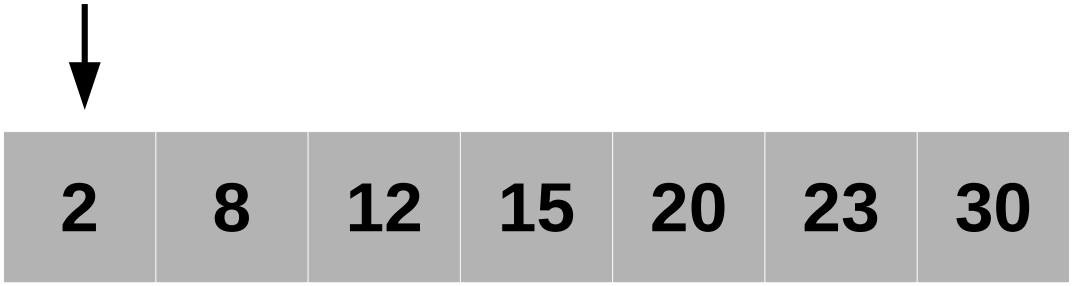
DCC405 – ESTRUTURA DE DADOS II

Aula 03 – Árvores – Conceitos Básicos

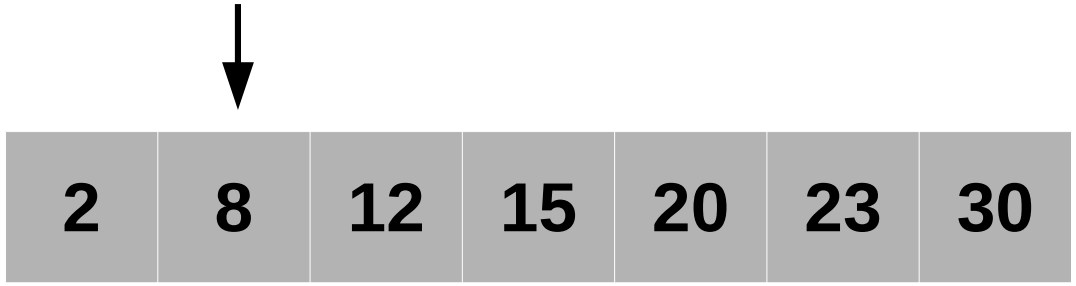
Buscando um elemento

- Imagina que queremos buscar um elemento **(15)** no arranjo (vetor) abaixo. Como fazer?



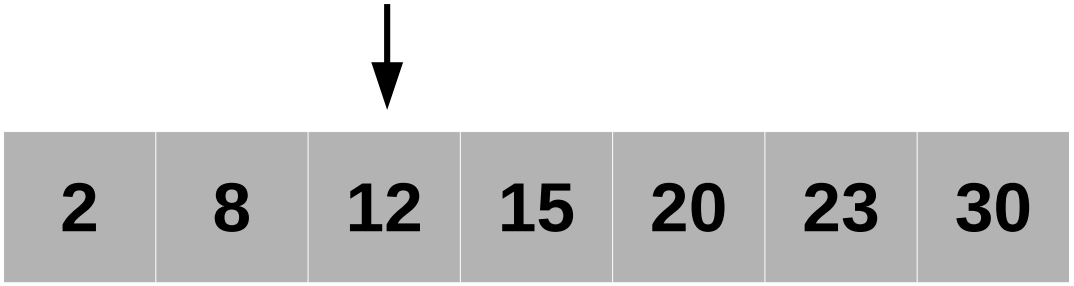
Buscando um elemento

- Imagina que queremos buscar um elemento **(15)** no arranjo (vetor) abaixo. Como fazer?



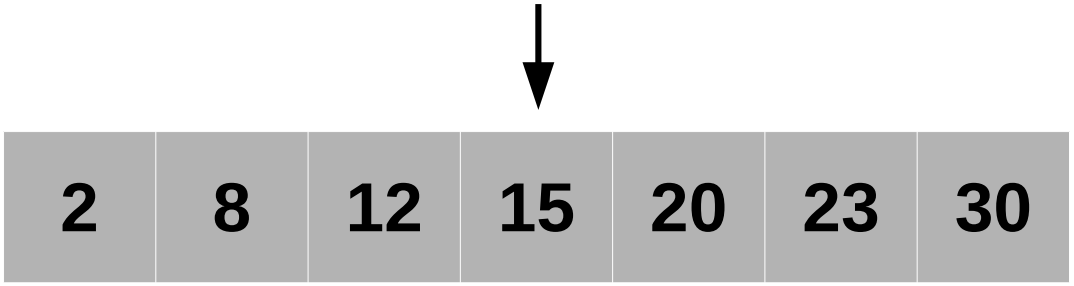
Buscando um elemento

- Imagina que queremos buscar um elemento **(15)** no arranjo (vetor) abaixo. Como fazer?



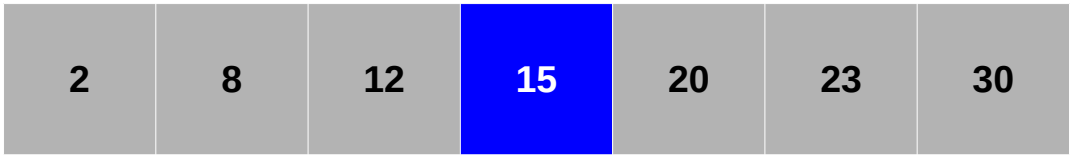
Buscando um elemento

- Imagina que queremos buscar um elemento **(15)** no arranjo (vetor) abaixo. Como fazer?



Buscando um elemento

- Imagina que queremos buscar um elemento **(15)** no arranjo (vetor) abaixo. Como fazer?



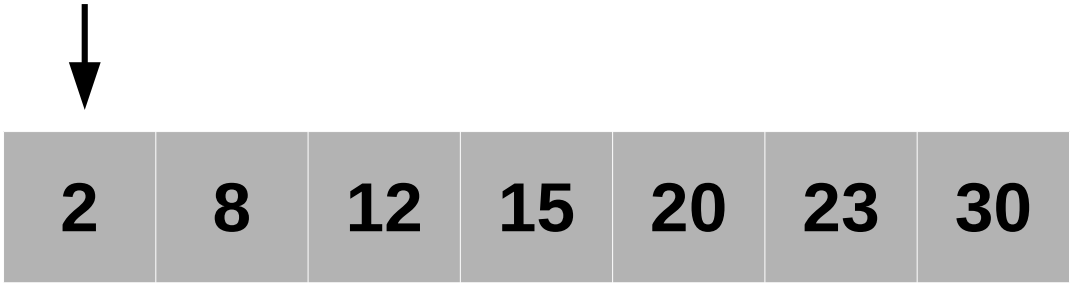
Buscando um elemento

- E o **16**?

2	8	12	15	20	23	30

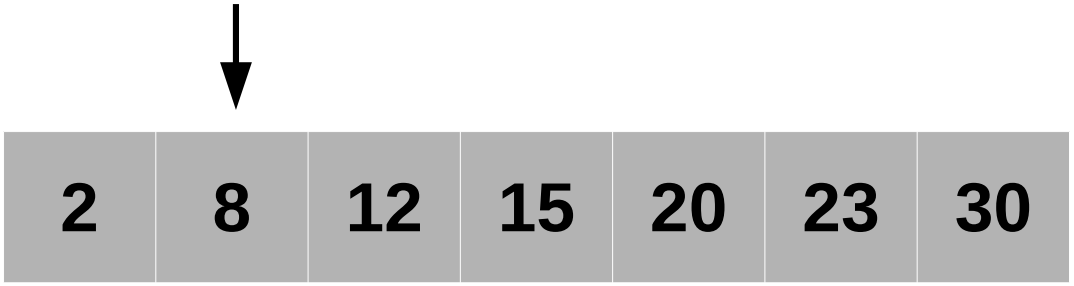
Buscando um elemento

- E o **16**?



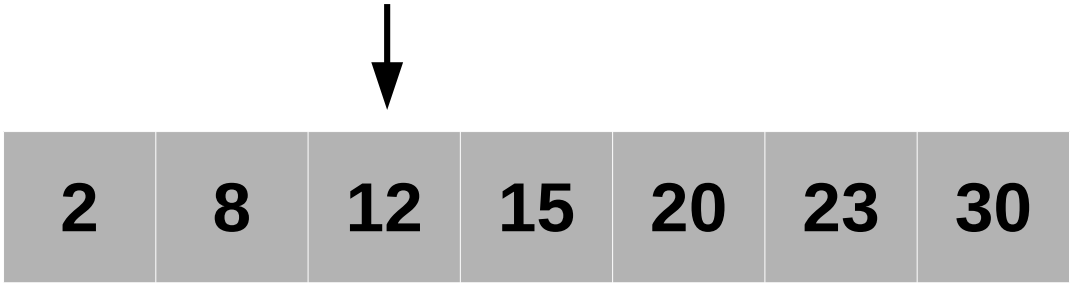
Buscando um elemento

- E o **16**?



