ESTRUTURA DE DADOS

Engenharia da Computação

Prof. Renato Matroniani

EDUCAÇÃO METODISTA

ALGORITMOS DE ORDENAÇÃO

- Normalmente, o usuário que vai inserir os dados **não está** ou **não pode estar** preocupado com a **ordem de entrada** dos dados no momento de sua inserção na relação, de modo que é comum encontrarmos elementos dispostos de maneira aleatória nos sistemas.
- Muitas vezes, necessitamos que esses dados apresentem uma ordem para que possamos realizar ações específicas (por exemplo, busca de dados).
- Devido a essas necessidades, foram desenvolvidos vários algoritmos de ordenação que consistem, basicamente, em realizar comparações sucessivas e trocar os elementos de posição.

EDUCAÇÃO METODISTA

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

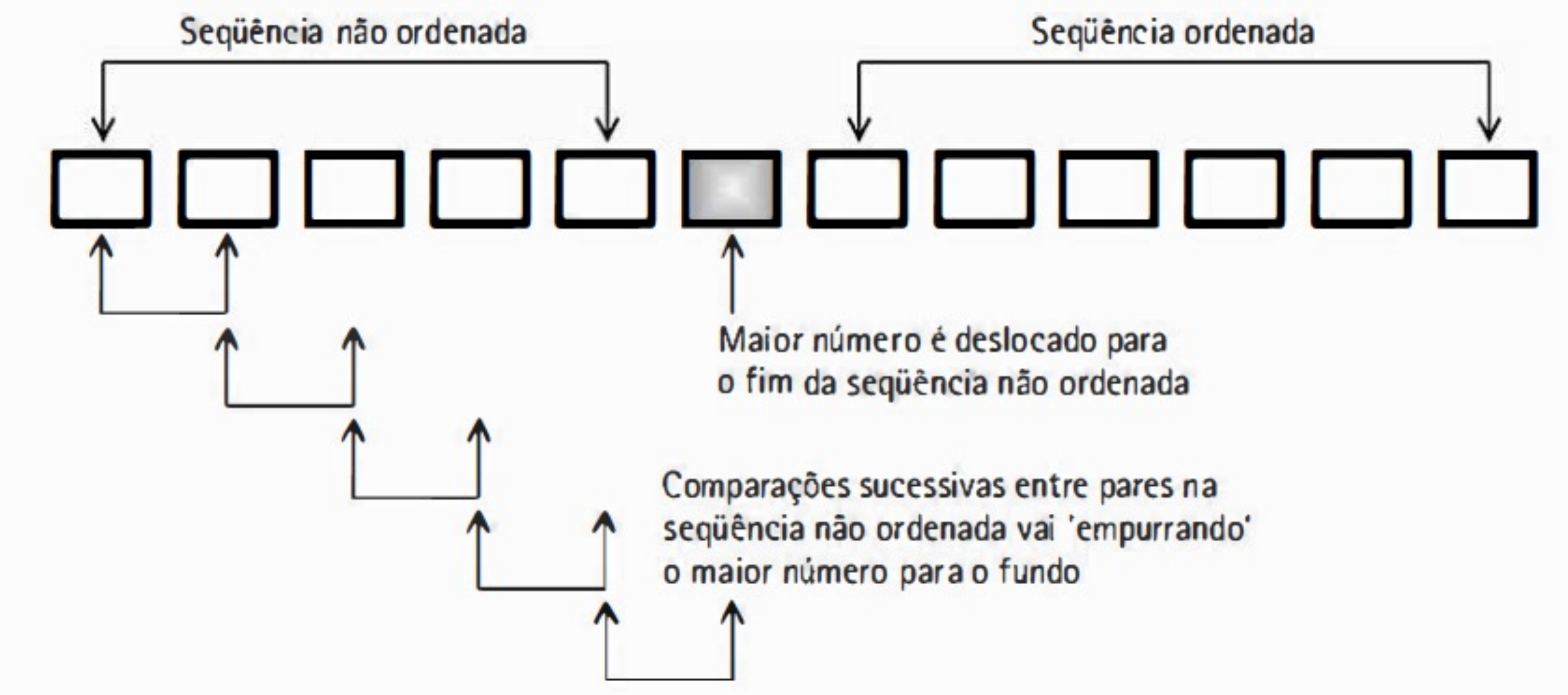
ORDENAÇÃO POR TROCA – *BUBBLE SORT*

- O método de ordenação por trocas é considerado o mais simples de todos.
- **Consiste em comparar pares consecutivos de valores e permutá-los caso estejam fora de ordem.**
- O algoritmo determina uma sequência de comparações sistemáticas que varrem a sequência de dados como um todo, fazendo com que o maior valor (ou menor, de acordo com a ordem desejada) acabe no final da sequência e uma nova série de comparações sistemáticas se inicia.

EDUCAÇÃO METODISTA

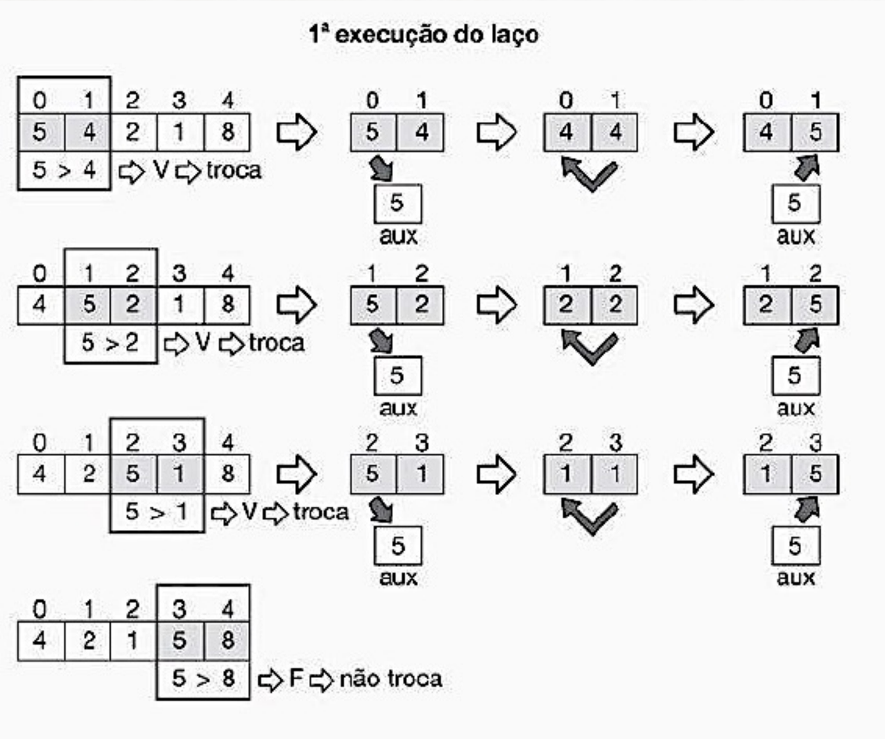
Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR TROCA – *BUBBLE SORT*



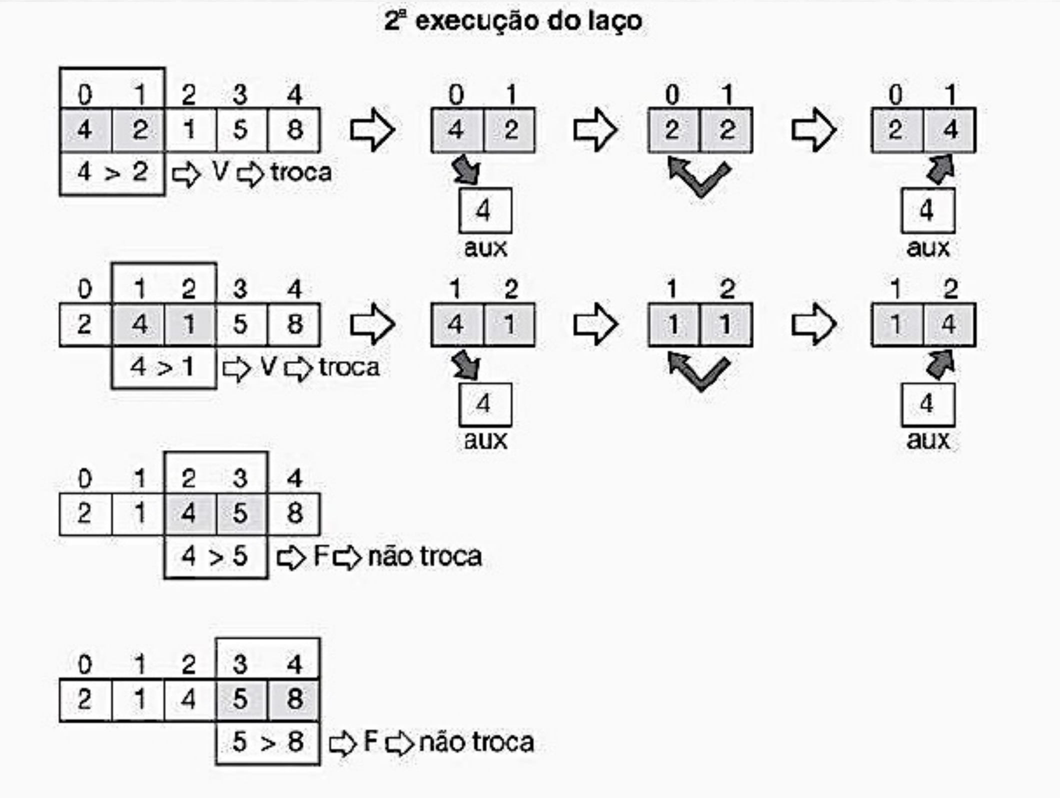
Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR TROCA – *BUBBLE SORT*



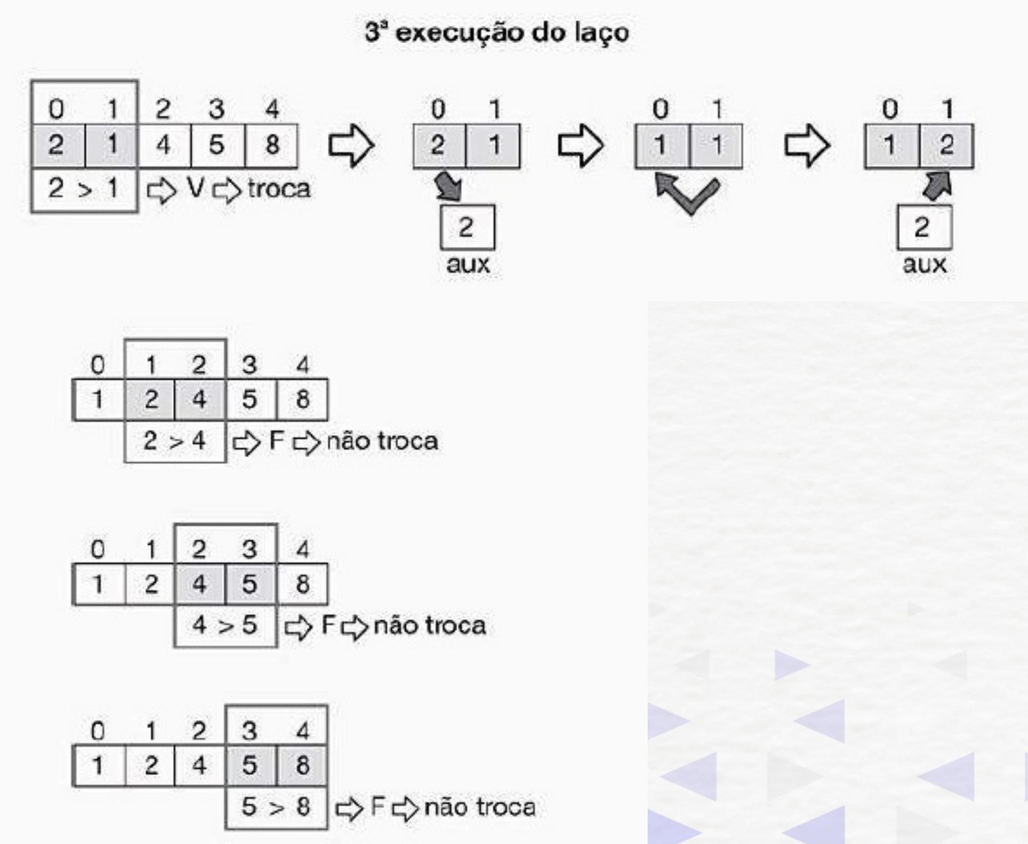
Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR TROCA – *BUBBLE SORT*



Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

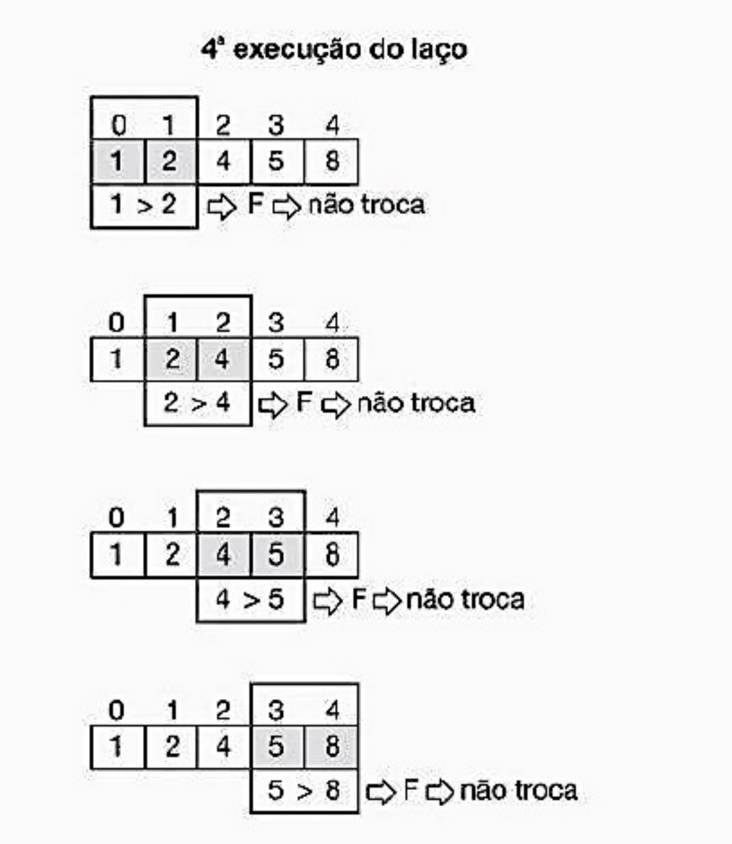
ORDENAÇÃO POR TROCA – *BUBBLE SORT*

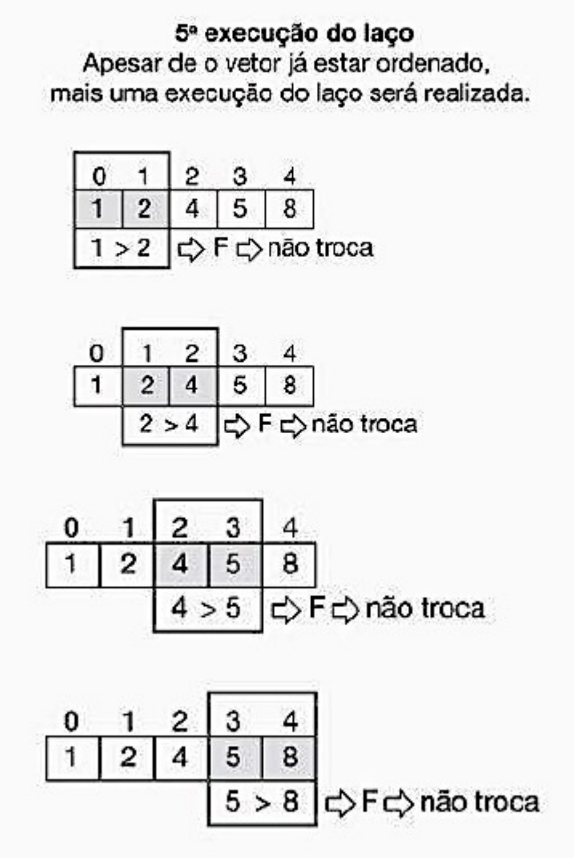


EDUCAÇÃO METODISTA

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR TROCA – *BUBBLE SORT*





EDUCAÇÃO METODISTA

base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR TROCA – *BUBBLE SORT*

```
algoritmo
declare X[5], n, i, aux numérico
// carregando os números no vetor
para i ← 0 até 4 faça
    início
     escreva "Digite o ",i+1,"° número: "
     leia X[i]
    fim
// ordenando de forma crescente
// laço com a quantidade de elementos do vetor
para n ← 1 até 5 faça
início
// laço que percorre da primeira à
// penúltima posição do vetor
    para i ← 0 até 3 faça
    início
        se (X[i] > X[i+1])
        então início
                 aux ← X[i]
                 X[i] ← X[i+1]
                 X[i+1] ← aux
              fim
    fim
fim
// mostrando o vetor ordenado
para i ← 0 até 4 faça
    início
     escreva i+1,"° número: ",X[i]
    fim
fim_algoritmo.
```

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR TROCA – *BUBBLE SORT*

- Análise de complexidade:
- fator relevante para T(n) é o número de comparações realizadas.
- Para o algoritmo do exemplo, temos:

$$n(n-1) = n^2 - n$$

Para um vetor de tamanho 5, no primeiro laço o algoritmo faz 5 interações, no segundo 5 x (5-1) = 20, e assim por diante.

$T(n) = O(n^2)$ para o *Bubble Sort*.

```
para n ← 1 até 5 faça
    início
    para i ← 0 até 4 faça
        início
        se (X[i] > X[i+1])
        então início
                aux ← X[i]
                X[i] ← X[i+1]
                X[i+1] ← aux
              fim
    fim
fim
```

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

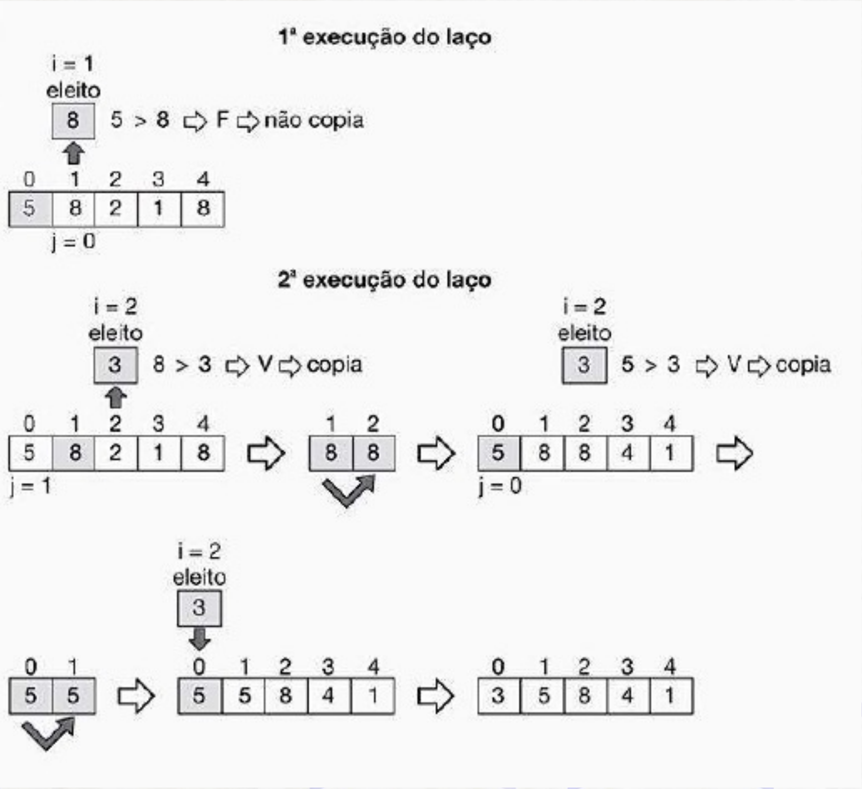
ORDENAÇÃO POR INSERÇÃO – *INSERTION SORT*

- O *insertion sort* é um algoritmo de ordenação onde é eleito o segundo número do vetor para iniciar as comparações.
- Os elementos à esquerda desse “eleito” estarão sempre ordenados de forma crescente ou decrescente.
- Os laços de comparações são executados do segundo elemento até o último.
- Enquanto existirem elementos à esquerda do eleito para comparações e a ordenação não for atendida, o laço será executado.
- Se o eleito está na posição i, os elementos à esquerda serão i-1 até 0. O laço é executado enquanto j>=0 e elemento [j] > eleito

EDUCAÇÃO METODISTA

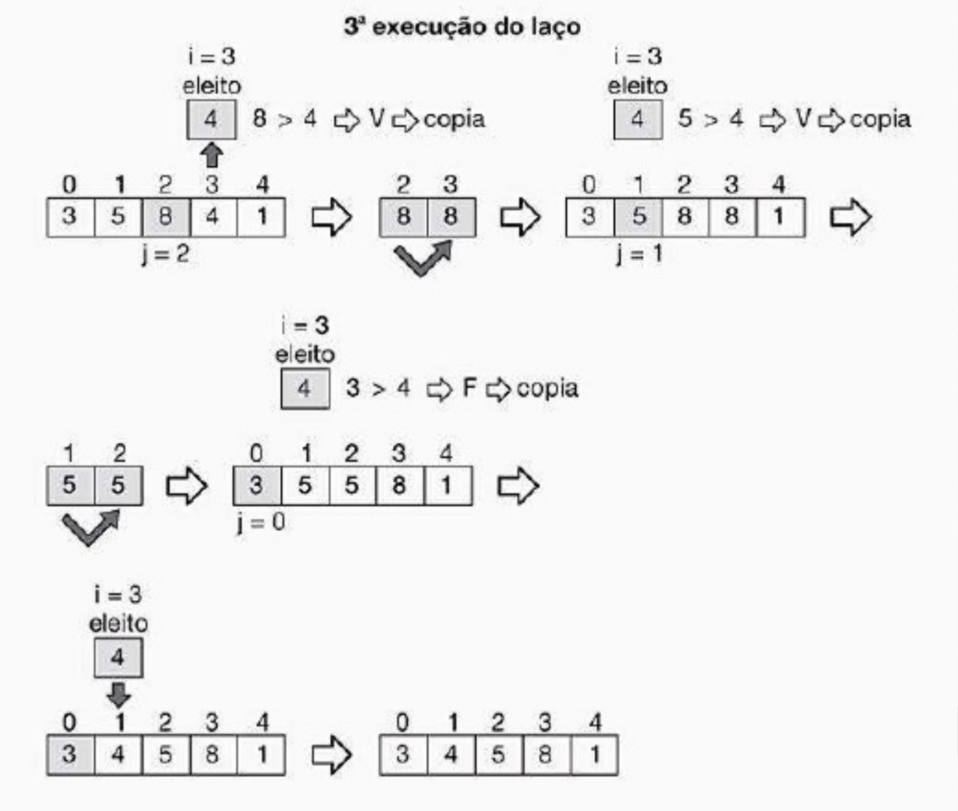
Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR INSERÇÃO – *INSERTION SORT*



