

**INSTITUTO FEDERAL DE MATO GROSSO  
CAMPUS RONDONÓPOLIS**  
**Curso Superior de Análise e Desenvolvimento de Sistemas**

# **Estrutura de Dados e Análise de Algoritmos**

Prof. Daniel Domingos Alves

[daniel.alves@ifmt.edu.br](mailto:daniel.alves@ifmt.edu.br)

**Métodos de Pesquisa**  
**24/10/2024**

# Métodos de pesquisa



# Pesquisa em Memória Primária

- Existe uma variedade enorme de métodos de pesquisa.
- A escolha do método de pesquisa mais adequado a uma determinada aplicação depende principalmente:
  - (i) da quantidade dos dados envolvidos
  - (ii) de o arquivo estar sujeito a inserções e retiradas frequentes, ou de o conteúdo do arquivo ser praticamente estável

# Pesquisa

- O problema de procurar (pesquisar) alguma informação numa tabela ou num catálogo é muito comum
- Exemplo:
  - procurar o telefone de uma pessoa no catálogo
  - procurar o nº da conta de um certo cliente
  - consultar um determinado saldo em um terminal de autoatendimento

# Pesquisa

- A tarefa de “pesquisa”, “procura” ou “busca” é, como se pode imaginar, *uma função muito utilizada*
- As rotinas que executam a busca devem ser eficientes (menor tempo possível)

# Pesquisa

## EFICIÊNCIA

- O **TEMPO GASTO** pesquisando dados em tabelas depende do **TAMANHO** da tabela e do **ALGORITMO** utilizado na busca.

# Algoritmos de Pesquisa

- Pesquisa Sequencial
- Pesquisa Sequencial com Sentinela
- Pesquisa Binária

# Pesquisa Sequencial



# Pesquisa Sequencial

- O método de pesquisa mais simples que existe funciona da seguinte forma:
  - A partir do primeiro registro, pesquise sequencialmente até encontrar a chave procurada; então pare.
- Apesar da simplicidade, a pesquisa sequencial envolve algumas ideias interessantes, servindo para ilustrar vários aspectos e convenções a serem utilizadas em outros métodos de pesquisa.

# Pesquisa Sequencial

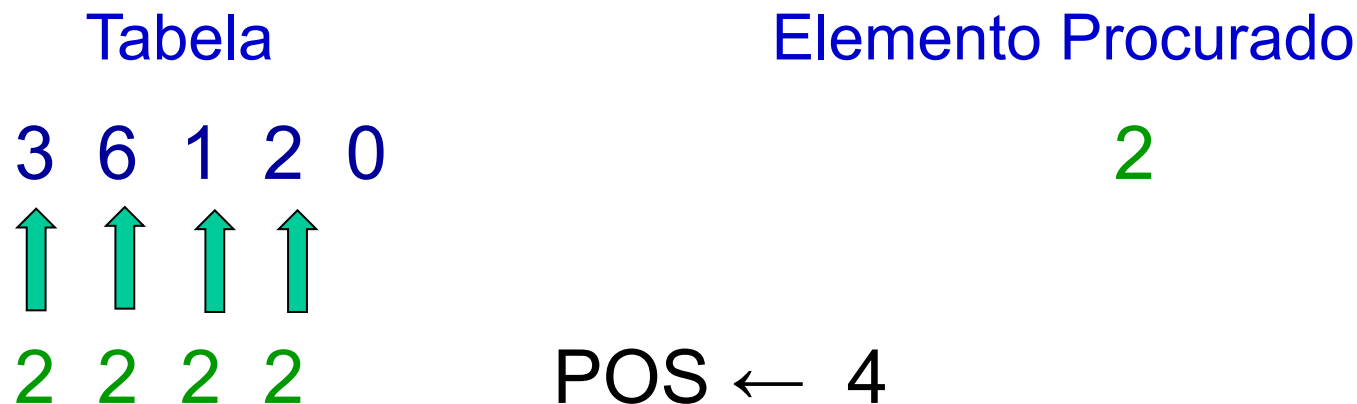
- A ideia básica da Pesquisa Sequencial é localizar o elemento procurado por meio de comparações sucessivas e sequenciais, a partir do primeiro elemento do vetor.
- A pesquisa termina quando o elemento é encontrado ou quando é atingido o fim do vetor.

