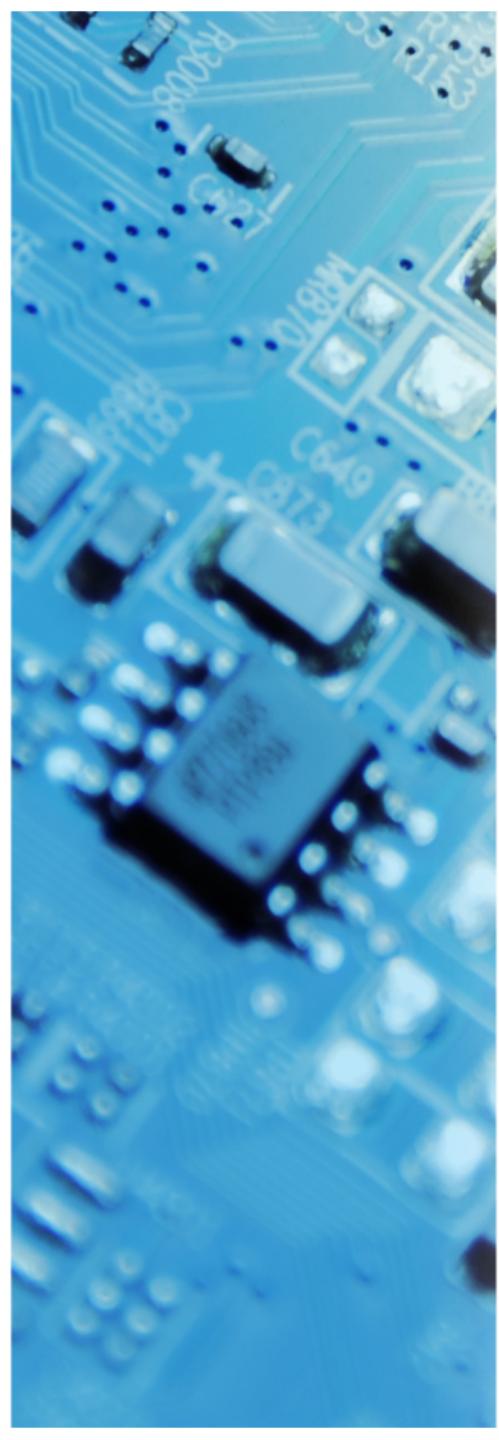
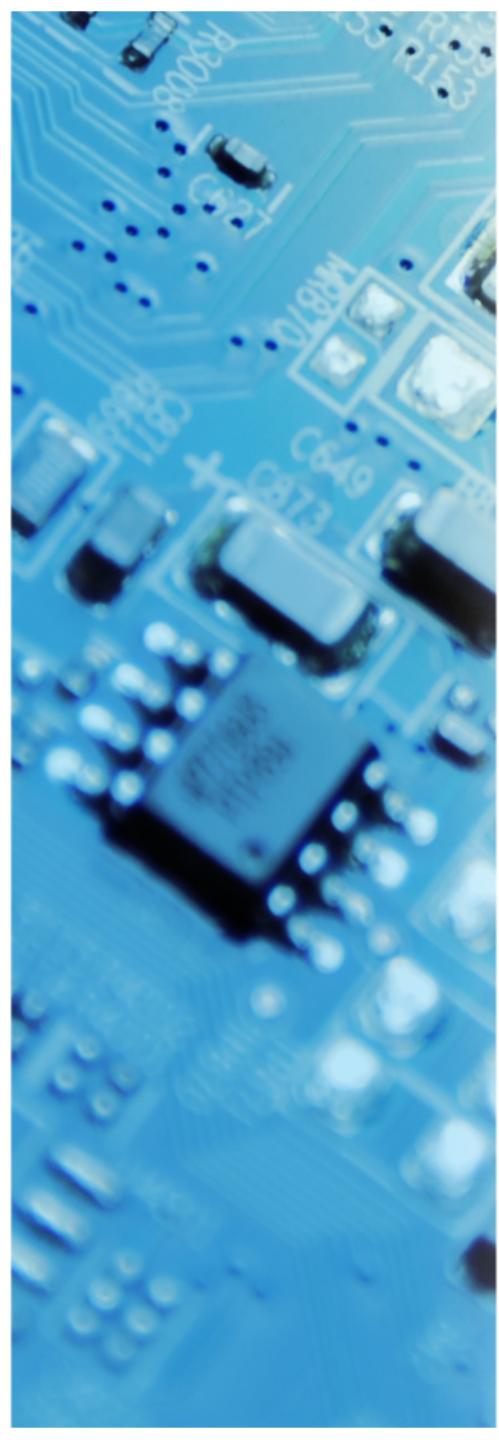


Métodos de ordenação



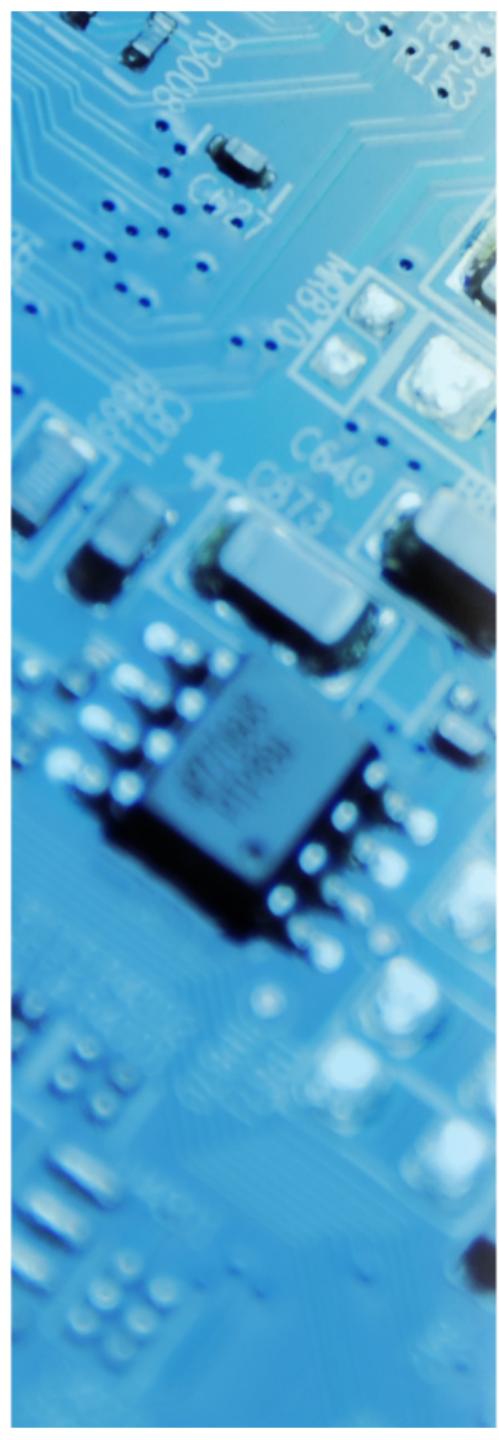
Métodos de ordenação

- A ordenação é uma operação comum em conjuntos de dados.
- Um conjunto de dados pode ser uma lista, vetor ou qualquer outro conjunto de objetos.
- “Ordenação é o ato de se colocar os elementos de uma sequência de informações, ou dados, em uma relação de ordem predefinida. O termo técnico em inglês para ordenação é sorting, cuja tradução literal é ‘classificação’” (WIKIPÉDIA, 2021, texto digital).



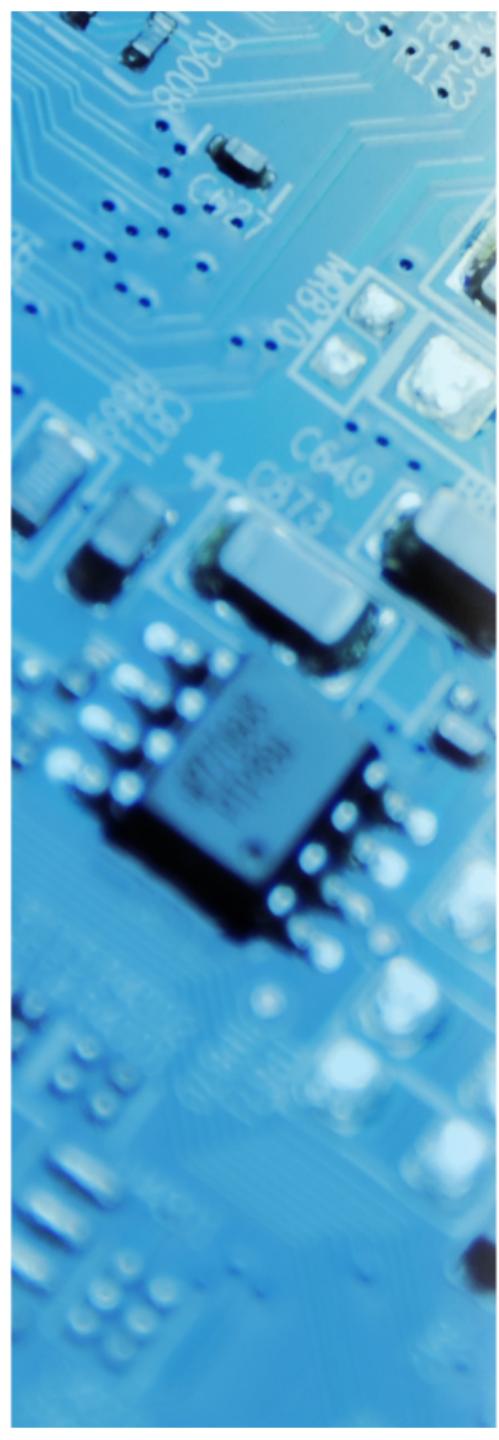
Métodos de ordenação

- Na computação a ordenação é essencial e pode, em alguns casos reduzir a complexidade de um problema.
- Existem diversos algoritmos que podem ser usados para realizar a ordenação.
- Ao longo dos anos foram propostos diferentes métodos e alguns deles foram sendo aperfeiçoados.
- Cada método é avaliado seguindo critérios de eficiência e desempenho.