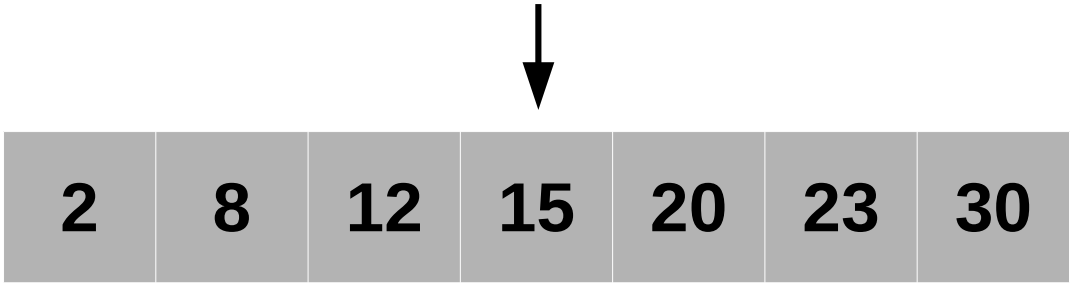
Buscando um elemento

- E o **16**?



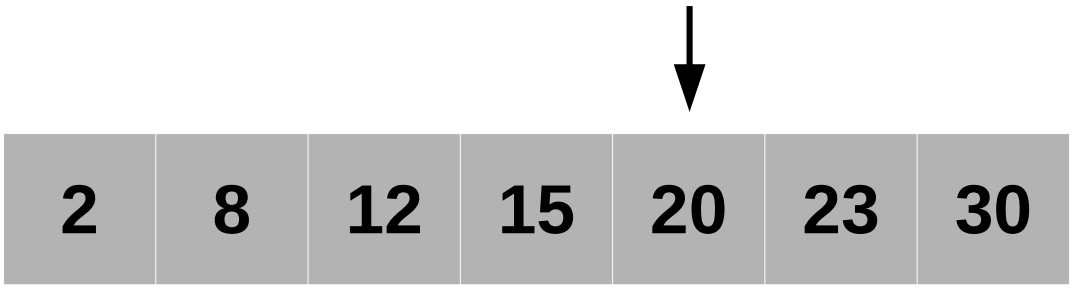
Buscando um elemento

- E o **16**?



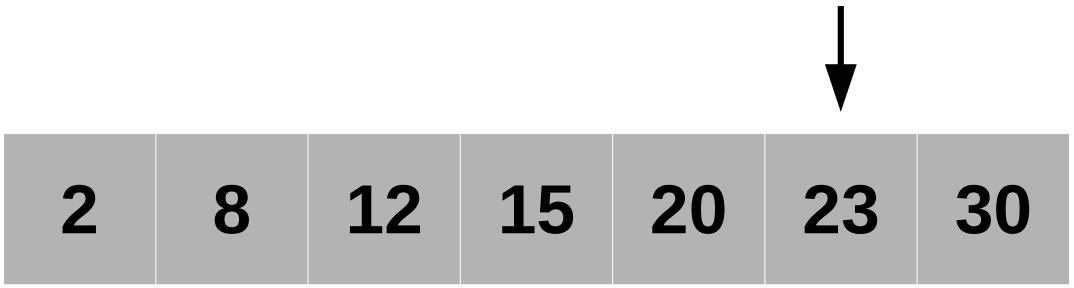
Buscando um elemento

- E o **16**?



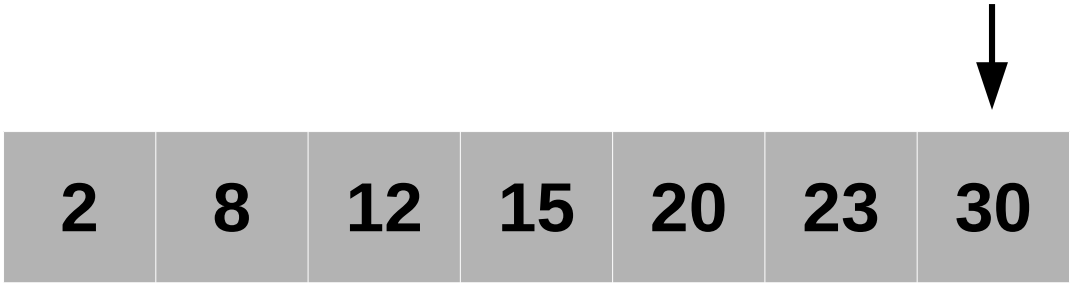
Buscando um elemento

- E o **16**?



Buscando um elemento

- E o **16**?



Buscando um elemento

- E o **16**?

2	8	12	15	20	23	30

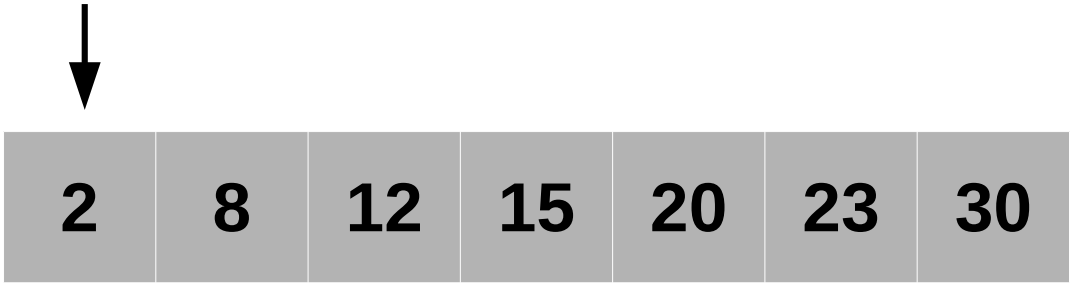
Buscando um elemento

- Será que não poderíamos aproveitar o fato de que o arranjo está ordenado? Ex: buscando o **16**

2	8	12	15	20	23	30

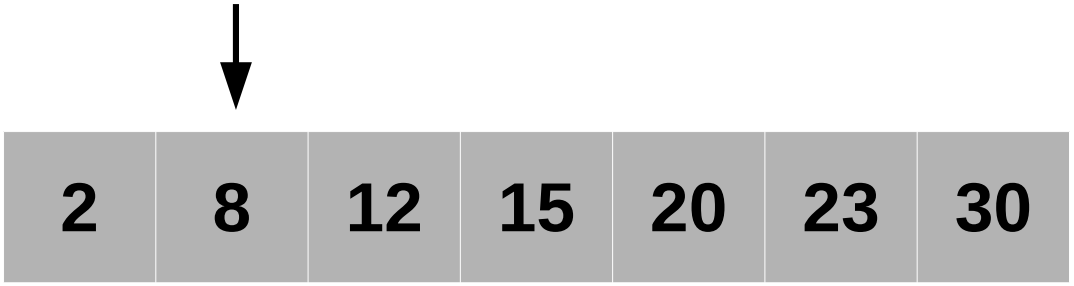
Buscando um elemento

- Será que não poderíamos aproveitar o fato de que o arranjo está ordenado? Ex: buscando o **16**



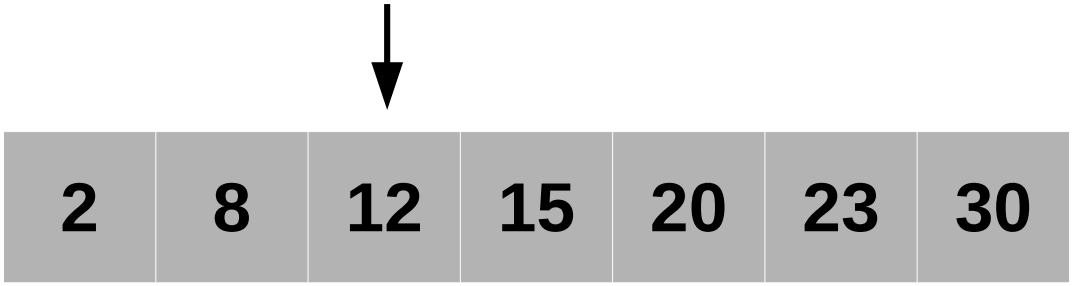
Buscando um elemento

- Será que não poderíamos aproveitar o fato de que o arranjo está ordenado? Ex: buscando o **16**



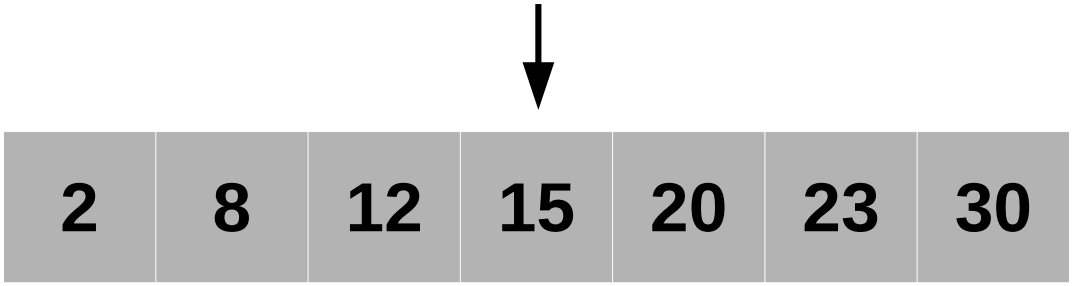
Buscando um elemento

- Será que não poderíamos aproveitar o fato de que o arranjo está ordenado? Ex: buscando o **16**



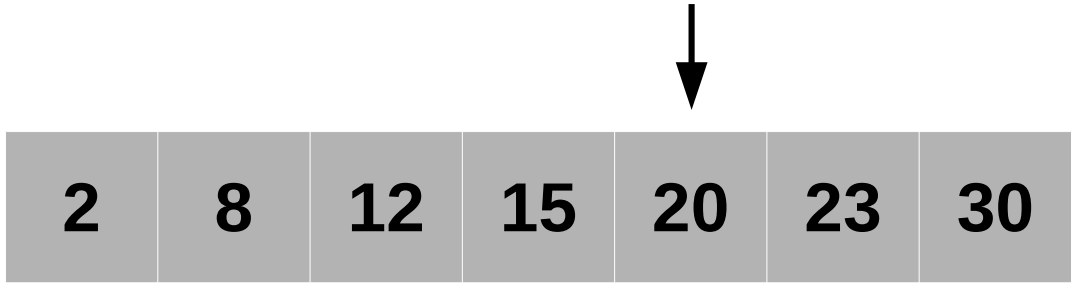
Buscando um elemento

- Será que não poderíamos aproveitar o fato de que o arranjo está ordenado? Ex: buscando o **16**



Buscando um elemento

- Será que não poderíamos aproveitar o fato de que o arranjo está ordenado? Ex: buscando o **16**



Buscando um elemento

- Será que não poderíamos aproveitar o fato de que o arranjo está ordenado? Ex: buscando o **16**

2	8	12	15	20	23	30

Buscando um elemento

- Ainda temos um problema.
- Se o número buscado for maior que todos (ex: **31**), ainda corremos o arranjo inteiro. Teria como melhorar?

