UFMS DIGITAL

Algoritmos e Programação II

Prof. MSc. Samuel Benjoino Ferraz Aquino

UFMS DIGITAL

Módulo 3 – Algoritmos recursivos

Prof. MSc. Samuel Benjoino Ferraz Aquino

Roteiro

- Unidade 1: Recursão e Algoritmos Recursivos
- Unidade 2: Algoritmos de Busca
- Resumo

UFMS DIGITAL

Unidade I

Recursividade e Algoritmos Recursivos

Introdução

- Programas desenvolvidos até o momento
  - Variáveis
  - Estruturas condicionais
  - Estruturas de repetição
  - **Funções**

Introdução

- Exemplo de criação e uso de funções

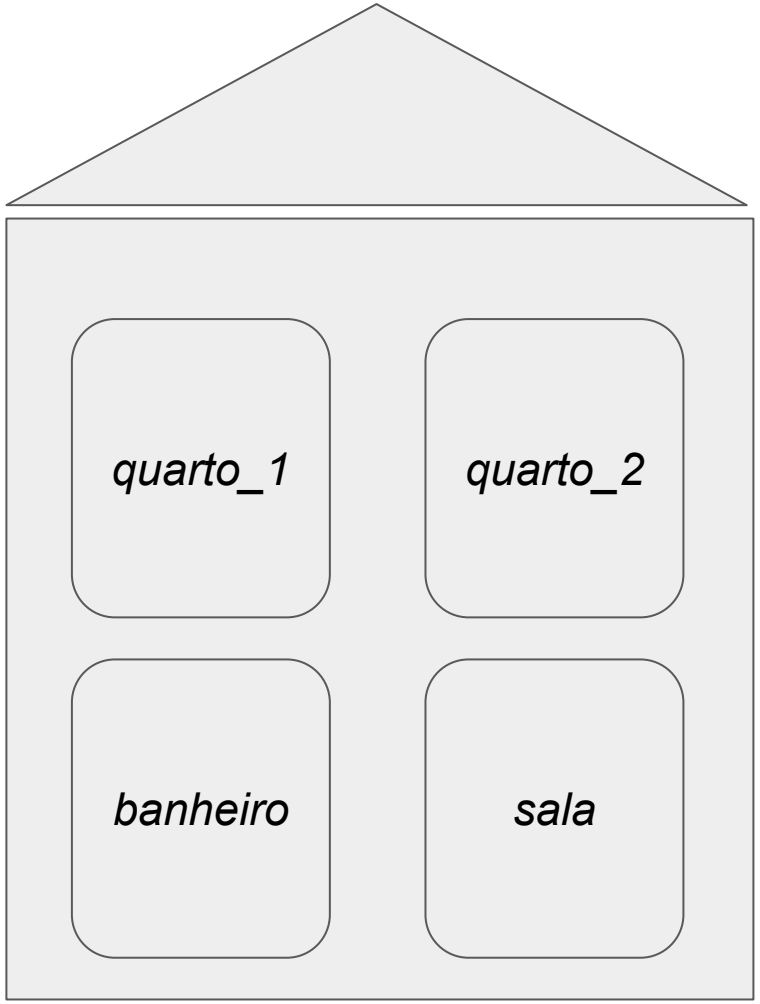
```
def funcao1(valor):
    valor = valor + 1
    valor = valor * 2
    funcao2(valor)

def funcao2(valor):
    valor = valor / 3
```

Introdução

- Em alguns problemas, é útil que uma função chame a si mesma para resolver um problema!

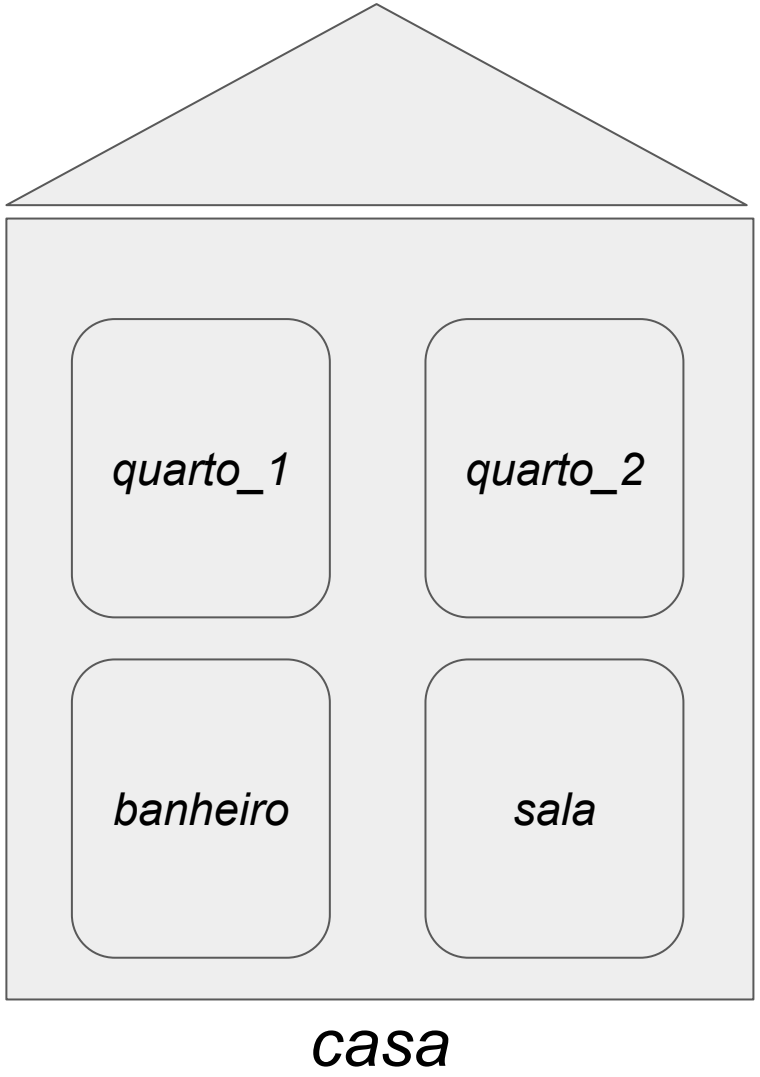
- Parece estranho, mas fazemos isso na vida real!

