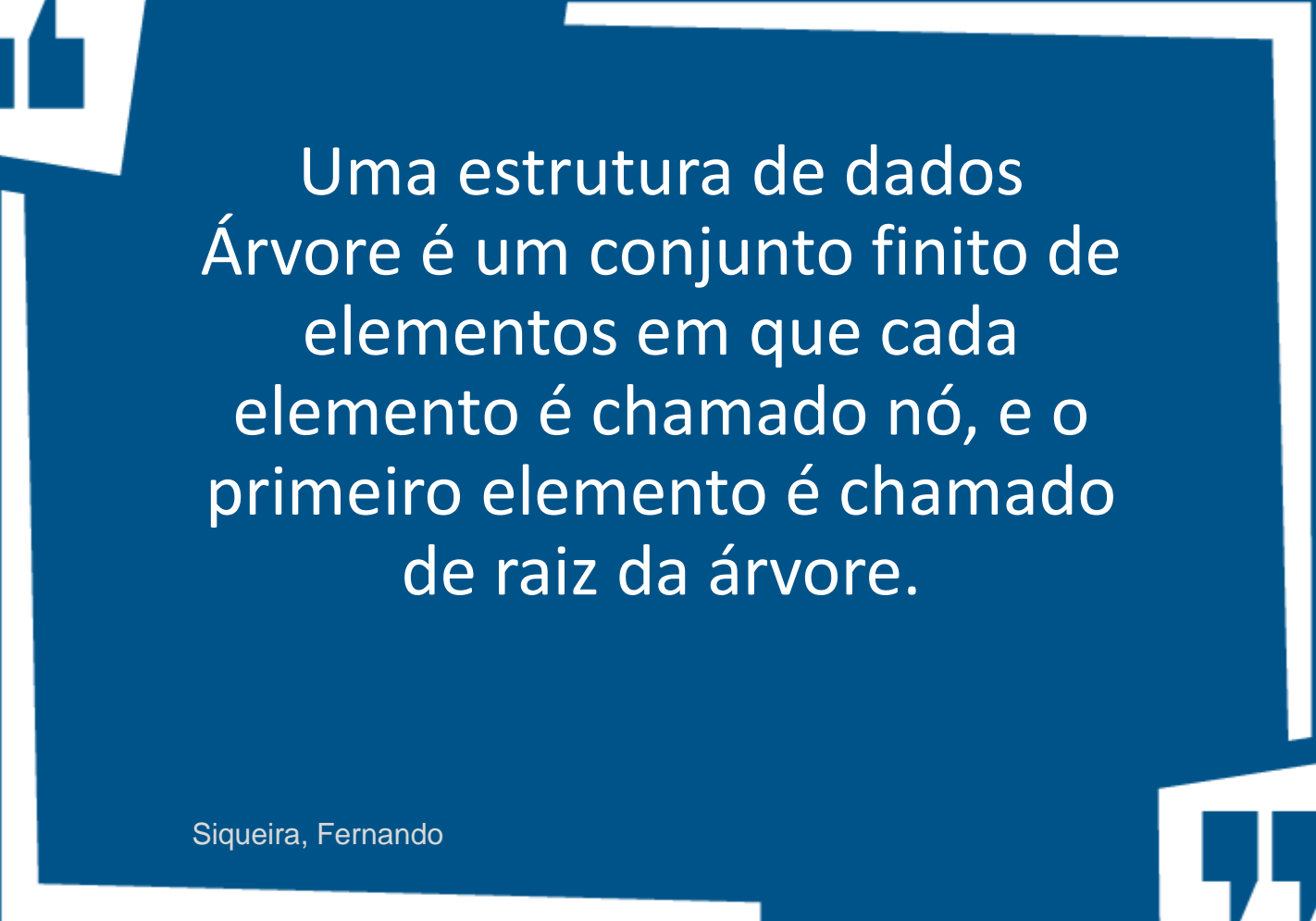



# Estrutura de Dados II



# Unidade 2 - Introdução

Prof. Sandro T. Pinto



Uma estrutura de dados  
Árvore é um conjunto finito de  
elementos em que cada  
elemento é chamado nó, e o  
primeiro elemento é chamado  
de raiz da árvore.

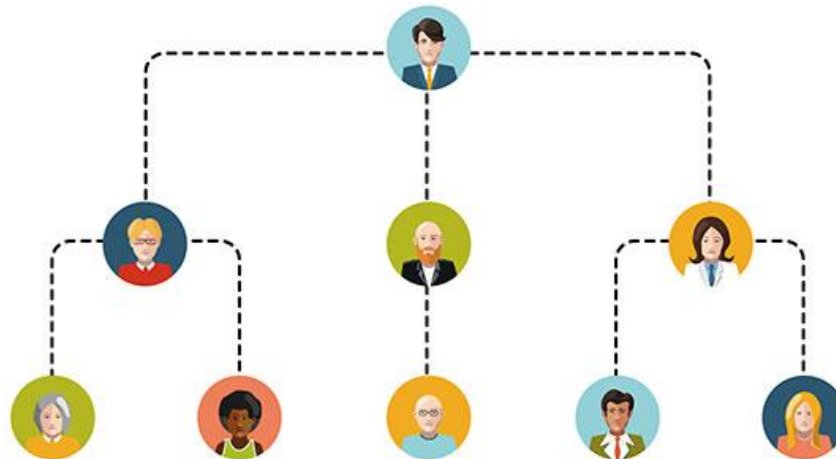
Siqueira, Fernando



## Unidade 2

### Árvores

É uma estrutura de dados que organiza seus elementos de forma **hierárquica**, onde existe um elemento que fica no topo da árvore, chamado de **raiz** e existem os elementos subordinados a ele, que são chamados de **nós filhos**. Cada nós filho pode conter zero, um ou mais de um nós filhos



## Unidade 2

### Árvores

Buscando um elemento:

Imagine que precisamos buscar um elemento (12) no conjunto abaixo. De que forma faremos?

3	5	8	12	25	28	35
---	---	---	----	----	----	----

Primeira forma é buscar elemento por elemento

3	5	8	12	25	28	35
↑	↑	↑	↑			



## Unidade 2

### Árvores

Agora e se pretendermos buscar o valor (13)?

3	5	8	12	25	28	35
---	---	---	----	----	----	----

Não tem na lista, mas ele vai percorrer todos os valores para ai sim apresentar que não encontrou o valor.

