



Universidade Federal do ABC
Centro de Matemática, Computação e Cognição

Algoritmos para Pesquisa

Monael Pinheiro Ribeiro, D.Sc.

Problema da Busca

- Formalmente:

- Suponha uma coleção **V** de elementos de tamanho **n**:

$$\mathbf{V} = \{v_0, v_1, v_2, \dots, v_{n-1}\}$$

- E um elemento **x** qualquer.

- Averiguar se **x** = **v_i**, onde $0 \leq i < n$











- Informalmente:

- Verificar se um elemento **x** está no vetor **V** de tamanho **n**. Se sim, retorne o índice **i**, tal que **v_i** = **x**, caso contrário retorne -1.

Busca Linear

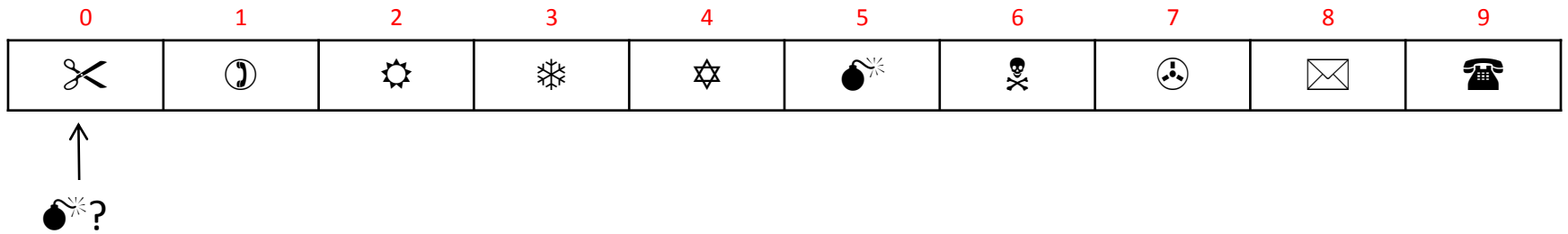
- Acesse cada posição i , $0 \leq i < n$, do vetor \mathbf{V} , averiguando se $\mathbf{v}_i = \mathbf{x}$.
 - Caso verdade, retorne i .
 - Caso contrário, verifique o próximo i .
- Caso todo vetor seja verificado e nunca satisfizesse a condição $\mathbf{v}_i = \mathbf{x}$, então retorne -1.

Busca Linear

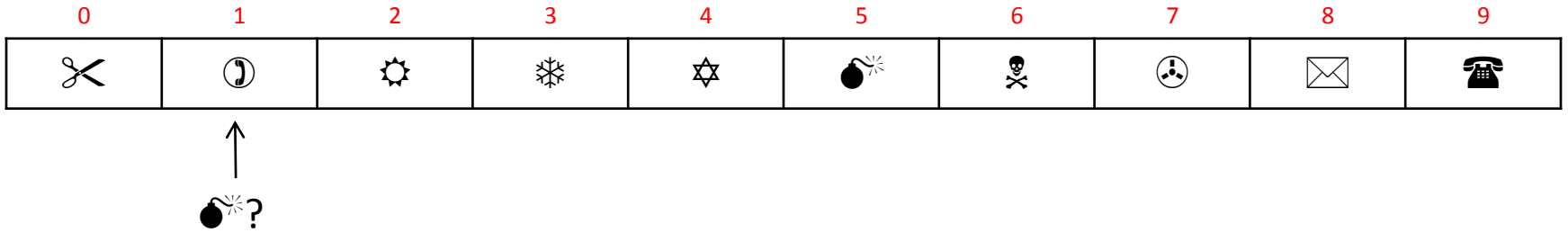
0	1	2	3	4	5	6	7	8	9
									

Buscar na lista: 

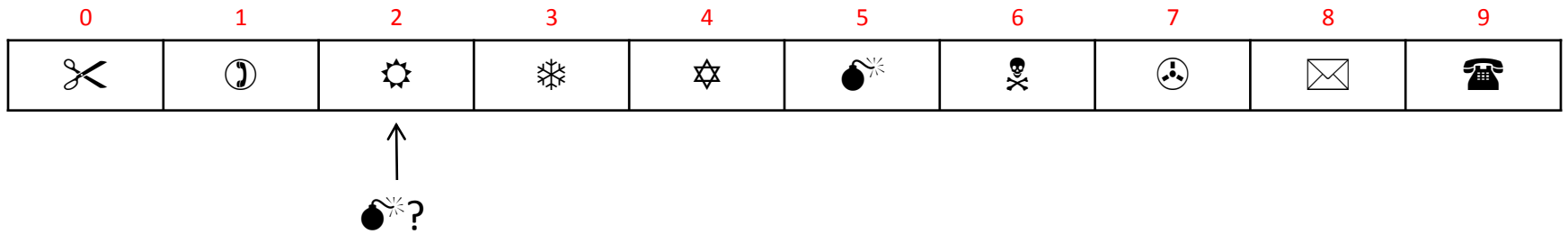
Busca Linear



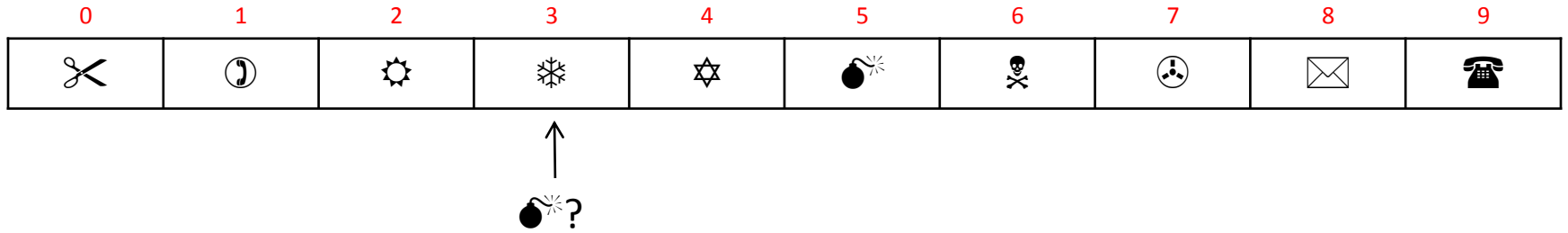
Busca Linear



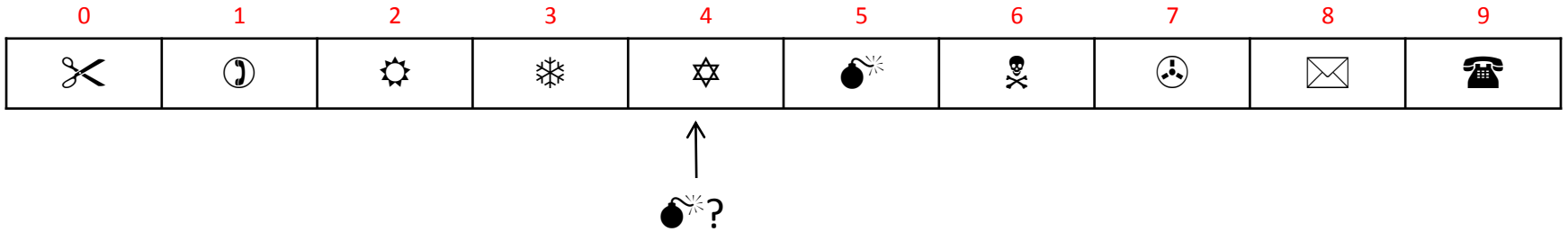
Busca Linear



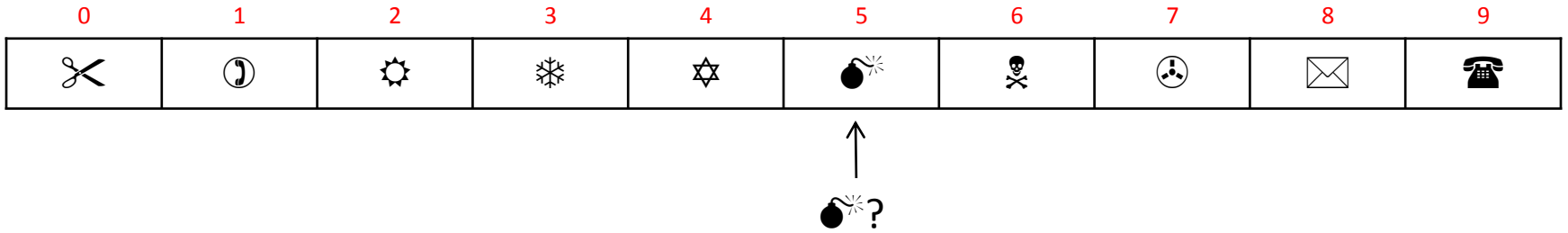
Busca Linear



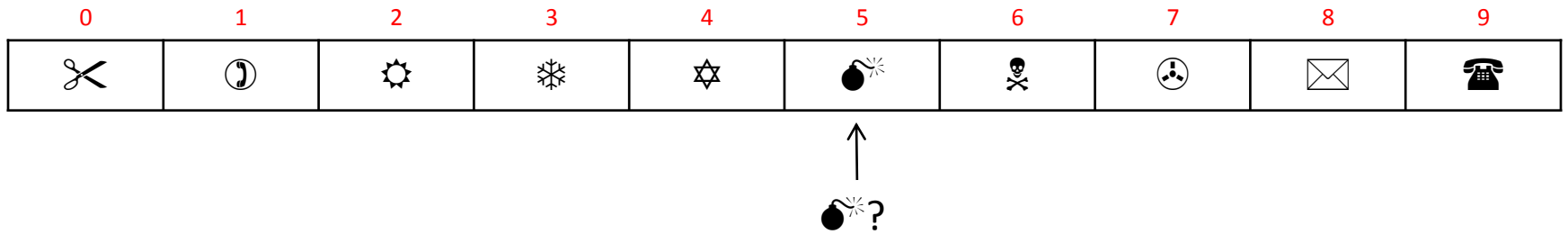
Busca Linear



Busca Linear

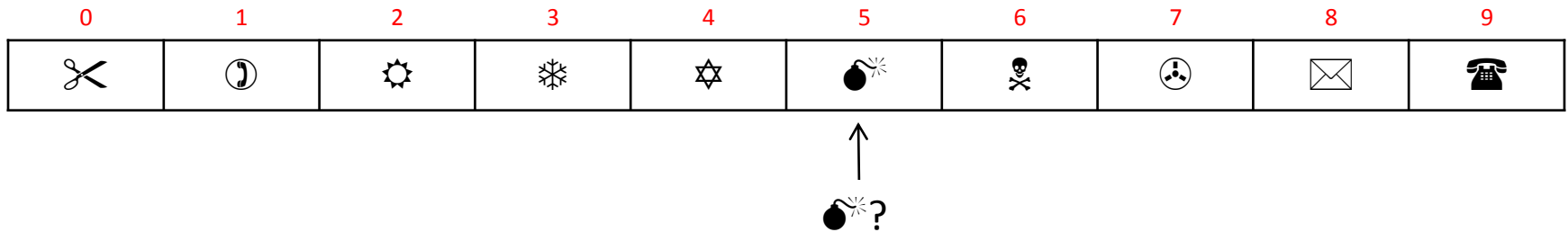


Busca Linear



Encontramos  com 6 comparações

Busca Linear













Qual o pior caso desse Algoritmo?

