

4ª EDIÇÃO

Estruturas de Dados

Busca

Luiza
<CODE>

Ementa:

1. Definição do problema
 - a. O que é busca?
 - b. Exemplos
2. Busca sequencial
 - a. Qual a ideia do algoritmo?
 - b. Qual a eficiência do algoritmo?
3. Busca binária
 - a. Qual a ideia do algoritmo?
 - b. Qual a eficiência do algoritmo?

1. Definição do problema

1. Definição do problema

1.1 O que é busca?

Queremos encontrar um ou mais **itens** com **uma determinada propriedade** em uma grande **coleção** de itens.

1. Definição do problema

1.1 O que é busca?

Queremos encontrar um ou mais **itens** com **uma determinada propriedade** em uma grande **coleção** de itens (que pode ou não estar ordenada).

1. Definição do problema

1.1 O que é busca?

Queremos encontrar um ou mais **itens** com **uma determinada propriedade** em uma grande **coleção** de itens (que pode ou não estar ordenada).

Exemplos:

- Encontrar anotações que fiz sobre estruturas de dados no meu caderno
- Encontrar o número do telefone do meu primo chamado Bob numa lista telefônica
- Encontrar as figurinhas duplicadas da minha coleção da Copa do Mundo

1. Definição do problema

1.1 O que é busca?

Queremos encontrar um ou mais **itens** com **uma determinada propriedade** em uma grande **coleção** de itens (que pode ou não estar ordenada).

Exemplos:

- Encontrar **anotações** que fiz **sobre estruturas de dados** no meu **caderno**
- Encontrar **o número do telefone** do meu primo **chamado Bob** numa **lista telefônica**
- Encontrar **as figurinhas duplicadas** da minha **coleção da Copa do Mundo**

1. Definição do problema

1.2 Escopo da aula

Nessa aula vamos nos concentrar em resolver problemas de busca em listas de JavaScript.

2. Busca sequencial

2. Busca sequencial

2.1 Qual a ideia do algoritmo?

Percorrer a coleção, item por item, até que a propriedade procurada seja encontrada ou até que a coleção termine.

2. Busca sequencial

2.1 Qual a ideia do algoritmo?

Percorrer a coleção, item por item, até que a propriedade procurada seja encontrada ou até que a coleção termine.

Exemplo: encontrar a posição do número **1** na lista a seguir

| | | | | | | | | | | | | | |
|---|---|---|---|----|----|---|---|----|----|---|----|----|----|
| 5 | 8 | 3 | 4 | 12 | 33 | 1 | 6 | 98 | 21 | 1 | 45 | 11 | 23 |
|---|---|---|---|----|----|---|---|----|----|---|----|----|----|

2. Busca sequencial

2.1 Qual a ideia do algoritmo?

Percorrer a coleção, item por item, até que a propriedade procurada seja encontrada ou até que a coleção termine.

Exemplo: encontrar a posição do número **1** na lista a seguir

== 1?

| | | | | | | | | | | | | | |
|---|---|---|---|----|----|---|---|----|----|---|----|----|----|
| 5 | 8 | 3 | 4 | 12 | 33 | 1 | 6 | 98 | 21 | 1 | 45 | 11 | 23 |
|---|---|---|---|----|----|---|---|----|----|---|----|----|----|

2. Busca sequencial

2.1 Qual a ideia do algoritmo?

Percorrer a coleção, item por item, até que a propriedade procurada seja encontrada ou até que a coleção termine.

Exemplo: encontrar a posição do número **1** na lista a seguir

== 1?

| | | | | | | | | | | | | | |
|---|---|---|---|----|----|---|---|----|----|---|----|----|----|
| 5 | 8 | 3 | 4 | 12 | 33 | 1 | 6 | 98 | 21 | 1 | 45 | 11 | 23 |
|---|---|---|---|----|----|---|---|----|----|---|----|----|----|

2. Busca sequencial

2.1 Qual a ideia do algoritmo?

Percorrer a coleção, item por item, até que a propriedade procurada seja encontrada ou até que a coleção termine.

Exemplo: encontrar a posição do número **1** na lista a seguir

== 1?

| | | | | | | | | | | | | | |
|---|---|---|---|----|----|---|---|----|----|---|----|----|----|
| 5 | 8 | 3 | 4 | 12 | 33 | 1 | 6 | 98 | 21 | 1 | 45 | 11 | 23 |
|---|---|---|---|----|----|---|---|----|----|---|----|----|----|

2. Busca sequencial

2.1 Qual a ideia do algoritmo?

Percorrer a coleção, item por item, até que a propriedade procurada seja encontrada ou até que a coleção termine.