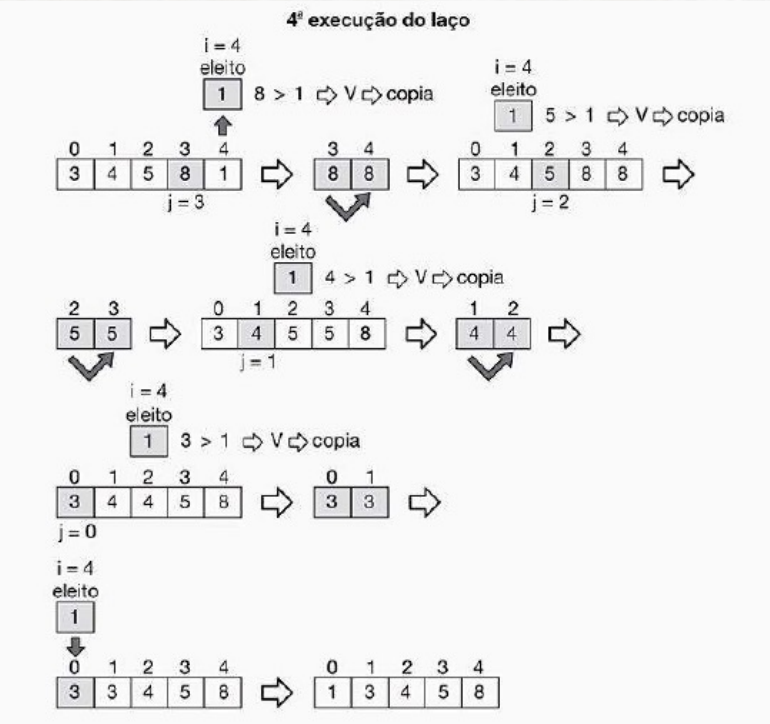
Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR INSERÇÃO – *INSERTION SORT*



Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR INSERÇÃO – *INSERTION SORT*



Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G .S., 2010

ORDENAÇÃO POR INSERÇÃO – *INSERTION SORT*

```
algoritmo
declare X[5], i, j, eleito numérico
// carregando os números no vetor
para i ← 0 até 4 faça
    início
     escreva "Digite o ",i+1,"° número: "
     leia X[i]
    fim
// ordenando de forma crescente
// laço com a quantidade de elementos do vetor – 1
para i ← 1 até 4 faça
    início
    eleito ← X[i]
    j ← i - 1
    // laço que percorre os elementos
    // à esquerda do número eleito
    // ou até encontrar a posição para
    // recolocação do número eleito
    // respeitando a ordenação procurada
    enquanto (j ≥ 0 E X[j] > eleito)
        início
         X[j+1] ← X[j]
         j ← j - 1
        fim
    X[j+1] ← eleito
    fim
// mostrando o vetor ordenado
para i ← 0 até 4 faça
    início
     escreva i+1,"° número: ",X[i]
    fim
fim_algoritmo.
```

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR INSERÇÃO – *INSERTION SORT*

- Análise de complexidade

```
para i ← 1 até 4 faça
início
 eleito ← X[i]
 j ← i – 1
 enquanto (j ≥ 0 E X[j] > eleito)
      início
       X[j+1] ← X[j]
              j ← j – 1
      fim
 X[j+1] ← eleito
fim
```

$$T(n) = 2 + 3 + 4 + \ldots + n$$

$$T(n) = \left(\sum_{i=1}^{n} i\right) - 1$$

$$T(n) = \frac{(1+n)n}{2} - 1$$

$$T(n) = \frac{n^2 + n}{2} - 1$$

$$T(n) = O(n^2), \text{para } c = 2, n \geq 1.$$

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR SELEÇÃO – *SELECTION SORT*

- Processo de ordenação por seleção:
- cada número do vetor, a partir do primeiro, é "**eleito**" e **comparado com o menor** número que esteja à **direita** do vetor (para **ordem crescente**).
- para ordem **decrescente**, a comparação é feita com o **maior** à **direita**.
- se for satisfeita a condição, o número troca de posição com o **eleito**.
- dessa forma, os números à esquerda ficam ordenados.

EDUCAÇÃO METODISTA

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR SELEÇÃO – *SELECTION SORT*

- Os laços de comparação são executados do primeiro ao penúltimo elemento, ou seja n-1 vezes (n = número de elementos do vetor).
- “o número da última posição não tem elementos à direita”.

```
for (i=0; i < n-1; i++)
```

- Além desse laço de comparação (para percorrer o vetor), temos um segundo laço a ser executado para encontrar o menor à direita do número eleito:

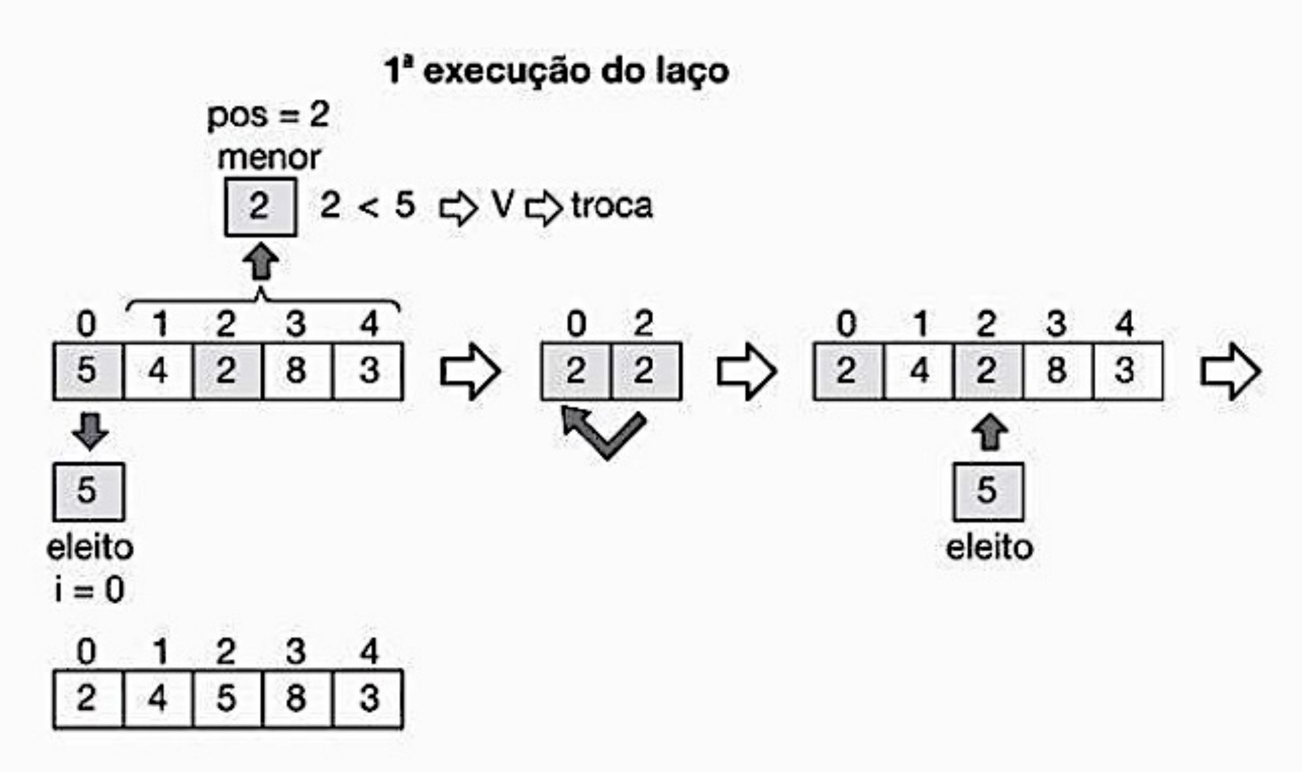
```
for (j=i+2; j <= n-1 ; j++)
```

- o primeiro elemento à direita do número eleito é considerado o menor número no início do processo.

EDUCAÇÃO METODISTA

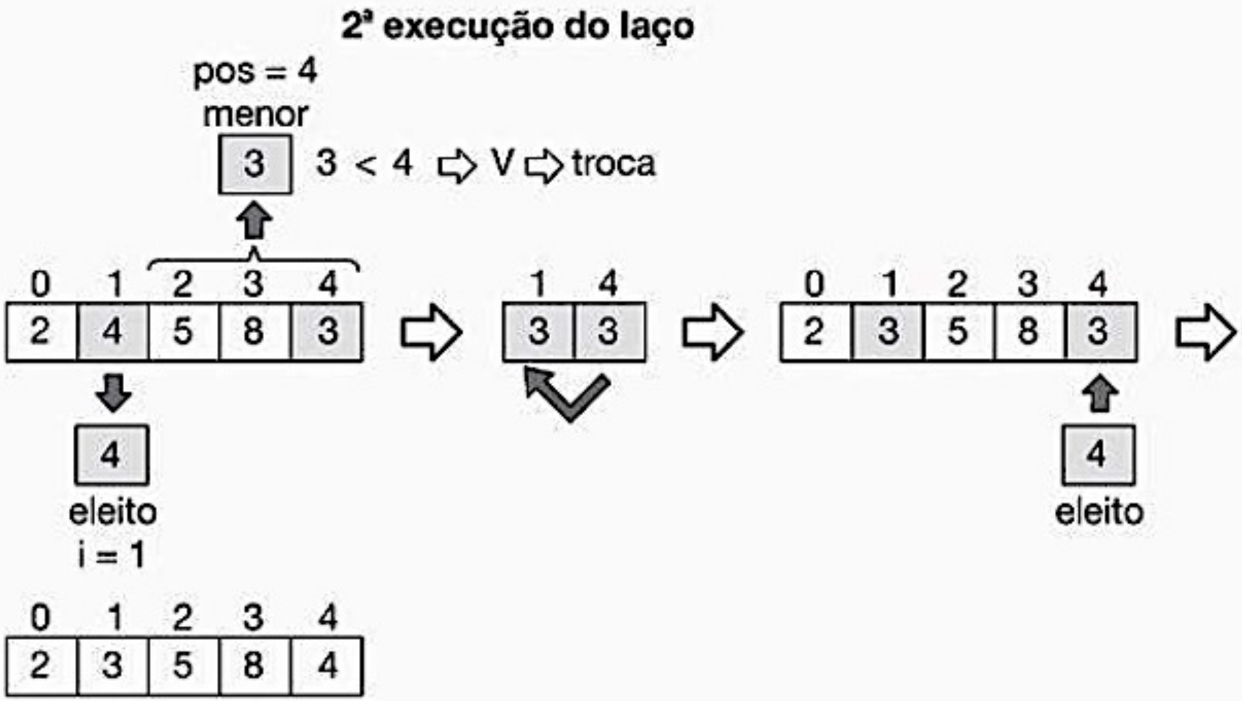
Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR SELEÇÃO – *SELECTION SORT*



