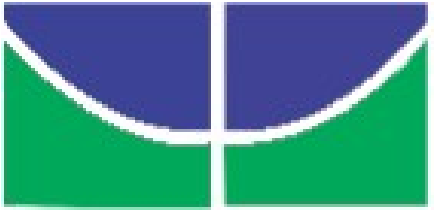


Algoritmos de ordenação – parte I

Prof. Dr. Wesin Ribeiro

Nessa aula você irá aprender

- O que é ordenação
- Ordenação por inserção
- Ordenação por seleção
- Analisar a complexidade dos algoritmos

O que é ordenação?

- É a tarefa de colocar um conjunto de dados (arranjo) em uma determinada ordem.
- Entrada: <a1, a2, … an>
- Saída: (a’1, a’2, … a’n), a’1 <= a’2 <= … a’n

Por que ordenar?

- Considerado o problema mais fundamental no estudo de algoritmos
- Necessidade de ordenar informações
- Subrotina chave
- Problema de interesse histórico
- Permite acesso mais eficiente aos dados

Tipos de ordenação

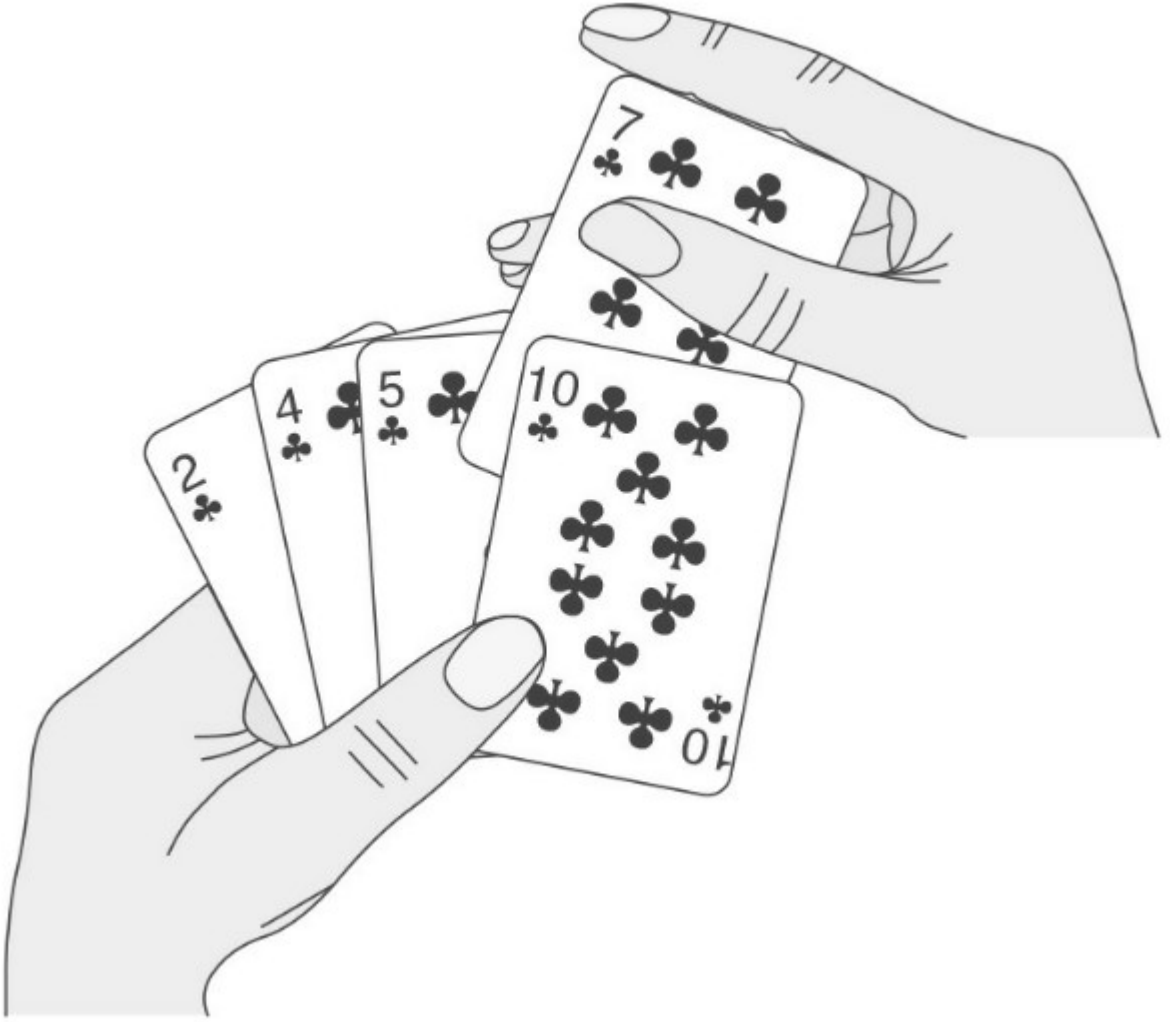
- Numérica (1,2,3,4 e 5)
- Lexicográfica (ordem alfabética)
- Podendo ambas serem crescente ou decrescente