Exemplo: encontrar a posição do número **1** na lista a seguir

== 1?

| | | | | | | | | | | | | | |
|---|---|---|---|----|----|---|---|----|----|---|----|----|----|
| 5 | 8 | 3 | 4 | 12 | 33 | 1 | 6 | 98 | 21 | 1 | 45 | 11 | 23 |
|---|---|---|---|----|----|---|---|----|----|---|----|----|----|

2. Busca sequencial

2.1 Qual a ideia do algoritmo?

Percorrer a coleção, item por item, até que a propriedade procurada seja encontrada ou até que a coleção termine.

Exemplo: encontrar a posição do número **1** na lista a seguir

== 1?

| | | | | | | | | | | | | | |
|---|---|---|---|----|----|---|---|----|----|---|----|----|----|
| 5 | 8 | 3 | 4 | 12 | 33 | 1 | 6 | 98 | 21 | 1 | 45 | 11 | 23 |
|---|---|---|---|----|----|---|---|----|----|---|----|----|----|

2. Busca sequencial

2.1 Qual a ideia do algoritmo?

Percorrer a coleção, item por item, até que a propriedade procurada seja encontrada ou até que a coleção termine.

Exemplo: encontrar a posição do número **1** na lista a seguir

== 1?

| | | | | | | | | | | | | | |
|---|---|---|---|----|----|---|---|----|----|---|----|----|----|
| 5 | 8 | 3 | 4 | 12 | 33 | 1 | 6 | 98 | 21 | 1 | 45 | 11 | 23 |
|---|---|---|---|----|----|---|---|----|----|---|----|----|----|

2. Busca sequencial

2.1 Qual a ideia do algoritmo?

Percorrer a coleção, item por item, até que a propriedade procurada seja encontrada ou até que a coleção termine.

Exemplo: encontrar a posição do número **1** na lista a seguir

== 1?

| | | | | | | | | | | | | | |
|---|---|---|---|----|----|---|---|----|----|----|----|----|----|
| 5 | 8 | 3 | 4 | 12 | 33 | 1 | 6 | 98 | 21 | 1 | 45 | 11 | 23 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |

2. Busca sequencial

2.2 Qual a eficiência do algoritmo?

Para uma lista de tamanho **n**.

Qual o **melhor caso**?

2. Busca sequencial

2.2 Qual a eficiência do algoritmo?

Para uma lista de tamanho **n**.

Qual o **melhor caso**?

- Se o elemento procurado for o primeiro a ser testado. Assim fazemos apenas 1 operação: **$O(1)$**

2. Busca sequencial

2.2 Qual a eficiência do algoritmo?

Para uma lista de tamanho **n**.

Qual o **melhor caso**?

- Se o elemento procurado for o primeiro a ser testado. Assim fazemos apenas **1** operação: **$O(1)$**

Qual o **pior caso**?

- Se o elemento procurado for o último da lista ou não estiver presente nela. Assim fazemos **n** operações: **$O(n)$**

2. Busca sequencial

2.3 Implementação

```
1 function linearSearch(list, item) {  
2   for (var i = 0; i < list.length; i++) {  
3     if (list[i] == item) {  
4       return i;  
5     }  
6   }  
7   return null;  
8 }
```

3. Busca binária

3. Busca binária

3.1 Qual a ideia do algoritmo?

Assumindo que a coleção está ordenada, comparamos o item procurado com o item “ao meio” da coleção. Daí temos 3 possibilidades:

- a. O item “ao meio” é **igual** ao item procurado
 - i. finalizamos a busca.
- b. O item “ao meio” é **maior** que o item procurado
 - i. Como a lista está ordenada, o item procurado só pode estar na metade esquerda da lista.
 - ii. Assim, aplicamos o mesmo processo recursivamente à sublista da esquerda.
- c. O item “ao meio” é **menor** que o item procurado
 - i. Como a lista está ordenada, o item procurado só pode estar na metade direita da lista.
 - ii. Assim, aplicamos o mesmo processo recursivamente à sublista da direita.