2	8	12	15	20	23	30

Buscando um elemento

- Ainda temos um problema.
- Se o número buscado for maior que todos (ex: **31**), ainda corremos o arranjo inteiro. Teria como melhorar?

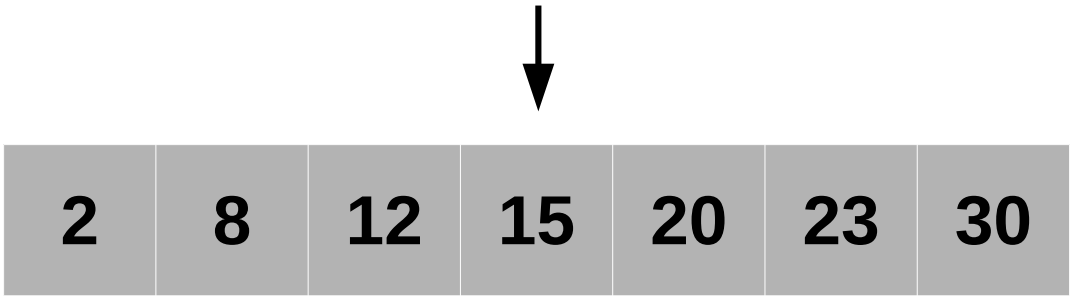
- Possível solução: **Busca Binária**

2	8	12	15	20	23	30

Buscando um elemento

- Ainda temos um problema.
- Se o número buscado for maior que todos (ex: **31**), ainda corremos o arranjo inteiro. Teria como melhorar?

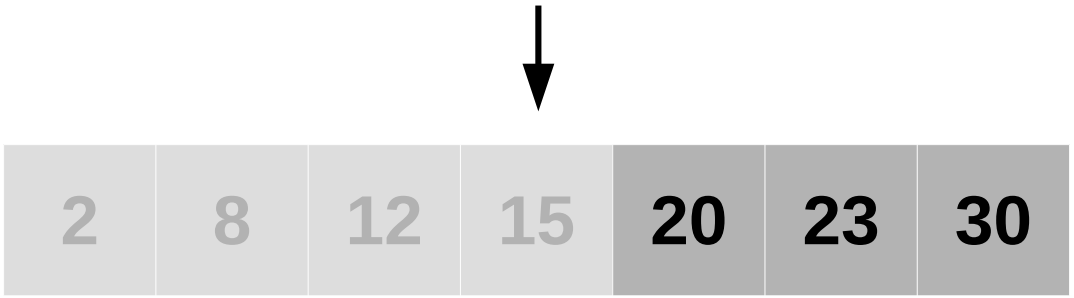
- Possível solução: **Busca Binária**



Buscando um elemento

- Ainda temos um problema.
- Se o número buscado for maior que todos (ex: **31**), ainda corremos o arranjo inteiro. Teria como melhorar?

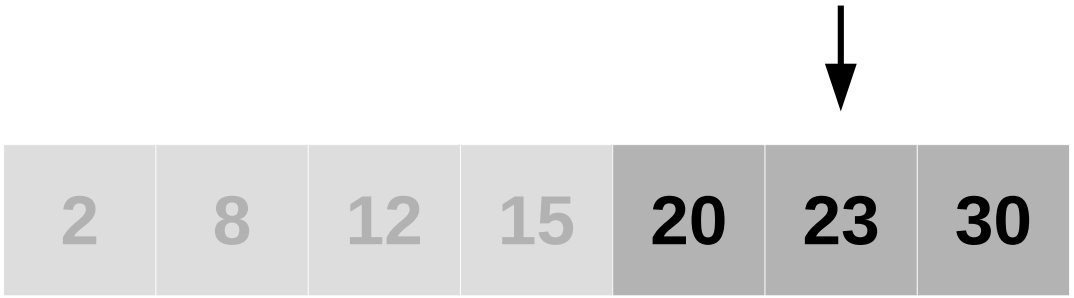
- Possível solução: **Busca Binária**



Buscando um elemento

- Ainda temos um problema.
- Se o número buscado for maior que todos (ex: **31**), ainda corremos o arranjo inteiro. Teria como melhorar?

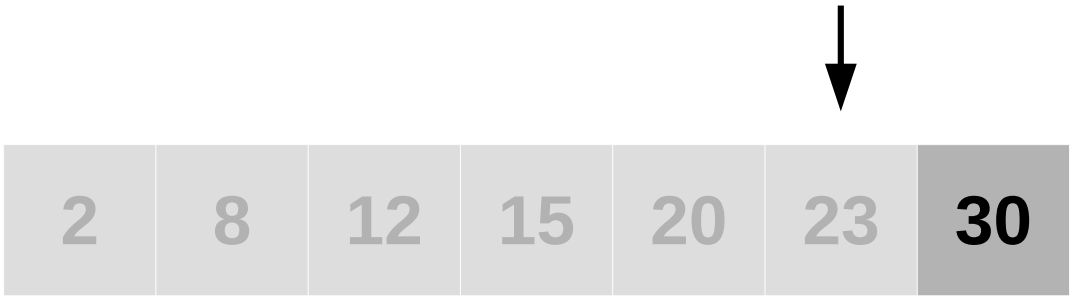
- Possível solução: **Busca Binária**



Buscando um elemento

- Ainda temos um problema.
- Se o número buscado for maior que todos (ex: **31**), ainda corremos o arranjo inteiro. Teria como melhorar?

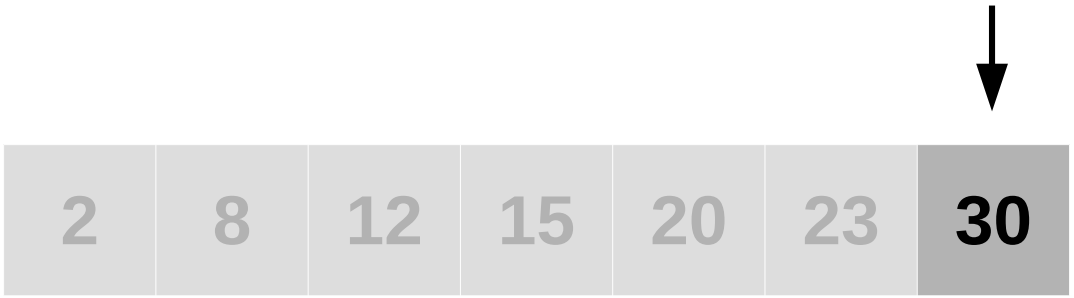
- Possível solução: **Busca Binária**



Buscando um elemento

- Ainda temos um problema.
- Se o número buscado for maior que todos (ex: **31**), ainda corremos o arranjo inteiro. Teria como melhorar?

- Possível solução: **Busca Binária**



Buscando um elemento

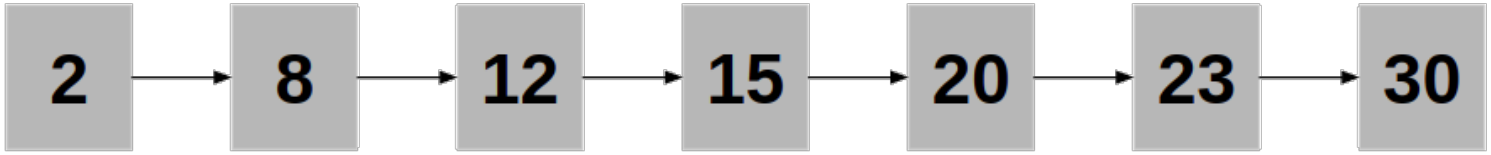
- Ainda temos um problema.
- Se o número buscado for maior que todos (ex: **31**), ainda corremos o arranjo inteiro. Teria como melhorar?

