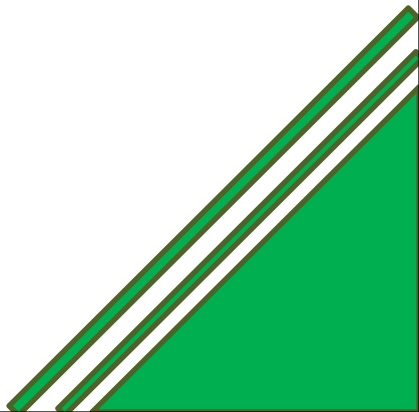
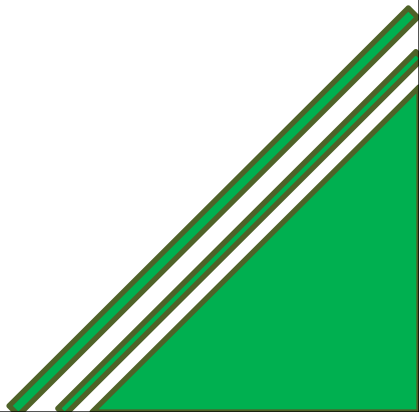
Estrutura de Dados 2

Prof.ª Franciny Medeiros
Franciny@ufj.edu.br



Algoritmos de Ordernação



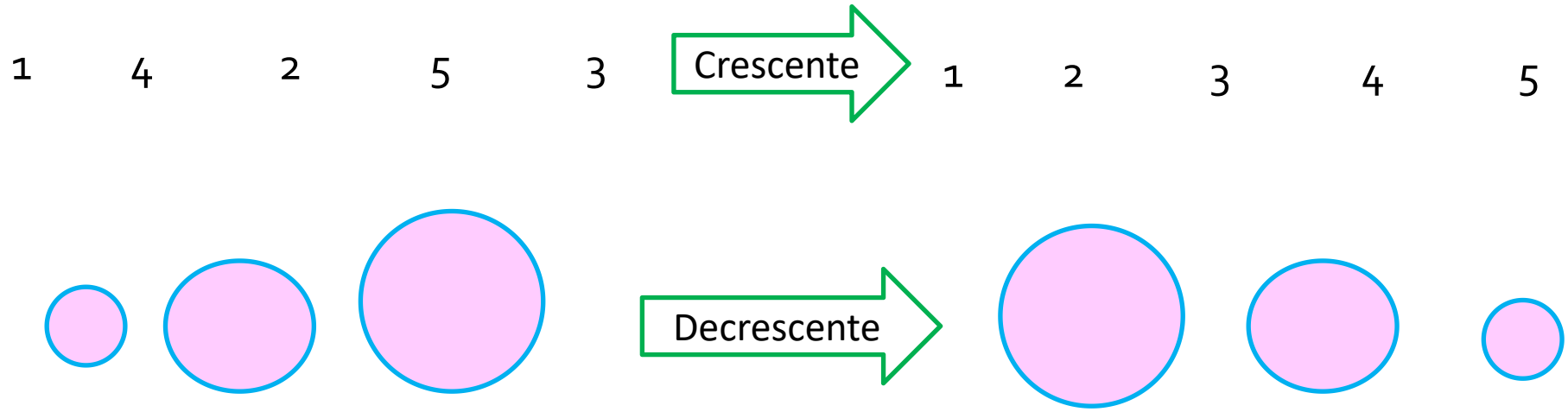
Objetivos

- Apresentar os métodos de ordenação mais importantes.

- Mostrar algoritmos para realizar uma mesma tarefa.

Ordenação

- Ordenar corresponde ao processo de rearranjar um conjunto de objetos em uma ordem específica.



Ordenação

- Objetivo da ordenação:
  - facilitar a recuperação posterior de elementos do conjunto ordenado.
    - Aumentar eficiência no acesso.

- Listas telefônicas.
- Dicionários.
- Tabelas e arquivos...

Notações

Sejam os itens

$$A_1, A_2, \ldots, A_n$$

Ordenar significa permutar estes itens em uma ordem

$$A_{k_1}, A_{k_2}, \ldots, A_{k_n}$$

Tal que, dada uma função de ordenação *f*, tem-se a seguinte relação

$$f(A_{k_1}) \leq f(A_{k_2}) \leq \ldots \leq f(A_{k_n})$$

Notações

- Qualquer tipo de chave, sobre o qual exista uma relação de ordem total <, para dada uma função de ordenação, pode ser utilizado.