(13)



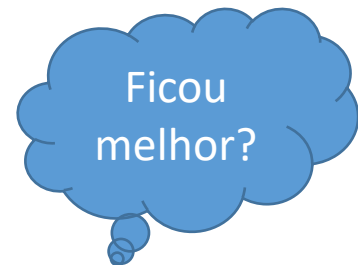
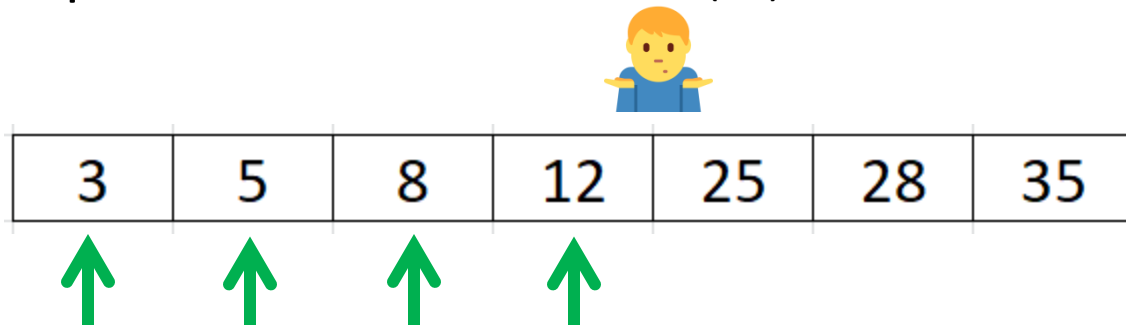
3	5	8	12	25	28	35
↑	↑	↑	↑	↑	↑	↑

## Unidade 2

### Árvores

Veja que no teste anterior precisou percorrer toda a lista para identificar que o valor não foi encontrado. Existe uma forma que podemos utilizar na busca, já que os valores estão ordenados, veja:

Neste caso é verificado sempre a questão do próximo maior, ou seja, o 13 é maior que 25, não então **encerra a busca** e informa que “não encontrou”. (13)



## Unidade 2

### Árvores

Agora e se o número for depois do último como por exemplo (36), como ficaria? Ou seja, ficaria na mesma situação do início, certo?

3	5	8	12	25	28	35
↑	↑	↑	↑	↑	↑	↑

(36)



Tem como melhorar?



## Unidade 2

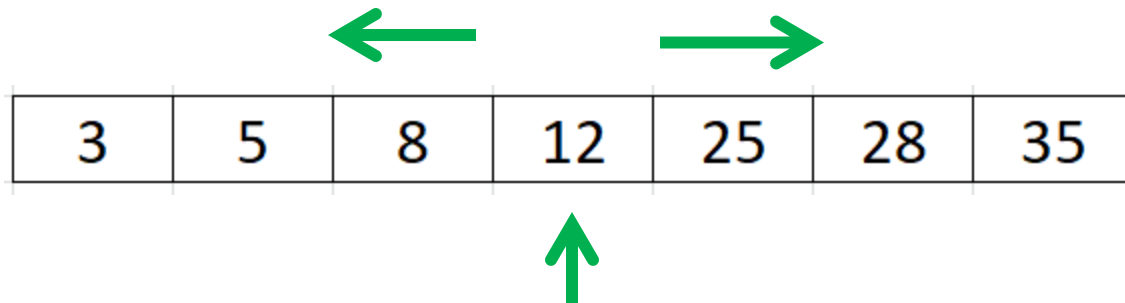
### Árvores



Sim.... E como?

Com a **busca binária**.....

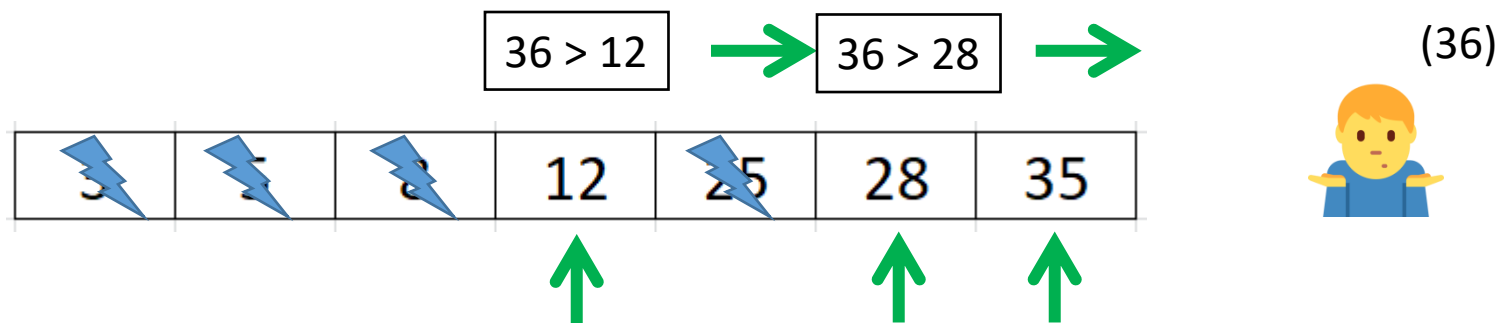
A busca binária é um eficiente **algoritmo** para encontrar um item em uma **lista ordenada** de itens. Ela funciona **dividindo** repetidamente pela metade a porção da lista que deve conter o item, até reduzir as localizações possíveis a apenas uma.



## Unidade 2

### Árvores

Como desejamos encontrar o número (36), na busca binária, ela seleciona o valor do meio, neste exemplo o (12), e verifica que o **número é maior**, portanto deve percorrer para a direita, descartando a possibilidade dos valores a esquerda.

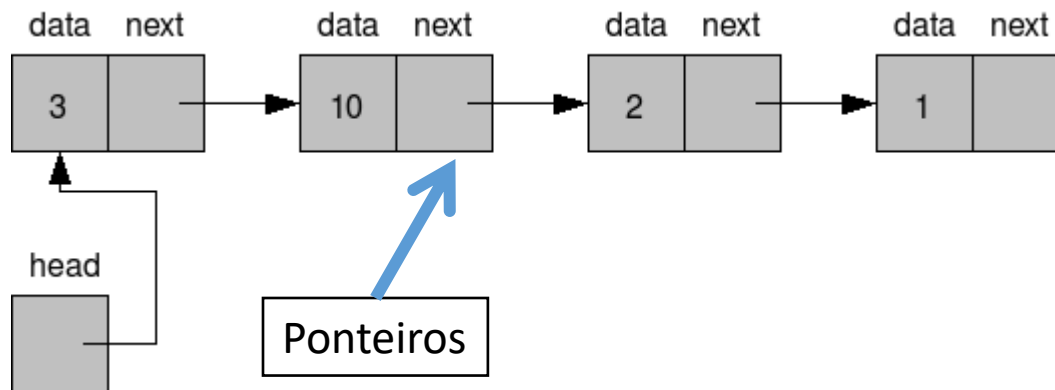


Após esta separação o algoritmo seleciona o meio da **segunda metade** ou seja o valor (28) e verifica novamente, e assim em diante.....

## Unidade 2

### Árvores

Neste exemplo da busca binária funciona bem, porque temos um arranjo organizado na sequência na memória, mas e se for uma **lista ligada** por exemplo:



Não vai ser possível, porque quem é o valor do meio? Não sei.....

