

INSTITUTO FEDERAL

Norte de Minas Gerais

Campus Januária

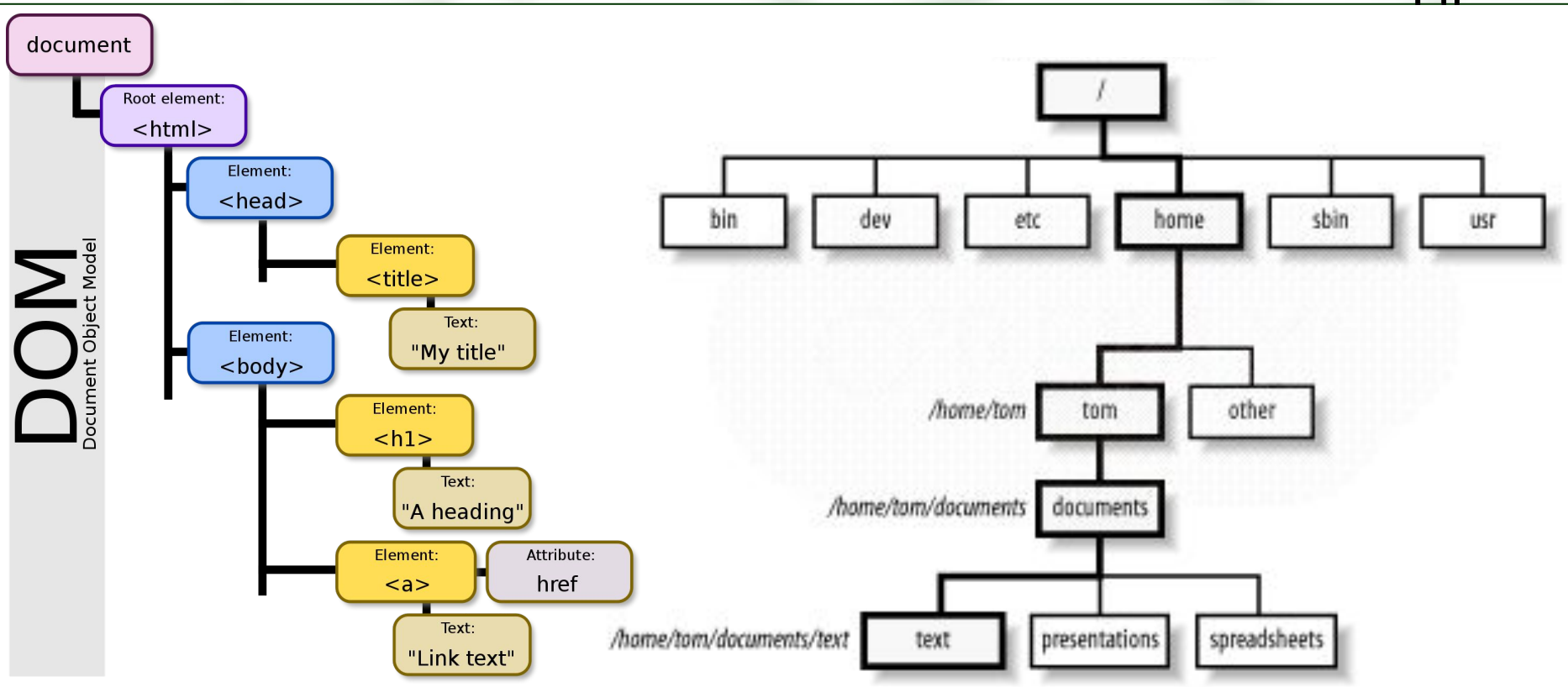
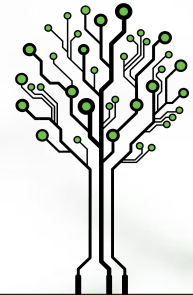
Estruturas de Dados 2

- Árvores -



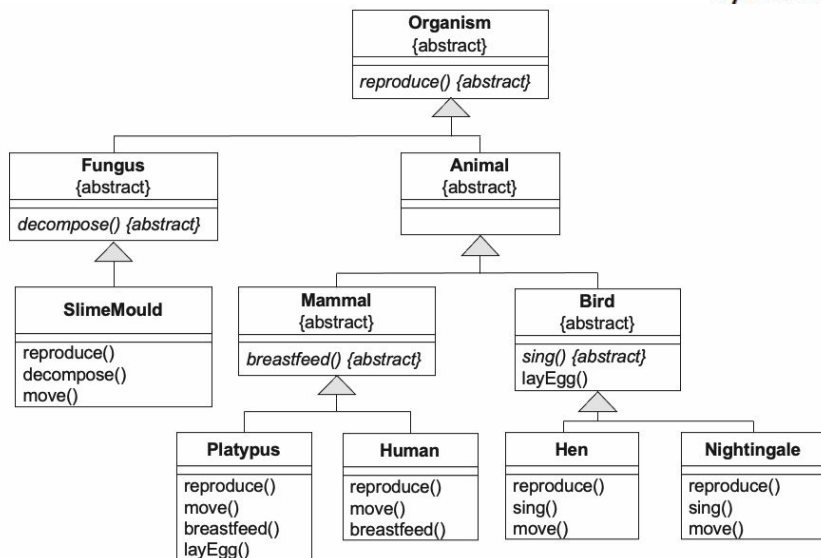
Árvores

- **Árvores** são uma das **Estruturas de Dados** mais importantes da computação...





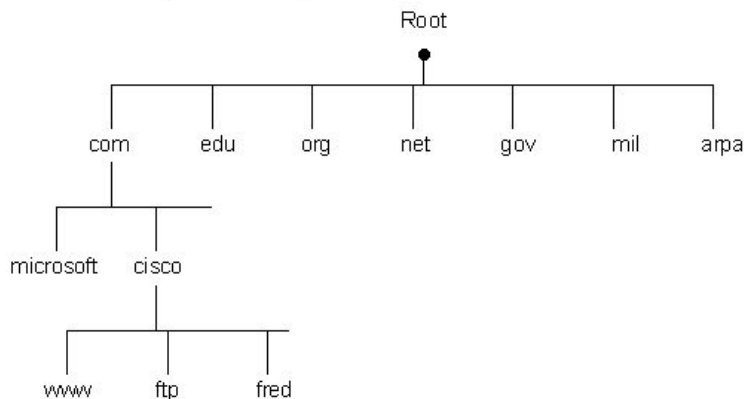
Árvores



\$> pstree

```

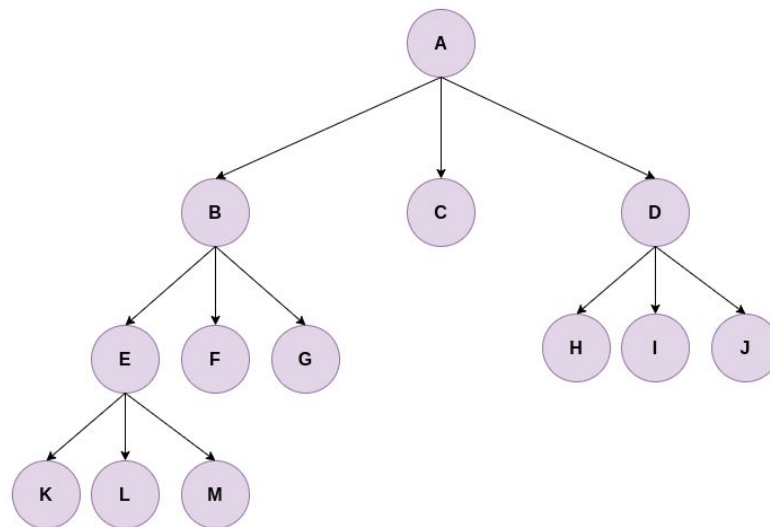
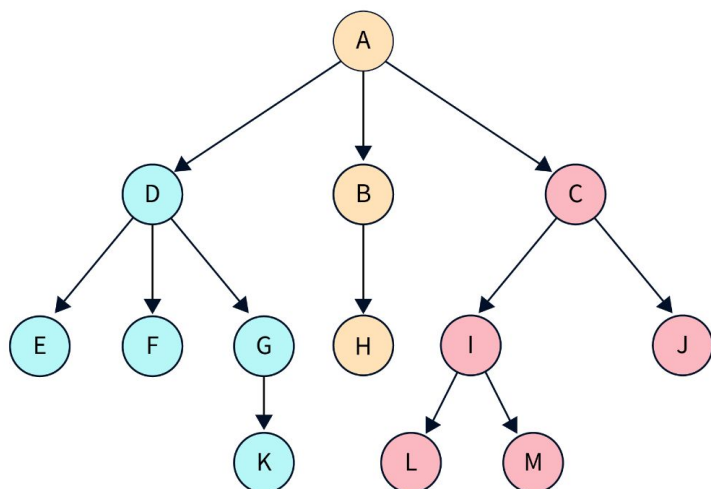
systemd--ModemManager--2*[{ModemManager}]
--NetworkManager--2*[{NetworkManager}]
--accounts-daemon--2*[{accounts-daemon}]
--acpid
--agetty
--apache2--2*[apache2--26*[{apache2}]]
--avahi-daemon--avahi-daemon
--blkmapd
--chrome_crashpad--2*[{chrome_crashpad}]
--chrome_crashpad--{chrome_crashpad}
--colord--2*[{colord}]
--containerd--8*[{containerd}]
--containerd-shim--entrypoint.sh--katharanp
--11*[{containerd-shim}]
--containerd-shim--entrypoint.sh--katharanp
--10*[{containerd-shim}]
--core--13*[{core}]
--core--wsatspi
--13*[{core}]
--cron
--csd-printer--2*[{csd-printer}]
--cups-browsed--2*[{cups-browsed}]
    
```





Árvores

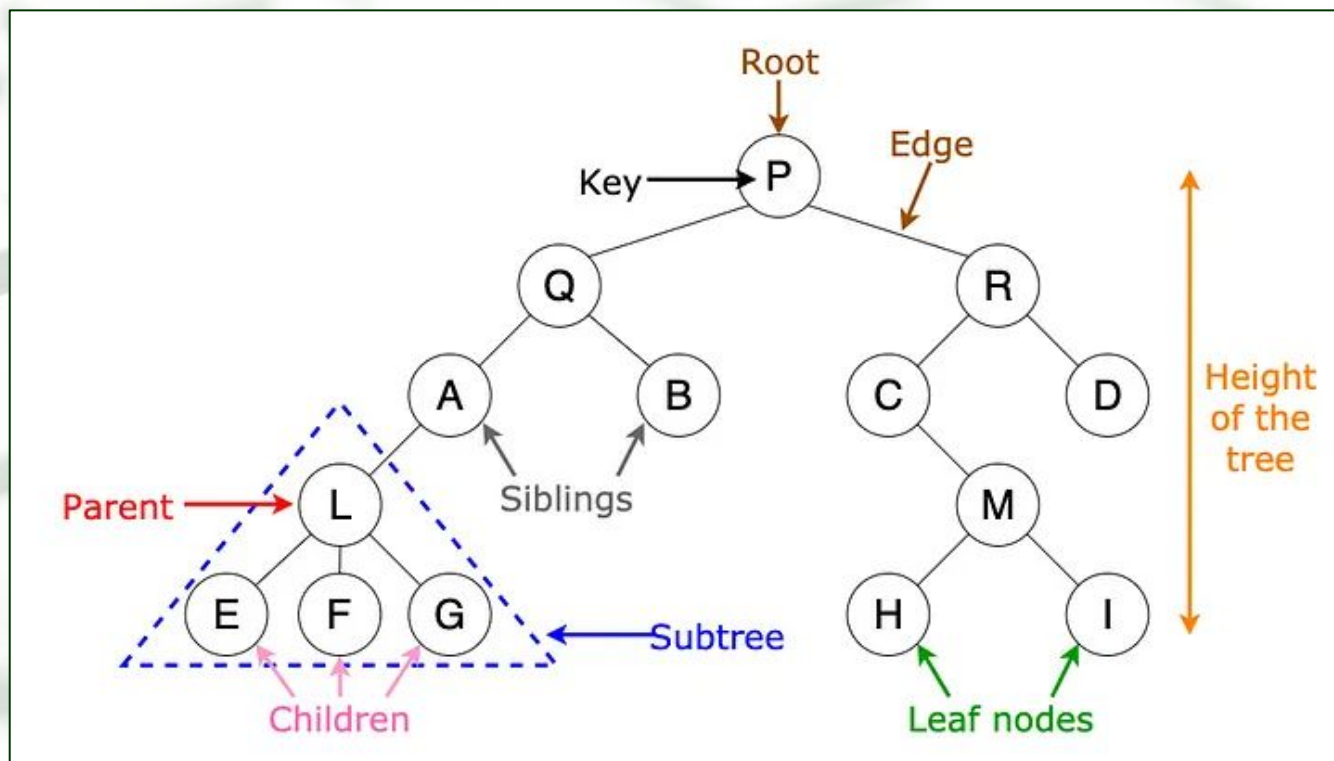
- Pode ser vista como uma **extensão de uma Lista Encadeada**, mas onde cada nó pode ter mais de um **sucessor**, neste caso, chamados de **nós filhos**.
- Árvores são **Estruturas de Dados não Lineares**.





Árvores

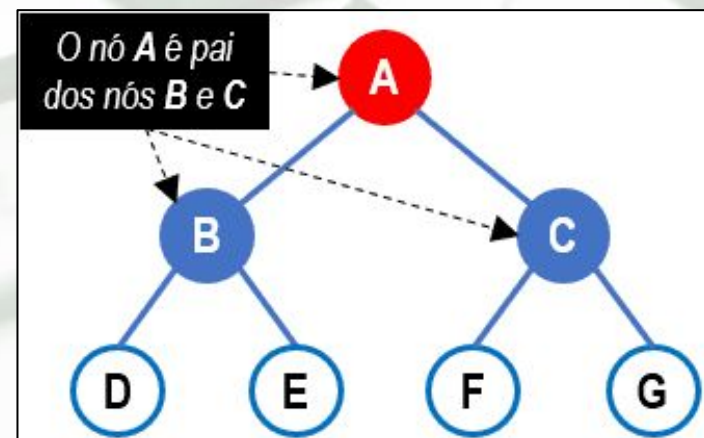
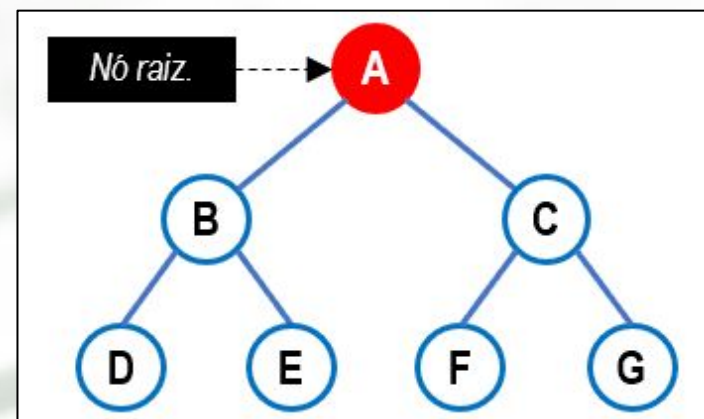
- As Árvores possuem diversas características técnicas que as distinguem entre si. Veremos as mais importantes...