Classificação dos métodos de ordenação

- Interna:
  - O arquivo a ser ordenado cabe todo na memória principal
  - Qualquer registro é imediatamente acessado
- Externa
  - Não cabe na memória principal
  - Acessado por partes

Ordenação por Inserção

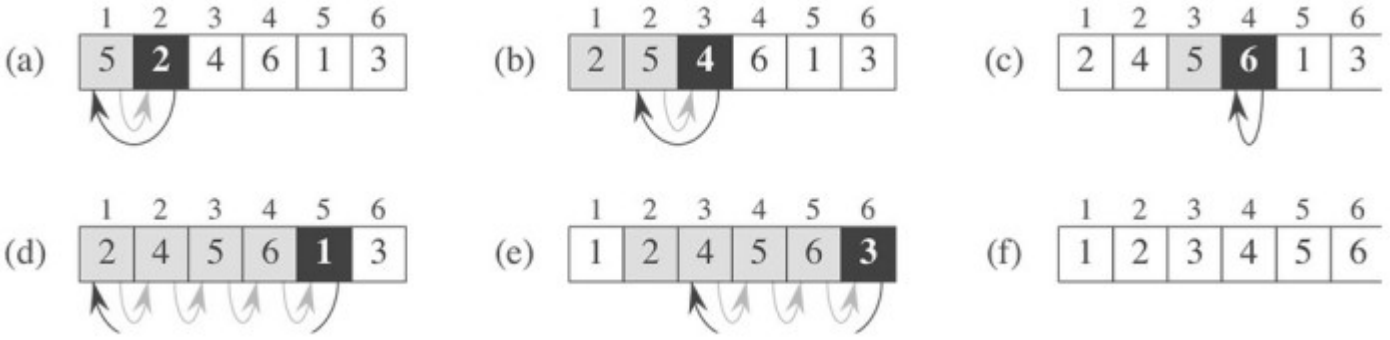
Inspirado na ordenação de cartas em um baralho



Mecânica da ordenação por inserção

- Dois sub vetores (esquerdo e direito), com o da esquerda ordenado e o da direita desordenado.
- Começa com um elemento apenas no sub vetor da esquerda, e os demais no da direita.
- Passa-se um elemento de cada vez do sub vetor da direita para o sub vetor da esquerda
- manter o sub vetor da esquerda ordenado.
- Termina quando o sub vetor da direita (desordenado) fica vazio.

