

DCC405 – ESTRUTURA DE DADOS II

Aula 14 – Busca Sequencial e Busca Binária

- Nós já implementamos a busca de elementos nas aulas passadas. Porém para consolidar esse conteúdo e comparar com outras buscas que darão base a algoritmos de busca, se faz necessário a revisão de alguns conceitos.

Busca em arranjo

- **Suponha que temos um arranjo de inteiros. Como fazemos para verificar se um determinado número está lá?**

Busca em arranjo

- **Suponha que temos um arranjo de inteiros. Como fazemos para verificar se um determinado número está lá?**
 - Varremos o arranjo, da esquerda para a direita. Se acharmos o número, então ele está no arranjo. Se chegarmos ao final do arranjo e não acharmos ele não está.

Busca em arranjo

- **Suponha que temos um arranjo de inteiros. Como fazemos para verificar se um determinado número está lá?**
 - Varremos o arranjo, da esquerda para a direita. Se acharmos o número, então ele está no arranjo. Se chegarmos ao final do arranjo e não acharmos ele não está.

Busca Sequencial ou Busca Linear

Busca Sequencial - Exemplo

Elemento buscado: 32

n = 10

									n-1
0	1	2	3	4	5	6	7	8	9
52	125	-4	32	55	69	-78	200	0	63

Busca Sequencial - Exemplo

Elemento buscado: 32

$n = 10$

0	1	2	3	4	5	6	7	8	9
52	125	-4	32	55	69	-78	200	0	63

$n-1$


i

Busca Sequencial - Exemplo

Elemento buscado: 32

$n = 10$

0	1	2	3	4	5	6	7	8	n-1 9
52	125	-4	32	55	69	-78	200	0	63



Busca Sequencial - Exemplo

Elemento buscado: 32

$n = 10$

0	1	2	3	4	5	6	7	8	9
52	125	-4	32	55	69	-78	200	0	63

$n-1$

i

Busca Sequencial - Exemplo

Elemento buscado: 32

$n = 10$

0	1	2	3	4	5	6	7	8	9
52	125	-4	32	55	69	-78	200	0	63




Encontrou! Índice 3 (quarta posição)

Busca Sequencial - Exemplo

Elemento buscado: -53

$n = 10$

0	1	2	3	4	5	6	7	8	9	n-1
52	125	-4	32	55	69	-78	200	0	63	



i

Busca Sequencial - Exemplo

Elemento buscado: -53

$n = 10$

									n-1
0	1	2	3	4	5	6	7	8	9
52	125	-4	32	55	69	-78	200	0	63




i

Busca Sequencial - Exemplo

Elemento buscado: -53

n = 10

									n-1
0	1	2	3	4	5	6	7	8	9
52	125	-4	32	55	69	-78	200	0	63



i

Busca Sequencial - Exemplo

Elemento buscado: -53

$n = 10$

0	1	2	3	4	5	6	7	8	9
52	125	-4	32	55	69	-78	200	0	63

$n-1$

i

Busca Sequencial - Exemplo

Elemento buscado: -53

$n = 10$

0	1	2	3	4	5	6	7	8	9
52	125	-4	32	55	69	-78	200	0	63

$n-1$


↑
 i

Busca Sequencial - Exemplo

Elemento buscado: -53

$n = 10$

									n-1
0	1	2	3	4	5	6	7	8	9
52	125	-4	32	55	69	-78	200	0	63



Busca Sequencial - Exemplo

Elemento buscado: -53

$n = 10$

										n-1
0	1	2	3	4	5	6	7	8	9	
52	125	-4	32	55	69	-78	200	0	63	
										</

Busca Sequencial - Exemplo

Elemento buscado: -53

n = 10

									n-1
0	1	2	3	4	5	6	7	8	9
52	125	-4	32	55	69	-78	200	0	63

↑
i

Não encontrou!