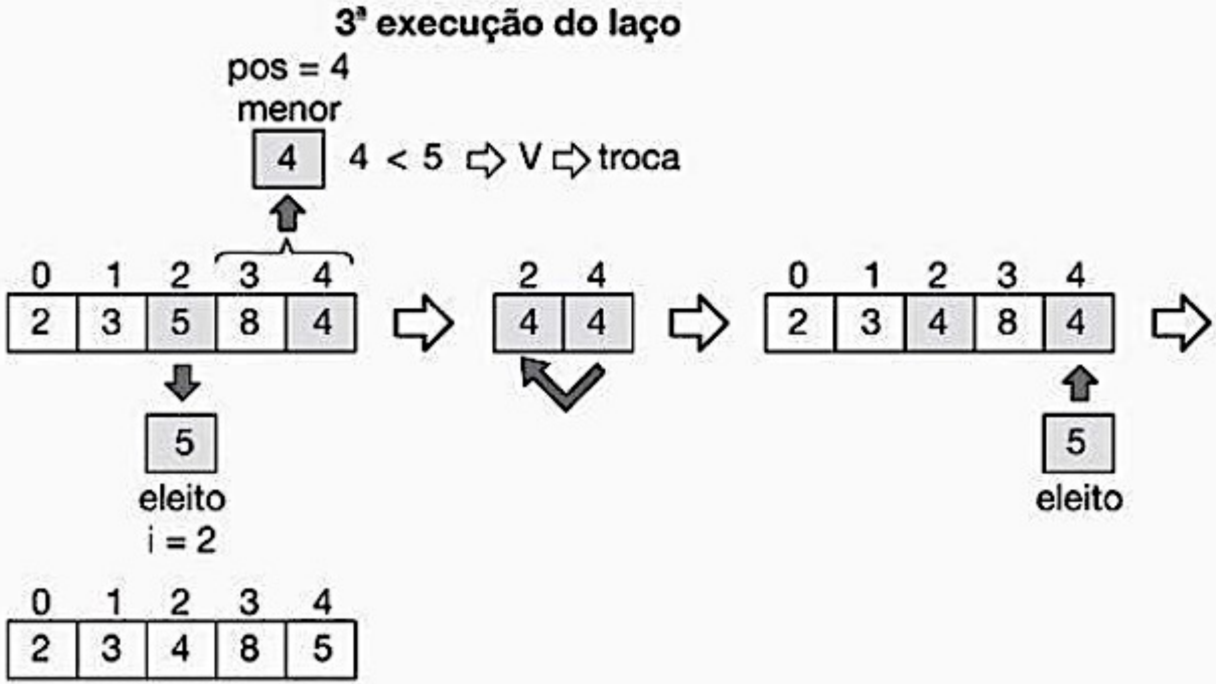
Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR SELEÇÃO – *SELECTION SORT*



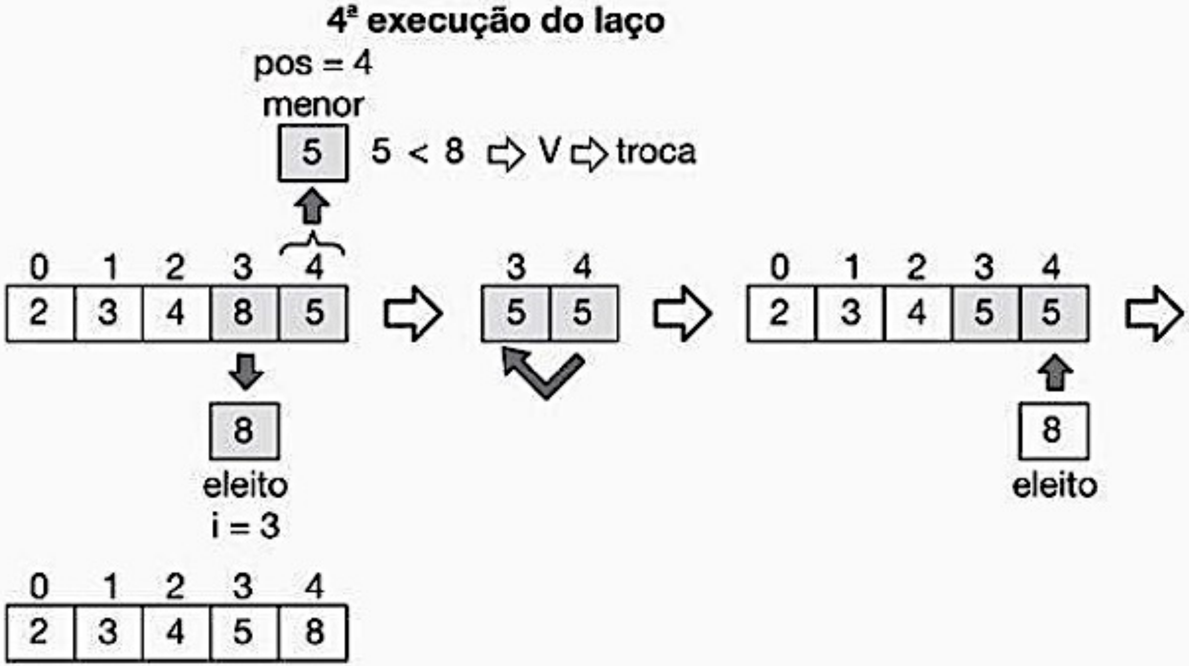
Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR SELEÇÃO – *SELECTION SORT*



Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR SELEÇÃO – *SELECTION SORT*



Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR SELEÇÃO – *SELECTION SORT*

```
algoritmo
declare X[5], i, j, eleito, menor, pos numérico
// carregando os números no vetor
para i ← 0 até 4 faça
    início
     escreva "Digite o ",i+1,"° número: "
     leia X[i]
    fim
// ordenando de forma crescente
// laço que percorre da 1ª posição à
// penúltima posição do vetor
// elegendo um número para ser comparado
para i ← 0 até 3 faça
 início
    eleito ← X[i]
    // encontrando o menor número à direita do eleito
    // com sua respectiva posição
    // posição do eleito = i
    // primeiro número à direita do eleito
    // na posição = i + 1
    menor ← X[i+1]
    pos ← i+1
    // laço que percorre os elementos que estão à direita do
    // número eleito, retornando o menor número à direita e
    // sua posição
    para j ←i+2 até 4 faça
    início
       se (X[j] < menor)
       então início
              menor ← X[j]
              pos ← j
            fim
    fim
    // troca do número eleito com o número da posição pos
    // o número da posição pos é o menor número à direita
    // do número eleito
    se (menor < eleito)
    então início
            X[i] ← X[pos]
            X[pos] ← eleito
          fim
fim
// mostrando o vetor ordenado
para i ← 0 até 4 faça
    início
    escreva i+1,"° número: ",X[i]
    fim
fim_algoritmo.
```

EDUCAÇÃO METODISTA

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR SELEÇÃO – *SELECTION SORT*

- análise de complexidade:
- $T(N) = \frac{(1+n-1)\cdot(n-1)}{2} = \frac{n^2-n}{2}$
- o tempo de execução do *selection sort* é $\boldsymbol{\Theta}(\boldsymbol{n^2})$
- independente do valor de entrada, o algoritmo se comportará da mesma maneira.

1

n-1

n-1

```
para i ← 0 até 3 faça
 início
  eleito ← X[i]
  menor ← X[i+1]
  pos ← i + 1
  para j ←i+1 até 4 faça
   início
    se (X[j] < menor)
    então início
            menor ← X[j]
            pos ← j
          fim
   fim
 se (menor < eleito)
 então início
        X[i] ← X[pos]
        X[pos] ← eleito
      fim
fim
```

EDUCAÇÃO METODISTA

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR INTERCALAÇÃO – *MERGE SORT*

- Processo de ordenação por intercalação;
- o vetor é dividido em vetores com metade do tamanho original através de procedimento recursivo;
- ele continua sendo dividido até que o vetor fique com apenas um elemento;
- esses elementos são ordenados e intercalados.
- técnica da divisão e conquista (e combinação).

EDUCAÇÃO METODISTA

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR INTERCALAÇÃO – *MERGE SORT*

- Os três passos da combinação e conquista:
- dividir o problema em determinado número de subproblemas;
- conquistar os subproblemas solucionando-os recursivamente.
- combinar as soluções dos subproblemas na solução do problema inicial.

EDUCAÇÃO METODISTA

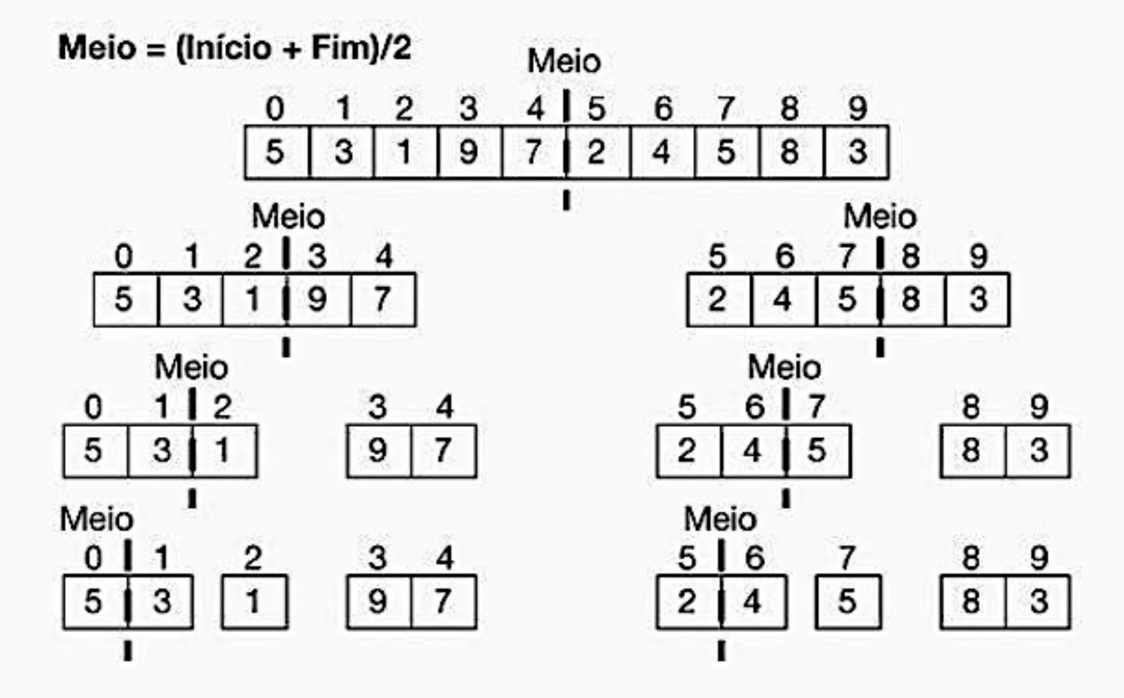
Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR INTERCALAÇÃO – *MERGE SORT*

- na ordenação por intercalação, para um vetor de tamanho n:
  - dividir o vetor em duas sequências de tamanho n/2;
  - ordenar as duas sequências recursivamente por intercalação;
  - intercalar as duas sequências ordenadas para gerar a solução.

