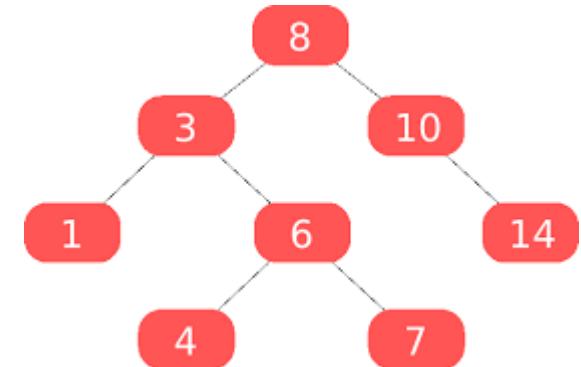


ESTRUTURA DE DADOS

Busca binaria

Análise da busca binaria:

A busca binaria é um algoritmo mais eficiente, entretanto, requer que o vetor esteja ordenado pelos valores da chave de busca.



Análise da busca binaria :

A ideia do algoritmo e a seguinte (assuma que o vetor está ordenado):

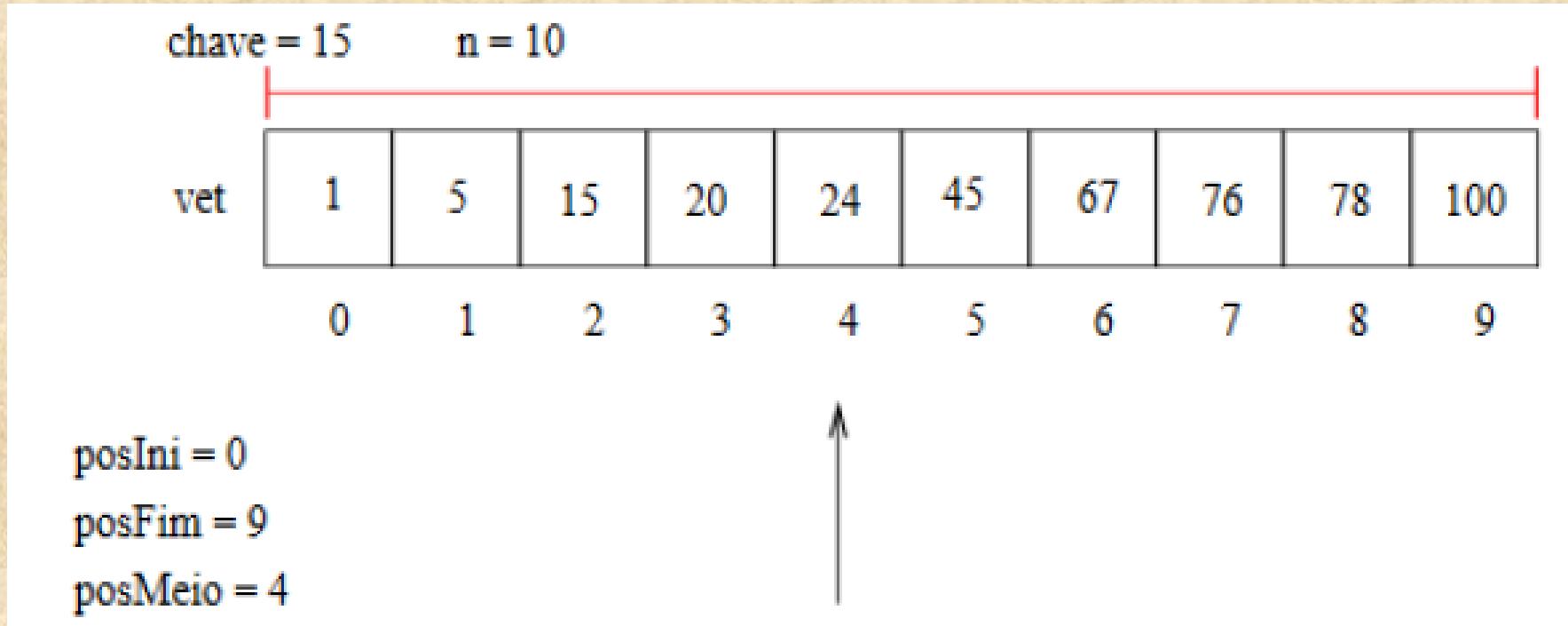
1 - Verifique se a chave de busca é igual ao valor da posição do meio do vetor.

2- Caso seja igual, devolva esta posição;

3- Caso o valor desta posição seja maior, então repita o processo mas considere que o vetor tem metade do tamanho, indo até a posição anterior a do meio.

4- Caso o valor desta posição seja menor, então repita o processo mas considere que o vetor tem metade do tamanho e inicia na posição seguinte a do meio.

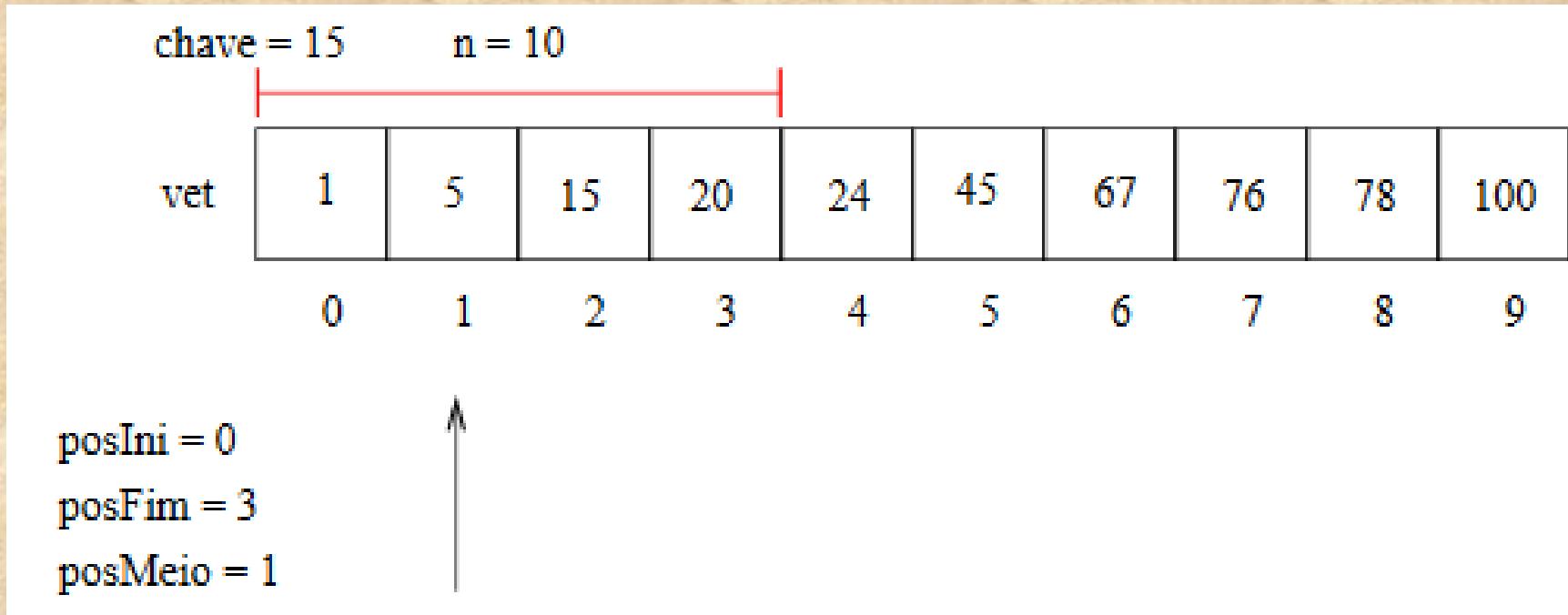
Análise da busca binaria :



Como o valor da posição do meio é maior do que a chave, atualizamos **posFim** do vetor considerado.

