

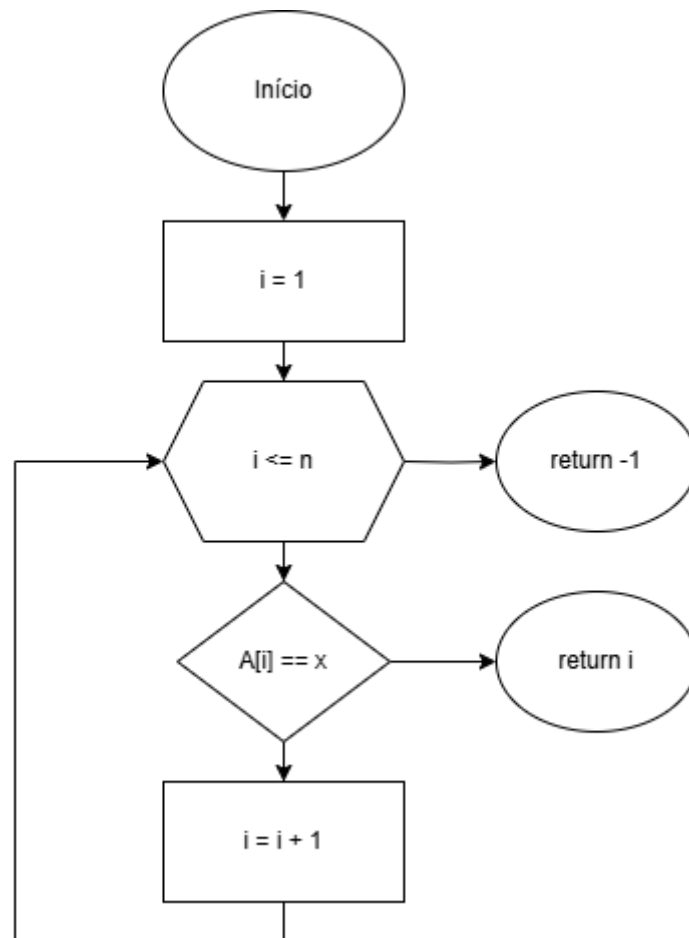
---

**Algoritmo**      **BUSCA LINEAR( $A, n, x$ )**

---

```
1  $i = 1$ 
2 enquanto  $i \leq n$  faça
3   se  $A[i] == x$  então
4     devolve  $i$ 
5    $i = i + 1$ 
6 devolve -1
```

---



$x \in A$ :

px = Quantidade de vezes que se passa pelo loop

n = Tamanho do Array

i = 1: Atribuição - T

px \* (

i <= n: Condição - T

A[i]: Acesso - T

A[i] == x: Condição - T

i + 1: Operação Aritmética - T

i = i + 1: Atribuição - T

)

return i = Retorno - T

Exceto na vez que se encontra o x em A, então não se passará por i = i + 1: -2t

**Fórmula:  $T(n) = t + 5tp_x - 2t + t = 5tp_x$**

x = A[i]:

i = 1: Atribuição - T

i <= n: Condição - T

A[i]: Acesso - T

A[i] == x: Condição - T

return i = Retorno - T

**Fórmula: 5t**

x = A[n]:

n = Tamanho do Array

i = 1: Atribuição - T

n \* (

i <= n: Condição - T

A[i]: Acesso - T

A[i] == x: Condição - T

i + 1: Operação Aritmética - T

i = i + 1: Atribuição - T

)

return i = Retorno - T

Exceto na última vez não se passa por i = i + 1: -2t

**Fórmula:  $t + 5tn - 2t + t = 5tn$**

x ∉ A:

n = Tamanho do Array

i = 1: Atribuição - T

n \* (

i <= n: Condição - T

A[i]: Acesso - T

A[i] == x: Condição - T

i + 1: Operação Aritmética - T

i = i + 1: Atribuição - T

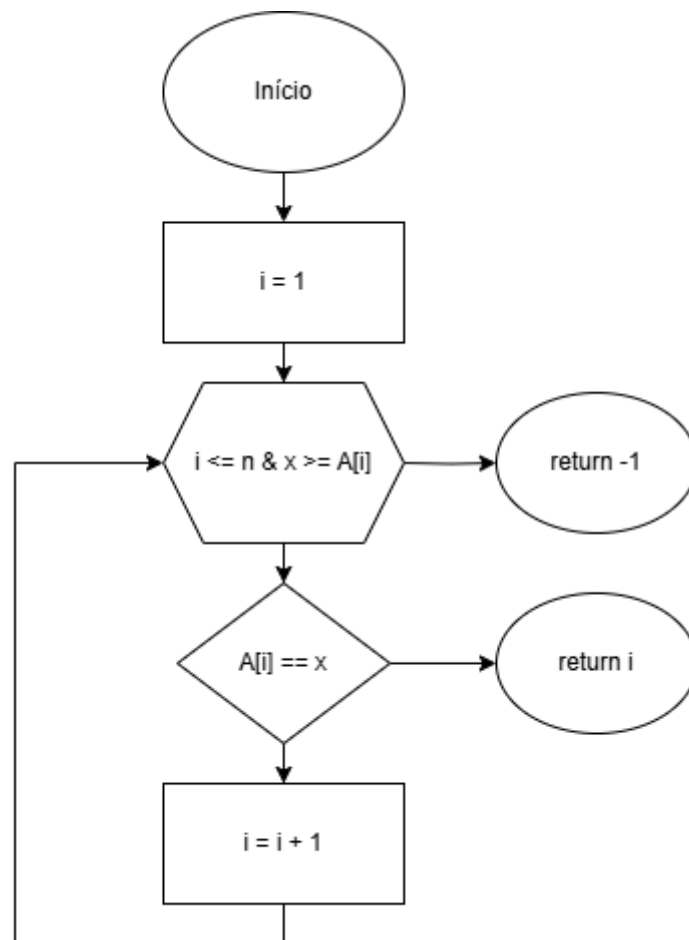
)

return i = Retorno - T

Quando o programa passar por todos os elementos do Array ele verificará uma última vez se o  $i \leq n$ , e depois irá para o retorno: + T

**Fórmula:  $t + 5tn + t + t = 5tn + 3t$**

Algoritmo	BUSCA LINEAR EM ORDEM( $A, n, x$ )
1	$i = 1$
2	enquanto $i \leq n$ e $x \geq A[i]$ faça
3	se $A[i] == x$ então
4	devolve $i$
5	$i = i + 1$
6	devolve -1



$x \in A$ :

px = Quantidade de vezes que se passa pelo loop

```

i = 1: Atribuição - T
px * (
i <= n: Condição - T
A[i]: Acesso - T
x >= A[i] - Condição - T
A[i]: Acesso - T
A[i] == x: Condição - T
i + 1: Operação Aritmética - T
i = i + 1: Atribuição - T
)
return i = Retorno - T

```

Exceto na vez que se encontra o x em A, então não se passará por  $i = i + 1$ :  $-2t$

**Fórmula:  $t + 7tpx - 2t + t: 7tpx$**

$x = A[i]:$

```

i = 1: Atribuição - T
i <= n: Condição - T
A[i]: Acesso - T
x >= A[i] - Condição - T
A[i]: Acesso - T
A[i] == x: Condição - T
return i = Retorno - T

```

**Fórmula:  $7t$**

$x = A[n]:$

$n =$  Tamanho do Array

```

i = 1: Atribuição - T
n * (
i <= n: Condição - T
A[i]: Acesso - T
x >= A[i] - Condição - T
A[i]: Acesso - T
A[i] == x: Condição - T
i + 1: Operação Aritmética - T
i = i + 1: Atribuição - T
)
return i = Retorno - T

```

Exceto na última vez não se passa por  $i = i + 1$ :  $-2t$

**Fórmula:  $t + 7tn - 2t + t: 7tn$**

$x \notin A$ :