Introdução



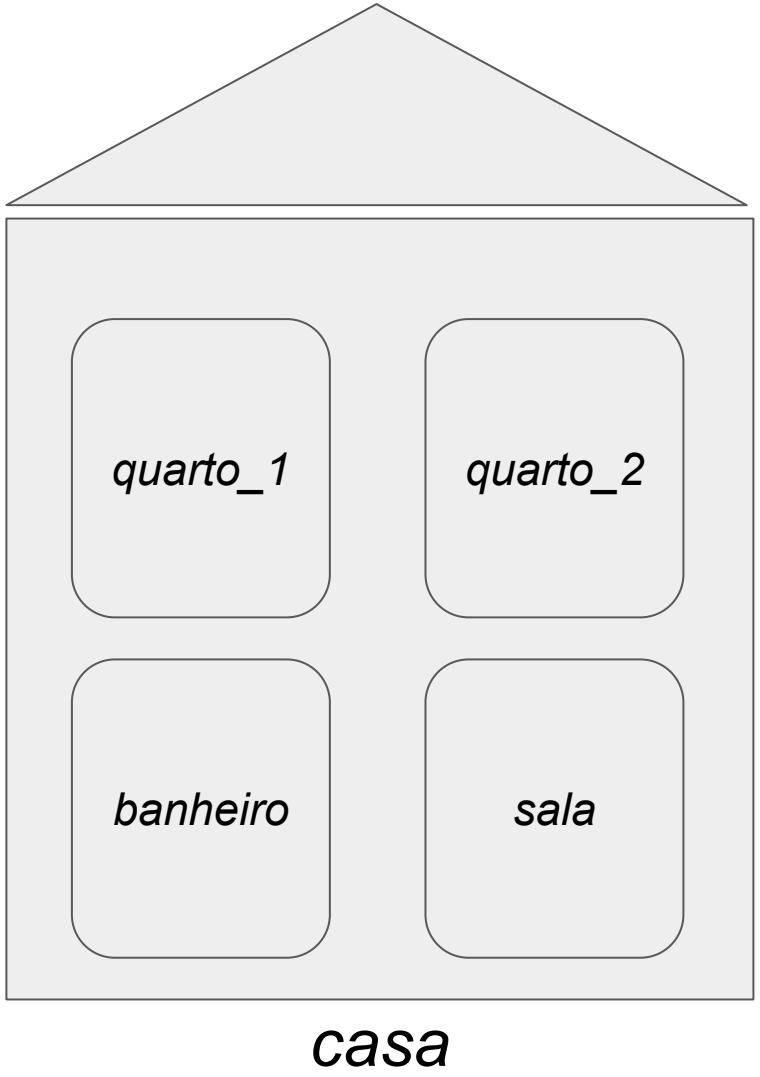
casa

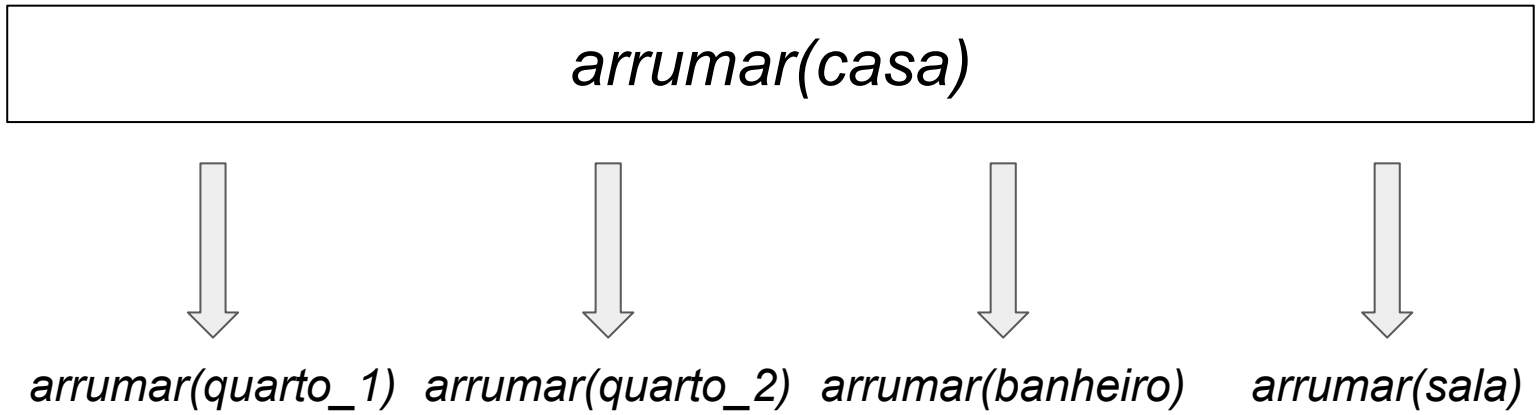
Introdução



arrumar(casa)

Introdução


quarto_1
quarto_2
banheiro
sala
casa


arrumar(casa)
arrumar(quarto_1)
arrumar(quarto_2)
arrumar(banheiro)
arrumar(sala)

Recursividade

- *Capacidade de uma função resolver um problema usando chamadas da própria função*