Busca binaria

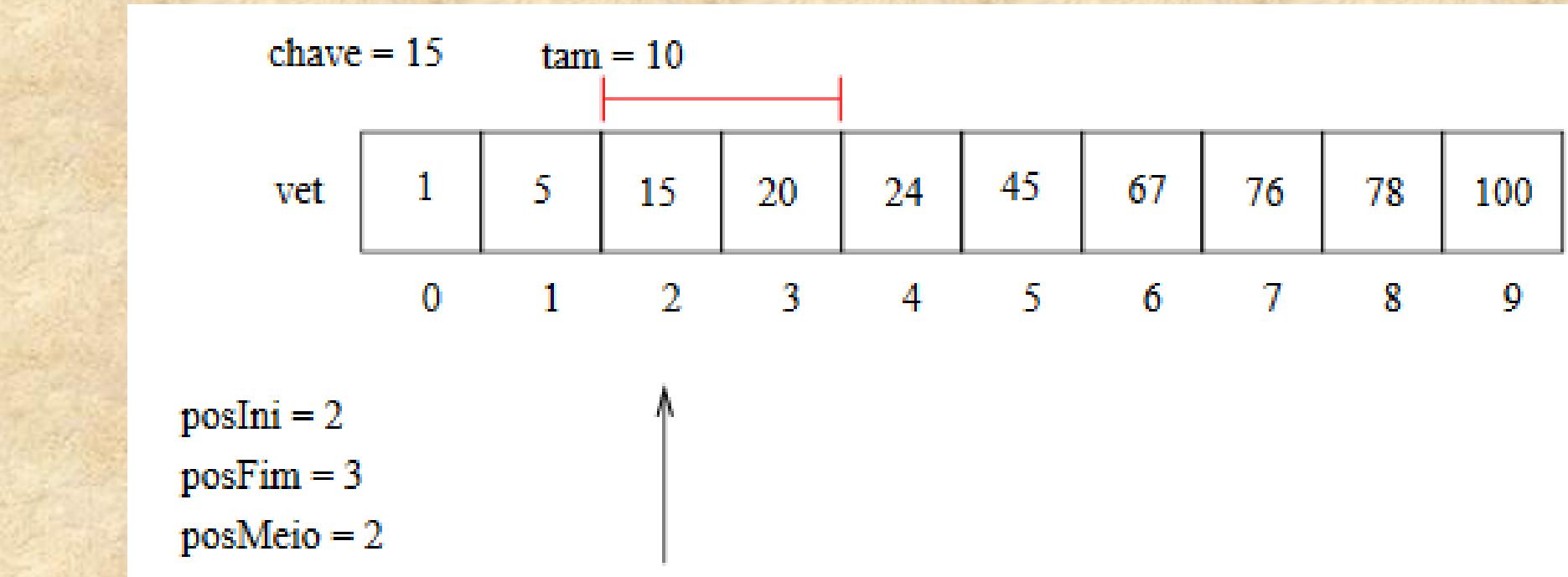
Análise da busca binaria :



Como o valor da posição do meio é menor do que a chave, atualizamos *posIni* no vetor considerado.

Busca binaria

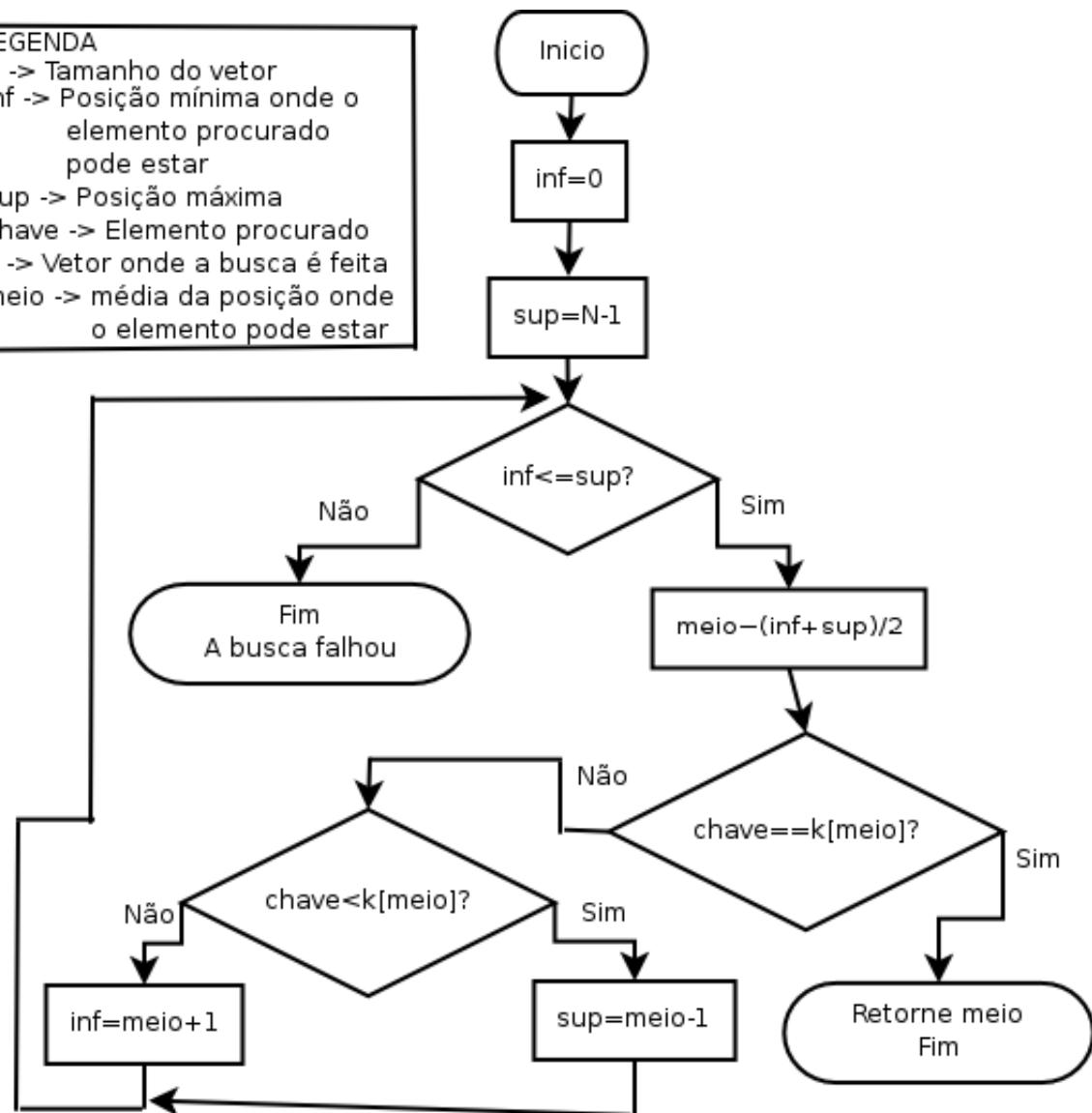
Análise da busca binária :



Finalmente, encontramos a chave na posição do meio do trecho do vetor ainda considerado e devolvemos a sua posição.

