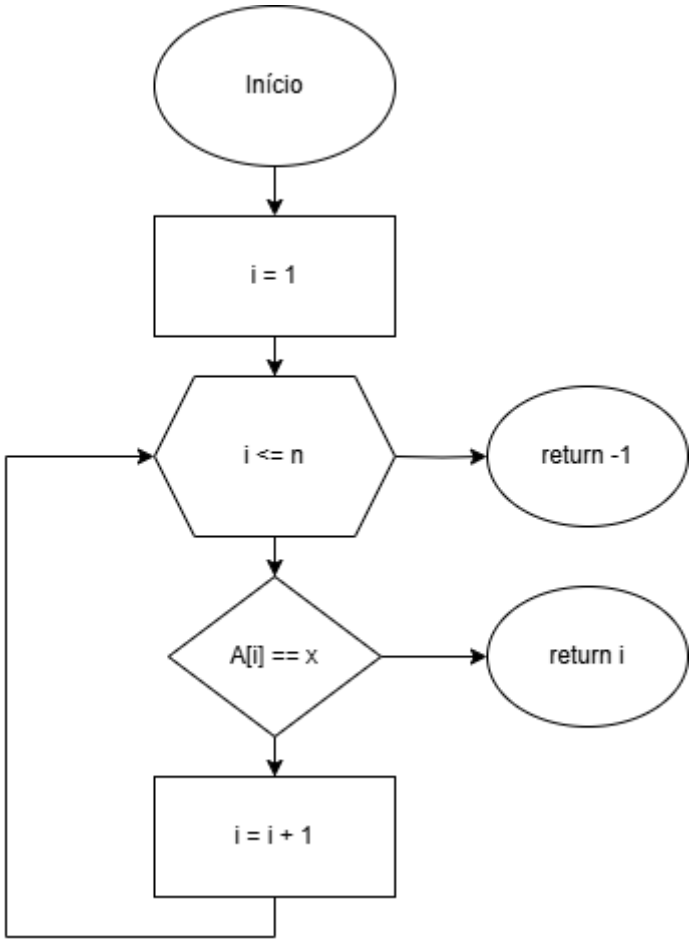
```
Algoritmo  BuscaLinear(A, n, x)
1 i = 1
2 enquanto i ≤ n faça
3     se A[i] == x então
4         devolve i
5     i = i + 1
6 devolve −1
```



x ∈ 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 + 5tpx – 2t + t = 5tpx

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

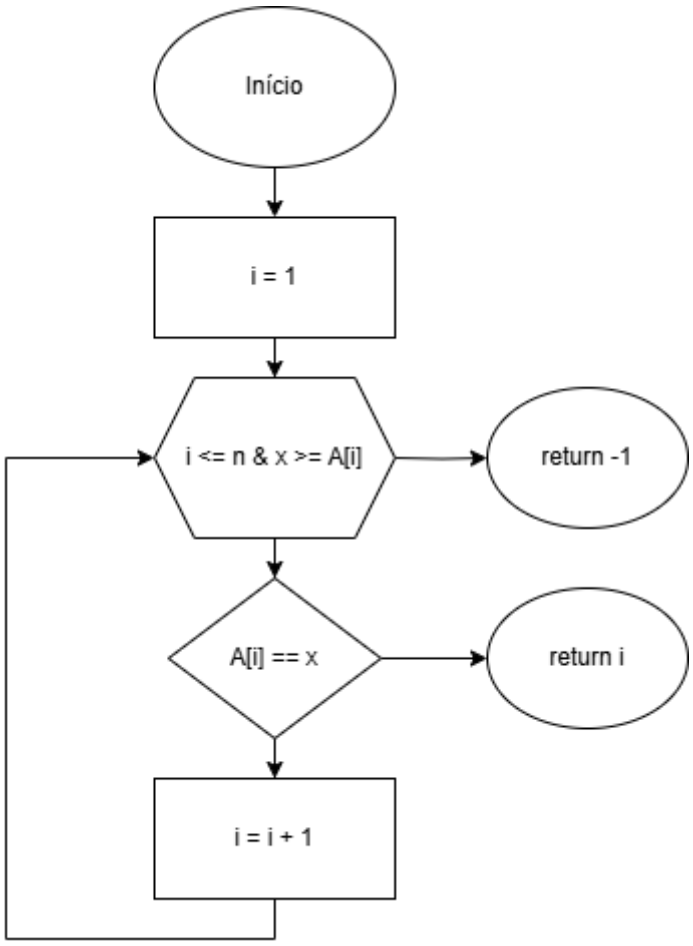
)
return i = Retorno - T

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

Fórmula: t + 5tn + t + t = 5tn + 3t

Algoritmo BUSCALINEAREMORDEM(A, n, x)

```
1 i = 1
2 enquanto i ≤ n e x ≥ A[i] faça
3     se A[i] == x então
4         devolve i
5     i = i + 1
6 devolve −1
```



x ∈ 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 ∉ A:

n = Tamanho do Array

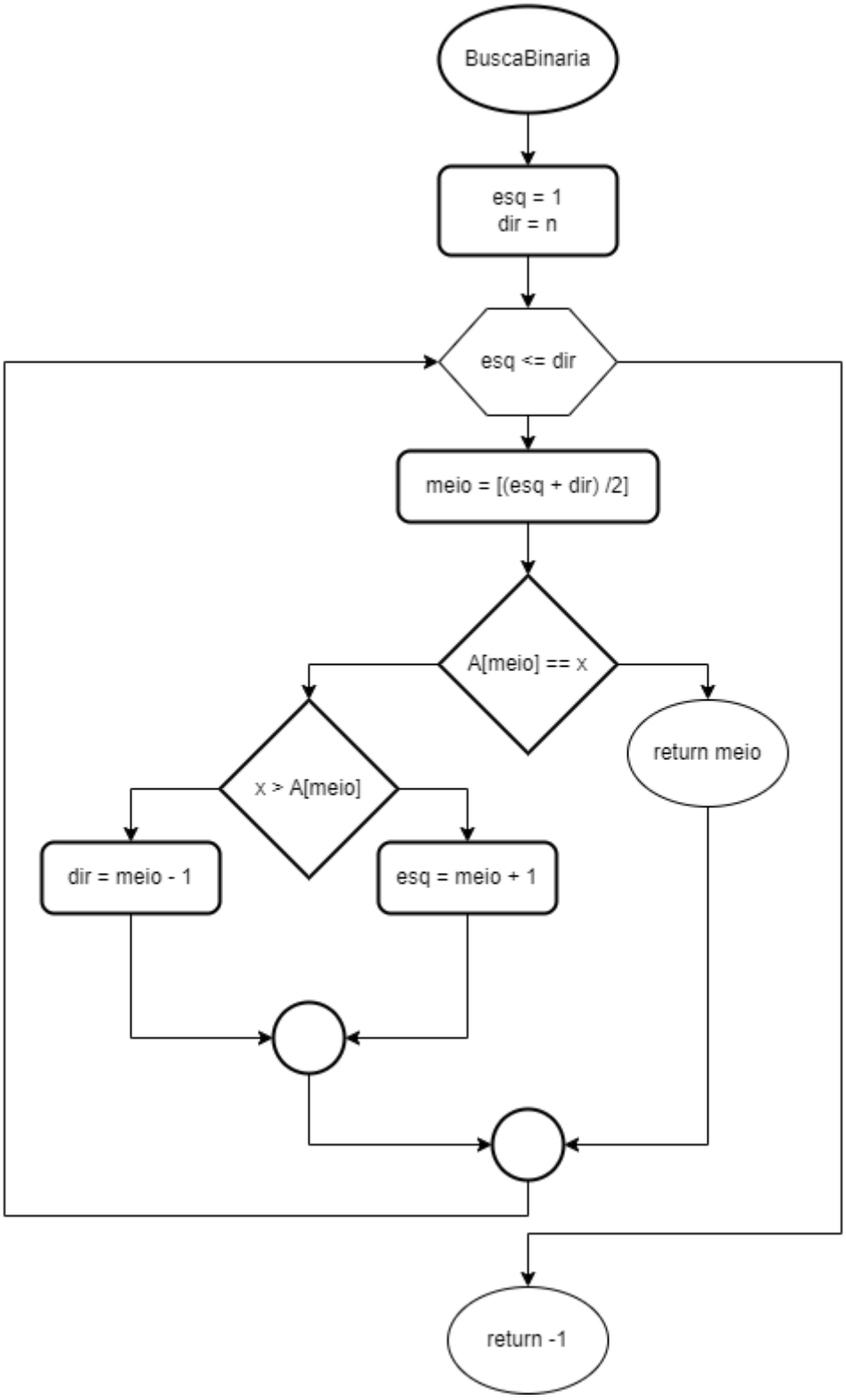
i = 1: Atribuição - T
n * (
i <= n: Condição - T
A[i]: Acesso - T
A[i] == x: Condição - T
i + 1: Operação Aritmética - T
i = i + 1: Atribuição -T
)
return i = Retorno - T

Quando o programa passar por todos os elementos do Array ele verificará uma última vez se o i <= 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 ≤ dir faça
4      meio = ⌊(esq + dir)/2⌋
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 ∈ A:

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

$\log_2(n)$ * (
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 vez que se encontra o x em A, então não se passará por x > A[meio] e dir = meio – 1 ou esq = meio + 1: - 4t

Fórmula: 2t + $\log_2(n)$ * **10t – 4t + t =** $\log_2(n)$ * **10t - t**

x = A[i]:

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

$\log_2(n)$ * (
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 vez que se encontra o x em A, então não se passará por x > A[meio] e dir = meio – 1 ou esq = meio + 1: - 4t

Fórmula: 2t + $\log_2(n)$ * **10t – 4t + t =** $\log_2(n)$ * **10t - t**

x = A[n]:

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

$\log_2(n)$ * (
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 vez que se encontra o x em A, então não se passará por x > A[meio] e dir = meio – 1 ou esq = meio + 1: - 4t

Fórmula: 2t + $\log_2(n)$ * **10t – 4t + t =** $\log_2(n)$ * **10t - t**

x ∉ A:

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

$\log_2(n)$ * (
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 o esq <= dir, e depois irá para o retorno: + T

Fórmula: 2t + $\log_2(n)$ * **10t + t + t =** $\log_2(n)$ * **10t + 3t**