- Se encaixa bem para problemas com determinadas características

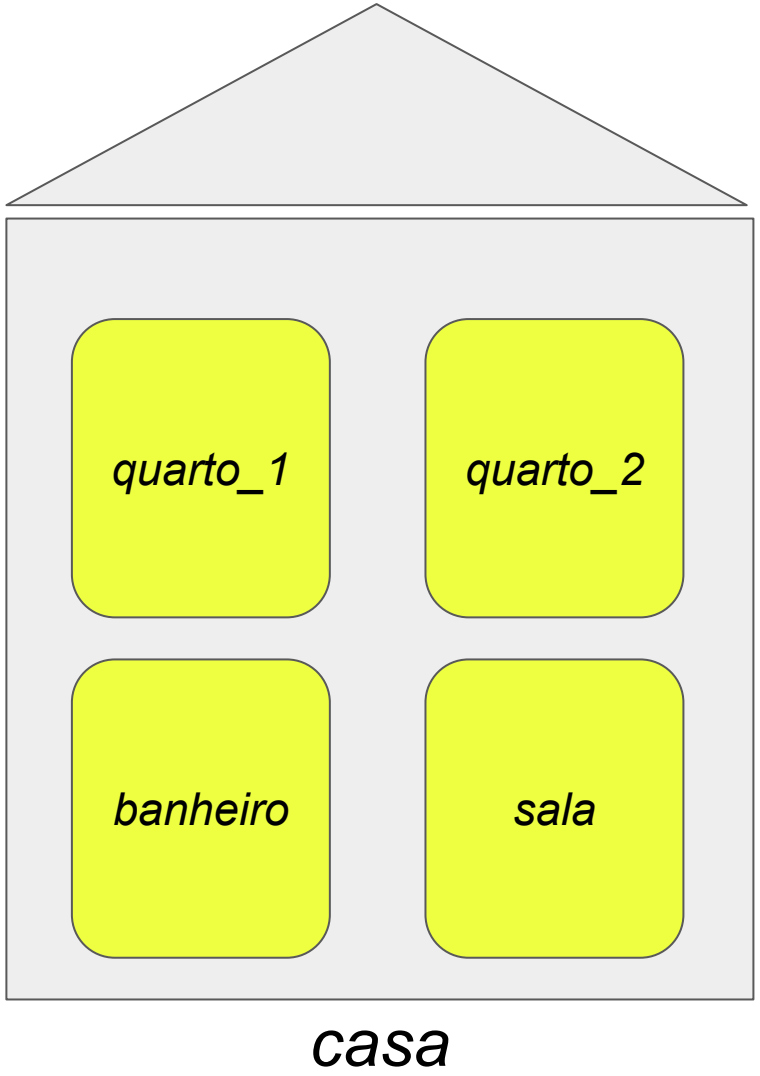
- Simula um laço

Algoritmos Recursivos

- Passos para o desenvolvimento de uma função recursiva:

    1 - **Dividir** um problema em problemas menores
    2 - **Resolver** os problemas menores com a própria função
    3 - **Parar** quando a solução for pequena o suficiente

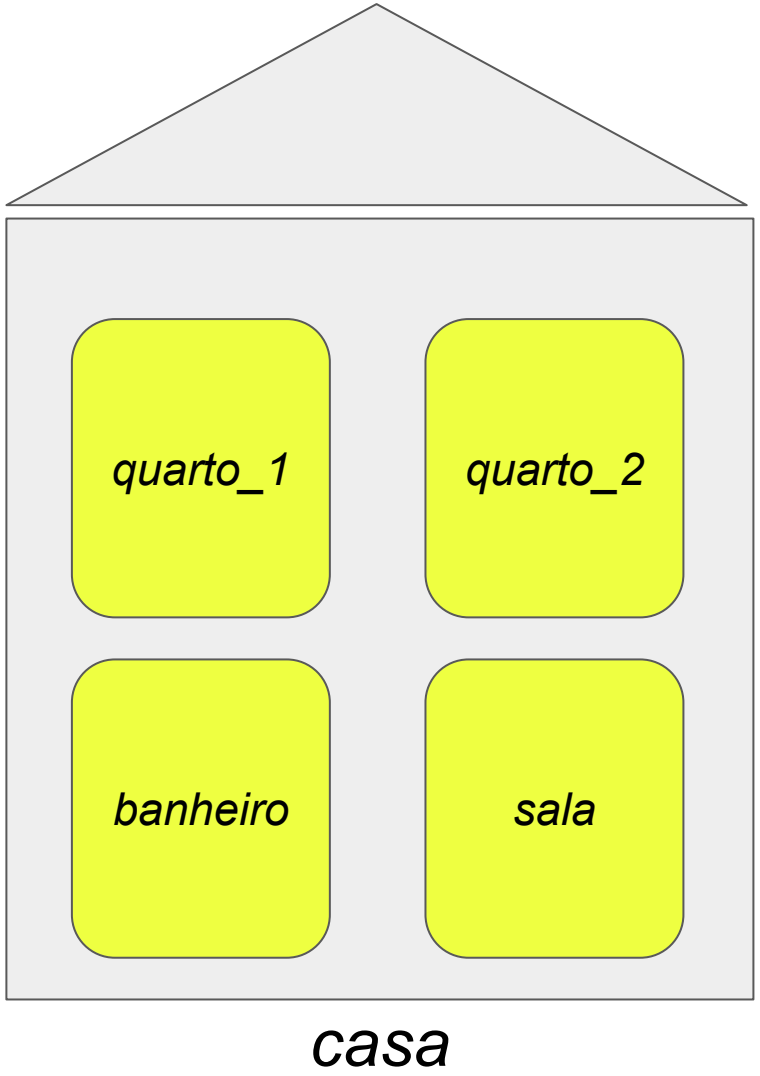
Algoritmos Recursivos



- Passos para o desenvolvimento de uma função recursiva:

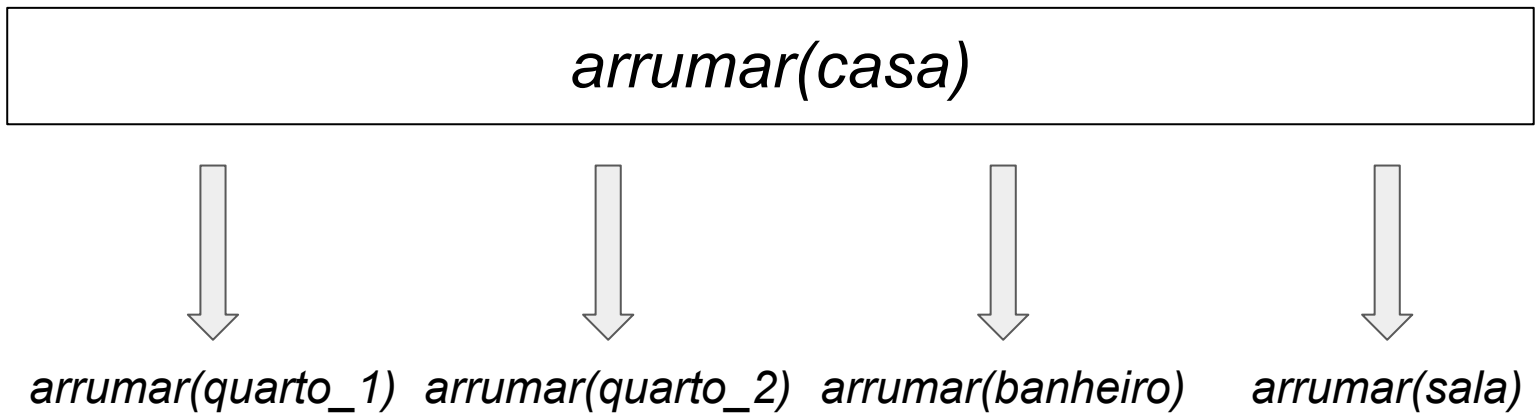
    1 - **Dividir** um problema em problemas menores

Algoritmos Recursivos

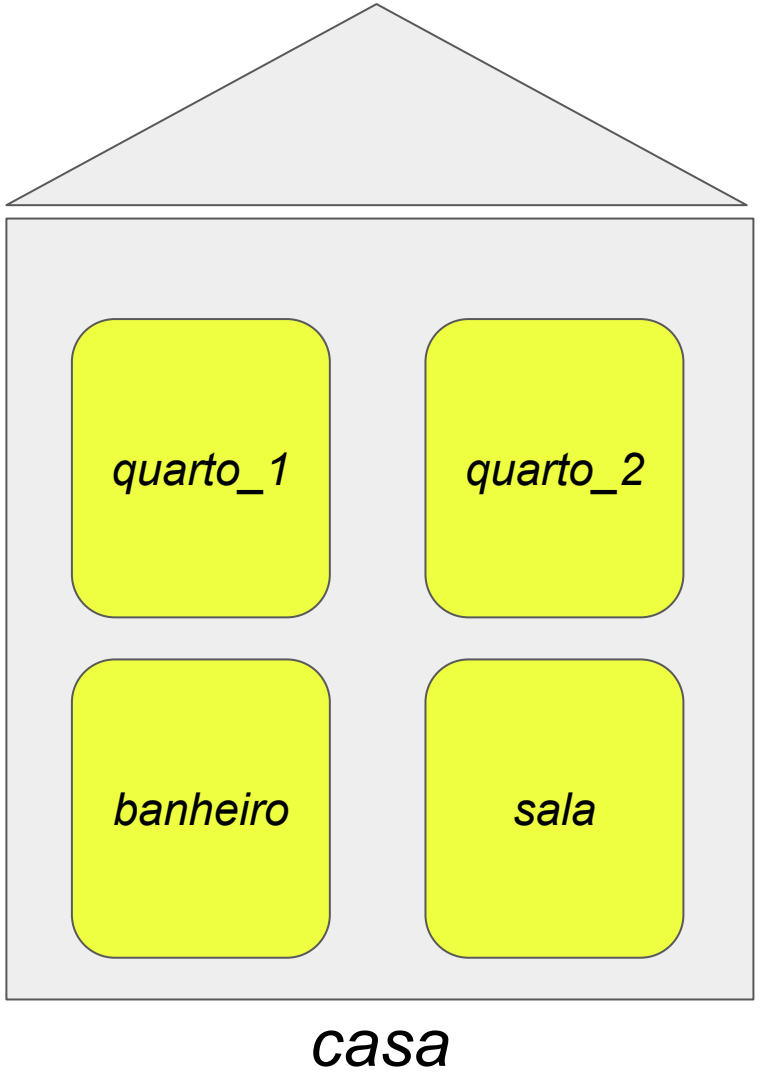


- Passos para o desenvolvimento de uma função recursiva:

  2 - **Resolver** os problemas menores com a própria função



Algoritmos Recursivos

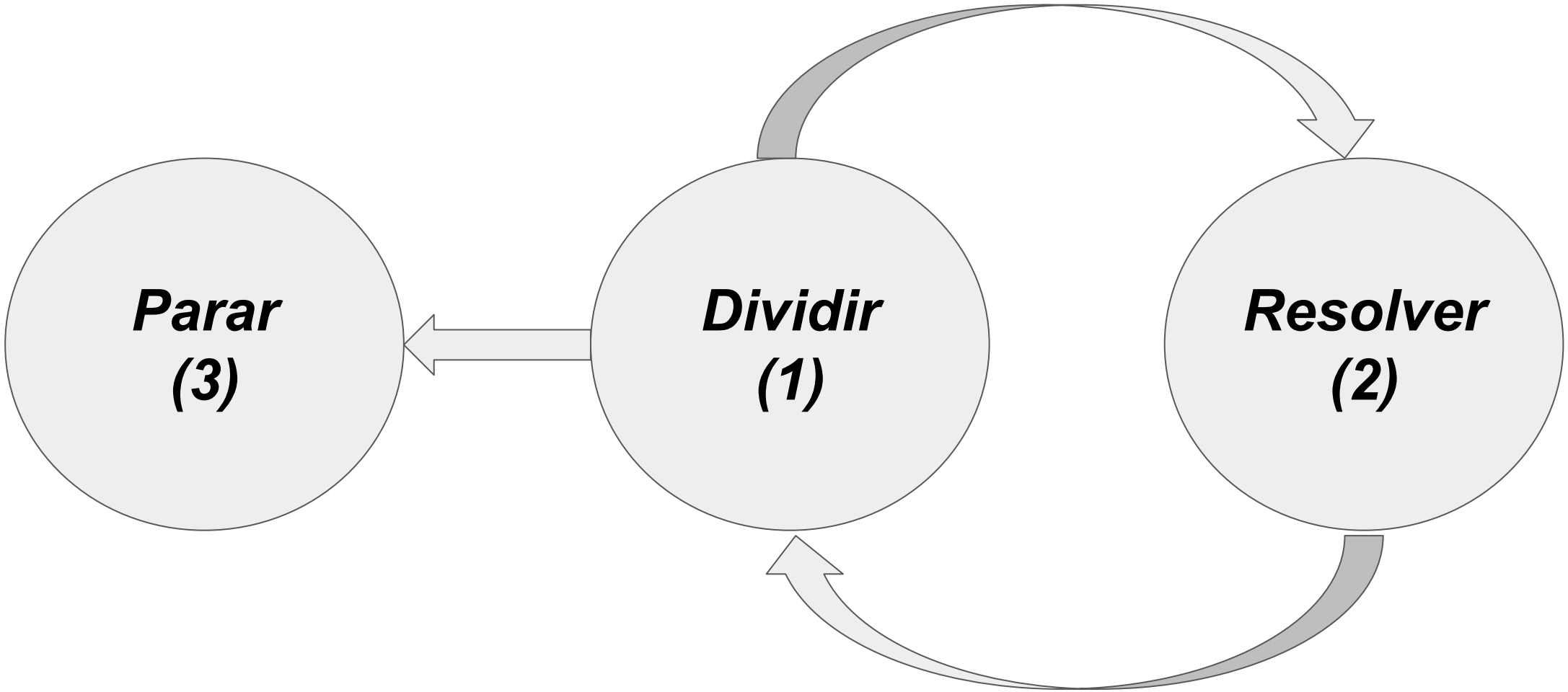


- Passos para o desenvolvimento de uma função recursiva:

  3 - **Parar** quando a solução for pequena o suficiente

  Ex: vou **parar** a arrumação do quarto 1 quando a cama estiver pronta.

Algoritmos Recursivos



Algoritmos Recursivos

- Vantagem
  - Implementação “direta” para alguns problemas
- Desvantagem
  - Maior consumo de memória
- Recursão é indicada quando solução iterativa simples não é possível

UFMS DIGITAL

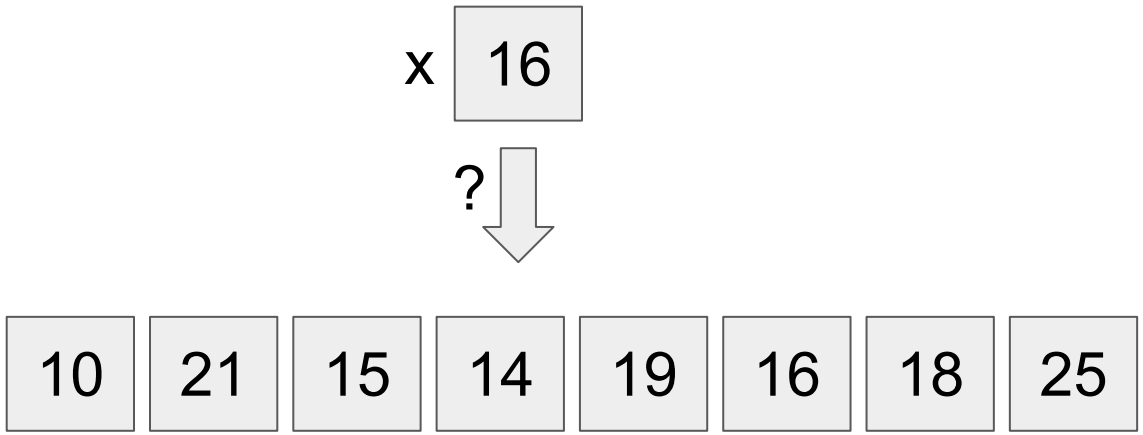
Unidade II

Algoritmos de Busca

Algoritmos de Busca

- A busca de informação é uma das aplicações mais importantes dos computadores

- Problema básico de busca: encontrar um valor **x** em um vetor



Algoritmos de Busca

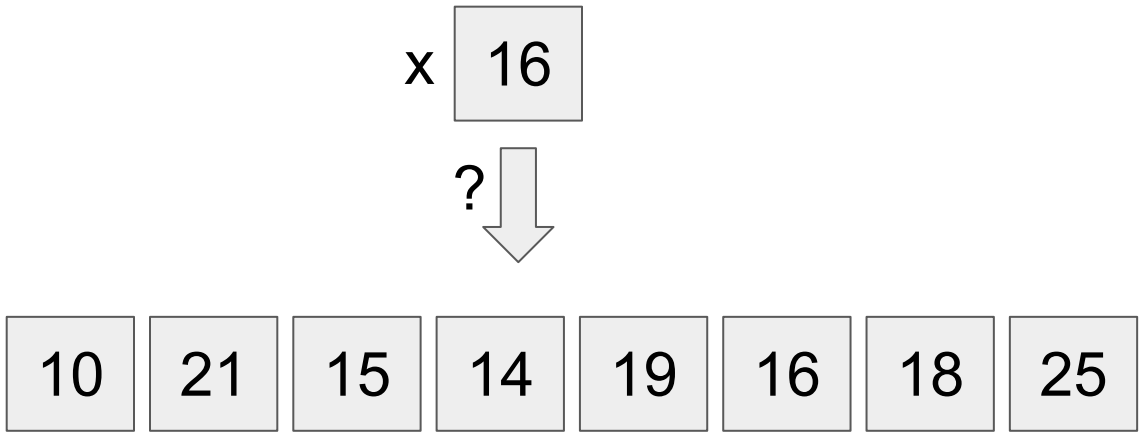
- Dois principais tipos de busca
  - Busca sequencial/linear
  - Busca binária

Algoritmos de Busca

- *Busca sequencial*
  - Estratégia mais básica
  - Procura o elemento em todas as posições