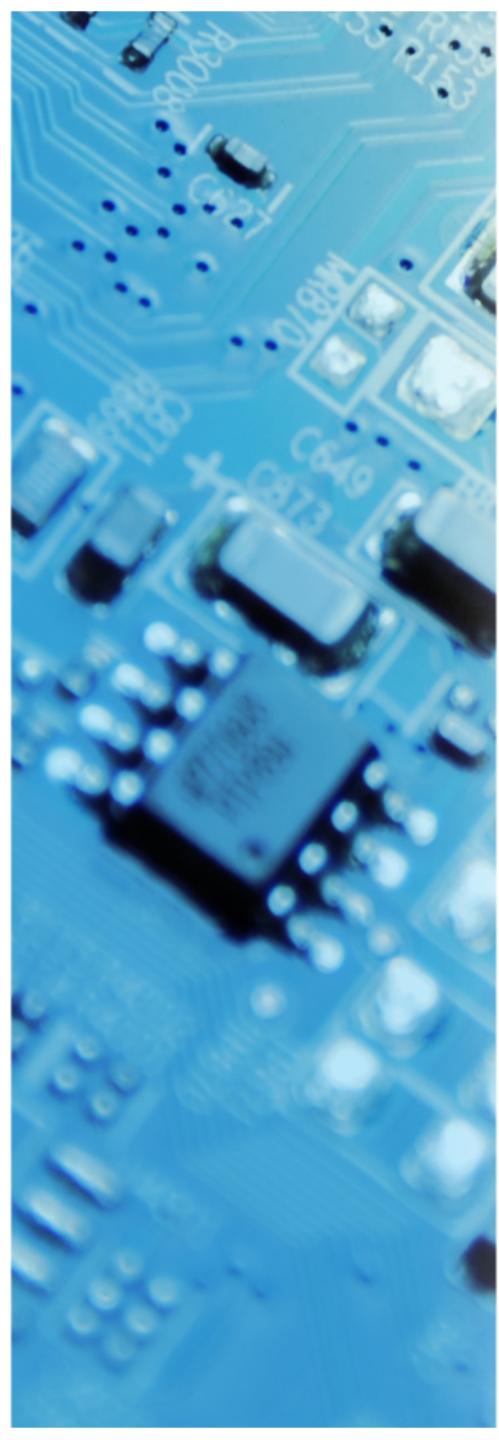
Métodos de ordenação

- A escolha de um método envolve diversas questões, entre elas:
 - Qual é o tamanho da coleção que está sendo classificada?
 - Quanta memória está disponível?
 - A coleção irá aumentar de tamanho?
 - Quais os tipos dos dados armazenados?



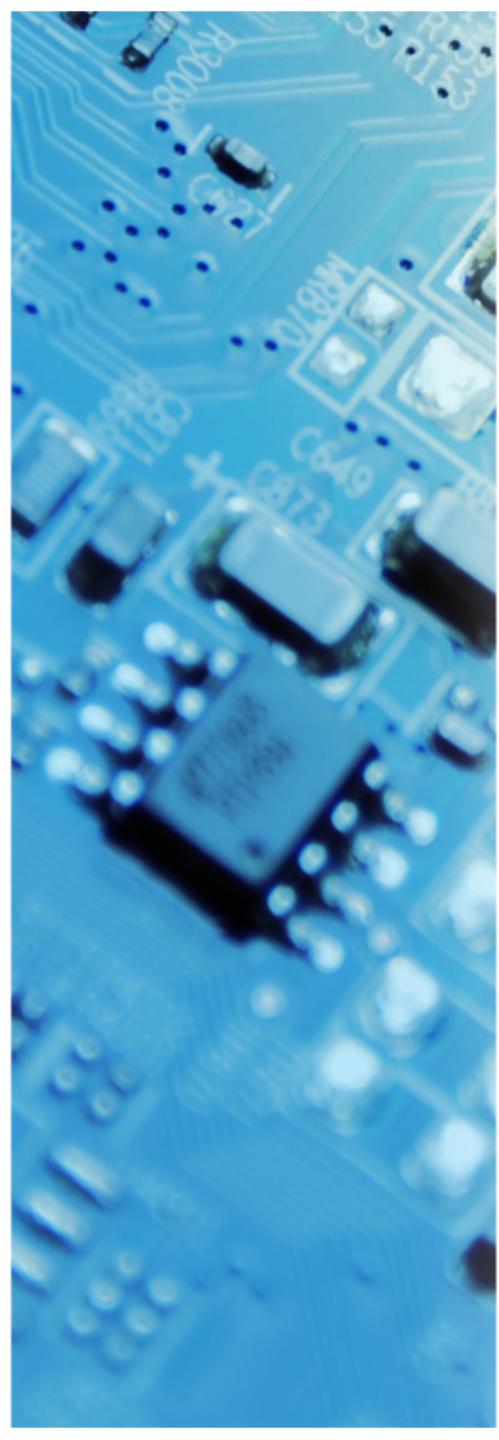
Métodos de ordenação

- Problema:
 - Dada uma sequência de valores, por exemplo, [4, 2, 3, 0, 9, 3, 5, 7], é necessário colocar os mesmos em ordem crescente.
 - O método de ordenação, recebe o array com $n = 8$ e deve trocar os valores de posição de modo a conseguir a seguinte situação final para o array: [0, 2, 3, 3, 4, 5, 7, 9]



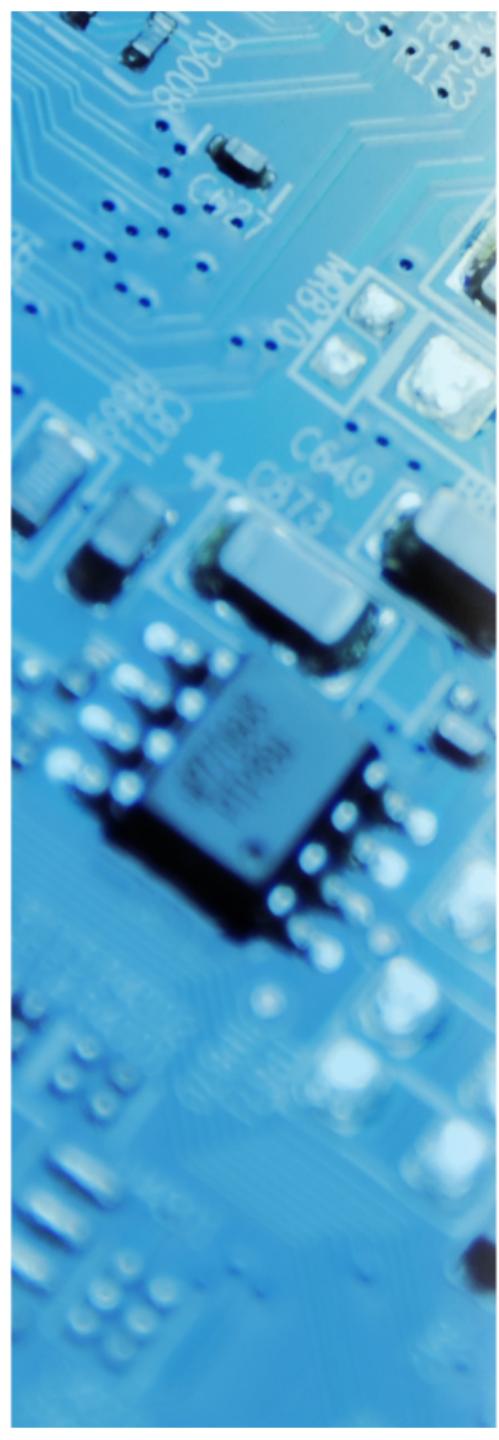
Métodos de ordenação

- Como a classificação é muito importante, ela foi estudada intensivamente e muitos algoritmos foram desenvolvidos.
- Alguns desses algoritmos são adaptações de esquemas que usamos na vida cotidiana.
- Um exemplo é a classificação de cartas em uma mão.
- É possível ir da esquerda para a direita e colocar cada carta sucessivamente em sua posição correta.
- Esse é o princípio do Insertion sort.