- Possível solução: **Busca Binária**

2	8	12	15	20	23	30

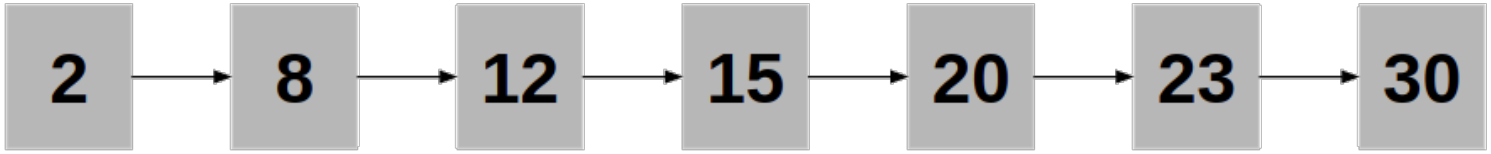
Buscando um elemento

- **Busca Binária** é mais eficiente… mas depende de arranjos estáticos.
- E se tivéssemos uma lista ligada? Ops!



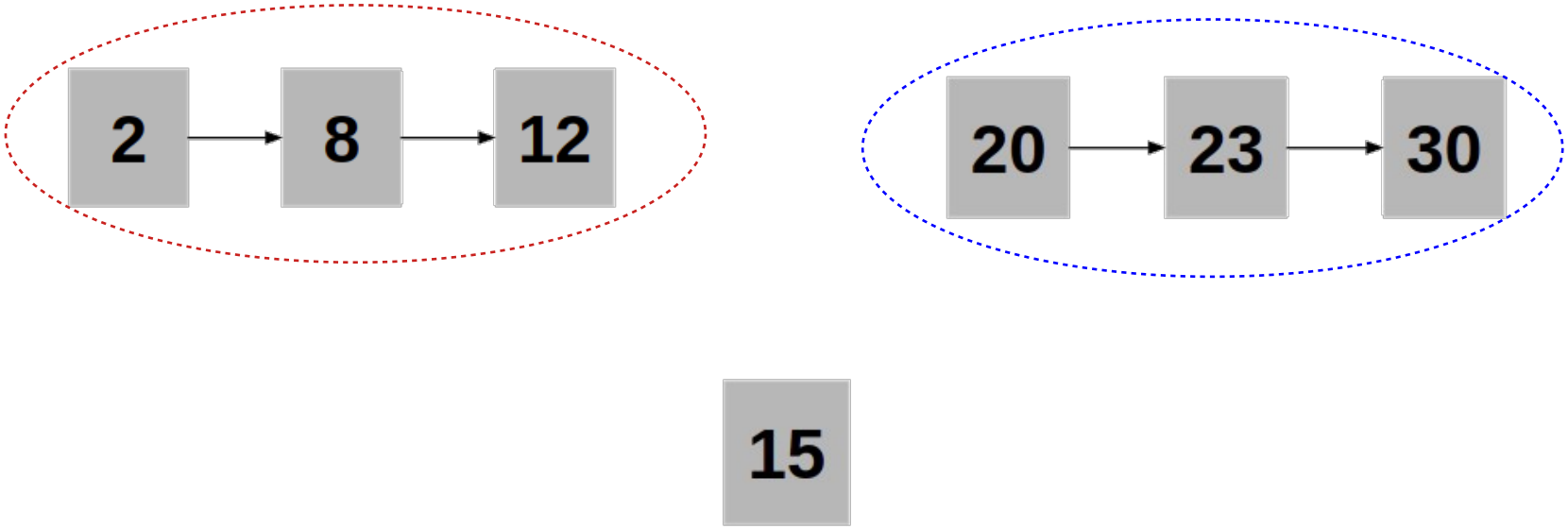
Buscando um elemento

- Será que não podemos ter uma estrutura dinâmica que nos ajude nessa tarefa?



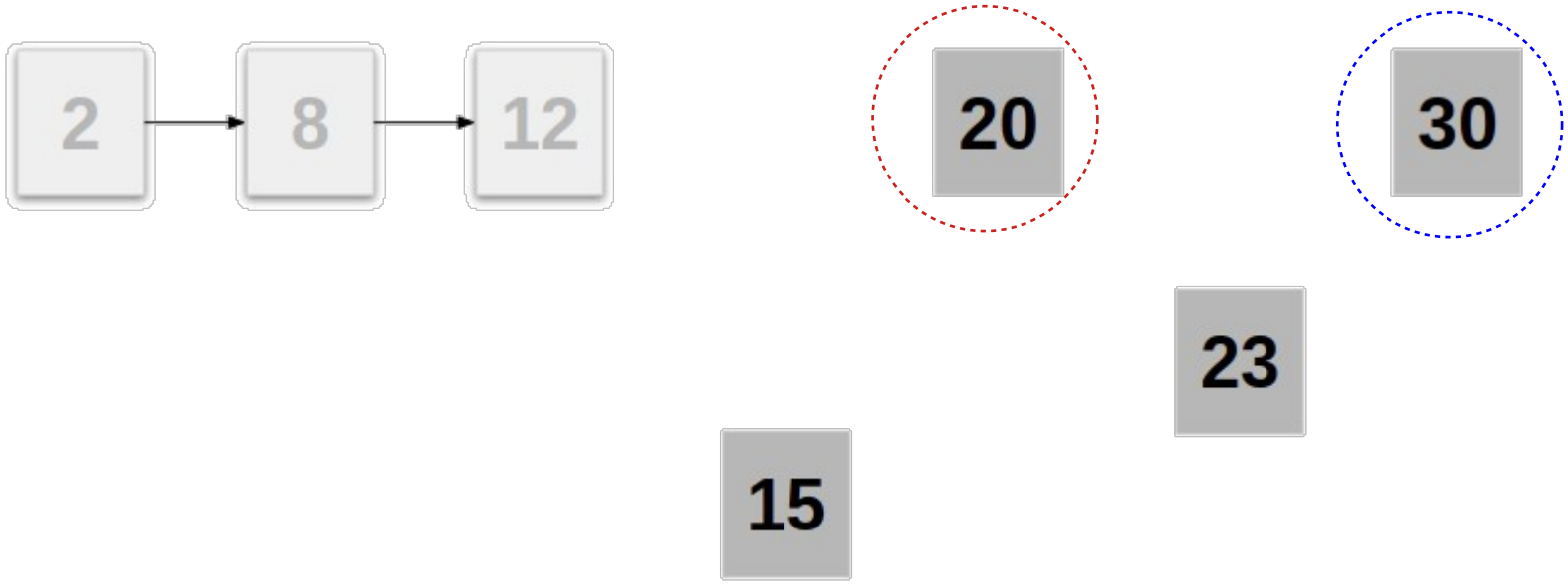
Buscando um elemento

- Será que não podemos ter uma estrutura dinâmica que nos ajude nessa tarefa?



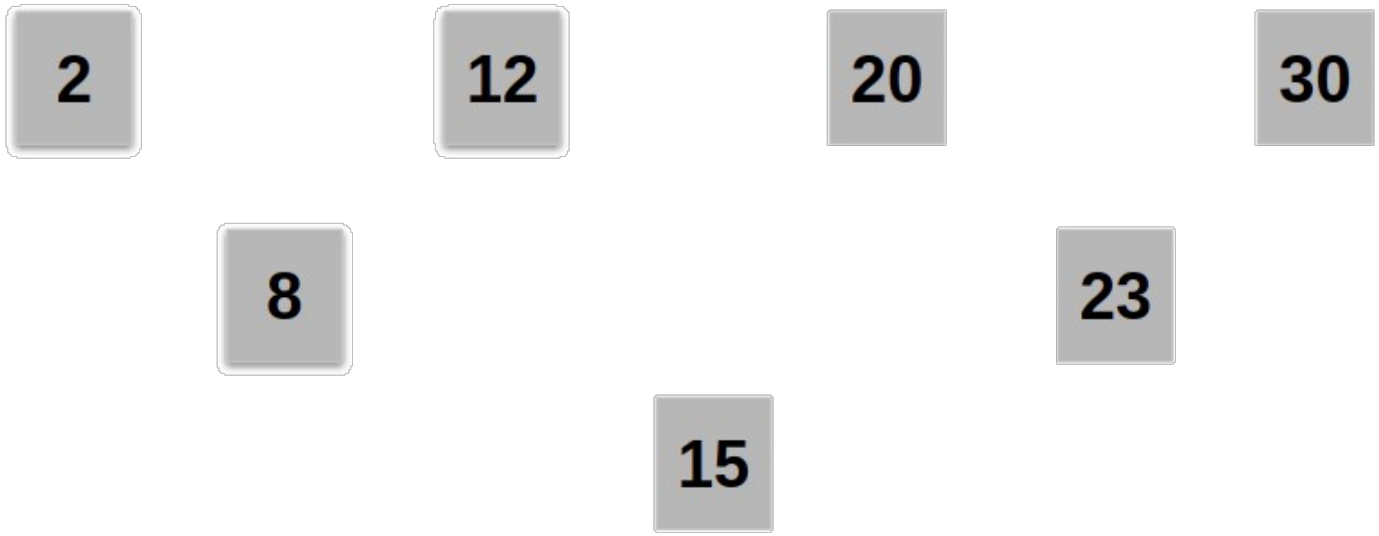
Buscando um elemento

- Será que não podemos ter uma estrutura dinâmica que nos ajude nessa tarefa?



