

Algoritmos de Busca

Prof. Gabriel Sobral

FIAP

19 de agosto de 2024

`profgabriel.sobral@fiap.com.br`

Tópicos

Busca Sequencial

Busca Binária

Busca Sequencial

	0	1	2	3	4	5	6
V	23	4	67	-8	54	90	21

elem

54

 Elemento procurado

	0	1	2	3	4	5	6
i=0	23	4	67	-8	54	90	21

i=1	23	4	67	-8	54	90	21
-----	----	---	----	----	----	----	----

i=2	23	4	67	-8	54	90	21
-----	----	---	----	----	----	----	----

i=3	23	4	67	-8	54	90	21
-----	----	---	----	----	----	----	----

i=4	23	4	67	-8	54	90	21
-----	----	---	----	----	----	----	----

V : um vetor

x : um elemento a ser procurado em V

Algoritmo 1: Busca Sequencial(V, x)

para cada v *em* V **faça**

se $v = x$ **então**

retorna

 Sim

fim

retorna *Não*

```
def busca_sequencial(vetor: list[any], x: any) -> int:
    for i, elemento in enumerate(vetor):
        if elemento == x:
            return i

    return -1
```

Busca Sequencial

- ▶ pior caso: $O(n)$
- ▶ fácil implementação
- ▶ não é o algoritmo mais eficiente de busca

Busca Binária

	0	1	2	3	4	5	6	7	8	9
V	-8	-5	1	4	14	21	23	54	67	90

elem

4

 Elemento procurado

	0	1	2	3	4	5	6	7	8	9
meio=4	-8	-5	1	4	14	21	23	54	67	90

Valor é menor:
buscar no início

	0	1	2	3	4	5	6	7	8	9
meio=1	-8	-5	1	4	14	21	23	54	67	90

Valor é maior:
buscar no final

	0	1	2	3	4	5	6	7	8	9
meio=2	-8	-5	1	4	14	21	23	54	67	90

Valor é maior:
buscar no final

	0	1	2	3	4	5	6	7	8	9
meio=3	-8	-5	1	4	14	21	23	54	67	90

Valor é igual:
terminar a busca

V : um vetor

x : um elemento a ser procurado em V

Algoritmo 2: Busca Binária(V, e, d, x)

se $e > d$ **então**

 | **retorna** -1

fim

$meio \leftarrow (d + e)/2$

se $V[meio] = x$ **então**

 | **retorna** $meio$

senão

 | **se** $x > vetor[meio]$ **então**

 | Busca Binária($V, meio + 1, d, x$)

 | **senão**

 | Busca Binária($V, e, meio - 1, x$)

 | **fim**

fim


```
def busca_binaria_recursiva(vetor: list[any],
                             esquerdo: int,
                             direito: int,
                             x: int) -> int:

    if esquerdo > direito:
        return -1
    else:
        meio = (esquerdo + direito) // 2
        if vetor[meio] == x:
            return meio
        elif x > vetor[meio]:
            return busca_binaria_recursiva(vetor, meio + 1,
                                            direito, x)
        else:
            return busca_binaria_recursiva(vetor, esquerdo,
                                            meio - 1, x)
```

```
def busca_binaria_iterativa(vetor: list[any],
                             esquerdo: int,
                             direito: int,
                             x: int) -> int:
    while esquerdo <= direito:
        meio = esquerdo + (direito - esquerdo) // 2
        if vetor[meio] == x:
            return meio
        elif vetor[meio] < x:
            esquerdo = meio + 1
        else:
            direito = meio - 1
    return -1
```

Busca Binária

- ▶ pior caso: $O(\lg n)$
- ▶ fácil implementação
- ▶ algoritmo eficiente (quando o vetor está ordenado)