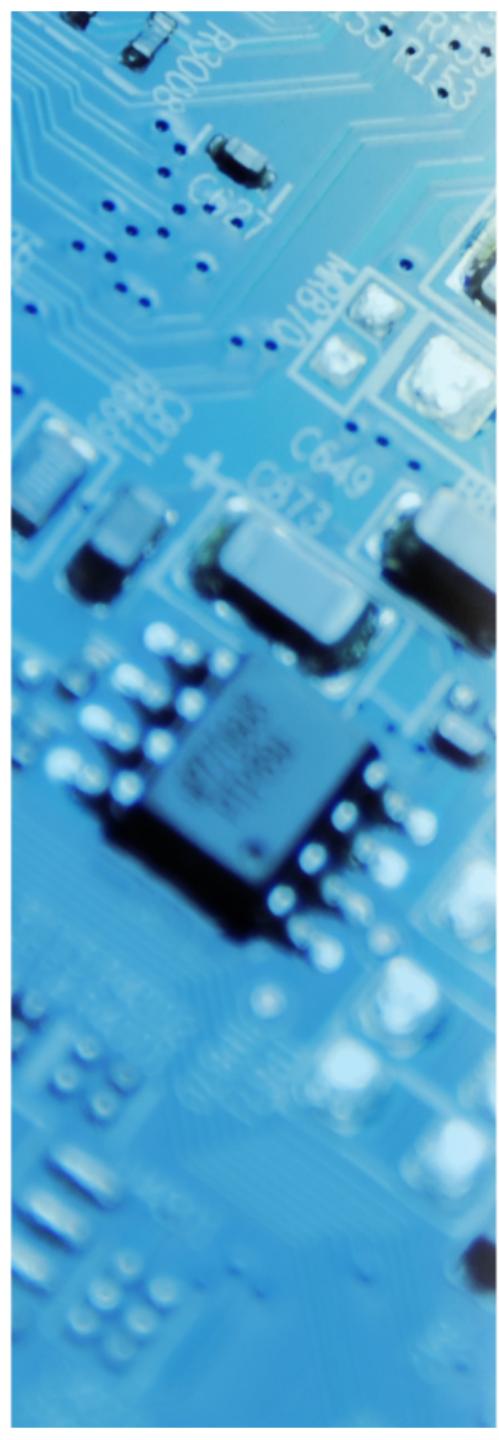
Métodos de ordenação

- Alguns métodos são estranhos à maneira como os humanos fazem as coisas.
- Foram propostos para classificar milhares ou até milhões de registros armazenados no computador.
- Um exemplo é o Quicksort que nenhum humano usaria para ordenar documentos pessoais, por exemplo.
- Porém, o Quicksort é na maioria dos casos o algoritmo de classificação padrão escolhido para a maioria das bibliotecas de software.



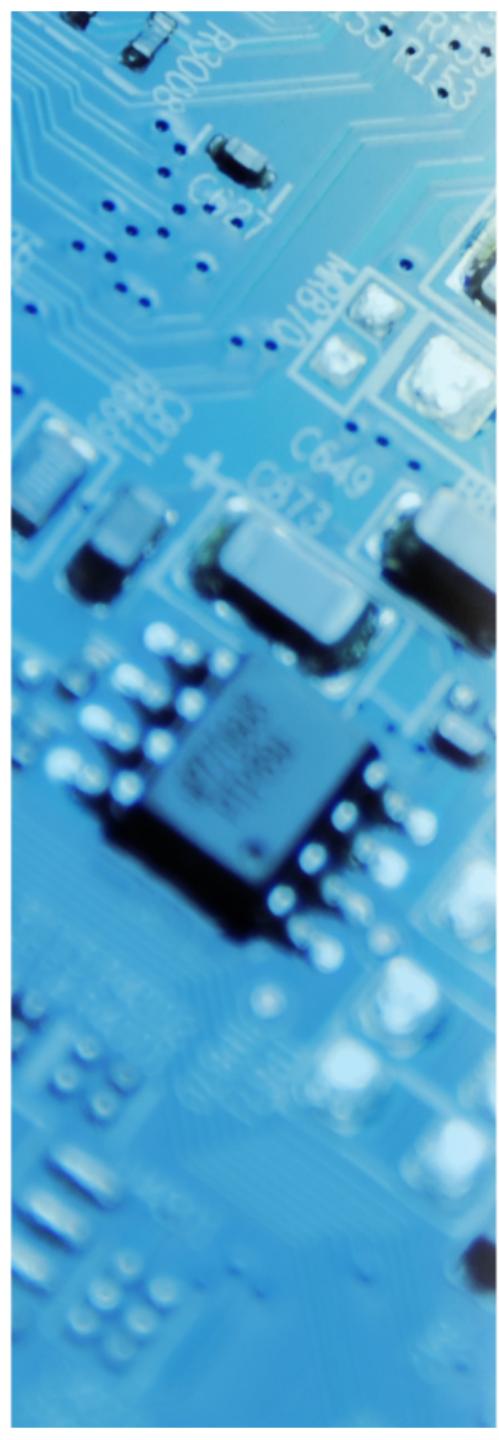
Métodos de ordenação

- Após anos de estudo sobre estes métodos, ainda existem problemas não resolvidos relacionados à classificação.
- Novos algoritmos ainda estão sendo desenvolvidos e refinados para aplicações especiais.
- O estudo dos algoritmos de classificação permite compreender os problemas enfrentados no projeto e análise de algoritmos.
- O entendimento destes problemas também contribui significativamente para aumentar a capacidade de resolver problemas computacionais.



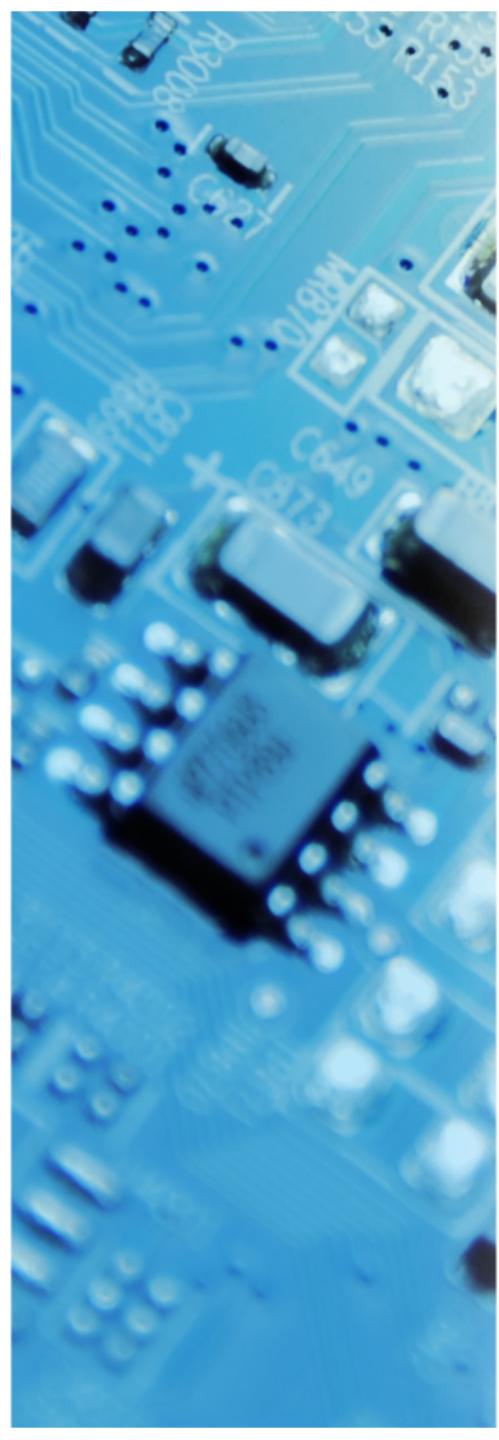
Comparação de métodos

- Alguns métodos permitem a existência de valores com chaves duplicadas, outros não.
- Em algumas situações há uma ordem padrão dos valores duplicados, que deve ser mantida.
- Algoritmos que mantém esta ordem padrão para valores duplicados são definidos como estáveis.
- A comparação entre dois métodos normalmente é feita executando cada um sobre os mesmos dados, medindo o tempo.
- Esta comparação é denominada empírica.



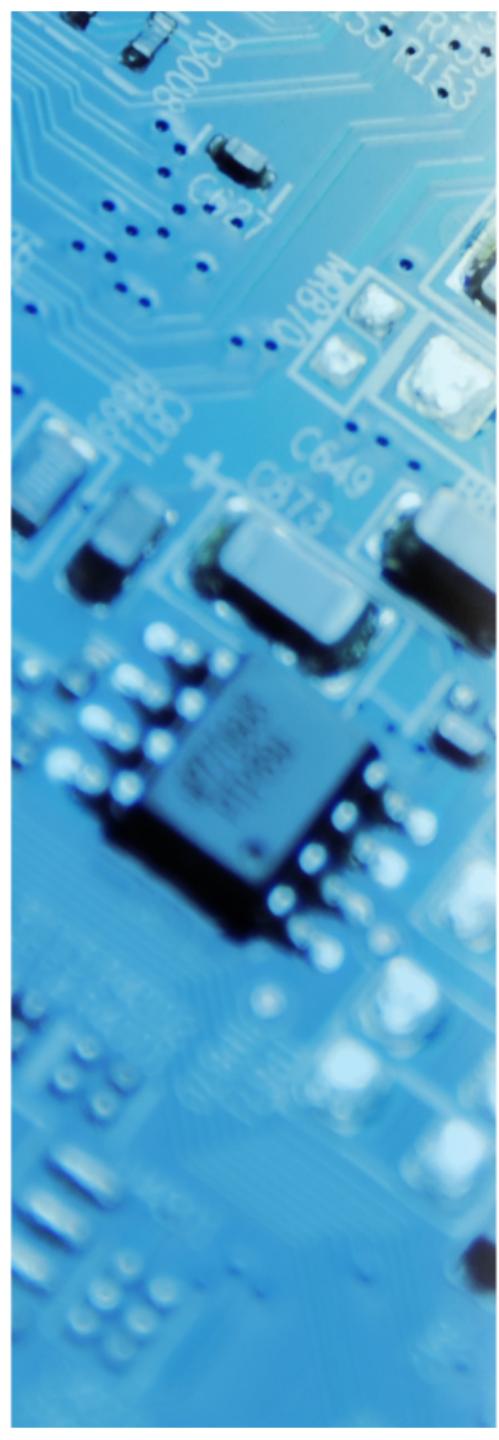
Comparação de métodos

- Comparações empíricas podem ser suscetíveis à variações nos dados, na organização inicial e nos tipos de dados que devem ser classificados.
- Outra forma de analisar é comparar os números de trocas feitos por cada algoritmo.
- Casos em que chaves podem ter tamanhos diferentes, exigem estratégias especiais de comparação.
- Alguns métodos usam uma estratégia in-place (ordenação no local) e outros out-of-place (fora do local).