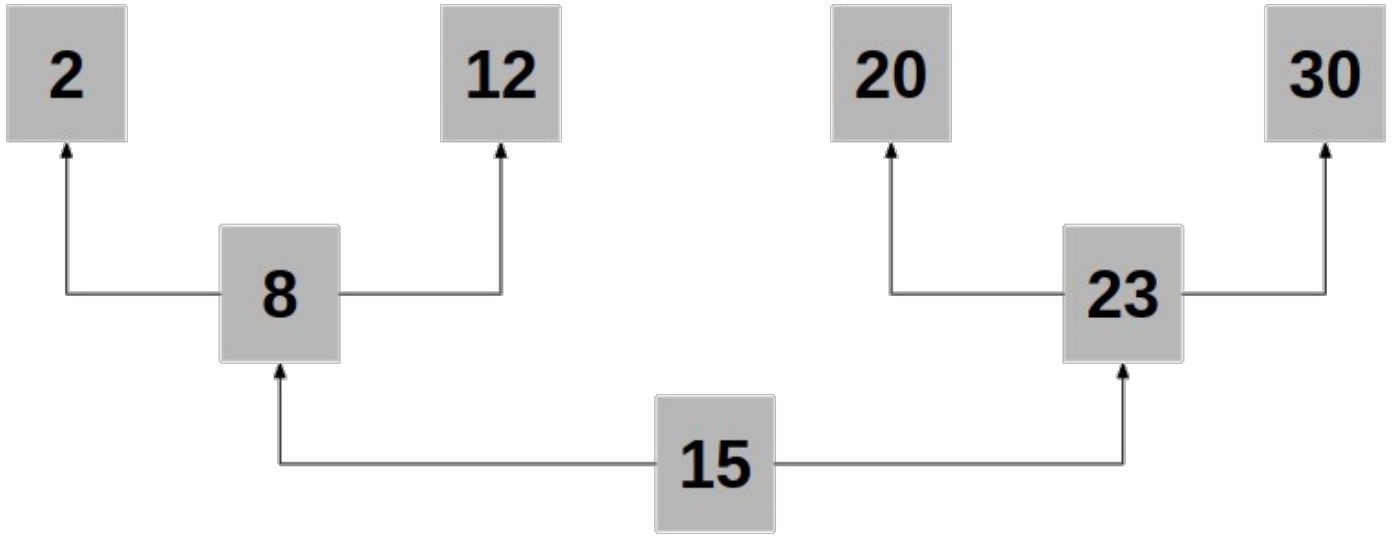
Buscando um elemento

- Será que não podemos ter uma estrutura dinâmica que nos ajude nessa tarefa?



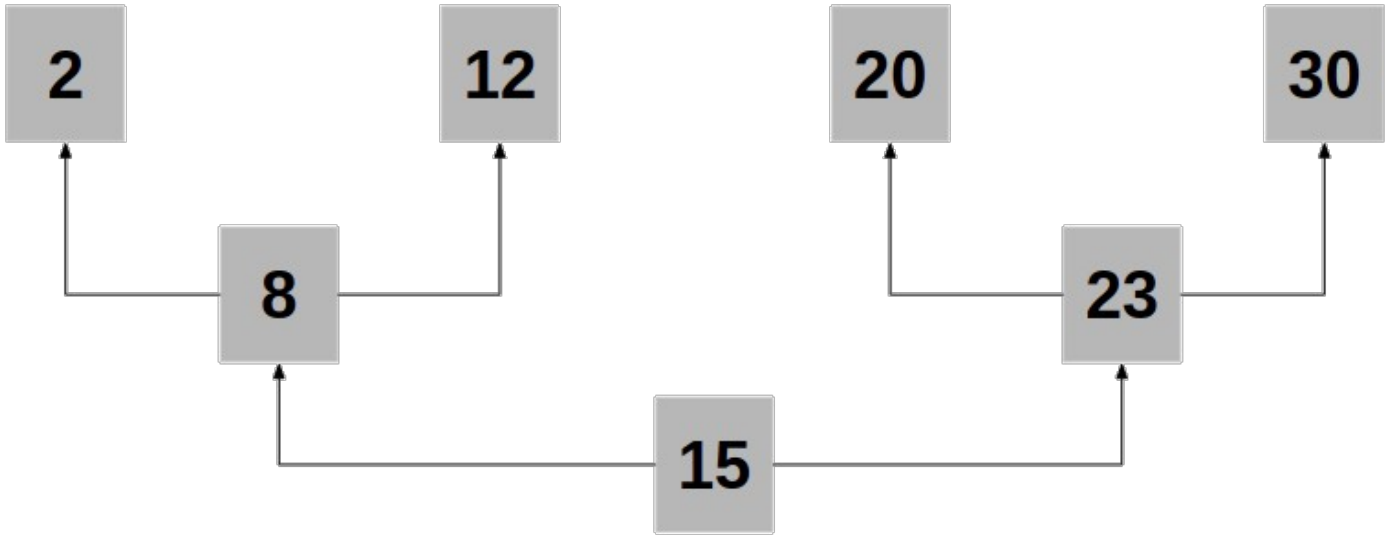
Buscando um elemento

- Será que não podemos ter uma estrutura dinâmica que nos ajude nessa tarefa?



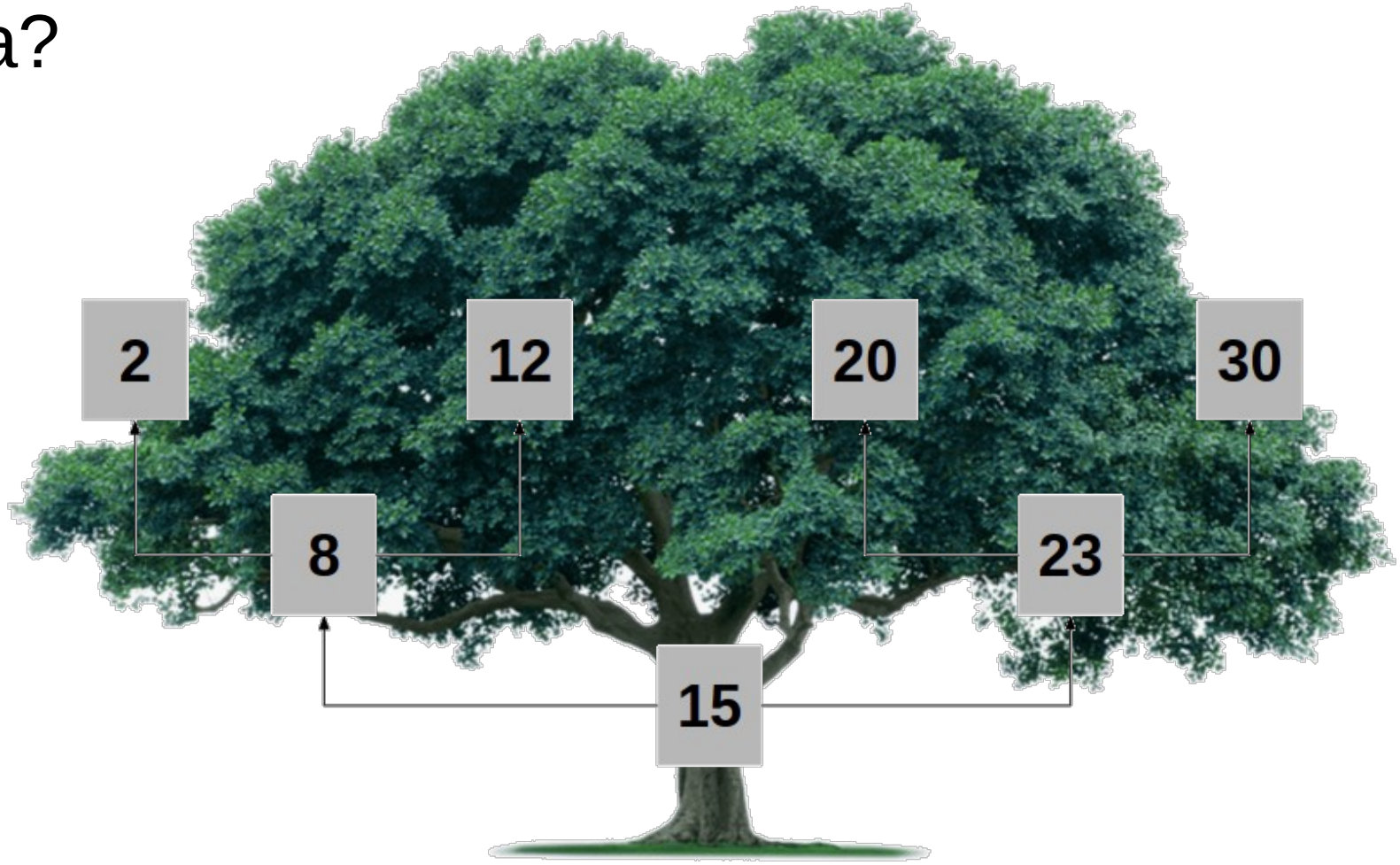
Buscando um elemento

- Eis a nossa estrutura. Que nome daremos a ela?