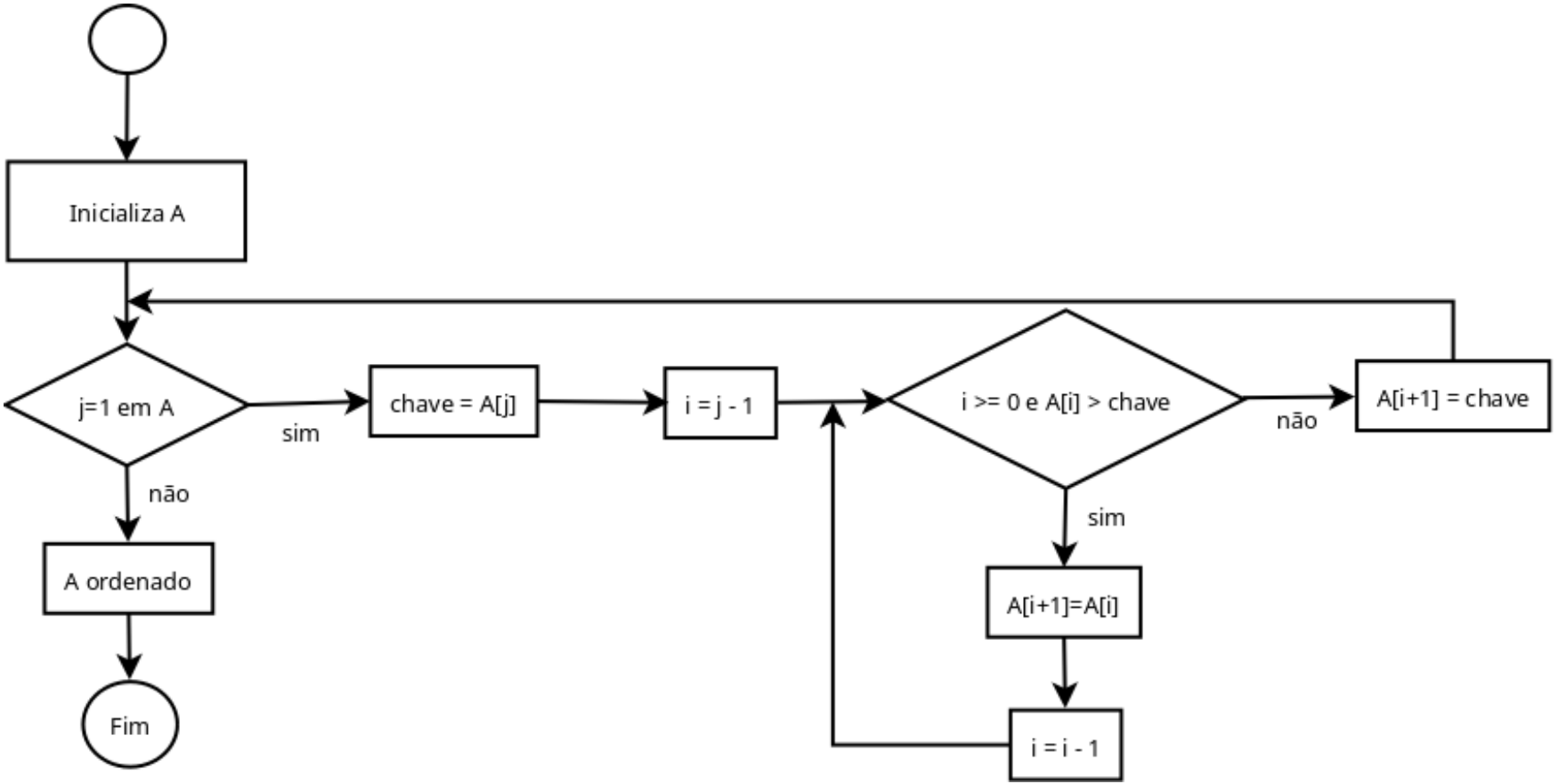
Exemplo considerando A=<5,2,4,6,1,3>



```
INSERTION-SORT(A)
1   for j = 2 to A·comprimento
2       chave = A[j]
3       // Inserir A[j] na sequência ordenada A[1.. j − 1].
4       i = j − 1
5       while i > 0 e A[i] > chave
6           A[i + 1] = A[i]
7           i = i − 1
8       A[i + 1] = chave
```

Pseudo-código da ordenação por inserção

Fluxograma da ordenação por inserção



Análise da ordenação por inserção

INSERTION-SORT(A)	*custo*	*vezes*
1 **for** $j = 2$ **to** A·*comprimento*	c_1	n
2 $chave = A[j]$	c_2	$n-1$
3 //Inserir $A[j]$ na sequência ordenada $A[1..j-1]$.	0	$n-1$
4 $i = j-1$	c_4	$n-1$
5 **while** $i > 0$ e $A[i] > chave$	c_5	$\sum_{j=2}^{n} t_j$
6 $A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7 $i = i-1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
8 $A[i+1] = chave$	c_8	$n-1$

Análise da ordenação por inserção

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1) .$$

$$\begin{aligned} T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1) \\ &= (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8) . \end{aligned}$$

