

# CI1055: Algoritmos e Estruturas de Dados I

Profs. Drs. Marcos Castilho, Bruno Müller Jr e Carmem Hara

Departamento de Informática/UFPR

18 de agosto de 2020

## Resumo

Busca em vetores e manipulação de vetores ordenados

## Problema

Determinar se um valor pertence ao vetor.

Aplicações:

- localizar um livro na biblioteca
- encontrar o cadastro de um cliente de um banco
- encontrar dados de uma pessoa pelo seu CPF

- Existem diversos algoritmos de busca
- Nesta disciplina: somente os mais básicos
  - Busca simples
  - Busca com sentinela
  - Busca em vetor ordenado

# Função de busca em vetor

## A função deve receber:

- o elemento que vai ser buscado
- o vetor onde vai ser feita a busca
- o tamanho do vetor

## A função deve retornar:

- se o elemento for encontrado: o rótulo do elemento no vetor
- caso contrário: 0 (zero)

## Exemplo:

- $\text{busca}(23, v, 10) \rightarrow 4$
- $\text{busca}(13, v, 10) \rightarrow 0$

1	2	3	4	5	6	7	8	9	10
15	12	27	23	7	2	0	18	19	21

# Versão 1- Busca simples

```
1 function busca( x: real; var v: vetor_r; n: integer): integer;  
2 var i: integer;  
3 begin  
4     busca:= 0;  
5     for i:= 1 to n do  
6         if v[i] = x then  
7             busca:= i;  
8 end;
```

# Análise da Versão 1

```
1 function busca( x: real; var v: vetor_r; n: integer): integer;  
2 var i: integer;  
3 begin  
4     busca:= 0;  
5     for i:= 1 to n do  
6         if v[i] = x then  
7             busca:= i;  
8 end;
```

- Quantas vezes o comando dentro do for é executado?
  - n vezes
- Quantas comparações são feitas a cada repetição?
  - 2 comparações: ( $i \leq n$ ) e ( $v[i] = x$ )
  - O custo é  $n*2 = 2n$

```
1 function busca( x: real; var v: vetor_r; n: integer): integer;  
2 var i: integer;  
3 begin  
4     busca:= 0;  
5     for i:= 1 to n do  
6         if v[i] = x then  
7             busca:= i;  
8 end;
```

- Quantas vezes o comando dentro do for é executado?
- **n** vezes
- Quantas comparações são feitas a cada repetição?
- **2** comparações: ( $i \leq n$ ) e ( $v[i] = x$ )
- O custo é  $n*2 = 2n$

## Versão 2 - Para quando o elemento é encontrado

```
1 function busca( x: real; var v: vetor_r; n: integer): integer;  
2 var i : integer;  
3     achou : boolean;  
4 begin  
5     achou:= false;  
6     i:= 0;  
7     while (i < n) and not achou do  
8     begin  
9         i:= i + 1;  
10        if v[i] = x then  
11            achou:= true;  
12        end;  
13        if achou then  
14            busca:= i  
15        else  
16            busca:= 0;  
17 end;
```



- Na média: esta função é melhor do que a Versão 1
- No pior caso: o custo é mais alto
  - Acontece quando o valor não é encontrado
  - Quantidade de repetições:  $n$
  - Comparações por repetição:  $3 (i < n)$  (*notachou*) ( $v[i] = x$ )
  - Custo:  $3n$

## Versão 3 - Busca com sentinela

- Diminuir a quantidade de comparações garantindo que o valor buscado SEMPRE é encontrado

1	2	3	4	5	6	7	8	9	10
15	12	27	23	7	2	0	18	19	21

1	2	3	4	5	6	7	8	9	10	11
15	12	27	23	7	2	0	18	19	21	13

- $\text{busca}(13, v, 10) \rightarrow 11$
- **Idéia**
  - Colocar o elemento buscado no final do vetor (**sentinela**)
  - Com a garantia que o elemento sempre vai ser achado não é preciso testar o final do vetor

```
1 function busca( x: real; var v: vetor_r; n: integer ): integer;  
2 var i: integer;  
3 begin  
4     v[n+1]:= x;  
5     i:= 1;  
6     while v[i]  $\diamond$  x do  
7         i:= i + 1;  
8     if i  $\leq$  n then  
9         busca:= i  
10    else  
11        busca:= 0;  
12 end;
```