A relação < deve satisfazer as condições:

- a < b OU a = b OU a > b
- Se a < b e b < c, então a < c.

Observações

- Qualquer tipo de dado sobre o qual exista uma regra de ordenação bem definida pode ser utilizado.

- Um método de ordenação é dito **estável**, se a ordem relativa dos itens com **chaves iguais** mantém-se inalterada pelo processo de ordenação.

Observações

Exemplo: uma lista de funcionários é ordenada pelo campo salário, então um método de ordenação estável produz uma lista em que os funcionários com mesmo salário aparecem em ordem alfabética.

Nome	Salário
João	R$ 1000,00
Maria	R$ 1350,00
Anita	R$ 1000,00

Método de ordenação estável

Nome	Salário
Anita	R$ 1000,00
João	R$ 1000,00
Maria	R$ 1350,00

Observações

- Para a apresentação dos algoritmos será utilizado:

Um ***arranjo A*** composto por uma repetição do tipo de dados ***item***, onde ***item*** é composto por um tipo inteiro.

Métodos de Ordenação

- Ordenação por seleção.
- Ordenação por inserção.
- Ordenação por permutação/troca.
- Ordenação por inserção com incrementos.
- Ordenação de árvores.
- Ordenação por particionamentos.

Métodos de ordenação com o principio de distribuição

- Métodos que não fazem ordenação baseados em comparação de chaves.

Exemplo: Ordem um baralho, onde

A < 2 < 3 < J < Q < K

♣ < ♦ < ♥ < ♠

Ordenando o baralho...

1. Distribuir as cartas em 6 montes, colocando em cada monte todos os ases, todos os dois,..., todos os reis.
2. Colete os montes na ordem (ás primeiro, depois dois...).
3. Distribuir as cartas em 4 montes, colocando em cada monte o naipe correspondente.
4. Colete os montes na ordem (paus, ouros, copas e espadas).

Métodos de ordenação com o principio de distribuição

- Também conhecidos como ordenação digital, radixsort ou bucketsort.

- Exemplos:
  - Classificadoras de cartões perfurados utilizam o princípio da distribuição para ordenar uma massa de cartões.
  - Carteiro no momento de distribuir as correspondências por rua ou bairro

Métodos de ordenação com o principio de distribuição

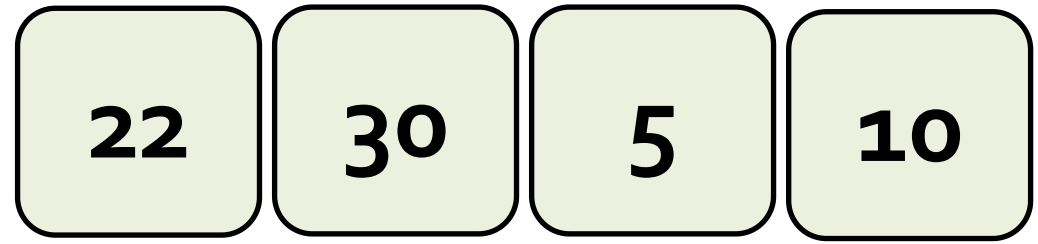
- Dificuldades:
  - Problema para lidar com cada “monte”.
  - Cada monte é reservado um área, isso demanda memória extra.
  - Complexidade linear.

Ordenação por Seleção

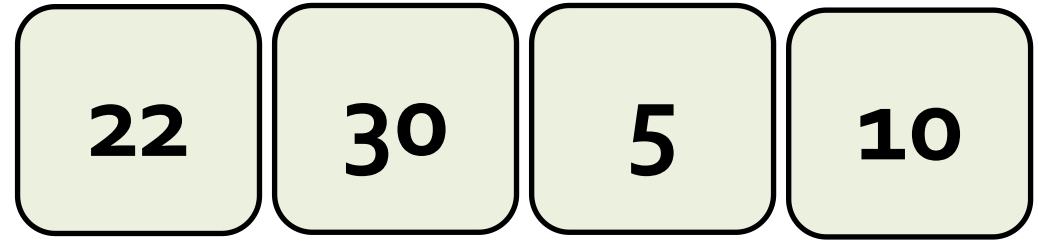
(Selection sort)

Passos:

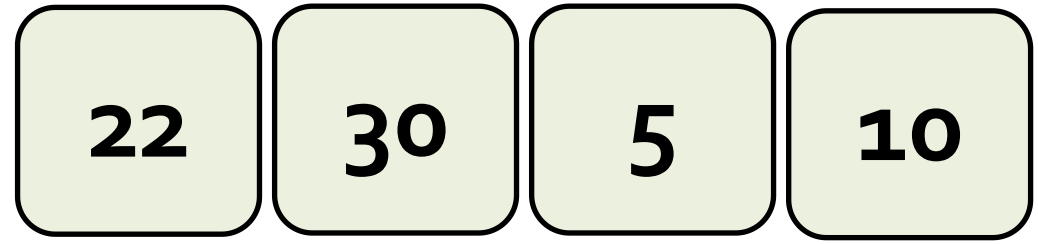
1. Selecione o menor item.
2. Troque este item com o item A[1]
3. Repita o passo 1 e 2 com os n-1 itens restantes, depois com os n-1 itens restantes...



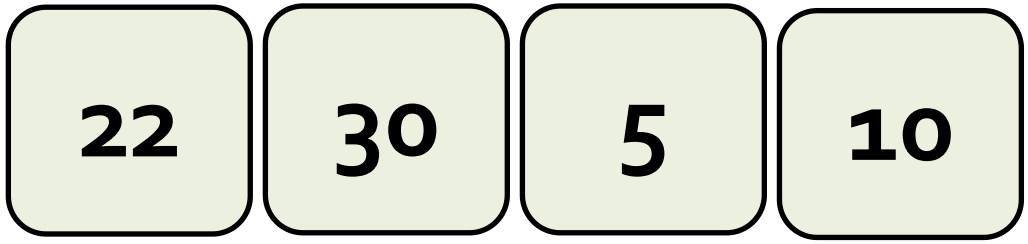
Menor item: 22



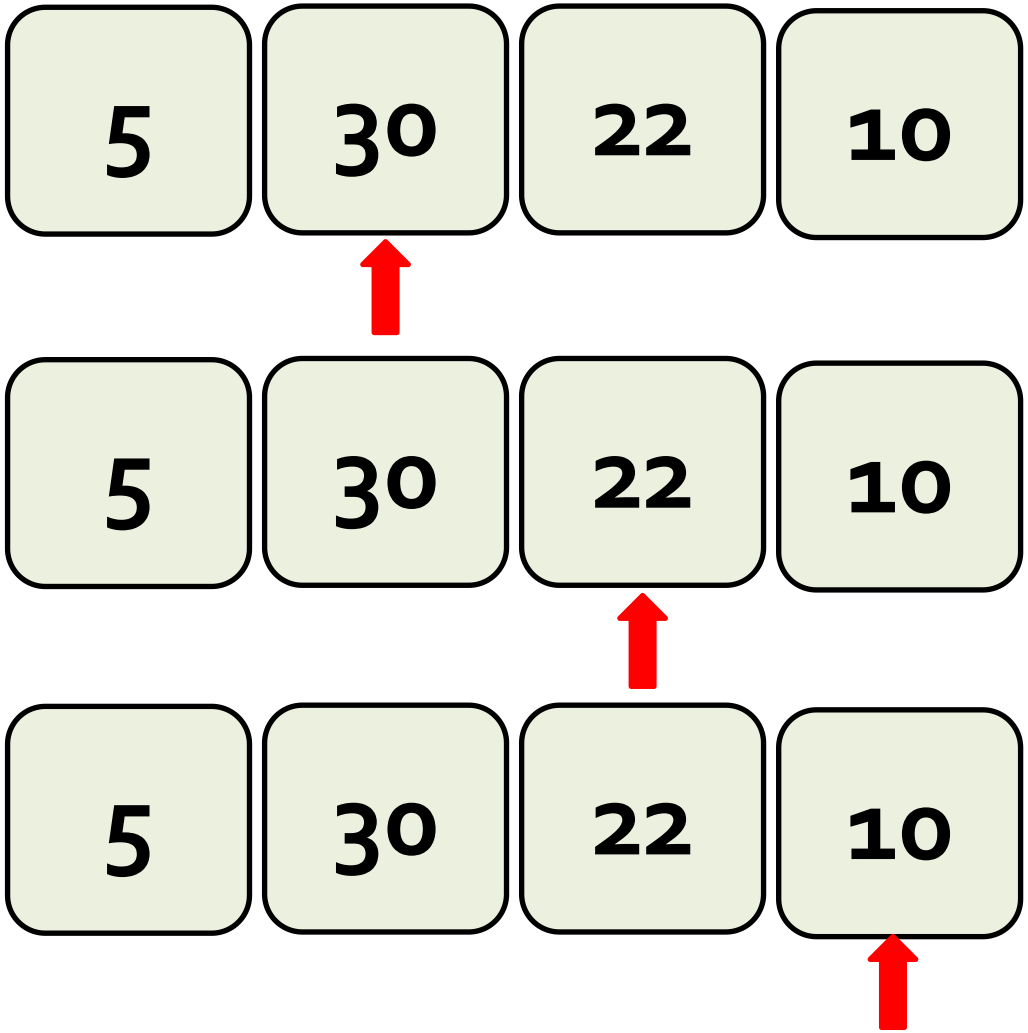
Menor item: 22