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

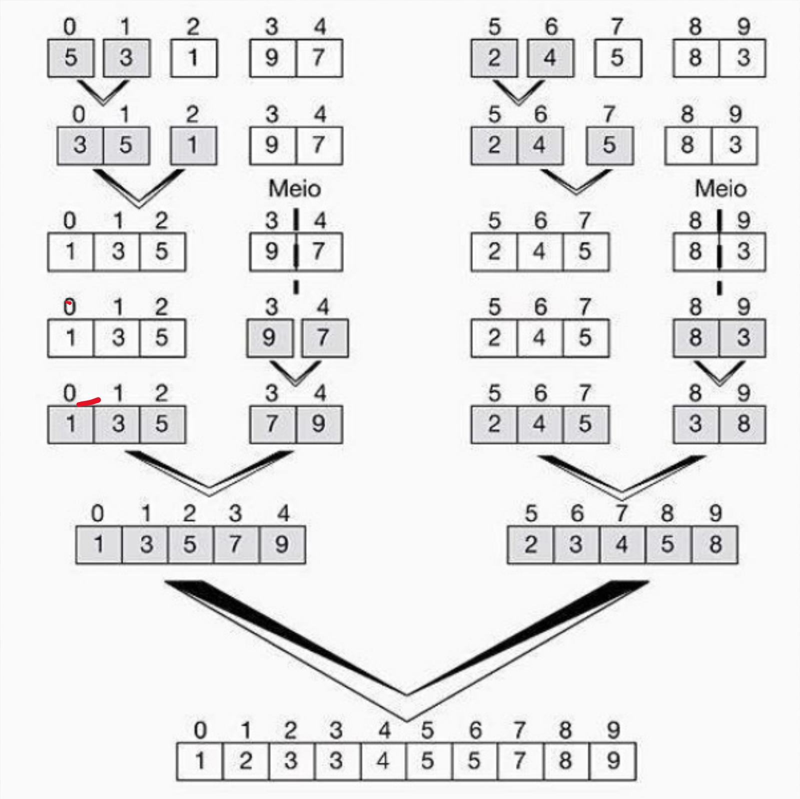
ORDENAÇÃO POR INTERCALAÇÃO – *MERGE SORT*



EDUCAÇÃO METODISTA

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR INTERCALAÇÃO – *MERGE SORT*



Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR INTERCALAÇÃO – *MERGE SORT*

```
algoritmo
declare X[10], i numérico
// carregando os números no vetor
para i ← 0 até 9 faça
    início
     escreva "Digite o ",i+1,"° número: "
     leia X[i]
    fim
// ordenando de forma crescente
merge(X,0,9);
// mostrando o vetor ordenado
para i ← 0 até 9 faça
    início
     escreva i+1,"° número: ",X[i]
    fim
fim_algoritmo.

Função intercala (X, início, fim, meio)
  início
   declare poslivre, início_vetor1 numérico
           início_vetor2, i, aux[10] numérico
```

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR INTERCALAÇÃO – *MERGE SORT*

```
início_vetor1 ← início
início_vetor2 ← meio+1
poslivre ← início
enquanto (início_vetor1 ≤ meio e início_vetor2 ≤ fim)
    início
        se (X[início_vetor1] ≤ X[início_vetor2])
        então início
                aux[poslivre] ← X[início_vetor1]
                início_vetor1 ← início_vetor1+1
              fim
        senão início
                aux[poslivre] ← X[início_vetor2]
                início_vetor2 ← início_vetor2+1
              fim
     poslivre ← poslivre+1
     fim
// se ainda existem números no primeiro vetor
// que não foram intercalados
        para i ← início_vetor1 até meio faça
        início
            aux[poslivre] ← X[i]
            poslivre ← poslivre+1
        fim
```

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR INTERCALAÇÃO – *MERGE SORT*

```
// se ainda existem números no primeiro vetor
// que não foram intercalados
        para i ← início_vetor1 até meio faça
        início
            aux[poslivre] ← X[i]
            poslivre ← poslivre+1
        fim
// se ainda existem números no segundo vetor
// que não foram intercalados
        para i ← início_vetor2 até fim faça
        início
            aux[poslivre] ← X[i]
            poslivre ← poslivre+1
        fim
// retorna os valores do vetor aux para o vetor X
    para i ← início até fim faça
    início
        X[i] ← aux[i]
    fim
```

Educação Metodista

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR INTERCALAÇÃO – *MERGE SORT*

```
fim_função_intercala.
Função merge (X, início, fim)
     início
        declare meio numérico
        se (início < fim)
        então início
                meio ← parteinteira((início + fim)/2)
                merge(X,início,meio)
                merge(X,meio+1,fim)
                intercala(X,início, fim, meio)
                fim
fim_função_merge.
```

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO POR INTERCALAÇÃO – *MERGE SORT*

- análise de complexidade:
- $T(N) = 2T\left(\frac{n}{2}\right) + n$
- $2T\left(\frac{n}{2}\right)$ → duas chamadas recursivas
- n → intercalação das duas metades
- o restante do algoritmo tem custo de tempo constante e é desconsiderado neste cálculo.
- $T(N) = \Theta(nlogn)$

```
Função merge (X, início, fim)
 início
  declare meio numérico
  se (início < fim)
    então início
           meio ← parteinteira((início + fim)/2)
            merge(X,início,meio)
            merge(X,meio+1,fim)
            intercala(X,início, fim, meio)
           fim
fim_função_merge.
```

$$\frac{n}{2} \to \frac{n}{4} \to \frac{n}{8} \to \dots \to \frac{n}{2^k} \leq 1$$

EDUCAÇÃO METODISTA

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

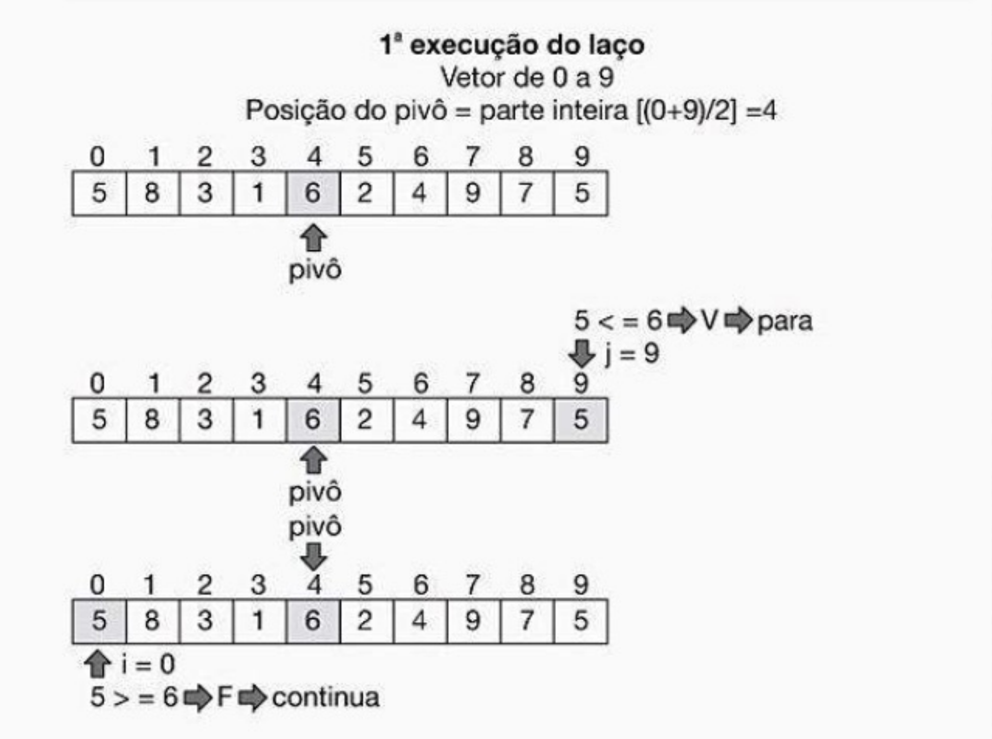
ORDENAÇÃO RÁPIDA – *QUICK SORT*

- Assim como no Merge Sort, o vetor é dividido em dois por método recursivo.
- A divisão ocorre até que o vetor fique com apenas um elemento.
- Os demais vetores ficam ordenados também a medida que ocorre a divisão.
- Também se baseia nos passos de divisão e conquista.
- Vamos analisar as diferenças, de forma visual, com o *Merge Sort*:

EDUCAÇÃO METODISTA

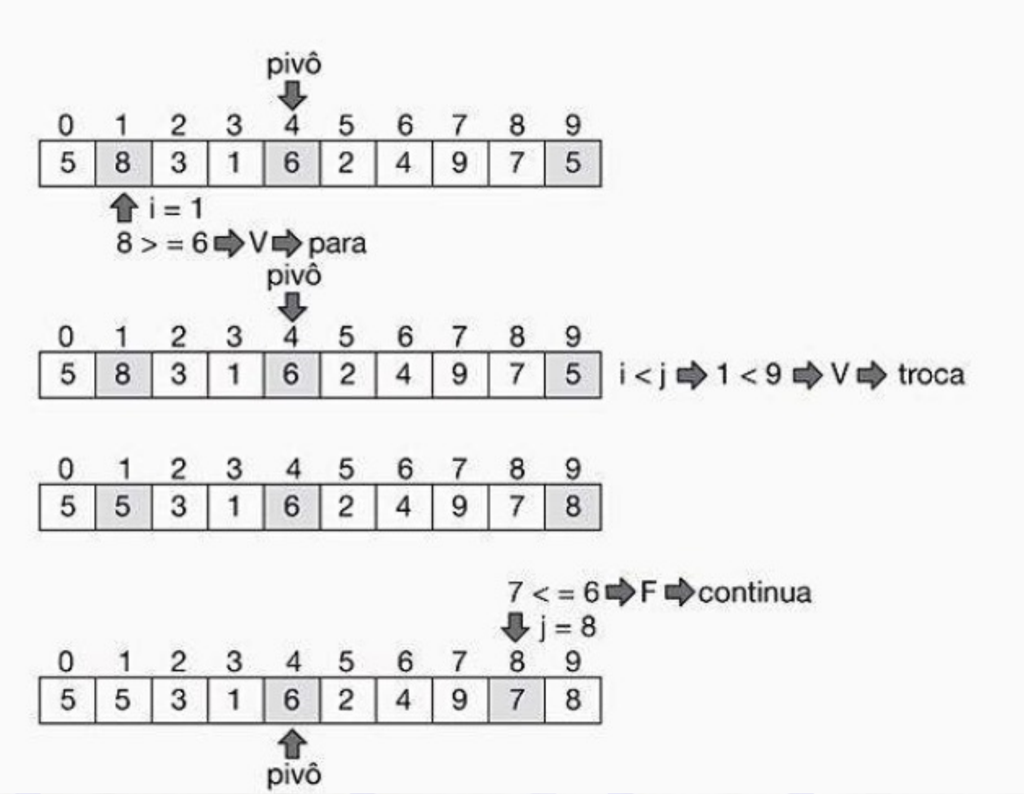
Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO RÁPIDA – *QUICK SORT*



