

Listas Lineares, Pilhas, Filas e Busca Binária

Sumário

- Referência Bibliografia da Aula
- Repositório *Git*
- Introdução
- Estruturas de Dados

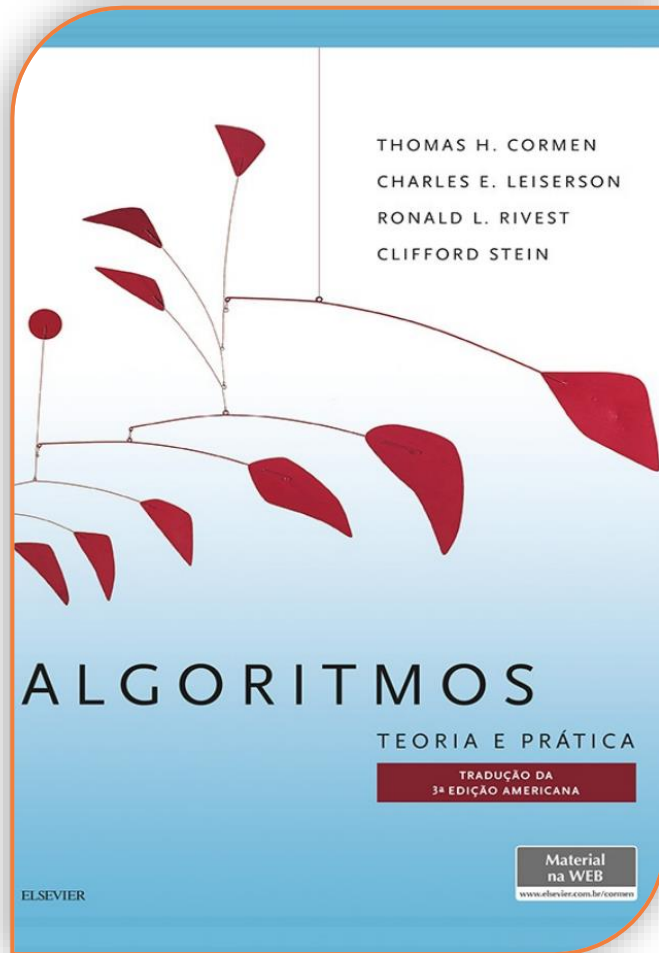



Sumário

- Listas Lineares
- Pilhas
- Filas
- Busca Binária



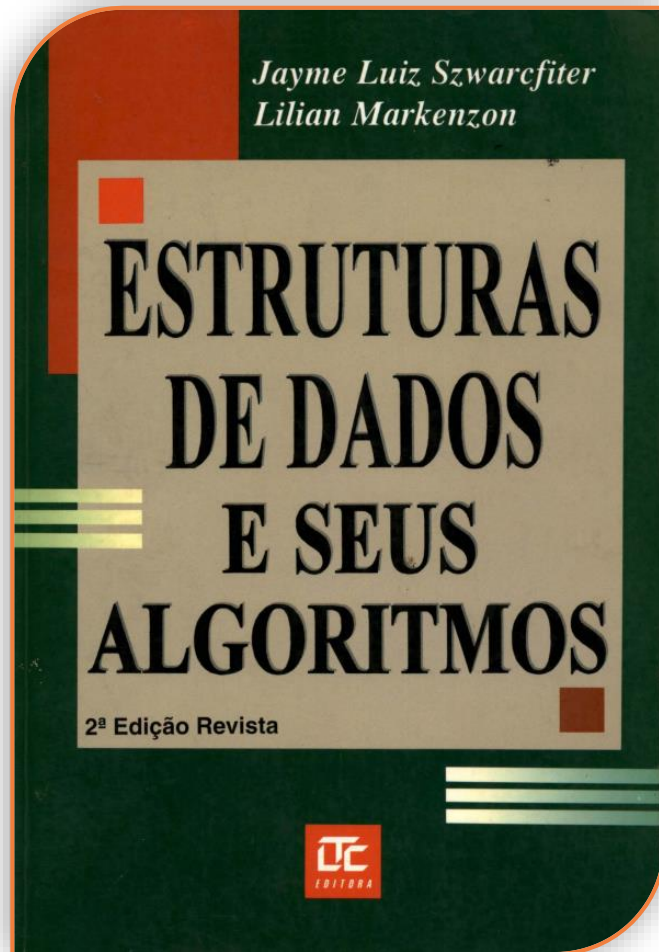
Referência Bibliografia da Aula



-  CORMEN, T.H. et al. **Algoritmos: Teoria e Prática**. 3ª ed. Rio de Janeiro: Elsevier, 2012.




Referência Bibliografia da Aula



-  SZWACTFITER, J.L. et al. **Estruturas de Dados e Seus Algoritmos**. 2ª ed. Editora LTC, 1997.




Objetivo Geral da Aula

-  Os estudantes devem conhecer a **Estrutura de Dados** do tipo **Lista**, compreendendo seu funcionamento interno e sabendo aplica-la na resolução de problemas.




Objetivo Geral da Aula

-  Os estudantes devem conhecer as **Estruturas de Dados** dos tipos **Pilhas** e **Filas**, suas diferentes formas de implementação sabendo aplica-las na resolução de problemas.




Objetivo Geral da Aula

-  Os estudantes devem conhecer a técnica de **Busca Binária**, compreendendo o quanto ela pode acelerar as buscas, suas limitações, formas de implementação, e saber aplicá-la na resolução de problemas reais.



Repositório *Git*

- github.com/whoisraibolt/UNIFESO-CCOMP-EDP
-  Repositório com o conteúdo da disciplina **Estrutura de Dados e Paradigmas** do curso **Ciência da Computação** da **UNIFESO** — Centro Universitário Serra dos Órgãos.



Introdução

- **Algoritmo:**

- Procedimento computacional bem definido que, a partir da manipulação de um dado gerado de **entrada** produz um outro dado de **saída**.



Introdução

- **Tipo Abstrato de Dados:**
 - Dados manipulados e dispostos de maneira **homogênea**.
 - Composto por **modelo matemático** e **conjunto de operações** definidos sobre o modelo em questão.