- No caso médio:
  - Quantidade de repetições:  $n / 2$
  - Comparações por repetição: 1
- No pior caso:
  - Quantidade de repetições:  $n+1$
  - Comparações por repetição: 1
- É preciso garantir uma posição a mais no vetor.

```
1 const min_r = 1; max_r = 50;  
2     min_i = 1; max_i = 10;  
3  
4 type vetor_r = array[min_r .. max_r+1] of real;  
5     vetor_i = array[min_i .. max_i+1] of integer;
```

- Primeira idéia: parar quando o elemento do vetor for maior ou igual ao procurado.

1	2	3	4	5	6	7	8	9	10	11
0	2	7	12	15	18	19	21	23	27	13

- busca( 13, v, 10 ): pode parar no elemento com rótulo 5

## Versão 4 - Busca em vetor ordenado

```
1 function busca( x: real; var v: vetor_r; n: integer): integer;  
2 var i: integer;  
3 begin  
4     v[n+1]:= x;  
5     i:= 1;  
6     while v[i] < x do  
7         i:= i + 1;  
8     if (v[i] = x) and (i <= n) then  
9         busca:= i  
10    else  
11        busca:= 0;  
12 end;
```

- No caso médio:
  - Quantidade de repetições:  $n / 2$
  - Comparações por repetição: 1
- No pior caso:
  - Quantidade de repetições:  $n+1$
  - Comparações por repetição: 1

## Problema

A função não está explorando completamente o fato do vetor estar ordenado.

# Busca binária

- Similar à busca de uma palavra em um dicionário, com palavras ordenadas de 'A' a 'Z'.
- Exemplo: palavra 'jovem'  
**Passo 1:** abre o dicionário mais ou menos no meio e verifica a palavra.



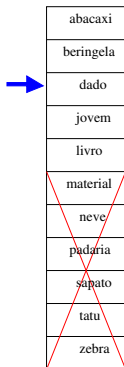
abacaxi
beringela
dado
jovem
livro
material
neve
padaria
sapato
tatu
zebra

Dicionário: 11 elementos



# Busca binária

- Similar à busca de uma palavra em um dicionário, com palavras ordenadas de 'A' a 'Z'.
- Exemplo: palavra 'jovem'  
**Passo 2:** desconsidera a parte que não pode conter 'jovem' e repete o processo.



abacaxi
beringela
→ dado
jovem
livro
<del>material</del>
<del>neve</del>
<del>padaria</del>
<del>sapato</del>
tatu
<del>zebra</del>

Dicionário: 5 elementos

# Busca binária

- Similar à busca de uma palavra em um dicionário, com palavras ordenadas de 'A' a 'Z'.
- Exemplo: palavra 'jovem'  
**Passo 3:** desconsidera a parte que não pode conter 'jovem' e repete o processo.



<del>abacaxi</del>
beringela
<del>dado</del>
jovem
livro
<del>material</del>
neve
<del>padaria</del>
<del>sapato</del>
tatu
<del>zebra</del>

Dicionário: 2 elementos

- Só 3 comparações até achar a palavra
- A cada repetição o dicionário tem o tamanho dividido pela metade
- Comportamento *logaritmico*

# Controle do espaço de busca no vetor

Busca do número **7**

Início

	<b>início</b>									<b>fim</b>
	1	2	3	4	5	6	7	8	9	10
0	2	7	12	15	18	19	21	23	27	

# Controle do espaço de busca no vetor

Busca do número **7**

Calcula o meio e compara com 7

	<b>inicio</b>				<b>meio</b>					<b>fim</b>
	1	2	3	4	5	6	7	8	9	10
0	2	7	12	15	18	19	21	23	27	