Árvores

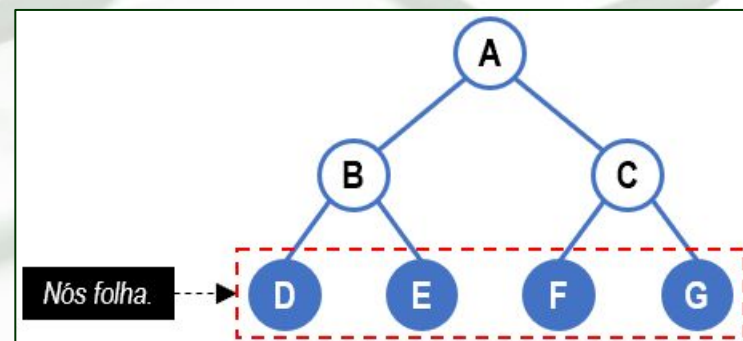
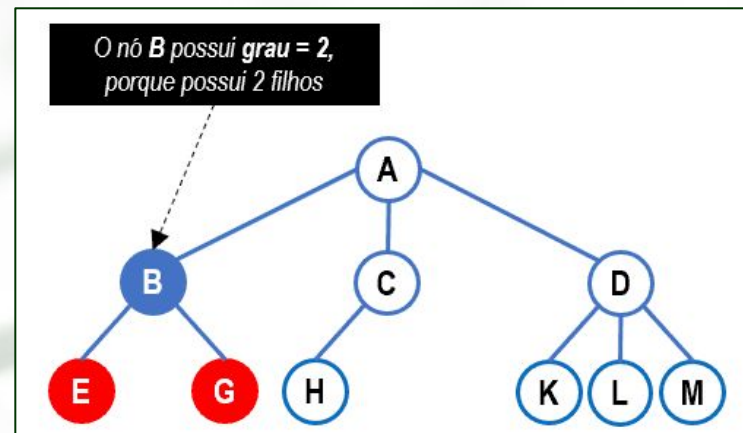
- A representação usual é invertida, **tendo o nó raiz no topo, e as folhas na base.**
- Cada nó possui um único pai (exceto o nó raiz). Cada nó filho por sua vez, pode ser considerada uma sub-árvore (idéia de **recursividade**).
- Todo percurso é **único!** Ao contrário, seria um grafo.





Árvores

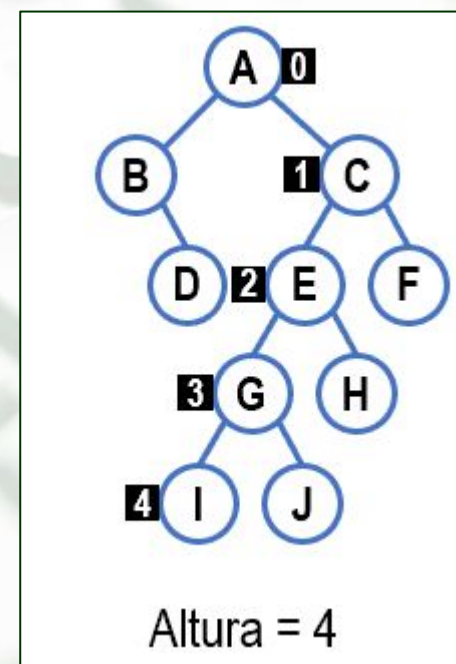
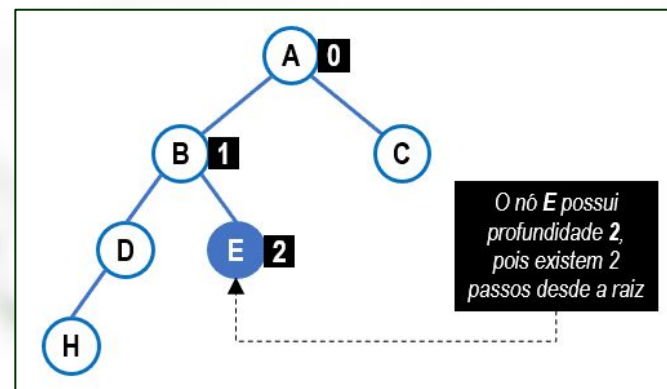
- O número máximo de filhos de um nó determina o **Grau** deste nó (ou da sub-árvore).
- O maior grau dentre todos os nós, determina o **Grau da Árvore** como um todo.
- Nós de grau 0 são chamados de Folhas.





Árvores

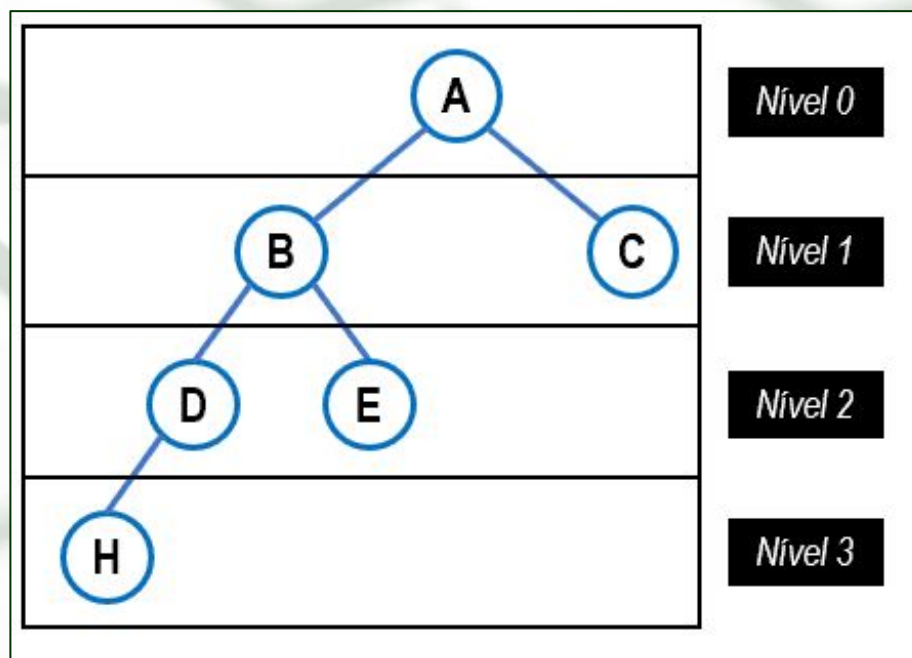
- A **Profundidade de um nó** é a sua distância para alcançar a raiz.
- A **Altura de uma árvore** representa a maior profundidade dentre todos os nós.





Árvores

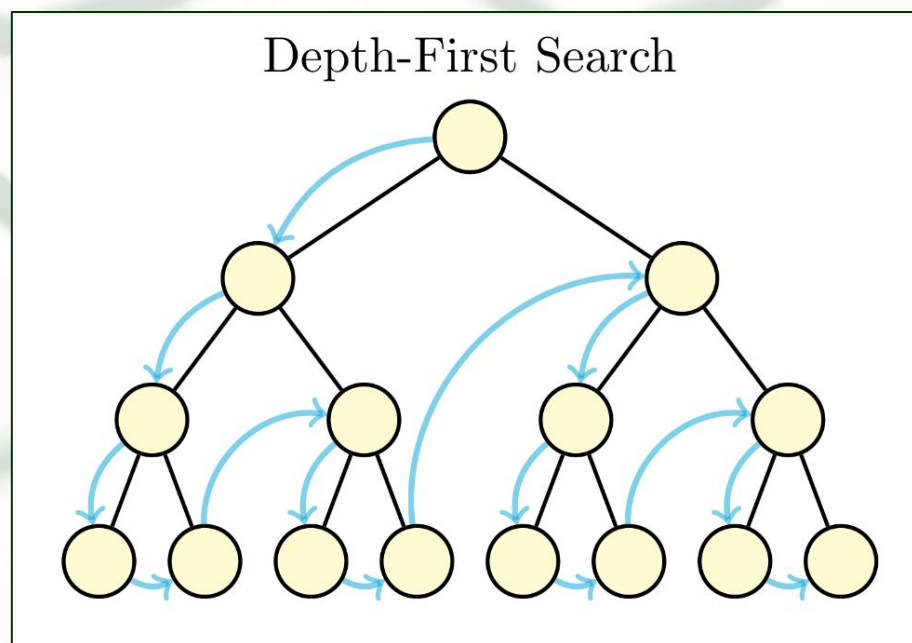
- O **Nível** define as fatias de profundidade de uma árvore, de modo que os filhos de um determinado nó (nós irmãos) sempre estarão no mesmo nível.





Árvores

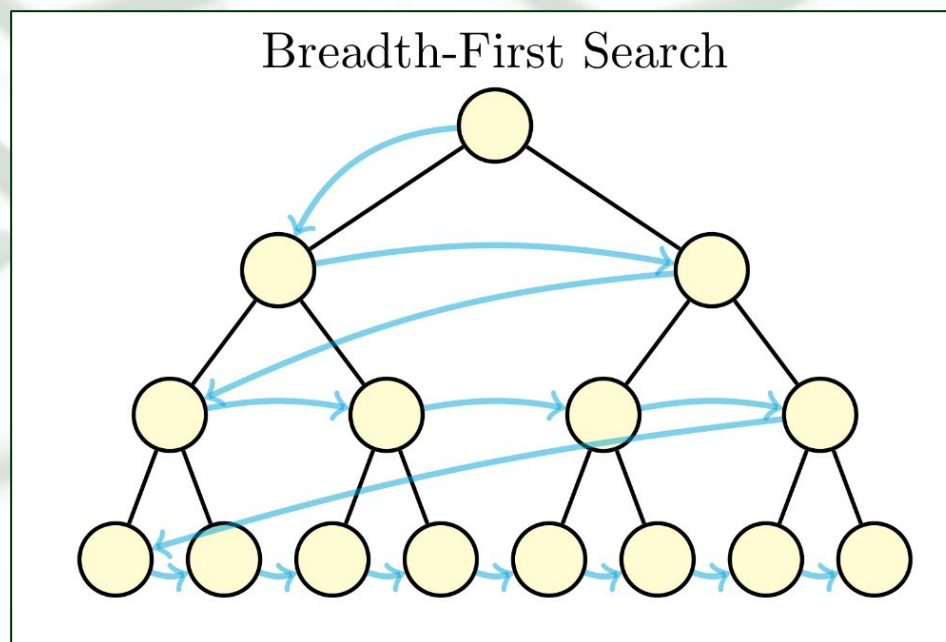
- **A Busca em Profundidade - DFS (*Depth-First Search*)** é o método de travessia em árvore que, começando do nó raiz, explora tanto quanto possível cada uma das sub-árvores, antes de retroceder ao nó pai.





Árvores

- A **Busca em Largura** - BFS (*Breadth-First Search*), ou busca em amplitude, é o método de travessia em árvore que, começando do nó raiz, explora todos os nós de um mesmo nível, antes de ir aos nós filhos.

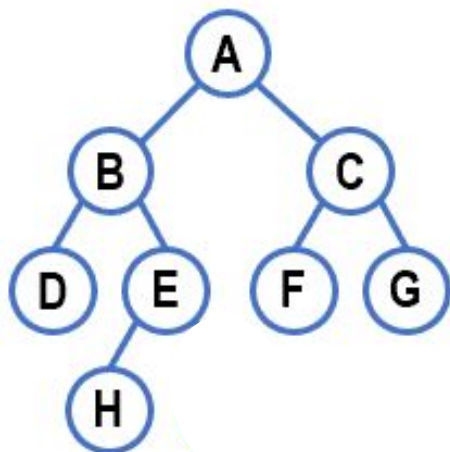




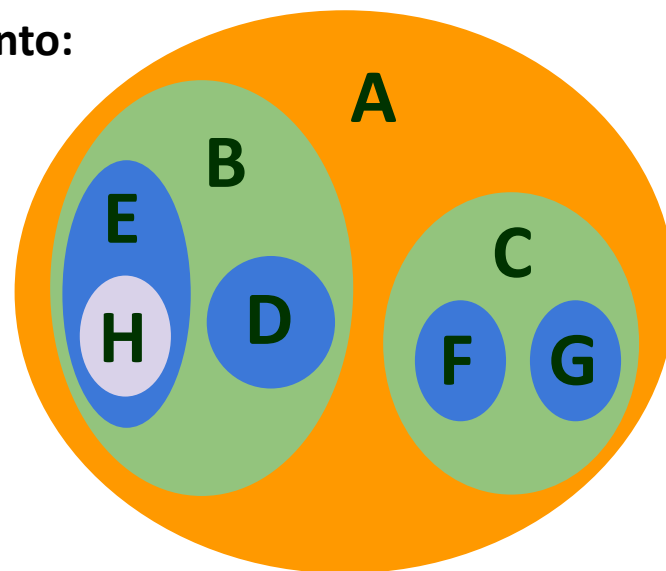
Árvores

■ Outras formas de representação das Árvores...