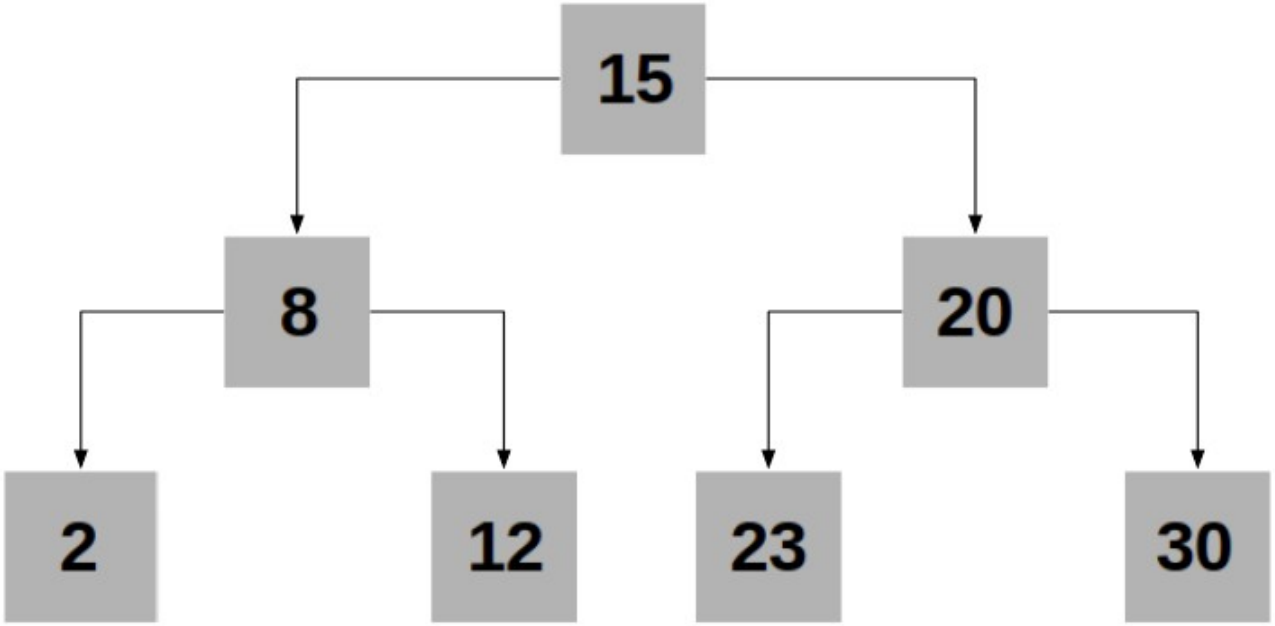
Buscando um elemento

- Eis a nossa estrutura. Que nome daremos a ela?



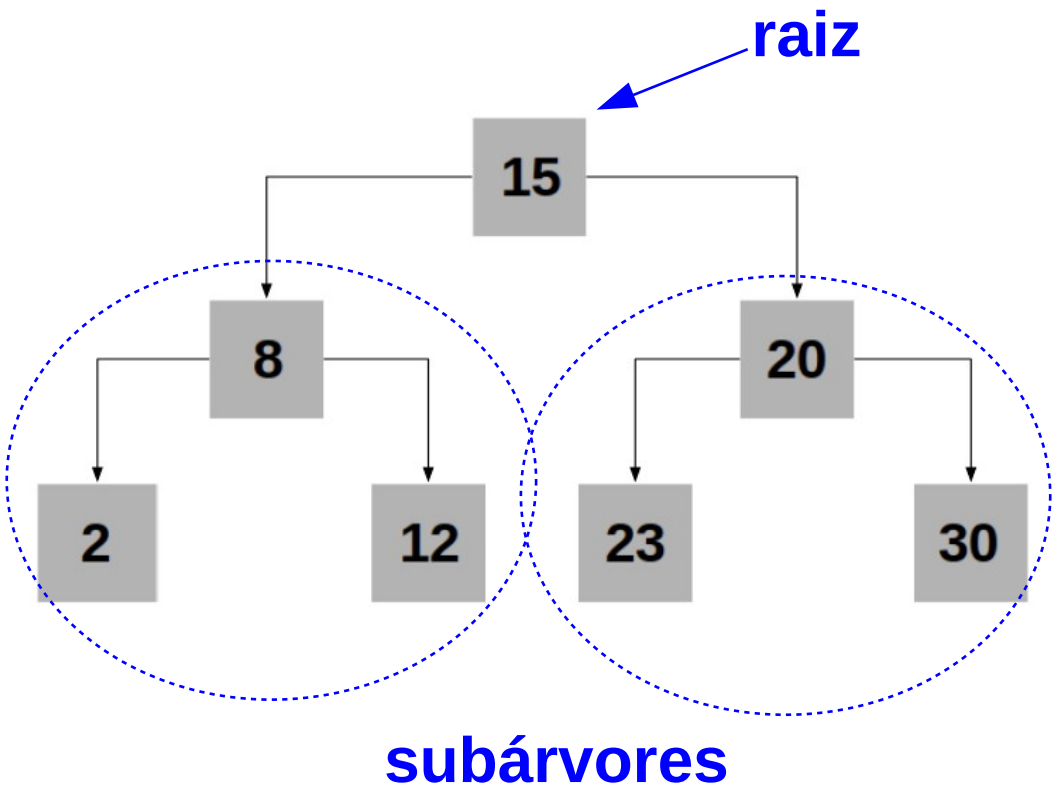
Buscando um elemento

- Uma observação: em computação costumamos representar a árvore de forma invertida



Árvore - definição

- Uma **árvore** é um conjunto de **nós** consistindo de um nó chamado **raiz**, abaixo do qual estão as **subárvores** que compõem essa árvore.



Árvore - definição

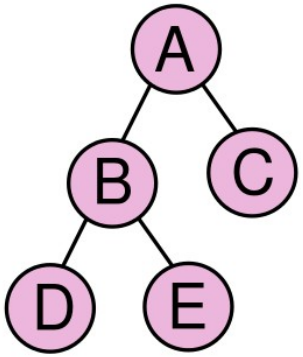
- **Definição formal:**
  - Formalmente, definimos uma árvore ***T*** como um conjunto finito de zero ou mais nós tal que:
    - se o número de nós = 0, temos uma árvore vazia, ou
    - se o número de nós > 0
      - existe um nó especialmente denominado raiz de ***T***
      - os nós restantes formam m >= 0 conjuntos disjuntos $p_1, p_2, \ldots, p_m$, cada um desses conjuntos é uma árvore em si, chamada subárvore de raiz ***T***, ou simplesmente subárvore.

Fonte: Szwarcfiter, Jayme Luiz; Markenzon, Lilian (2014). Estruturas de Dados e seus Algoritmos 3ª ed. Rio de Janeiro: LTC. ISBN 978-85-216-1750-1

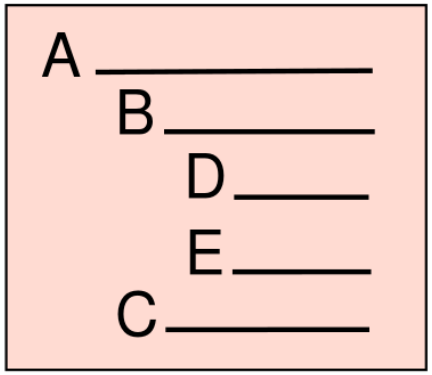
Árvores - conceitos

- São um tipo especial de grafo
- Qualquer par de vértices (nós) está conectado a apenas uma aresta
- Grafo conexo (todos estão conectados)
- Acíclico (não possui ciclos)

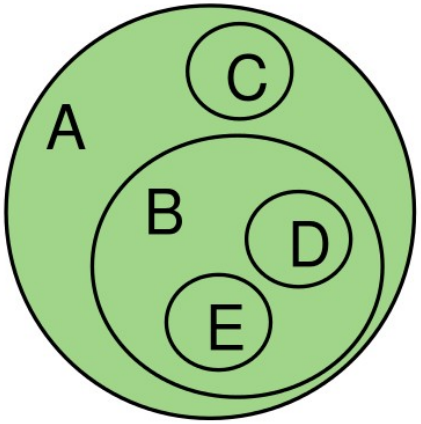
Árvores - Representações



a) Hierárquica



b) Diagrama de barras



c) Diagrama de inclusão (Diagrama de Veen)

(A(B(D)(E))(C))

d) Aninhamento

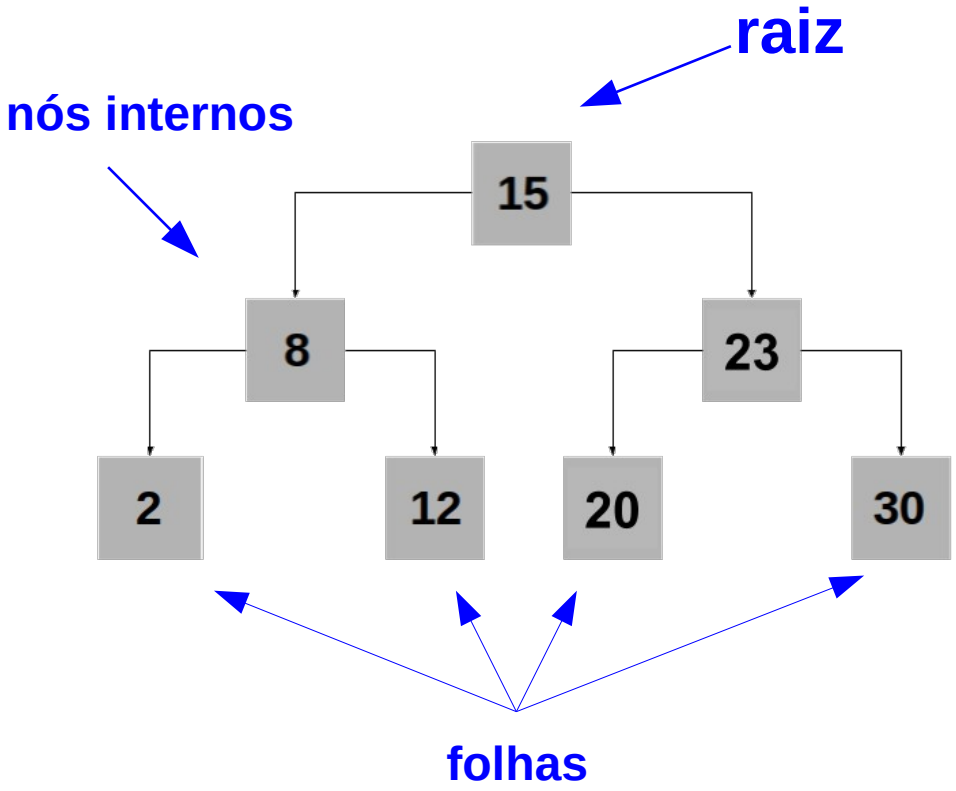
1A; 1.1B; 1.1.1D; 1.1.2E; 1.2C

e) Numeração por níveis

Árvore - Definições

GRAU

- O número de subárvores de cada nó é chamado de **grau** desse nó. No nosso exemplo ao lado, todo nó tem grau 2, exceto as **folhas**, que têm grau 0.