# Algoritmo de Pesquisa

- Para os algoritmos de pesquisa que se seguem vamos denotar por:
  - TAB: um vetor contendo N elementos inteiros distintos
  - DADO: elemento a ser procurado em TAB
  - ACHOU: indica o sucesso ou falha na pesquisa
  - POS: aponta para a posição do elemento encontrado

# Pesquisa Sequencial

- Comparar o elemento procurado (DADO) com cada um dos elementos da tabela TAB na sequência em que aparecem na tabela



# Pesquisa Sequencial

programa BUSCA1

declarar

  I {variável de controle}

  N {tamanho da tabela},

  DADO {elemento a ser procurado na tabela},

  POS {posição em que se encontra o elemento}

  :inteiros

  ACHOU {valor lógico que representa o sucesso da busca}:lógica

  TAB {tabela a ser consultada} vetor

início

  solicitar a entrada do tamanho da tabela, ler (N)

  solicitar a entrada dos dados da tabela

  para I de 1 até N faça

    ler ( TAB[I] )

  fim para

  solicitar a entrada do elemento a ser procurado, ler(DADO)

## Pesquisa Sequencial

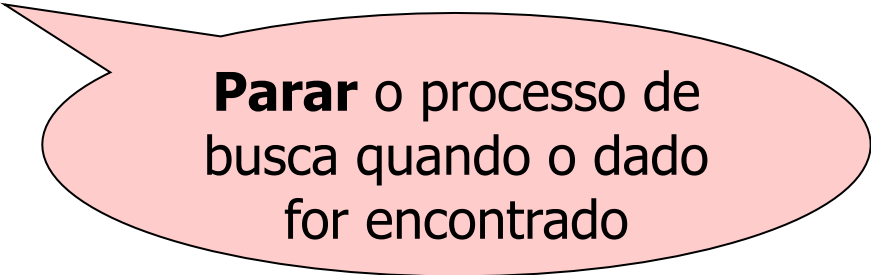
```
ACHOU ← falso
para I de 1 até N faça
    se TAB[I] = DADO
        então início
            ACHOU ← verdade
            POS ← I
        fim
    fim-se
fim para
se ACHOU
    então escrever (DADO, ' se encontra na posicao ', POS)
    senão escrever (DADO, ' nao se encontra na tabela')
fim se
fim programa
```

**INEFICIENTE:** O processo de busca continua mesmo depois que o elemento for encontrado

## Pesquisa Sequencial

```
ACHOU ← falso
para I de 1 até N faça
    se TAB[I] = DADO
        então início
            ACHOU ← verdade
            POS ← I
        fim
    fim-se
fim para
se ACHOU
    então escrever (DADO, ' se encontra na posicao ', POS)
    senão escrever (DADO, ' nao se encontra na tabela')
fim se
fim programa
```

**INEFICIENTE:** O processo de busca continua mesmo depois que o elemento for encontrado



**Parar** o processo de busca quando o dado for encontrado

## Pesquisa Sequencial

```
ACHOU ← falso  
I ← 1  
enquanto (ACHOU = falso) e (I ≤ N) faça  
    se DADO = TAB[I]  
        então  
            ACHOU ← verdade  
            POS ← I  
        fim  
    senão I ← I + 1  
fim se  
fim enquanto
```

```
se ACHOU  
    então escrever (DADO, ' se encontra na posicao ', POS)  
    senão escrever (DADO, ' nao se encontra na tabela')  
fim se  
fim programa
```