Grafo:



Conjunto:



Parênteses:

(A (B (D,E (H)), C(F, G)))

Array:

A	B	C	D	E	F	G	/	/	H	/	/	/	/	/
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---



Árvores

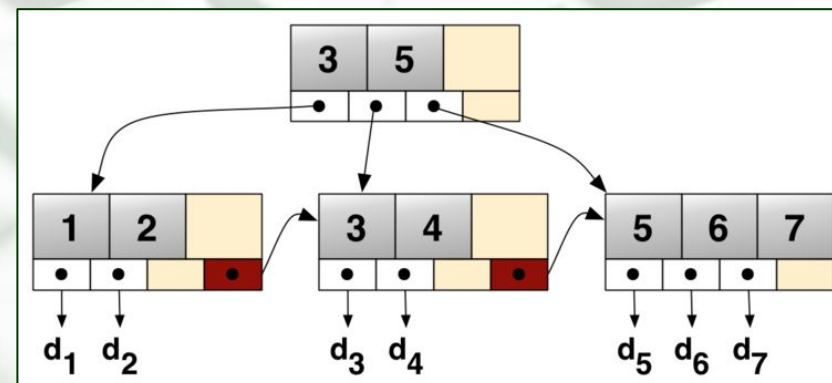
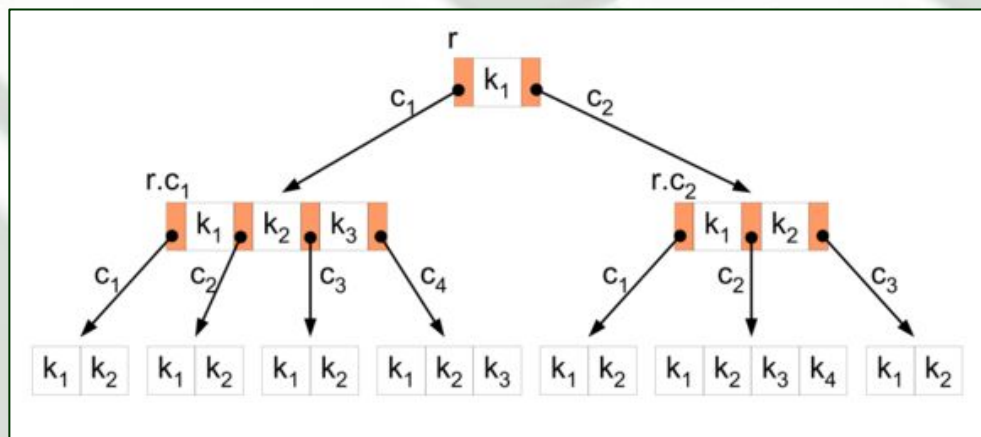
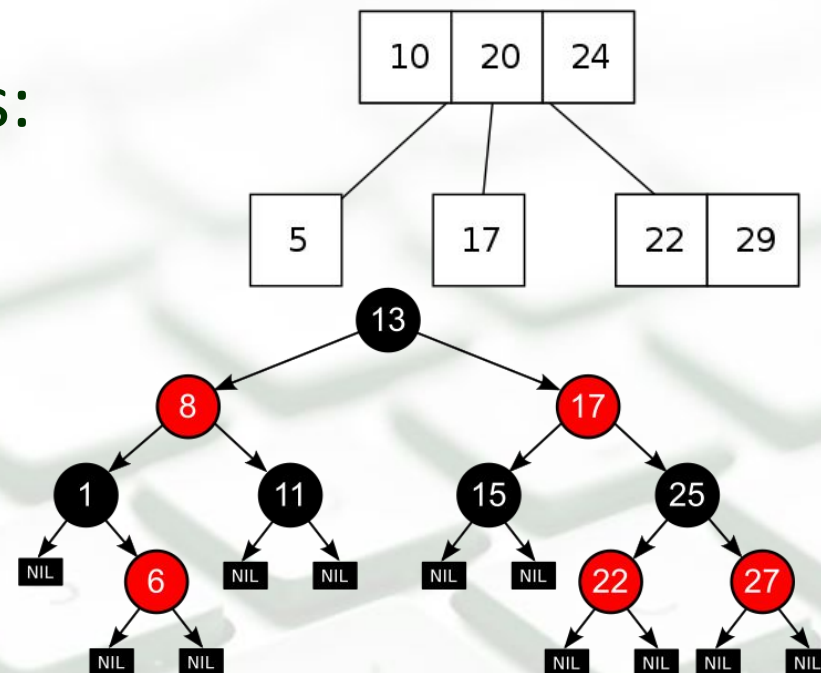
- Como percebemos, existe um grande número de conceitos técnicos inerentes às características das Árvores.
- Da mesma forma, existem **inúmeras variantes de Estruturas de Dados em formato de Árvores**, cada qual, sendo melhor ou pior aplicável a diversos problemas existentes na computação...



Árvores

■ Principais Tipos de Árvores:

- Árvores Binárias (ABB)
- Árvores AVL (Balanceadas)
- Árvores 2-3-4
- Árvores Rubro-Negras
- Árvores B e B+

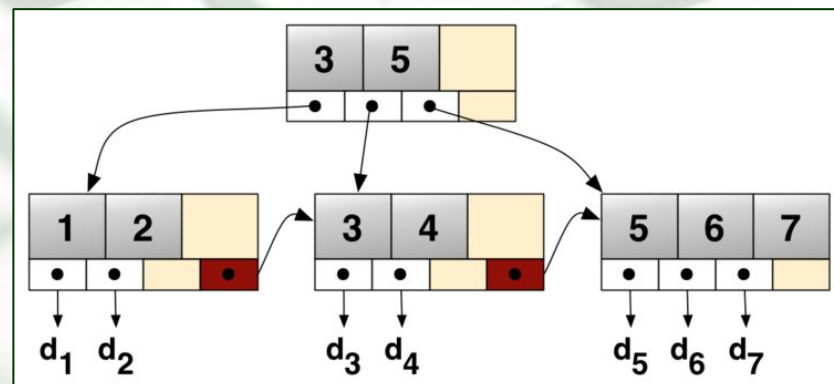
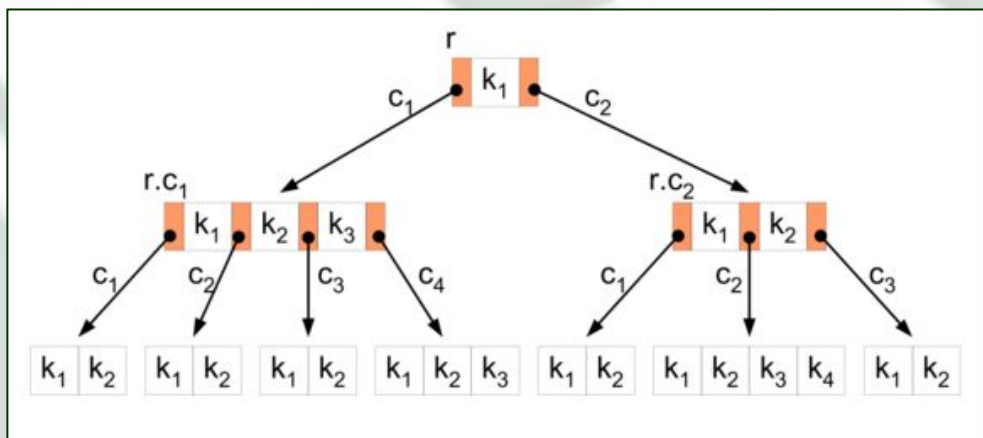
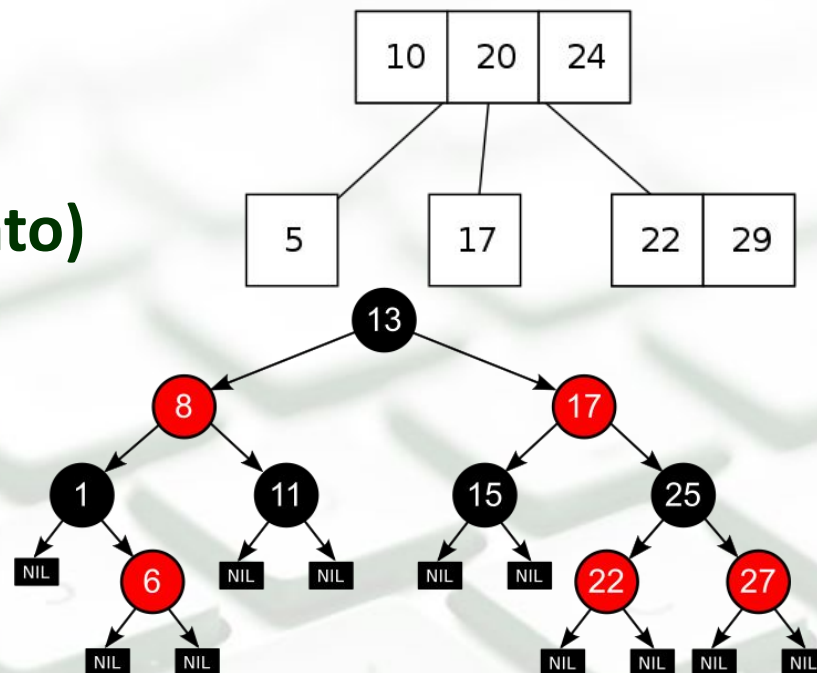




Seminários

■ Temas para Apresentação:

- Árvores AVL (Balanceamento)
- Árvores Splay
- Árvores B e B+
- Árvores Rubro-Negras
- Árvores 2-3-4

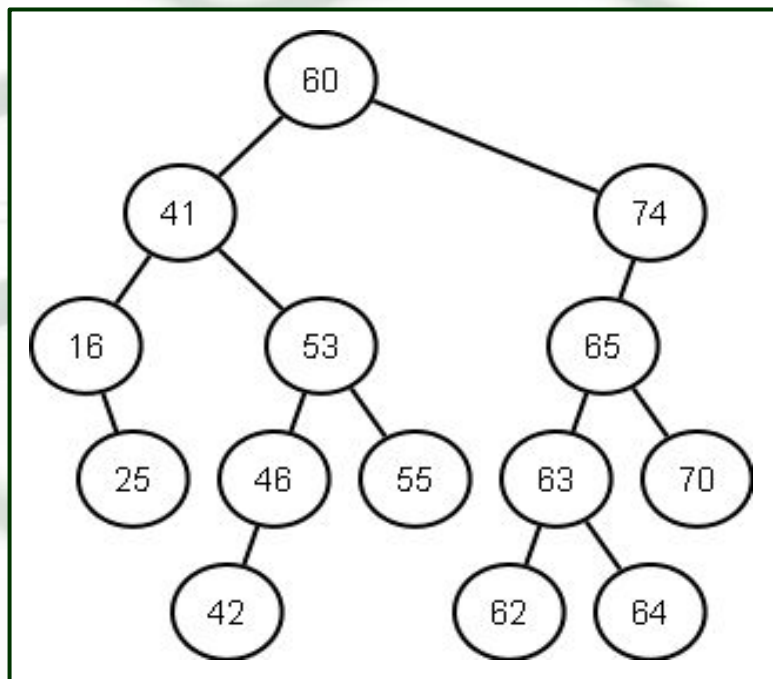




Árvores Binárias

- **Árvores Binárias** são aquelas de grau máximo 2.
- A implementação mais comum das Árvores Binárias são as ABB (**Árvores Binárias de Busca**)

Nas ABB, todos os nós da subárvore esquerda possuem um valor chave (valor de identificação) **inferior à chave do nó raiz**, e todos os nós da subárvore direita possuem um valor de chave **superior à chave do nó raiz**.

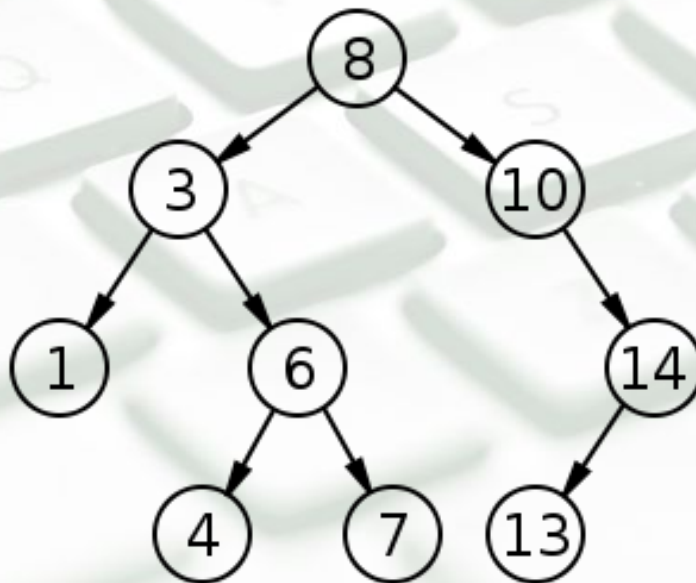




Árvores Binárias

■ Responda rápido...

Qual a **Ordem de Complexidade (Big O)** para métodos de Inserção e Consulta em uma Árvore Binária de Busca?



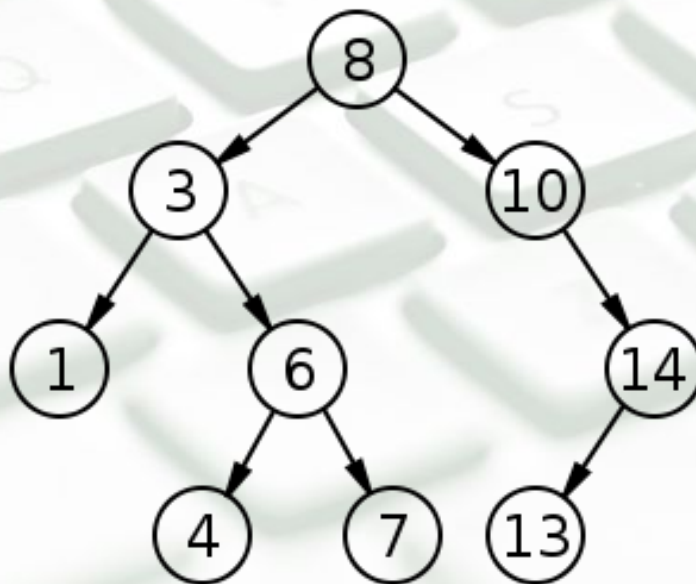


Árvores Binárias

■ Resposta rápido...

Qual a **Ordem de Complexidade (Big O)** para métodos de Inserção e Consulta em uma Árvore Binária de Busca?

Inserção
Logarítmica



Consulta
Logarítmica



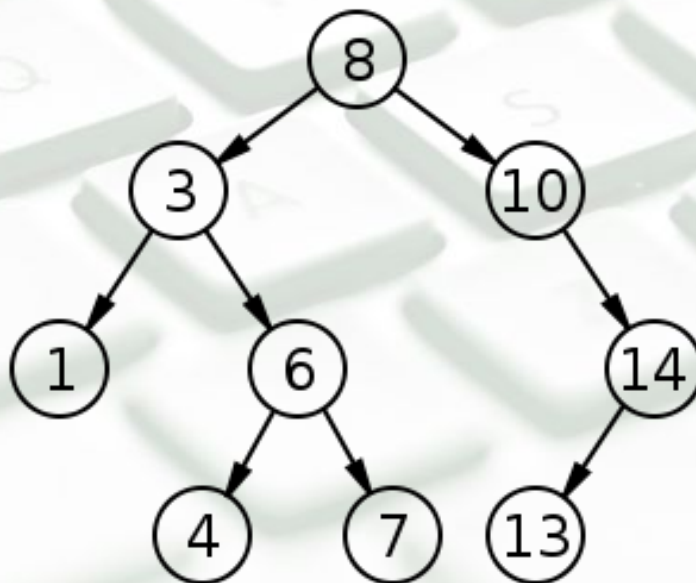
Árvores Binárias

■ Resposta rápido...

Qual a **Ordem de Complexidade (Big O)** para métodos de Inserção e Consulta em uma Árvore Binária de Busca?