Quando ele terá o maior esforço computacional?

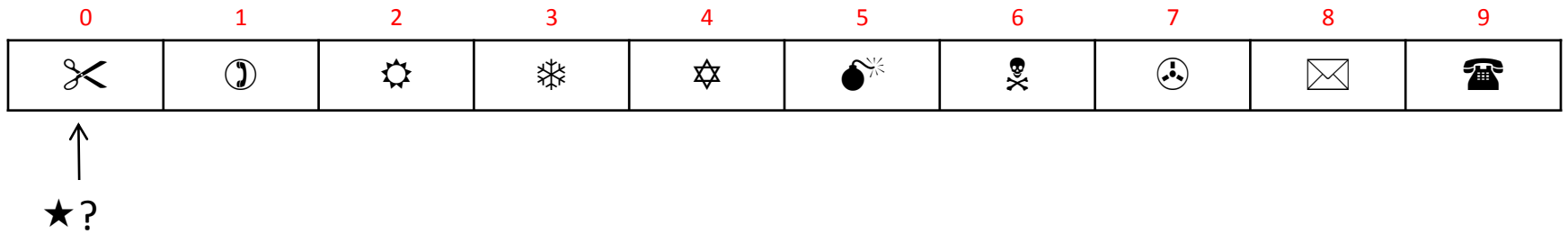
Quando ele fará o maior número de comparações?