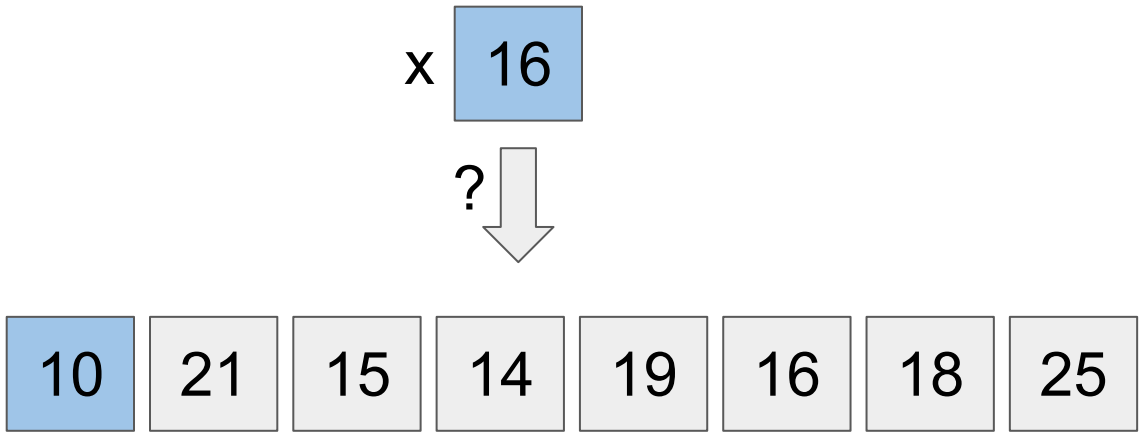
Algoritmos de Busca

- *Busca sequencial*
  - Estratégia mais básica
  - Procura o elemento em todas as posições



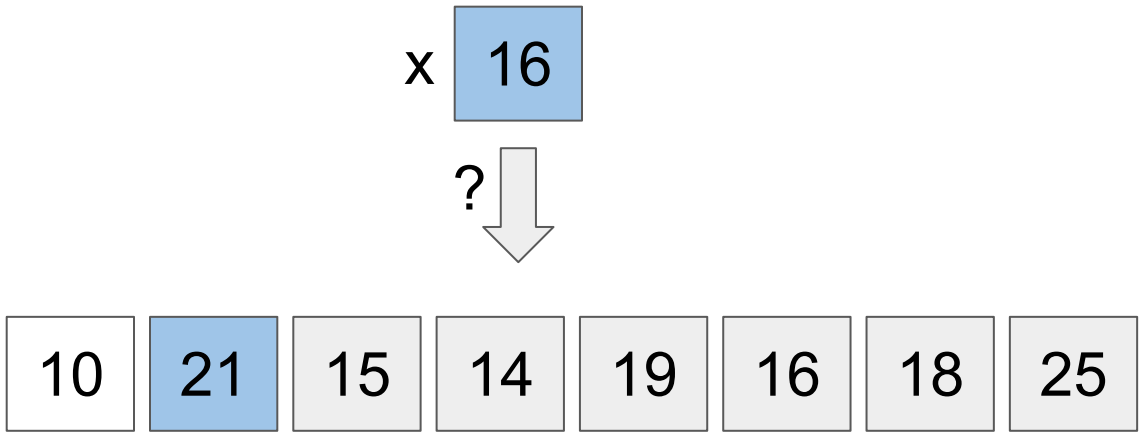
Algoritmos de Busca

- *Busca sequencial*
  - Estratégia mais básica
  - Procura o elemento em todas as posições



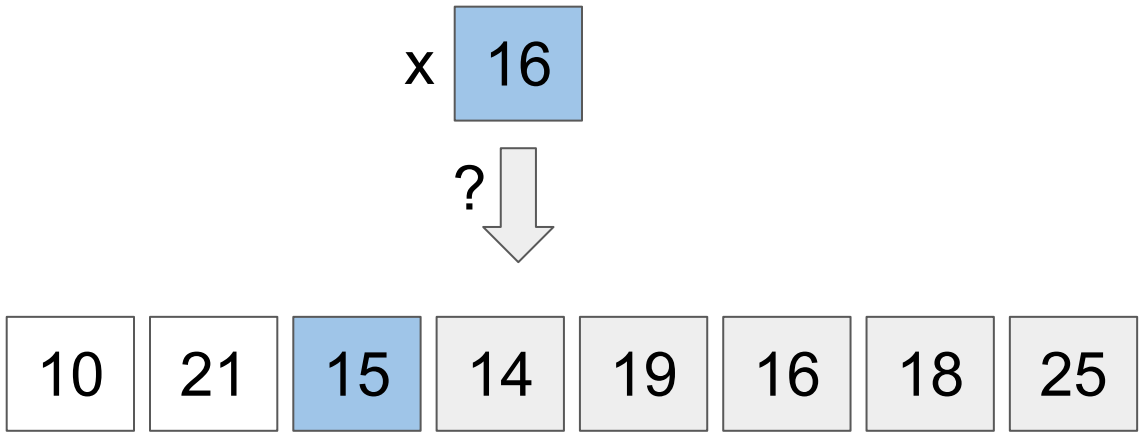
Algoritmos de Busca

- *Busca sequencial*
  - Estratégia mais básica
  - Procura o elemento em todas as posições



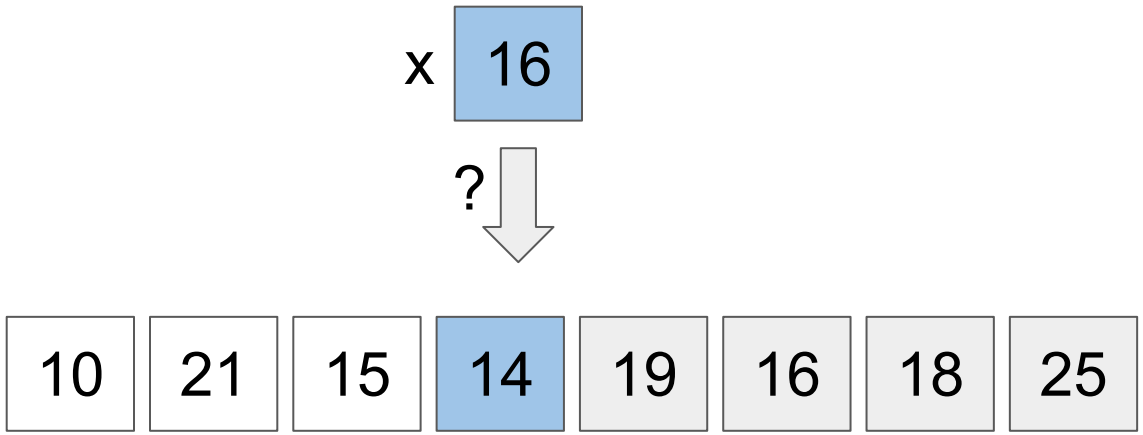
Algoritmos de Busca

- *Busca sequencial*
  - Estratégia mais básica
  - Procura o elemento em todas as posições