LEGENDA

N -> Tamanho do vetor
inf -> Posição mínima onde o elemento procurado pode estar
sup -> Posição máxima
chave -> Elemento procurado
k -> Vetor onde a busca é feita
meio -> média da posição onde o elemento pode estar



Em nosso dia-a-dia nos deparamos com muitos exemplos de árvores:

- Árvore genealógica;
- Organograma de uma empresa;
- Tabela de um torneio esportivo.
- Na computação:
 - Organização da estrutura de arquivos (diretório);
 - Armazenamento e busca eficiente de dados;
 - Ordenação;
 - Árvores de decisão.

Buscar o número 26 ?

- 2 5 8 10 15 20 25

• 2

5

8

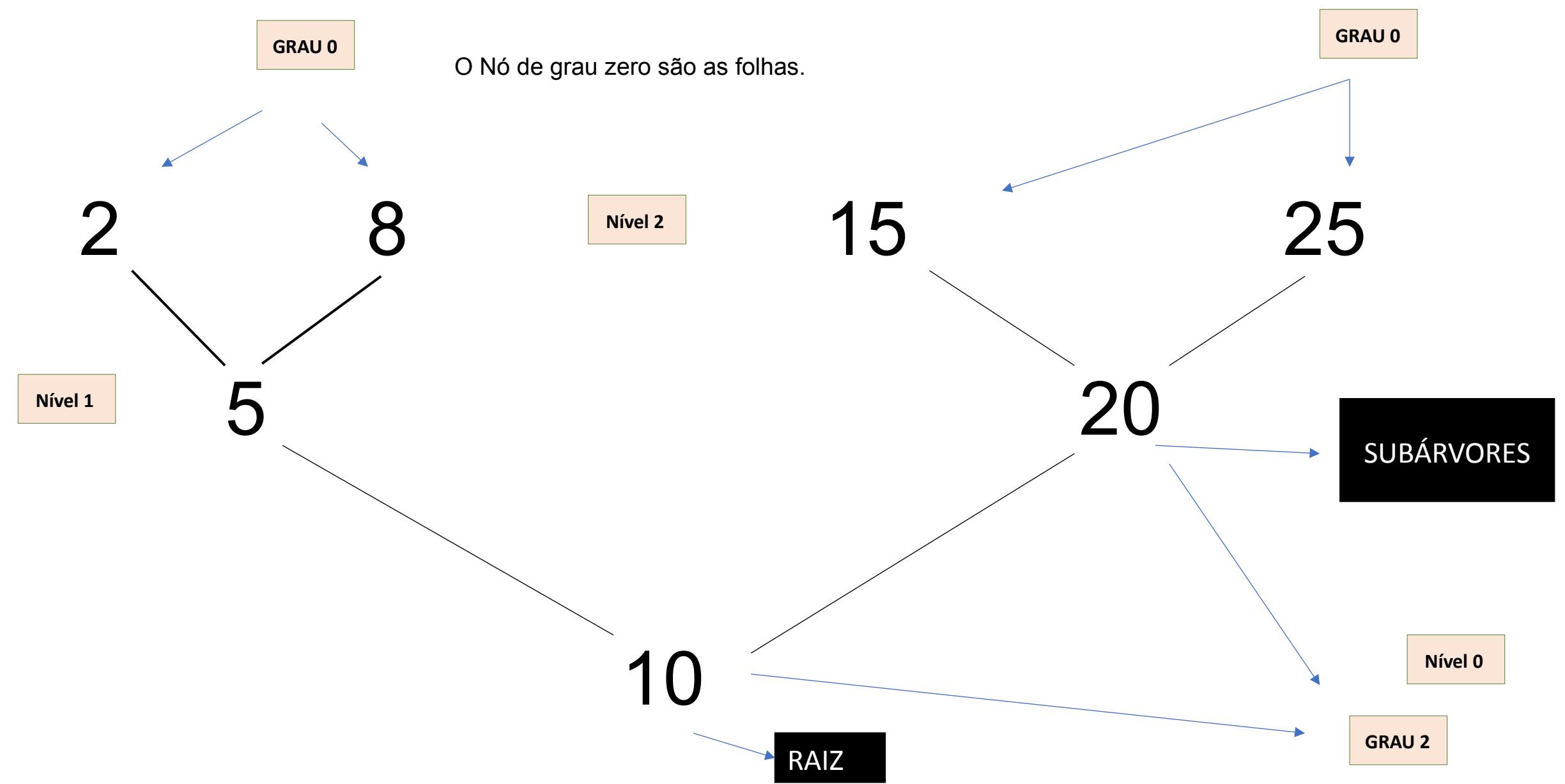
15

20

25

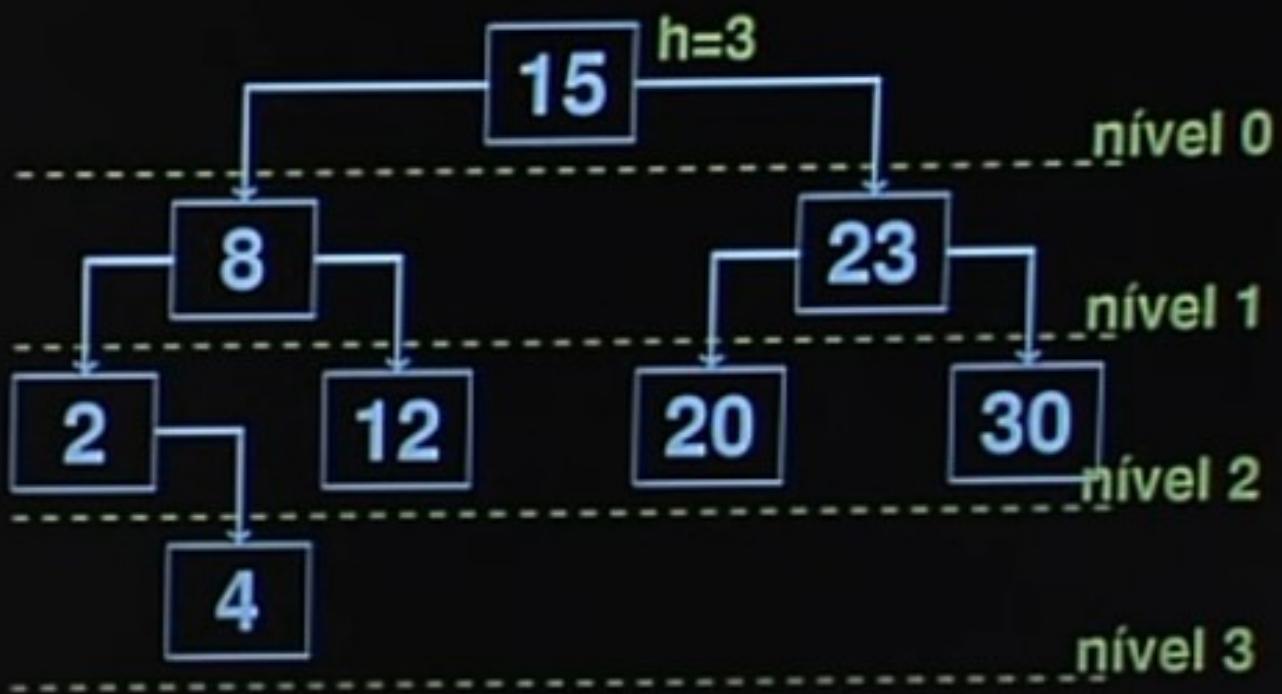
•

10



Árvores

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



Árvores

Árvore livre: grafo não direcionado acíclico e conectado. É comum dizer apenas que o grafo é uma árvore omitindo o “livre”.