Pesquisa Sequencial com uso da variável lógica



## Pesquisa Sequencial

```
I ← 1  
POS ← 0  
enquanto (I ≤ N) e (POS = 0) faça  
    se DADO = TAB[I]  
        então POS ← I  
    fim-se  
    I ← I + 1  
fim enquanto
```

```
se POS ≠ 0  
    então escrever (DADO, ' se encontra na posicao ', POS)  
    senão escrever (O elemento nao pertence ao conjunto)  
fim-se  
fim programa
```

Pesquisa Sequencial sem uso da variável lógica

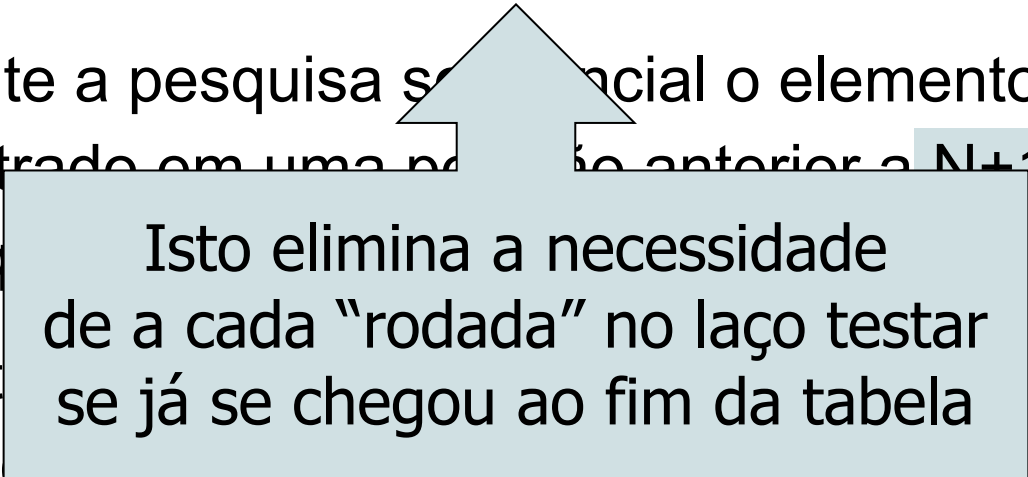
# Pesquisa Sequencial com Sentinela

# Pesquisa Sequencial com Sentinela

- A ideia básica da Pesquisa Sequencial com Sentinela é usar o elemento procurado como um aviso que a tabela acabou → inserir o elemento procurado no final da tabela.
- Se durante a pesquisa sequencial o elemento procurado for encontrado em uma posição anterior a  $N+1$ , isto significa que o elemento pertence a tabela.
- No entanto, se o elemento procurado só for encontrado na posição  $N+1$ , isto significa que ele não está na tabela.

# Pesquisa Sequencial com Sentinela

- A ideia básica da Pesquisa Sequencial com Sentinela é usar o elemento procurado como um aviso que a tabela acabou → inserir o elemento procurado no final da tabela.
- Se durante a pesquisa sequencial o elemento procurado for encontrado em uma posição anterior a  $N+1$ , isto significa que o elemento está na tabela.
- No entanto, se o elemento procurado não for encontrado na posição  $N+1$ , isto significa que ele não está na tabela.