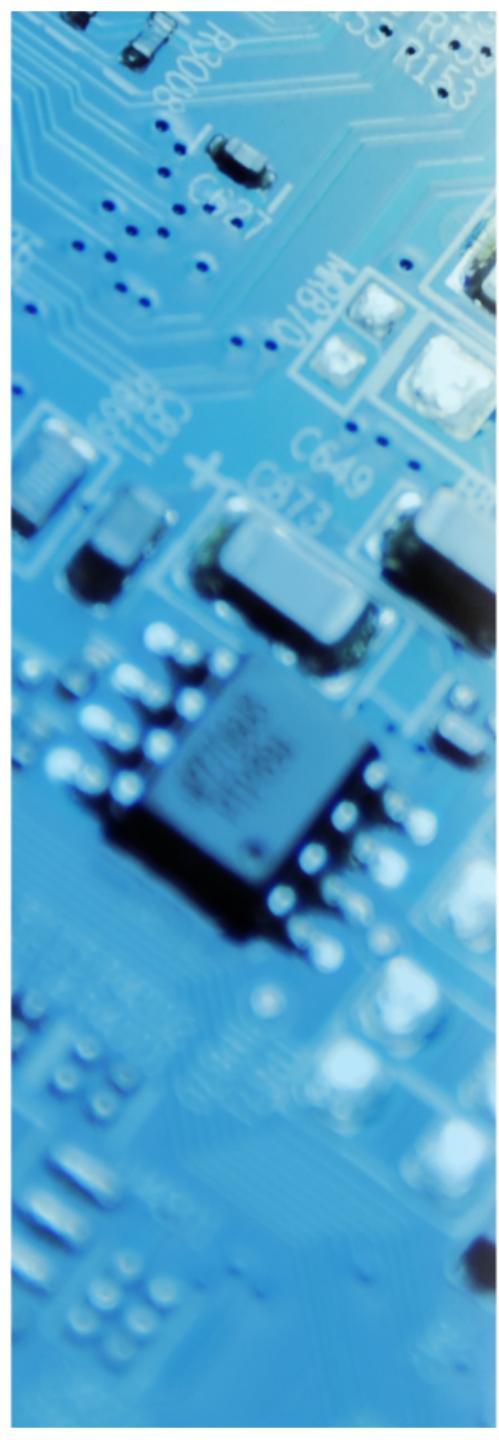
Bubble sort

- O método da bolha é o mais utilizado para aprendizado e desenvolvimento da lógica referente à implementação de algoritmos de ordenação.
- Método simples que envolve movimentação direta e objetiva dos dados, porém, é um dos que apresenta pior desempenho.
- Por este motivo é mais adotado para ensino e compreensão dos conceitos e menos em aplicações reais.
- Baseado no princípio que uma bolha irá subir e estourar na superfície.

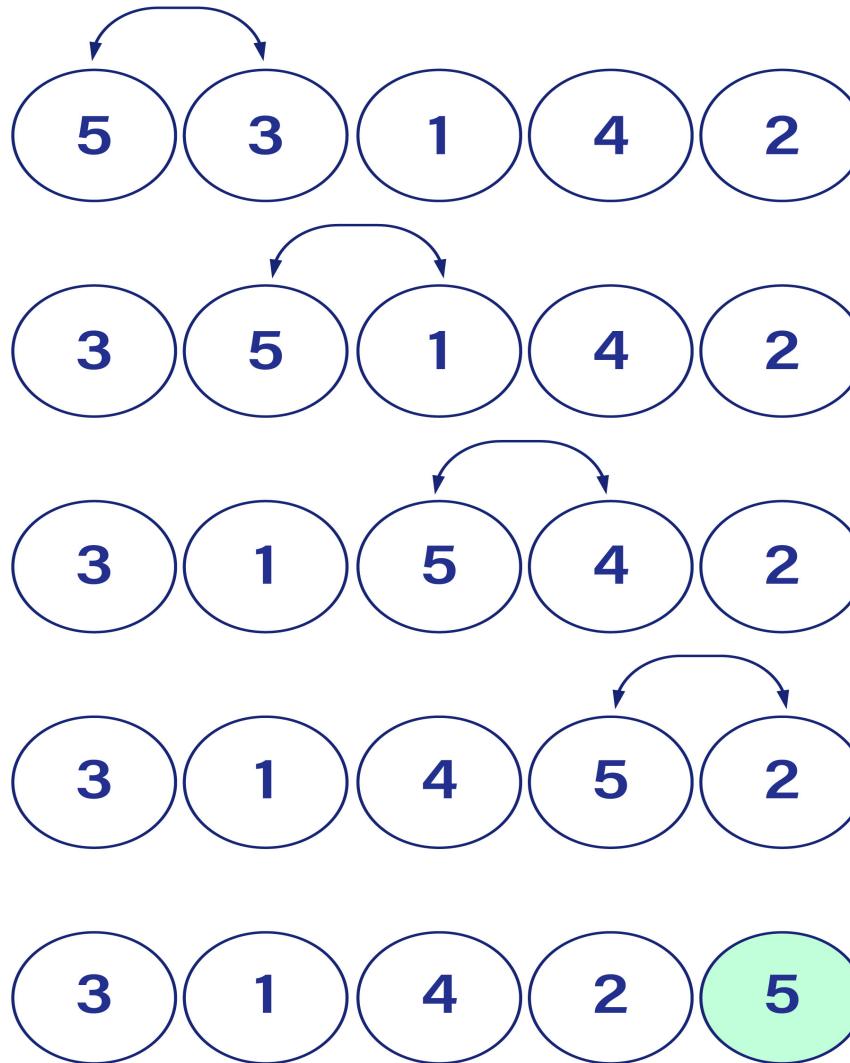


Bubble sort

- O princípio de funcionamento do Bubble sort está na comparação e troca de valores entre posições consecutivas, ou seja, posições adjacentes entre os elementos comparados.
- Os valores mais altos (ou mais baixos) “borbulham” para o final da lista ou array.
- Começa no início da lista e compara pares de itens de dados à medida que alcança o final.
- Sempre que os itens no par estão fora de ordem, o algoritmo faz a troca.
- Repete o processo do início da lista e passa para o penúltimo item e assim por diante.