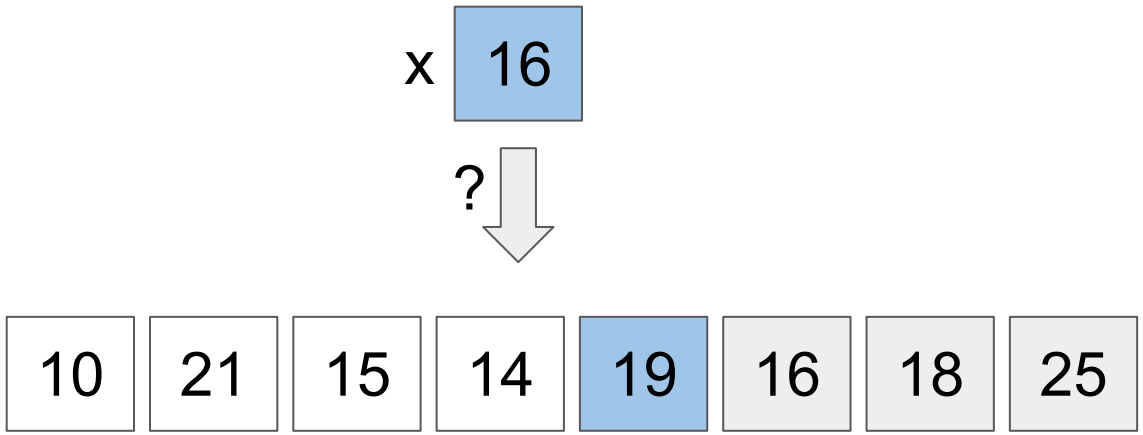
Algoritmos de Busca

- *Busca sequencial*
  - Estratégia mais básica
  - Procura o elemento em todas as posições



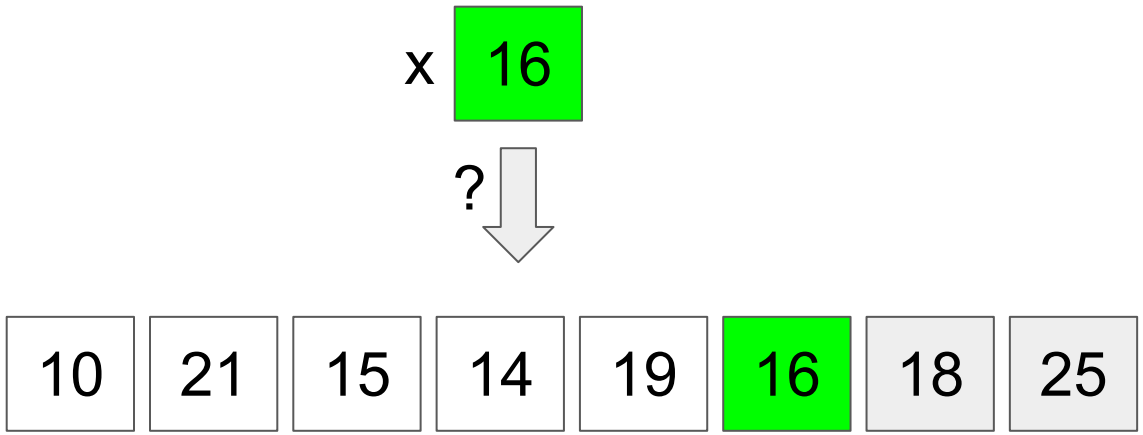
Algoritmos de Busca

- *Busca sequencial*
  - Estratégia mais básica
  - Procura o elemento em todas as posições



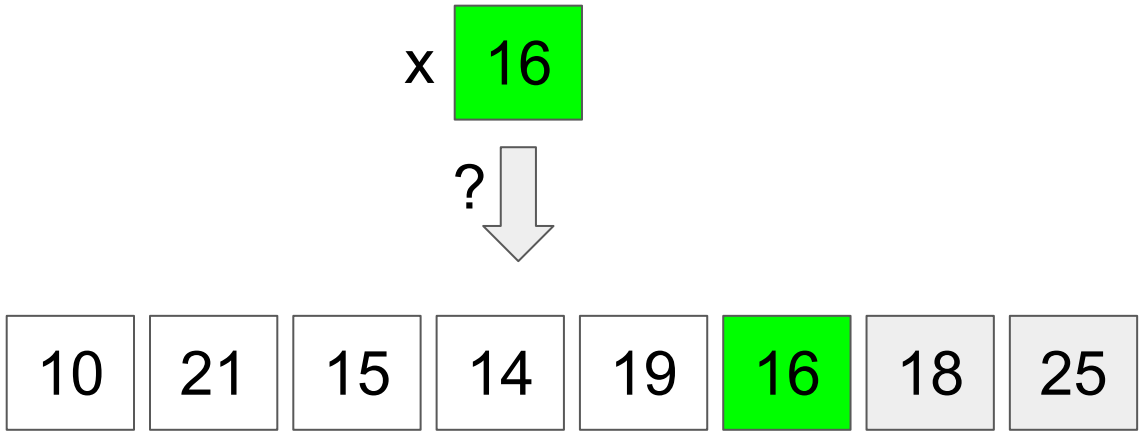
Algoritmos de Busca

- *Busca sequencial*
  - Estratégia mais básica
  - Procura o elemento em todas as posições