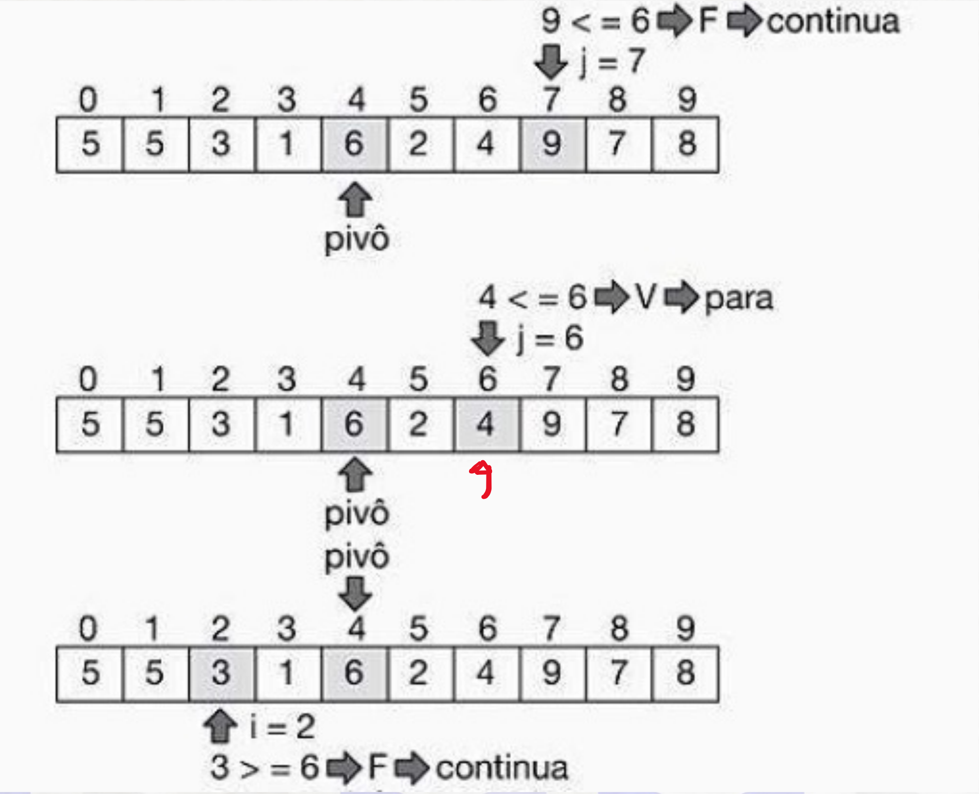
Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO RÁPIDA – *QUICK SORT*



Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

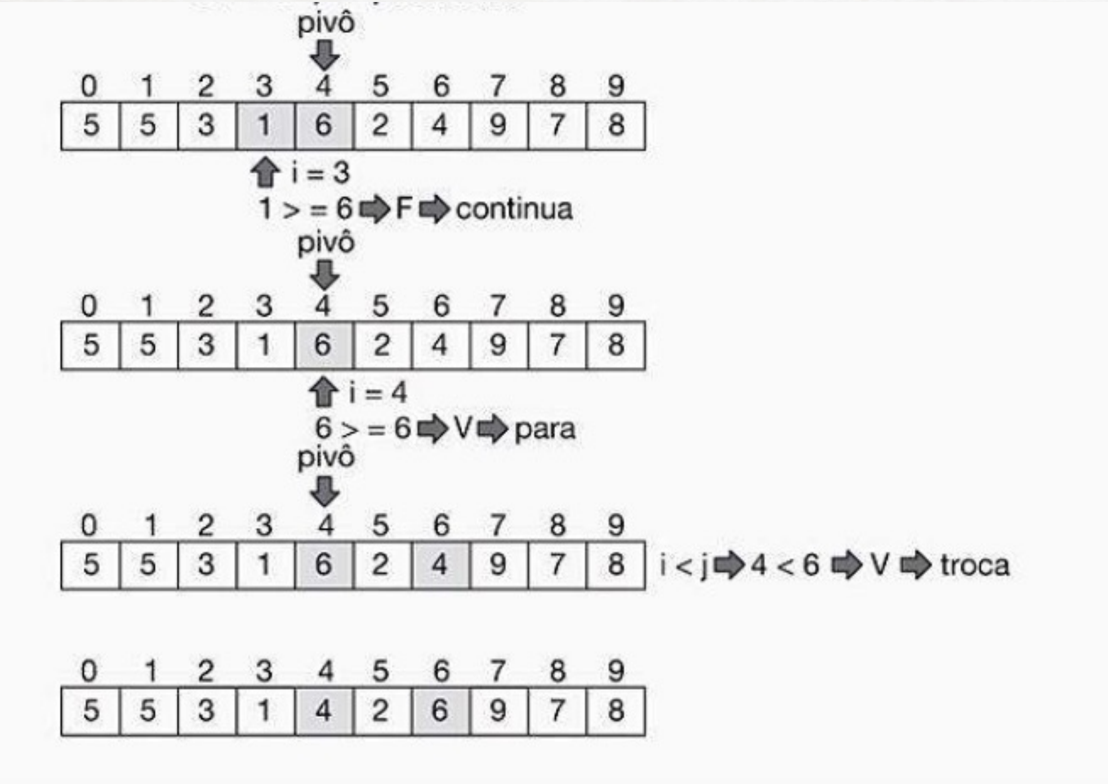
ORDENAÇÃO RÁPIDA – *QUICK SORT*



EDUCAÇÃO METODISTA

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

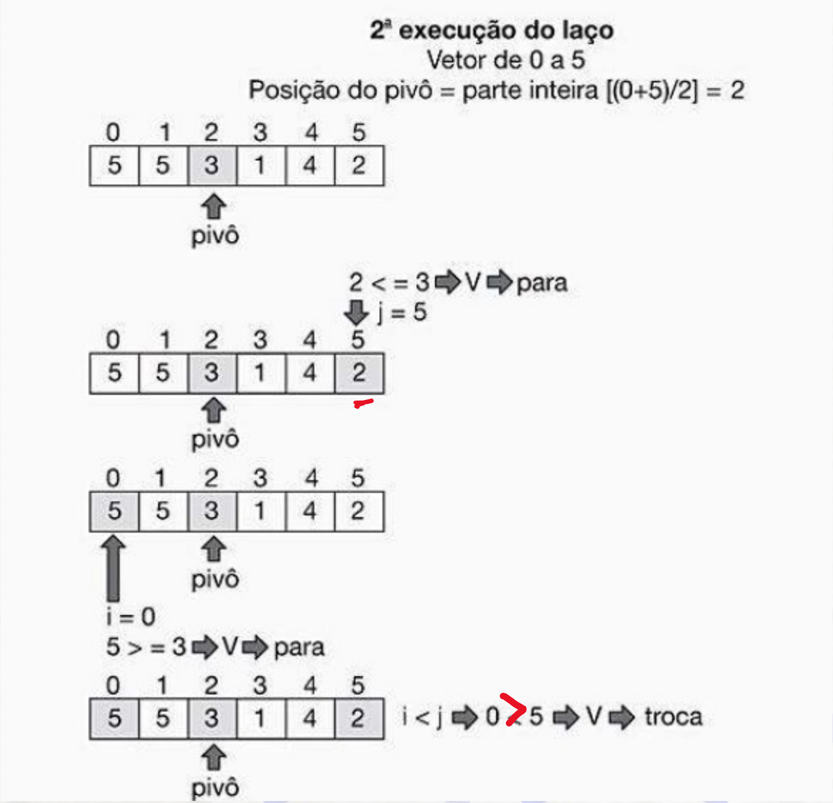
ORDENAÇÃO RÁPIDA – *QUICK SORT*



EDUCAÇÃO METODISTA

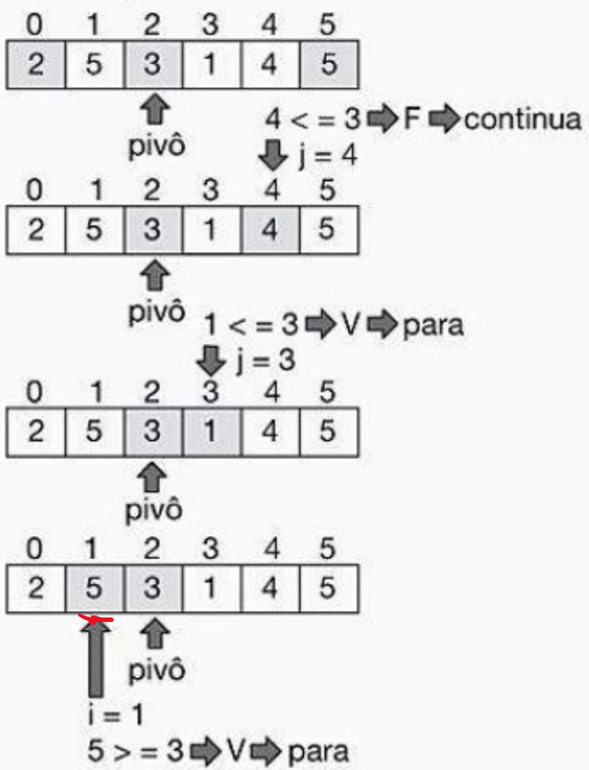
Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