Inserção

Logarítmica



Consulta

Logarítmica



Árvores Binárias

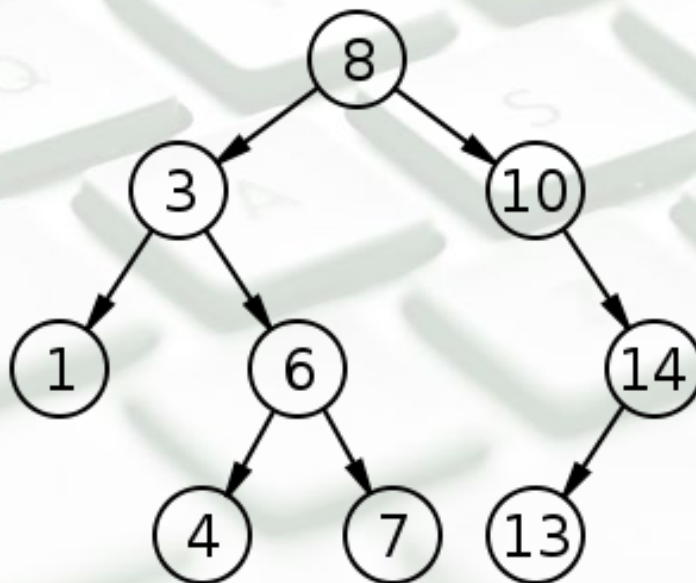
■ Responda rápido...

Qual a **Ordem de Complexidade (Big O)** para métodos de Inserção e Consulta em uma Árvore Binária de Busca?

Inserção

Logarítmica

$O(n) \Rightarrow \text{Linear}$



Consulta

Logarítmica

$O(n) \Rightarrow \text{Linear}$



Árvores Binárias

■ Resposta

Qual a Complexidade de Inserção?

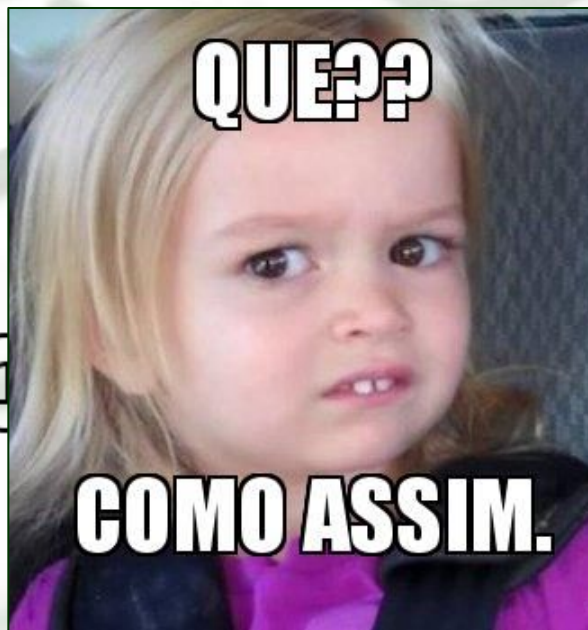
Data Structure	Time Complexity			
	Worst			
	Access	Search	Insertion	Deletion
<u>Stack</u>	$O(n)$	$O(n)$	$O(1)$	$O(1)$
<u>Queue</u>	$O(n)$	$O(n)$	$O(1)$	$O(1)$
<u>Binary Search Tree</u>	$O(n)$	$O(n)$	$O(n)$	$O(n)$

Qual a Complexidade dos métodos de Busca?

Inserção

~~Logarítmica~~

$O(n) \Rightarrow$ Linear



Consulta

~~Logarítmica~~

$O(n) \Rightarrow$ Linear



Árvores Binárias

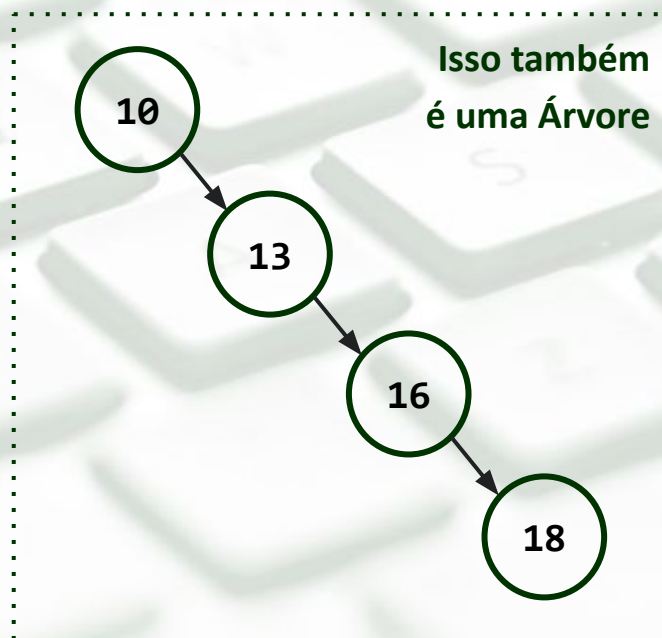
■ Responda rápido...

Qual a **Ordem de Complexidade (Big O)** para métodos de Inserção e Consulta em uma Árvore Binária de Busca?

Inserção

~~Logarítmica~~

$O(n) \Rightarrow$ Linear



Consulta

~~Logarítmica~~

$O(n) \Rightarrow$ Linear



Árvores Binárias

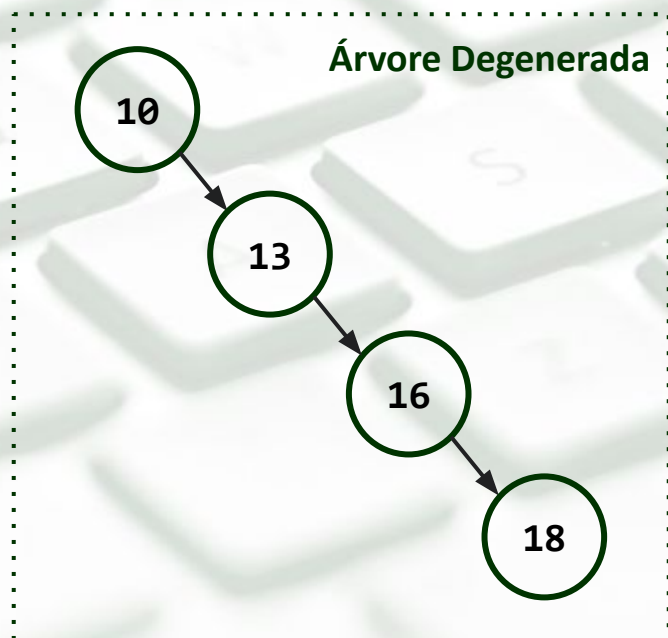
■ Responda rápido...

Qual a **Ordem de Complexidade (Big O)** para métodos de Inserção e Consulta em uma Árvore Binária de Busca?

Inserção

~~Logarítmica~~

$O(n) \Rightarrow$ Linear



Consulta

~~Logarítmica~~

$O(n) \Rightarrow$ Linear



Árvores Binárias

- Obviamente, **Árvores Degeneradas** é um caso de **excepcionalidade** considerando a maioria das construções de Árvores Binárias.
- Por este motivo, tende-se a considerar a **Ordem de Complexidade Theta***, tanto para inserção quanto para consulta de nós.

$$\Theta(\log(n))$$

**Ordem de Complexidade Theta considera o caso médio ao invés do pior caso.*



Árvores Binárias

Data Structure	Time Complexity							
	Average				Worst			
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion
<u>Stack</u>	$\theta(n)$	$\theta(n)$	$\theta(1)$	$\theta(1)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$
<u>Queue</u>	$\theta(n)$	$\theta(n)$	$\theta(1)$	$\theta(1)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$
<u>Binary Search Tree</u>	$\theta(\log(n))$	$\theta(\log(n))$	$\theta(\log(n))$	$\theta(\log(n))$	$O(n)$	$O(n)$	$O(n)$	$O(n)$

- Por este motivo, tende-se a considerar a **Ordem de Complexidade Theta***, tanto para inserção quanto para consulta de nós.

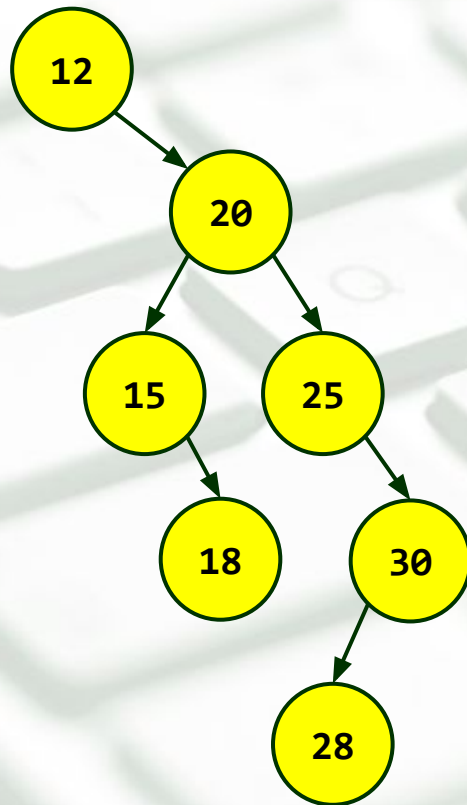
$$\Theta(\log(n))$$

**Ordem de Complexidade Theta considera o caso médio ao invés do pior caso.*



Árvores Binárias

- Outro problema que impacta no desempenho de uma ABB é quando ela se torna **desbalanceada...**



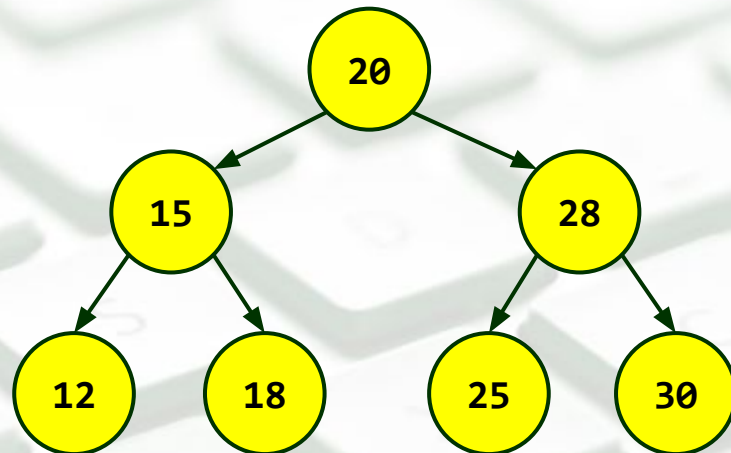
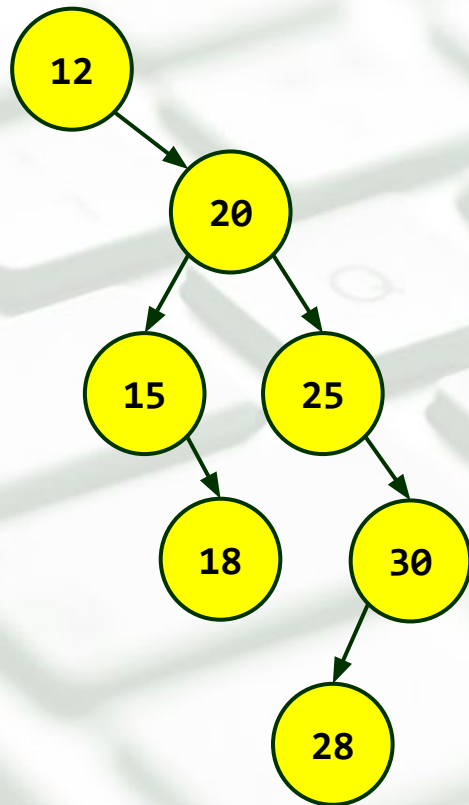
Uma árvore está
desbalanceada quando:

$$\text{Altura} > \log_2(n)$$



Árvores Binárias

- Outro problema que impacta no desempenho de uma ABB é quando ela se torna **desbalanceada...**



O mesmo conjunto de elementos, em uma Árvore Desbalanceada e em uma Árvore Balanceada.



Árvores Binárias

- A grande importância das Árvores Binárias se dá ao fato de **unir o que há de melhor (vantagens) nas duas estruturas de dados vistas anteriormente...**

Comparativo	Complexidade de Operações	Alocação de Memória
Arrays (Busca Binária)	Logarítmico	Estática
Listas Encadeadas (Stack / Queue)	Linear	Dinâmica
Árvores Binárias	Logarítmico	Dinâmica



Árvores Binárias

■ Estrutura Base de um Nó de Árvore Binária

```
typedef struct No{  
    long chave;  
    void* dado;  
    struct No* esq;  
    struct No* dir;  
}No;
```



Árvores Binárias

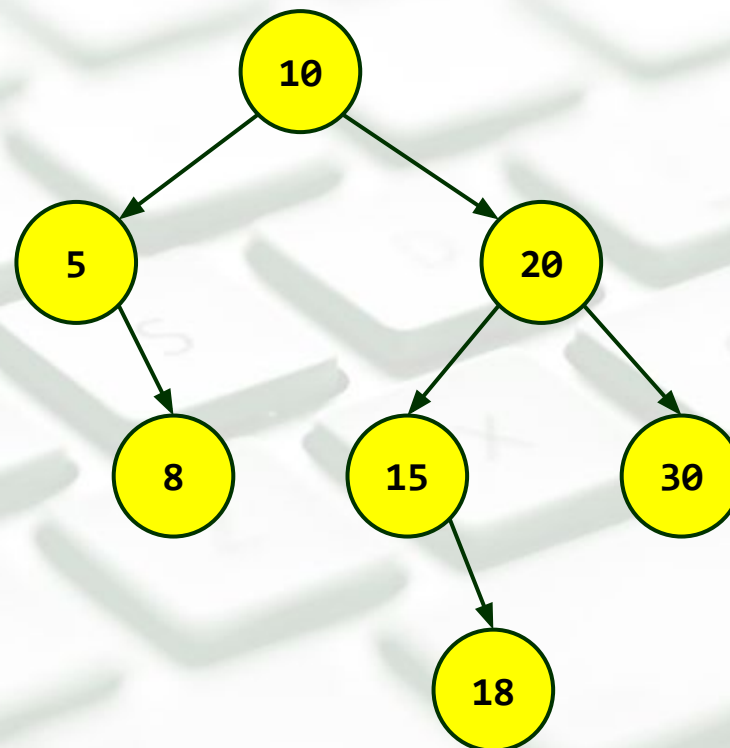
- Consulta / Inserção de Nós em ABB
- Procure o “local” adequado para inserir a nova chave, começando a partir do nó-raiz...
- Para cada nó visitado, compare:
 - Se a nova chave for menor ou igual do que o valor do nó visitado, repita todo o processo para sub-árvore esquerda.
 - Se a nova chave for maior que o valor no nó visitado, repita todo o processo para sub-árvore direita.
- Se um ponteiro (filho esquerdo/direito) nulo é atingido, insira o novo nó neste ramo.
- **A inserção sempre se dá como um nó folha, e não exige deslocamentos!**