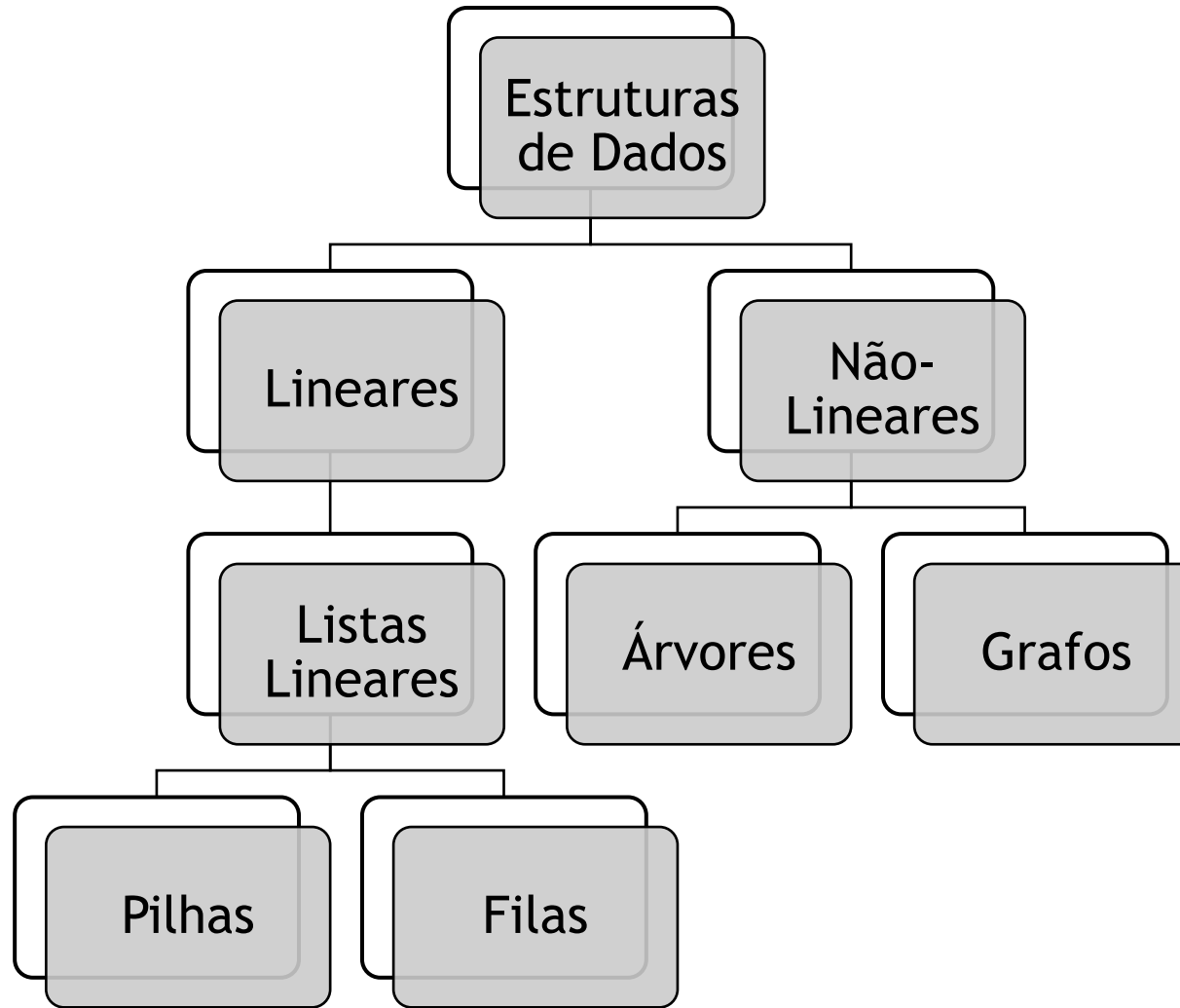
Introdução

- **Estruturas de Dados:**

- Empregada na representação do **modelo matemático**.
- **Formas de organizar dados** de modo a atender aos **diferentes requisitos de processamento**, levando em consideração suas **relações lógicas**.



Estruturas de Dados



Listas Lineares



Listas Lineares

- Uma **Lista Linear** é uma **Estrutura de Dados Linear** de **manipulação** mais **simples**, que agrupa informações referentes a um conjunto de elementos que, de alguma forma, se **relacionam entre si**.
- **Exemplos:**
 - **Notas de alunos, informações sobre clientes de uma empresa, entre outros.**

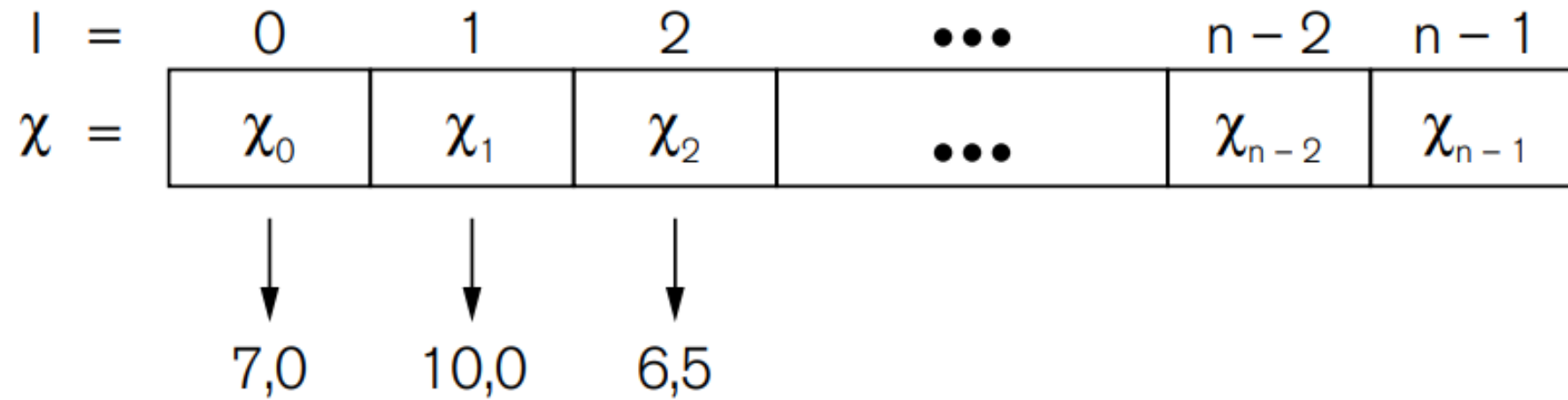


Listas Lineares

- Na **Lista Linear** cada **nó** é **precedido** por um **nó** e **sucedido** por outro **nó**.
- **Exceto** o **primeiro nó**, que não possui **predecessor** e o **último nó**, que não possui **sucessor**.



Listas Lineares



Cliente				
Cód. Cliente	Nome	Endereço	Fone	RG



Listas Lineares

- Em uma **Lista Linear**, cada **nó** é formado por **campos**.
- Que armazenam as características distintas dos elementos da lista.