Menor item: 5



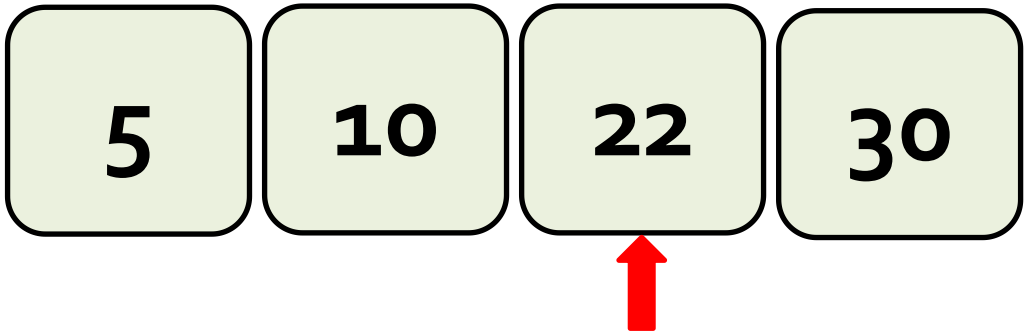
Menor item: 5
Troca 5 com 22 (A[i])



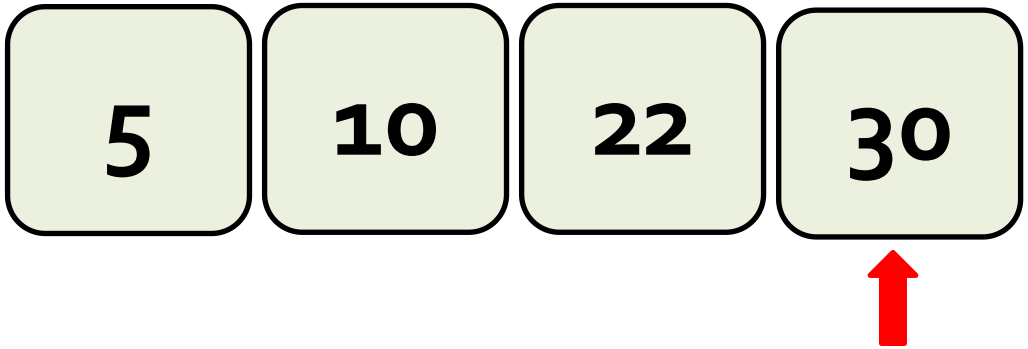
Menor item: 30

Menor item: 22

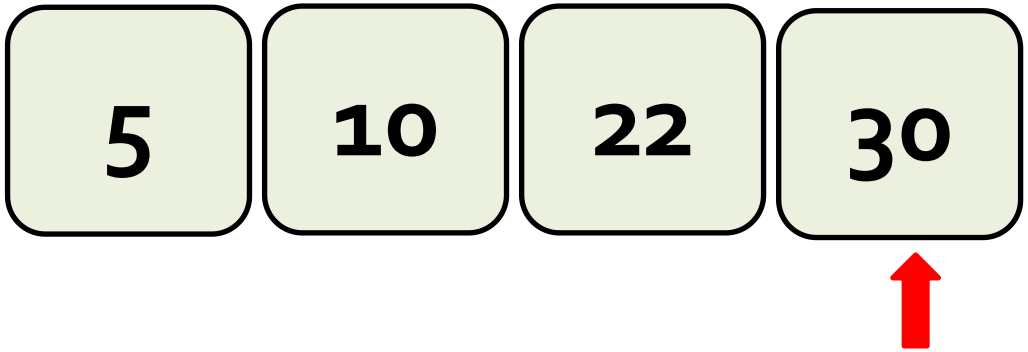
Menor item: 10
Troca 10 com 30 (A[2])



Menor item: 22



Menor item: 22
Não faz nada.