Árvores Binárias

■ Consulta / Inserção de Nós em ABB

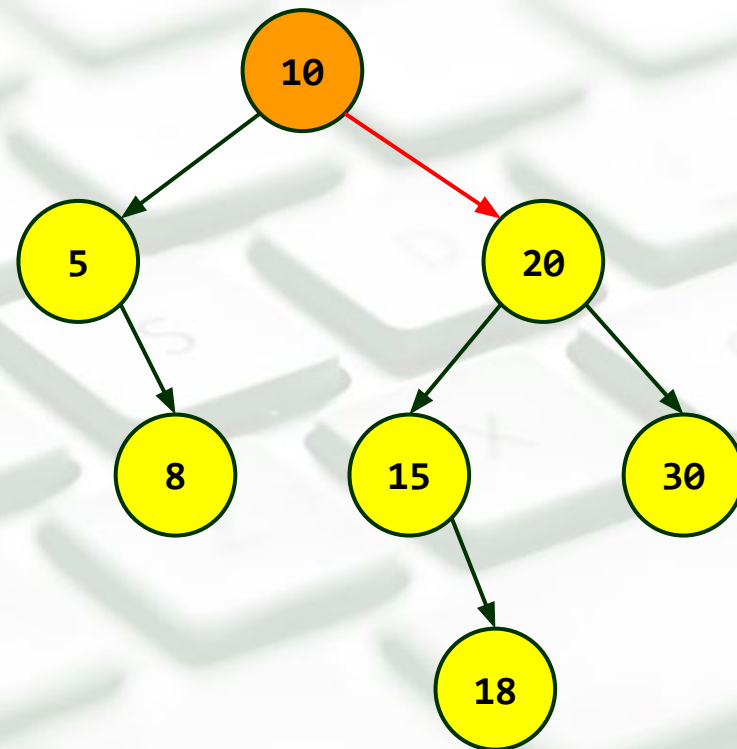
Novo Nó



Árvores Binárias

■ Consulta / Inserção de Nós em ABB

Novo Nó

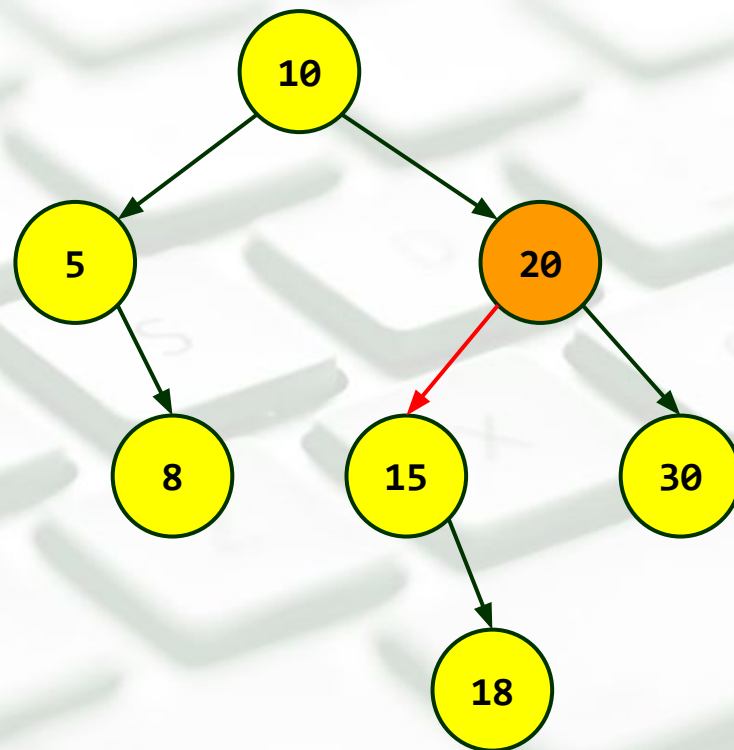




Árvores Binárias

■ Consulta / Inserção de Nós em ABB

Novo Nó

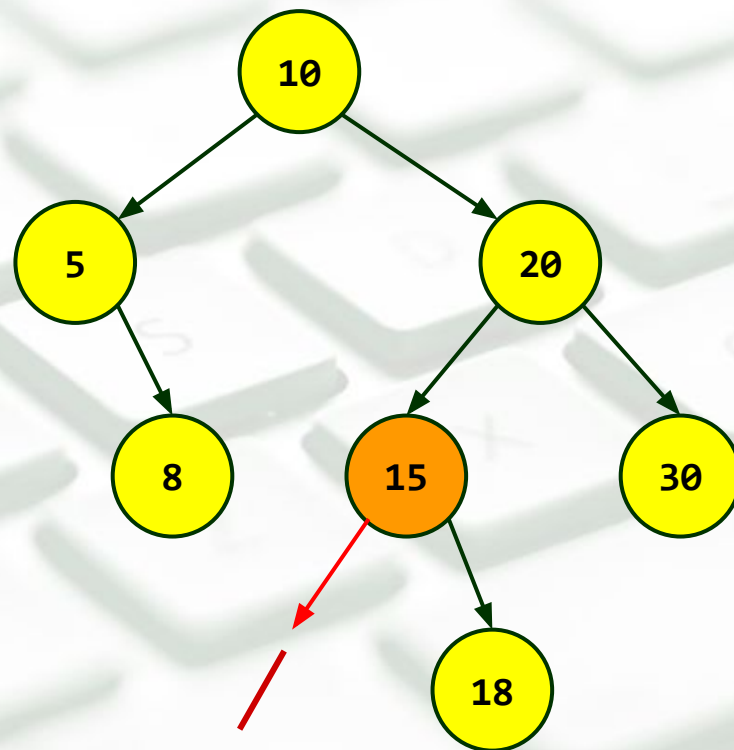




Árvores Binárias

■ Consulta / Inserção de Nós em ABB

Novo Nó

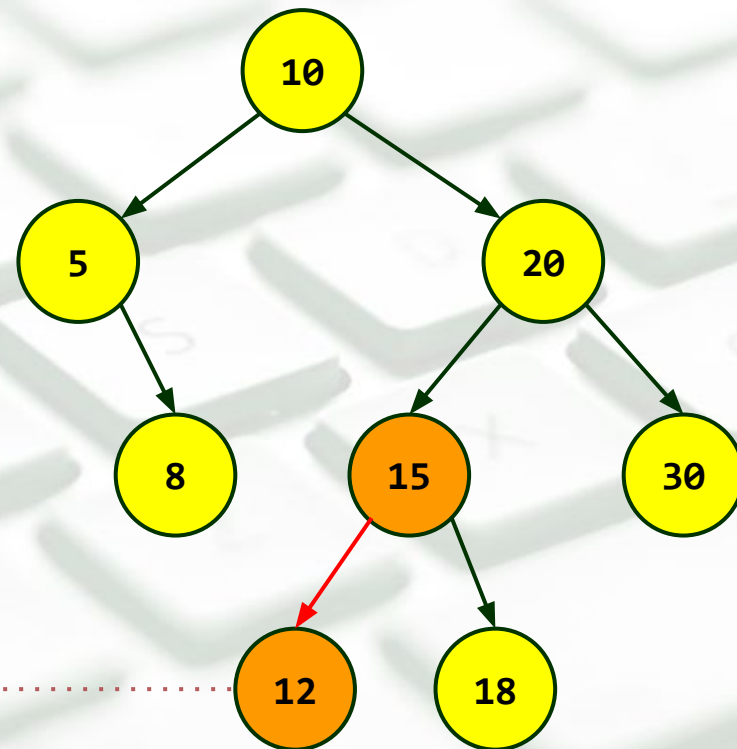




Árvores Binárias

■ Consulta / Inserção de Nós em ABB

Novo Nó





Árvores Binárias

■ Consulta / Inserção de Nós em ABB



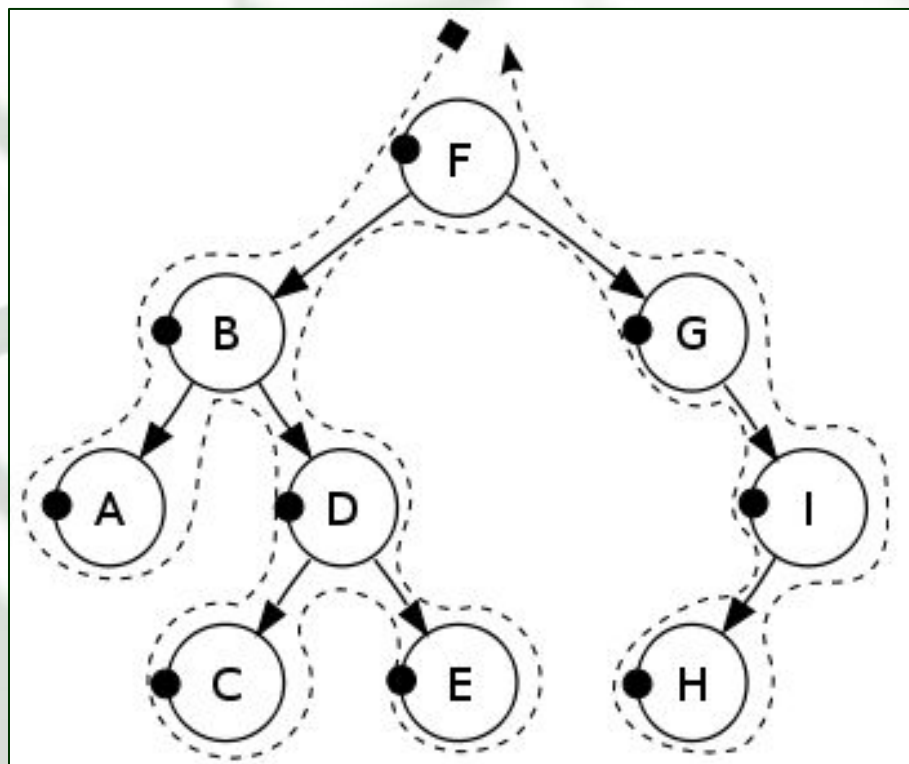
www.mathwarehouse.com





Árvores Binárias

- Percorrer Nós em ABB
 - *Percurso em Profundidade (DFS)*



03 variantes...

Pré Ordem (Raiz -> E -> D)

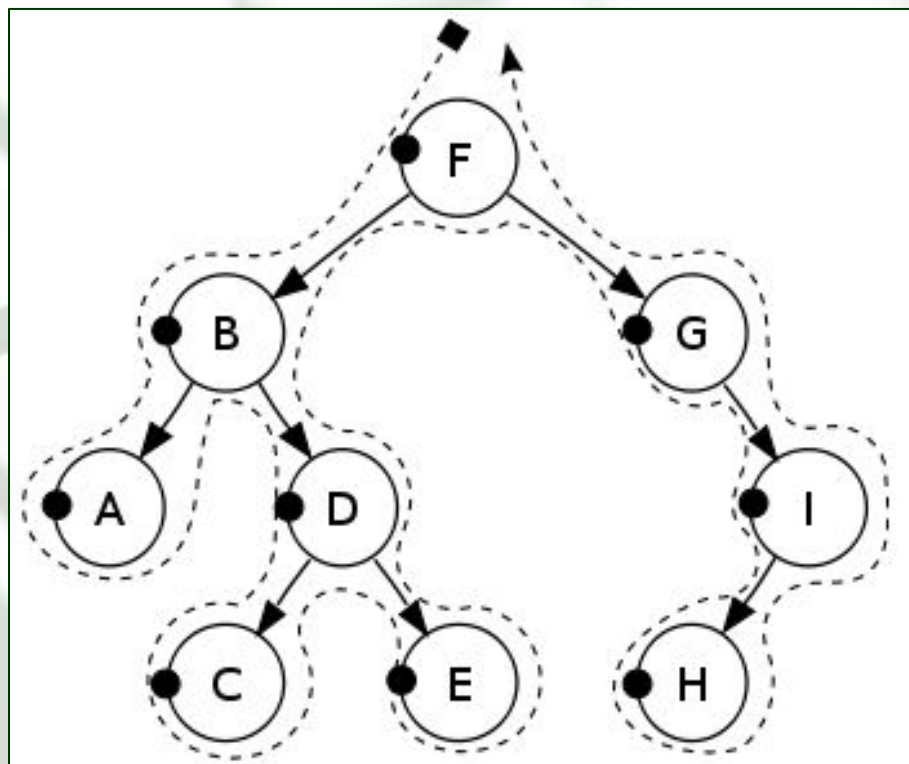
```
void percorre(a){
    if (a){
        printf("%c ", a->chave);
        percorre(a->esq);
        percorre(a->dir);
    }
}
```

>> Qual sequência???



Árvores Binárias

- Percorrer Nós em ABB
 - *Percurso em Profundidade (DFS)*



03 variantes...

Pré Ordem (Raiz -> E -> D)

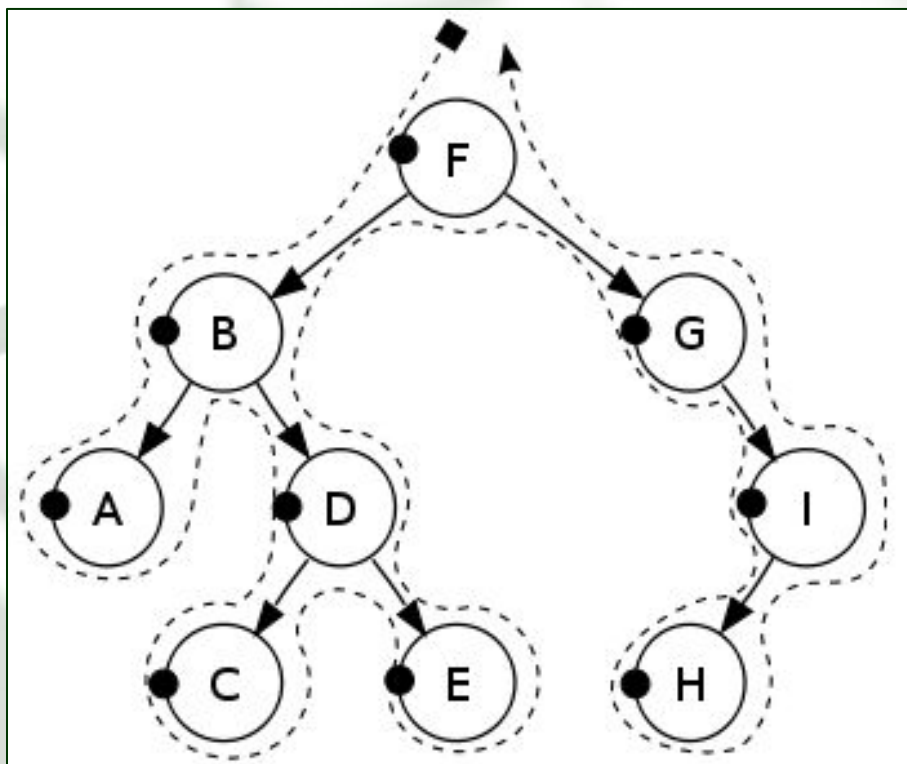
```
void percorre(a){
    if (a){
        printf("%c ", a->chave);
        percorre(a->esq);
        percorre(a->dir);
    }
}
```

>> F B A D C E G I H



Árvores Binárias

- Percorrer Nós em ABB
 - *Percurso em Profundidade (DFS)*



03 variantes...