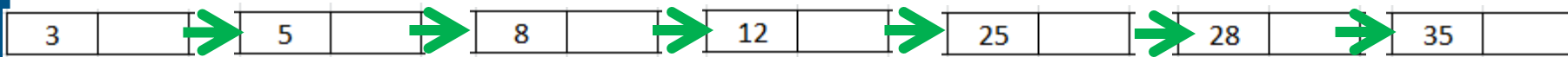


## Unidade 2

### Árvores

Uma lista encadeada ou **lista ligada** é uma estrutura de dados linear e dinâmica. Ela é composta por várias células que estão interligadas através de ponteiros, ou seja, cada célula possui um ponteiro que aponta para o endereço de memória da próxima célula.

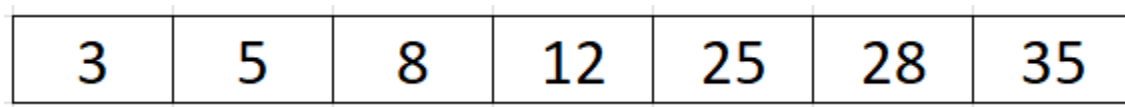
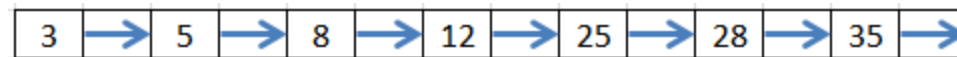
Desse modo, as células da estrutura **não precisam estar em posições contíguas da memória**. Isso faz com que a estrutura se torne dinâmica, pois há qualquer momento, é possível alocar uma nova célula e mudar os ponteiros das células já existentes, de modo que a nova célula seja inserida na estrutura com êxito, na posição desejada pelo programador.



## Unidade 2

### Árvores

Para resolver este problema precisamos de uma estrutura que não necessariamente precisa estar no meio, apenas selecionamos uma e a partir dela vamos dividindo veja:

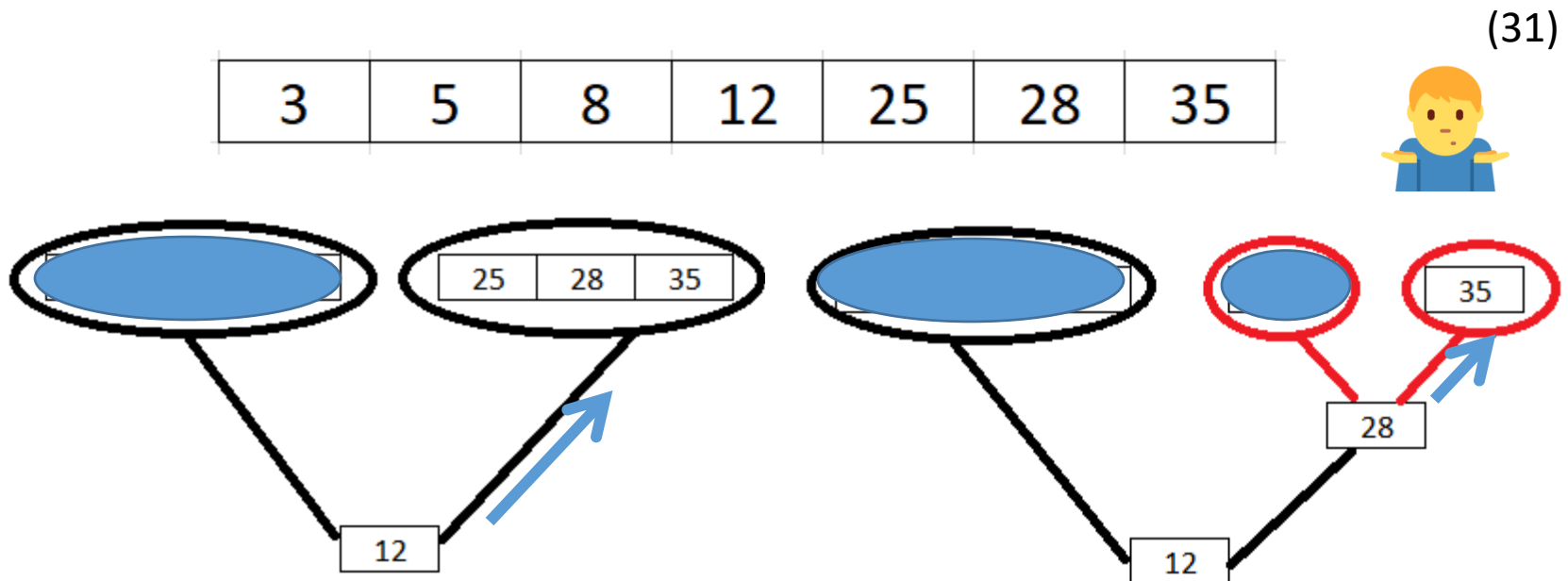


Para simplificar vou deixar sem os ponteiros

## Unidade 2

### Árvores

Para resolver separamos o (12), e como estamos buscando o número (31), identificamos que ele se encontra do lado direito:

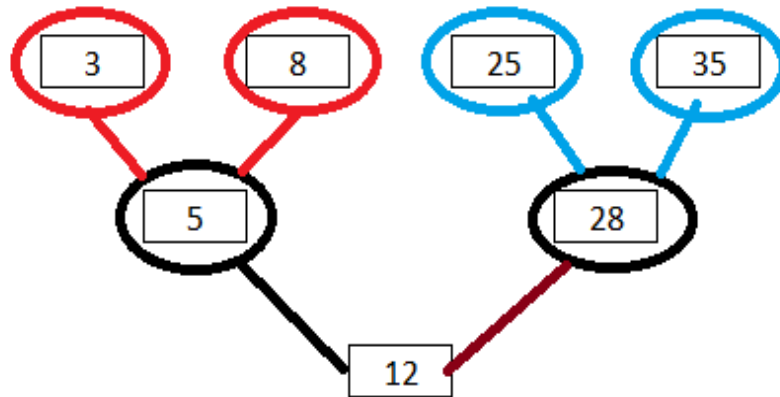


## Unidade 2

### Árvores

A partir disso podemos observar que temos uma estrutura de árvore.

3	5	8	12	25	28	35
---	---	---	----	----	----	----

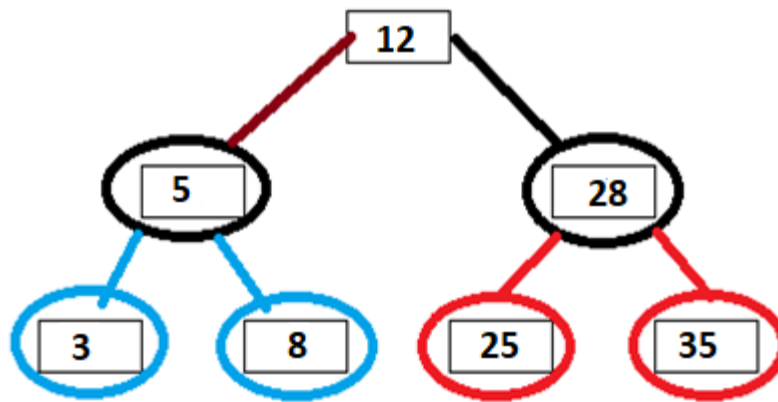


## Unidade 2

### Árvores

Só que na computação temos o costume de ver a árvore invertida.