Busca Linear

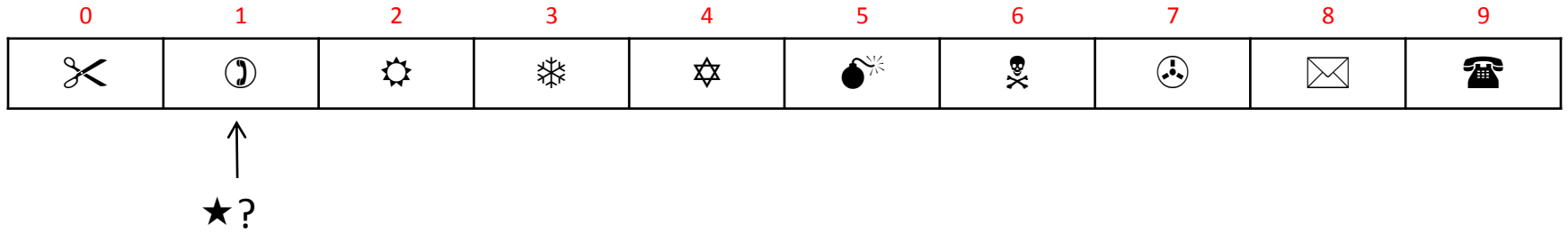
0	1	2	3	4	5	6	7	8	9
									

Buscar na lista: ★

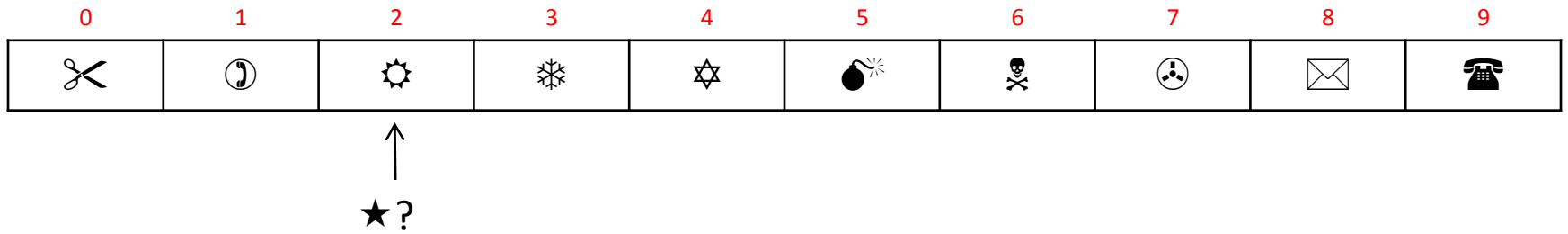
Busca Linear



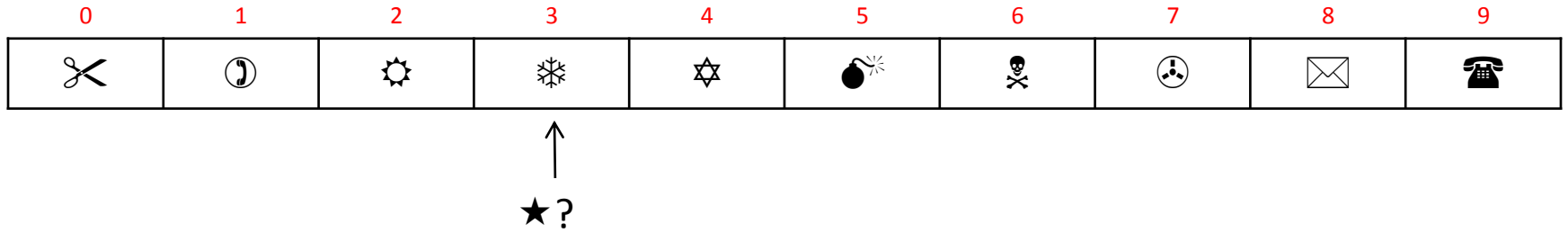
Busca Linear



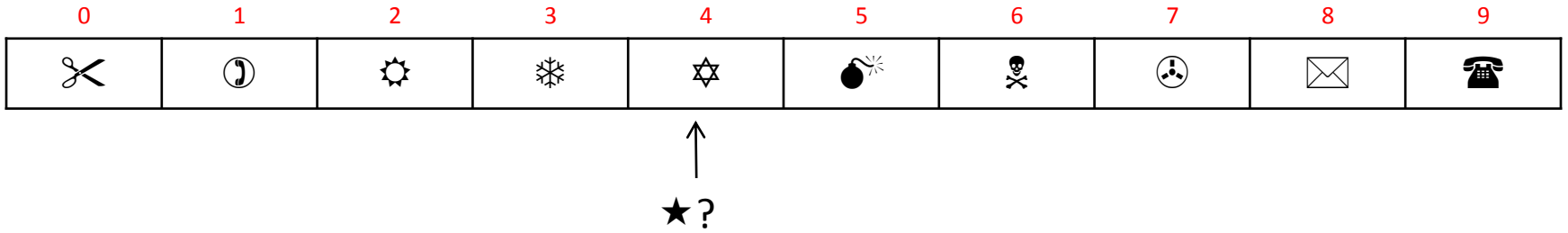
Busca Linear



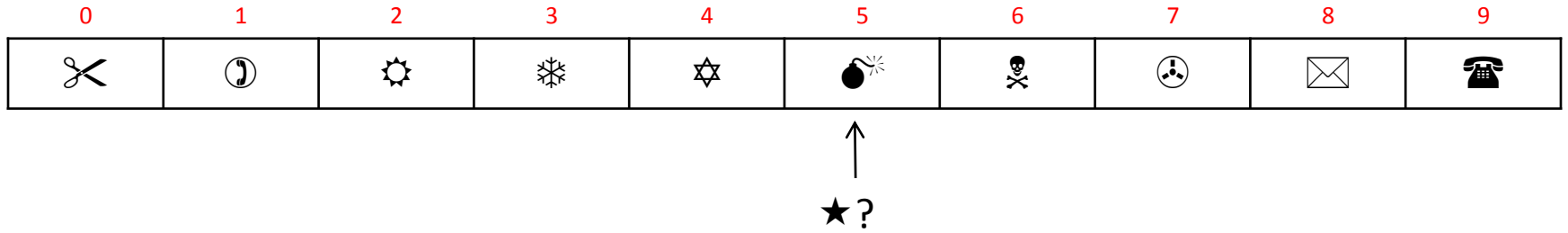
Busca Linear



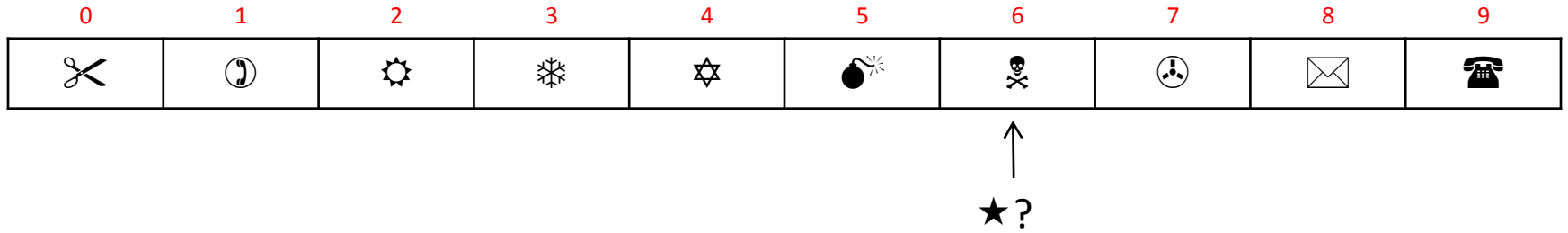
Busca Linear



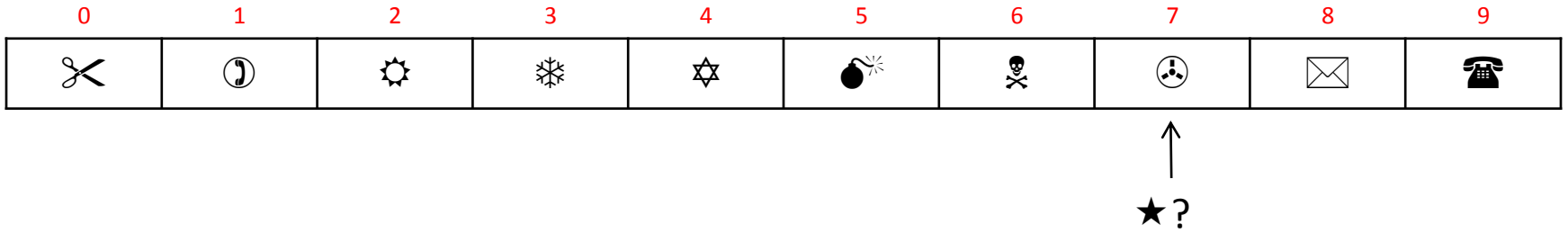
Busca Linear



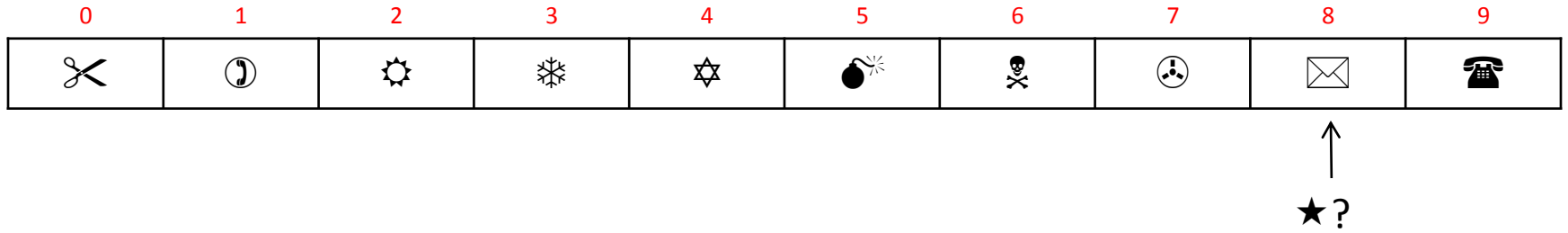
Busca Linear



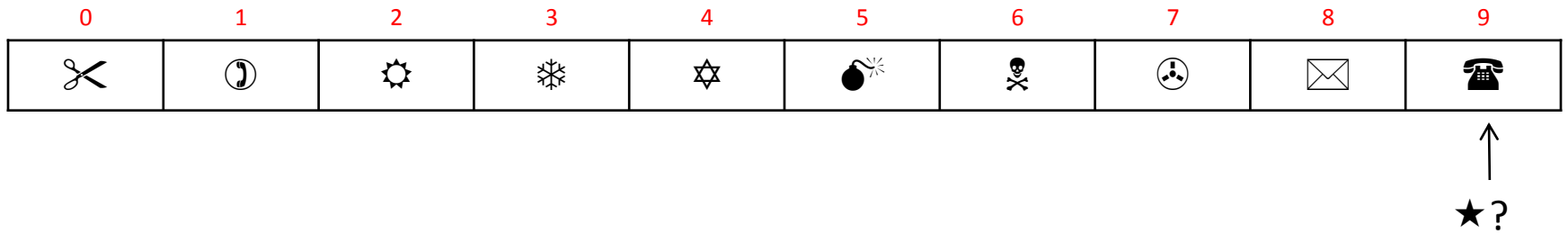
Busca Linear













Busca Linear

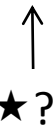


Busca Linear













Busca Linear

0	1	2	3	4	5	6	7	8	9
									



Foram necessárias 10 comparações para determinar que ★ não aparece na lista.











Busca Linear

0	1	2	3	4	5	6	7	8	9
									



Foram necessárias 10 comparações para determinar que ★ não aparece na lista.
E se a lista tivesse 100 elementos?

Busca Linear

0	1	2	3	4	5	6	7	8	9
									



Foram necessárias 10 comparações para determinar que ★ não aparece na lista.

E se a lista tivesse 100 elementos?

E se a lista tivesse 1×10^6 ?

E se a lista tivesse n elementos?

Complexidade de Algoritmos

- Foram necessárias 10 comparações para determinar que ★ não aparece na lista.
 - E se a lista tivesse 100 elementos?
 - E se a lista tivesse 1×10^6 ?
 - E se a lista tivesse n elementos?
- Perceba que a quantidade de comparações aumenta conforme aumenta o tamanho da lista.

Complexidade de Algoritmos

- Perceba que a quantidade de comparações aumenta conforme aumenta o tamanho da lista.

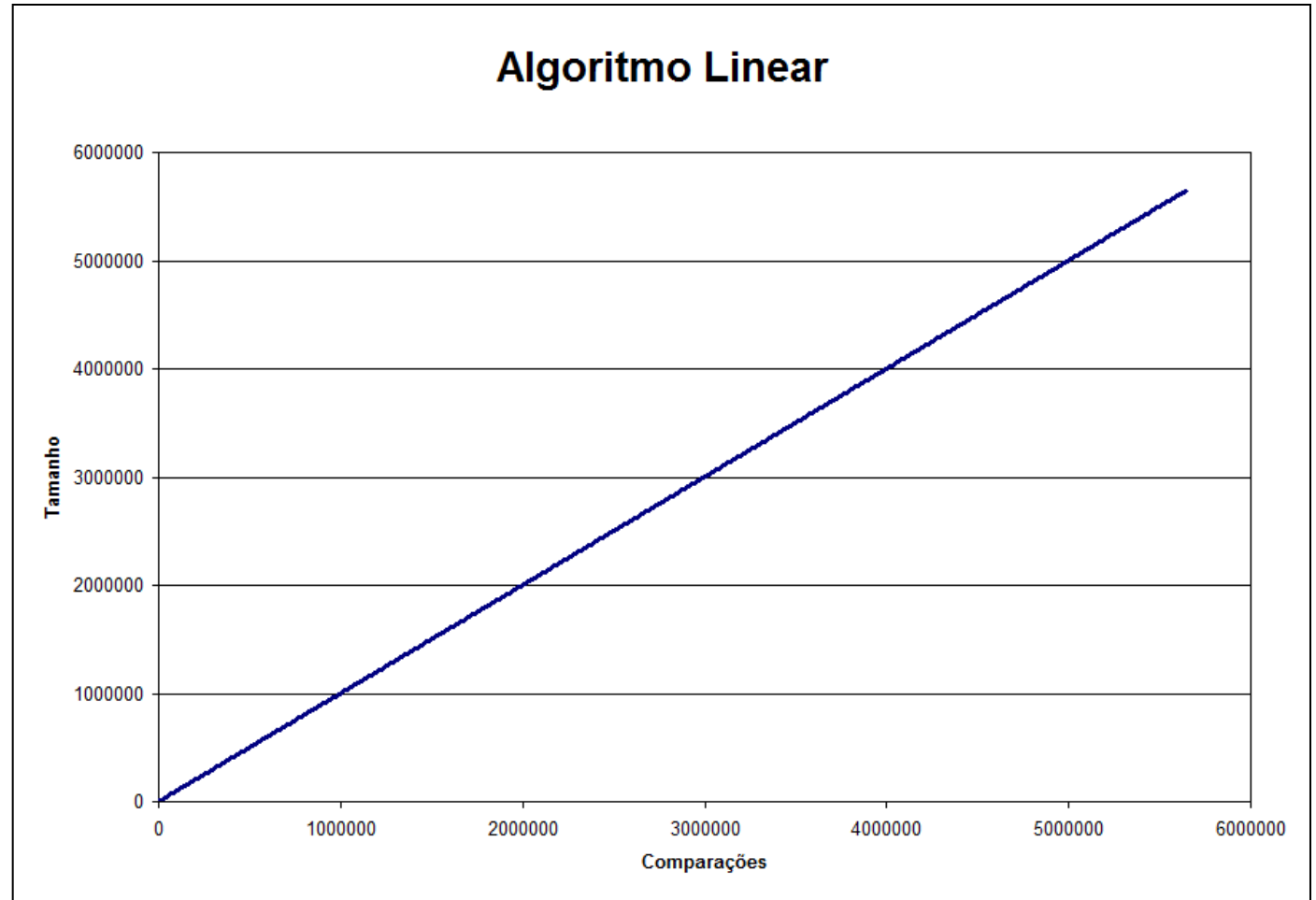
Tamanho	Comparações
10	10
100	100
1×10^6	1×10^6
1×10^{100}	1×10^{100}

- O esforço do algoritmo (quantidade de comparações) aumenta conforme aumenta-se o tamanho da lista a uma **taxa linear**.

Complexidade de Algoritmos

- Linear

$O(n)$



Busca Linear

- Algoritmo em C

```
01. int busca(int *v, int n, int x)
02. {
03.     int i;
04.     for(i=0; i<n; i++)
05.     {
06.         if(v[i] == x)
07.         {
08.             return i;
09.         }
10.     }
11.     return -1;
12. }
```

Busca Linear

- Seria possível fazer a busca linear recursiva?

Busca Linear

- Seria possível fazer a busca linear recursiva?
 - Qual o caso base?

Busca Linear

- Seria possível fazer a busca linear recursiva?
 - Qual o caso base?
 - Quando se encontra o elemento buscado.

Busca Linear

- Seria possível fazer a busca linear recursiva?
 - Qual o caso base?
 - Quando se encontra o elemento buscado.
 - Quando não há mais onde buscar.
(Acaba o espaço de busca)

Busca Linear

- Seria possível fazer a busca linear recursiva?
 - Qual o caso base?
 - Quando se encontra o elemento buscado.
 - Quando não há mais onde buscar.
(Acaba o espaço de busca)
 - Qual o passo recursivo?

Busca Linear

- Seria possível fazer a busca linear recursiva?
 - Qual o caso base?
 - Quando se encontra o elemento buscado.
 - Quando não há mais onde buscar.
(Acaba o espaço de busca)
 - Qual o passo recursivo?
 - Decremente o espaço de busca.

Exemplo

- Achar o professor na Universidade para pedir nota no último dia do quadrimestre.



Exemplo

- Achar o professor na Universidade para pedir nota no último dia do quadrimestre.

[illegible]

Exemplo

- Achar o professor na Universidade para pedir nota no último dia do quadrimestre.

Sala 1	Sala 2	Sala 3	Sala 4	Sala 5	Sala 6	Sala 7	Sala 8	Sala 9	Sala 10
									
Sala 11	Sala 12	Sala 13	Sala 14	Sala 15	Sala 16	Sala 17	Sala 18	Sala 19	Sala 20
Sala 21	Sala 22	Sala 23	Sala 24	Sala 25	Sala 26	Sala 27	Sala 28	Sala 29	Sala 30

Você tem um espaço de busca de 30 salas.

Quando você desistirá de procurá-lo?

Exemplo

- Achar o professor na Universidade para pedir nota no último dia do quadrimestre.

Sala 1	Sala 2	Sala 3	Sala 4	Sala 5	Sala 6	Sala 7	Sala 8	Sala 9	Sala 10
									
Sala 11	Sala 12	Sala 13	Sala 14	Sala 15	Sala 16	Sala 17	Sala 18	Sala 19	Sala 20
Sala 21	Sala 22	Sala 23	Sala 24	Sala 25	Sala 26	Sala 27	Sala 28	Sala 29	Sala 30

Você tem um espaço de busca de 30 salas.