Isto elimina a necessidade de a cada "rodada" no laço testar se já se chegou ao fim da tabela

# Pesquisa Sequencial com Sentinela

Tabela

3 6 1 2 0

3 6 1 2 0 9

Elemento Procurado

9



**POS = N+1**

Significa que o elemento não  
está na tabela

# Pesquisa Sequencial com Sentinela

Tabela

3 6 1 2 0

3 6 1 2 0 6

Elemento Procurado

6



**POS = 2**

Significa que o elemento está  
na tabela

# Pesquisa Sequencial com Sentinela

programa BUSCA4

declarar

I {variável de controle}  
N {tamanho da tabela},  
DADO {elemento a ser procurado na tabela},  
POS {posição em que se encontra o elemento}  
:inteiros  
ACHOU {valor lógico que representa o suceso da busca}:lógica  
TAB {tabela a ser consultada} vetor

início

solicitar a entrada do tamanho da tabela, ler (N)  
solicitar a entrada dos dados da tabela  
para I de 1 até N faça  
    ler ( TAB[I] )  
fim para  
solicitar a entrada do elemento a ser procurado, ler(DADO)

## Pesquisa Sequencial com Sentinela

```
TAB[N+1] ← DADO
POS ← 1
enquanto TAB[POS] ≠ DADO faça
    POS ← POS + 1
fim enquanto
se POS ≤ N
    então ACHOU ← verdade
    senão ACHOU ← falso
fim se
```

```
se ACHOU
    então escrever (DADO, ' se encontra na posicao ', POS)
    senão escrever (DADO, ' nao se encontra na tabela')
fim se
fim programa
```

Se sabemos de antemão que o dado se encontra em algum lugar na tabela, só é necessário testar uma vez, ao final, para saber a posição em que ele foi encontrado



# Análise do algoritmo de pesquisa sequencial

- Ao analisar um algoritmo de pesquisa, contamos o número de comparações que foram realizadas, por quê?
  - Porque este número nos dá as informações mais úteis
  - Este critério para a contagem do número de comparações pode ser aplicado igualmente a outros algoritmos de busca

# Quantas Comparações?

- Pesquisa sequencial de um vetor que {a) encontra seu alvo, (b) não encontrar seu alvo

(a) Uma busca por **8**

Comparar com 9

<b>9</b>	5	8	4	7
----------	---	---	---	---

$8 \neq 9$ , assim continue a busca... Comparar com 5

9	<b>5</b>	8	4	7
---	----------	---	---	---

(b) Uma busca por **6**

Comparar com 9

<b>9</b>	5	8	4	7
----------	---	---	---	---

$6 \neq 9$ , assim, continue a busca... Comparar com 5

9	<b>5</b>	8	4	7
---	----------	---	---	---

# Análise do algoritmo de pesquisa sequencial

- Suponha que o item que procura está na lista
  - Então o número de comparações depende de onde na lista o item de pesquisa está localizado.
  - Se o item de busca é o primeiro elemento da lista, então é feita apenas **uma** comparação (melhor caso).
  - Agora se o item da lista é o último, então o algoritmo faz **n** comparações (pior caso)

# Análise do algoritmo de pesquisa sequencial

- Melhor caso
  - o item a ser procurado está na primeira posição  $O(1)$
- Pior caso
  - o item a ser procurado não está na lista, ou está na última posição. Assim, varre o vetor inteiro  $n$  passos  $O(n)$
- Caso médio
  - procura em metade dos elementos do vetor  $O(n/2)$

# Pesquisa Binária

# Pesquisa Binária

- A ideia básica da Pesquisa Binária consiste em diminuir cada vez mais o intervalo de busca.
- Neste método, a tabela a ser pesquisada deve estar previamente ordenada (classificada).
- Encontra-se, inicialmente, o elemento central da tabela dividindo-a, assim, em duas metades.
- Verifica-se em que metade o elemento procurado se encontra e abandona-se a outra metade

# Pesquisa Binária

- Conforme o resultado da operação efetuada, toma-se como novo intervalo de pesquisa uma das metades do intervalo anterior e o processo de busca é repetido.
- O término do processo se dá quando o elemento desejado é localizado ou quando o intervalo de busca torna-se vazio (significando que o elemento desejado não está presente na tabela).