Busca Sequencial - Exemplo

Elemento buscado: -53

$n = 10$

0	1	2	3	4	5	6	7	8	n-1 9
52	125	-4	32	55	69	-78	200	0	63

↑
 i

Não encontrou!
Como sabemos disso? $i == n$

Busca Sequencial - Exemplo

Elemento buscado: **-53**

$n = 10$

0	1	2	3	4	5	6	7	8	n-1 9
52	125	-4	32	55	69	-78	200	0	63

↑
 i

Não encontrou!
Como sabemos disso? $i == n$

Retorna -1

Busca Sequencial

```
#include <stdio.h>

int buscaSeq(int vetor[], int tamanho, int el) {
    int i;
    for (i = 0; i < tamanho; i++)
        if (vetor[i] == el) return i;
    return -1;
}

int main() {
    int n = 6;

    int v[] = {9, 8, 4, 6, 3, 4};
    printf("%d\n", buscaSeq(v, n, 4));
    printf("%d\n", buscaSeq(v, n, 12));

    return 0;
}
```

Busca Sequencial

```
#include <stdio.h>

int buscaSeq(int vetor[], int tamanho, int el) {
    int i;
    for (i = 0; i < tamanho; i++)
        if (vetor[i] == el) return i;
    return -1;
}

int main() {
    int n = 6;

    int v[] = {9, 8, 4, 6, 3, 4};
    printf("%d\n", buscaSeq(v, n, 4));
    printf("%d\n", buscaSeq(v, n, 12));

    return 0;
}
```

Saída:

```
acauan:aca$ ./"buscaSeq"
2
-1
```

Custo computacional:
 $O(n)$

Busca Sequencial

- O que posso fazer para melhorar a busca por **63**?

									n-1
0	1	2	3	4	5	6	7	8	9
52	125	-4	32	55	69	-78	200	0	63

Busca Sequencial

- O que posso fazer para melhorar a busca por **63**?

									n-1
0	1	2	3	4	5	6	7	8	9
52	125	-4	32	55	69	-78	200	0	63

- E se o vetor estivesse **ordenado**?

									n-1
0	1	2	3	4	5	6	7	8	9
-78	-4	0	32	52	55	63	69	125	200

Busca Sequencial

- E se o vetor estivesse **ordenado**?

0	1	2	3	4	5	6	7	8	9	n-1
-78	-4	0	32	52	55	63	69	125	200	

→ Podemos **parar a busca** assim que encontrarmos um número maior que ele

```
#include <stdio.h>

int buscaSeq(int vetor[], int tamanho, int el) {
    int i;
    for (i = 0; i < tamanho; i++) {
        if (vetor[i] == el) return i;
        if (vetor[i] > el) break;
    }

    return -1;
}

int main() {
    int n = 10;

    int v[] = {-78, -4, 0, 32, 52, 55, 63, 69, 125, 200};
    printf("%d\n", buscaSeq(v, n, 63));

    return 0;
}
```


Busca Sequencial

Na Busca Sequencial em um arranjo ordenado:

→ Potencialmente executamos **menos comparações** no caso do elemento não estar no arranjo.

→ Estar ordenado também **torna fácil** algumas tarefas:

- Busca pelo menor elemento: $v[0]$
- Busca pelo maior elemento: $v[n-1]$

Busca Sequencial

Ainda assim, no pior caso, teremos que olhar o arranjo inteiro, quando:

- O elemento buscado for o último
- O elemento buscado não estiver no arranjo, mas for maior que o último.

Busca Sequencial

Ainda assim, no pior caso, teremos que olhar o arranjo inteiro, quando:

- O elemento buscado for o último
- O elemento buscado não estiver no arranjo, mas for maior que o último.

Teria como melhorar isso?

Busca Sequencial

Ainda assim, no pior caso, teremos que olhar o arranjo inteiro, quando:

- O elemento buscado for o último
- O elemento buscado não estiver no arranjo, mas for maior que o último.

Teria como melhorar isso?

Resp.: Sim → Busca Binária

Busca Binária

→ **Condição obrigatória para busca binária: Arranjo ordenado.**

Algoritmo:

- Verifica se o elemento buscado é o do **meio do arranjo**
 - Se não for, verifique se é **maior**:
 - Se for, repita a busca na **metade direita** do arranjo.
 - Se não for, repita a busca na **metade esquerda** do arranjo.

Busca Binária - Exemplo

- Ex.: Buscando **52**

0	1	2	3	4	5	6	7	8	9
-78	-4	0	32	52	55	63	69	125	200

↑ Início

↑ Fim

Busca Binária - Exemplo

- Ex.: Buscando **52**

0	1	2	3	4	5	6	7	8	9
-78	-4	0	32	52	55	63	69	125	200
↑ Início				↑ Meio					↑ Fim

Como descobrir o meio de
uma maneira eficaz?

