

# Algoritmos e Estrutura de Dados II

## Algoritmos de Busca

prof. Frederico Santos de Oliveira

Universidade Federal de Mato Grosso  
Instituto de Engenharia



# Roteiro

- 1 Introdução
- 2 Objetivos
- 3 Pesquisa Sequencial
- 4 Pesquisa Binária
- 5 Referências bibliográficas

## Busca

Segundo Sedgewick e Wayne (2011), sem algoritmos de busca, o desenvolvimento da infra-estrutura computacional que gozamos no mundo moderno não teria sido possível.

- A busca é uma das funções mais importantes, principalmente em Bancos de Dados;
- Visa encontrar um registro a partir de uma chave;
- Neste contexto, utiliza-se o termo **tabela de símbolos** (ou dicionário).

# Introdução

- A tabela de símbolos é um tipo abstrato de dados formado por registros, em que cada registro armazena um **valor** (a informação) e uma **chave** (para busca);
- A natureza do valor e da chave depende da aplicação;
- A tabela de símbolos, independente de como é implementada, deve disponibilizar as ações de inserir, buscar e remover registros.

## Busca

De acordo com Ziviani (2011), a escolha do método de pesquisa mais adequado depende principalmente (i) da quantidade de dados e (ii) da quantidade de operações.

# Introdução

- Diversas estruturas de dados suportam implementar a tabela de símbolos:
  - Vetores;
  - Listas encadeadas;
  - Árvores (Binária, AVL, Rubro-Negra);
  - Tabelas *hash*.
- A estrutura de dados utilizada irá definir o método de busca possível.

# Objetivos

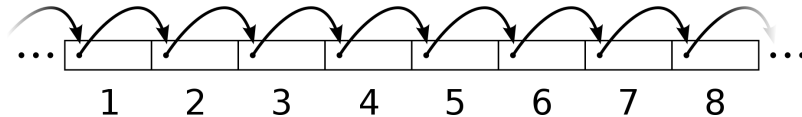
Esta aula tem como objetivos:

- 1 Apresentar os métodos de busca: sequencial e binária;
- 2 Demonstrar a execução dos algoritmos de busca por meio de exemplos;
- 3 Analisar o desempenho dos métodos apresentados.

# Pesquisa Sequencial

- Também chamada de pesquisa exaustiva;
- É o método de pesquisa mais simples que existe;
- Utilizado em dados desordenados;
- Pesquisa sequencialmente do primeiro até o último registro;
  - Quando encontrar a chave desejada, para.
- Formas de implementação:
  - Utilizando vetor;
  - Utilizando lista encadeada.

# Execução Pesquisa Sequencial





# Pseudo-código pesquisa sequencial

---

## Algoritmo 1: Pesquisa Sequencial

---

**Entrada:** Vetor  $V[0..n-1]$  de tamanho  $n$ , chave  $x$  de busca.

**Saída:** Posição  $i$  de  $x$  em  $V$  se  $x \in V$  ou  $-1$  se  $x \notin V$ .

```
1 início
2   para  $(i \leftarrow 0$  até  $n-1)$  faça
3       se  $(x = V[i])$  então
4           retorna  $i$ 
5   retorna  $-1$ 
```

---

Adaptado de Oliveira (2011)

# Pesquisa Sequencial

Chaves ordenadas

E se as chaves estiverem ordenadas, há melhora?

## Chaves ordenadas

E se as chaves estiverem ordenadas, há melhora?

Sim!

Ao encontrar um elemento maior que  $x$ , encerra a busca.

# Pseudo-código pesquisa sequencial (Chaves ordenadas)

---

## Algoritmo 2: Pesquisa Sequencial Ordenada

---

**Entrada:** Vetor  $V[0..n-1]$  de tamanho  $n$  em ordem crescente, chave  $x$  de busca.

**Saída:** Posição  $i$  de  $x$  em  $V$  se  $x \in V$  ou  $-1$  se  $x \notin V$ .

```
1 início
2   para ( $i \leftarrow 0$  até  $n-1$ ) faça
3     se ( $x = V[i]$ ) então
4       retorna  $i$ 
5     senão se ( $V[i] > x$ ) então
6       retorna  $-1$ 
7   retorna  $-1$ 
```

---

# Pesquisa Sequencial

Análise da busca sequencial em termos da ordem de crescimento no número de operações:

- Melhor caso: 1 operação =  $O(1)$ ;
- Pior caso:  $n$  operações =  $O(n)$ ;
- Caso médio:  $\frac{(n+1)}{2}$  operações =  $O(n)$ .

# Pesquisa Sequencial

## Vantagens

- Codificação simples;
- Há melhora de desempenho se os dados buscados com maior frequência forem movidos para o início da sequência;
- Dados podem estar desordenados:
  - Facilidade na inserção de dados.

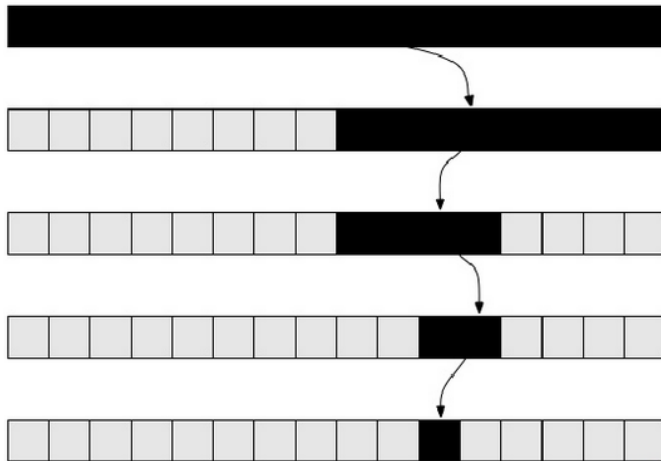
## Desvantagem

Ineficiente para uma quantidade grande de dados.