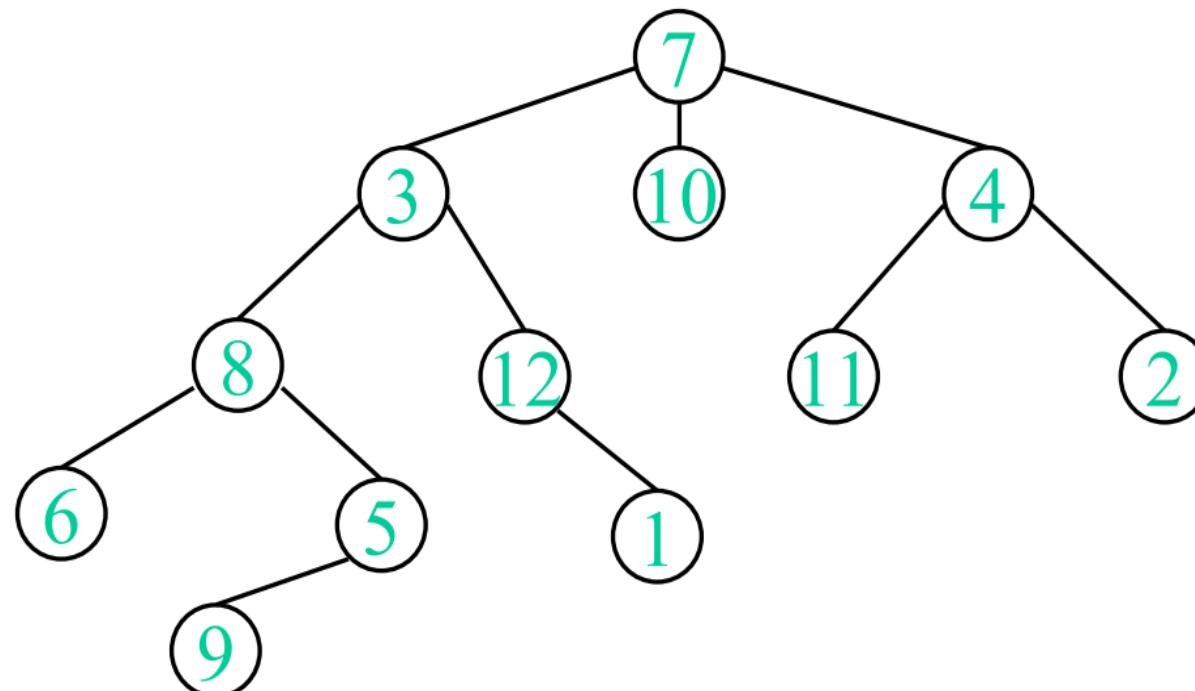
- **Floresta**: grafo não direcionado acíclico, podendo ou não ser conectado.
- **Árvore geradora de um grafo conectado $G = (V, A)$** : subgrafo que contém todos os vértices de G e forma uma árvore.

Árvore Livre

- Uma árvore (livre) é um grafo acíclico, não orientado e conectado.
- Uma floresta é um grafo acíclico, não orientado mas, possivelmente, desconectado.
- Considerando que $G = (V, E)$ é um grafo não orientado, é equivalente dizer:
 1. G é uma árvore.
 2. Um par de vértice qualquer (v, w) de G está conectado por apenas um caminho.
 3. G é conectado. A remoção de uma aresta desconecta G .
 4. G é conectado e $|E| = |V| - 1$.
 5. G é acíclico e $|E| = |V| - 1$.
 6. G é acíclico. A adição de uma aresta cria um ciclo em G .

Árvore Enraizada

- Tipo especial de árvore que apresenta um vértice (raiz) que se distingue dos demais.
- Utilizamos o termo nó para fazer referência aos vértices.



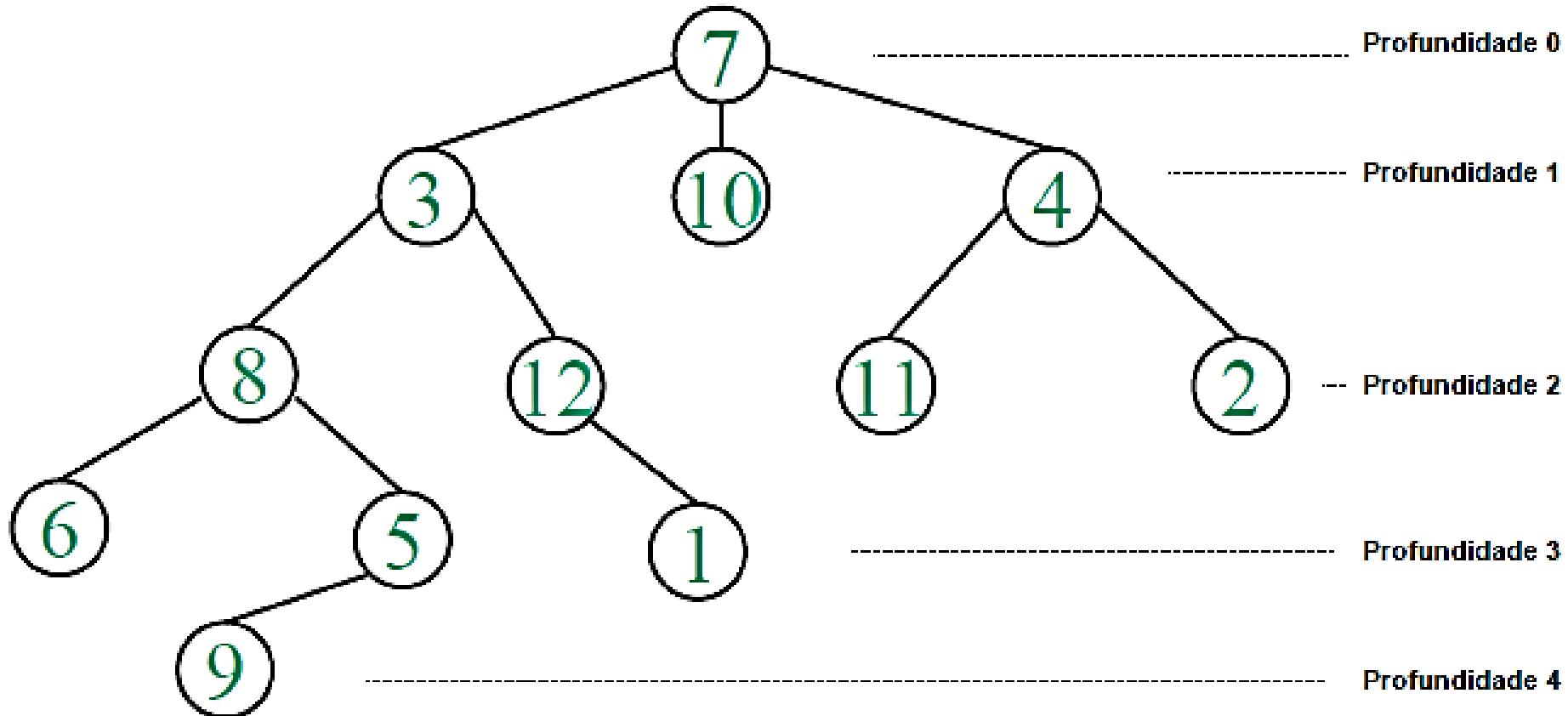
Algumas Definições

- Seja x um nó de uma árvore enraizada T com raiz r :
 - **Ancestral**: é qualquer nó y no caminho de r a x .
 - **Descendente**: x é um descendente de y se y é ancestral de x .
 - **Ancestral Próprio**: y é ancestral próprio de x se y é ancestral de x e $y \neq x$.
 - **Descendente Próprio**: y é descendente próprio de x se y é descendente de x e $y \neq x$.
 - **Sub-árvore enraizada em x** : árvore induzida pelos descendentes de x , com x sendo a raiz.
 - **Filho**: x é filho de y se ele é um descendente direto.
 - **Pai**: é o ancestral próprio mais próximo. A raiz é o único nó sem pai.

