

CI1055: Algoritmos e Estruturas de Dados I

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

Departamento de Informática/UFPR

30 de novembro de 2021

Resumo

Busca em vetores

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;
```

```
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: $3n$

- 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

- 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
material
neve
padaria
sapato
tatu
zebra

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.



abacaxi
beringela
dado
jovem
livro
material
neve
padaria
sapato
tatu
zebra

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

	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

	inicio				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**

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

Quantidade de comparações

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

- Slides feitos em \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/>