# Pesquisa Binária

- A pesquisa pode ser muito mais eficiente se os registros forem mantidos em ordem;
- Nesse caso, compare a chave com a chave do registro que se encontra no meio da tabela:
  - Se a chave é menor, procure na primeira metade da tabela;
  - Se é maior, procure na segunda metade da tabela;
  - Se for igual, a chave foi encontrada;
  - Repita o processo até que a chave seja encontrada ou que reste apenas um elemento.
- A **pesquisa binária** possui esse nome pois, a cada iteração, divide o vetor de busca pela metade;
- Pode ser implementada apenas utilizando vetores.

# Execução Pesquisa Binária





# Pseudo-código pesquisa binária (Iterativo)

**Algoritmo 3:** BuscaBináriaIterativa( $V, x, i, f$ )

**Entrada:** Vetor  $V$ , chave  $x$  de busca, índices inicial  $i$  e final  $f$  de  $V$ .

**Saída:** Índice  $m$  da posição de  $x$  em  $V$  ou determina que  $x \notin V$ .

```
1 início
2   encontrado  $\leftarrow$  FALSE
3   enquanto ( $i \leq f$ ) AND ( NOT (encontrado)) faça
4        $m \leftarrow \lfloor \frac{i+f}{2} \rfloor$ 
5       se ( $V[m] = x$ ) então
6           encontrado  $\leftarrow$  TRUE
7       senão se ( $x < V[m]$ ) então
8            $f \leftarrow m - 1$ 
9       senão
10           $i \leftarrow m + 1$ 
11   se (encontrado) então
12       retorna  $m$ 
13   senão
14       retorna “Não encontrado”
```

# Pesquisa binária

## Versão recursiva

É possível implementar uma versão mais elegante?

# Pesquisa binária

## Versão recursiva

É possível implementar uma versão mais elegante?

Sim!  
Utilizando recursão.

# Pseudo-código pesquisa binária (Recursivo)

**Algoritmo 4:** BuscaBinária( $V, x, i, f$ )

**Entrada:** Vetor  $V$ , chave  $x$  de busca, índices inicial  $i$  e final  $f$  de  $V$ .

**Saída:** Índice  $i$  da posição de  $x$  em  $V$  ou determina que  $x \notin V$ .

```
1 início
2   se  $(i = f)$  então
3       se  $(V[i] = x)$  então
4           retorna  $i$ 
5       senão
6           retorna “Não encontrado”
7    $m \leftarrow \lfloor \frac{i+f}{2} \rfloor$ 
8   se  $(V[m] > x)$  então
9       retorna BuscaBinária( $V, x, i, m - 1$ )
10  senão
11      retorna BuscaBinária( $V, x, m, f$ )
```

## Exemplo Pesquisa Binária

Procurar o registro com chave = 34.

3	4	11	12	21	34	65	77	78	98
---	---	----	----	----	----	----	----	----	----

## Exemplo Pesquisa Binária

Procurar o registro com chave = 34.

3	4	11	12	21	34	65	77	78	98
---	---	----	----	----	----	----	----	----	----

1ª Looping → 

3	4	11	12	21	34	65	77	78	98
---	---	----	----	----	----	----	----	----	----

2ª Looping → 

34	65	77	78	98
----	----	----	----	----

3ª Looping → 

34	65
----	----



valor igual a chave de busca!

# Análise da Pesquisa Binária

## Versão iterativa

- No melhor caso,  $O(1)$ ;
- No caso médio,  $O(\lg(n))$
- No pior caso,  $O(\lg(n))$ ;

## Versão recursiva

- A cada nível de recursão, o tamanho da entrada do problema é dividido por 2;
- Não é realizada nenhuma combinação, assim a recorrência é da forma  $T(n) = T(\frac{n}{2}) + c$ ;
- Utilizando o método mestre (caso 2), a solução é  $\theta(\lg(n))$ .

## Vantagens





- Muito mais rápido em relação à pesquisa sequencial;
- Simplicidade da implementação;
- Caso o registro não seja encontrado, já fornece a posição em que este pode ser inserido.

## Desvantagens

- Só pode ser realizada em vetores;
- Os dados devem estar ordenados;
- O custo de ordenação é alto ( $O(n \log n)$ ).



## Referências bibliográficas

-  FEOFILOFF, P. *Busca em vetor ordenado*. 2015. Disponível em: <http://www.ime.usp.br/{~}pf/algoritmos/aulas/bubi.ht>.
-  OLIVEIRA, S. L. G. *Algoritmos e seus fundamentos*. 1. ed. Lavras: Editora UFLA, 2011.
-  SEDGEWICK, R.; WAYNE, K. *Algorithms, 4th Edition*. Boston: Addison-Wesley, 2011. I-XII, 1-955 p. ISBN 978-0-321-57351-3.
-  ZIVIANI, N. *Projeto de Algoritmos: com implementações em Pascal e C*. São Paulo: Cengage Learning, 2011. ISBN 9788522110506.

# Algoritmos e Estrutura de Dados II

## Algoritmos de Busca

prof. Frederico Santos de Oliveira

Universidade Federal de Mato Grosso  
Instituto de Engenharia



# FIM