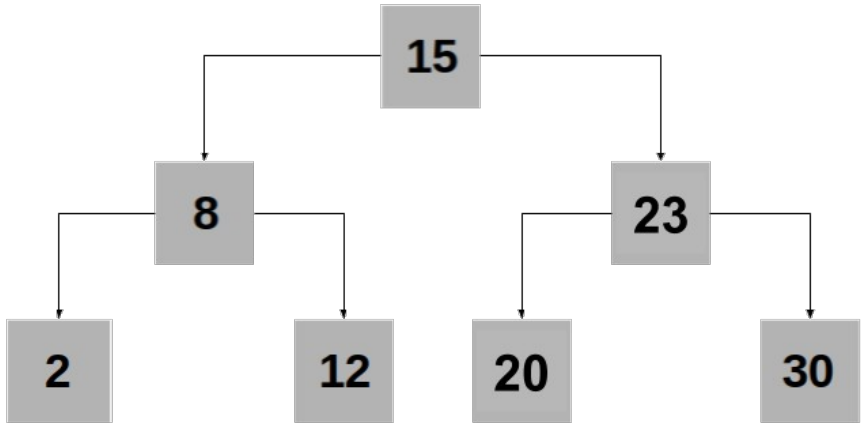
Árvore - Definições

Descendentes

- Nós abaixo de um determinado nó são seus descendentes.

  Descendentes do **8**: 2 e 12

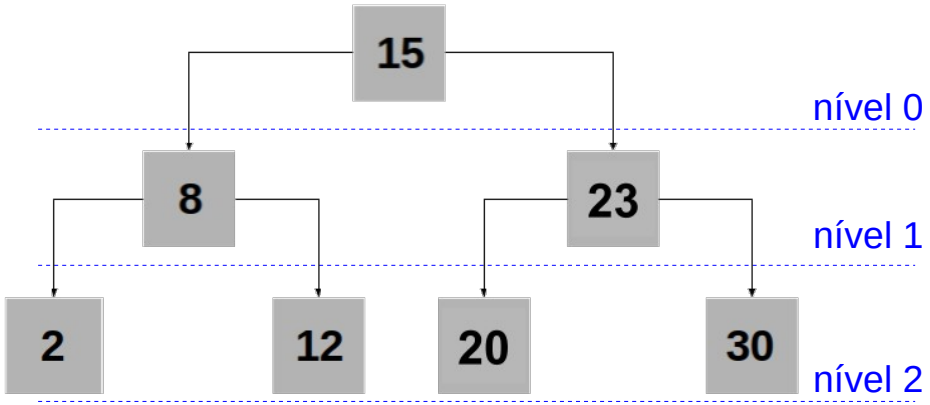
  Descendentes do **15**: todos os demais.



Árvore - Definições

NÍVEL

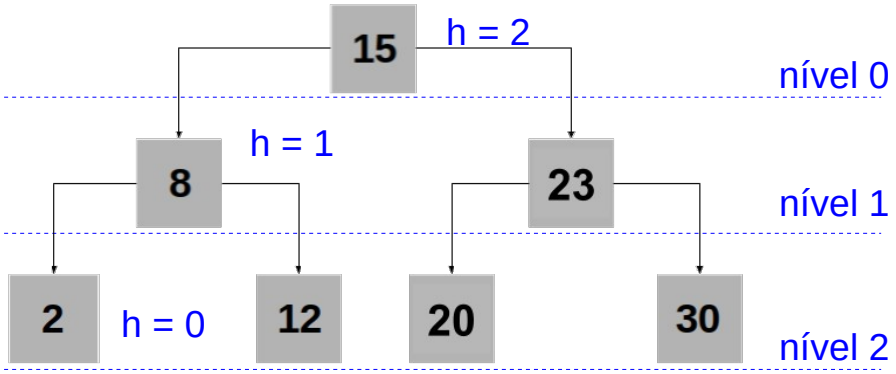
- O **nível** do nó raiz é 0 (zero).



Árvore - Definições

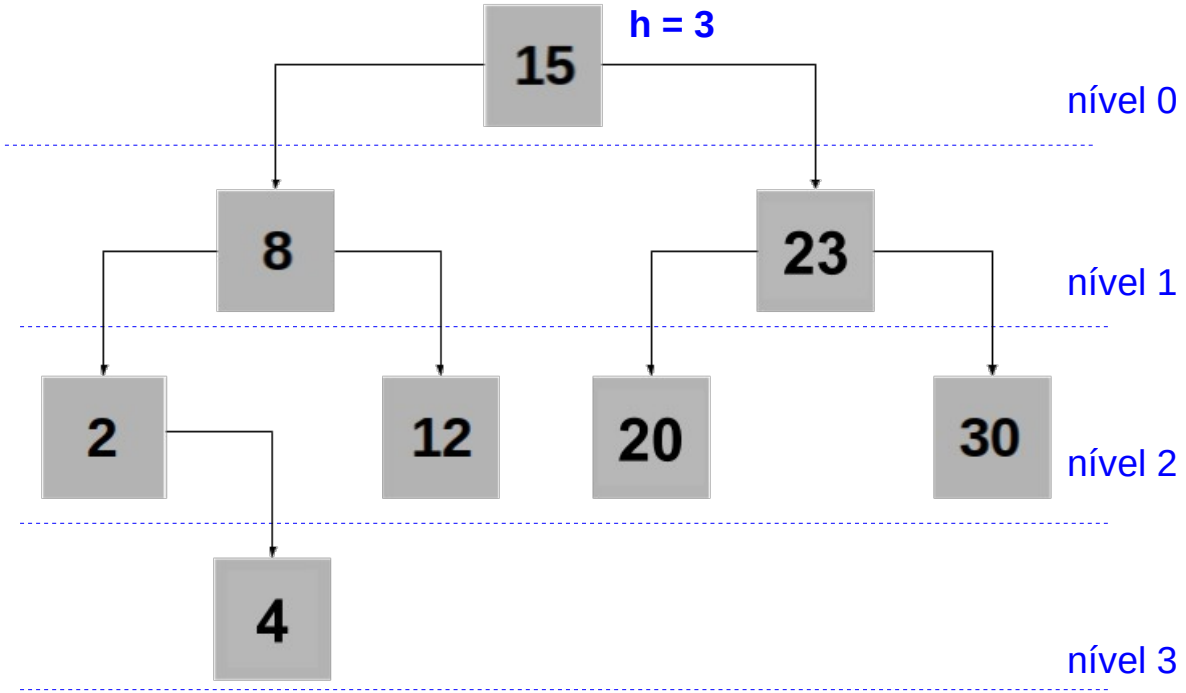
ALTURA

- A **altura** (**h**) de um nó é o comprimento do caminho mais longo entre ele e uma folha.
- **OBS:** Vale notar que a árvore é percorrida da raiz às folhas.