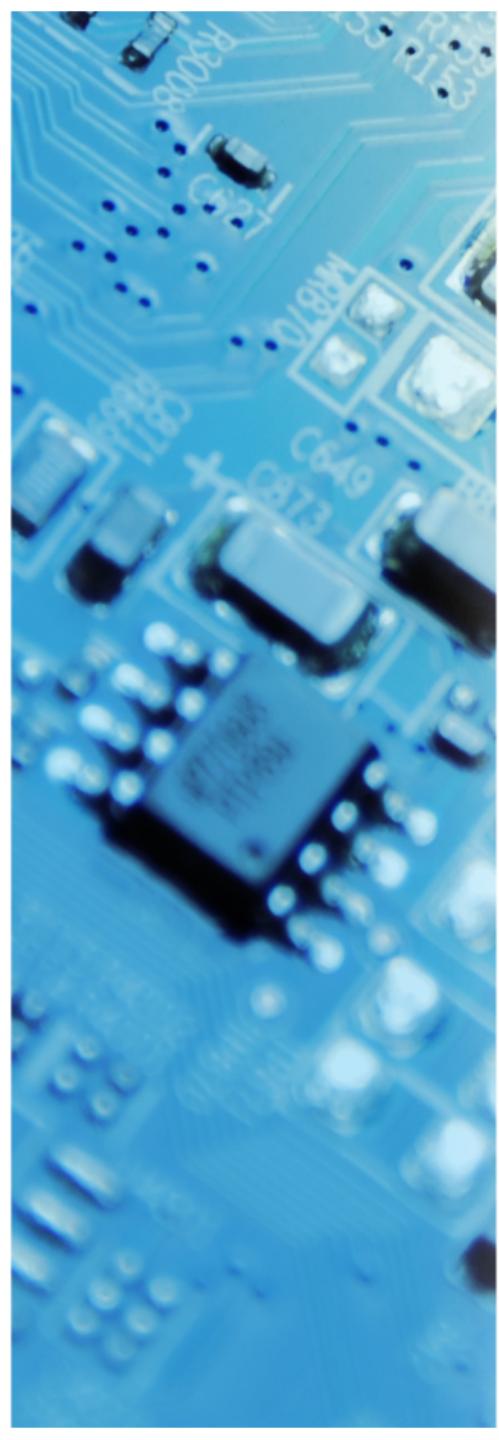
Bubble sort



<https://opendsa-server.cs.vt.edu/embed/bubblesortAV>

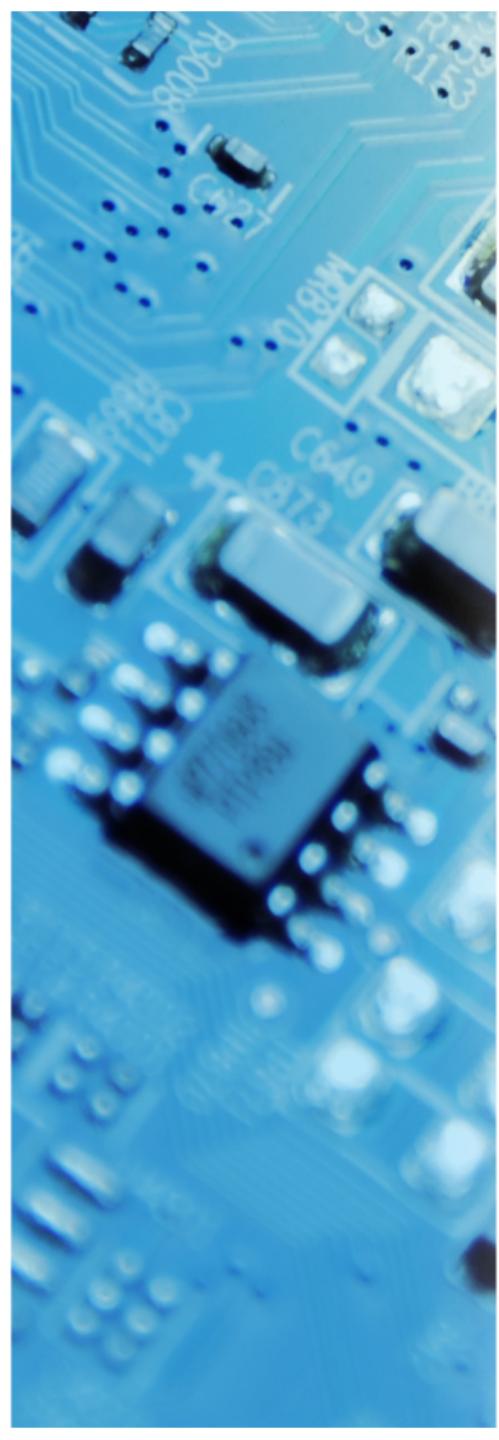
<https://visualgo.net/en>

Adaptado de Rodrigues et al. (2021).



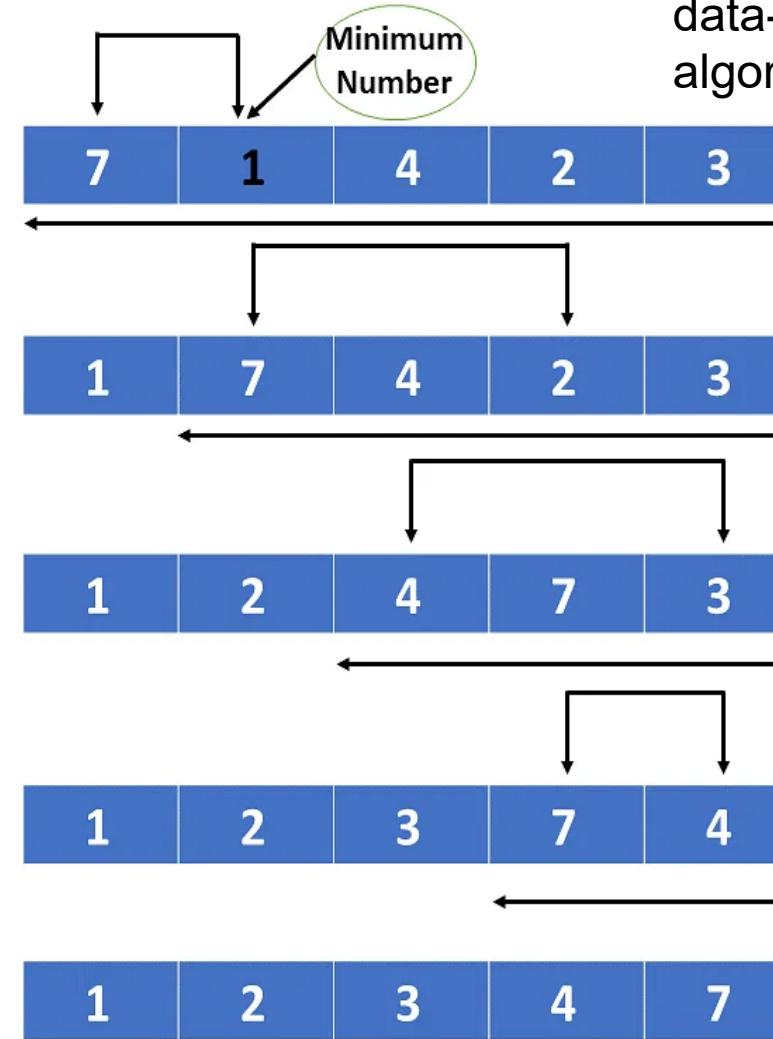
Selection sort

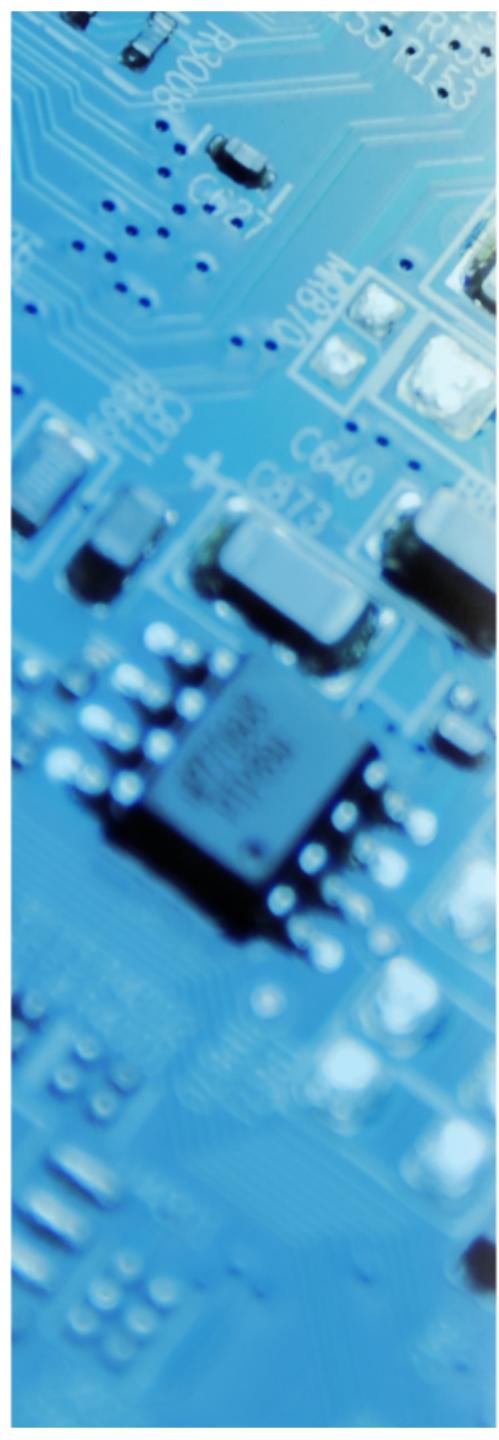
- Baseado em um ideia de que há varias cartas na mesa e é possível escolher a menor para inserir diretamente na posição correta.
- Inicia pelo primeiro valor armazenado (posição de destino) e busca no restante do vetor o menor elemento.
- Troca de posição com o valor da posição inicial.
- Avança para a posição seguinte e a partir desta repete o processo buscando no restante do vetor qual o menor valor para inserir na posição que é o destino atual.
- Neste caso o menor vai sempre ir para a sua posição final.
- A partir da posição de destino vai pesquisar todas as restantes, à direita.