# Controle do espaço de busca no vetor

Busca do número **7**

Muda o fim

	<b>inicio</b>			<b>fim</b>						
	1	2	3	4	5	6	7	8	9	10
0	2	7	12	15	18	19	21	23	27	

# Controle do espaço de busca no vetor

Busca do número **7**

Calcula o meio e compara com 7

início	meio		fim						
1	2	3	4	5	6	7	8	9	10
0	2	7	12	15	18	19	21	23	27

# Controle do espaço de busca no vetor

Busca do número **7**

Muda o início

		início	fim						
1	2	3	4	5	6	7	8	9	10
0	2	7	12	15	18	19	21	23	27

# Controle do espaço de busca no vetor

Busca do número **7**

Calcula o meio e compara com 7

Achou!!

			meio							
		inicio		fim						
1	2	3	4	5	6	7	8	9	10	
0	2	7	12	15	18	19	21	23	27	



# O que acontece quando o elemento não é encontrado

Busca do número 5

Calcula o meio e compara com 5

		meio								
		inicio	fim							
1	2	3	4	5	6	7	8	9	10	
0	2	7	12	15	18	19	21	23	27	

# O que acontece quando o elemento não é encontrado

Busca do número 5

Muda o fim: **fim** < **inicio**

		fim	inicio							
	1	2	3	4	5	6	7	8	9	10
	0	2	7	12	15	18	19	21	23	27

# Busca binária

```
1 function busca( x: real; var v: vetor_r; n: integer ): integer;  
2 var inicio, fim, meio: integer;  
3 begin  
4     inicio:= 1;  
5     fim:= n;  
6     meio:= (inicio + fim) div 2;  
7     while (v[meio] <> x) and (inicio <= fim) do  
8         begin  
9             if v[meio] > x then  
10                fim:= meio - 1  
11            else  
12                inicio:= meio + 1;  
13                meio:= (inicio + fim) div 2;  
14            end;  
15            if inicio <= fim then  
16                busca:= meio  
17            else  
18                busca:= 0;  
19 end;
```

# Comparação entre as Versões

Busca	$n = 10$	$n = 10^2$	$n = 10^4$	$n = 10^8$
<b>Simples</b>	20	200	20.000	200.000.000
<b>Com sentinela</b>	10	100	10.000	100.000.000
<b>Em vetor ordenado</b>	10	100	10.000	100.000.000
<b>Binária</b>	3	5	10	19

Quantidade de comparações

# Manipulação de vetores ordenados

- Remoção de elemento do vetor
- Inserção de elemento no vetor
- Intercalação de vetores ordenados

# Remoção de vetor ordenado

- Entrada do procedimento:
  - posição do elemento a ser removido
  - vetor ordenado de onde será feita a remoção
  - tamanho do vetor
- Saída
  - vetor ordenado sem o elemento
  - tamanho do vetor atualizado

										tam
1	2	3	4	5	6	7	8	9	10	
0	2	7	12	15	18	19	21	23	27	

Após **remove( 4, v, tam)**

										tam
1	2	3	4	5	6	7	8	9	10	
0	2	7	15	18	19	21	23	27	27	

# Remoção de vetor ordenado

```
1 procedure remove( pos: integer; var v: vetor_r; var n: integer );  
2 var i: integer;  
3 begin  
4     for i:= pos to n-1 do  
5         v[i]:= v[i+1];  
6     n:= n - 1;  
7 end;
```

# Inserção em vetor ordenado

- Entrada do procedimento:
  - elemento a ser inserido
  - vetor ordenado onde será feita a inserção
  - tamanho do vetor
- Saída
  - vetor ordenado com o elemento
  - tamanho do vetor ordenado

1	2	3	4	5	6	7	8	9	10	11
5	7	9	12	15	18	19	21	23	27	?