$n$  = Tamanho do Array

$i = 1$ : Atribuição - T

$n * ($

$i \leq n$ : Condição - T

$A[i]$ : Acesso - T

$A[i] == x$ : Condição - T

$i + 1$ : Operação Aritmética - T

$i = i + 1$ : Atribuição - T

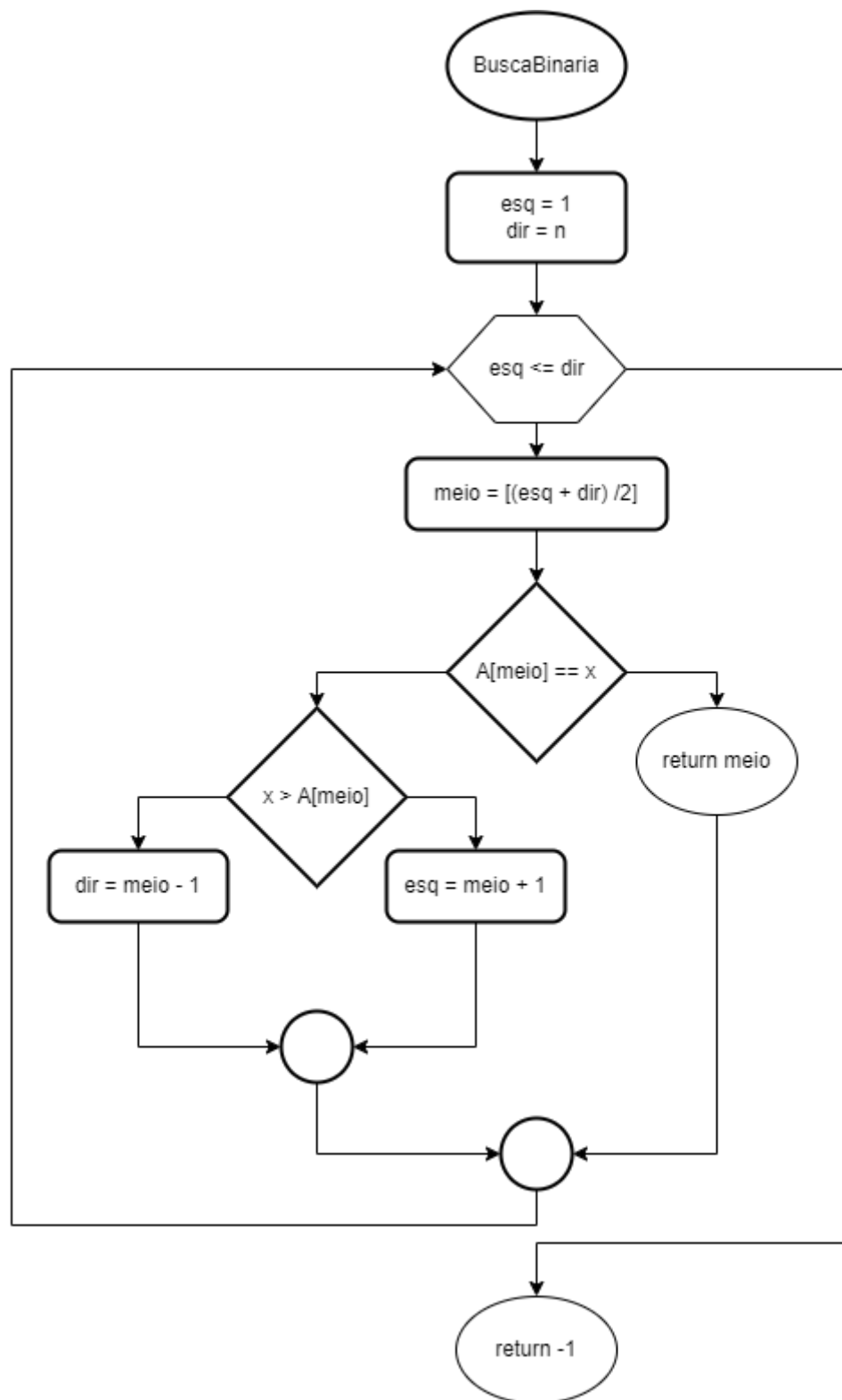
)

return  $i$  = Retorno - T

Quando o programa passar por todos os elementos do Array ele verificará uma última vez se o  $i \leq n$  e, e depois irá para o retorno. Um detalhe é que ao programa já verificar que essa primeira condição não é verdadeira, ele não realizará a segunda: + T