Selection sort

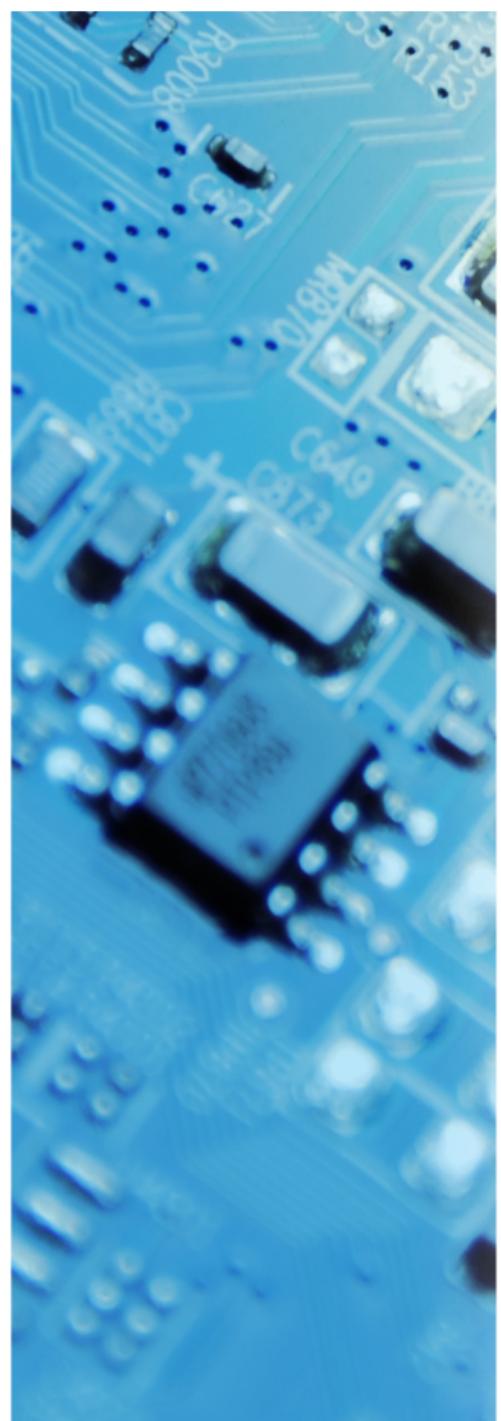
<https://www.simplilearn.com/tutorials/data-structure-tutorial/selection-sort-algorithm>





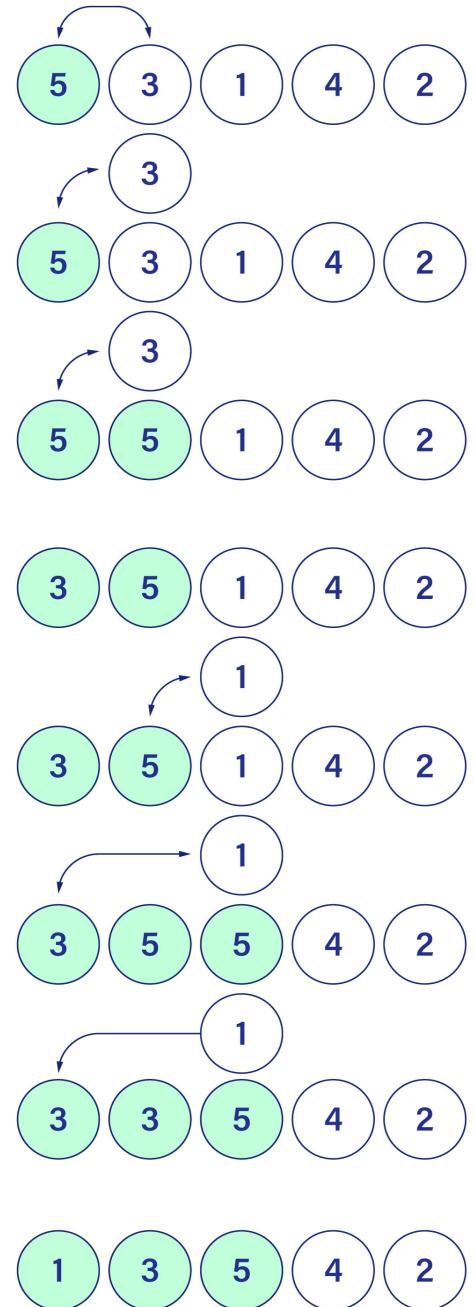
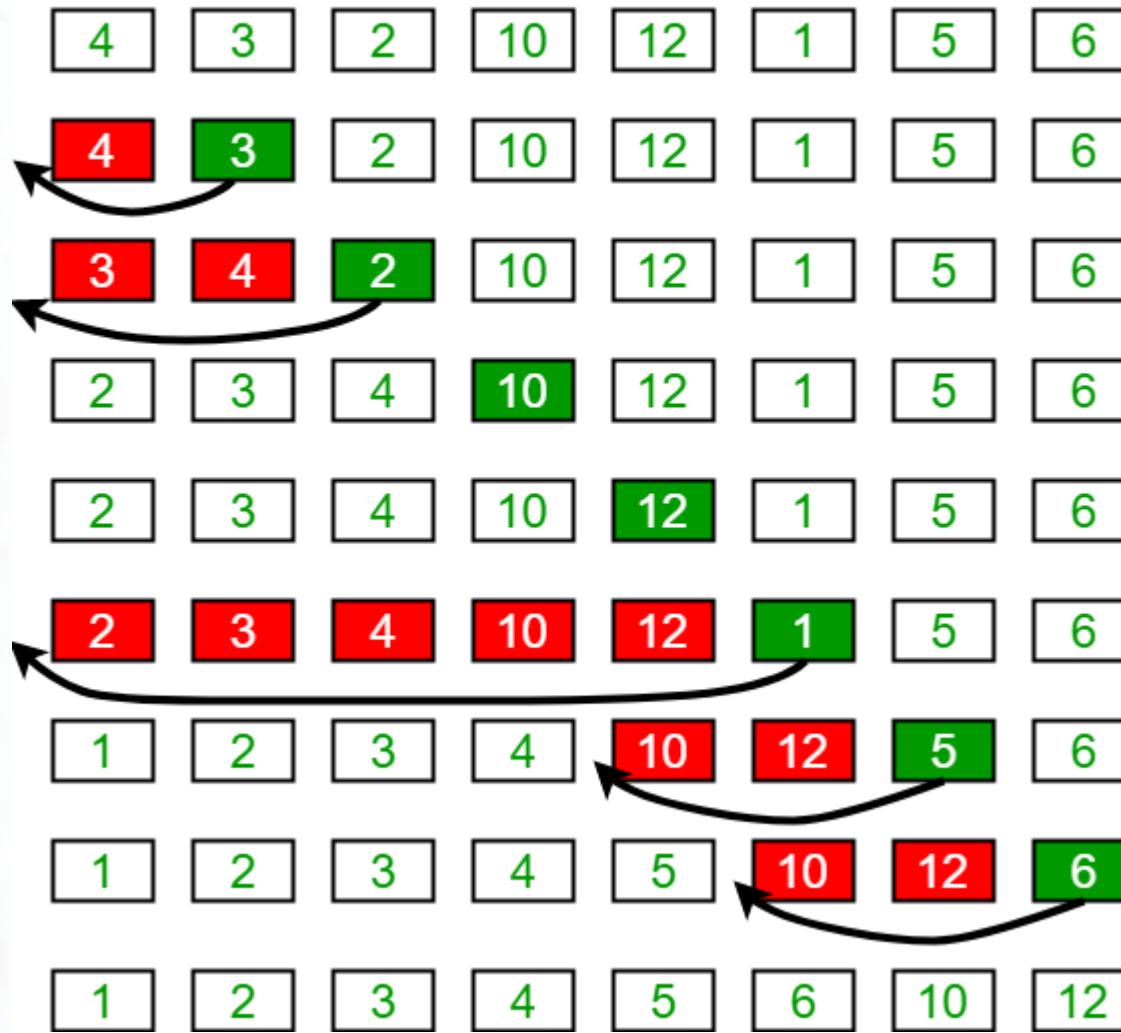
Insertion sort

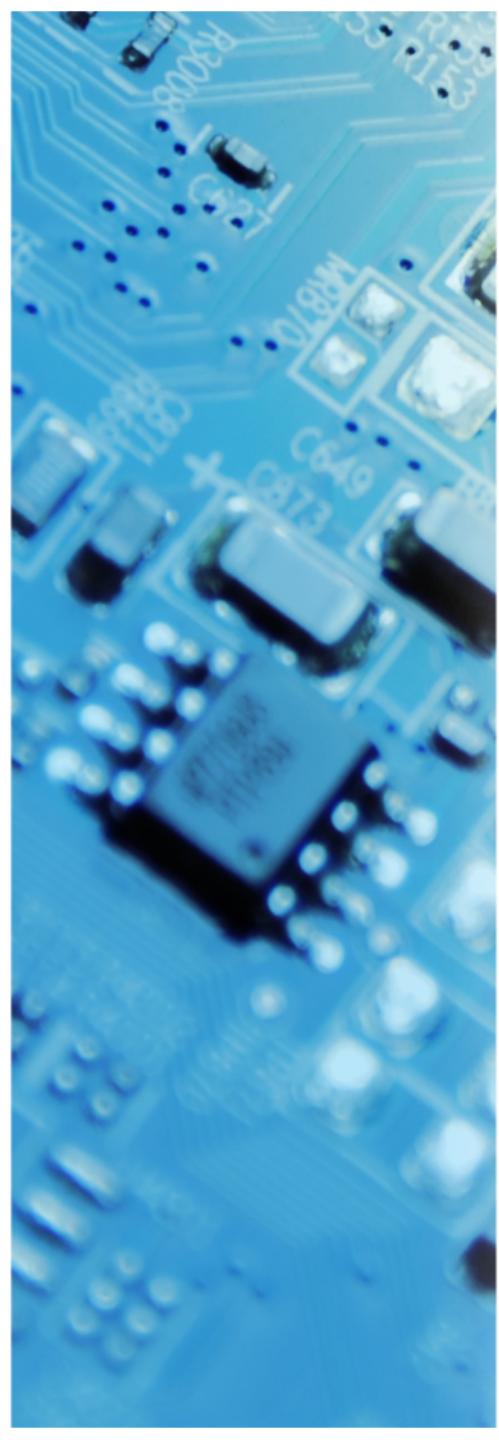
- Insere os dados um a um na posição correta de destino.
- Método eficiente, principalmente para conjuntos pequenos de dados.
- Define qual o próximo valor que será classificado, ou seja, inserido na sua posição correta.
- Inicia pelo segundo elemento e vai incrementando esta posição a cada passo.
- O elemento que será classificado é comparado com os anteriores para ver em que posição ele será inserido.
- Move todos os valores da posição que será inserido o elemento para a posição seguinte até ocupar a posição original do valor que está sendo classificado.