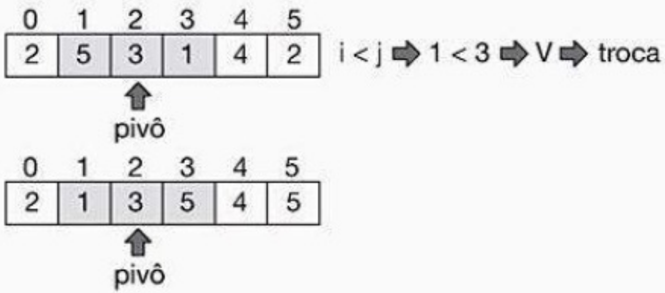
ORDENAÇÃO RÁPIDA – *QUICK SORT*



Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO RÁPIDA – *QUICK SORT*

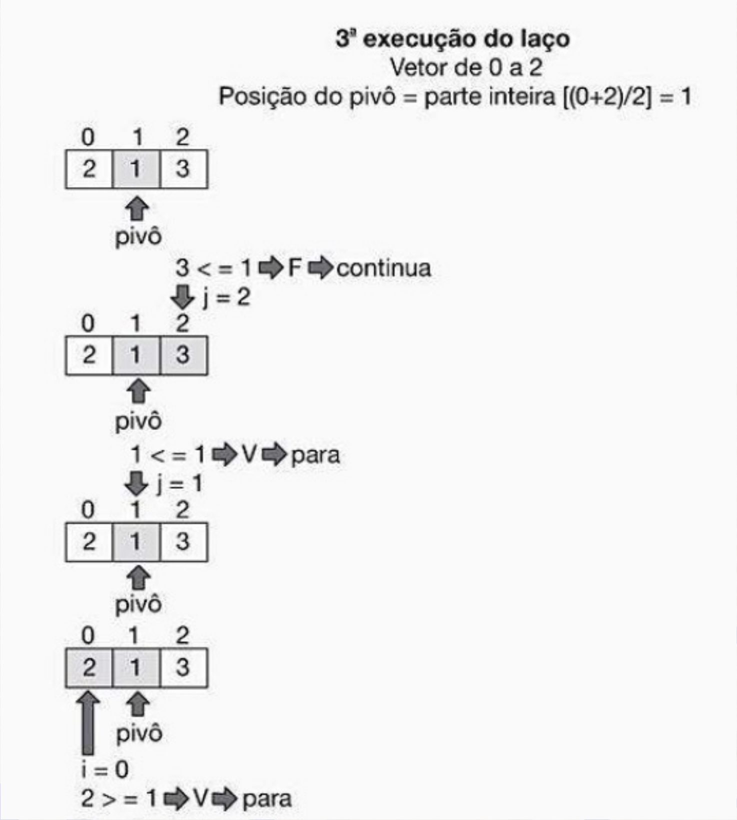


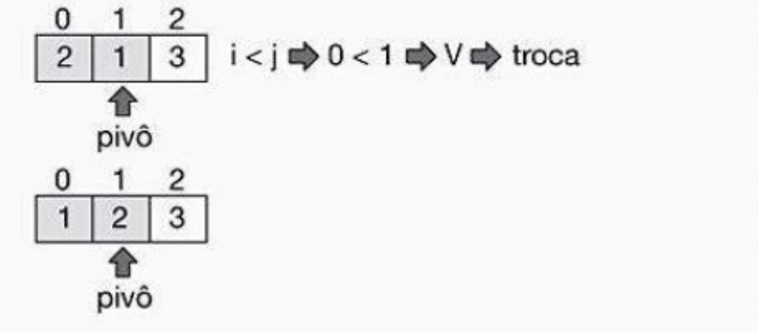


EDUCAÇÃO METODISTA

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO RÁPIDA – *QUICK SORT*

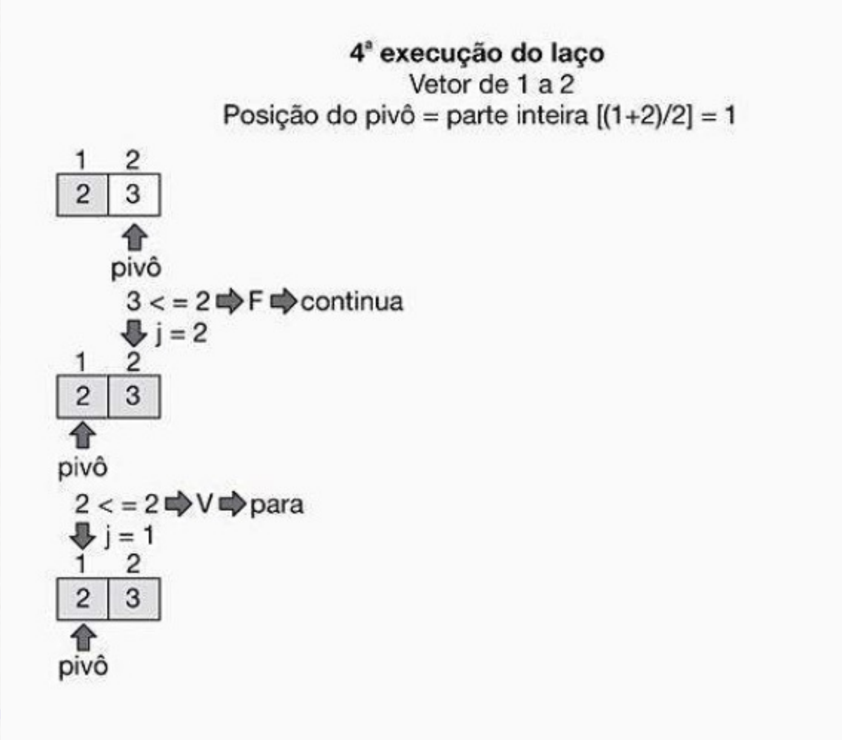


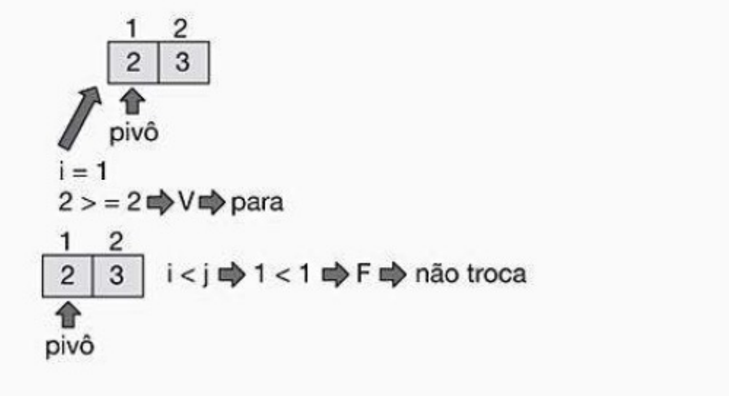


EDUCAÇÃO METODISTA

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

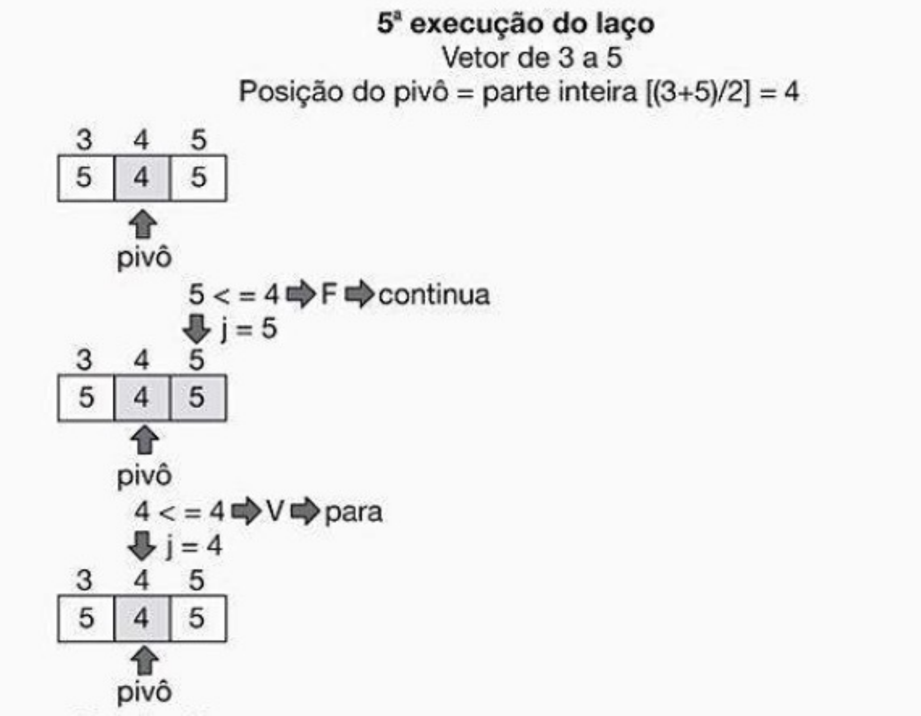
ORDENAÇÃO RÁPIDA – *QUICK SORT*

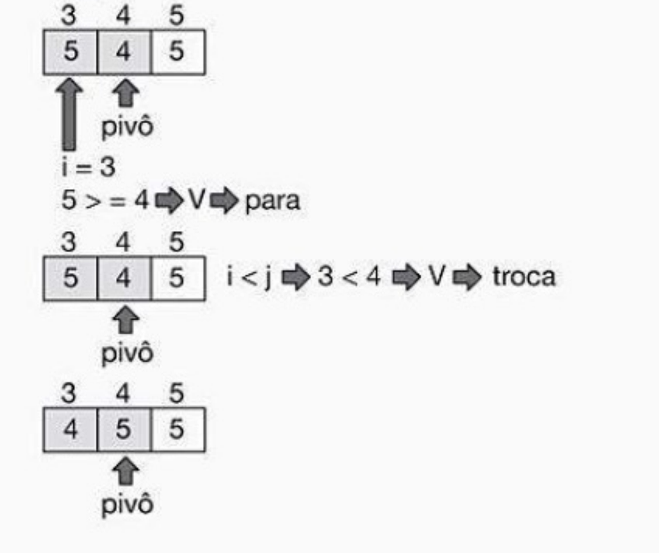




Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

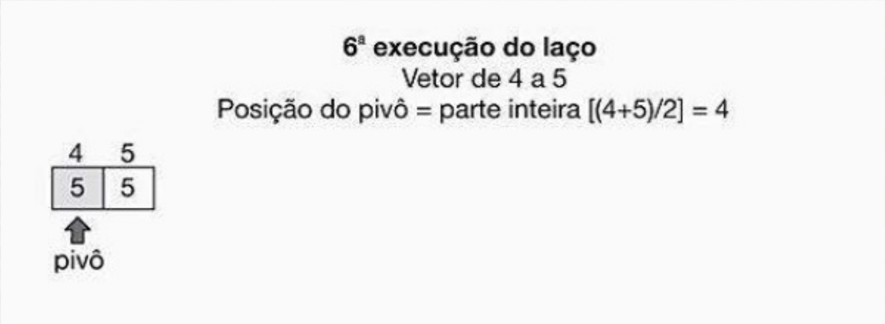