tam

Após **insere( 17, v, tam)**

1	2	3	4	5	6	7	8	9	10	11
5	7	9	12	15	17	18	19	21	23	27

tam



# Inserção em vetor ordenado - Versão 1

```
1 procedure insere( x: real; var v: vetor_r; var n: integer );
2 var i: integer;
3 begin
4     i:= n;
5     while x < v[i] do
6         begin
7             v[i+1]:= v[i];
8             i:= i - 1;
9         end;
10    v[i+1]:= x;
11    n:= n + 1;
12 end;
```

## Pergunta

O que acontece se tentarmos inserir o número 3?

										tam	
1	2	3	4	5	6	7	8	9	10	11	
5	7	9	12	15	18	19	21	23	27	?	

# Inserção em vetor ordenado com sentinela

```
1 procedure insere( x: real; var v: vetor_r; var n: integer );
2 var i: integer;
3 begin
4     v[0]:= x;
5     i:= n;
6     while x < v[i] do
7         begin
8             v[i+1]:= v[i];
9             i:= i - 1;
10        end;
11    v[i+1]:= x;
12    n:= n + 1;
13 end;
```

											tam	
0	1	2	3	4	5	6	7	8	9	10	11	
3	5	7	9	12	15	18	19	21	23	27	?	

# Inserção em vetor ordenado com sentinela

```
1  procedure insere( x: real; var v: vetor_r; var n: integer );
2  var i: integer;
3  begin
4      v[0]:= x;
5      i:= n;
6      while x < v[i] do
7          begin
8              v[i+1]:= v[i];
9              i:= i - 1;
10         end;
11         v[i+1]:= x;
12         n:= n + 1;
13 end;
```

Após a inserção

												tam
0	1	2	3	4	5	6	7	8	9	10	11	
3	3	5	7	9	12	15	18	19	21	23	27	

# Intercalação de vetores

- Entrada do procedimento
  - vetor ordenado  $v1$
  - tamanho de  $v1$  ( $n1$ )
  - vetor ordenado  $v2$
  - tamanho de  $v2$  ( $n2$ )
- Saída
  - vetor  $vR$  com elementos de  $v1$  e  $v2$  ordenados
  - tamanho de  $vR$  ( $nR$ )

	1	2	3	4	5	$n1$	7	8	9	10
<b>v1</b>	5	7	9	12	15	18				

	1	2	$n2$	4	5	6	7	8	9	10
<b>v2</b>	3	6	14							

	1	2	3	4	5	6	7	8	9	10
<b>vR</b>	3	5	6	7	9	12	14	15	18	

# Intercalação de vetores

```
1 procedure merge
2   (var vR, v1, v2: vetor_r;
3     var nR: integer; n1, n2: integer);
4 var iR, i1, i2: integer;
5 begin
6   iR:= 1; i1:= 1; i2:= 1;
7   while (i1 <= n1) and (i2 <= n2) do
8     begin
9       if v1[i1] < v2[i2] then
10        begin
11          vR[iR]:= v1[i1];
12          i1:= i1 + 1;
13        end
14        else
15        begin
16          vR[iR]:= v2[i2];
17          i2:= i2 + 1;
18        end;
19      iR:= iR + 1;
20    end;
```

```
21
22
23
24
25
26
27
28
29
30
31
32
33
34
    if i1 <= n1 then
      for i1:= i1 to n1 do
        begin
          vR[iR]:= v1[i1];
          iR:= iR + 1;
        end
      else
        for i2:= i2 to n2 do
          begin
            vR[iR]:= v2[i2];
            iR:= iR + 1;
          end;
        nR:= n1 + n2;
      end;
```

# Fim do tópico

- o conteúdo desta aula está no livro no capítulo 9, seção 5

- Slides feitos em  $\text{\LaTeX}$  usando beamer
- Licença

*Creative Commons* Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License.<http://creativecommons.org/licenses/by-nc-nd/2.5/br/>

Creative Commons Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License.<http://creativecommons.org/licenses/by-nc-nd/2.5/br/>