Insertion sort

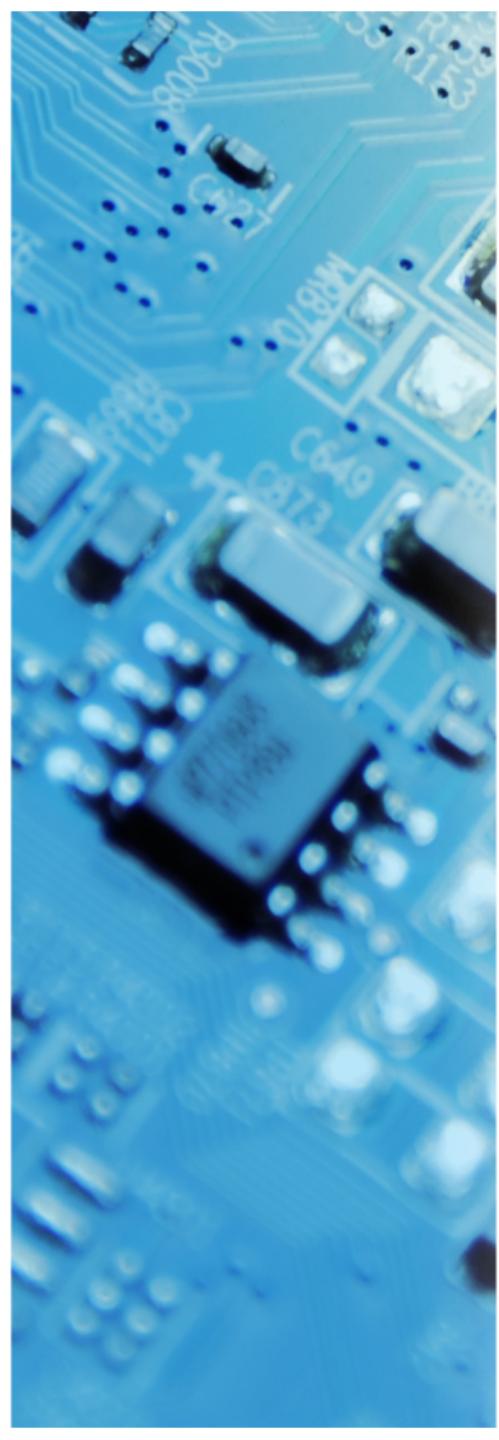
Insertion Sort Execution Example





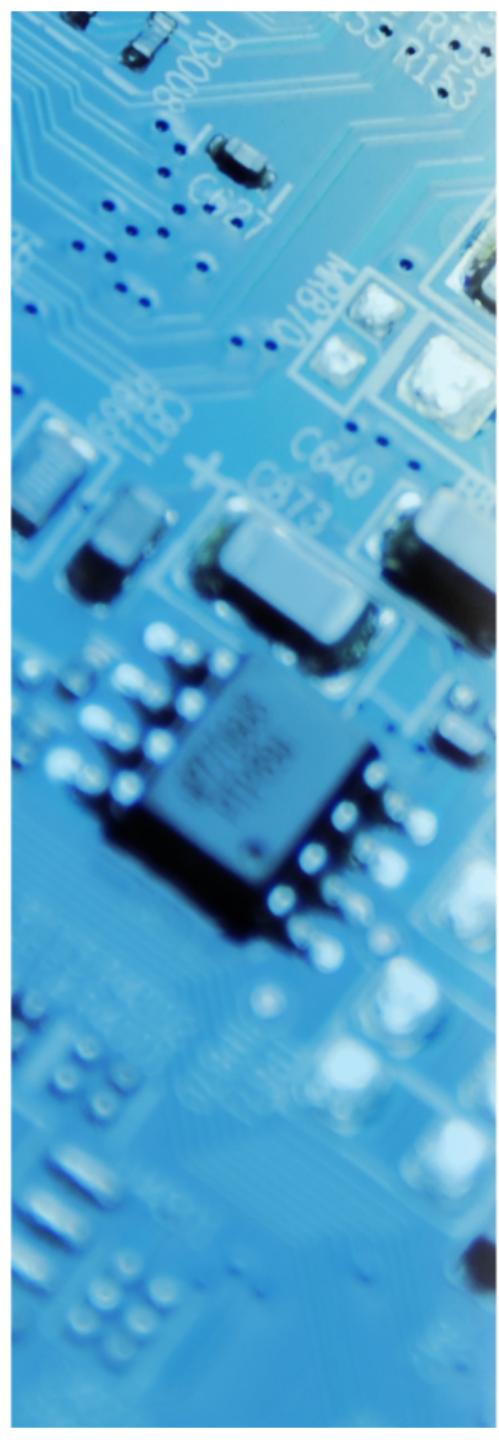
Quick sort

- Algoritmo mais utilizado, considerando mais eficiente na maiorias das situações.
- Baseado na ideia de dividir para conquistar, com ordenações de partes menores.
- Um dos elementos é selecionado como o pivô.
- Uma abordagem possível é escolher o elemento do meio, da posição central do vetor ou até mesmo o primeiro valor da lista.
- A escolha deste elemento é importante para garantir um melhor desempenho na classificação.