Busca Binária - Exemplo

- Ex.: Buscando **52**

0	1	2	3	4	5	6	7	8	9
-78	-4	0	32	52	55	63	69	125	200
↑ Início				↑ Meio					↑ Fim

v[m] == 52?

Busca Binária - Exemplo

- Ex.: Buscando **52**

0	1	2	3	4	5	6	7	8	9
-78	-4	0	32	52	55	63	69	125	200
↑				↑					↑
Início				Meio					Fim

$v[m] == 52?$

Achou em $O(1)$, apenas um passo.

Busca Binária - Exemplo

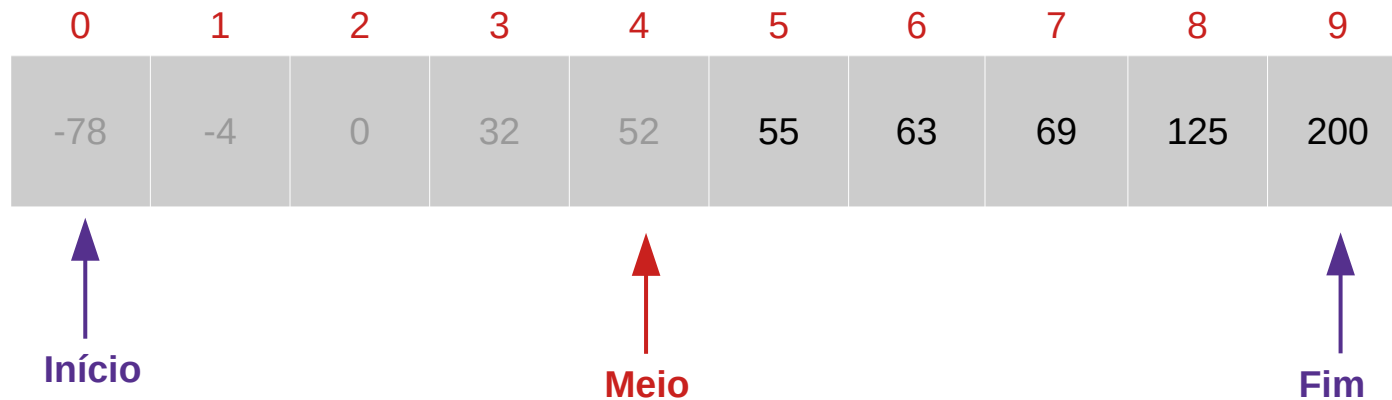
- Ex.: Buscando **55**

0	1	2	3	4	5	6	7	8	9
-78	-4	0	32	52	55	63	69	125	200
↑ Início				↑ Meio					↑ Fim

$v[m] == 55$? Não. $v[m] < 55$? Sim

Busca Binária - Exemplo

- Ex.: Buscando **55**



$v[m] == 55$? Não. $v[m] < 55$?

Podemos então descartar todo mundo a esquerda do meio

Busca Binária - Exemplo

- Ex.: Buscando **55**

0	1	2	3	4	5	6	7	8	9
-78	-4	0	32	52	55	63	69	125	200

↑
Início

↑
Fim

Atualizamos inicio e meio

Busca Binária - Exemplo

- Ex.: Buscando **55**

0	1	2	3	4	5	6	7	8	9
-78	-4	0	32	52	55	63	69	125	200


Diagram illustrating a binary search process on a sorted array. The array contains 10 elements, indexed 0 to 9. The target value 55 is located at index 5. The search process is shown with three pointers: 'Início' (Start) at index 5, 'Meio' (Middle) at index 7, and 'Fim' (End) at index 9. The 'Meio' pointer is highlighted in red, indicating the current middle element being compared.

Atualizamos início e meio

Busca Binária - Exemplo

- Ex.: Buscando **55**

0	1	2	3	4	5	6	7	8	9
-78	-4	0	32	52	55	63	69	125	200




$v[m] == 55?$ Não. $v[m] < 55?$ Não.

Busca Binária - Exemplo

- Ex.: Buscando **55**

0	1	2	3	4	5	6	7	8	9
-78	-4	0	32	52	55	63	69	125	200



$v[m] == 55$? Não. $v[m] < 55$? Não.

