort é relativamente direta e pode ser implementada sem muitas complicações.

QUANDO UTILIZAR?

■ Limitação de memória

O heapsort usa pouca memória extra, pois realiza a ordenação na própria lista original. Ideal para sistemas com restrições de espaço.

■ garantir tempo de execução consistente

Como o tempo de execução do heapsort não muda com a ordem dos dados, ele é uma escolha eficiente em muitas situações

■ ordenação confiável para sistemas

Como o heapsort não depende da ordem inicial dos dados, ele é ideal quando o desempenho precisa ser constante, independente da entrada.



DESVANTAGENS

Não é estável, pode ser mais lento na prática, implementação pode ser menos intuitiva.

■ Estabilidade

não é um algoritmo estável, ou seja a ordem de elementos iguais pode ser alterada. Isso pode ser um problema em algumas aplicações.

■ Desempenho em prática

Embora sua complexidade seja boa, na prática, pode ser mais lento do que outros métodos como quicksort ou mergesort devido a fatores como acesso à memória.

■ Estrutura de Dados

pode ser menos intuitivo do que outros métodos de ordenação, e sua implementação pode ser mais complexa do que, por exemplo, o bubble sort ou insertion sort em pequenos conjuntos de dados.

QUANDO NÃO UTILIZAR?

■ Listas pequenas ou quase ordenadas

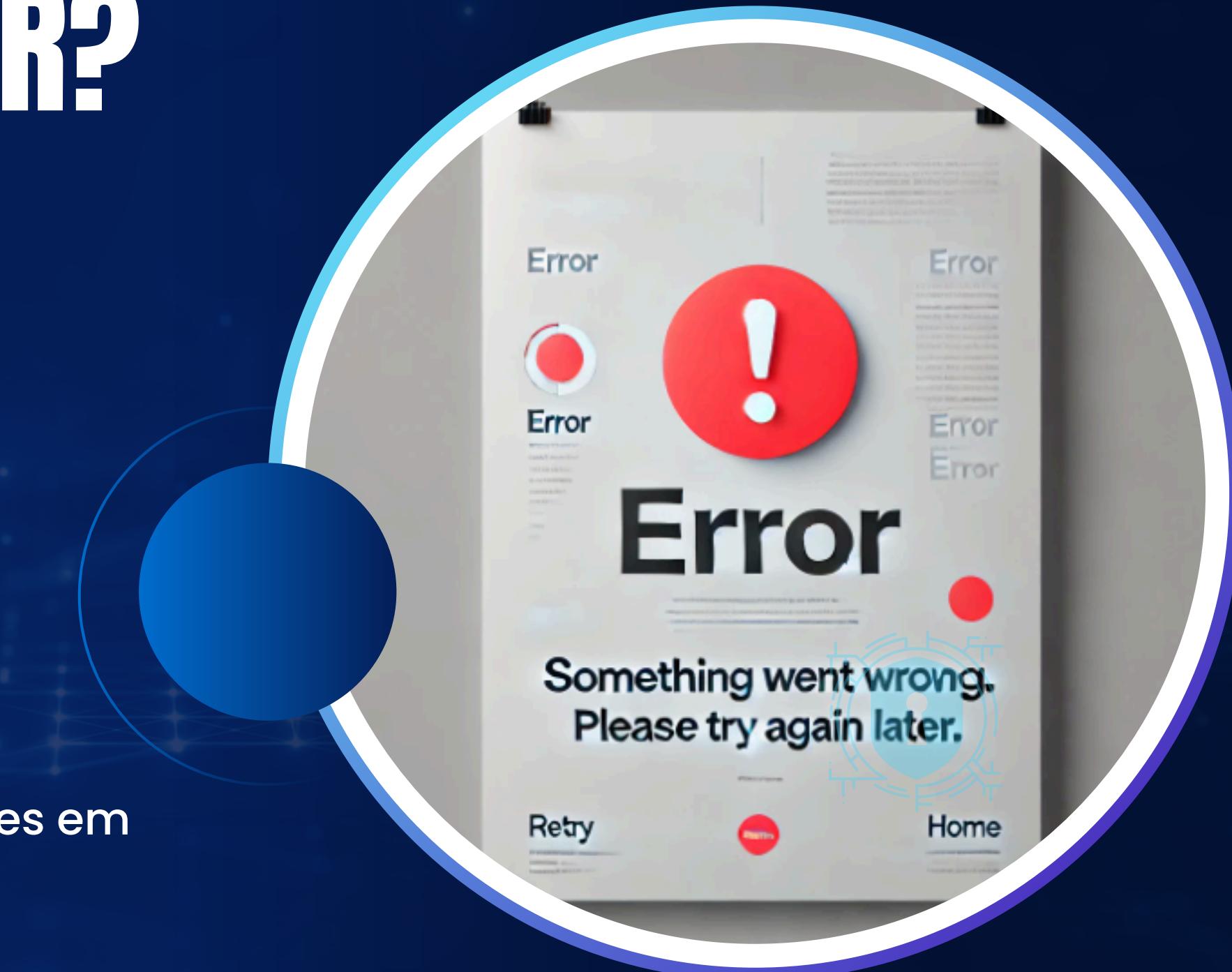
Métodos como quicksort ou insertion sort podem ser mais rápidos.

■ precisa de ordenação estável

não preserva a ordem de itens iguais, então, para dados que precisam de estabilidade, o mergesort é uma opção melhor.

■ Busca desempenho máximo para listas grandes em média

O quicksort, em geral, é mais rápido para grandes listas, quando a variação no tempo não é um problema.





OBRIGADO

alunos

Caio Monte Belo Rocha Segal,
Filipe Vieira Rocha,
Guilherme Paulo Miranda
João Pedro Estevão Louback,
Maria Clara Oliveira Estavanovic Moura
Paulo Victor Rangel da Silva Cruz,
Renan Libardi Rezende