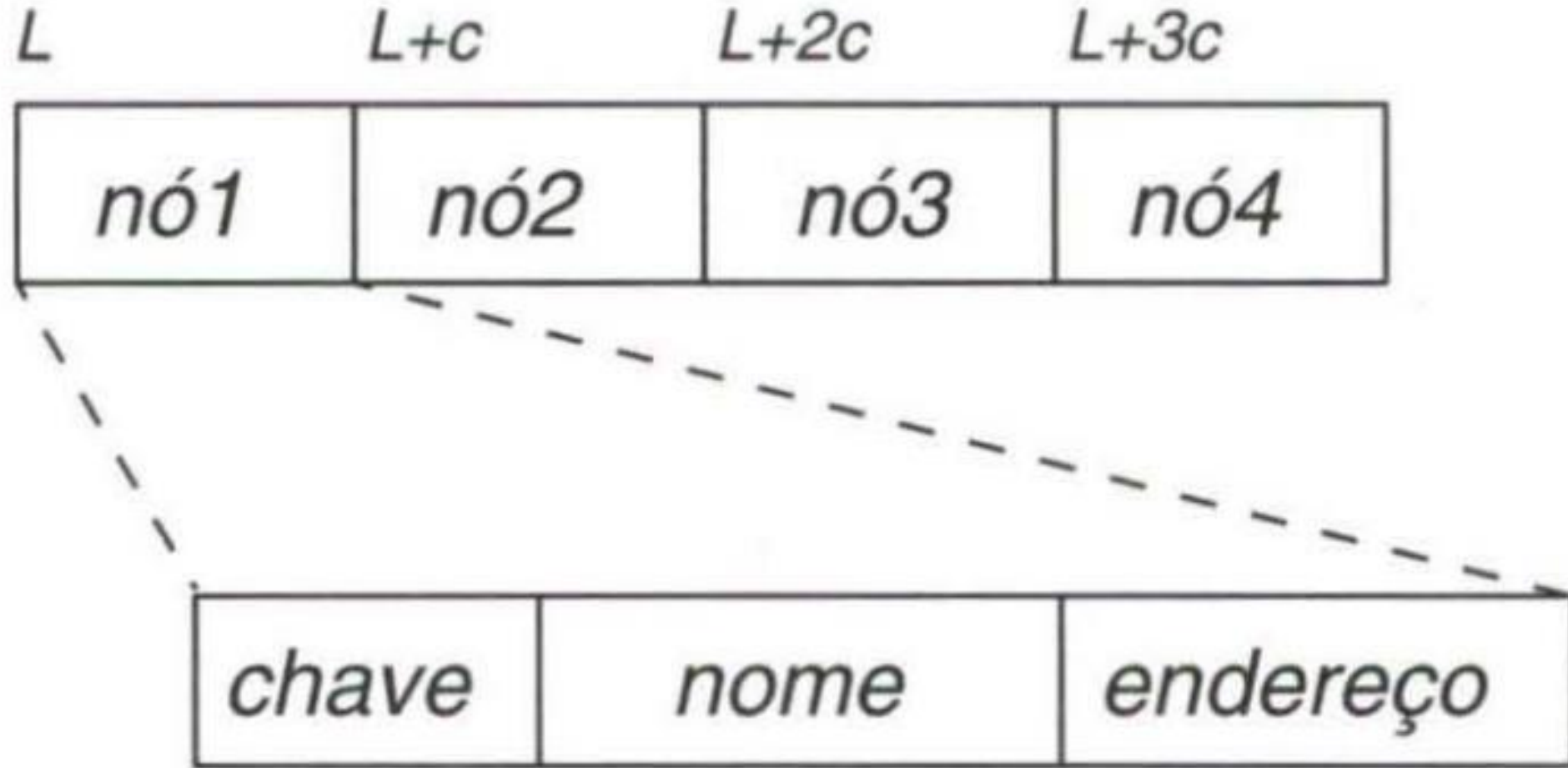


Listas Lineares

- Cada **nó** possui:
 - **Nome**: Nome da Lista.
 - **Chave**: **Identificador**; **elemento**; **valor** a ser encontrado.
 - **Endereço**: Posição; índice em que a **Chave** se encontra.



Listas Lineares



Listas Lineares

- **Tipos de Armazenamento:**

- Podem ser classificados de acordo com a **posição relativa** na memória, ou seja, posição contígua ou posição não contígua de **dois nós consecutivos** na lista.
 - **Alocação Sequencial** (posição contígua);
 - **Alocação Encadeada** (posição não contígua).



Listas Lineares

- Operações básicas:
 - Busca, inclusão e remoção de um determinado elemento.
- Por serem consideradas básicas, é **necessário** que os **algoritmos** que as implementem sejam **eficientes**.



Listas Lineares

- **Algoritmo 2.1:** Busca de um elemento na lista L

```
função busca1(x)  
  i := 1;  busca1 := 0  
  enquanto  $i \leq n$  faça  
    se  $\mathcal{L}[i].chave = x$  então  
      busca1 := i                                % chave encontrada  
      i :=  $n + 1$   
    senão i := i + 1                                % pesquisa prossegue
```



Listas Lineares

- **Algoritmo 2.2:** Busca de um elemento na lista L

```
função busca( $x$ )  
   $\mathcal{L}[n + 1].chave := x; \quad i := 1$   
  enquanto  $\mathcal{L}[i].chave \neq x$  faça  
     $i := i + 1$   
  se  $i \neq n + 1$  então  
     $busca := i$                                 % elemento encontrado  
  senão  $busca := 0$                              % elemento não encontrado
```



Listas Lineares

- O **algoritmo 2.2** se propõe a efetuar a mesma busca que o **algoritmo 2.1**.
- Entretanto, com a criação de um **novo nó**, que possui o **valor procurado** no **campo chave**, na posição **$n + 1$** . Dessa forma, o algoritmo sempre encontra um nó da tabela com as **características desejadas**.



Listas Lineares

- **Algoritmo 2.3:** Busca de um elemento na lista L , ordenada

```
função busca-ord( $x$ )  
   $\mathcal{L}[n + 1].chave := x; \quad i := 1$   
  enquanto  $\mathcal{L}[i].chave < x$  faça  
     $i := i + 1$   
  se  $i = n + 1$  ou  $\mathcal{L}[i].chave \neq x$  então  
     $busca-ord := 0$   
  senão  $busca-ord := i$ 
```



Listas Lineares

- **Algoritmo 2.5:** Inserção de um nó na Lista L

```
se  $n < M$  então  
    se  $busca(x) = 0$  então  
         $\mathcal{L}[n + 1] := novo\text{-}valor$   
         $n := n + 1$   
    senão “elemento já existe na tabela”  
senão overflow
```



Listas Lineares

- **Algoritmo 2.6:** Remoção de um nó na Lista L

```
se  $n \neq 0$  então  
     $indice := busca(x)$   
    se  $indice \neq 0$  então  
         $valor-recuperado := \mathcal{L}[indice]$   
        para  $i := indice, n - 1$  faça  
             $\mathcal{L}[i] := \mathcal{L}[i + 1]$   
         $n := n - 1$   
    senão “elemento não se encontra na tabela”  
senão underflow
```



Listas Lineares

- O **algoritmo 2.5** apresenta a **inserção** de um **nó** contido na variável novo de **chave x**.
- O **algoritmo 2.6** efetua a **remoção** de um **nó** sendo conhecido, no caso a **chave x**.
- Ambos os algoritmos consideram **Listas** não ordenadas.



Listas Lineares

- A **memória** pressuposta disponível tem $M + 1$ posições.
- **Overflow:**
 - Inserção em uma Lista que já ocupa M posições.
- **Underflow:**
 - Remoção de um elemento em uma Lista vazia.



Listas Lineares

- **Casos Particulares:**

- **Deque** (Abreviatura do inglês "double ended queue"): Inserções e remoções são **permitidas** nas **duas extremidades** da lista.
- **Pilha**: Inserções e as remoções são realizadas **somente** em **um extremo**.
- **Fila**: Inserções são **realizadas** em **um extremo** e remoções em **outro**.



Pilhas e Filas



Pilhas e Filas

- O armazenamento sequencial de Listas é empregado quando as estruturas sofrem poucas inserções e remoções.
- Existem estruturas que fazem uso de indicadores especiais, denominados ponteiros, para o acesso a posições selecionadas.