Algoritmos de Busca

- *Busca sequencial*
  - Possui variações de implementação
  - Menos eficiente que a *busca binária*

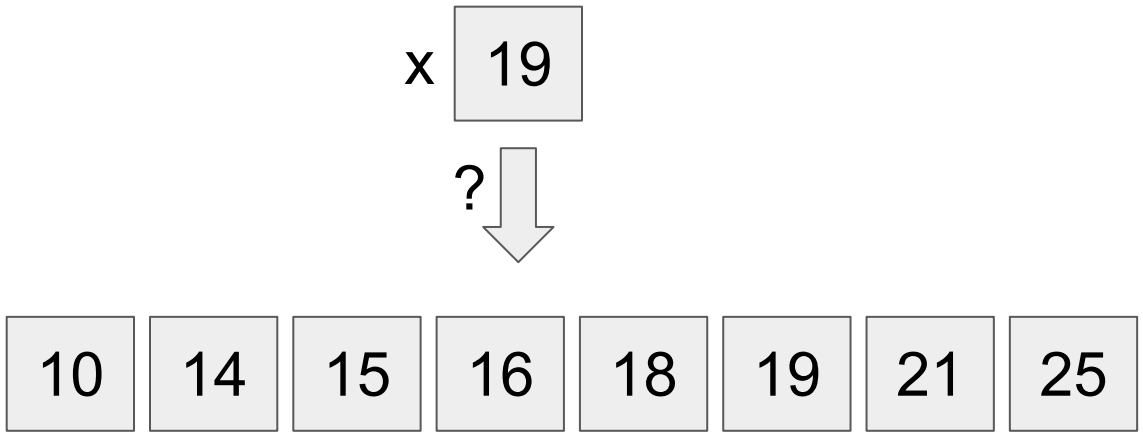


Algoritmos de Busca

- *Busca binária*
  - Assume uma ordenação prévia dos elementos
  - Mais eficiente que busca linear

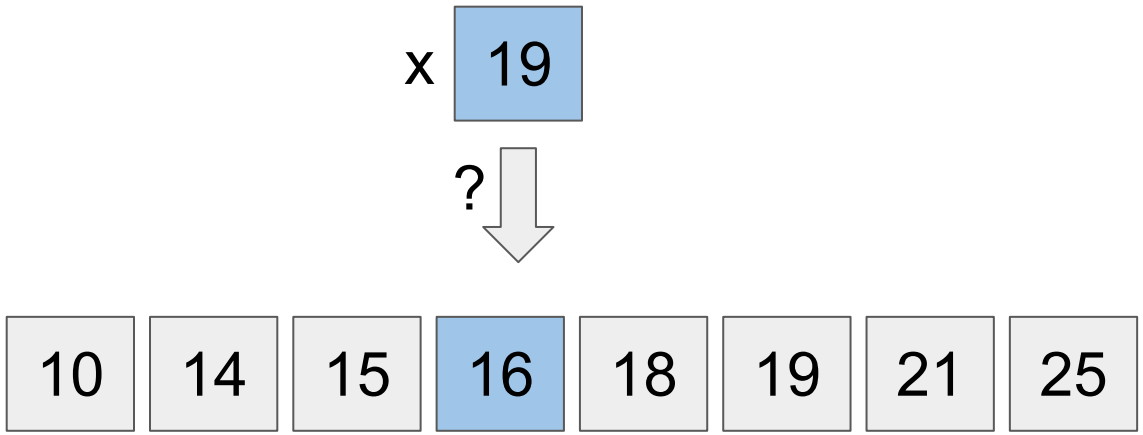
Algoritmos de Busca

- *Busca binária*
  - Assume uma ordenação prévia dos elementos
  - Mais eficiente que busca linear



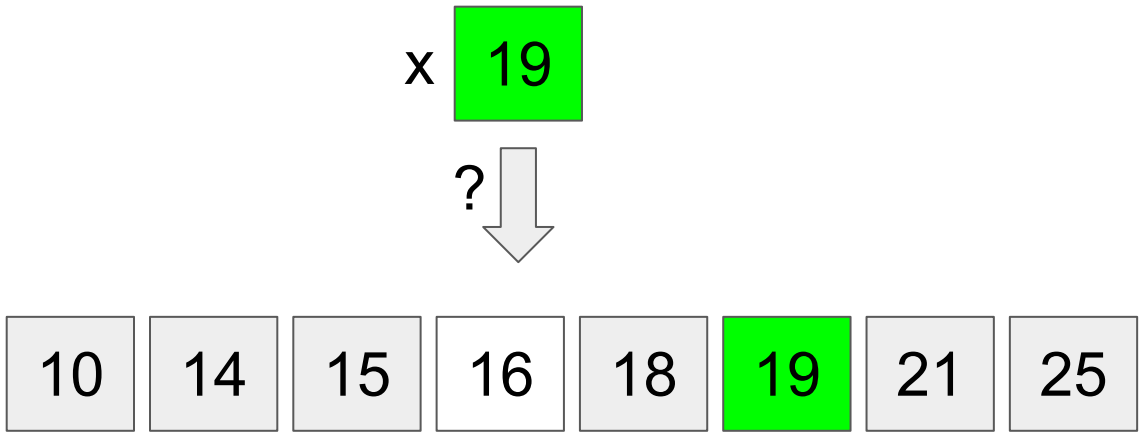
Algoritmos de Busca

- *Busca binária*
  - Assume uma ordenação prévia dos elementos
  - Mais eficiente que busca linear



Algoritmos de Busca

- *Busca binária*
  - Assume uma ordenação prévia dos elementos
  - Mais eficiente que busca linear
    - Apenas 2 comparações nesse caso!



Algoritmos de Busca

- *Busca binária*
  - Vantagens:
    - Custo muito inferior à busca linear
    - Apropriada para conjuntos grandes
  - Desvantagens:
    - Precisa manter elementos ordenados
    - Recomendável para conjuntos com poucas alterações

Roteiro

- Unidade 1: Recursão e Algoritmos Recursivos
- Unidade 2: Algoritmos de Busca
- Resumo