Algumas Definições

- Folha: um nó sem filhos.
- Nó interno: um nó que não é folha.
- Grau: o grau de y é o número de filhos de y .
- Profundidade: o comprimento desde a raiz r até x é a profundidade de x em T .
- Altura:
 - a altura de um nó em uma árvore é o maior comprimento do nó até uma folha.
 - A altura de uma árvore é a altura de sua raiz.
 - Altura da árvore é a maior profundidade de qualquer nó da árvore

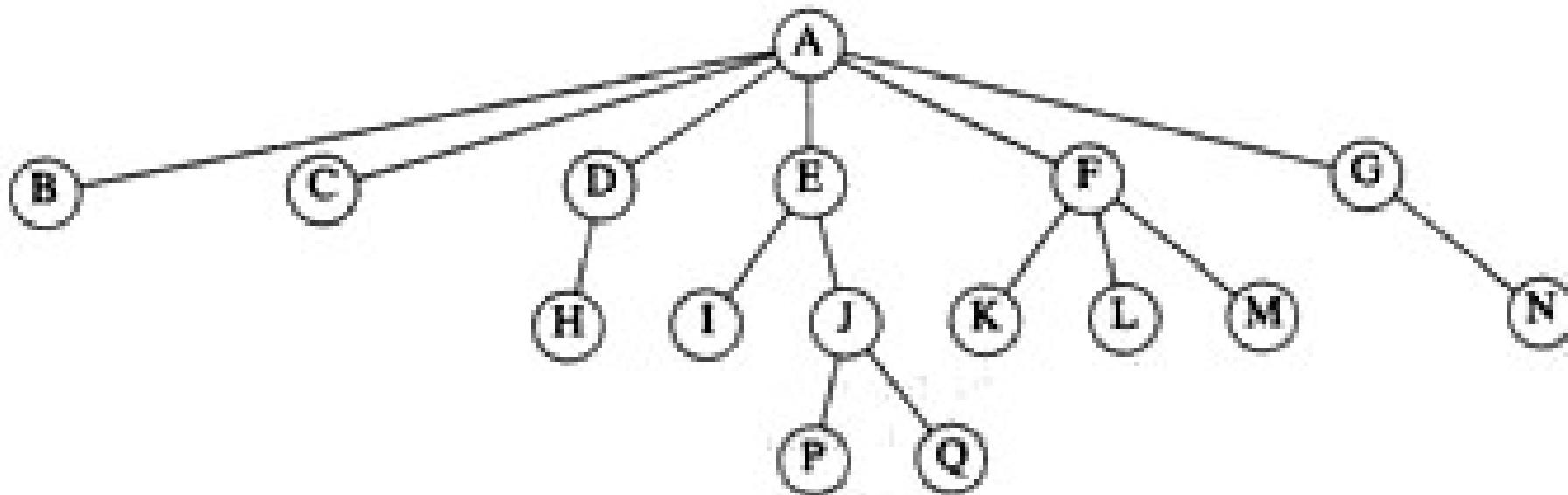
Exemplo



Altura da árvore é 4

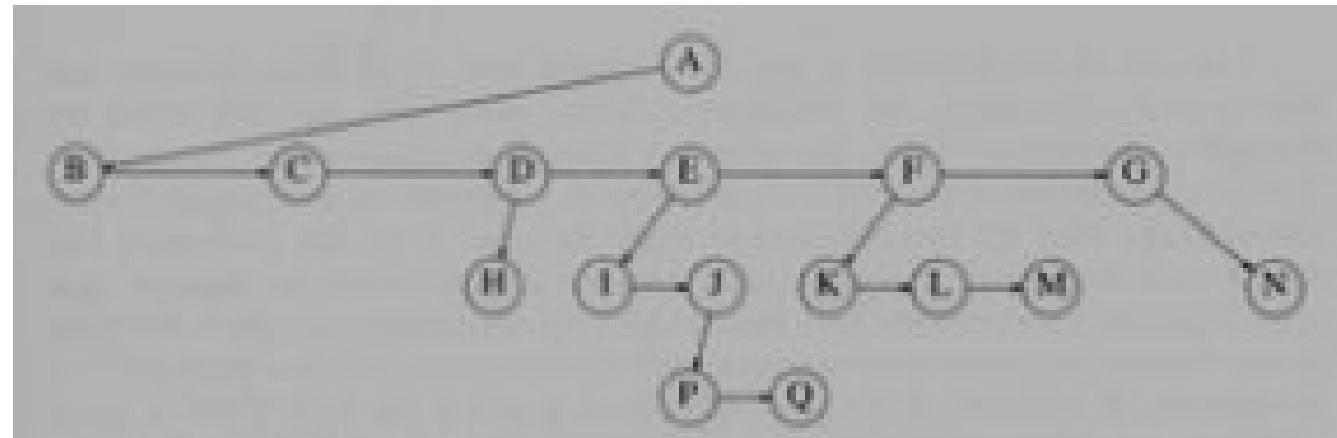
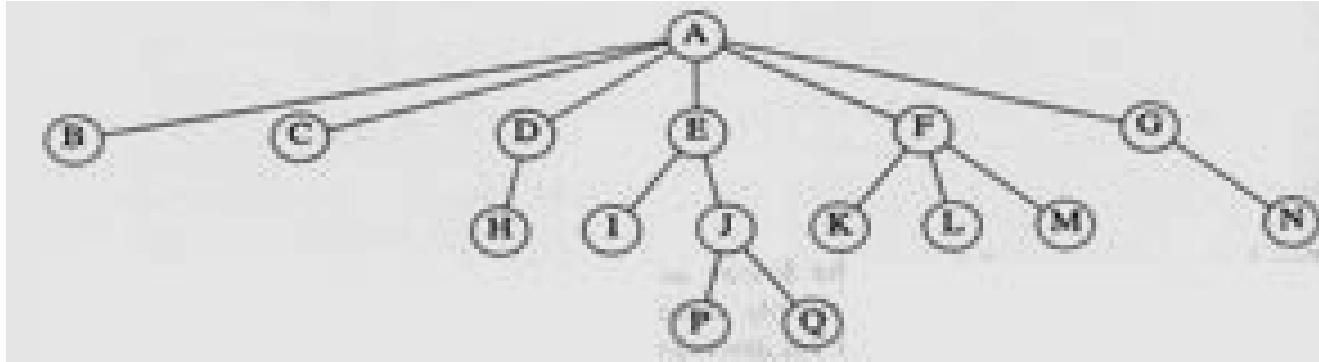
Implementação de Árvores

- Além da informação de cada nó, um link para cada um dos filhos.
- Inconveniente: não sabemos a priori a quantidade de filhos em cada nó.