# Pesquisa Binária

- A cada passo divide-se a área de pesquisa à metade
- Caso, por exemplo, um vetor tenha 1500 elementos

$$1500/2 \rightarrow 750 \quad 24/2 \rightarrow 12$$

$$750/2 \rightarrow 375 \quad 12/2 \rightarrow 6$$

$$376/2 \rightarrow 188 \quad 6/2 \rightarrow 3$$

$$188/2 \rightarrow 94 \quad 3/2 \rightarrow 1,5$$

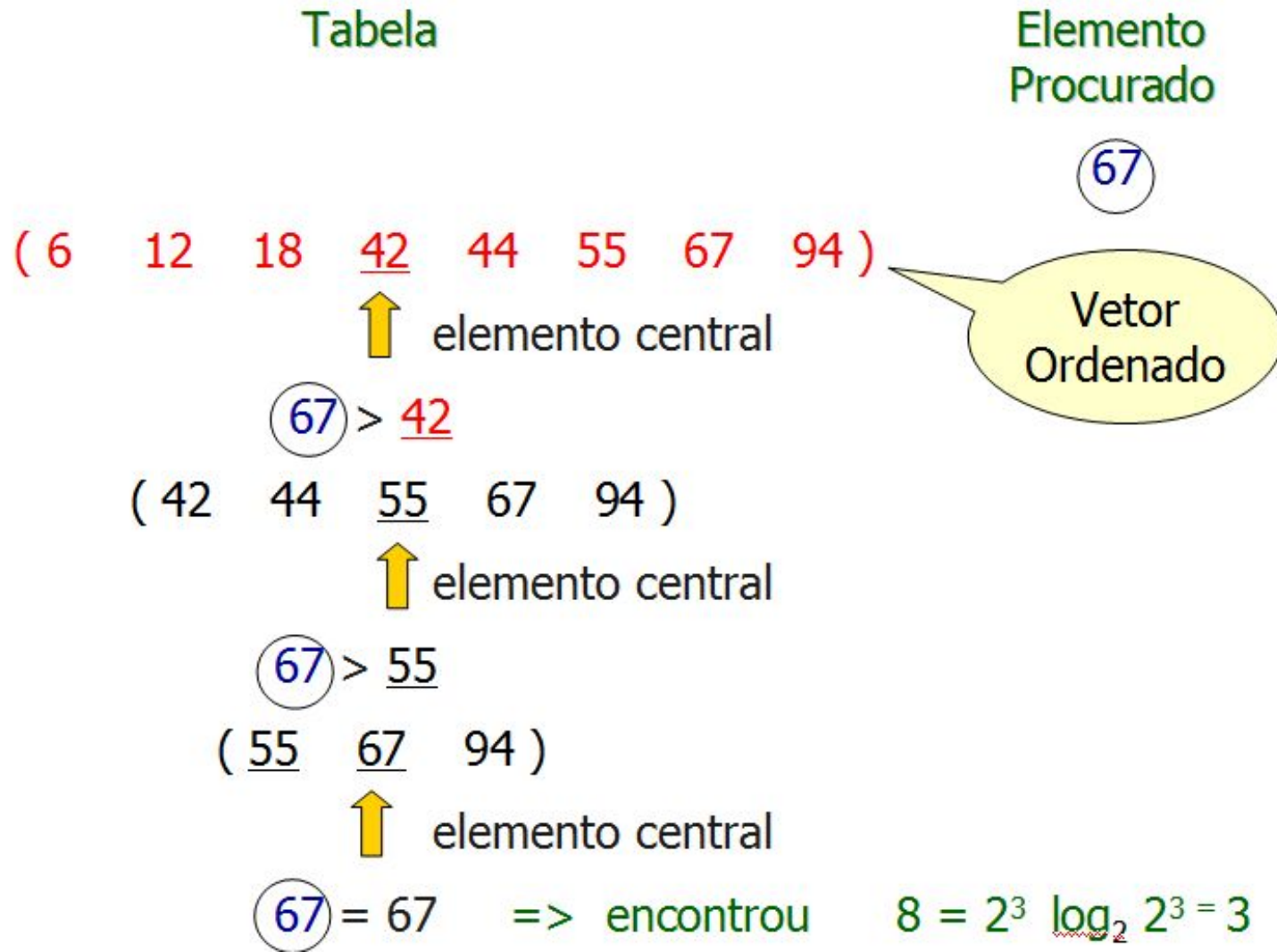
$$94/2 \rightarrow 47 \quad 2/2 \rightarrow 1$$