**Fórmula:  $t + 7tn + t + t: 7tn + 3t$**

Algoritmo	BUSCABINARIA( $A, n, x$ )
1	$esq = 1$
2	$dir = n$
3	enquanto $esq \leq dir$ faça
4	$meio = \lfloor (esq + dir) / 2 \rfloor$
5	se $A[meio] == x$ então
6	devolve $meio$
7	senão se $x > A[meio]$ então
8	$esq = meio + 1$
9	senão
10	$dir = meio - 1$
11	devolve $-1$



$x \in A$ :

esq = 1: Atribuição – T

dir = n: Atribuição – T

```

px * (
  esq <= dir: Condição – T
  esq + dir: Operação Aritmética – T
  (esq + dir) / 2: Operação Aritmética – T
  meio = (esq + dir) / 2: Operação Aritmética – T
  A[meio]: Acesso – T
  A[meio] == x: Condição – T
  A[meio]: Acesso – T
  x > A[meio]: Acesso – T

  meio – 1: Operação Aritmética – T
  dir = meio – 1: Atribuição – T
  OU
  meio + 1: Operação Aritmética – T
  esq = meio + 1: Atribuição – T
)
return meio: Retorno – T

```

Exceto na vez que se encontra o x em A, então não se passará por  $x > A[\text{meio}]$  e  $\text{dir} = \text{meio} - 1$  ou  $\text{esq} = \text{meio} + 1$ : - 4t

**Fórmula:  $2t + 8tpx - 4t + t = 8tpx - t$**

```

x = A[i]:

esq = 1: Atribuição – T
dir = n: Atribuição – T
esq <= dir: Condição – T
esq + dir: Operação Aritmética – T
(esq + dir) / 2: Operação Aritmética – T
meio = (esq + dir) / 2: Operação Aritmética – T
A[meio]: Acesso – T
A[meio] == x: Condição – T
return meio: Retorno - T

```

**Fórmula: 9t**

```

x = A[n]:

esq = 1: Atribuição – T
dir = n: Atribuição – T

n * (
  esq <= dir: Condição – T
  esq + dir: Operação Aritmética – T
  (esq + dir) / 2: Operação Aritmética – T
  meio = (esq + dir) / 2: Operação Aritmética – T
  A[meio]: Acesso – T

```

```

A[meio] == x: Condição – T
A[meio]: Acesso – T
x > A[meio]: Acesso – T

meio – 1: Operação Aritmética – T
dir = meio – 1: Atribuição – T
OU
meio + 1: Operação Aritmética – T
esq = meio + 1: Atribuição – T
)
return meio: Retorno – T

```

Exceto na última vez não se passa por  $x > A[\text{meio}]$  e  $\text{dir} = \text{meio} - 1$  ou  $\text{esq} = \text{meio} + 1$ : - 4t

**Fórmula:  $2t + 8tn - 4t + t = 8tn - t$**

$x \notin A$ :

```

esq = 1: Atribuição – T
dir = n: Atribuição – T

n * (
esq <= dir: Condição – T
esq + dir: Operação Aritmética – T
(esq + dir) / 2: Operação Aritmética – T
meio = (esq + dir) / 2: Operação Aritmética – T
A[meio]: Acesso – T
A[meio] == x: Condição – T
A[meio]: Acesso – T
x > A[meio]: Acesso – T

meio – 1: Operação Aritmética – T
dir = meio – 1: Atribuição – T
OU
meio + 1: Operação Aritmética – T
esq = meio + 1: Atribuição – T
)
return meio: Retorno – T

```

Quando o programa passar por todos os elementos do Array ele verificará uma última vez se  $\text{esq} \leq \text{dir}$ , e depois irá para o retorno: + T

**Fórmula:  $2t + 8tn + t + t = 8tn + 4t$**