Menor item: 30
Não faz nada.
Fim.

Ordenação por Seleção

(Selection sort)

```
1.  selection-sort(A)
2.   inteiro i, j, min
3.   item t
4.    para i <- 0 até n - 1 faça
5.     min <- i
6.      para j <- i+1 até n faça
7.       se A[j] < A [min] então
8.        min <- j
         fim-para
9.    t <- A[min]
10.   A[min] <- A[i]
11.   A[i] <- t;
12.
13.   fim-para
14. fim-algoritmo
```

Ordenação por Seleção

(Selection sort)

Vantagens

- ✓ método mais simples de ordenação;
- ✓ método adequado para arquivos pequenos.

Desvantagens

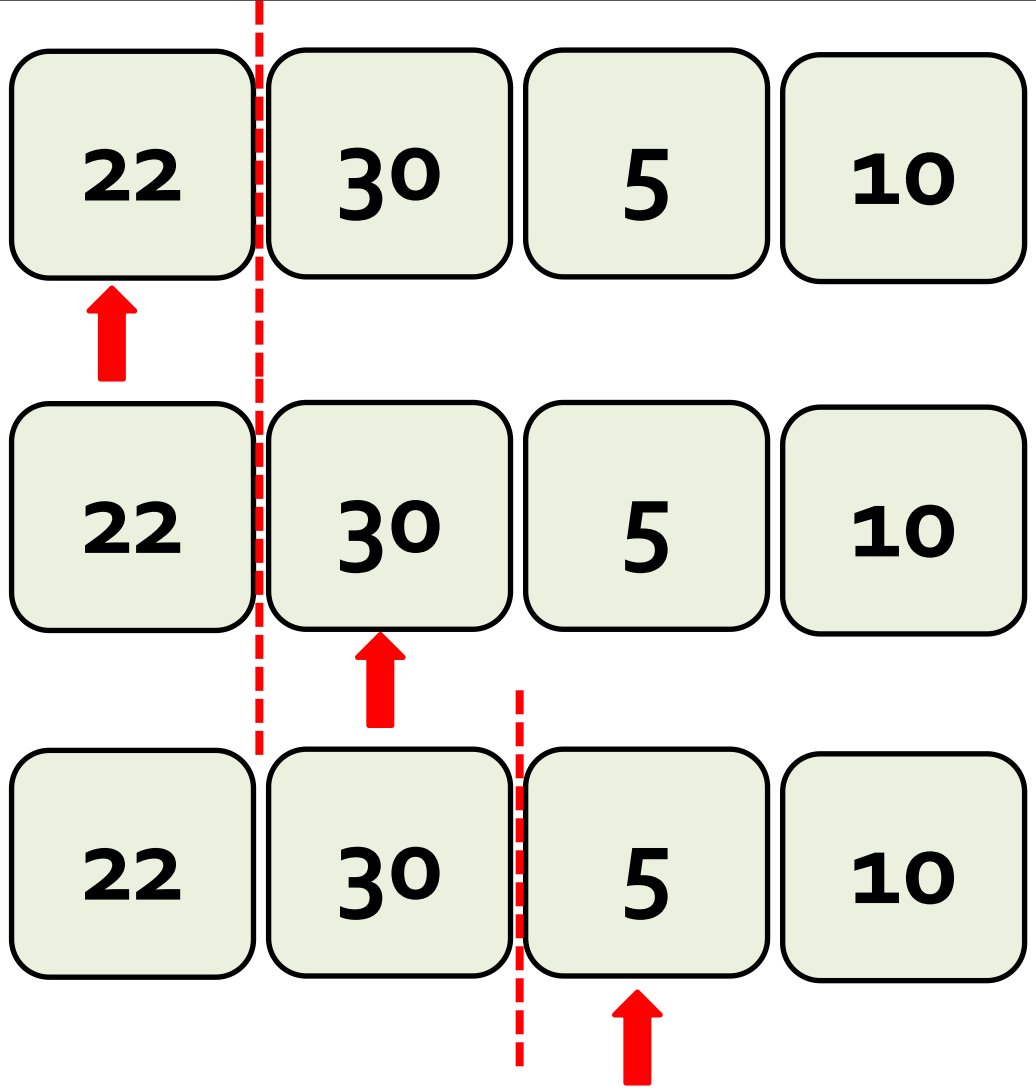
- ➢ complexidade computacional quadrática $\theta(n^2)$.