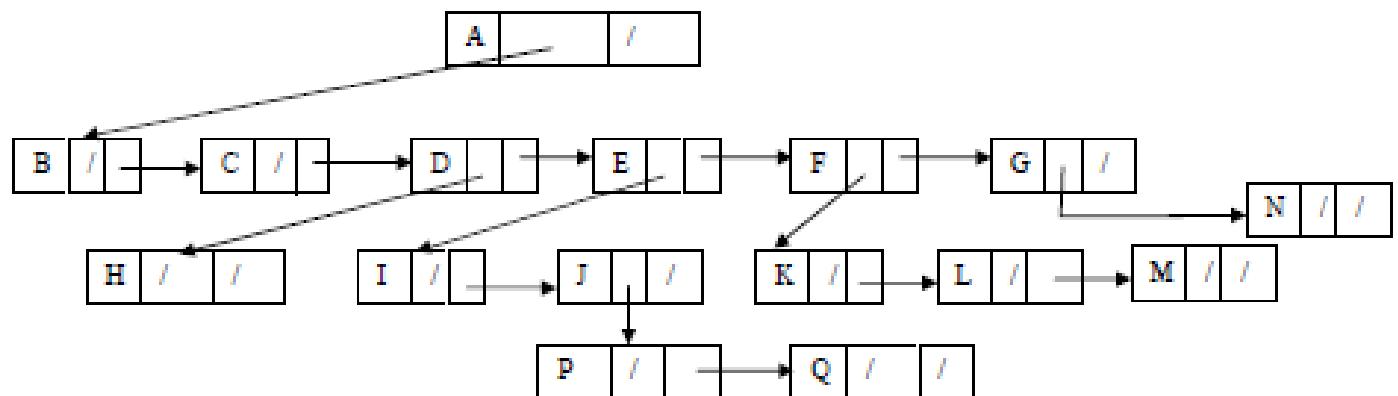
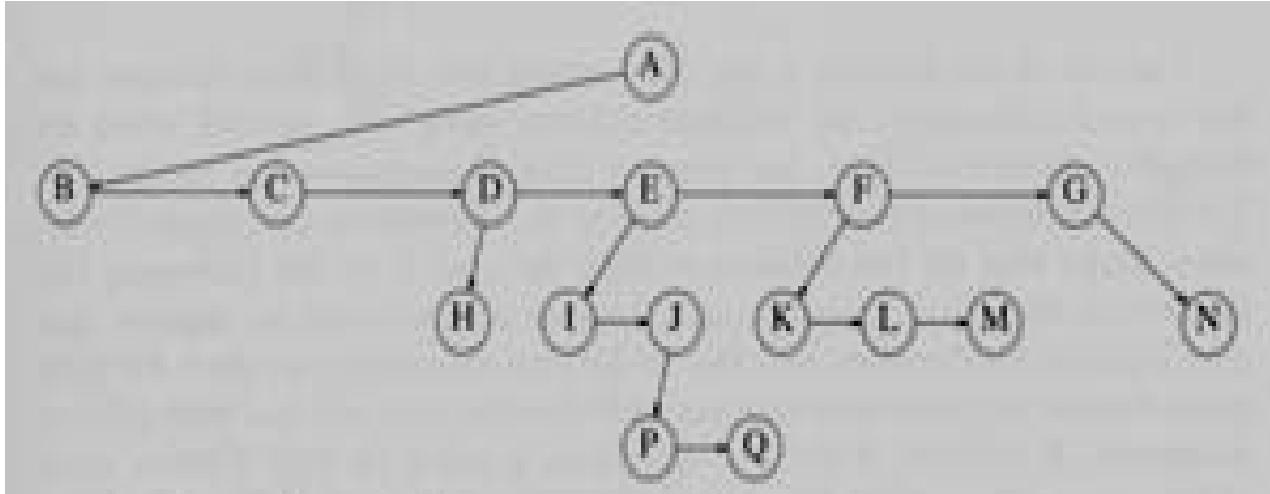


Implementação de Árvores

Os filhos de um nó são mantidos em uma lista encadeada.

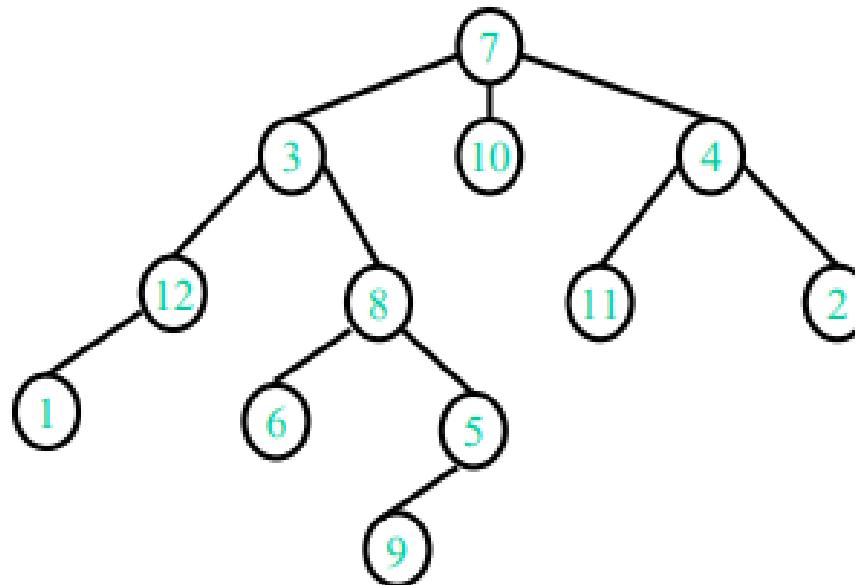
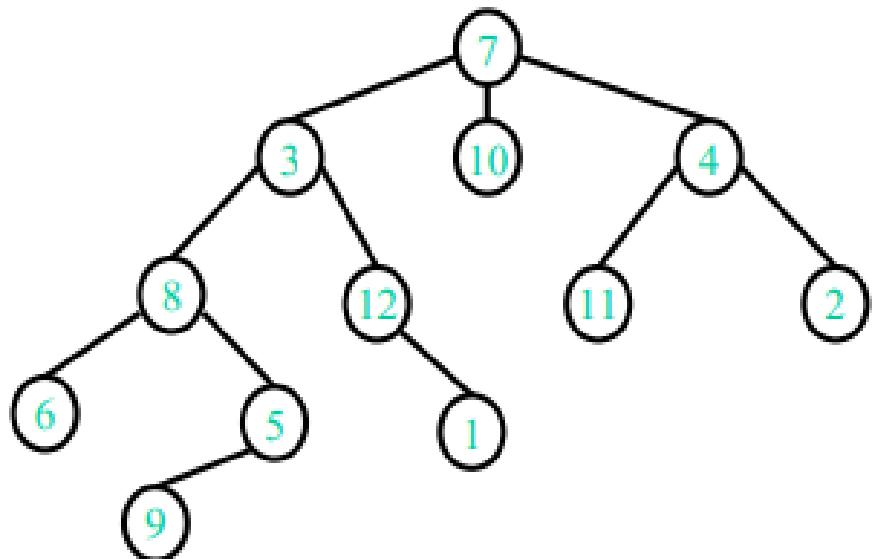