Quando você desistirá de procurá-lo?

1. Quando encontrá-lo.
2. Quando já não ter mais salas onde procurar.

Exemplo

- Achar o professor na Universidade para pedir nota no último dia do quadrimestre.

Sala 1	Sala 2	Sala 3	Sala 4	Sala 5	Sala 6	Sala 7	Sala 8	Sala 9	Sala 10
									
Sala 11	Sala 12	Sala 13	Sala 14	Sala 15	Sala 16	Sala 17	Sala 18	Sala 19	Sala 20
Sala 21	Sala 22	Sala 23	Sala 24	Sala 25	Sala 26	Sala 27	Sala 28	Sala 29	Sala 30

Você tem um espaço de busca de 30 salas.

Quando você desistirá de procurá-lo?

1. Quando encontrá-lo.
2. Quando já não ter mais salas onde procurar.

Exemplo

- Achar o professor na Universidade para pedir nota no último dia do quadrimestre.

Sala 1	Sala 2	Sala 3	Sala 4	Sala 5	Sala 6	Sala 7	Sala 8	Sala 9	Sala 10
									
Sala 11	Sala 12	Sala 13	Sala 14	Sala 15	Sala 16	Sala 17	Sala 18	Sala 19	Sala 20
Sala 21	Sala 22	Sala 23	Sala 24	Sala 25	Sala 26	Sala 27	Sala 28	Sala 29	Sala 30
									

Você tem um espaço de busca de 30 salas.

Quando você desistirá de procurá-lo?

1. Quando encontrá-lo.
2. Quando já não ter mais salas onde procurar.



Você inicia sua busca.
Ele está na sala 30?

Exemplo

- Achar o professor na Universidade para pedir nota no último dia do quadrimestre.

Sala 1	Sala 2	Sala 3	Sala 4	Sala 5	Sala 6	Sala 7	Sala 8	Sala 9	Sala 10
									
Sala 11	Sala 12	Sala 13	Sala 14	Sala 15	Sala 16	Sala 17	Sala 18	Sala 19	Sala 20
Sala 21	Sala 22	Sala 23	Sala 24	Sala 25	Sala 26	Sala 27	Sala 28	Sala 29	Sala 30

Você tem um espaço de busca de 29 salas.

Quando você desistirá de procurá-lo?

1. Quando encontrá-lo.
2. Quando já não ter mais salas onde procurar.

Exemplo

- Achar o professor na Universidade para pedir nota no último dia do quadrimestre.

Sala 1	Sala 2	Sala 3	Sala 4	Sala 5	Sala 6	Sala 7	Sala 8	Sala 9	Sala 10
									
Sala 11	Sala 12	Sala 13	Sala 14	Sala 15	Sala 16	Sala 17	Sala 18	Sala 19	Sala 20
Sala 21	Sala 22	Sala 23	Sala 24	Sala 25	Sala 26	Sala 27	Sala 28	Sala 29	Sala 30
									

Você tem um espaço de busca de 29 salas.

Quando você desistirá de procurá-lo?

1. Quando encontrá-lo.
2. Quando já não ter mais salas onde procurar.



Você inicia sua busca.
Ele está na sala 29?

Exemplo

- Achar o professor na Universidade para pedir nota no último dia do quadrimestre.

Sala 1	Sala 2	Sala 3	Sala 4	Sala 5	Sala 6	Sala 7	Sala 8	Sala 9	Sala 10
									
Sala 11	Sala 12	Sala 13	Sala 14	Sala 15	Sala 16	Sala 17	Sala 18	Sala 19	Sala 20
Sala 21	Sala 22	Sala 23	Sala 24	Sala 25	Sala 26	Sala 27	Sala 28	Sala 29	Sala 30

Você tem um espaço de busca de 28 salas.

Quando você desistirá de procurá-lo?

1. Quando encontrá-lo.
2. Quando já não ter mais salas onde procurar.

Exemplo

- Achar o professor na Universidade para pedir nota no último dia do quadrimestre.

Sala 1	Sala 2	Sala 3	Sala 4	Sala 5	Sala 6	Sala 7	Sala 8	Sala 9	Sala 10
									
Sala 11	Sala 12	Sala 13	Sala 14	Sala 15	Sala 16	Sala 17	Sala 18	Sala 19	Sala 20
Sala 21	Sala 22	Sala 23	Sala 24	Sala 25	Sala 26	Sala 27	Sala 28	Sala 29	Sala 30
									

Você tem um espaço de busca de **28 salas**



Quando você desistirá de procurá-lo?

- Quando encontrá-lo.
- Quando já não ter mais salas onde procurar.

Você inicia sua busca.
Ele está na sala 28?

Exemplo

- Achar o professor na Universidade para pedir nota no último dia do quadrimestre.

Sala 1	Sala 2	Sala 3	Sala 4	Sala 5	Sala 6	Sala 7	Sala 8	Sala 9	Sala 10
									
Sala 11	Sala 12	Sala 13	Sala 14	Sala 15	Sala 16	Sala 17	Sala 18	Sala 19	Sala 20
Sala 21	Sala 22	Sala 23	Sala 24	Sala 25	Sala 26	Sala 27	Sala 28	Sala 29	Sala 30

Você tem um espaço de busca de 27 salas.

Quando você desistirá de procurá-lo?

1. Quando encontrá-lo.
2. Quando já não ter mais salas onde procurar.

Exemplo

- Você só para de abrir as portas das salas e procurar seu professor (chamar a busca recursivamente).
 - Quando encontrá-lo em alguma sala

Exemplo

Você inicia sua busca.
Ele está na sala 8?

- Achar o professor na Universidade para pedir nota no último dia do quadrimestre.



Sala 1	Sala 2	Sala 3	Sala 4	Sala 5	Sala 6	Sala 7	Sala 8	Sala 9	Sala 10
									
Sala 11	Sala 12	Sala 13	Sala 14	Sala 15	Sala 16	Sala 17	Sala 18	Sala 19	Sala 20
Sala 21	Sala 22	Sala 23	Sala 24	Sala 25	Sala 26	Sala 27	Sala 28	Sala 29	Sala 30

Você tem um espaço de busca de 8 salas.

Quando você desistirá de procurá-lo?

1. Quando encontrá-lo.
2. Quando já não ter mais salas onde procurar.

Exemplo

- Achar o professor na Universidade para pedir nota no último dia do quadrimestre.

Sala 1	Sala 2	Sala 3	Sala 4	Sala 5	Sala 6	Sala 7	Sala 8	Sala 9	Sala 10
									
Sala 11	Sala 12	Sala 13	Sala 14	Sala 15	Sala 16	Sala 17	Sala 18	Sala 19	Sala 20
Sala 21	Sala 22	Sala 23	Sala 24	Sala 25	Sala 26	Sala 27	Sala 28	Sala 29	Sala 30

Você tem um espaço de busca de 7 salas.

Quando você desistirá de procurá-lo?

1. Quando encontrá-lo.
2. Quando já não ter mais salas onde procurar.

Exemplo

Você inicia sua busca.
Ele está na sala 7?

- Achar o professor na Universidade para pedir nota no último dia do quadrimestre.



Sala 1	Sala 2	Sala 3	Sala 4	Sala 5	Sala 6	Sala 7	Sala 8	Sala 9	Sala 10
									
Sala 11	Sala 12	Sala 13	Sala 14	Sala 15	Sala 16	Sala 17	Sala 18	Sala 19	Sala 20
Sala 21	Sala 22	Sala 23	Sala 24	Sala 25	Sala 26	Sala 27	Sala 28	Sala 29	Sala 30

Você tem um espaço de busca de 7 salas.

Quando você desistirá de procurá-lo?

1. Quando encontrá-lo.
2. Quando já não ter mais salas onde procurar.

Exemplo

**Sua busca terminou!
Retorne sala 7 ...**

- Achar o professor na Universidade para pedir nota no último dia do quadrimestre.

Sala 1	Sala 2	Sala 3	Sala 4	Sala 5	Sala 6	Sala 7	Sala 8	Sala 9	Sala 10
									
Sala 11	Sala 12	Sala 13	Sala 14	Sala 15	Sala 16	Sala 17	Sala 18	Sala 19	Sala 20
Sala 21	Sala 22	Sala 23	Sala 24	Sala 25	Sala 26	Sala 27	Sala 28	Sala 29	Sala 30

Você tem um espaço de busca de 7 salas.

Quando você desistirá de procurá-lo?

1. Quando encontrá-lo.
2. Quando já não ter mais salas onde procurar.

Exemplo

- Achar o professor na Universidade para pedir nota no último dia do quadrimestre.



Exemplo

- Você só para de abrir as portas das salas e procurar seu professor (chamar a busca recursivamente).
 - Quando encontrá-lo em alguma sala
 - Ou quando não tiver mais salas para buscá-lo.

Você inicia sua busca.
Ele está na sala 2?

Exemplo

- Achar o professor na Universidade para pedir nota no último dia do quadrimestre.



Sala 1	Sala 2	Sala 3	Sala 4	Sala 5	Sala 6	Sala 7	Sala 8	Sala 9	Sala 10
									
Sala 11	Sala 12	Sala 13	Sala 14	Sala 15	Sala 16	Sala 17	Sala 18	Sala 19	Sala 20
Sala 21	Sala 22	Sala 23	Sala 24	Sala 25	Sala 26	Sala 27	Sala 28	Sala 29	Sala 30

Você tem um espaço de busca de 2 salas.

Quando você desistirá de procurá-lo?

1. Quando encontrá-lo.
2. Quando já não ter mais salas onde procurar.

Você inicia sua busca.
Ele está na sala 1?

Exemplo

- Adotar o professor na Universidade para pedir nota no último dia do quadrimestre.



Sala 1	Sala 2	Sala 3	Sala 4	Sala 5	Sala 6	Sala 7	Sala 8	Sala 9	Sala 10
									
Sala 11	Sala 12	Sala 13	Sala 14	Sala 15	Sala 16	Sala 17	Sala 18	Sala 19	Sala 20
Sala 21	Sala 22	Sala 23	Sala 24	Sala 25	Sala 26	Sala 27	Sala 28	Sala 29	Sala 30

Você tem um espaço de busca de 1 sala.

Quando você desistirá de procurá-lo?

1. Quando encontrá-lo.
2. Quando já não ter mais salas onde procurar.

Exemplo

- Achar o professor na Universidade para pedir nota no último dia do quadrimestre.