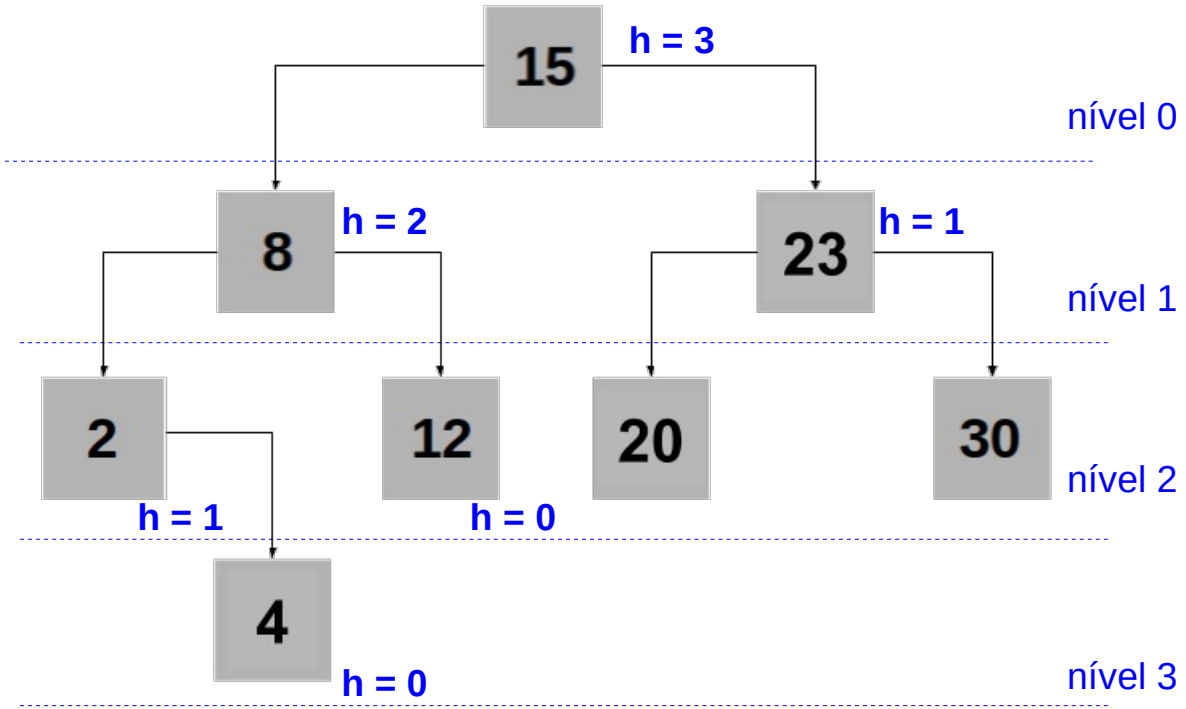
Árvore - Definições

- Nem sempre a árvore estará perfeitamente balanceada. Ainda assim, as definições de altura, nível etc valem.



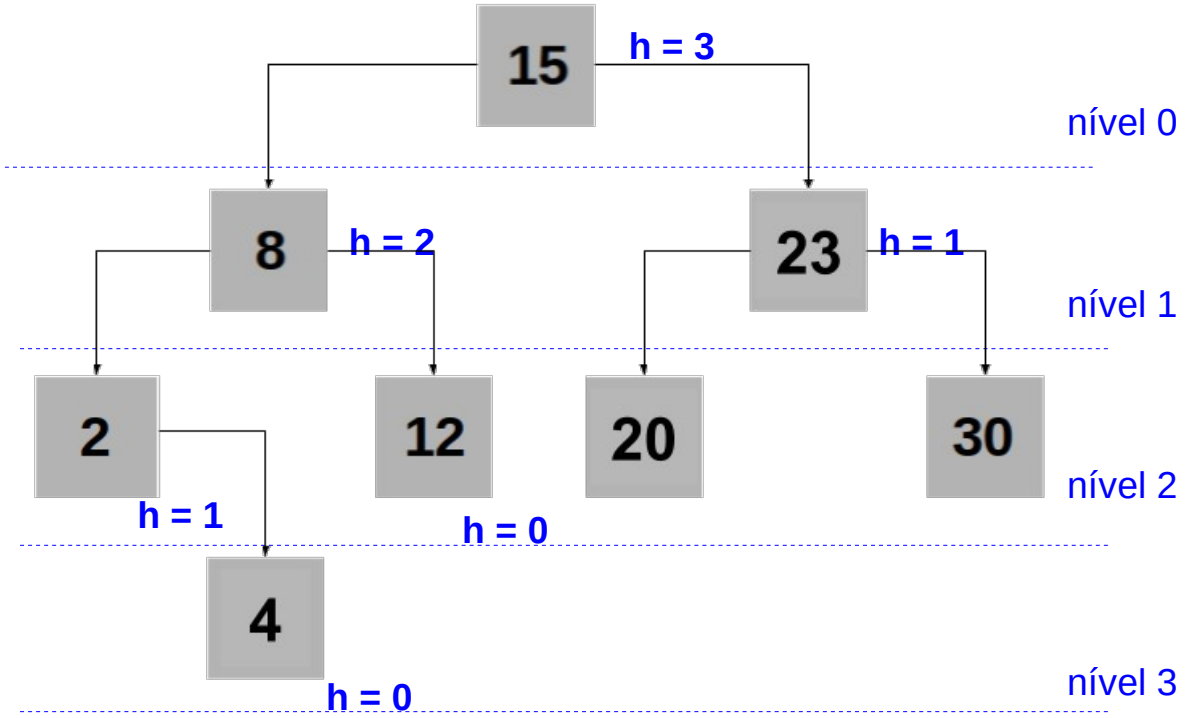
Árvore - Definições

- Nem sempre a árvore estará perfeitamente balanceada. Ainda assim, as definições de altura, nível etc valem.



Árvore - Definições

- A altura de uma árvore é a altura do nó raiz. Da mesma forma, o endereço de uma árvore na memória será o endereço de seu nó raiz.

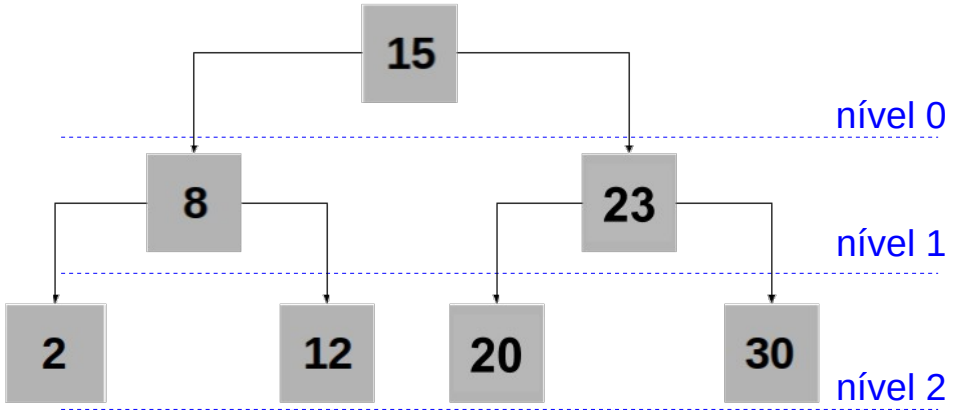


Uma função não recebe uma árvore. Ela recebe o endereço do nó raiz.

Árvore - Definições

PROFUNDIDADE

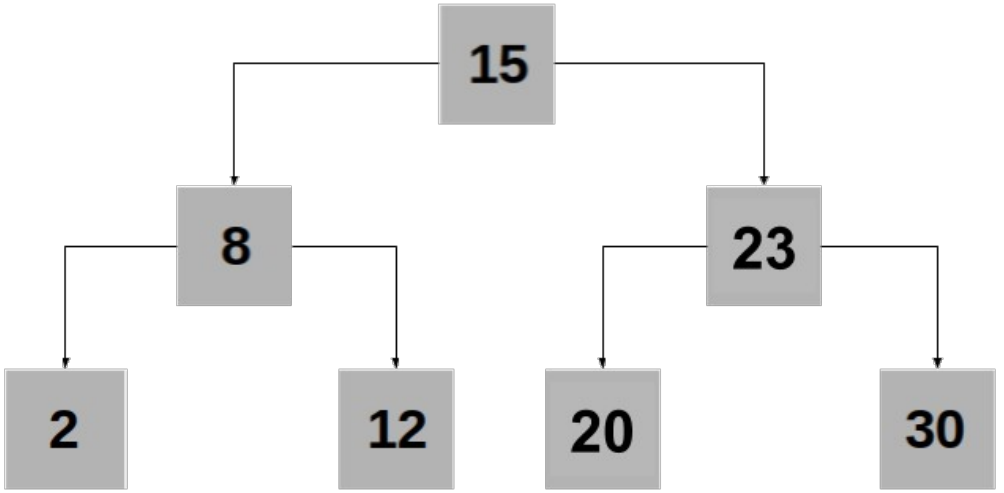
- A profundidade de um nó é a distância da raiz a esse nó.



- Profundidade de **15**: 0
- Profundidade de **8**: 1
- Profundidade de **12**: 2

Árvores Binárias

- Uma árvore binária é uma árvore em que, abaixo de cada nó, existem no máximo duas subárvores.

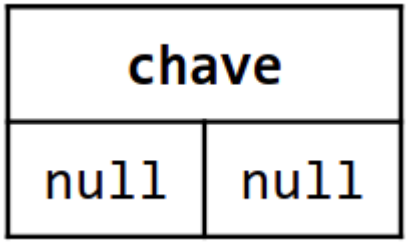


A árvore binária terá 0, 1 ou 2 descendentes.

Árvores Binárias

- Como representamos computacionalmente uma árvore binária?
  Unindo nós.

- E como representamos os nós?
  Com 2 ponteiros: uma para a subárvore da esquerda e um para subárvore da direita.
  Além de um campo para a chave de dados.



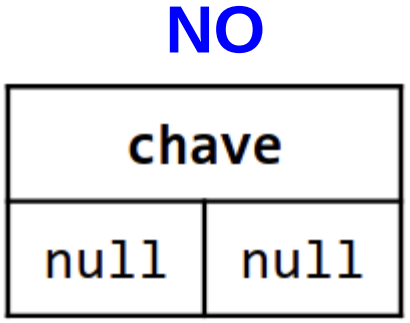
Árvores Binárias

```c
#include <stdio.h>
#include <stdio.h>

#define true 1
#define false 0

typedef struct no {
    int chave;
    struct no *esq;
    struct no *dir;
} NO;

typedef NO* PONT;
```



Obrigado!

Próxima aula

- Assunto: **Árvores binárias de busca.**

Referências

→ **Prof. Norton T. Roman e Luciano A. Digiampietri –** UNIVESP