3. Busca binária

3.1 Qual a ideia do algoritmo?

Exemplo: encontrar a posição do número **8** na lista ordenada a seguir:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| 1 | 1 | 3 | 4 | 5 | 6 | 8 | 11 | 12 | 21 | 23 | 33 | 45 | 98 |
|---|---|---|---|---|---|---|----|----|----|----|----|----|----|

3. Busca binária

3.1 Qual a ideia do algoritmo?

Exemplo: encontrar a posição do número **8** na lista ordenada a seguir:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| 1 | 1 | 3 | 4 | 5 | 6 | 8 | 11 | 12 | 21 | 23 | 33 | 45 | 98 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |

Etapas: 1

Posição do meio: $0 + [(13 - 0) + 1] / 2 = 0 + 7 = 7$

3. Busca binária

3.1 Qual a ideia do algoritmo?

Exemplo: encontrar a posição do número **8** na lista ordenada a seguir:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| 1 | 1 | 3 | 4 | 5 | 6 | 8 | 11 | 12 | 21 | 23 | 33 | 45 | 98 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |

Etapas: 1


Posição do meio: $0 + [(13 - 0) + 1] / 2 = 0 + 7 = 7$

3. Busca binária

3.1 Qual a ideia do algoritmo?

Exemplo: encontrar a posição do número **8** na lista ordenada a seguir:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| 1 | 1 | 3 | 4 | 5 | 6 | 8 | 11 | 12 | 21 | 23 | 33 | 45 | 98 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |



Etapa: 1

Posição do meio: $0 + [(13 - 0) + 1] / 2 = 0 + 7 = 7$

3. Busca binária

3.1 Qual a ideia do algoritmo?

Exemplo: encontrar a posição do número **8** na lista ordenada a seguir:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| 1 | 1 | 3 | 4 | 5 | 6 | 8 | 11 | 12 | 21 | 23 | 33 | 45 | 98 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |

Etapa: 2

Posição do meio: $0 + [(6 - 0) + 1] / 2 = 0 + 3 \rightarrow 3$

3. Busca binária

3.1 Qual a ideia do algoritmo?

Exemplo: encontrar a posição do número **8** na lista ordenada a seguir:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| 1 | 1 | 3 | 4 | 5 | 6 | 8 | 11 | 12 | 21 | 23 | 33 | 45 | 98 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |

Etapa: 2

Posição do meio: $0 + 3 = 0 + 3 \rightarrow 3$

Aqui estamos
"arrendondando" para
baixo! $7/2 = 3.5$ e o
inteiro mais próximo
de 3.5 é 3.

3. Busca binária

3.1 Qual a ideia do algoritmo?

Exemplo: encontrar a posição do número **8** na lista ordenada a seguir:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| 1 | 1 | 3 | 4 | 5 | 6 | 8 | 11 | 12 | 21 | 23 | 33 | 45 | 98 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |

Etapa: 2


Posição do meio: $0 + [(6 - 0) + 1] / 2 = 0 + 3 \rightarrow 3$

3. Busca binária

3.1 Qual a ideia do algoritmo?

Exemplo: encontrar a posição do número **8** na lista ordenada a seguir:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| 1 | 1 | 3 | 4 | 5 | 6 | 8 | 11 | 12 | 21 | 23 | 33 | 45 | 98 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |



Etapa: 2

Posição do meio: $0 + [(6 - 0) + 1] / 2 = 0 + 3 \rightarrow 3$

3. Busca binária

3.1 Qual a ideia do algoritmo?

Exemplo: encontrar a posição do número **8** na lista ordenada a seguir:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| 1 | 1 | 3 | 4 | 5 | 6 | 8 | 11 | 12 | 21 | 23 | 33 | 45 | 98 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |

Etapa: 3


Posição do meio: $4 + [(6 - 4) + 1] / 2 = 4 + 1 \rightarrow 5$

3. Busca binária

3.1 Qual a ideia do algoritmo?

Exemplo: encontrar a posição do número **8** na lista ordenada a seguir:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| 1 | 1 | 3 | 4 | 5 | 6 | 8 | 11 | 12 | 21 | 23 | 33 | 45 | 98 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |



Etapa: 3

Posição do meio: $4 + [(6 - 4) + 1] / 2 = 4 + 1 \rightarrow 5$

3. Busca binária

3.1 Qual a ideia do algoritmo?

Exemplo: encontrar a posição do número **8** na lista ordenada a seguir:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| 1 | 1 | 3 | 4 | 5 | 6 | 8 | 11 | 12 | 21 | 23 | 33 | 45 | 98 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |

Etapa: 4

Posição do meio: $6 + [(6 - 6) + 1] / 2 = 6 + 0 \rightarrow 6$

3. Busca binária

3.1 Qual a ideia do algoritmo?

Exemplo: encontrar a posição do número **8** na lista ordenada a seguir:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| 1 | 1 | 3 | 4 | 5 | 6 | 8 | 11 | 12 | 21 | 23 | 33 | 45 | 98 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |

3. Busca binária

3.2 Qual a eficiência do algoritmo?

Para uma lista de tamanho **n**.