3	5	8	12	25	28	35
---	---	---	----	----	----	----



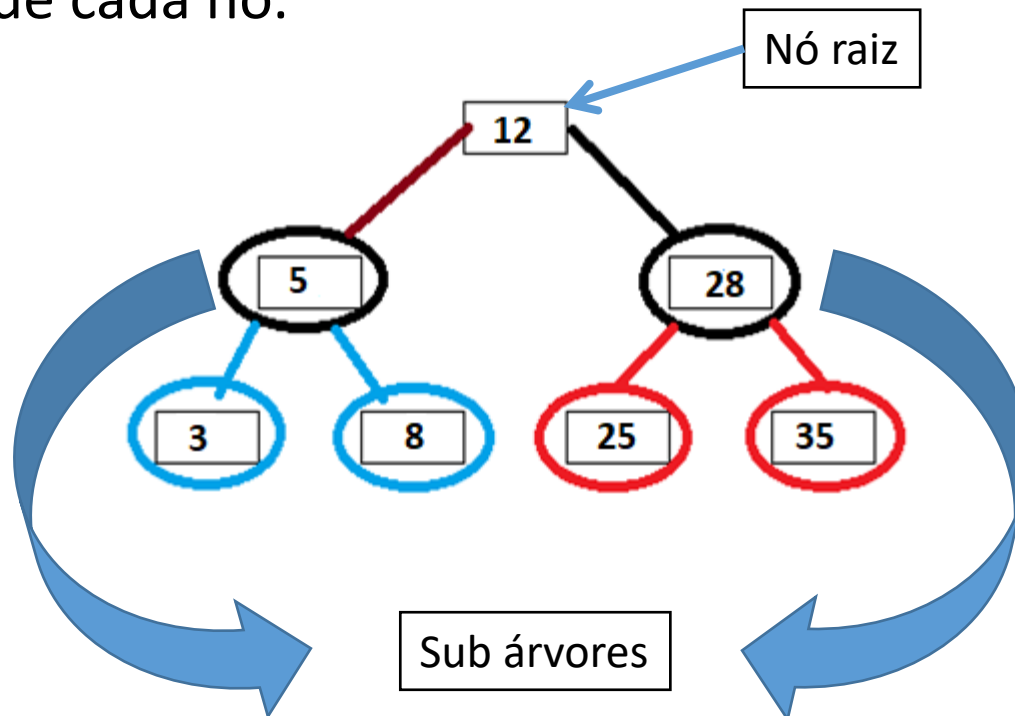


## Unidade 2

### Árvores

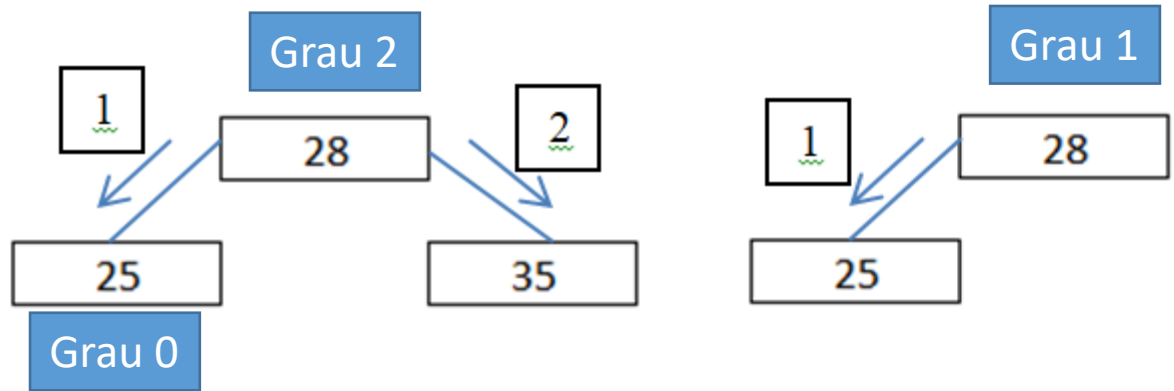
```
//Estrutura  
Struct str_no {  
    char dado;  
    int esquerda;  
    int direita;  
    int pai;  
}
```

O que é uma árvore: é um conjunto de nós, partindo de um nó denominado de raiz, composto de sub árvores que ficam embaixo de cada nó.



## Unidade 2

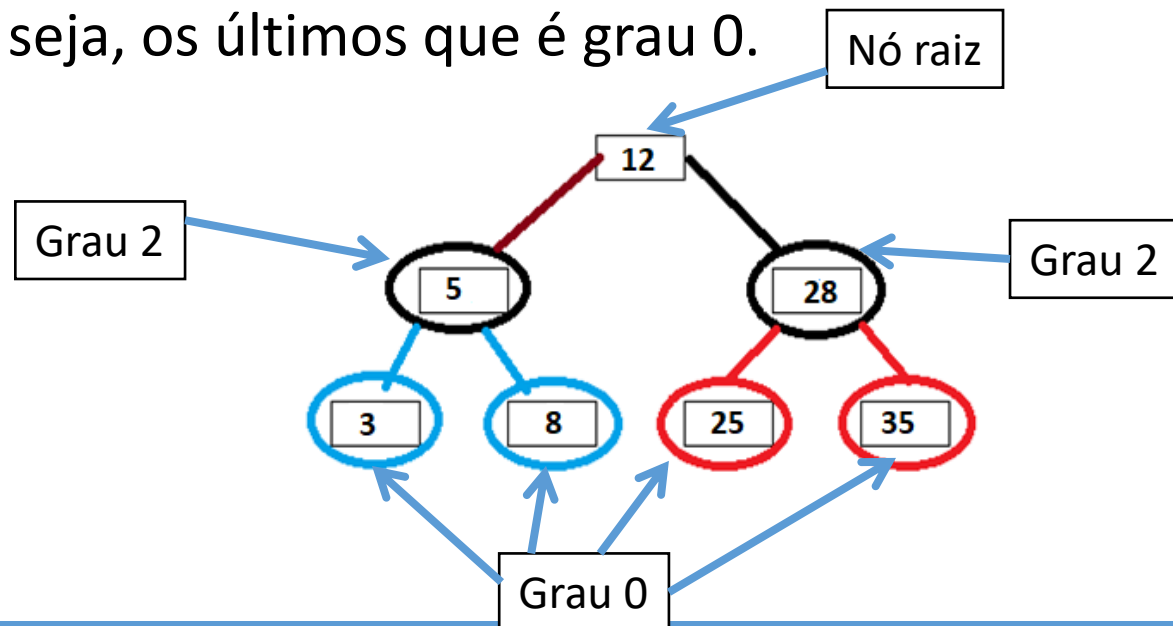
### Árvores



Definições da estrutura da árvore:

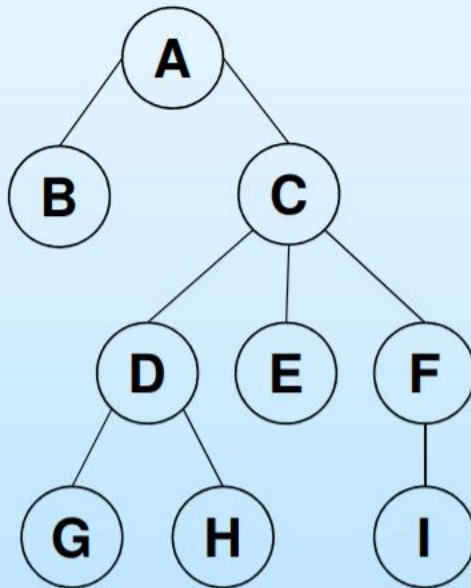
**Grau:** é o número de sub árvores de cada nó possui.

Podemos dizer que todos os nós da Figura são de grau 2 menos as bases, ou seja, os últimos que é grau 0.



# Unidade 2

## Árvores



- Graus dos nós

- $G(A)=2$

- $G(B)=0$

- $G(C)=3$

- $G(D)=2$

- $G(E)=0$

- $G(F)=1$

- $G(G)=0$

- $G(H)=0$

- $G(I)=0$

**Nós Internos**

**Folhas**

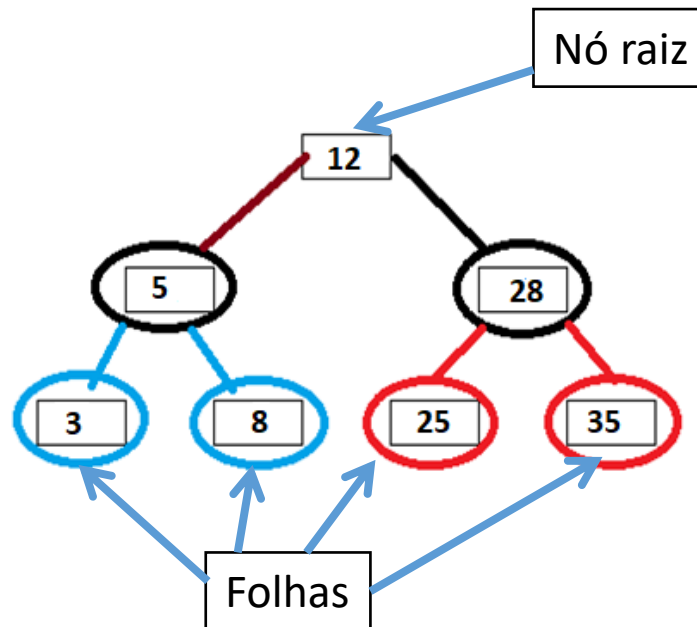
**Grau(T) = 3**

# Unidade 2

## Árvores

Definições da estrutura da árvore:

**Folhas:** ou nós externos, são nós de grau zero.



# Unidade 2

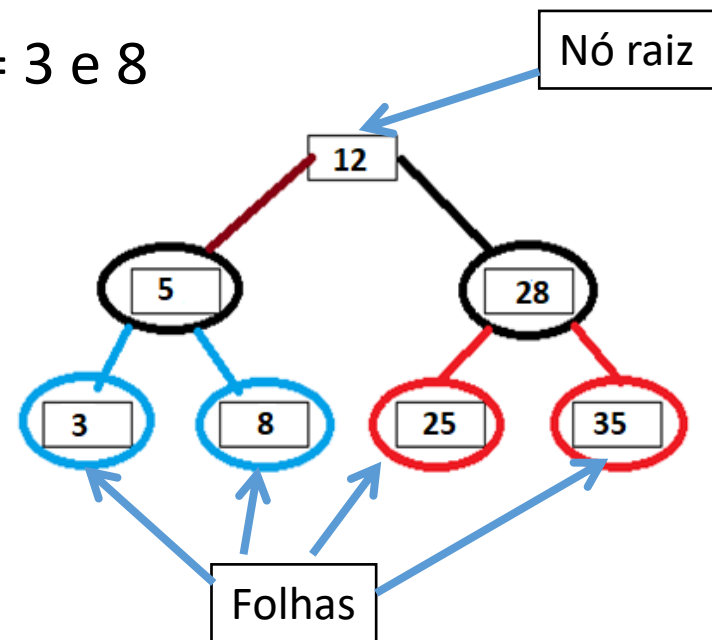
## Árvores

Definições da estrutura da árvore:

**Descendentes:** são nós logo abaixo de um nó.

Exemplo: Descendente do 5 = 3 e 8

Descendente do 12, todos.

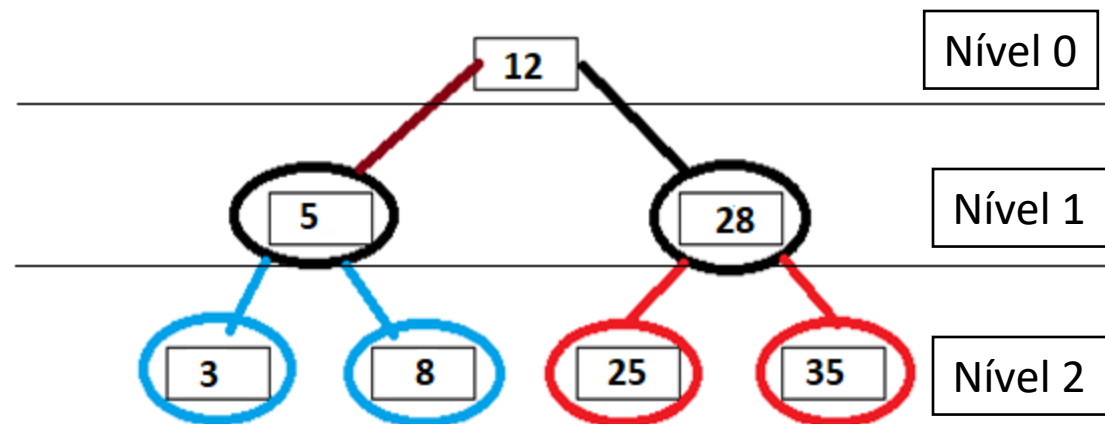


## Unidade 2

### Árvores

Definições da estrutura da árvore:

**Nível:** é o alinhamento do nós, começando pelo nó raiz com 0.

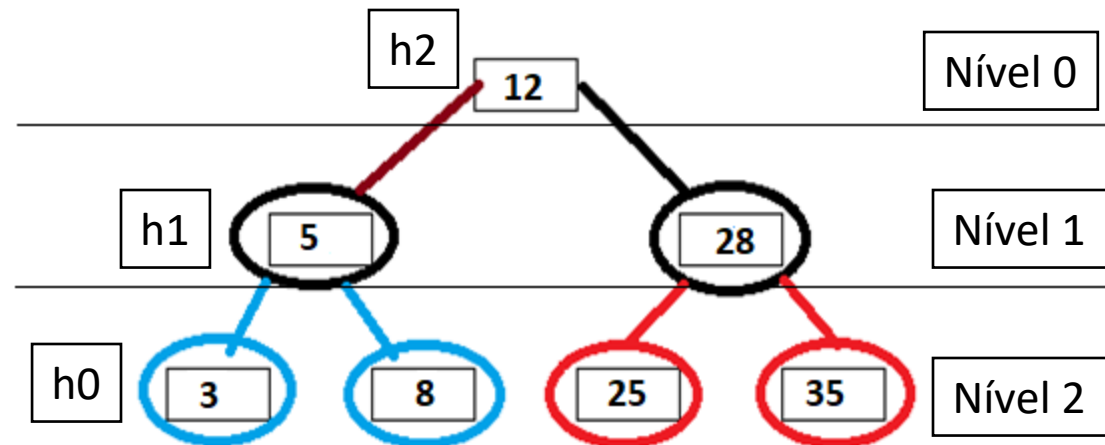


# Unidade 2

## Árvores

Definições da estrutura da árvore:

**Altura(h):** é o contrário do nível, ou seja, a altura(h) de um nó é o comprimento do caminho mais longo entre ele e uma folha.



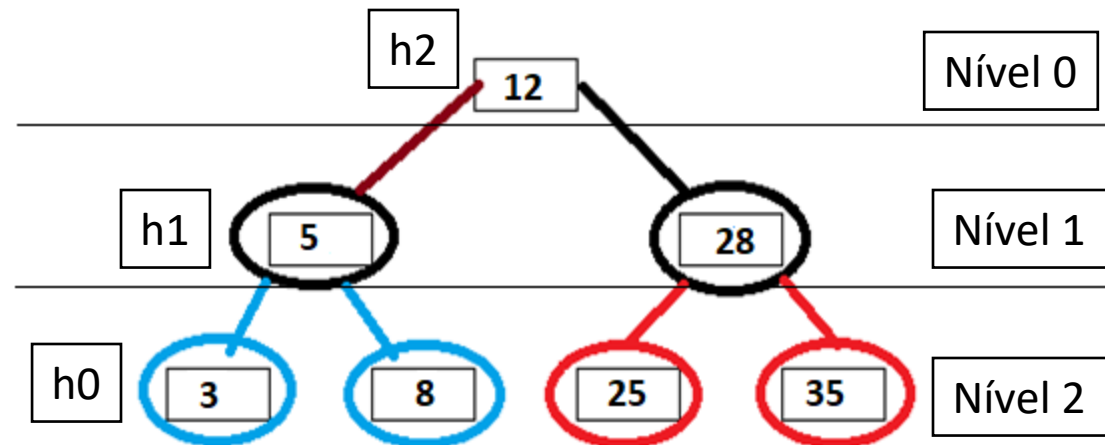
# Unidade 2

## Árvores

Definições da estrutura da árvore:

Neste caso toda a **árvore está balanceada**, ou seja, todos os nós com duas folhas, estão em mesmo nível e altura.

Exemplo: 5 e 28 que são sub árvores do 12, nível 1 e h1.



Más podemos ter uma árvore **não balanceada**.

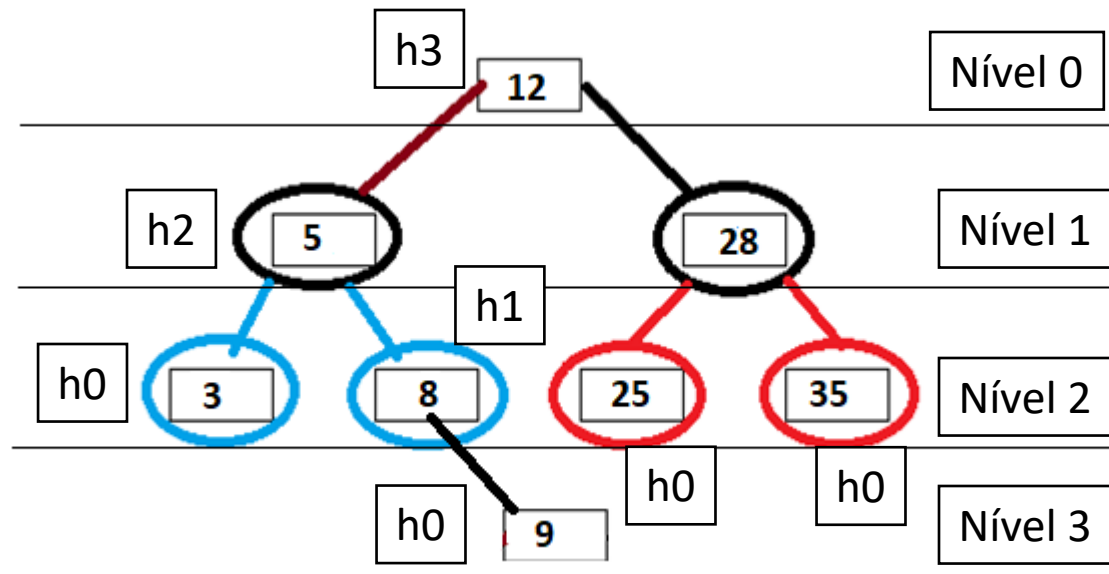


## Unidade 2

### Árvores

Definições da estrutura da árvore:

Mesmo tendo uma árvore não balanceada, ainda assim, as definições de nível e altura ainda são consideradas. A altura é sempre considerada pelo caminho mais longo.

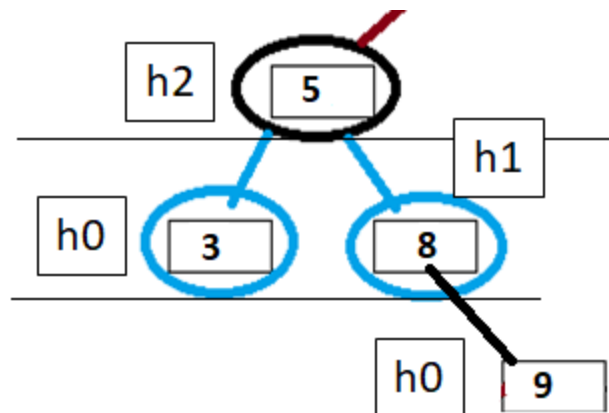


## Unidade 2

### Árvores

Definições da estrutura da árvore:

Veja no exemplo na sub árvore com o nó 5, ela tem dois caminhos só que o caminho para o nó 3 é uma folha, portanto  $h_0$ , já o caminho para o nó 8, como ele tem um novo caminho para o nó 9, o nó 8 é  $h_1$  e o nó 9  $h_0$ , sendo assim, a altura do nó 5 é 2, porque a **altura é o caminho mais longo**.