$$47/2 \rightarrow 23,5$$

O número máximo de passos é  $\log_2 N$ , arredondado ao inteiro mais próximo.



# Pesquisa Binária



# Pesquisa Binária

Programa PesquisaBinaria

{Pesquisa Binária}

declarar

I {variável de controle},

N {tamanho da tabela},

DADO {elemento procurado},

MEIO {posição central da tabela},

INIC {posição inicial do intervalo de busca},

FIM {posição final do intervalo de busca}

:inteiros

ACHOU {valor lógico que representa o sucesso da busca}:lógica

TAB {tabela a ser consultada} vetor

início

solicitar a entrada do tamanho da tabela, ler (N)

solicitar a entrada dos dados da tabela (ordenada)

para I de 1 até N faça

ler ( TAB[I] )

fim para

solicitar a entrada do elemento a ser procurado, ler(DADO)

## Pesquisa Binária

```
INIC ← 1
FIM ← N
ACHOU ← falso
enquanto (INIC <= FIM ) e (ACHOU = falso) faça
início
    MEIO ← (INIC + FIM) div 2
    se DADO = TAB[MEIO] então
        ACHOU ← verdade
    senão se DADO < TAB[MEIO] então
        FIM ← MEIO-1
    senão INIC ← MEIO+1
    fim se
fim se
fim enquanto
```

```
se ACHOU
    então escrever (DADO, “ se encontra na posicao”, MEIO)
    senão escrever (DADO, “nao se encontra na tabela”)
end.
```

INIC  $\leftarrow$  1

FIM  $\leftarrow$  N

Inicialização do intervalo de busca

## Pesquisa Binária

ACHOU  $\leftarrow$  falso

enquanto (INIC  $\leq$  FIM ) e (ACHOU = falso) faça  
início

MEIO  $\leftarrow$  (INIC + FIM) div 2

se DADO = TAB[MEIO] então

ACHOU  $\leftarrow$  verdade

senão se DADO < TAB[MEIO] então

FIM  $\leftarrow$  MEIO-1

senão INIC  $\leftarrow$  MEIO+1

fim se

fim se

fim enquanto

Determinação da posição central

Mudança do intervalo de busca

se ACHOU

então escrever (DADO, “ se encontra na posicao”, MEIO)

senão escrever (DADO, “nao se encontra na tabela”)

end.

INIC  $\leftarrow$  1

FIM  $\leftarrow$  N

ACHOU  $\leftarrow$  falso

enquanto (INIC  $\leq$  FIM ) e (ACHOU=falso) faça  
início

MEIO  $\leftarrow$  (INIC + FIM) div 2

se DADO = TAB[MEIO] então

ACHOU  $\leftarrow$  verdade

senão se DADO < TAB[MEIO] então

FIM  $\leftarrow$  MEIO-1

senão INIC  $\leftarrow$  MEIO+1

fim se

fim se

fim enquanto

## Pesquisa Binária

Teste de Mesa

N = 4

DADO = 6

TAB = -1 1 2 6

INIC	FIM	MEIO	TAB[MEIO]
------	-----	------	-----------

1	4	2	1	$\rightarrow$	$6 < 1$
---	---	---	---	---------------	---------

3	4	3	2	$\rightarrow$	$6 < 2$
---	---	---	---	---------------	---------

4	4	4	6	$\rightarrow$	$6 = 6$
---	---	---	---	---------------	---------