Qual o **pior caso**?

3. Busca binária

3.2 Qual a eficiência do algoritmo?

Para uma lista de tamanho **n**.

Qual o **pior caso**?

A cada etapa do algoritmo, dividimos a lista quase pela metade e seguimos assim até restar 1 elemento, seja ele o que estamos procurando ou não.

3. Busca binária

3.2 Qual a eficiência do algoritmo?

Para uma lista de tamanho **n**.

Qual o **pior caso**?

A cada etapa do algoritmo, dividimos a lista quase pela metade e seguimos assim até restar 1 elemento, seja ele o que estamos procurando ou não.

Reformulando: qual o número máximo de etapas do algoritmo?

3. Busca binária

3.2 Qual a eficiência do algoritmo?

Para uma lista de tamanho **n**.

Qual o **pior caso**?

A cada etapa do algoritmo, dividimos a lista quase pela metade e seguimos assim até restar 1 elemento, seja ele o que estamos procurando ou não.

Reformulando: qual o número máximo de etapas do algoritmo?

Reformulando a reformulação: quantas vezes podemos dividir **n** pela metade?

3. Busca binária

3.2 Qual a eficiência do algoritmo?

Quantas vezes podemos dividir **n** pela metade?

3. Busca binária

1.2 Qual a eficiência do algoritmo?

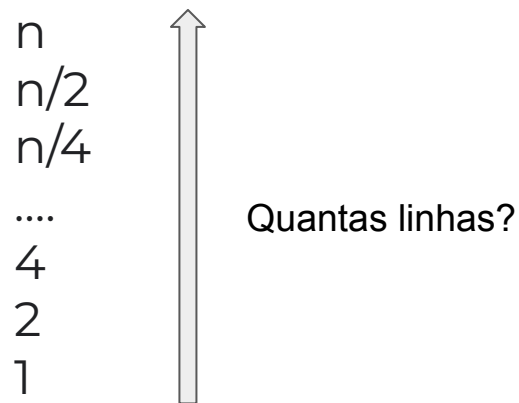
Quantas vezes podemos dividir **n** pela metade (vamos assumir que n é par para simplificar)?

n
n/2
n/4
....
4
2
1

3. Busca binária

3.2 Qual a eficiência do algoritmo?

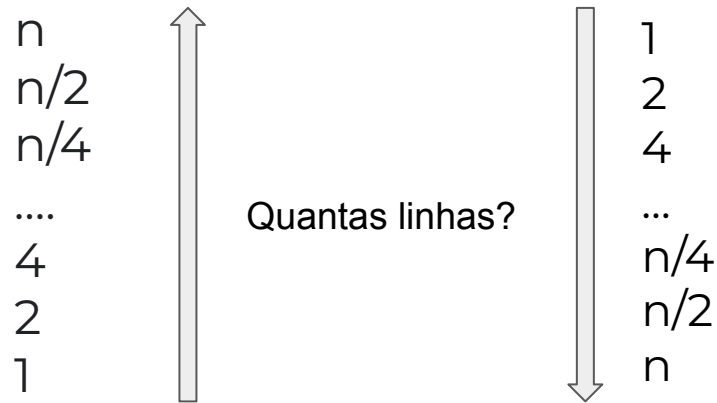
Quantas vezes podemos dividir **n** pela metade (vamos assumir que n é par para simplificar)?



3. Busca binária

3.2 Qual a eficiência do algoritmo?

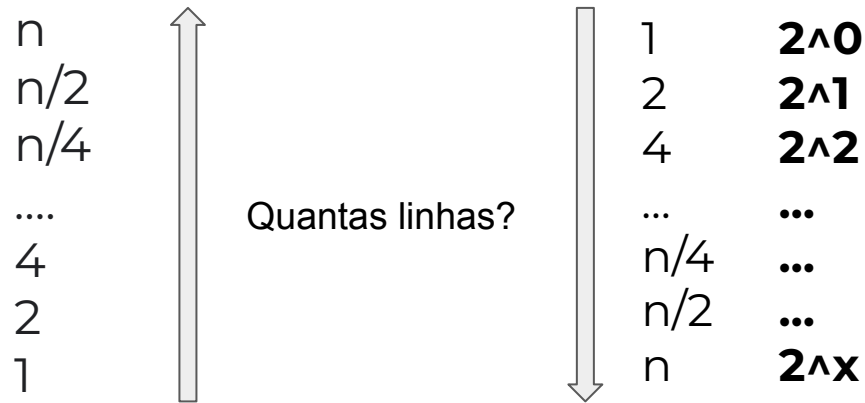
Quantas vezes podemos dividir **n** pela metade (vamos assumir que n é par para simplificar)?