Pilhas

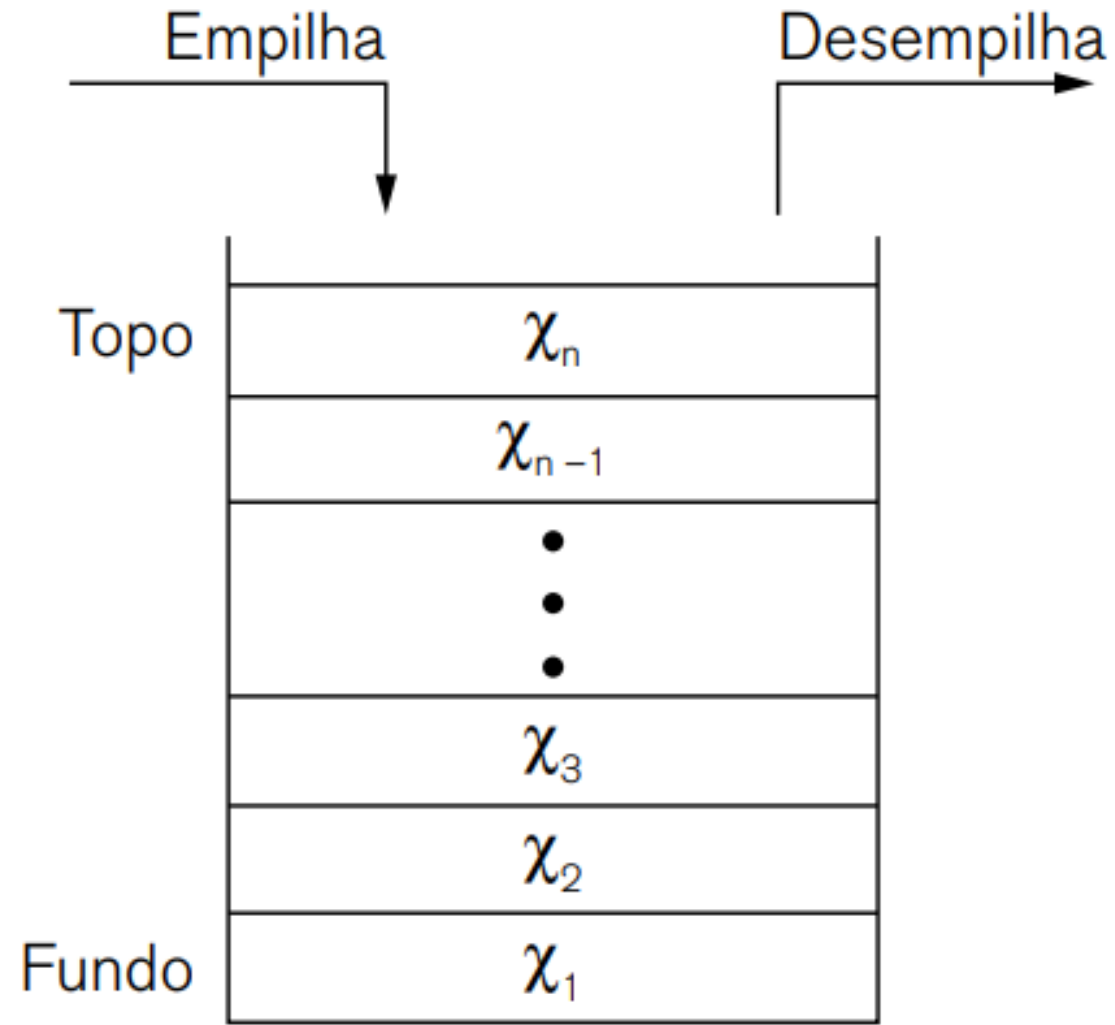


Pilhas

- No caso da **Pilha**, apenas um ponteiro precisa ser considerado:
 - **Ponteiro topo.**
- As **inserções e remoções** são **executadas** na **mesma extremidade** da lista.



Pilhas



Pilhas

- **Algoritmo 2.7:** Inserção na Pilha P

```
se  $topo \neq M$  então  
     $topo := topo + 1$   
     $\mathcal{P}[topo] := novo\text{-}valor$   
senão overflow
```



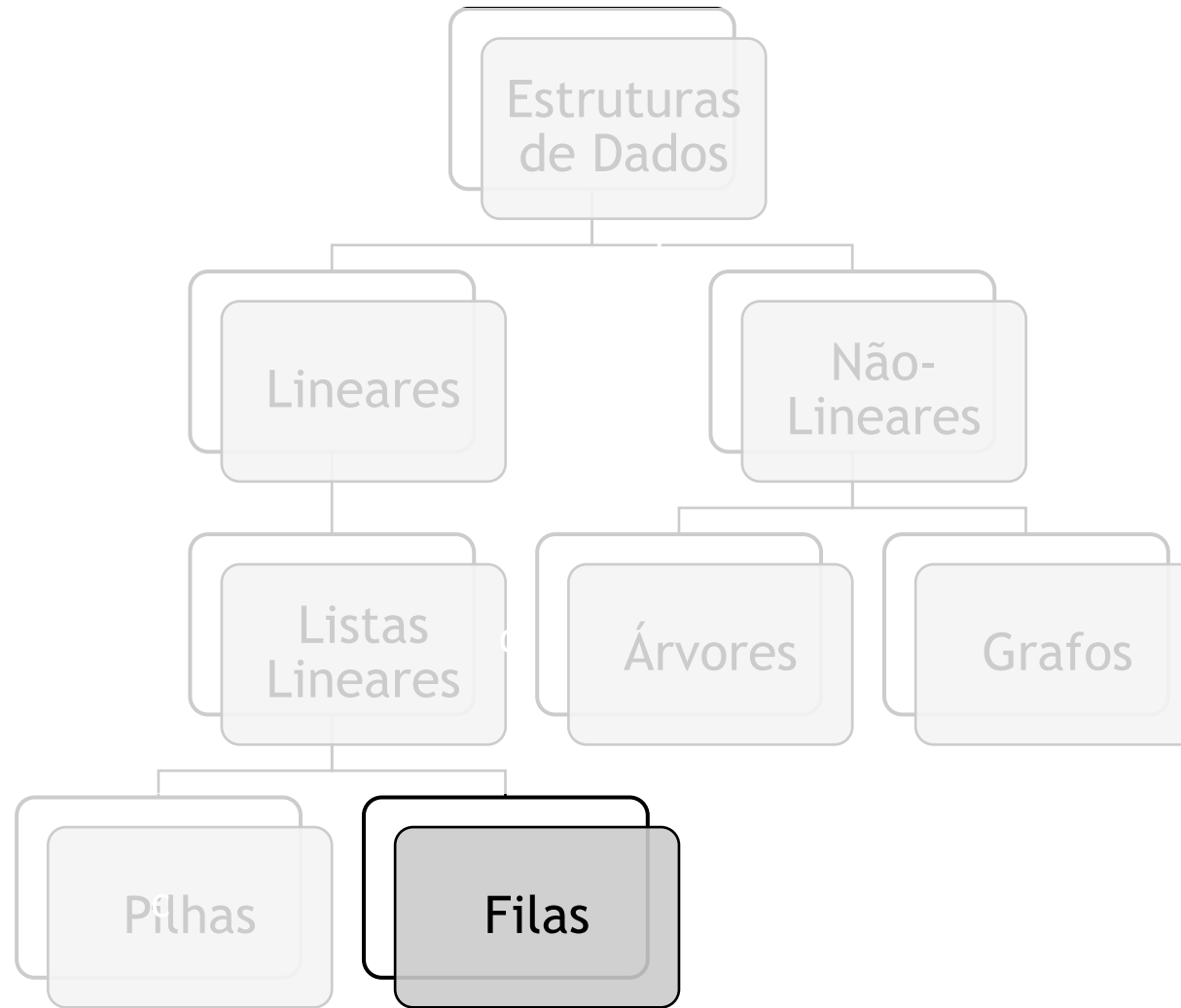
Pilhas

- **Algoritmo 2.8:** Remoção na Pilha P

```
se  $topo \neq 0$  então  
     $valor-recuperado := \mathcal{P}[topo]$   
     $topo := topo - 1$   
senão underflow
```