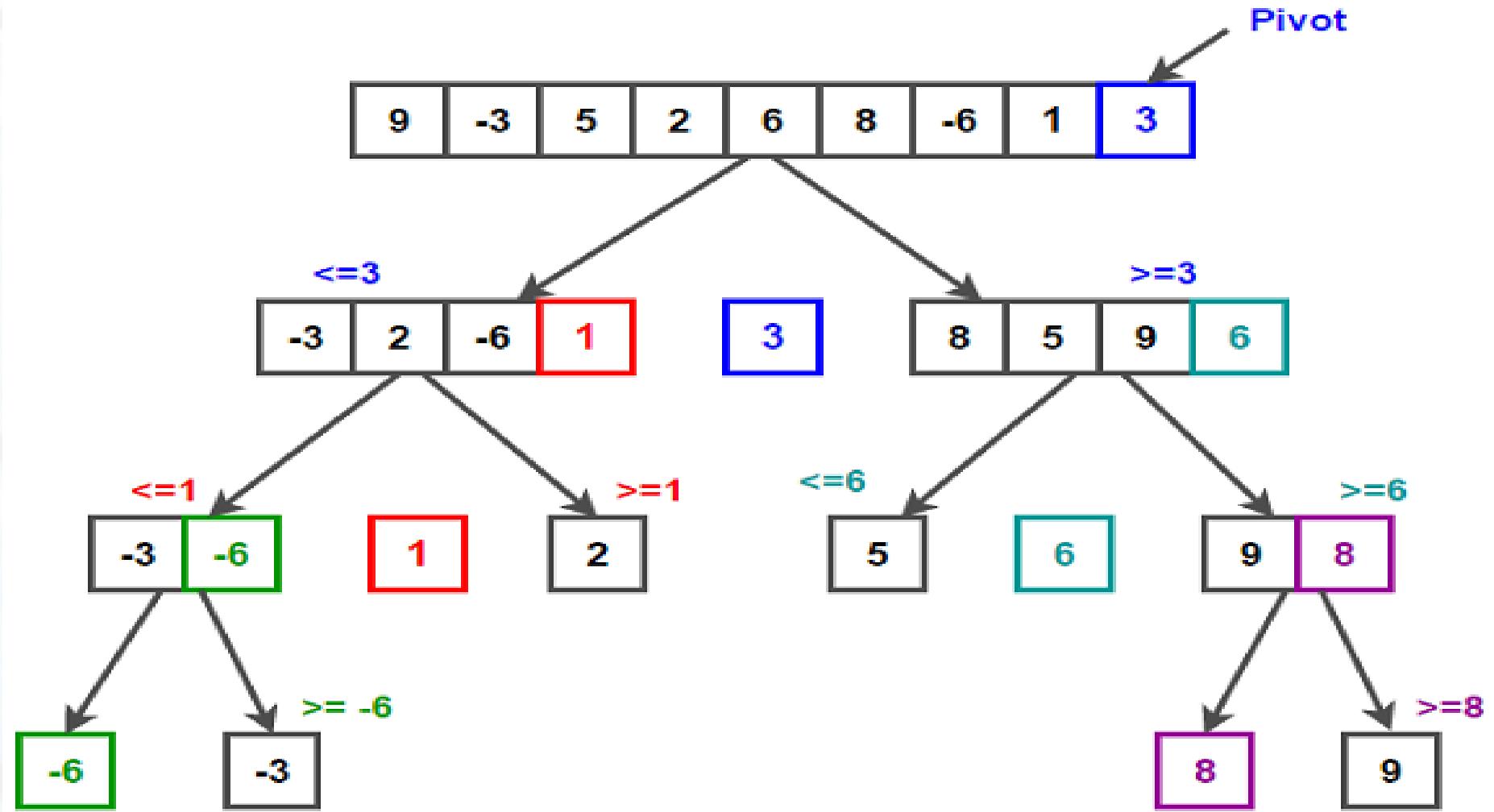


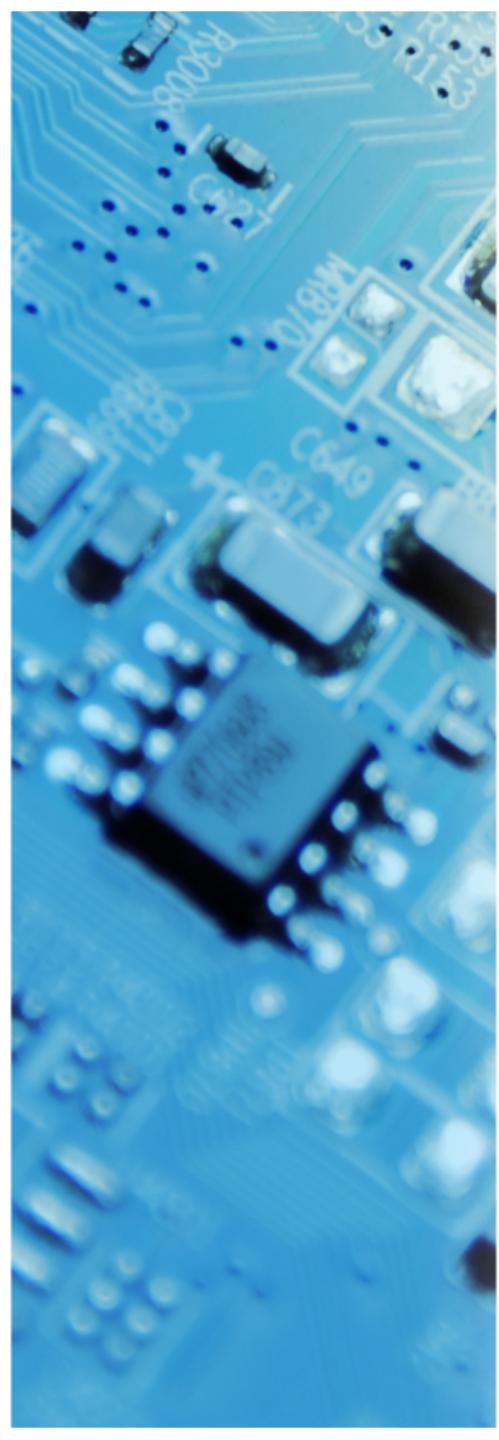
Quick sort

- A divisão nem sempre gera subvetores do mesmo tamanho, isso depende da área alocada e da escolha do pivô.
- Na maioria das implementações utiliza chamadas recursivas que consistem em aplicar o mesmo algoritmo nos subvetores resultantes da divisão.
- Em um exemplo possível busca todos os elementos da metade inicial e aplica o método nestes valores, em seguida aplica o método novamente nos valores da segunda metade.
- Outra abordagem consiste em criar um subvetor com os elementos que tem valor menor do que o valor do pivô e outro com os valores maiores.



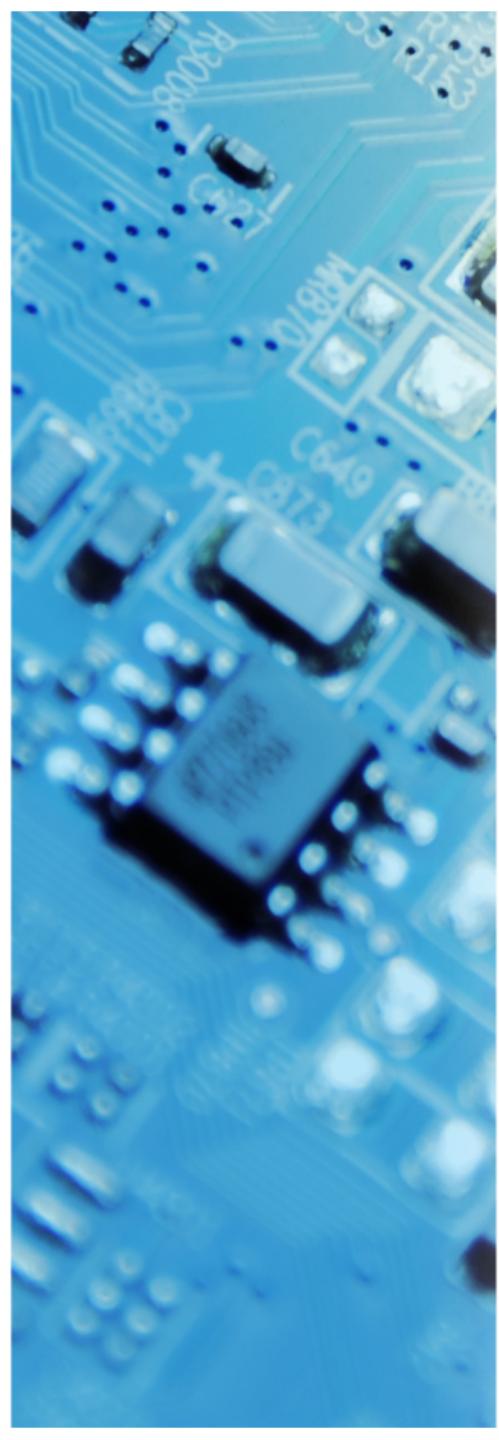
Quick sort





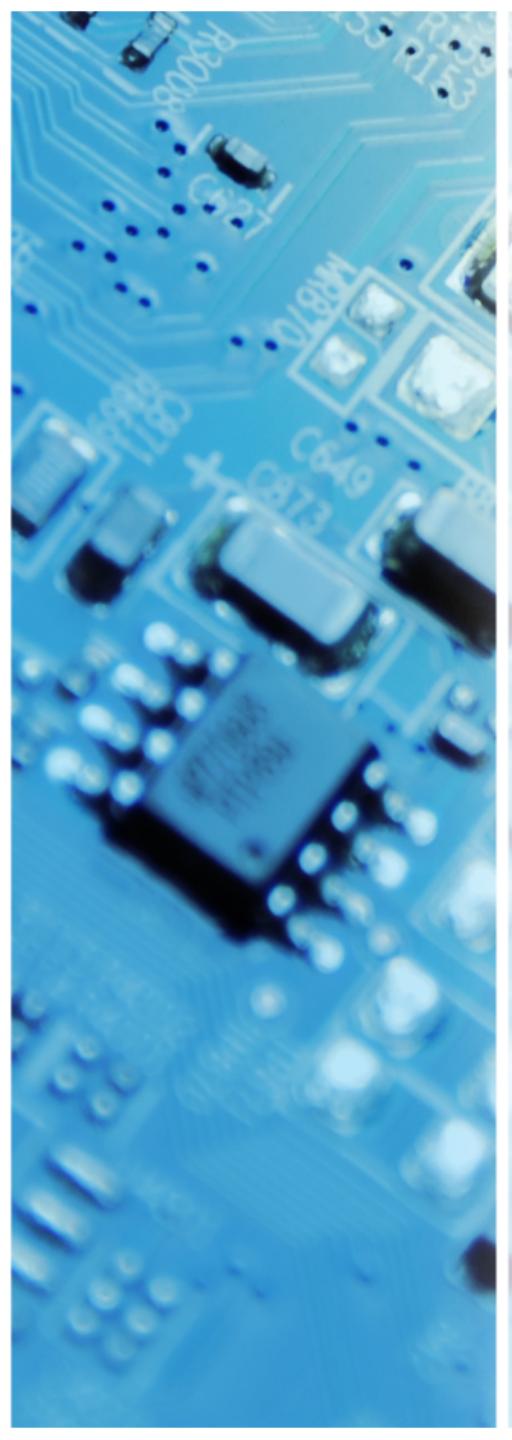
Demais métodos

- Há diversos outros métodos que apresentam um desempenho semelhante ao Quicksort
- O algoritmo MergeSort utiliza princípios de dividir e conquistar, semelhantes ao Quicksort.
- No caso do HeapSort a abordagem é baseado em uma estrutura de árvore e igualmente no conceito de divisão.



Desafio 1

- Executar os algoritmos disponibilizados para ordenar vetores com diferentes quantidades de dados (10, 100, 1000, 10000).
- Documentar as execuções e comparar os diferentes métodos.
- Entregar um documento com os dados (tamanho do vetor, maior valor, menor valor) e os resultados de cada método.
- Testar com o uso de diferentes estruturas, array, listas, tuplas. Pode ser gerado um array e os dados convertidos em outra estrutura.



Desafio 2

- Analisar os códigos das implementações de 2 métodos diferentes (escolher).
- Considerar um vetor com 10 elementos: [13, 21, 8, 11, 4, 1, 52, 70, 40, 32]
- Fazer o teste de mesa para os três métodos.
- Entregar a documentação do teste de mesa, mostrando a cada passo como ficou o vetor, qual a troca feita, os valores das variáveis.
- O teste de mesa deve ser feito manualmente e documentado passo a passo ou eventualmente é possível executar as rotinas, mostrando os valores das variáveis.
- Entretanto, além dos valores mostrados na execução, é importante indicar qual troca foi feita e como está o vetor a cada troca.