Em Ordem (E -> Raiz -> D)

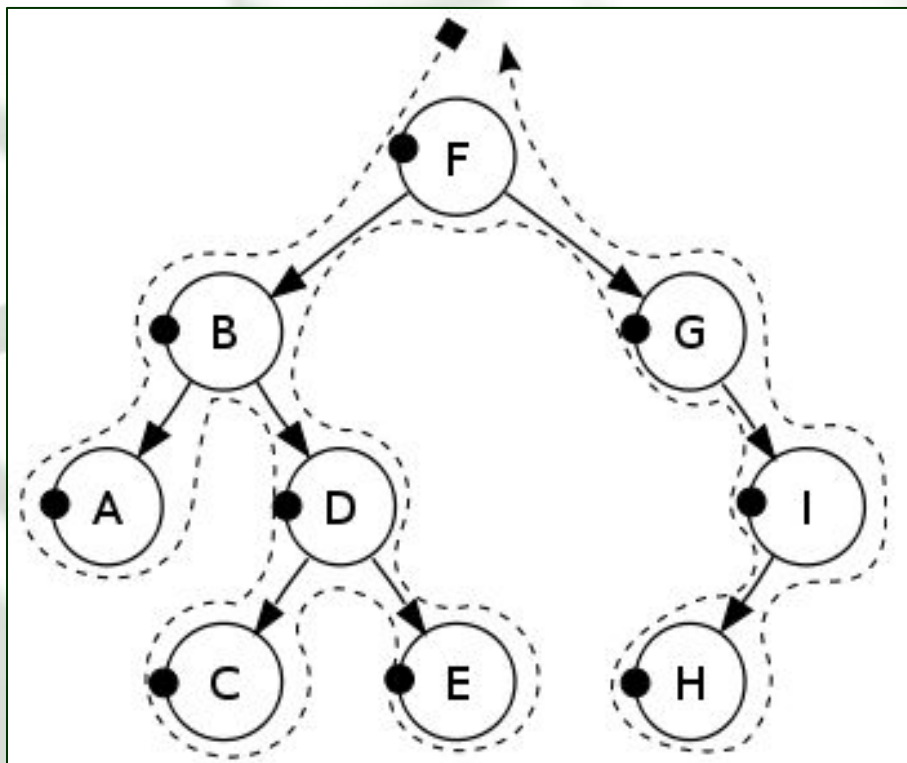
```
void percorre(a){
    if (a){
        percorre(a->esq);
        printf("%c ", a->chave);
        percorre(a->dir);
    }
}
```

>> Qual sequência???



Árvores Binárias

- Percorrer Nós em ABB
 - *Percurso em Profundidade (DFS)*



03 variantes...

Em Ordem (E -> Raiz -> D)

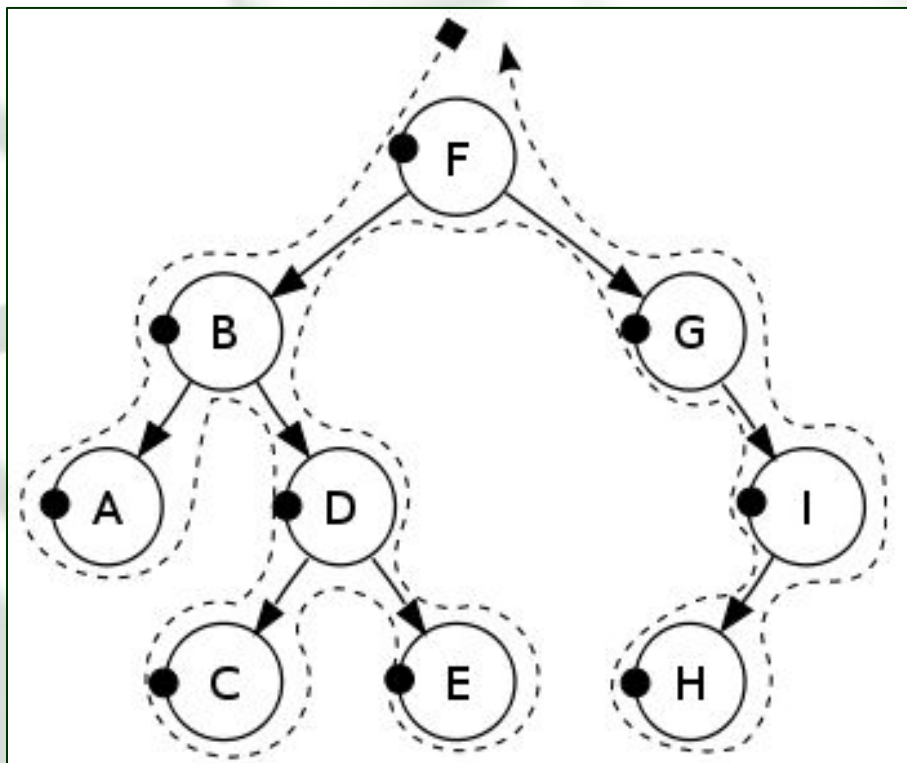
```
void percorre(a){
    if (a){
        percorre(a->esq);
        printf("%c ", a->chave);
        percorre(a->dir);
    }
}
```

>> A B C D E F G H I



Árvores Binárias

- Percorrer Nós em ABB
 - *Percurso em Profundidade (DFS)*



03 variantes...

Pós Ordem (E -> D -> Raiz)

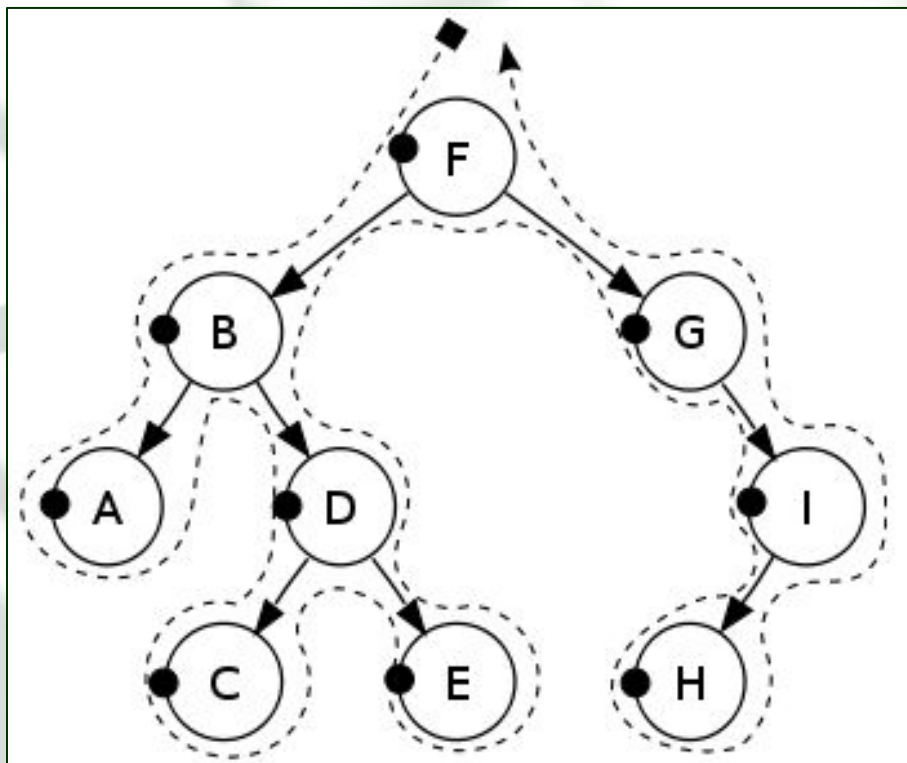
```
void percorre(a){
    if (a){
        percorre(a->esq);
        percorre(a->dir);
        printf("%c ", a->chave);
    }
}
```

>> Qual sequência???



Árvores Binárias

- Percorrer Nós em ABB
 - *Percurso em Profundidade (DFS)*



03 variantes...

Pós Ordem (E -> D -> Raiz)

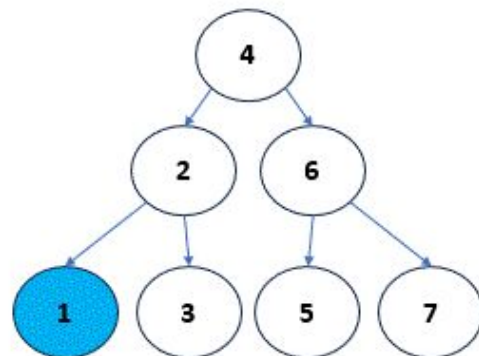
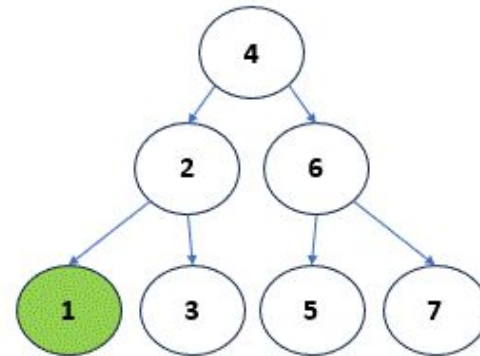
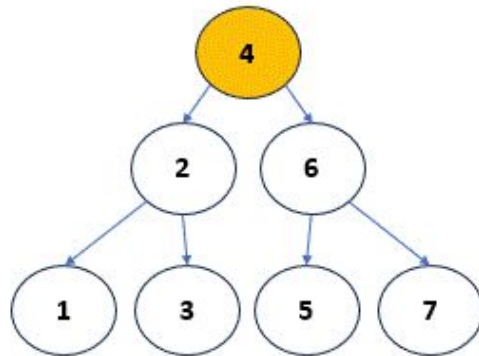
```
void percorre(a){
    if (a){
        percorre(a->esq);
        percorre(a->dir);
        printf("%c ", a->chave);
    }
}
```

>> A C E D B H I G F



Árvores Binárias

■ Percorrer Nós em ABB



Pre order ↓

In order

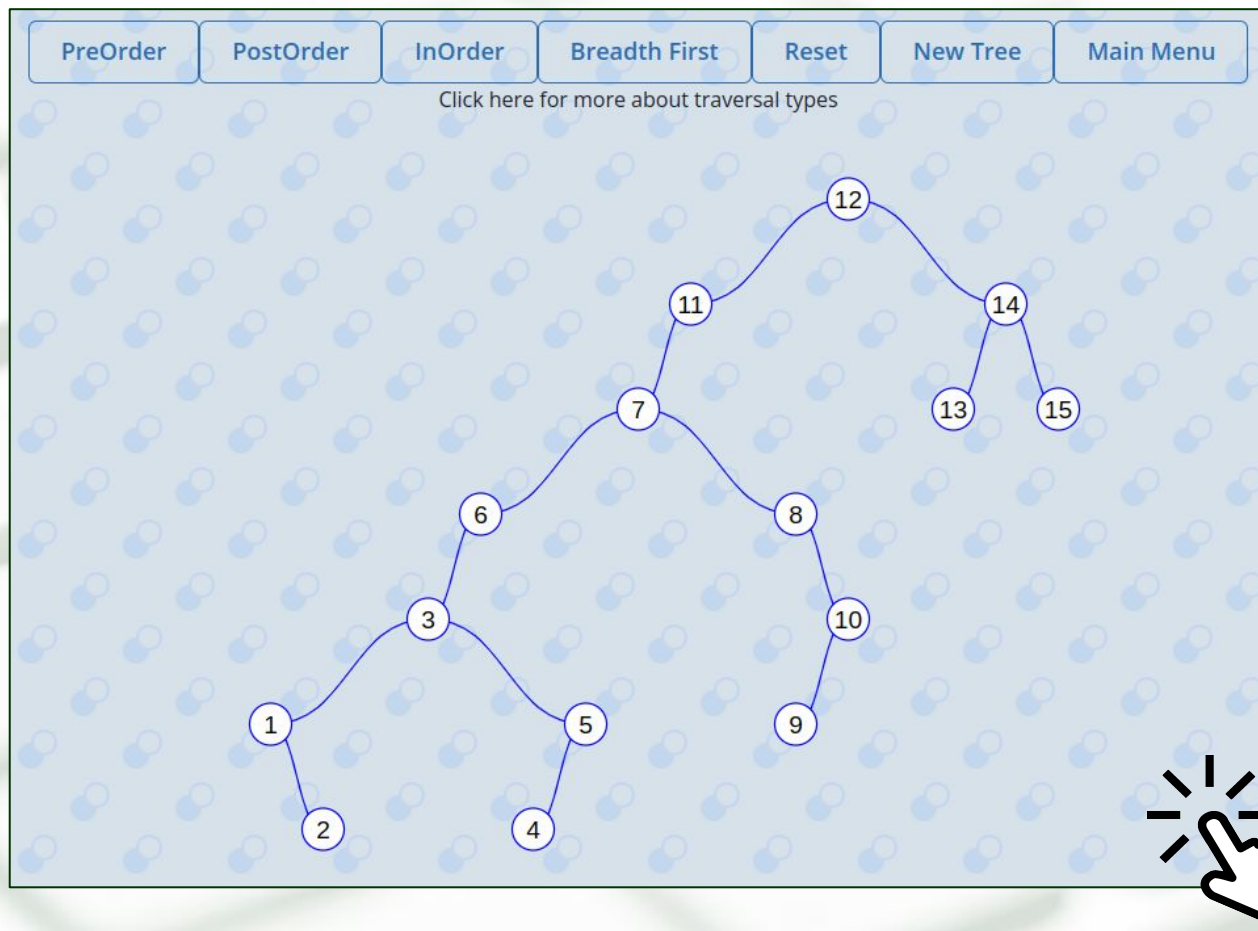
Post order ↑





Árvores Binárias

■ Site animado...