Descarta os que estão a direita do meio.

Busca Binária - Exemplo

- Ex.: Buscando **55**

0	1	2	3	4	5	6	7	8	9
-78	-4	0	32	52	55	63	69	125	200

↑ ↑
Início Fim

Atualizamos fim e meio

Busca Binária - Exemplo

- Ex.: Buscando **55**

0	1	2	3	4	5	6	7	8	9
-78	-4	0	32	52	55	63	69	125	200

↑ ↑ ↑
Início Meio Fim

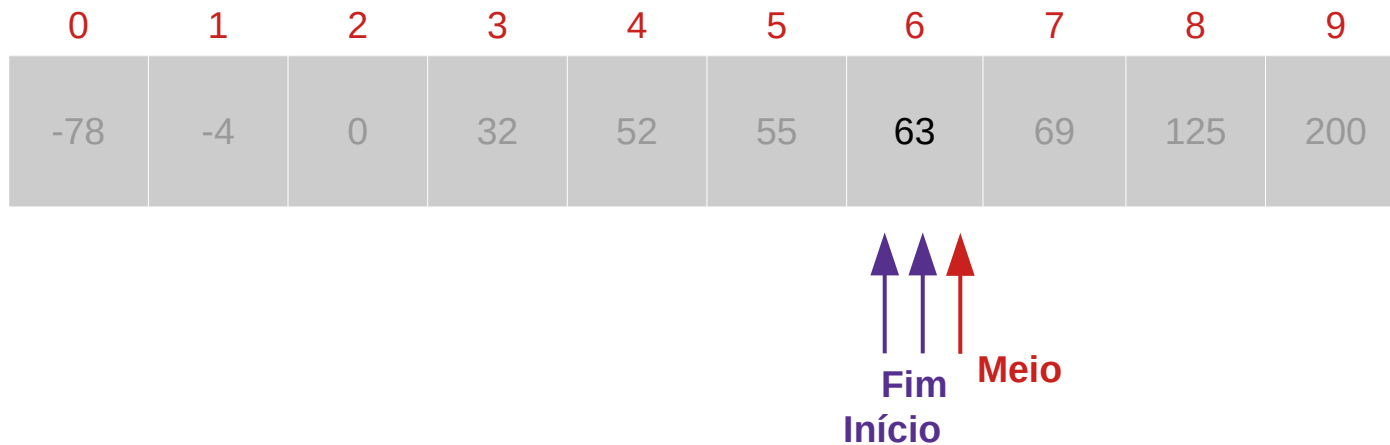
$v[m] == 55?$ Sim.

Encontrou!



Busca Binária - Exemplo

- Ex.: Buscando **60**



$v[m] == 60$? Não.

**E não tem mais o pra onde buscar.
Elemento não está no array**

Busca Binária - Implementação

```
#include <stdio.h>

int buscaBinaria(int vetor[], int tamanho, int el) {
    int ini = 0;
    int fim = tamanho - 1;

    while(ini <= fim) {
        int meio = (fim + ini) / 2;
        if(vetor[meio] < el)
            ini = meio + 1;
        else if(vetor[meio] > el)
            fim = meio - 1;
        else
            return meio;
    }
    return -1;
}

int main() {
    int n = 10;

    int v[] = {-78, -4, 0, 32, 52, 55, 63, 69, 125, 200};
    printf("%d\n", buscaBinaria(v, n, 55));

    return 0;
}
```

Saída:

```
acauan:aca$ ./"2.buscaBinaria"
5
```

Custo computacional:
 $O(\lg n)$

<https://pt.khanacademy.org/computing/computer-science/algorithms/binary-search/a/running-time-of-binary-search>

Complexidade Computacional

Exemplo:

- Lista telefônica com **18 milhões de entradas**
- Se cada comparação (a um elemento do arranjo) gasta $10\mu\text{s}$ (10 microsegundos ou milionésimo de segundo), como ficam os piores casos?
 - **Busca Sequencial:** $10/1000000 * 18000000 = 180\text{s} = 3 \text{ minutos}$
 - **Busca Binária:** $10/1000000 * \log_2 18000000 = 0.000241\text{s} = 0,24 \text{ milisegundos}$

Exercício

- Fazer **Exercício - Aula 14 - Busca Binária.pdf** no SIGAA.