Implementação de Árvores



Árvore Ordenada

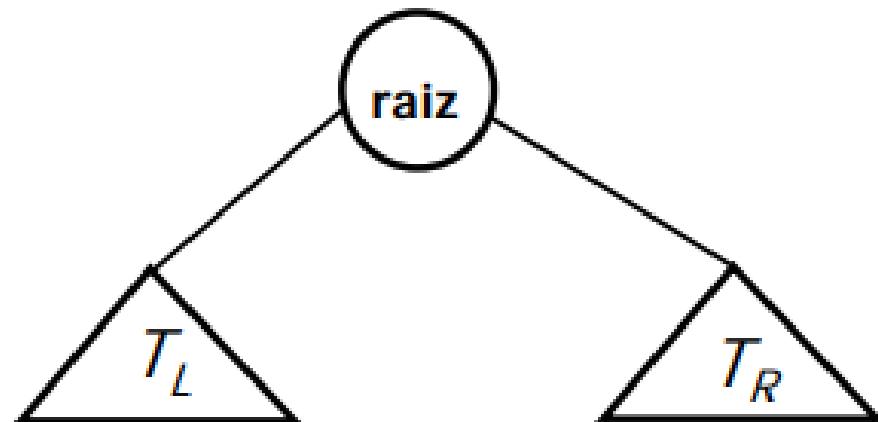
Árvore enraizada em que os filhos de cada nó estão ordenados.



Árvores enraizadas iguais.
Árvores ordenadas diferentes.

Árvore Binária

- Estrutura de nós que é definida recursivamente através de um conjunto de nós:
- – Não contém nenhum nó, ou;
- – Formada por 3 conjuntos disjuntos: um nó raiz, duas subárvores binárias (direita e esquerda).



Árvore Binária – Conceitos Importantes

- Árvore vazia ou nula: não contém nenhum nó.
- Filho da esquerda: raiz da sub-árvore da esquerda (quando houver).
- Filho da direita: raiz da sub-árvore da direita (quando houver).
- Filho ausente: quando a sub-árvore dé a árvore nula

Propriedades de árvores

- Demonstração.
- [⇒] Se G é uma árvore, então, por definição, G é conexo e sem circuitos. Como G é conexo, então existe um caminho entre cada par de vértices. Precisamos mostrar que este caminho é único. Vamos supor que existam dois caminhos distintos entre um par de vértices. Ora, se existem dois caminhos distintos entre um par de vértices então a união destes caminhos contém um circuito. Mas por hipótese, o grafo não possui circuitos, portanto existe apenas um caminho entre cada par de vértices.

- [\Leftarrow]

Vamos mostrar que se existe um, e apenas um, caminho entre cada par de vértices, então G é uma árvore. Como existe um caminho entre cada par de vértices, temos que G é conexo. Vamos supor que G contenha um circuito. A existência de um circuito no grafo implica que existe pelo menos um par de vértices a, b tais que existem dois caminhos distintos entre **a e b** . Mas por hipótese existe um e apenas um caminho entre cada par de vértices e portanto o grafo não tem circuitos. Por definição um grafo conexo e sem circuitos é uma árvore.

Propriedades de árvores

- **Teorema**
- Seja $G(V, A)$ um grafo com n vértices. As seguintes afirmações são equivalentes:
 - a) G é uma árvore.
 - b) G é conexo e possui $n - 1$ arestas.
 - c) G possui $n - 1$ arestas e não possui circuitos.
 - d) Existe exatamente um caminho entre cada par de vértices.
 - e) G não contém circuitos, e para todo $v, w \in V$, a adição da aresta (v, w) produz no grafo exatamente um circuito.

Observação Qualquer uma destas afirmativas pode ser usada como definição de uma árvore.

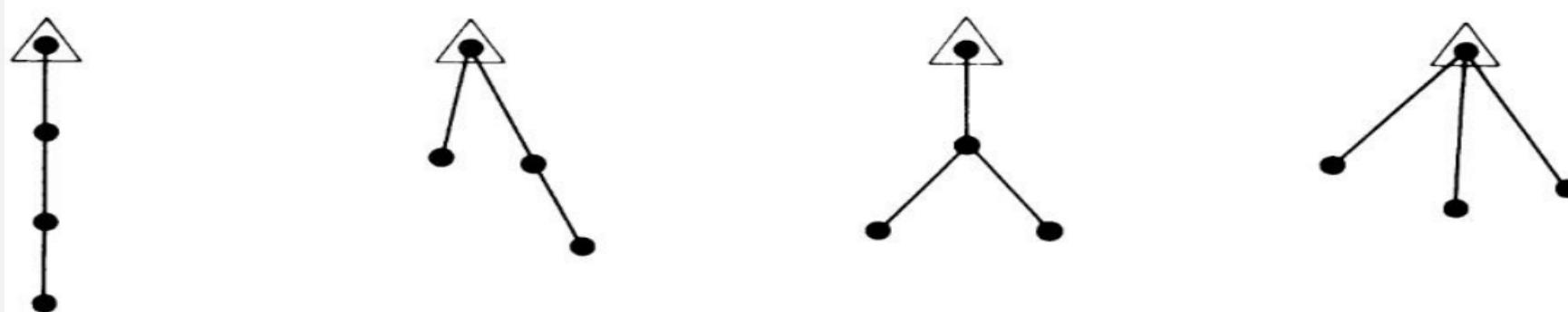
- **Demonstração.**

- Para mostrar a equivalência das afirmativas temos que mostrar que
 - $a) \Rightarrow b)$, $b) \Rightarrow c)$, etc.
 - Vamos mostrar apenas que $a) \Rightarrow b)$: Se G é uma árvore, então G é conexo e possui $n - 1$ arestas.
 - Como por hipótese G é uma árvore, temos que G é conexo.
 - Precisamos mostrar apenas que G possui $n - 1$ arestas. Vamos mostrar usando indução matemática sobre n .
 - Vamos verificar o resultado para um valor particular de n . Por exemplo para $n = 1$ e $n = 2$.
 - Para $n = 1$, temos 0 arestas.
 - Para $n = 2$ temos 1 aresta

Vamos supor agora que o resultado vale para um grafo G' com $k - 1$ vértices. Isto é, se G' é uma árvore então G' é conexo e possui $k - 2$ arestas. Vamos acrescentar uma nova aresta (v,w) a este grafo. Para manter o grafo conexo e sem circuitos um e apenas um dos vértices em (v,w) pode pertencer a G' . Assim ao acrescentar a aresta (v,w) a G' , precisamos acrescentar também um vértice. Assim teremos um novo grafo G'' com k vértices e $k - 1$ arestas. A forma como G'' foi construído garante que é conexo e sem circuitos. Portanto temos que G'' é uma árvore. Mostramos assim que se G é uma árvore então G é conexo com $n - 1$ arestas.