Ordenação por Seleção *(Insertion sort)*

Passos:

1. ordenar A utilizando um sub-arranjo ordenado localizado em seu inicio

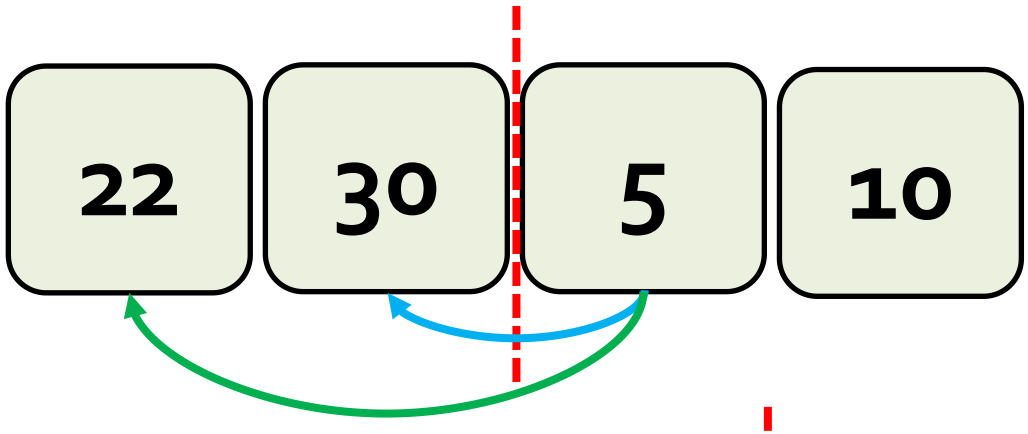
2. a cada passo, inserir ordenadamente a este sub-arranjo mais um elemento, até o último elemento de A