## Unidade 2

### Árvores

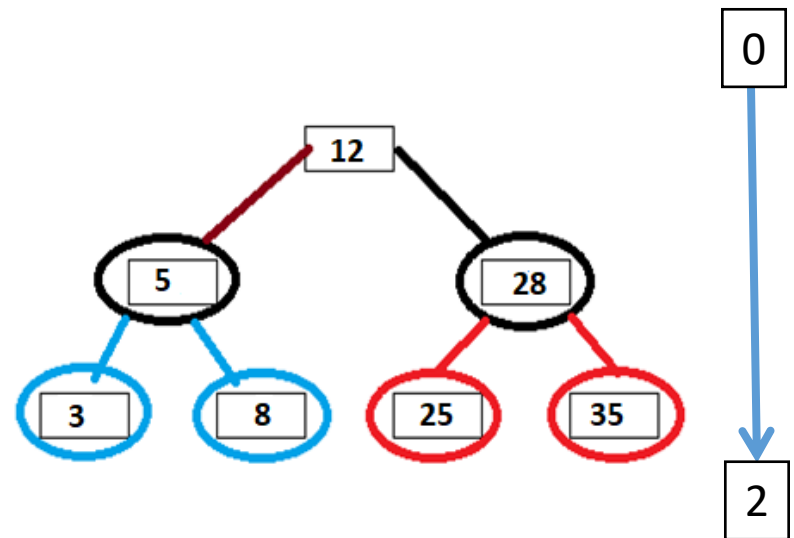
Definições da estrutura da árvore:

**Profundidade:** é a distância percorrida da raiz até ao nó desejado.

Profundidade do nó 12 é 0.

Profundidade do nó 5 é 1

Profundidade do nó 3 é 2



## Unidade 2

### Árvores

Portanto tudo em uma árvore é a partir de sua raiz, sendo assim a altura de uma árvore é altura da raiz, da mesma maneira **o endereço de uma árvore na memória, é o endereço de seu nó raiz**. Uma função não recebe uma árvore ela recebe a raiz.

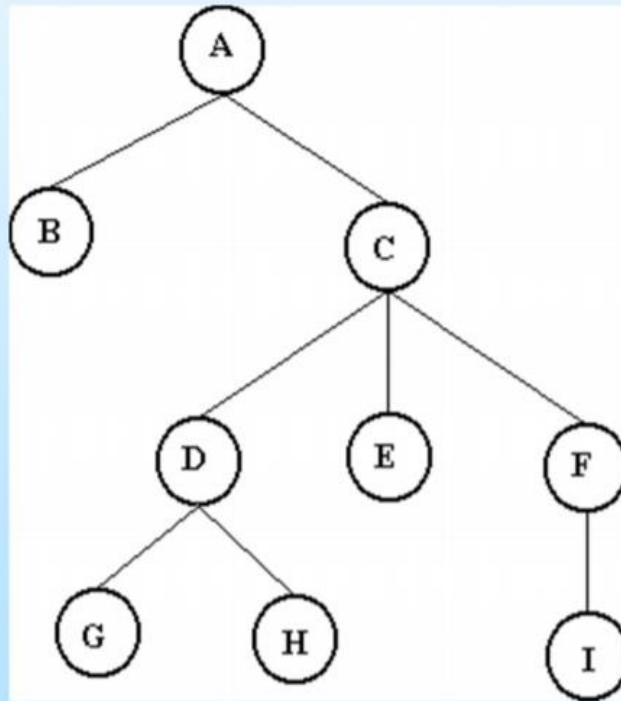
Isso porque a partir da raiz conseguimos acessar toda a estrutura da árvore.

Sendo assim a raiz é a parte principal da árvore.