3. Busca binária

3.2 Qual a eficiência do algoritmo?

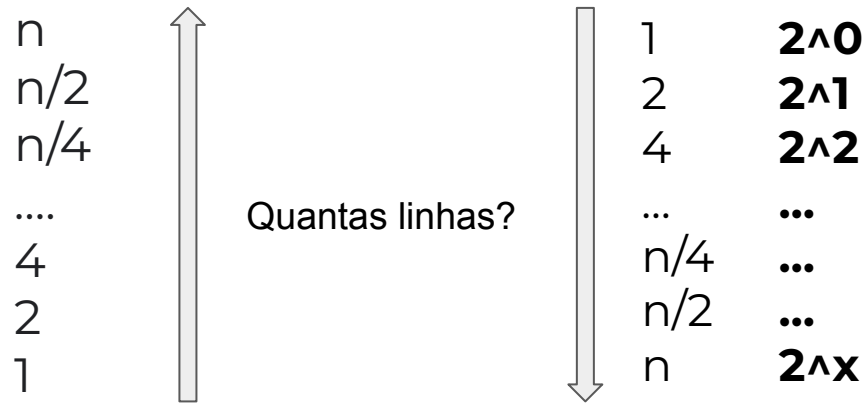
Quantas vezes podemos dividir **n** pela metade (vamos assumir que n é par para simplificar)?



3. Busca binária

3.2 Qual a eficiência do algoritmo?

Quantas vezes podemos dividir **n** pela metade (vamos assumir que n é par para simplificar)?



$$2^x = n$$
$$x = \log_2(n)$$

3. Busca binária

1.2 Qual a eficiência do algoritmo?

Para uma lista de tamanho **n**.

Qual o **pior caso**?

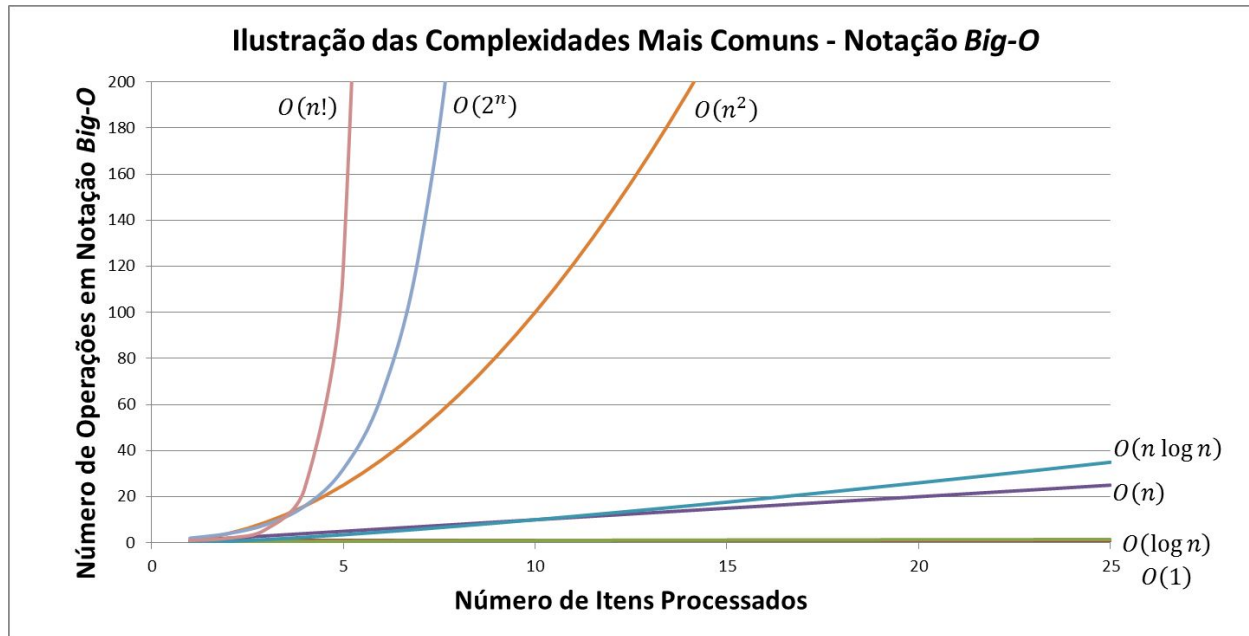
A cada etapa do algoritmo, dividimos a lista quase pela metade e seguimos assim até restar 1 elemento, seja ele o que estamos procurando ou não.