Filas

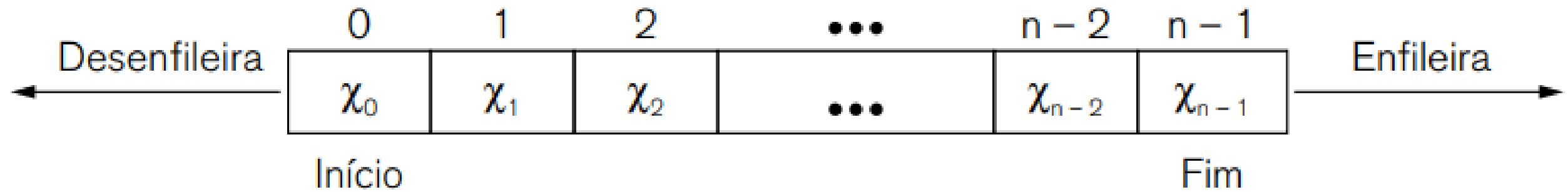


Filas

- As **Filas** exigem uma implementação um pouco mais elaborada.
- São necessários dois ponteiros:
 - Início de fila (***f***) e;
 - Fim (***r***).
- Para a inserção de um elemento, move-se o ponteiro ***r***.
- Para a remoção, move-se o ponteiro ***f***.



Filas



Filas

- **Algoritmo 2.9:** Inserção na Fila *F*

```
prov := r mod M + 1  
se prov ≠ f então  
    r := prov  
     $\mathcal{F}[r] := \text{novo-valor}$   
    se f = 0 então  
        f := 1  
senão overflow
```



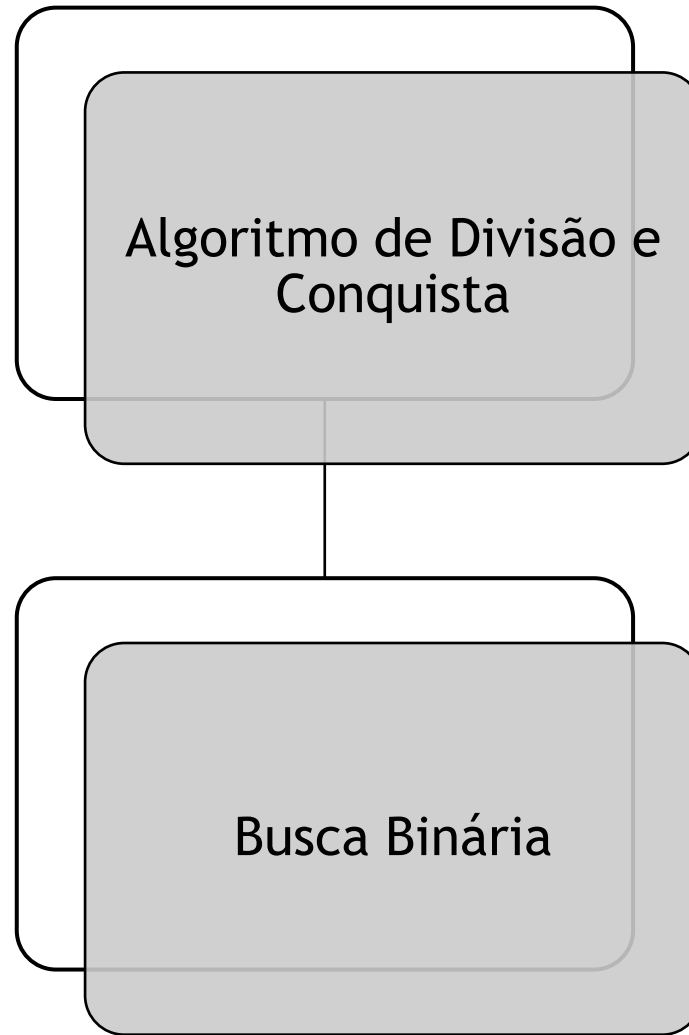
Filas

- **Algoritmo 2.10:** Remoção na Fila F

```
se  $f \neq 0$  então  
     $valor-recuperado := \mathcal{F}[f]$   
    se  $f = r$  então  
         $f := r := 0$   
    senão  $f := f \bmod M + 1$   
senão underflow
```



Busca Binária



Busca Binária

- A **Busca Binária** permite percorrer as **Listas**, como se **folheia** uma **lista telefônica**, por exemplo.
- Em uma Lista, o **primeiro nó pesquisado** é o que se encontra no **meio**.
- Se a **comparação não é positiva**, **metade** da **Lista** pode ser **abandonada**, ou seja, o valor procurado se encontra ou na metade inferior (**se for menor**), ou na metade superior (**se for maior**).



Busca Binária

• Algoritmo 2.4: Busca Binária

```
função busca-bin(x)  
  inf := 1;  sup := n;  busca-bin := 0  
  enquanto inf ≤ sup faça  
    meio :=  $\lfloor (\textit{inf} + \textit{sup}) / 2 \rfloor$            % índice a ser buscado  
    se  $\mathcal{L}[\textit{meio}].\textit{chave} = x$  então  
      busca-bin := meio           % elemento encontrado  
      inf := sup + 1  
    senão se  $\mathcal{L}[\textit{meio}].\textit{chave} < x$  então  
      inf := meio + 1  
    senão sup := meio - 1
```



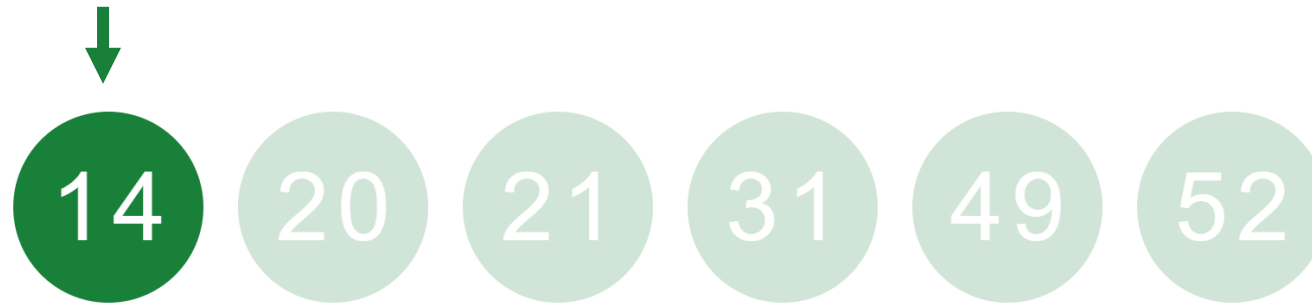
Busca Binária

- **Busca de um elemento:**
 - Queremos recuperar o elemento 31.



Busca Binária

- **Busca de um elemento:**
 - Queremos recuperar o elemento 31.