Sala 1	Sala 2	Sala 3	Sala 4	Sala 5	Sala 6	Sala 7	Sala 8	Sala 9	Sala 10
Sala 11	Sala 12	Sala 13	Sala 14	Sala 15	Sala 16	Sala 17	Sala 18	Sala 19	Sala 20
Sala 21	Sala 22	Sala 23	Sala 24	Sala 25	Sala 26	Sala 27	Sala 28	Sala 29	Sala 30

Você tem um espaço de busca de 0 salas.

Quando você desistirá de procurá-lo?

1. Quando encontrá-lo.
2. Quando já não ter mais salas onde procurar.

Exemplo

- Achar o professor na Universidade para pedir nota no último dia do quadrimestre.



Sala 1	Sala 2	Sala 3	Sala 4	Sala 5	Sala 6	Sala 7	Sala 8	Sala 9	Sala 10
Sala 11	Sala 12	Sala 13	Sala 14	Sala 15	Sala 16	Sala 17	Sala 18	Sala 19	Sala 20
Sala 21	Sala 22	Sala 23	Sala 24	Sala 25	Sala 26	Sala 27	Sala 28	Sala 29	Sala 30

Você tem um espaço de busca de 0 salas.

Quando você desistirá de procurá-lo?

- Quando encontrá-lo.
- Quando já não ter mais salas onde procurar.

Sua busca terminou!
Retorne -1 ...

Busca Linear Recursiva

- Algoritmo em C

```
1. int busca(int *v, int n, int x)
2. {
3.     if(n==0 || v[n-1] == x)
4.         return n-1;
5.     else
6.         return busca(v, n-1, x);
7. }
```

Complexidade de Algoritmos

- E qual a complexidade da Busca Linear Recursiva?
 - Qual o esforço computacional no pior caso?

Complexidade de Algoritmos

- E qual a complexidade da Busca Linear Recursiva?
 - Qual o esforço computacional no pior caso?
 - Linear
 - Portanto: **$O(n)$**

Complexidade de Algoritmos

- Existe algum algoritmo mais eficiente para se realizar a busca?
 - Sim. Entretanto, a lista de elementos deve estar ordenada.

Busca Binária

- Suponha que a lista de elementos esteja ordenada pelo campo ao qual você deseja realizar a busca.

0	1	2	3	4	5	6	7	8	9
43	65	69	76	85	89	93	95	99	107

- Na busca linear iterativa, quantas comparações são necessárias para encontrar o elemento 95?

Busca Binária

- Suponha que a lista de elementos esteja ordenada pelo campo ao qual você deseja realizar a busca.

0	1	2	3	4	5	6	7	8	9
43	65	69	76	85	89	93	95	99	107

- Na busca linear iterativa, quantas comparações são necessárias para encontrar o elemento 95?
- A Busca Binária faz isso com 2 comparações.

Busca Binária

- Funcionamento

0	1	2	3	4	5	6	7	8	9
43	65	69	76	85	89	93	95	99	107

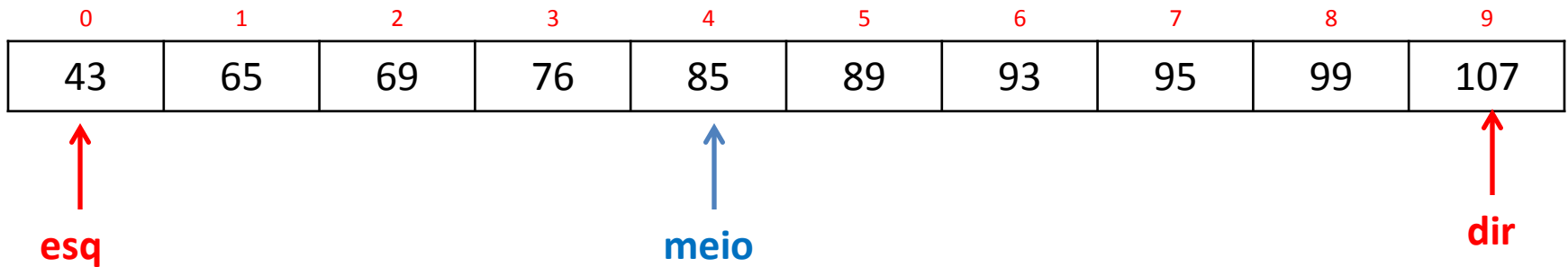
↑ esq

↑ dir

$$1. \text{ meio} = \lfloor (\text{esq} + \text{dir}) / 2 \rfloor$$

Busca Binária

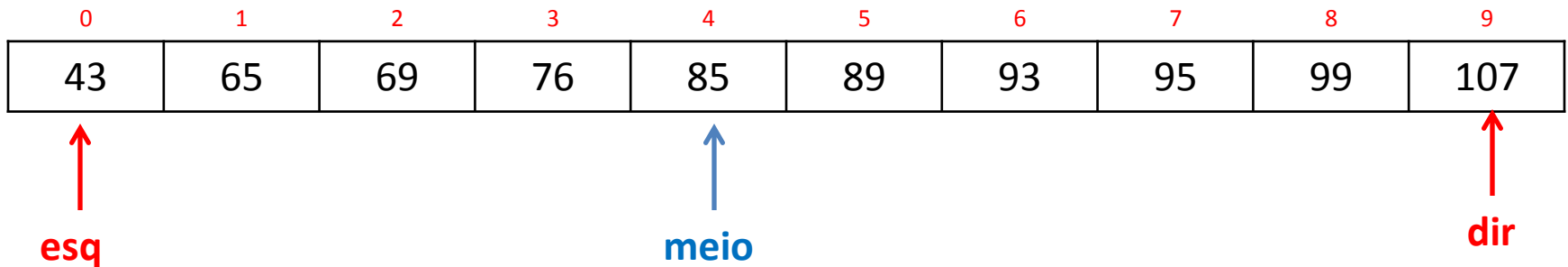
- Funcionamento



1. $\text{meio} = \lfloor (\text{esq} + \text{dir}) / 2 \rfloor$
2. É o 95?

Busca Binária

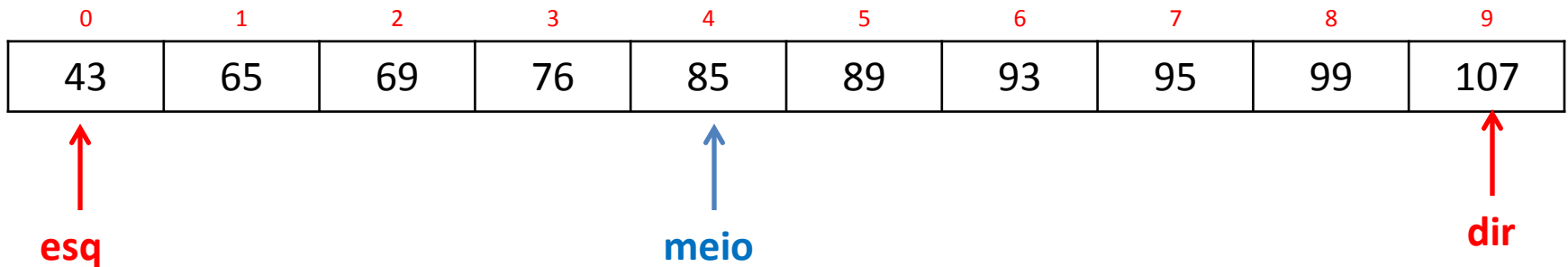
- Funcionamento



1. $\text{meio} = \lfloor (\text{esq} + \text{dir}) / 2 \rfloor$
2. É o 95?
3. Elemento na posição meio é > 95 ?

Busca Binária

- Funcionamento



1. $\text{meio} = \lfloor (\text{esq} + \text{dir}) / 2 \rfloor$
2. É o 95?
3. Elemento na posição meio é > 95 ?
 - 3.1. Não: Então: $\text{esq} = \text{meio} + 1$

Busca Binária

- Funcionamento

0	1	2	3	4	5	6	7	8	9
43	65	69	76	85	89	93	95	99	107

Diagram illustrating the binary search process on a sorted array. The array contains the values: 43, 65, 69, 76, 85, 89, 93, 95, 99, 107. The indices are 0 to 9. The current search range is from index 4 (85) to index 9 (107). The middle element is at index 4 (85), labeled 'meio'. The left boundary is at index 5 (89), labeled 'esq'. The right boundary is at index 9 (107), labeled 'dir'.

1. $\text{meio} = \lfloor (\text{esq} + \text{dir}) / 2 \rfloor$
2. É o 95?
3. Elemento na posição meio é > 95 ?
 - 3.1. Não: Então: $\text{esq} = \text{meio} + 1$

Busca Binária

- Funcionamento

0	1	2	3	4	5	6	7	8	9
43	65	69	76	85	89	93	95	99	107

Diagram illustrating the binary search process on a sorted array. The array contains the values: 43, 65, 69, 76, 85, 89, 93, 95, 99, 107. The indices are 0 through 9. The current search range is from index 4 (85) to index 9 (107). The middle element (meio) is at index 4 (85). The left pointer (esq) is at index 5 (89). The right pointer (dir) is at index 9 (107).

1. $\text{meio} = \lfloor (\text{esq} + \text{dir}) / 2 \rfloor$
2. É o 95?
3. Elemento na posição meio é > 95 ?
- 3.1. Não: Então: $\text{esq} = \text{meio} + 1$. Volte ao passo 1.

Busca Binária

- Funcionamento

0	1	2	3	4	5	6	7	8	9
43	65	69	76	85	89	93	95	99	107

Diagram illustrating the binary search process on a sorted array. The array contains the values: 43, 65, 69, 76, 85, 89, 93, 95, 99, 107. The indices 0 through 9 are shown above the corresponding elements. The current search range is defined by 'esq' (left) at index 5 (value 89) and 'dir' (right) at index 9 (value 107). The 'meio' (middle) element is at index 7 (value 95), indicated by a blue arrow. Red arrows point to the 'esq' and 'dir' positions.

1. $\text{meio} = \lfloor (\text{esq} + \text{dir}) / 2 \rfloor$

2. É o 95?

3. Elemento na posição meio é > 95 ?

3.1. Não: Então: $\text{esq} = \text{meio} + 1$. Volte ao passo 1.

Busca Binária

- Pior caso:

0	1	2	3	4	5	6	7	8	9
43	65	69	76	85	89	93	95	99	107

- Na busca linear iterativa, quantas comparações são necessárias para determinar que o elemento 74 não está na lista?

Busca Binária

- Pior caso:

0	1	2	3	4	5	6	7	8	9
43	65	69	76	85	89	93	95	99	107

- Na busca linear iterativa, quantas comparações são necessárias para determinar que o elemento 74 não está na lista?
- A busca binária faz isso com **4 comparações**.