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

and

$$\sum_{j=2}^{n} (j-1) = \frac{n(n-1)}{2}$$

$$\begin{aligned} T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5 \left(\frac{n(n+1)}{2} - 1 \right) \\ &\quad + c_6 \left(\frac{n(n-1)}{2} \right) + c_7 \left(\frac{n(n-1)}{2} \right) + c_8(n-1) \\ &= \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2} \right) n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8 \right) n \\ &\quad - (c_2 + c_4 + c_5 + c_8) . \end{aligned}$$

Tempo de execução da ordenação por inserção

- Melhor caso
  - Quando os elementos já estão ordenados
  - O(N)
- Pior caso
  - Quando os elementos estão ordenados na ordem inversa
  - O(N^2)

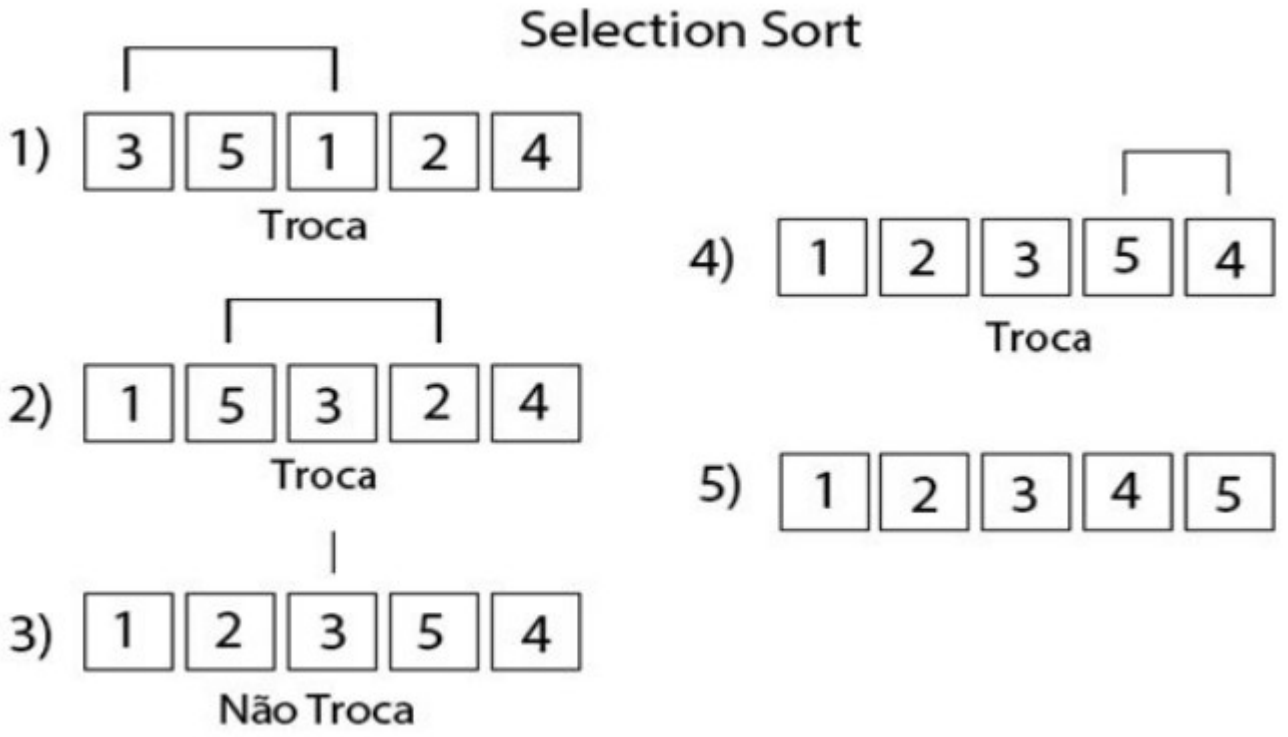
Ordenação por seleção

Seleciona o menor elemento para ser inserido na posição correta

Mecânica do algoritmo

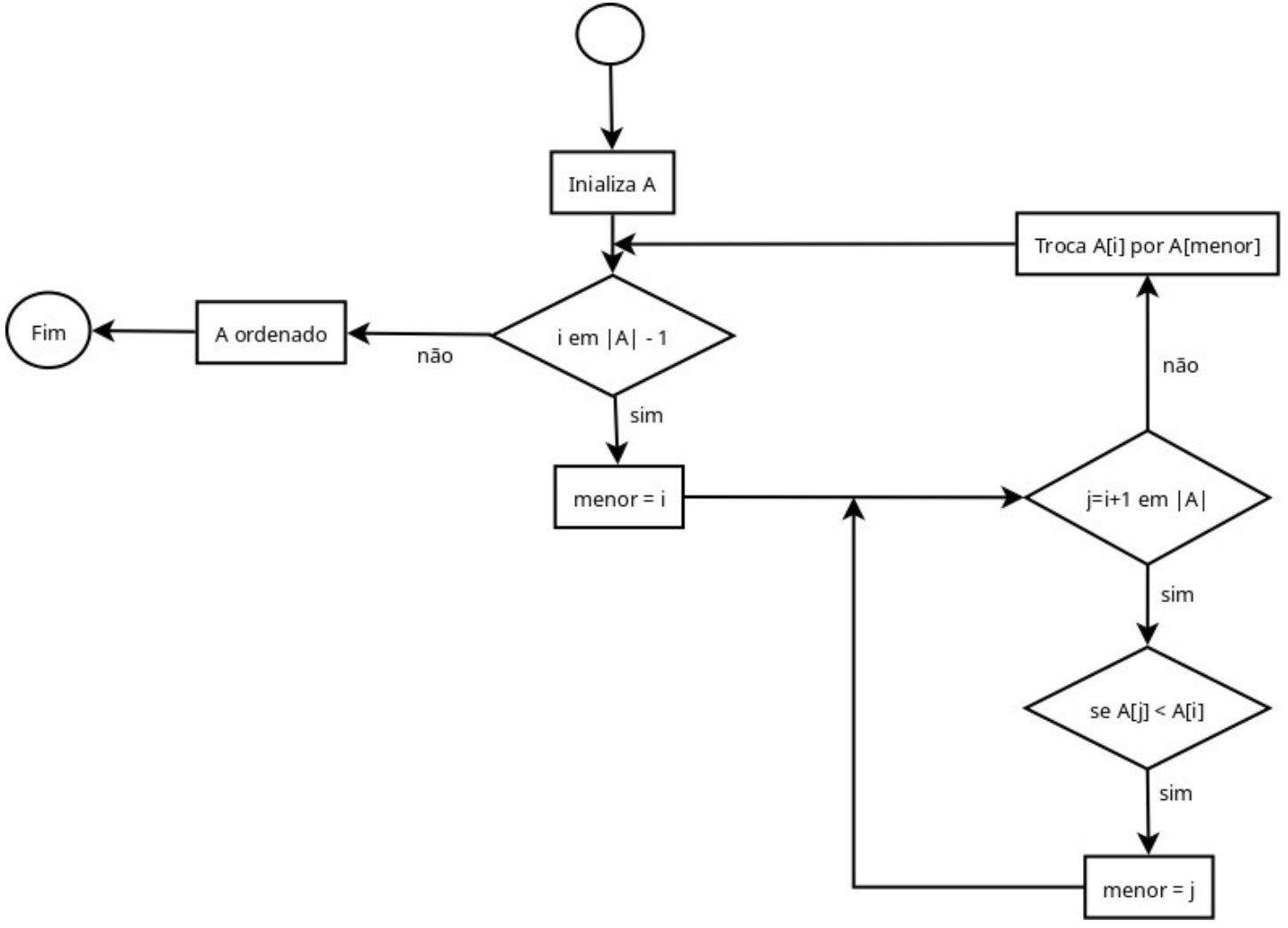
- Definir o mínimo do vetor, sendo ele o primeiro elemento no início do laço
- Comparar com todos os elementos do vetor até encontrar o menor dentre todos os elementos
- Troca como termo mínimo inicial
- Repetir para o restante do vetor (excluindo os que já foram ordenados)

Exemplo considerando A=<3,5,1,2,4>



```
SelectionSort(A)
    Para i = 0 até A.comprimento – 1
        Menor= i
        Para j = i + 1 até A.comprimento
            Se A[j] menor que A[i]
                Menor = j
        Troca A[i] por A[Menor]
```