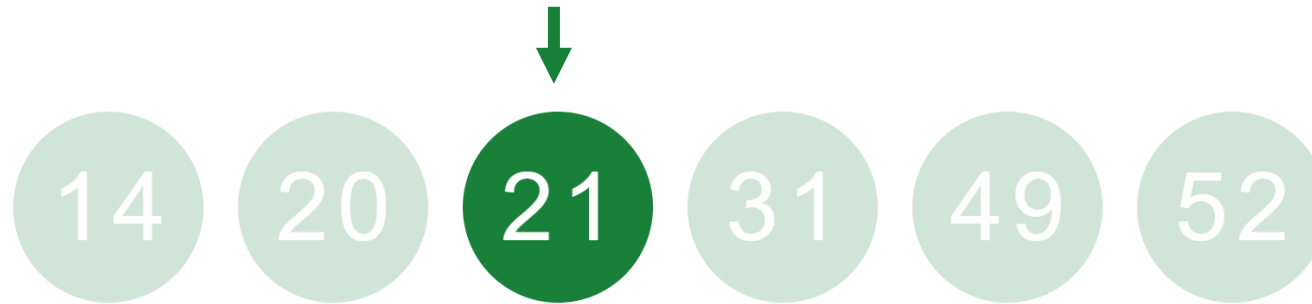
Busca Binária

- **Busca de um elemento:**
 - Queremos recuperar o elemento 31.



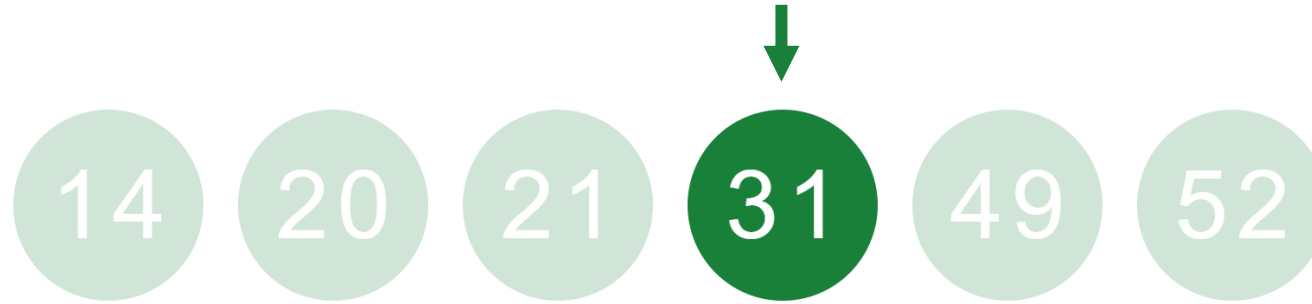
Busca Binária

- **Busca de um elemento:**
 - Queremos recuperar o elemento 31.



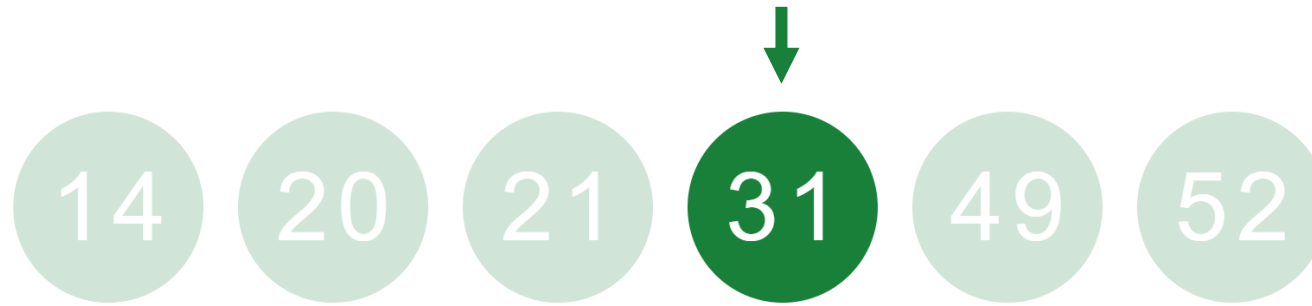
Busca Binária

- **Busca de um elemento:**
 - Queremos recuperar o elemento 31.



Busca Binária

- **Busca de um elemento:**
 - Queremos recuperar o elemento 31.



4 passos!



Busca Binária

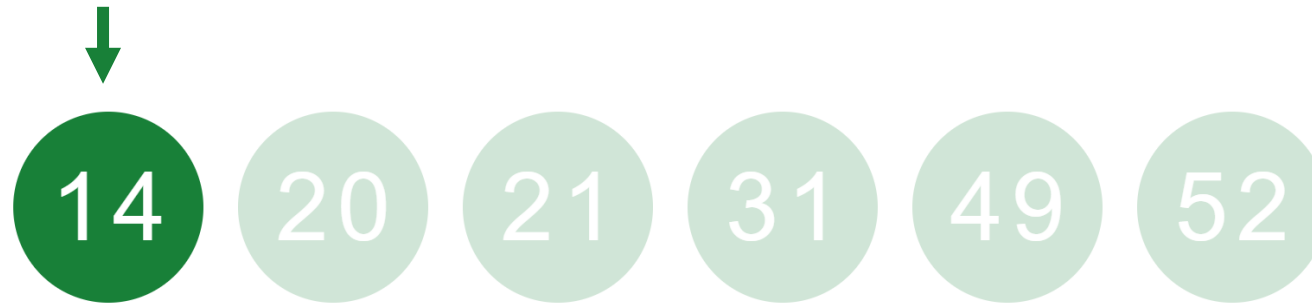
- **Busca de um elemento:**
 - Queremos recuperar o elemento 37.

14 20 21 31 49 52



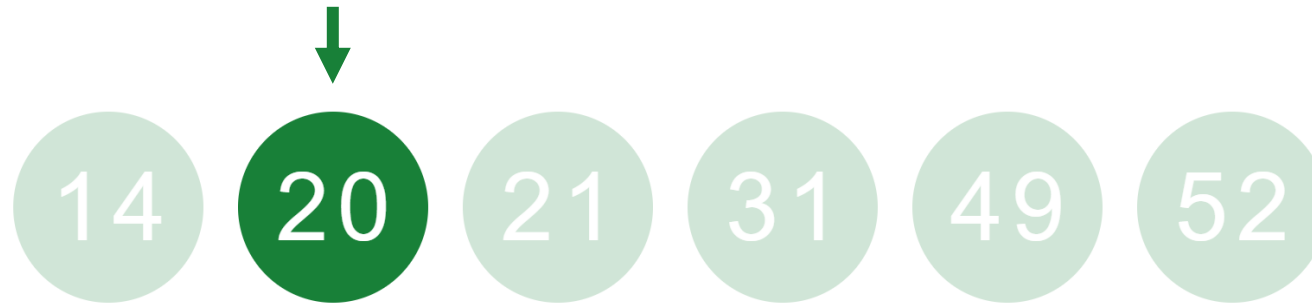
Busca Binária

- **Busca de um elemento:**
 - Queremos recuperar o elemento 37.



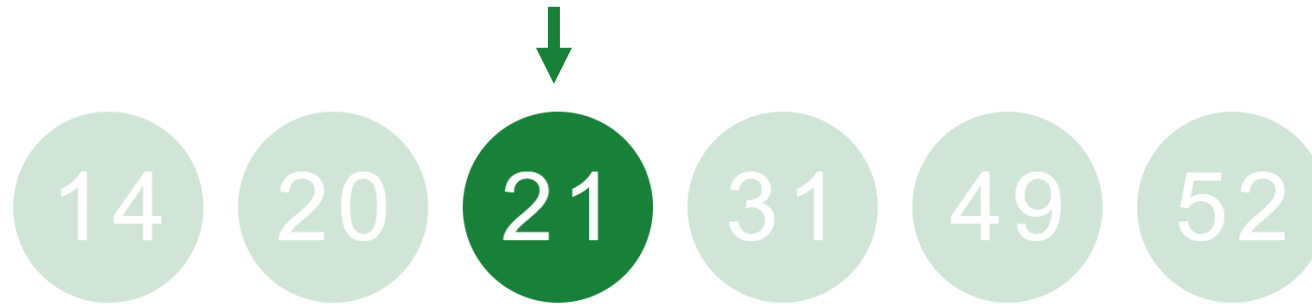
Busca Binária

- **Busca de um elemento:**
 - Queremos recuperar o elemento 37.