Busca Binária

- Funcionamento

0	1	2	3	4	5	6	7	8	9
43	65	69	76	85	89	93	95	99	107

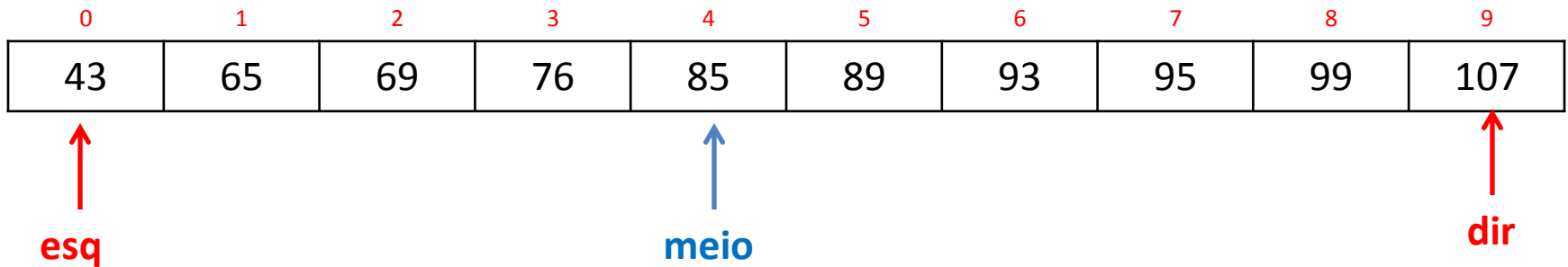
↑ esq

↑ dir

$$1. \text{ meio} = \lfloor (\text{esq} + \text{dir}) / 2 \rfloor$$

Busca Binária

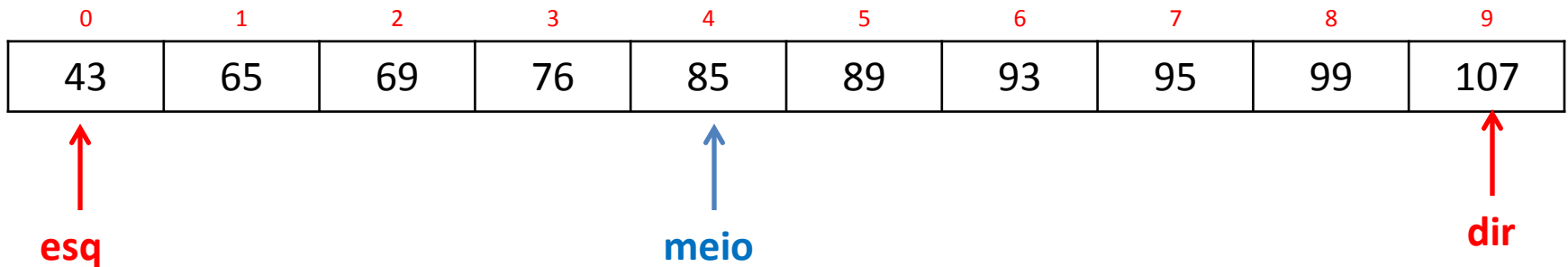
- Funcionamento



1. $\text{meio} = \lfloor (\text{esq} + \text{dir}) / 2 \rfloor$
2. É o 74?

Busca Binária

- Funcionamento



1. $\text{meio} = \lfloor (\text{esq} + \text{dir}) / 2 \rfloor$
2. É o 74?
3. Elemento na posição meio é > 74 ?

Busca Binária

- Funcionamento

0	1	2	3	4	5	6	7	8	9
43	65	69	76	85	89	93	95	99	107
↑				↑					↑
esq				meio					dir

1. $\text{meio} = \lfloor (\text{esq} + \text{dir}) / 2 \rfloor$
2. É o 74?
3. Elemento na posição meio é > 74 ?
 - 3.1. Não: Então: $\text{esq} = \text{meio} + 1$
 - 3.2. Sim: Então: $\text{dir} = \text{meio} - 1$

Busca Binária

- Funcionamento

0	1	2	3	4	5	6	7	8	9
43	65	69	76	85	89	93	95	99	107
↑			↑	↑					
esq			dir	meio					

1. $\text{meio} = \lfloor (\text{esq} + \text{dir}) / 2 \rfloor$

2. É o 74?

3. Elemento na posição meio é > 74 ?

3.1. Não: Então: $\text{esq} = \text{meio} + 1$. Volte ao passo 1.

3.2. Sim: Então: $\text{dir} = \text{meio} - 1$. Volte ao passo 1.

Busca Binária

- Funcionamento

0	1	2	3	4	5	6	7	8	9
43	65	69	76	85	89	93	95	99	107
↑	↑		↑						
esq	meio		dir						

1. $\text{meio} = \lfloor (\text{esq} + \text{dir}) / 2 \rfloor$

2. É o 74?

3. Elemento na posição meio é > 74 ?

3.1. Não: Então: $\text{esq} = \text{meio} + 1$. Volte ao passo 1.

3.2. Sim: Então: $\text{dir} = \text{meio} - 1$. Volte ao passo 1.

Busca Binária

- Funcionamento

0	1	2	3	4	5	6	7	8	9
43	65	69	76	85	89	93	95	99	107

Diagram illustrating the binary search process on a sorted array. The array contains the values: 43, 65, 69, 76, 85, 89, 93, 95, 99, 107. The indices are 0 through 9. The current search range is from index 1 (65) to index 3 (76). The middle element (meio) is at index 1 (65). The search range is updated to the right (esq) at index 2 (69) and the right boundary (dir) remains at index 3 (76).

1. $\text{meio} = \lfloor (\text{esq} + \text{dir}) / 2 \rfloor$

2. É o 74?

3. Elemento na posição meio é > 74 ?

3.1. Não: Então: $\text{esq} = \text{meio} + 1$. Volte ao passo 1.

3.2. Sim: Então: $\text{dir} = \text{meio} - 1$. Volte ao passo 1.

Busca Binária

- Funcionamento

0	1	2	3	4	5	6	7	8	9
43	65	69	76	85	89	93	95	99	107

↑ ↑ ↑
esq meio dir

1. $\text{meio} = \lfloor (\text{esq} + \text{dir}) / 2 \rfloor$

2. É o 74?

3. Elemento na posição meio é > 74 ?

3.1. Não: Então: $\text{esq} = \text{meio} + 1$. Volte ao passo 1.

3.2. Sim: Então: $\text{dir} = \text{meio} - 1$. Volte ao passo 1.

Busca Binária

- Funcionamento

0	1	2	3	4	5	6	7	8	9
43	65	69	76	85	89	93	95	99	107

Diagram illustrating the binary search process on a sorted array. The array contains the values: 43, 65, 69, 76, 85, 89, 93, 95, 99, 107. The indices are 0 through 9. The current search range is defined by 'esq' (left) at index 2 and 'dir' (right) at index 3. The middle element 'meio' is at index 2 (value 69).

1. $\text{meio} = \lfloor (\text{esq} + \text{dir}) / 2 \rfloor$

2. É o 74?

3. Elemento na posição meio é > 74 ?

3.1. Não: Então: $\text{esq} = \text{meio} + 1$. Volte ao passo 1.

3.2. Sim: Então: $\text{dir} = \text{meio} - 1$. Volte ao passo 1.

Busca Binária

- Funcionamento

0	1	2	3	4	5	6	7	8	9
43	65	69	76	85	89	93	95	99	107

Diagram illustrating the binary search process on a sorted array. The array contains the values [43, 65, 69, 76, 85, 89, 93, 95, 99, 107]. The current search range is from index 2 (value 69) to index 4 (value 85). The middle element is at index 3 (value 76). The labels 'esq' (left), 'meio' (middle), and 'dir' (right) are shown below the array, with arrows pointing to their respective indices.

1. $\text{meio} = \lfloor (\text{esq} + \text{dir}) / 2 \rfloor$

2. É o 74?

3. Elemento na posição meio é > 74 ?

3.1. Não: Então: $\text{esq} = \text{meio} + 1$. Volte ao passo 1.

3.2. Sim: Então: $\text{dir} = \text{meio} - 1$. Volte ao passo 1.

Busca Binária

- Funcionamento

0	1	2	3	4	5	6	7	8	9
43	65	69	76	85	89	93	95	99	107

↑ ↑ ↑
dir esq meio

1. $\text{meio} = \lfloor (\text{esq} + \text{dir}) / 2 \rfloor$

2. É o 74?

3. Elemento na posição meio é > 74 ?

3.1. Não: Então: $\text{esq} = \text{meio} + 1$. Volte ao passo 1.

3.2. Sim: Então: $\text{dir} = \text{meio} - 1$. Volte ao passo 1.

Busca Binária

- Funcionamento

0	1	2	3	4	5	6	7	8	9
43	65	69	76	85	89	93	95	99	107

↑ ↑ ↑
dir esq meio

esq ficou maior que dir.

Portanto a busca terminou sem encontrar o 74.

Deste modo, antes do passo 1. deve-se sempre verificar se $esq \leq dir$

1. $meio = \lfloor (esq + dir) / 2 \rfloor$

2. É o 74?

3. Elemento na posição meio é > 74 ?

3.1. Não: Então: $esq = meio + 1$. Volte ao passo 1.

3.2. Sim: Então: $dir = meio - 1$. Volte ao passo 1.

Busca Binária

- Algoritmo em C

Busca Binária

- Algoritmo em C

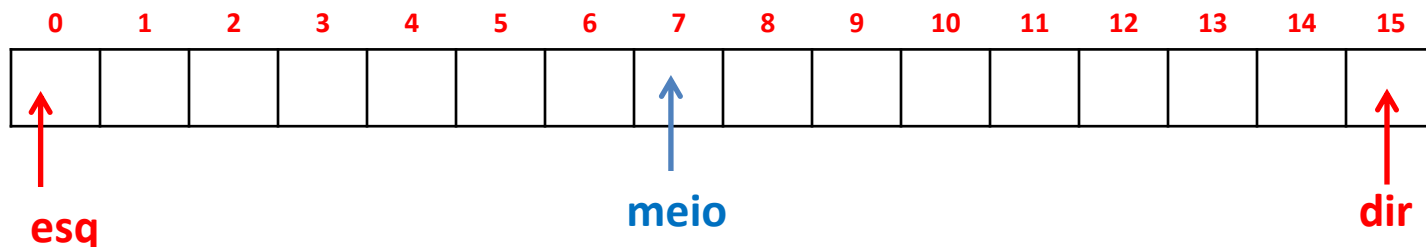
```
01. int buscaBin(int *v, int n, int x)
02. {
03.     int esq=0, dir=n-1, meio;
04.     while(esq<=dir)
05.     {
06.         meio = (esq+dir)/2;
07.         if(v[meio] == x)
08.         {
09.             return meio;
10.         }
11.         else if(v[meio] > x)
12.         {
13.             dir = meio-1;
14.         }
15.         else
16.         {
17.             esq = meio + 1;
18.         }
19.     }
20.     return -1;
21. }
```


Análise de Algoritmos

- No pior caso (elemento não existir) qual o esforço computacional da Busca Binária?
 - Quantas comparações a Busca Binária fará?