Pseudo código da ordenação por seleção



Fluxograma da ordenação por seleção

Tempo de execução da ordenação por seleção

- Melhor caso
  - Quando os elementos já estão ordenados
  - O(N^2)
- Pior caso
  - Quando os elementos estão ordenados na ordem inversa
  - O(N^2)

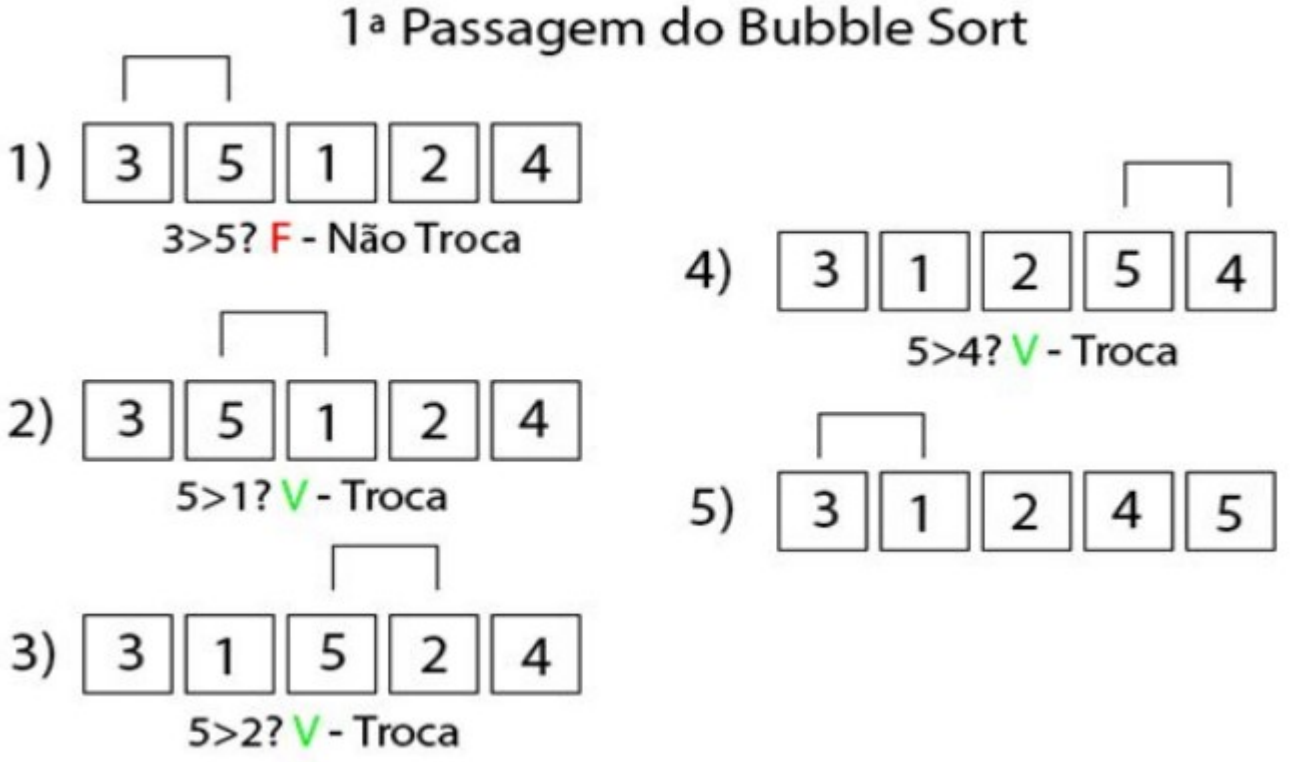
Bublesort

Semelhante a ordenação por seleção, mas comparara apenas elementos adjacentes

Mecânica do algoritmo

- Compara o elemento i com o elemento i+1
- Ou seja, um elemento na posição 2 será comparado ao elemento da posição 3.
- Se o elemento 2 for maior que o 3, troca de lugar (bolha)
- Repete o procedimento até não houver mais trocas (bolhas)