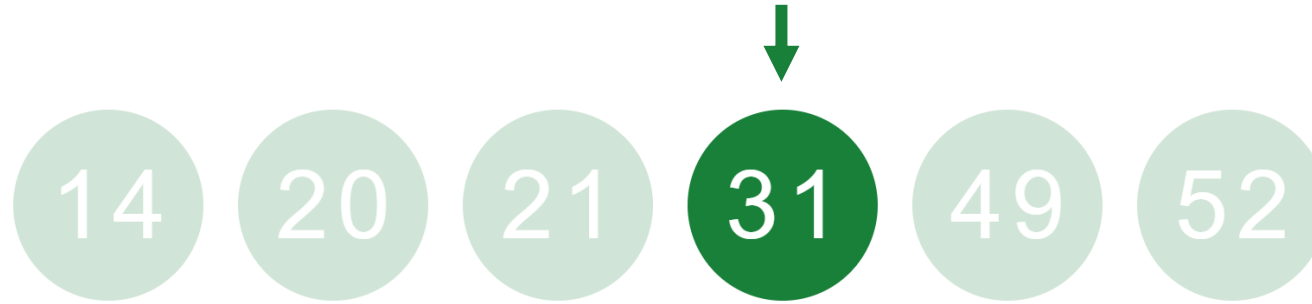
Busca Binária

- **Busca de um elemento:**
 - Queremos recuperar o elemento 37.



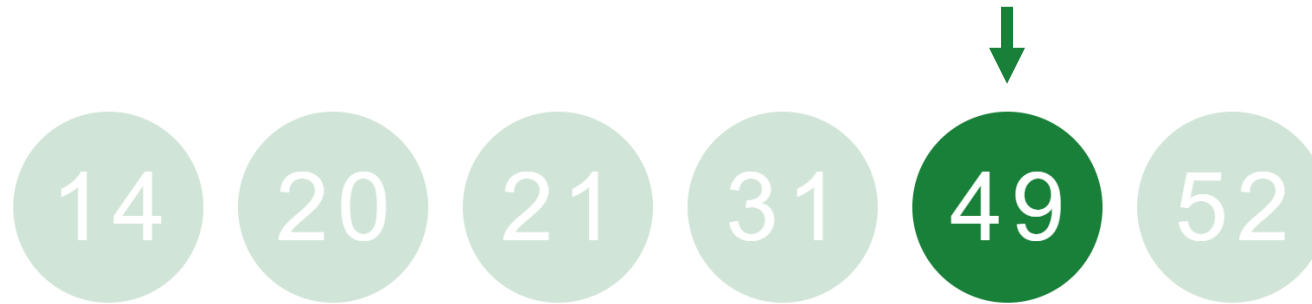
Busca Binária

- **Busca de um elemento:**
 - Queremos recuperar o elemento 37.



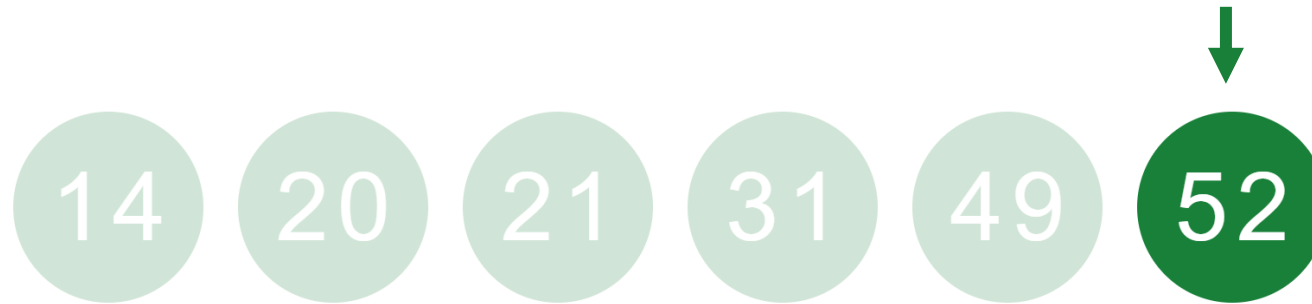
Busca Binária

- **Busca de um elemento:**
 - Queremos recuperar o elemento 37.



Busca Binária

- **Busca de um elemento:**
 - Queremos recuperar o elemento 37.



Busca Binária

- **Busca de um elemento:**
 - Queremos recuperar o elemento 37.



6 passos!
Podemos melhorar?



Busca Binária

- Busca de um elemento:
 - Lista ordenada. Queremos recuperar o elemento 37.

14 20 21 31 49 52



Busca Binária

- Busca de um elemento:
 - Lista ordenada. Queremos recuperar o elemento 37.



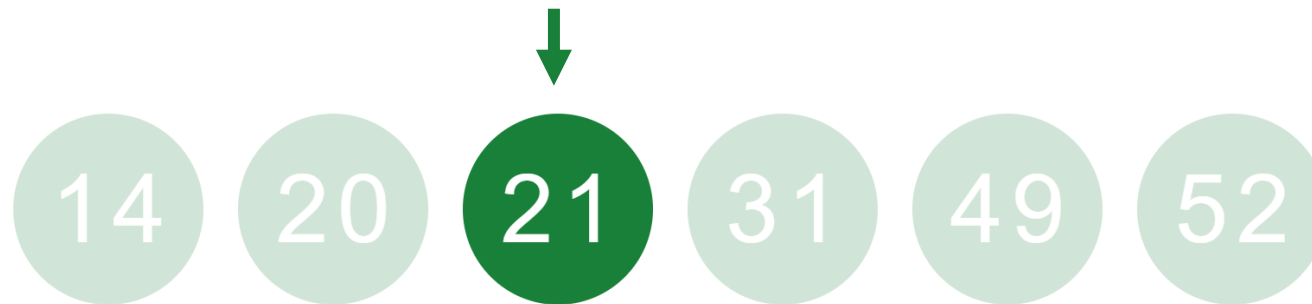
Busca Binária

- Busca de um elemento:
 - Lista ordenada. Queremos recuperar o elemento 37.



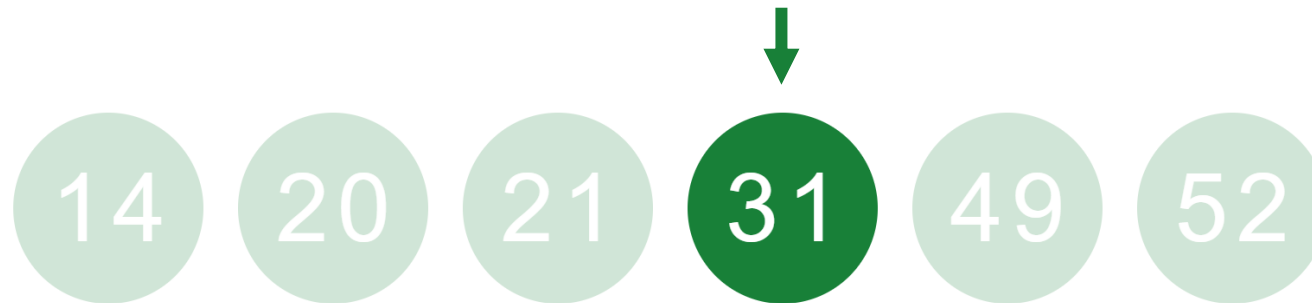
Busca Binária

- **Busca de um elemento:**
 - **Lista ordenada.** Queremos recuperar o elemento 37.