Análise de Algoritmos

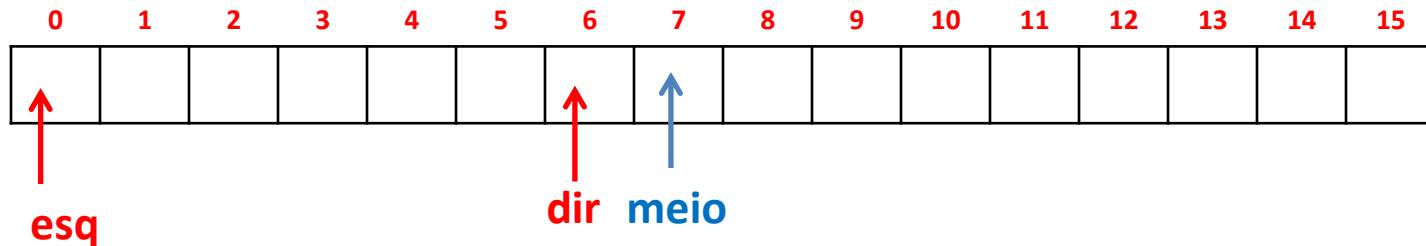
- No pior caso (elemento não existir) qual o esforço computacional da Busca Binária?
 - Quantas comparações a Busca Binária fará?
 - Perceba que a cada iteração o vetor é dividido pela metade.



- Tamanho do Espaço de Busca: **16**
- Comparações: **0**

Análise de Algoritmos

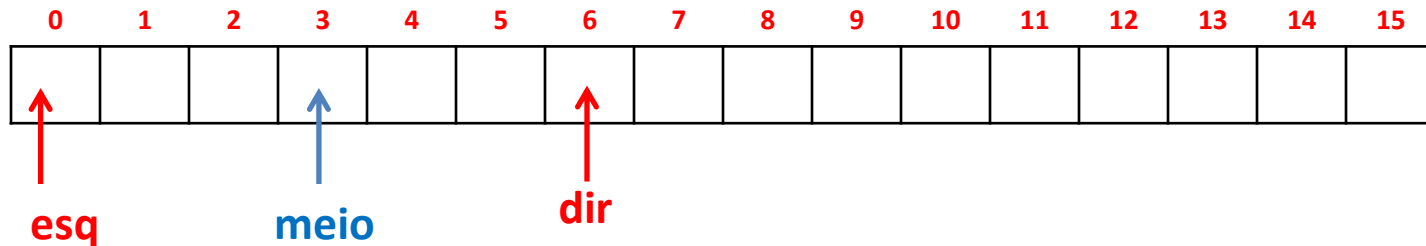
- No pior caso (elemento não existir) qual o esforço computacional da Busca Binária?
 - Quantas comparações a Busca Binária fará?
 - Perceba que a cada iteração o vetor é dividido pela metade.



- Tamanho do Espaço de Busca: ~~16~~ 7
- Comparações: ~~0~~ 1

Análise de Algoritmos

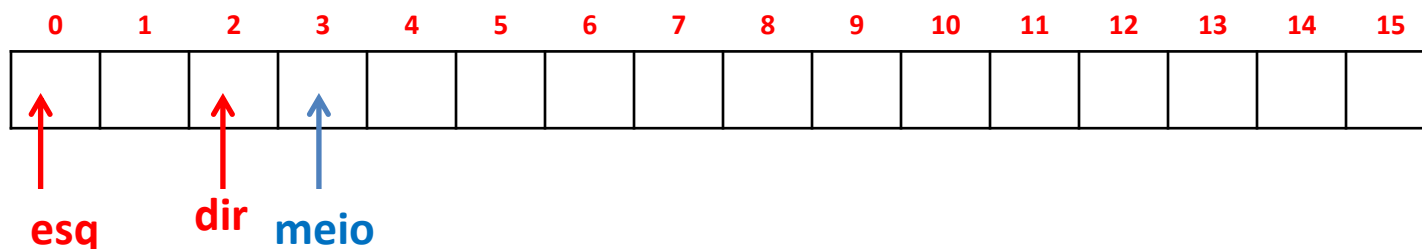
- No pior caso (elemento não existir) qual o esforço computacional da Busca Binária?
 - Quantas comparações a Busca Binária fará?
 - Perceba que a cada iteração o vetor é dividido pela metade.



- Tamanho do Espaço de Busca: ~~16~~ 7
- Comparações: ~~0~~ 1

Análise de Algoritmos

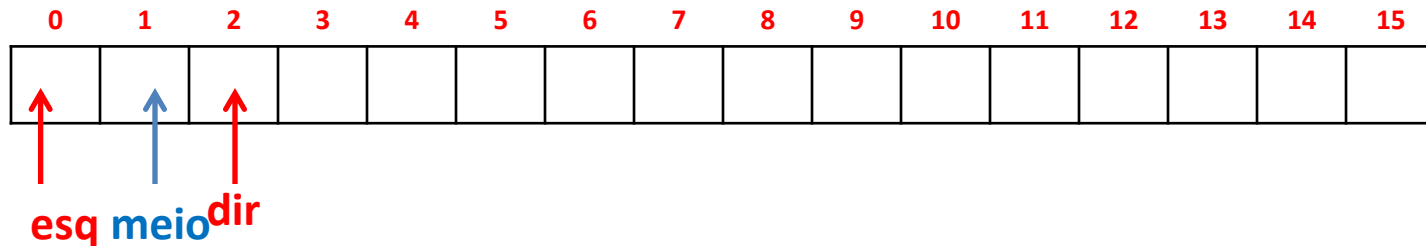
- No pior caso (elemento não existir) qual o esforço computacional da Busca Binária?
 - Quantas comparações a Busca Binária fará?
 - Perceba que a cada iteração o vetor é dividido pela metade.



- Tamanho do Espaço de Busca: ~~16~~ 7 3
- Comparações: ~~0~~ 1 2

Análise de Algoritmos

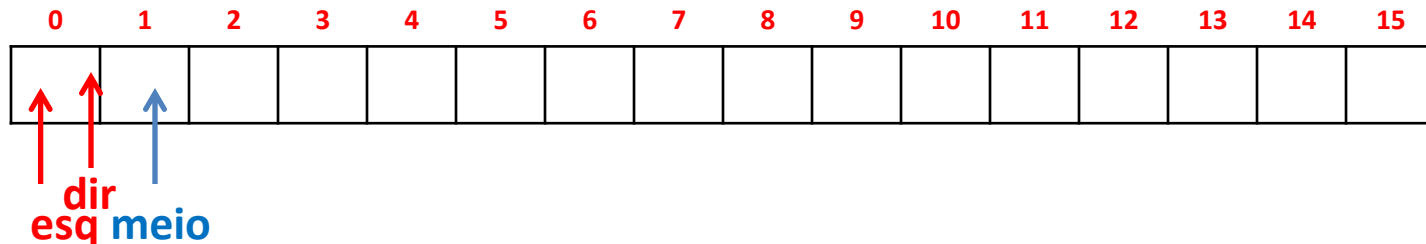
- No pior caso (elemento não existir) qual o esforço computacional da Busca Binária?
 - Quantas comparações a Busca Binária fará?
 - Perceba que a cada iteração o vetor é dividido pela metade.



- Tamanho do Espaço de Busca: ~~16~~ 7 3
- Comparações: ~~0~~ 1 2

Análise de Algoritmos

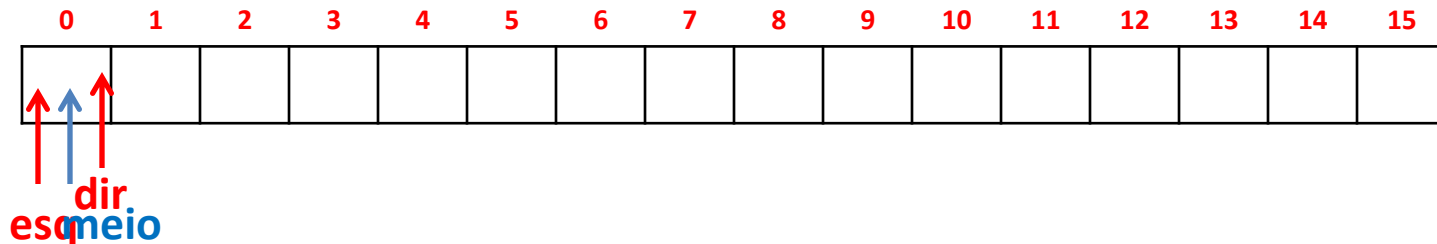
- No pior caso (elemento não existir) qual o esforço computacional da Busca Binária?
 - Quantas comparações a Busca Binária fará?
 - Perceba que a cada iteração o vetor é dividido pela metade.



- Tamanho do Espaço de Busca: ~~16~~ **7 3 1**
- Comparações: ~~0~~ **1 2 3**

Análise de Algoritmos

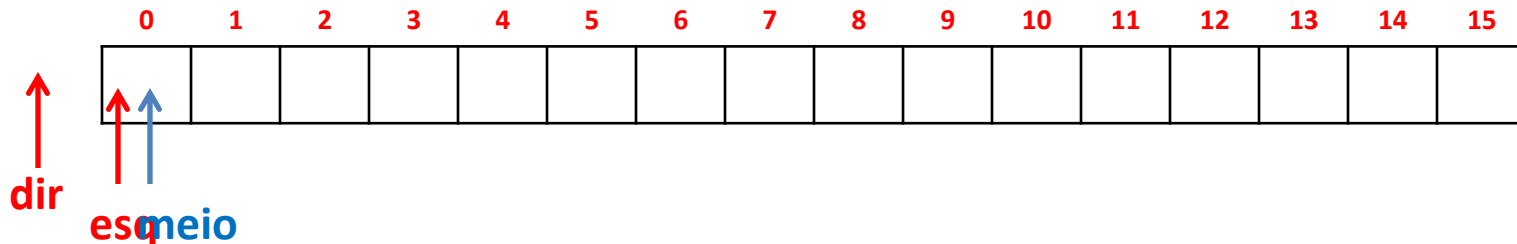
- No pior caso (elemento não existir) qual o esforço computacional da Busca Binária?
 - Quantas comparações a Busca Binária fará?
 - Perceba que a cada iteração o vetor é dividido pela metade.