Árvores Binárias

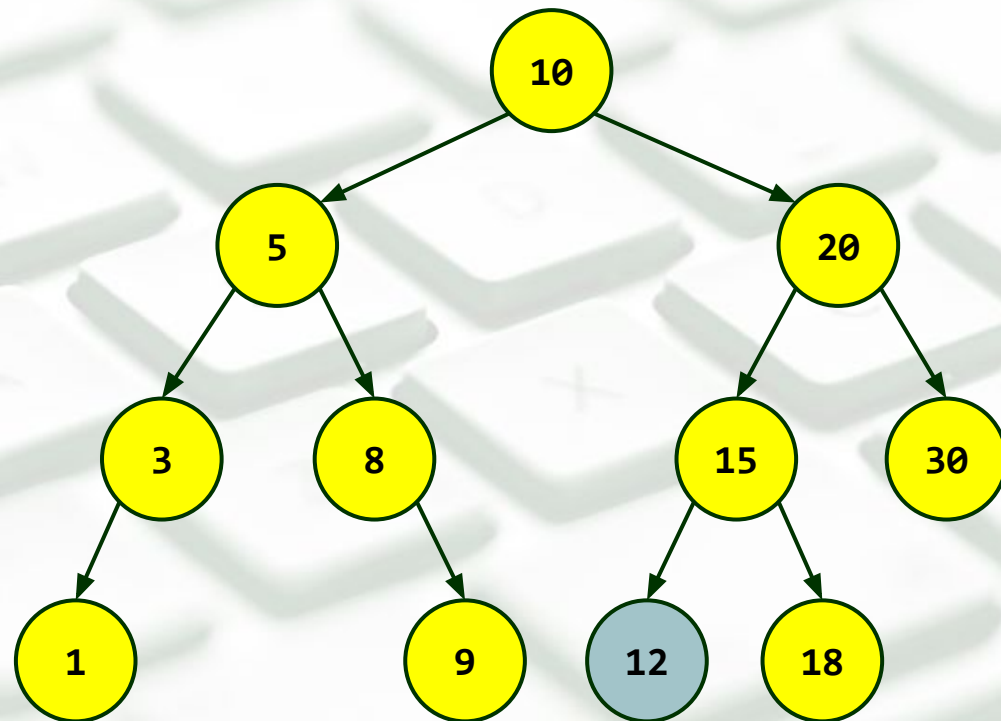
- Exclusão de Nós em ABB
 - *Existem 03 situações possíveis...*

Quando o Nó a ser excluído é uma FOLHA:



Quando o nó a ser removido não possui filhos, a exclusão pode acontecer de forma direta, sem re-arranjo da árvore.

O ponteiro do pai deve ser atualizado para NULO





Árvores Binárias

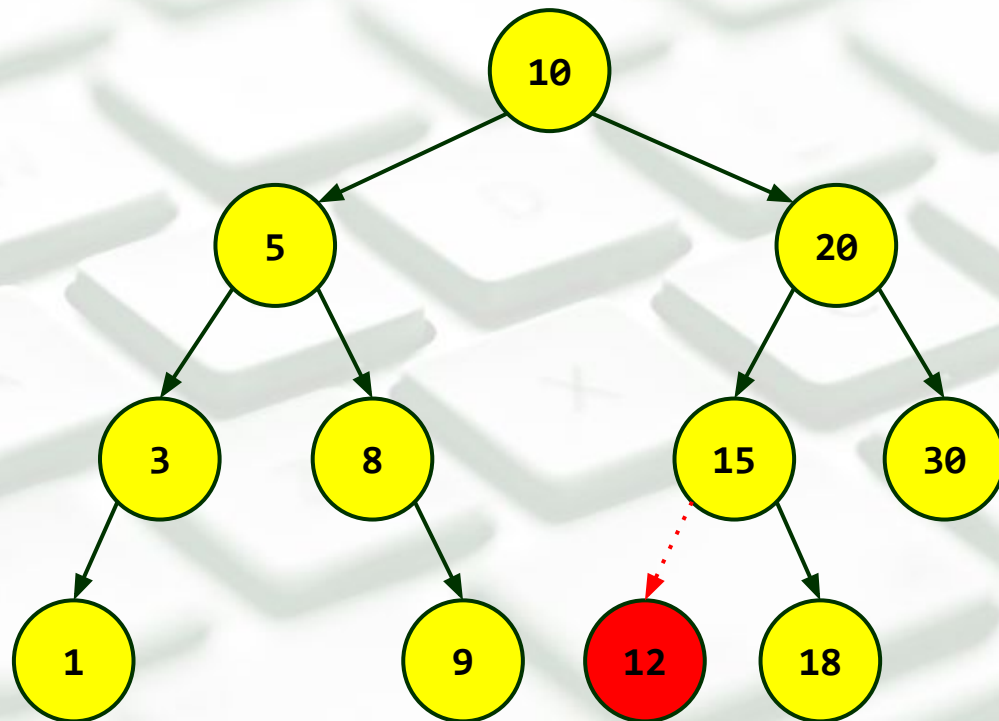
- Exclusão de Nós em ABB
 - *Existem 03 situações possíveis...*

Quando o Nó a ser excluído é uma FOLHA:



Quando o nó a ser removido não possui filhos, a exclusão pode acontecer de forma direta, sem re-arranjo da árvore.

O ponteiro do pai deve ser atualizado para NULO





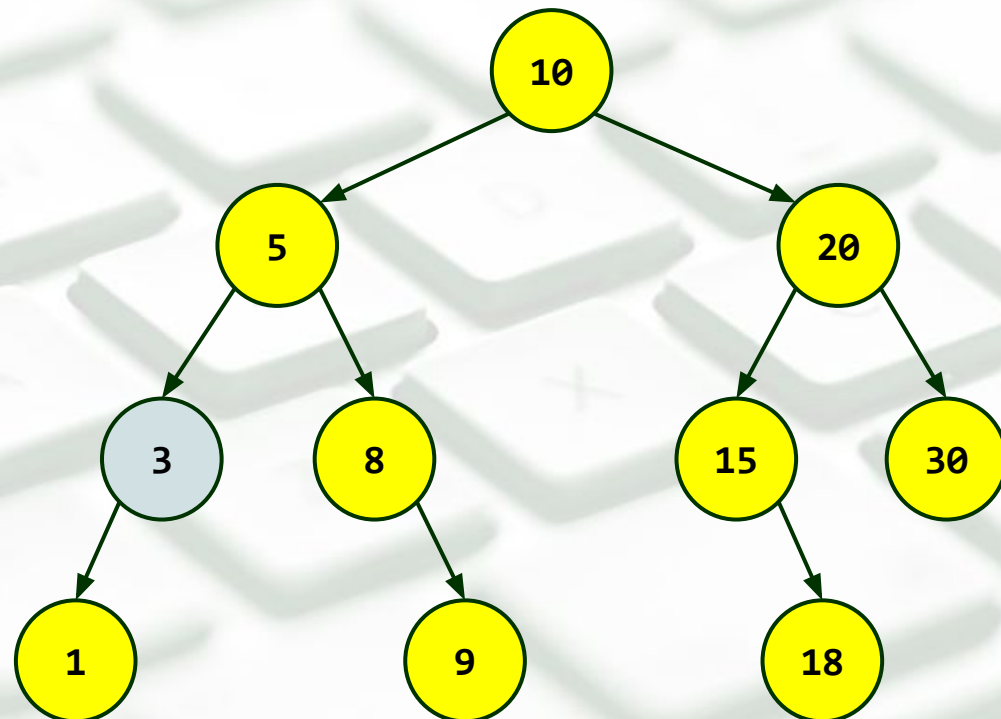
Árvores Binárias

- Exclusão de Nós em ABB
 - *Existem 03 situações possíveis...*

Quando o Nó a
ser excluído
possui UM FILHO:



Quando o nó a ser removido possui UM ÚNICO filho (esquerda ou direita), a exclusão desse nó deve acontecer após este filho substituí-lo no arranjo da árvore.





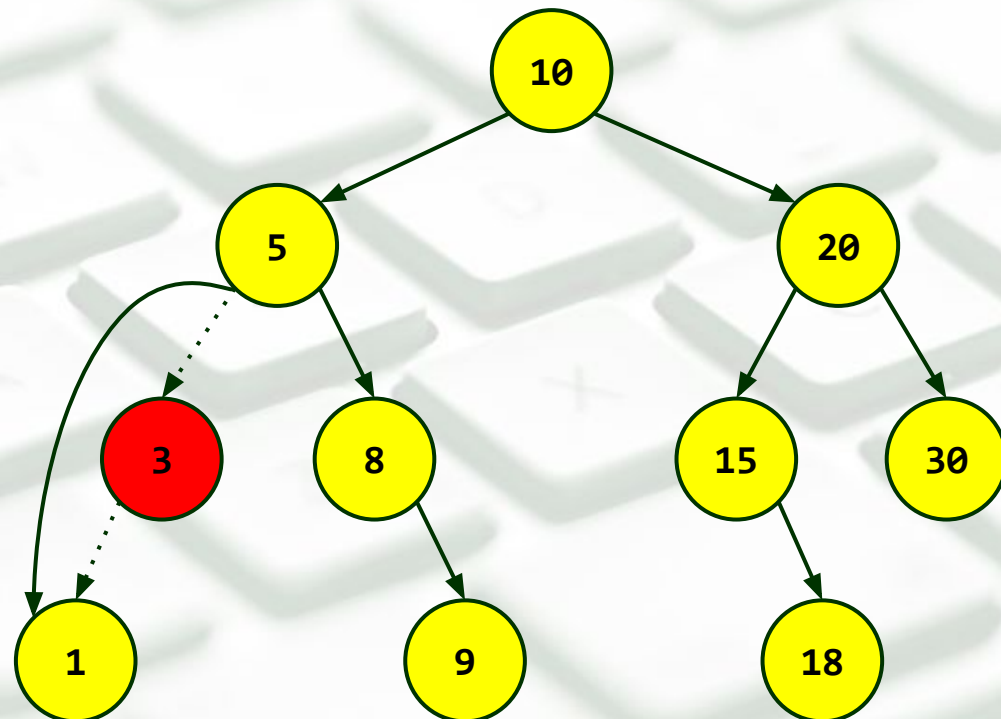
Árvores Binárias

- Exclusão de Nós em ABB
 - *Existem 03 situações possíveis...*

Quando o Nó a
ser excluído
possui UM FILHO:



Quando o nó a ser removido possui UM ÚNICO filho (esquerda ou direita), a exclusão desse nó deve acontecer após este filho substituí-lo no arranjo da árvore.





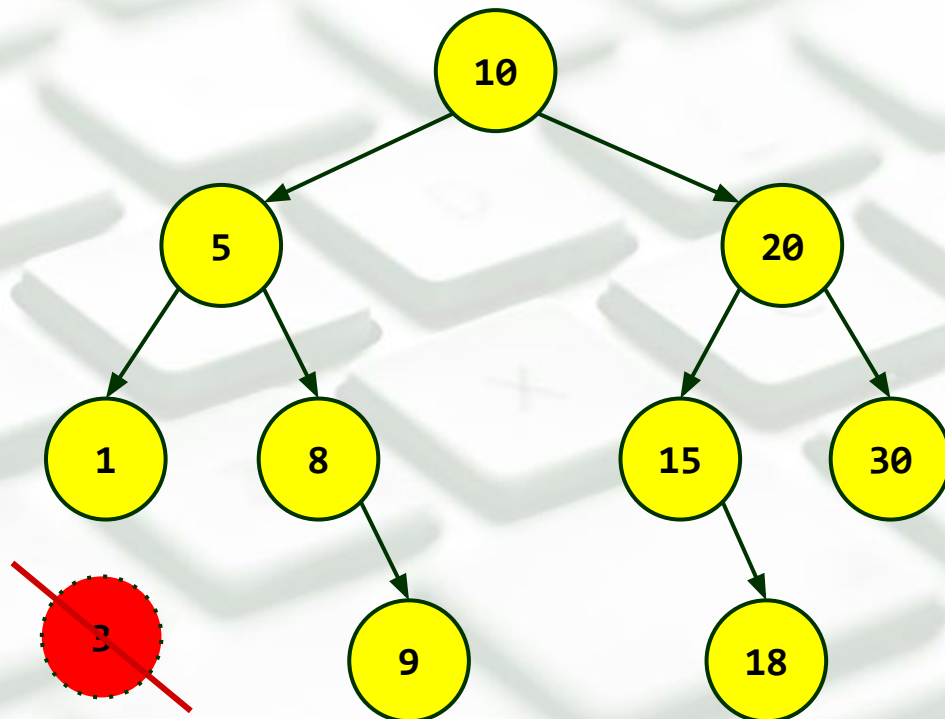
Árvores Binárias

- Exclusão de Nós em ABB
 - *Existem 03 situações possíveis...*

Quando o Nó a
ser excluído
possui UM FILHO:



Quando o nó a ser removido possui UM ÚNICO filho (esquerda ou direita), a exclusão desse nó deve acontecer após este filho substituí-lo no arranjo da árvore.





Árvores Binárias

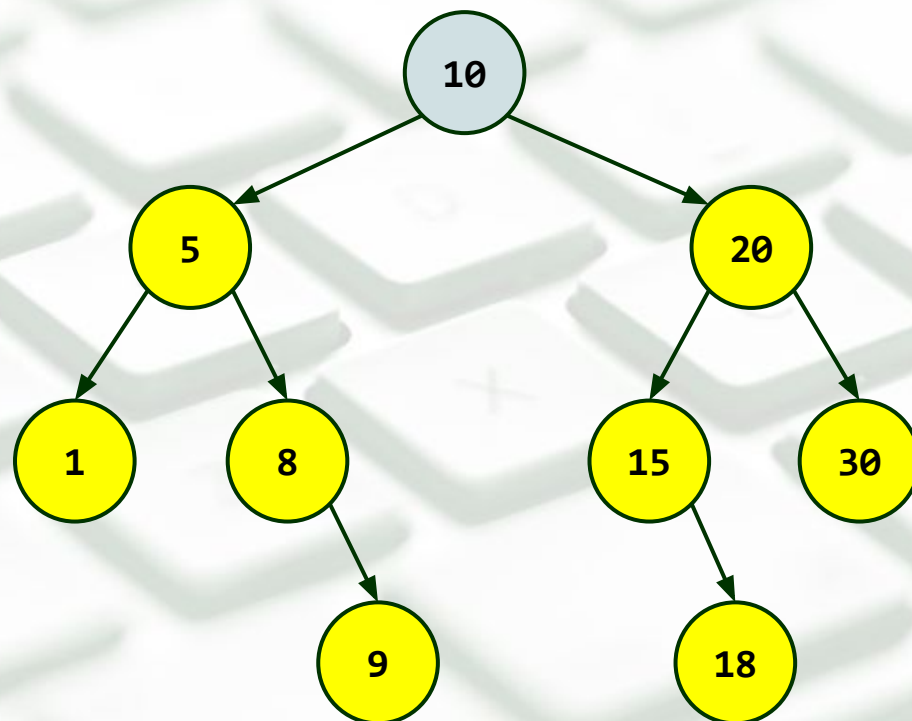
- Exclusão de Nós em ABB
 - *Existem 03 situações possíveis...*

Quando o Nó a ser
excluído possui
DOIS FILHOS:



Quando o nó a ser removido
possui DOIS FILHOS, deve ser
substituído por:

- Maior elemento da subárvore esquerda; ou
- Menor elemento da subárvore direita.