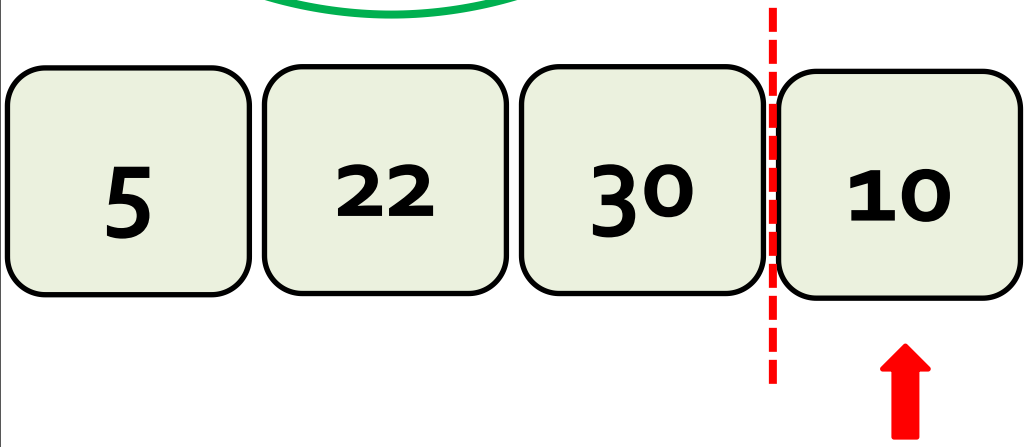
sub-A = {22}
sub-B = {30, 5, 10}

22 < 30
sub-A = {22, 30}
sub-B = {5, 10}

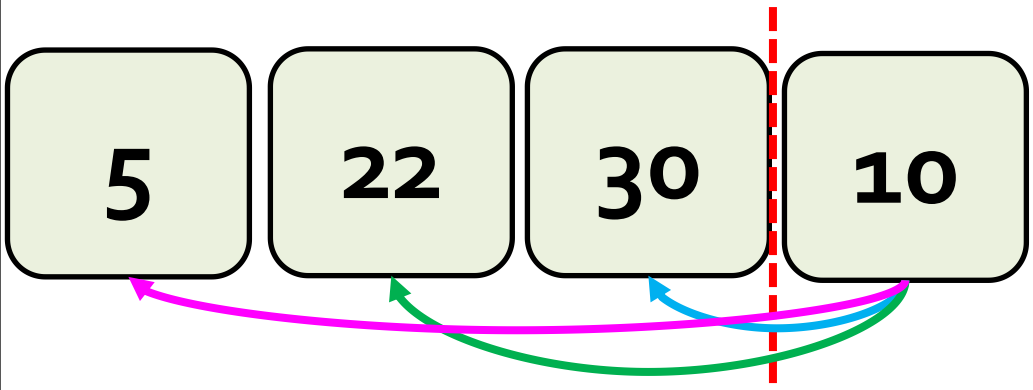
5 < 30



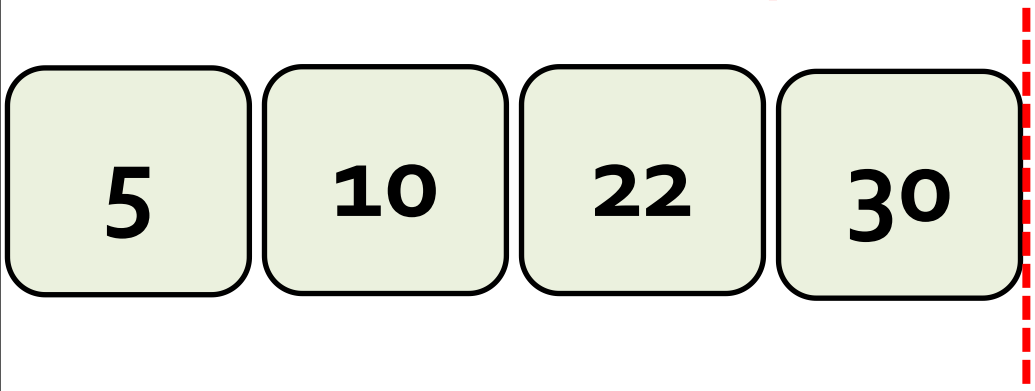
5 < 30 = T
5 < 22 = T
Inserir 5



sub-A = {5, 22, 30}
sub-B = {10}



10 < 30 = T
10 < 22 = T
10 < 5 = F
Inserir 10



sub-A = {5, 22, 30}
sub-B = {10}

Ordenação por Inserção

(Insertion sort)

```
1.  insertion-sort(A)
2.    inteiro i, j
3.    item t
4.    para i <- 2 até n faça
5.      t <- A[i], j <- i-1, A[0] <- t
6.      enquanto t < A[j] faça
7.        A[j+1] <- A[j]
8.        j <- j -1
9.      fim-enquanto
10.     A[j+1] <- t
11.   fim-para
12.  fim-algoritmo
```

Ordenação por Inserção

(Insertion sort)

Vantagens

- ✓ método adequado para quando o arquivo está quase ordenado;
- ✓ complexidade linear θ(n).

Desvantagens

- ➢ problemático com arquivos grandes.

Ordenação por Troca
(Bubble sort)

Passos

1. comparar a posição i e i+1 e trocar ordenadamente.
2. Repetir o passo 1 até n.

Ordenação por Troca

(Bubble sort)

```
void bubble_sort(int vetor[], int n){
    int i, j, aux;
    for (i = 0; i < n-1; i++){
      for (j = 0; j < n-1-i; j++){
            if (vetor[j] > vetor[j+1]){
               aux = vetor[j];
               vetor[j] = vetor[j+1];
               vetor[j+1] = aux;
             }
      }
    }
}
```

Atividade

- Execute o teste de mesa para o algoritmo do bubble sort, insertion sort e selection sort com a sequência: 22, 43, 4, 16

Shell Sort

- O shell sort é um algoritmo de ordenação criado por Donald Shell em 1959.

- O shell sort tem complexidade quadrática
  - O algoritmo de ordem quadrática mais eficiente.

Shell Sort

- Melhoria do método de ordenação por inserção.

- O algoritmo passa várias vezes pela lista dividindo o grupo maior em menores
  - Nos grupos menores é aplicado o método da ordenação por inserção.

Shell Sort - funcionamento

- Número de elementos e divide por 2.
  - Exemplo: r = 8/2 = 4
- Depois, divide r por 2 novamente...
  - Até que r = 1.

- Quando r for igual a 1, os elementos estarão ordenados.