ORDENAÇÃO RÁPIDA – *QUICK SORT*

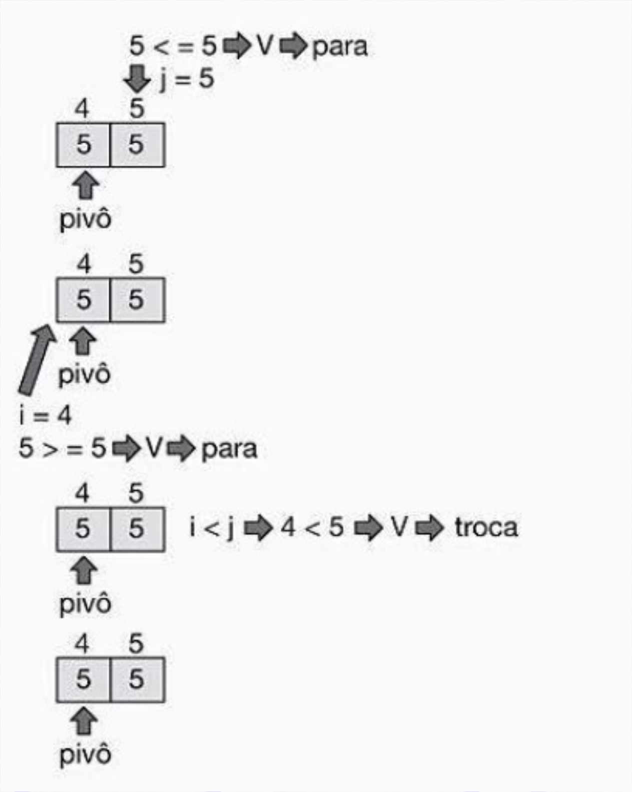




Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

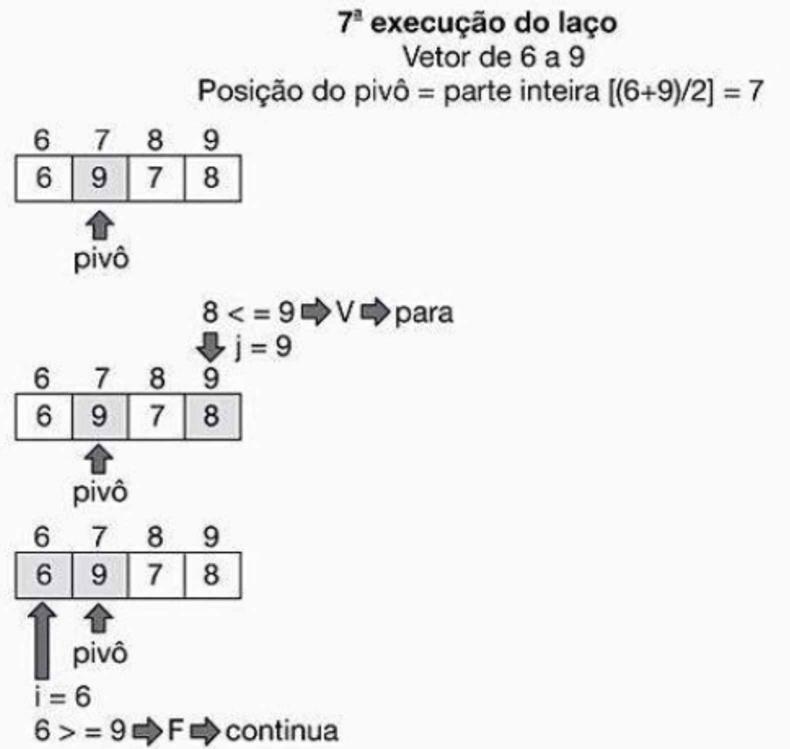
ORDENAÇÃO RÁPIDA – *QUICK SORT*

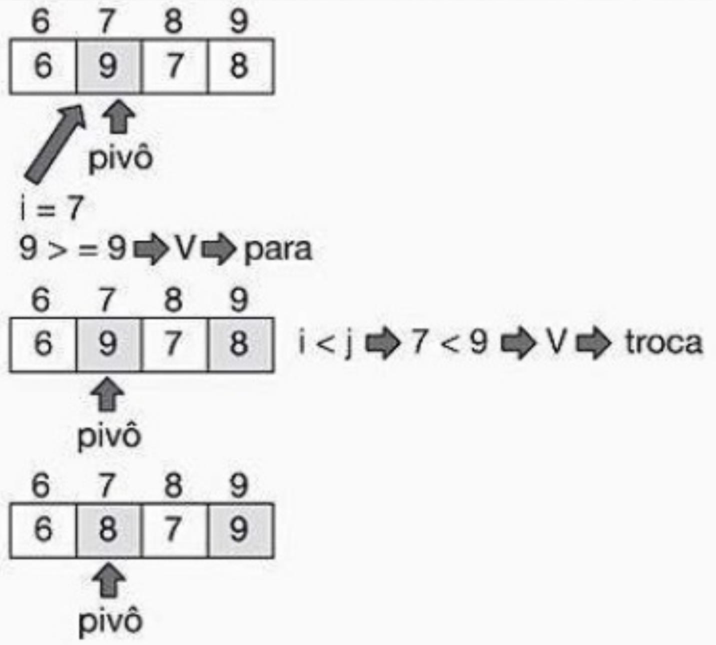




Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

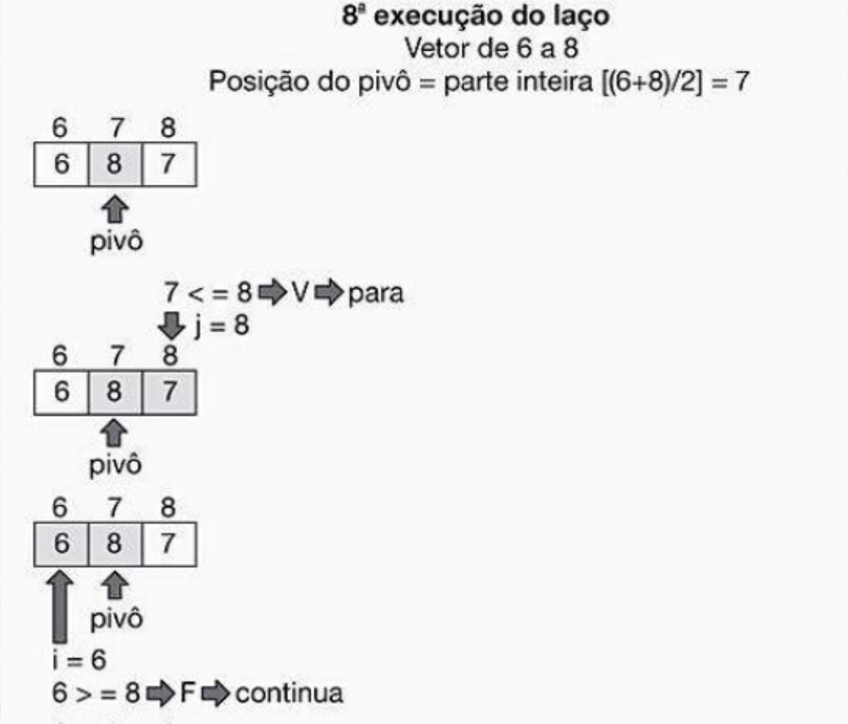
ORDENAÇÃO RÁPIDA – *QUICK SORT*

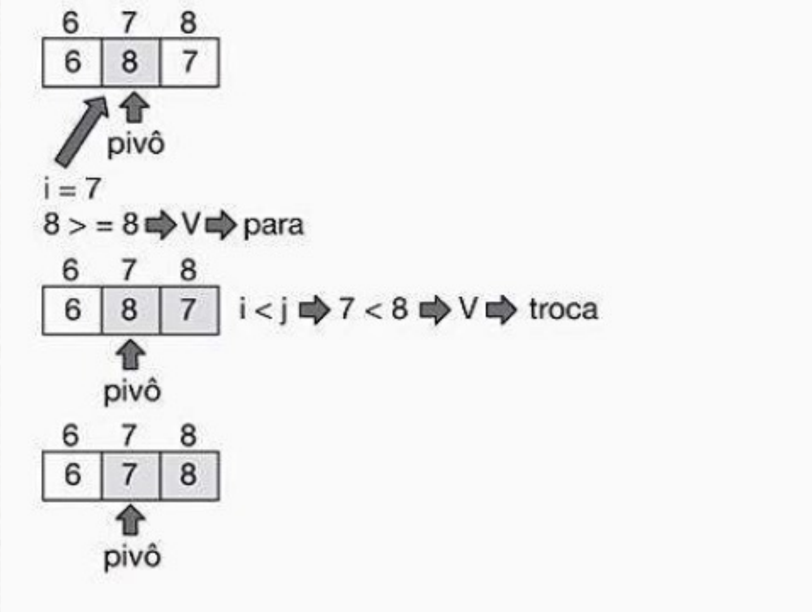




Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

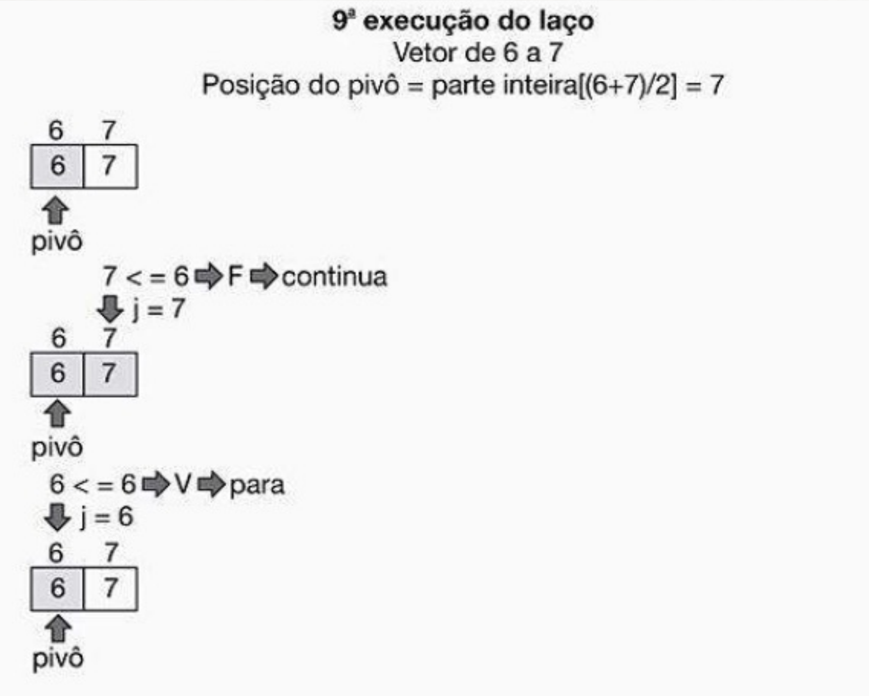
ORDENAÇÃO RÁPIDA – *QUICK SORT*

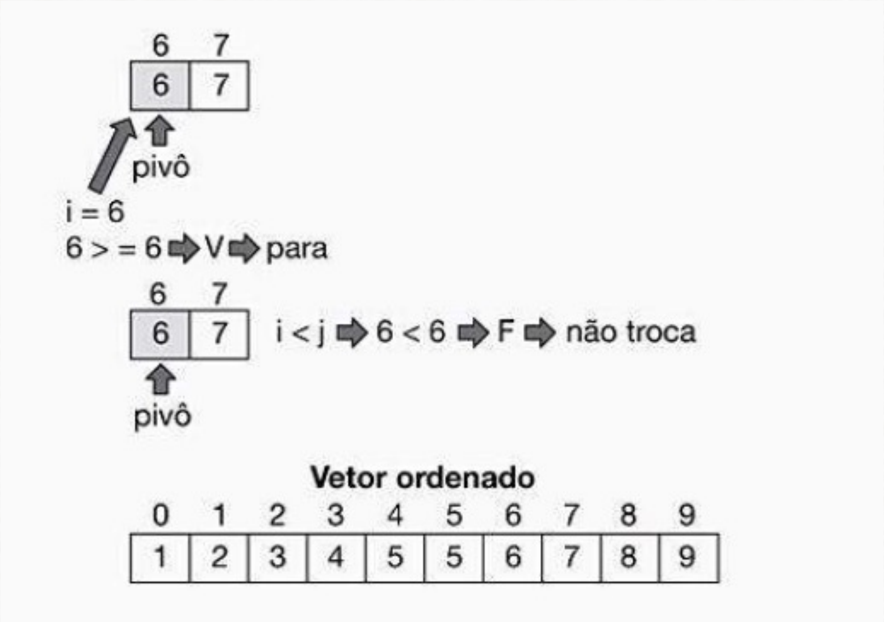




Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ORDENAÇÃO RÁPIDA – *QUICK SORT*





Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010