Reformulando: qual o número máximo de etapas do algoritmo?

$O(\log_2(n))$

3. Busca binária

1.2 Qual a eficiência do algoritmo?



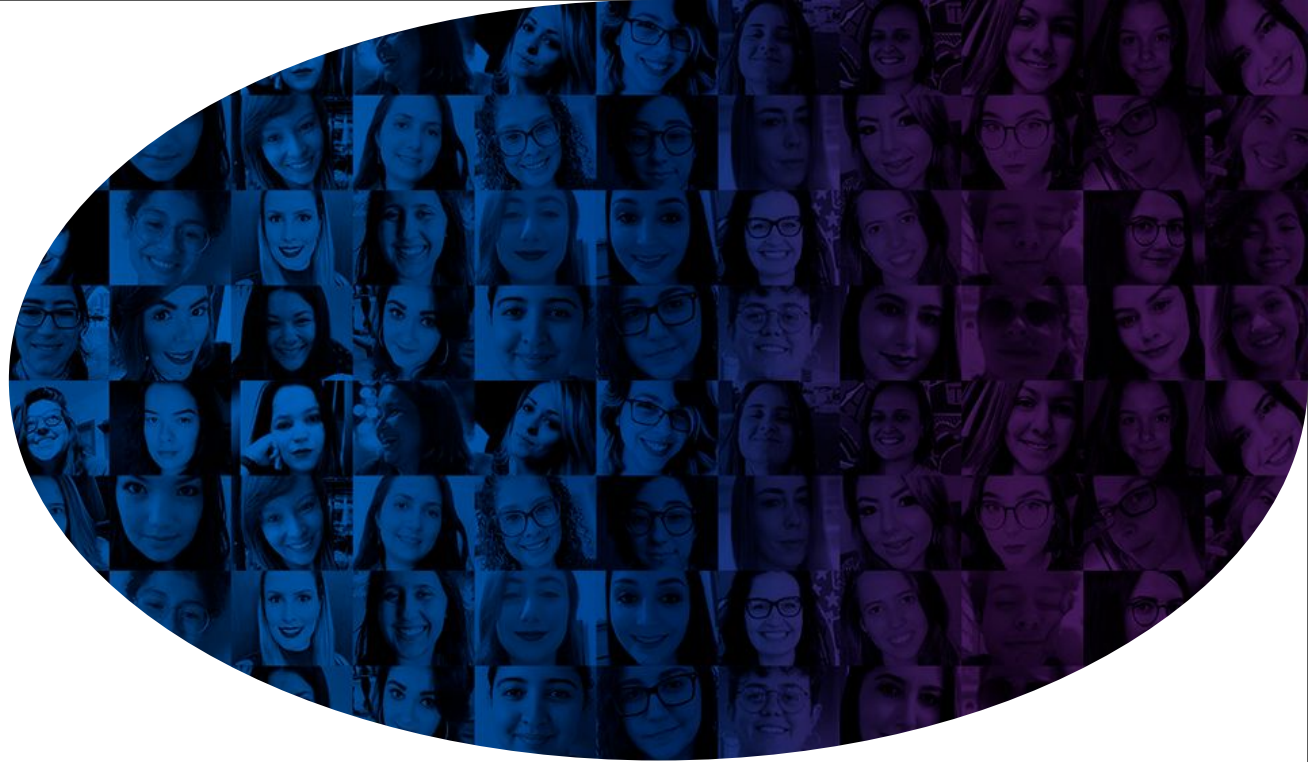
<https://i.stack.imgur.com/R6E94.png>

Luiza
<CODE>

3. Busca binária

3.2 Implementação

```
1 function binarySearch(list, start, end, item) {  
2   const mid = start + Math.floor((end - start)/2);  
3   if(start <= end){  
4     if(list[mid] == item){  
5       return mid;  
6     }  
7     if(item < list[mid]){  
8       return binarySearch(list, start, mid-1, item);  
9     }  
10    if(item > list[mid]){  
11      return binarySearch(list, mid+1, end, item);  
12    }  
13  }  
14  return null;  
15 };
```



Perguntas?

Magalu



#VemSerFeliz