- Tamanho do Espaço de Busca: ~~16~~ **7 3 1**
- Comparações: ~~0~~ **1 2 3**

Análise de Algoritmos

- No pior caso (elemento não existir) qual o esforço computacional da Busca Binária?
 - Quantas comparações a Busca Binária fará?
 - Perceba que a cada iteração o vetor é dividido pela metade.



- Tamanho do Espaço de Busca: ~~16~~ **7 3 1 0**
- Comparações: ~~0~~ **1 2 3 4**

Análise de Algoritmos

- Pensem...   

Tamanho	Comparações (Acumuladas)
16	1
7	+1 (2)
3	+1 (3)
1	+1 (4)
0	

Análise de Algoritmos

- Pensem...   

Tamanho	Comparações (Acumuladas)
2^4	1
2^3-1	2
2^2-1	3
2^1-1	4
2^0-1	

Análise de Algoritmos

- Pensem...



Tamanho	Comparações (Acumuladas)
2^4	1
2^3-1	2
2^2-1	3
2^1-1	4
2^0-1	

- Sempre divido o vetor pela metade.
 - Quantas vezes consigo dividi-lo pela metade?

Análise de Algoritmos

- Pensem...   

Tamanho	Comparações (Acumuladas)
2^4	1
2^3-1	2
2^2-1	3
2^1-1	4
2^0-1	

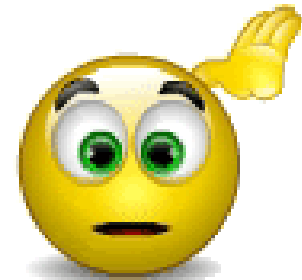
- Sempre divido o vetor pela metade.
 - Quantas vezes consigo dividi-lo pela metade?
Quantas vezes consigo dividir um vetor de tamanho 16 pela metade?

Análise de Algoritmos

- Pensem...



Tamanho	Comparações (Acumuladas)
2^4	1
2^3-1	2
2^2-1	3
2^1-1	4
2^0-1	



- Sempre divido o vetor pela metade.
 - Quantas vezes consigo dividi-lo pela metade?
Quantas vezes consigo dividir um vetor de tamanho 16 pela metade?

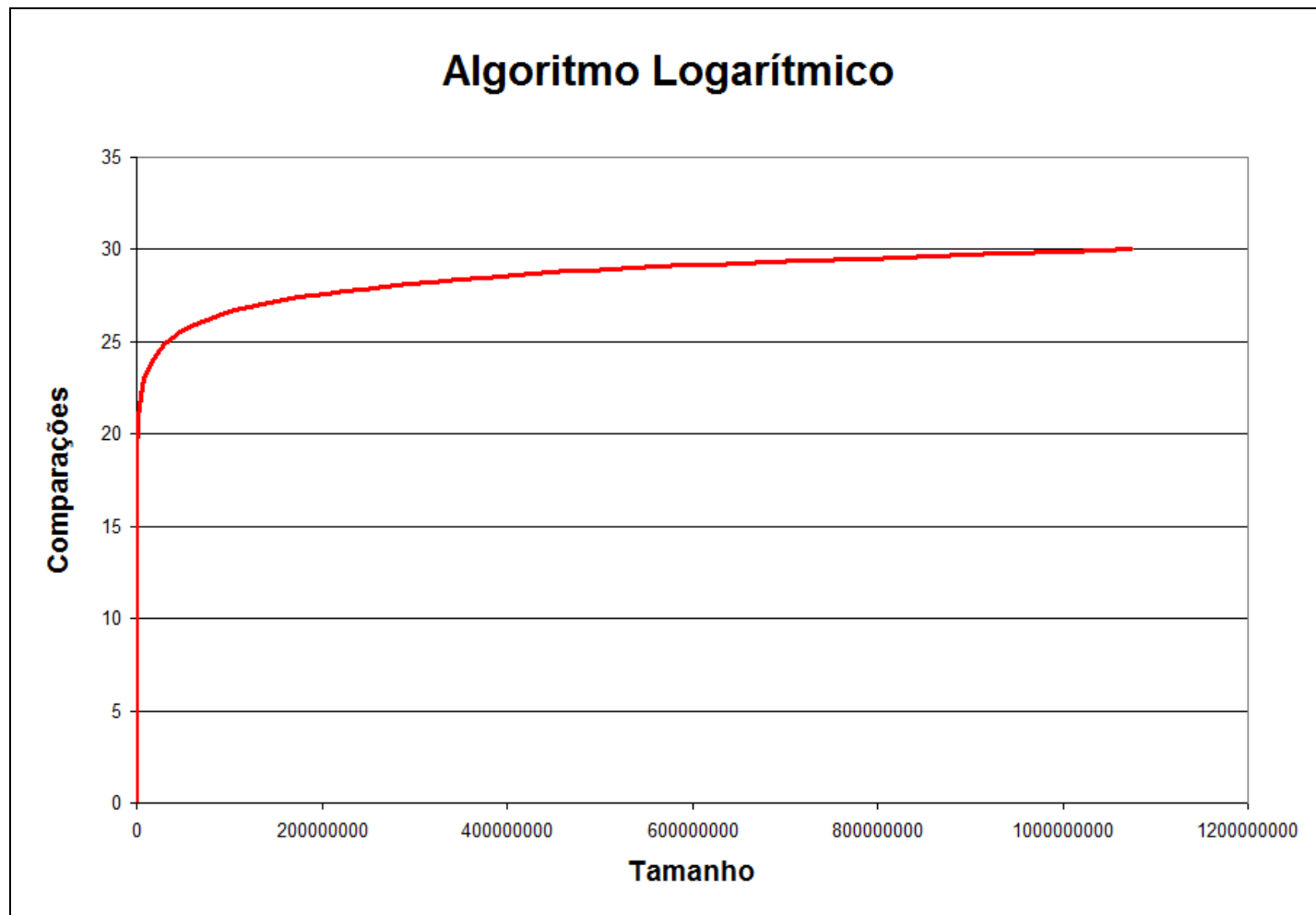
Análise de Algoritmos

- Pior Caso: Busca Binária
 - No pior caso são necessárias **$\log_2 n$** comparações.
 - Portanto a Busca Binária é um algoritmo **logarítmico** ou **$O(\log_2 n)$** .

Análise de Algoritmos

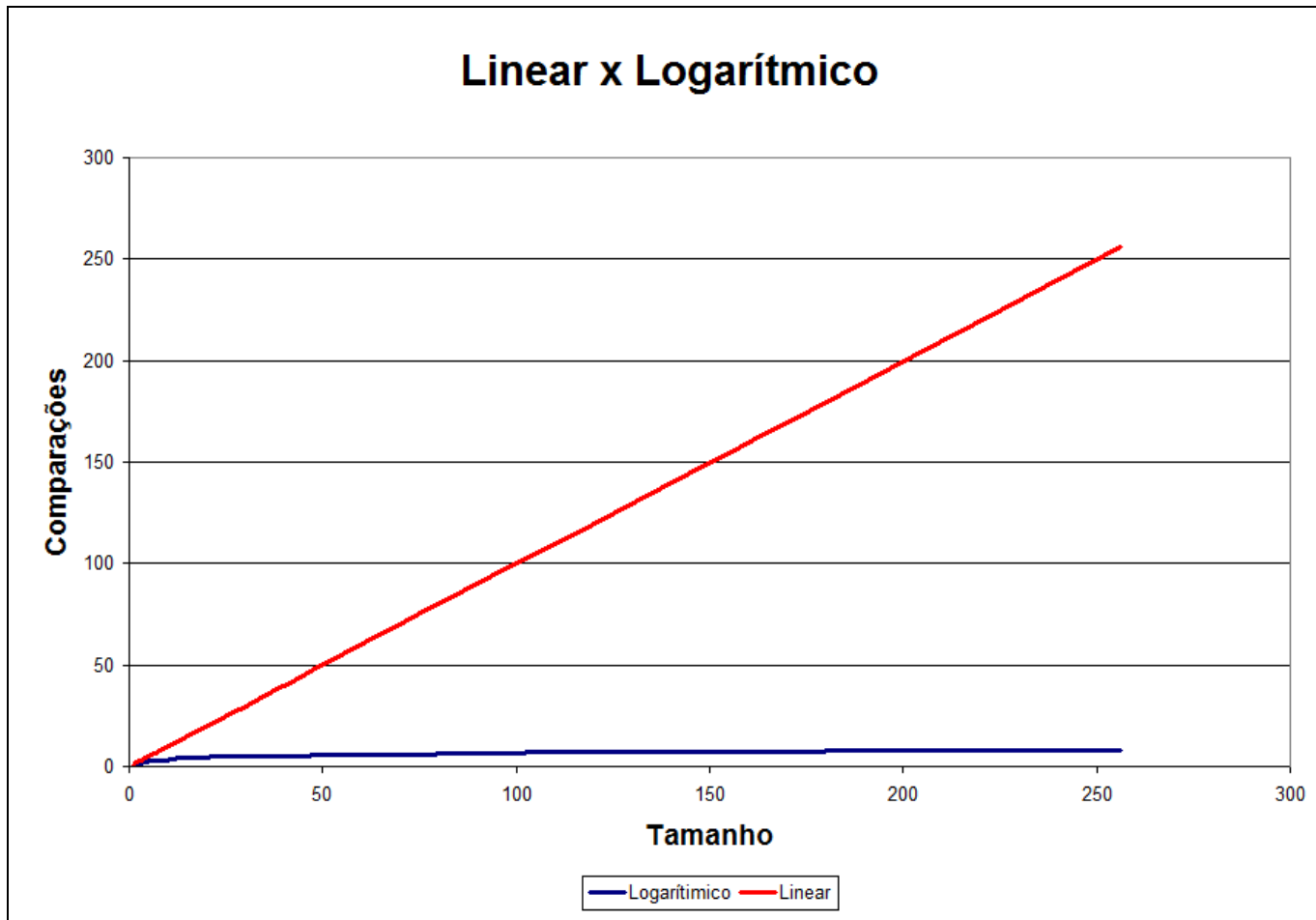
- Pior Caso: Busca Binária

$O(\log_2 n)$



Análise de Algoritmos

- Algoritmo Linear x Algoritmo Logarítmico



Busca Binária Recursiva

- Implementação Recursiva:

Busca Binária Recursiva

- Implementação Recursiva:

```
01. int buscaBin(int *v, int esq, int dir, int x)
02. {
03.     int meio;
04.     if(esq <= dir)
05.     {
06.         meio = (esq+dir)/2;
07.         if(v[meio] == x)
08.         {
09.             return meio;
10.         }
11.         else if(v[meio] > x)
12.         {
13.             return buscaBin(v, esq, meio-1, x);
14.         }
15.         else
16.         {
17.             return buscaBin(v, meio+1, dir, x);
18.         }
19.     }
20.     return -1;
21. }
```