## Unidade 2

### Árvores – Alinhamento dos nós

- Hierárquica



- Alinhamento dos nós

**A**  
**B**  
**C**  
  
**D**  
  
**E**  
**F**  
  
**G**  
**H**  
  
**I**

## Unidade 2

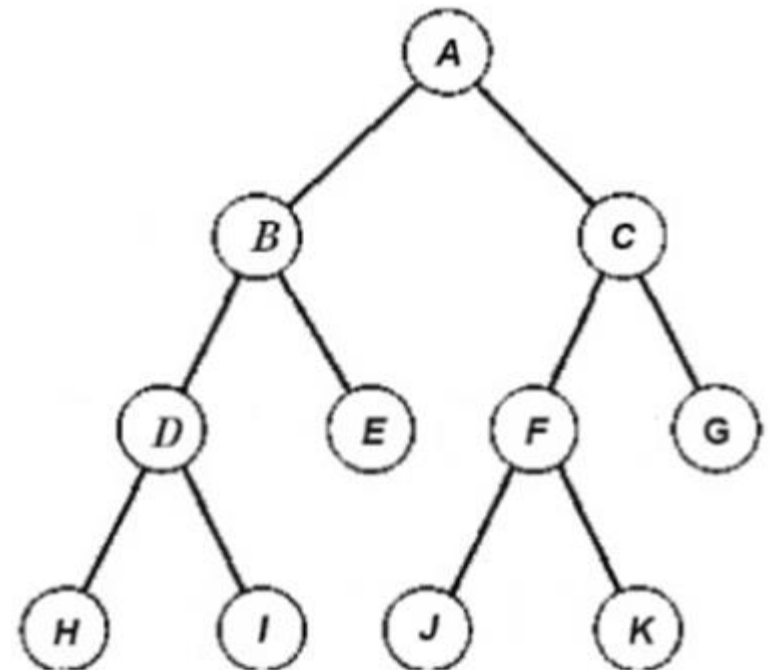
### Árvores – Atividade 1

Altura: Vai 0 a 3

Nós: 11

Níveis: Vai de 0 a 3

Folhas: 6



## Unidade 2

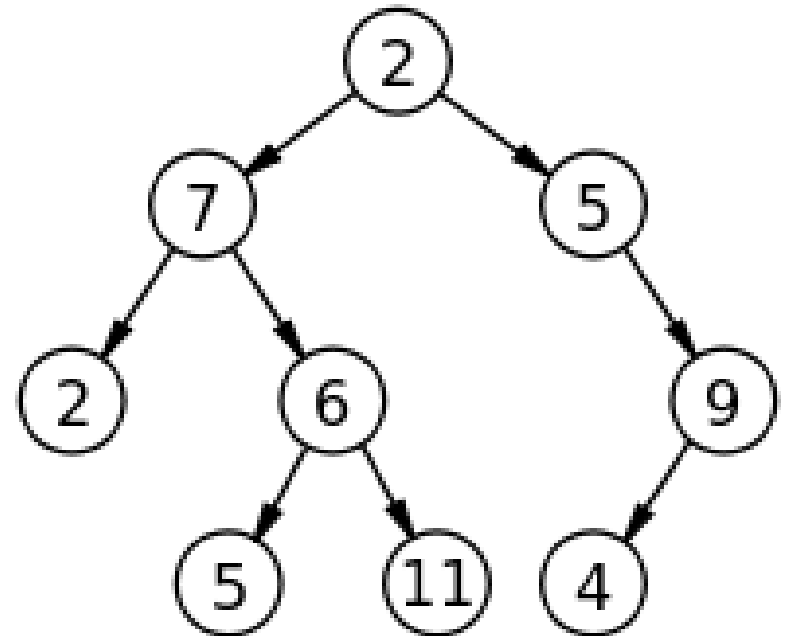
### Árvores – Atividade 2

Altura: Vai 0 a 3

Nós: 9

Níveis: Vai de 0 a 3

Folhas: 4



## Unidade 2

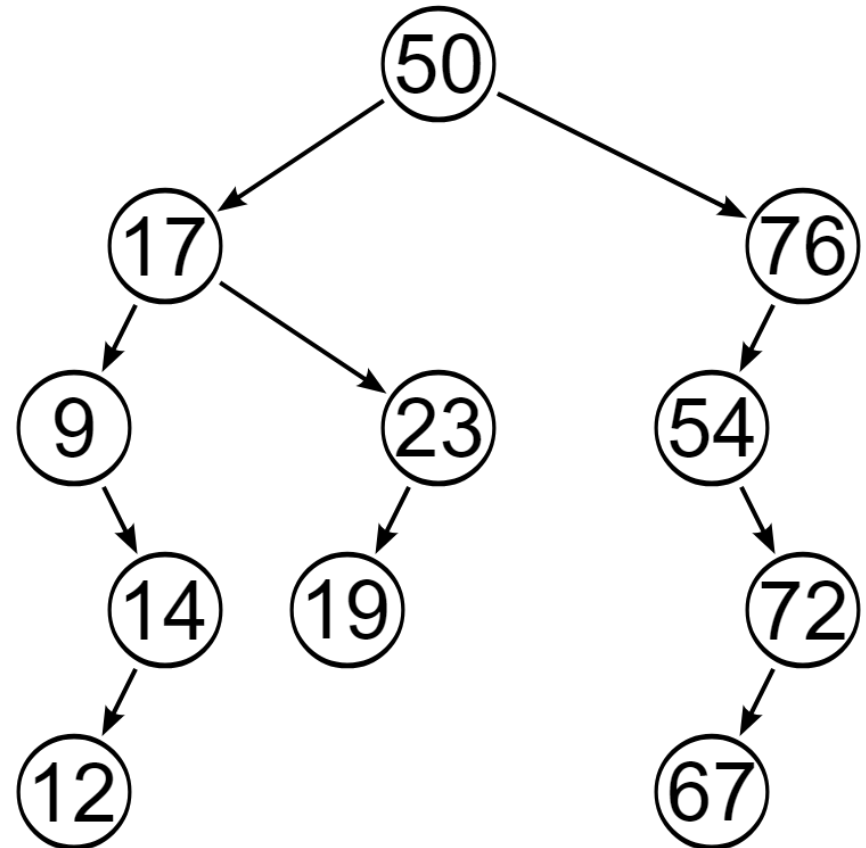
### Árvores – Atividade 3

Altura: Vai 0 a 4

Nós: 11

Níveis: Vai de 0 a 4

Folhas: 3





## Unidade 2

### Árvores Binárias

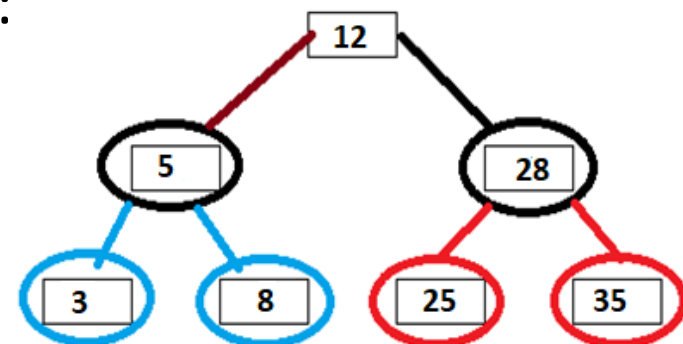
Uma árvore binária é uma árvore que abaixo um nó raiz, existem no máximo duas sub árvores.

Uma árvore binária em que cada nó tem 0 ou 2 filhos.

A quantidade de nós presentes numa árvore estritamente binária se dá pela formula:  $n = (2 * f) - 1$ , onde,  $f$  = número de folhas na árvore e  $n$  é a quantidade de nós.

Para determinar o número de folhas:

$$f = (n+1)/2$$



## Unidade 2

$$n = (2 * f) - 1$$

$$f = (n+1)/2$$

### Árvores – Atividade 4

Uma árvore estritamente binária possui exatamente **4 folhas**, qual a quantidade de nós presentes na árvore?

Resposta: 7 nós.

$$n = (2 * 4) - 1 = 7$$

Uma árvore estritamente binária possui exatamente 7 nós, A | B | C | D | E | F | G. Qual a quantidade de folhas?

Resposta: 4 folhas.

$$f = (7+1)/2 = (8)/2 = 4$$

## Unidade 2

### Árvores Binárias

Como representar uma árvore binária em código?

Unindo os nós

Como representamos os nós?

Criando uma estrutura que guarde a informação necessária para a montagem da árvore, exemplo:

Criar 2 ponteiros: um para sub árvore da esquerda e outra para a sub árvore da direita, e também um campo para chave e os dados.

Chave	
null	null

# Unidade 2

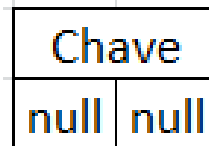
## Árvores Binárias

### Estrutura básica:

Guardamos aqui somente a chave? **não**. Podemos guardar qualquer informação para sua árvore.

Se estamos buscando alguma coisa precisamos de: uma **chave de busca**. Para verificar cada nó.

```
struct noArvore {  
    char info;  
    struct noArvore* esq;  
    struct noArvore* dir;  
};
```



Ponteiros para subir a árvore da esquerda e da direita.

## Unidade 2



## Referências:

SZWARCFITER, Jayme Luiz; MARKENZON, Lilian. Estruturas de dados e seus algoritmos. 2ed. Rio de Janeiro: LTC, 1994. 320p.

TENENBAUM, Aaron M.; LANGSAM, Yedidiah; AUGENSTEIN, Moshé J.. Estruturas de dados usando C. São Paulo: Makron Books, 1995. 884p.

VELOSO, Paulo et al.. Estruturas de dados. Rio de Janeiro: Campus, 2001. 228p

## Atividades - Unidade 2