Árvores Binárias

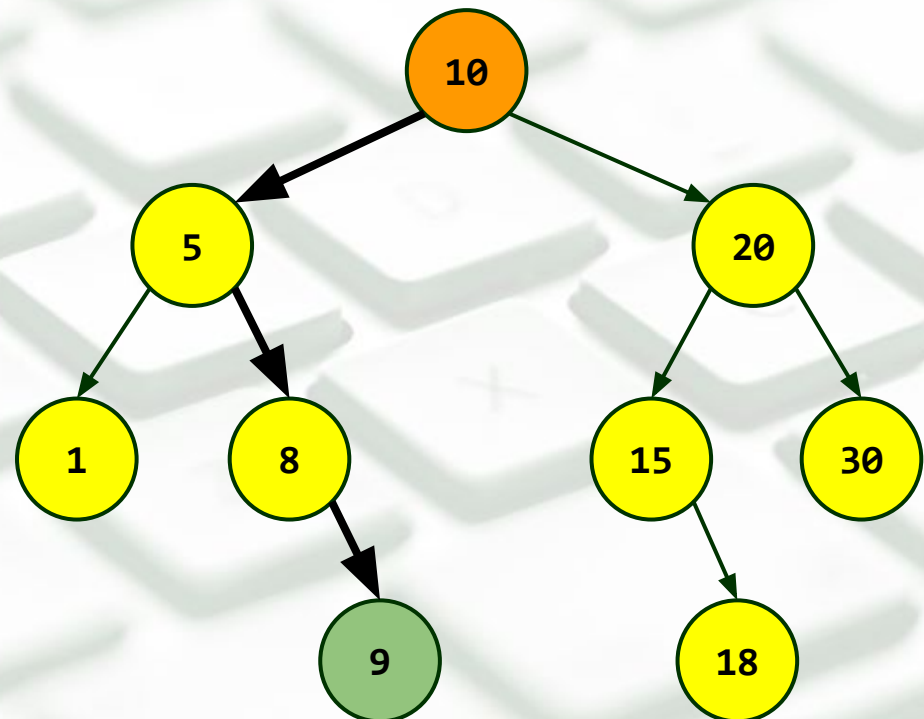
- Exclusão de Nós em ABB
 - *Existem 03 situações possíveis...*

Quando o Nó a ser
excluído possui
DOIS FILHOS:



Quando o nó a ser removido
possui DOIS FILHOS, deve ser
substituído por:

- Maior elemento da subárvore esquerda; ou**
- Menor elemento da subárvore direita.





Árvores Binárias

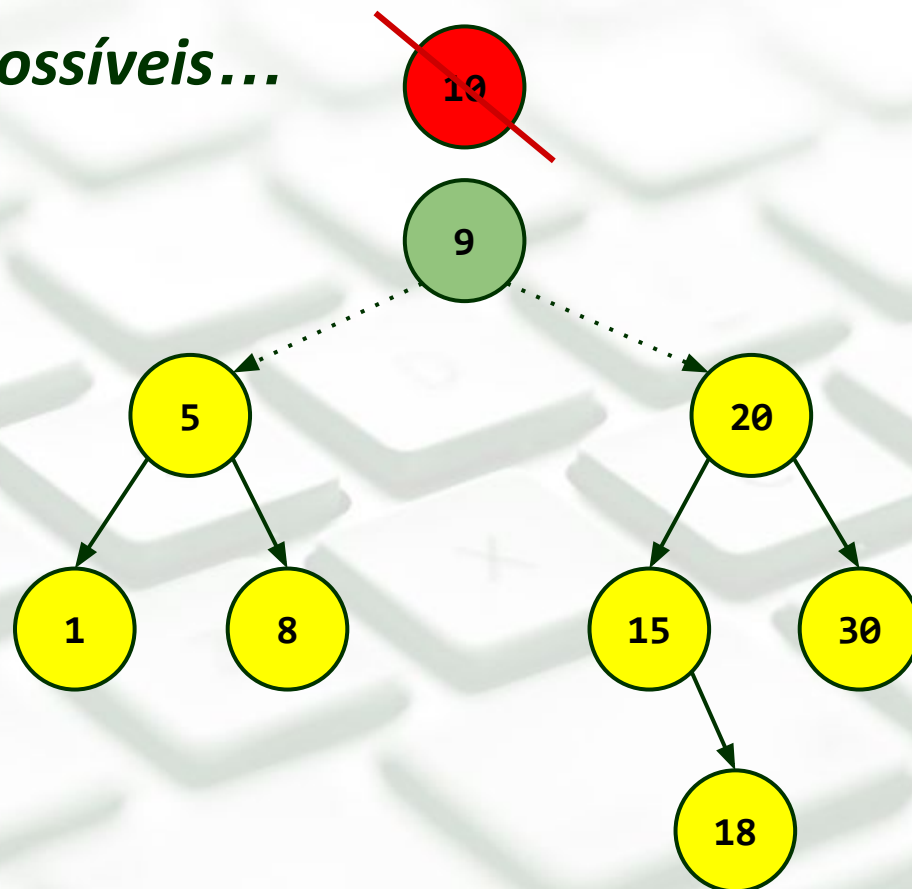
- Exclusão de Nós em ABB
 - *Existem 03 situações possíveis...*

Quando o Nó a ser
excluído possui
DOIS FILHOS:



Quando o nó a ser removido
possui DOIS FILHOS, deve ser
substituído por:

- Maior elemento da subárvore esquerda; ou**
- Menor elemento da subárvore direita.





Árvores Binárias

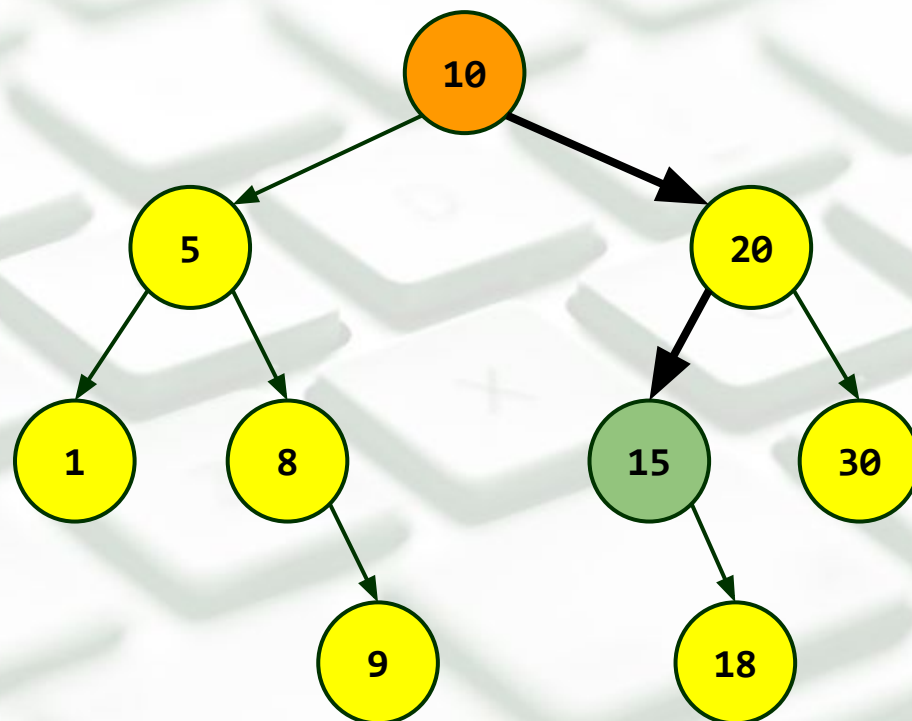
- Exclusão de Nós em ABB
 - *Existem 03 situações possíveis...*

Quando o Nó a ser
excluído possui
DOIS FILHOS:



Quando o nó a ser removido
possui DOIS FILHOS, deve ser
substituído por:

- Maior elemento da subárvore esquerda; ou
- Menor elemento da subárvore direita.**





Árvores Binárias

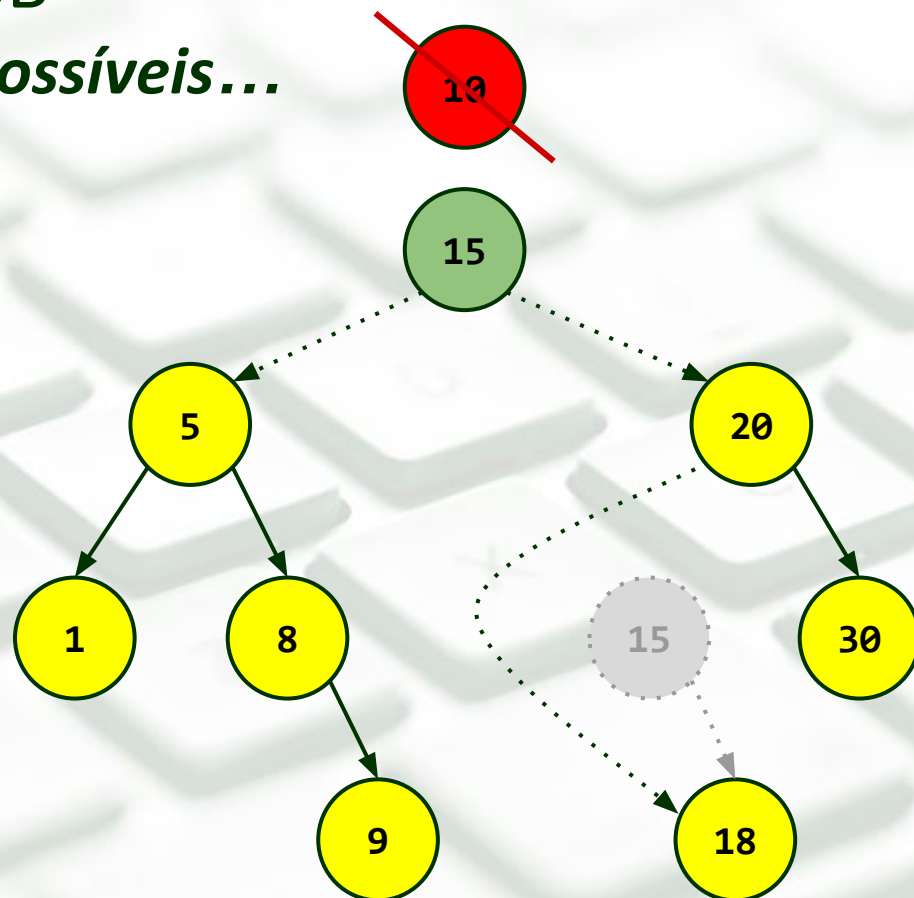
- Exclusão de Nós em ABB
 - *Existem 03 situações possíveis...*

Quando o Nó a ser
excluído possui
DOIS FILHOS:



Quando o nó a ser removido
possui DOIS FILHOS, deve ser
substituído por:

- a) Maior elemento da subárvore esquerda; ou
- b) **Menor elemento da subárvore direita.**





Árvores Binárias

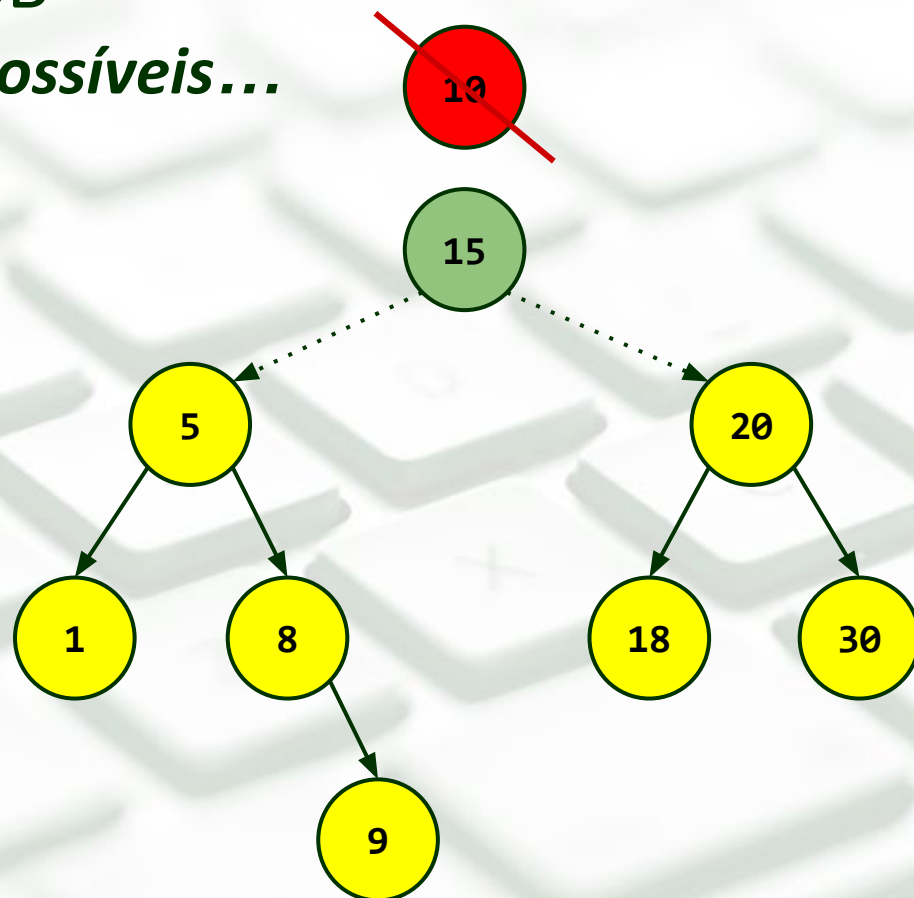
- Exclusão de Nós em ABB
 - *Existem 03 situações possíveis...*

Quando o Nó a ser
excluído possui
DOIS FILHOS:



Quando o nó a ser removido
possui DOIS FILHOS, deve ser
substituído por:

- a) Maior elemento da subárvore esquerda; ou
- b) **Menor elemento da subárvore direita.**





- Criar uma **biblioteca com funções genéricas** para uma *Binary Search Tree (BST)*.

```
BSTree abb = new(BSTree)
```

<Criar uma Árvore Vazia>

```
abb->insert(Object)
```

<Adicionar Objeto na Árvore>

```
abb->search(key)
```

<Retornar Object* de chave key>

```
abb->remove(key)
```

<Remover objeto de chave key>

```
abb->clear()
```

<Limpar/Excluir todos Obj da Árvore>

```
abb->preorder()
```

<Impressão dos Objetos sob
percurso pre-order>

```
abb->inorder()
```

<Impressão dos Objetos sob
percurso in-order>

```
abb->postorder()
```

<Impressão dos Objetos sob
percurso post-order>



Referências

- Feofiloff, Paulo. Projeto de Algoritmos
- Apostila UFPR
- Algol.dev - Árvores
- Recurso Educacional Aberto para o Ensino de Algoritmos e Estruturas de Dados
- FreeCodeCamp
- <https://medium.com/@yatshunlee>