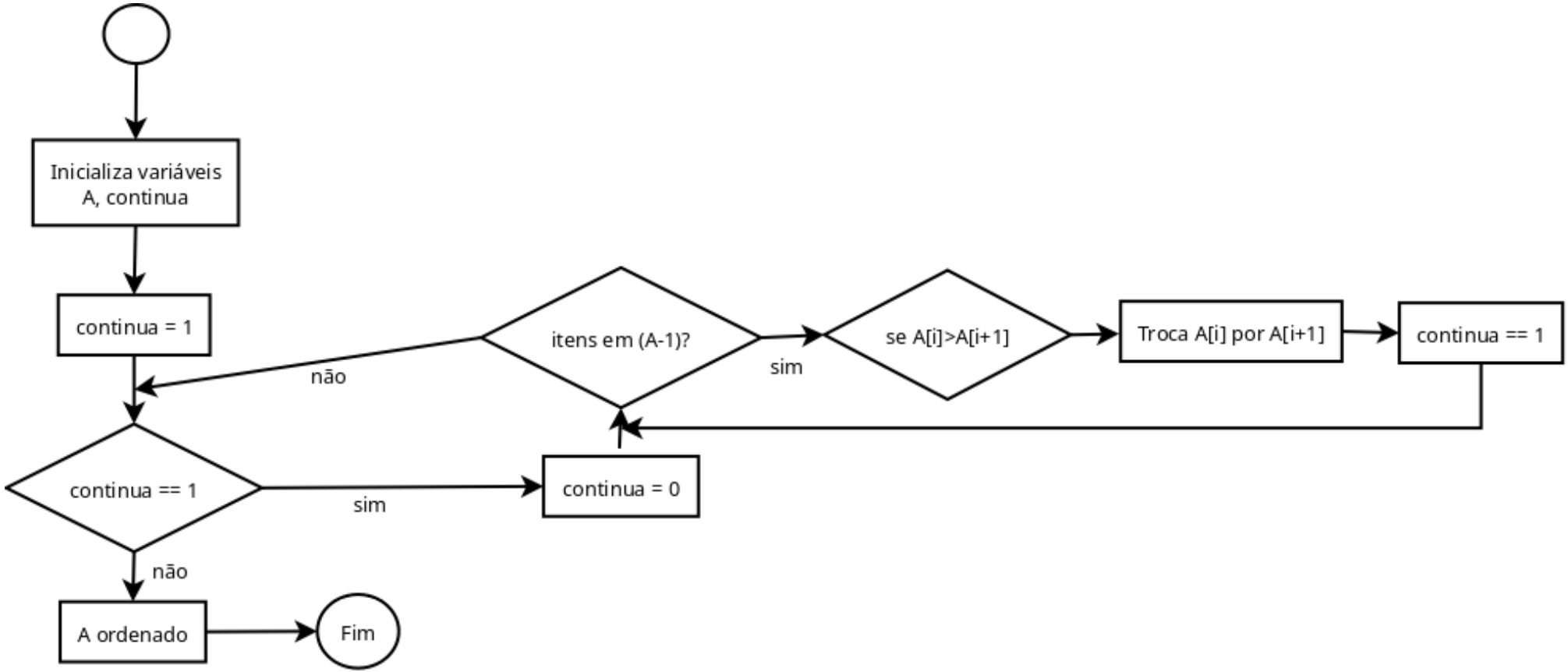
Exemplo considerando A=<3,5,1,2,4>



```
BubleSort(A)
  Continua = 1
  Enquanto continua == 1
    // interrompe se não houve trocas
    Continua = 0
    Para i = 0 até A.comprimento -1
    Se A[i] > A[i+1]
       Troca A[i] por A[i+1]
       Continua = 1
```

Pseudo código da ordenação por seleção

Fluxograma ordenação por seleção



Tempo de execução da ordenação por seleção

- Melhor caso
  - Quando os elementos já estão ordenados
  - O(N)
- Pior caso
  - Quando os elementos estão ordenados na ordem inversa
  - O(N^2)

Desafio

- Crie um programa que gere um vetor com números aleatórios e ordene usando os algoritmos de ordenação aprendidos nessa aula.
- O usuário pode escolher qual o algoritmo será utilizado para ordenar o vetor
- Extra: calcule o tempo de execução de cada algoritmo