Busca Binária

- **Busca de um elemento:**
 - **Lista ordenada.** Queremos recuperar o elemento 37.



Busca Binária

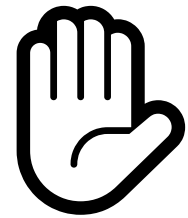
- Busca de um elemento:
 - Lista ordenada. Queremos recuperar o elemento 37.



Busca Binária

- **Busca de um elemento:**
 - **Lista ordenada.** Queremos recuperar o elemento 37.

5 passos!



Lembre-se, a lista está ordenada.

$37 < 49$



Busca Binária

- **Busca de um elemento:**
 - **Lista ordenada.** Queremos recuperar o elemento 53.



Ainda percorreríamos a lista inteira.
Seria eficiente? Podemos melhorar?



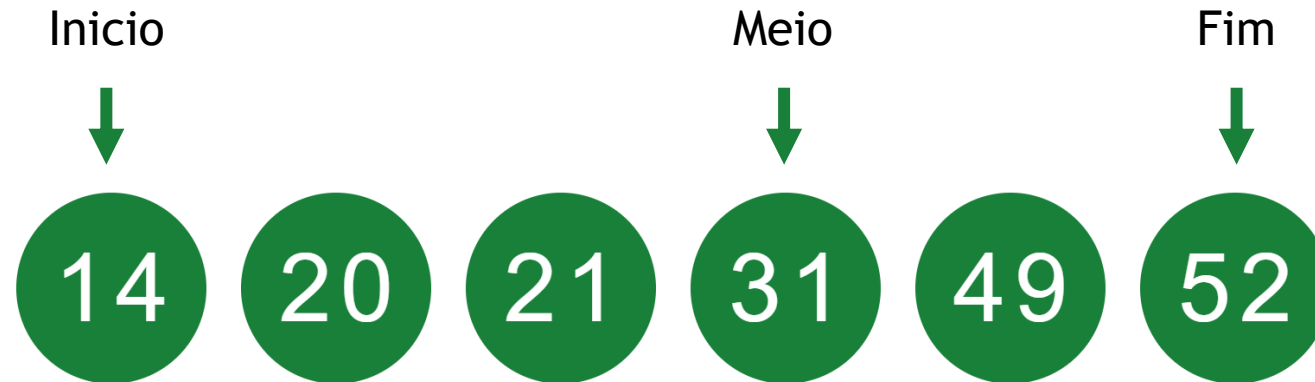
Busca Binária

- Busca de um elemento:
 - Busca Binária. Queremos recuperar o elemento 53.



Busca Binária

- Busca de um elemento:
 - Busca Binária. Queremos recuperar o elemento 53.



Busca Binária

- Busca de um elemento:
 - Busca Binária. Queremos recuperar o elemento 53.

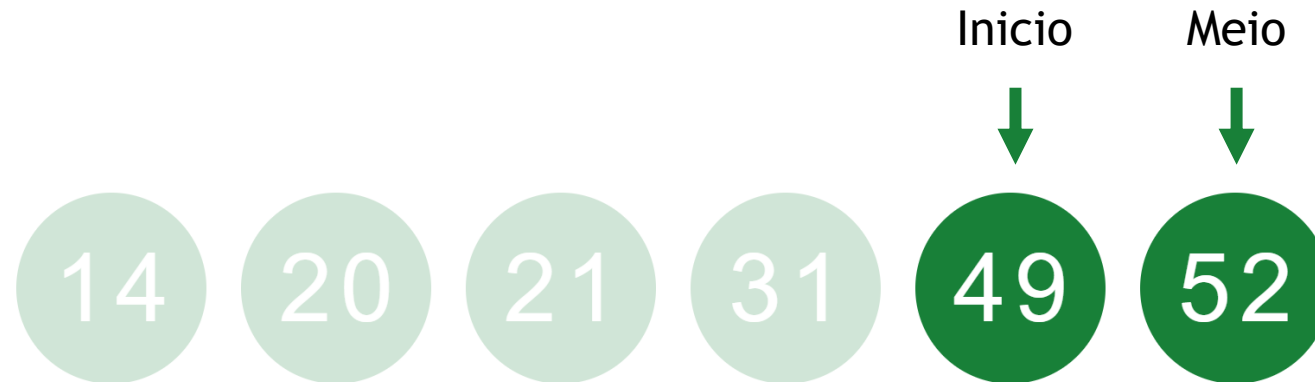


Se elemento buscado > valor meio então descarta esq



Busca Binária

- **Busca de um elemento:**
 - **Busca Binária.** Queremos recuperar o elemento 53.



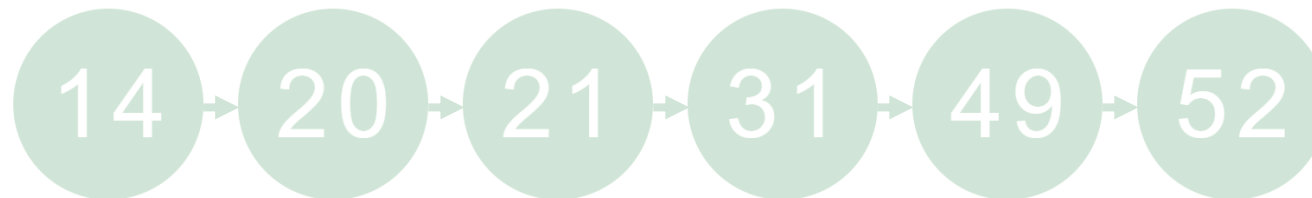
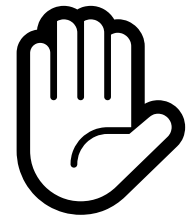
Se **elemento buscado** > **valor meio** então **descarta esq**



Busca Binária

- **Busca de um elemento:**
 - **Busca Binária.** Queremos recuperar o elemento 53.

2 passos!

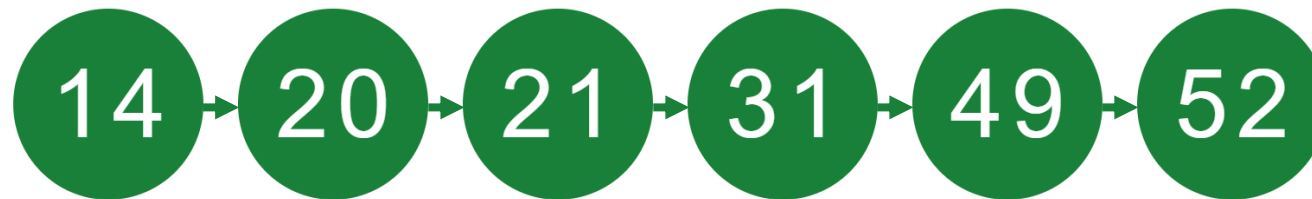


Bem mais eficiente!!!



Busca Binária

- Busca de um elemento:
 - Busca Binária com Lista Encadeada.



Seria possível?



Listas Lineares, Pilhas, Filas e Busca Binária

Até a próxima!