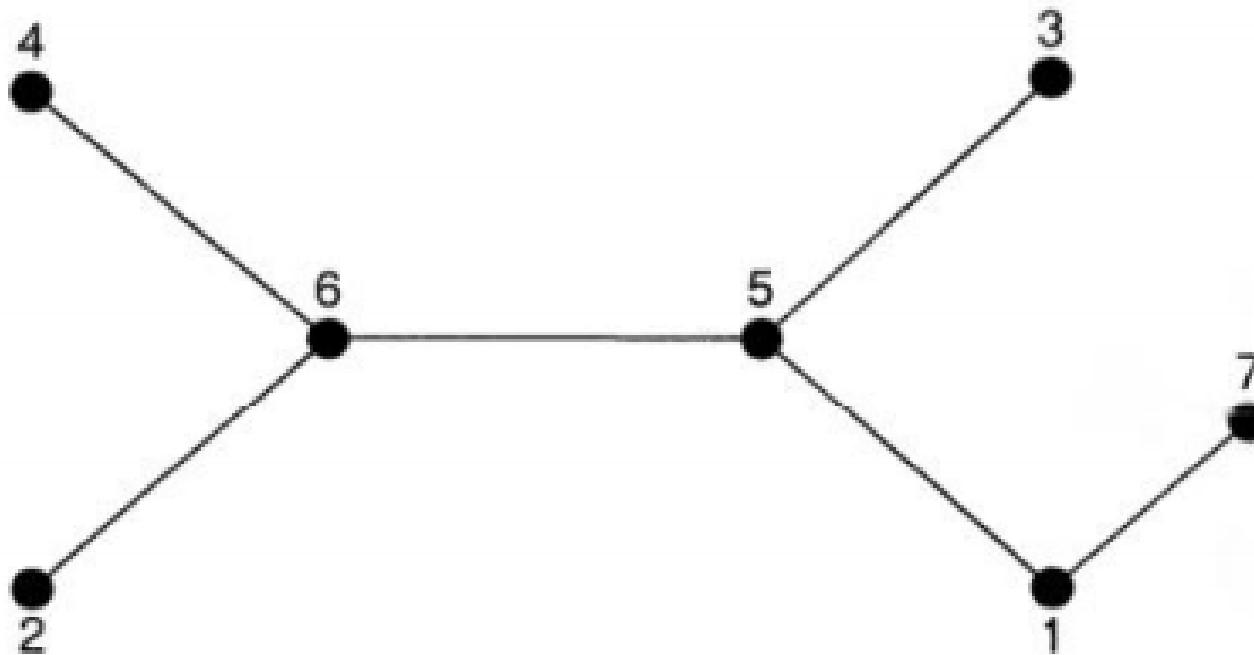
Raízes e Árvores Binárias

- **Definição**
 - Uma árvore na qual podemos distinguir um determinado vértice, denominado vértice raiz, é chamada de árvore enraizada.
- Por exemplo as árvores de 4 vértices abaixo são enraizadas.



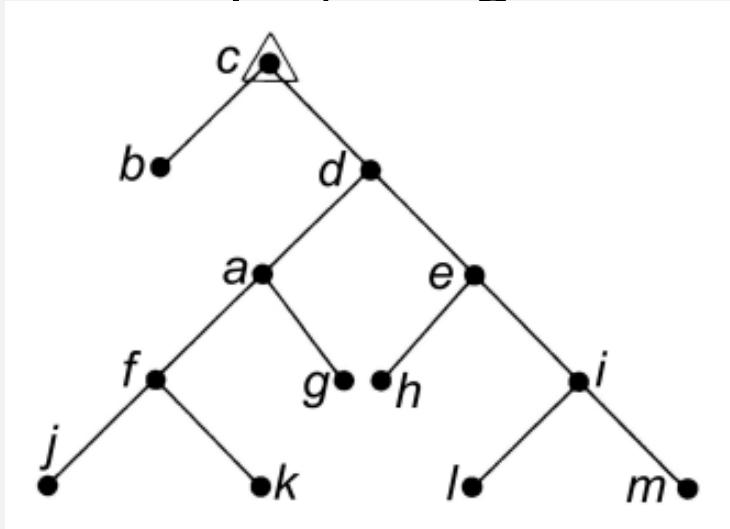
Em geral, o vértice raiz aparece naturalmente com a aplicação que o grafo representa.

- Uma árvore não enraizada é chamada de árvore livre.



Raízes e Árvores Binárias

- Se representarmos uma árvore enraizada com o vértice raiz posicionado na parte superior da figura, podemos definir níveis na árvore. Considere, por exemplo, a seguinte árvore enraizada:



Dizemos que o vértice raiz, c, está no nível zero; os vértices b e d no nível 1, os vértices a, e e no nível 2, os vértices f , g, h e i no nível 3 e j, k, l e m no nível 4.

- Definição

- A distância entre dois vértices v e w em um grafo G , denotada por $d(v,w)$, é igual ao comprimento do menor caminho entre v e w .

- Definição

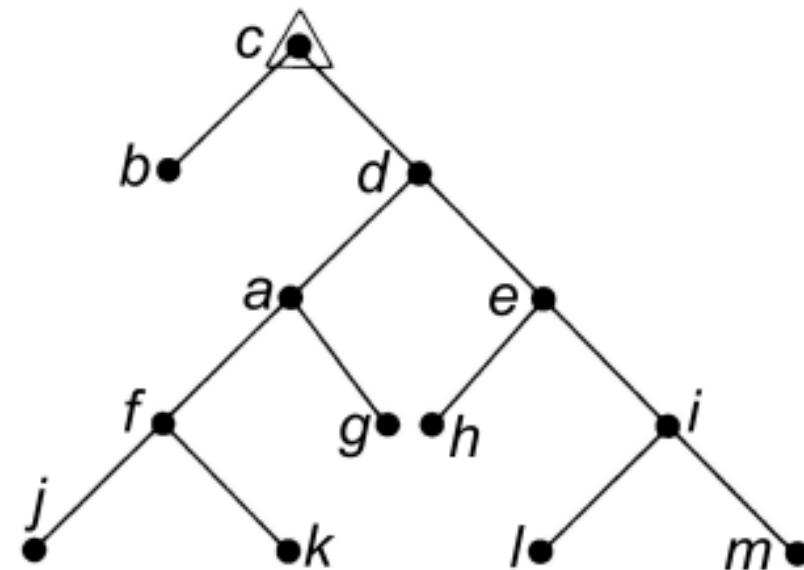
- O nível de um vértice x em uma árvore enraizada é igual à distância entre o vértice raiz e o vértice x . A altura de uma árvore enraizada é o comprimento do maior caminho existente na árvore a partir do vértice raiz.

Definições

- **Definição :**
 - Uma árvore binária completa é uma árvore enraizada tal que existe exatamente um vértice de grau dois e cada um dos vértices restantes tem grau 1 ou 3. Naturalmente o vértice de grau 2 é o vértice raiz da árvore.
- **Definição :**
 - Um vértice não pendente em uma árvore é chamado de vértice interno.

Exemplo

- Abaixo temos um exemplo de árvore binária completa.



- Os vértices d, a, e, f , i são vértices internos.

Propriedades de árvores binárias

- **Proposição**
 - O número de vértices em uma árvore binária completa (com três ou mais vértices) é sempre ímpar.
 - **Demonstração.**
 - Existe exatamente um vértice de grau par. Os $n - 1$ vértices restantes tem grau ímpar. Mas sabemos que o número de vértice com grau ímpar é par. Portanto, se $n - 1$ é par, n é ímpar.

Proposição

Quantos vértices pendentes existem em uma árvore binária completa com n vértices?

Demonstração.

Seja p o número de vértices pendentes. Então, existem $n - p - 1$ vértices de grau 3. Além disso,

$$\sum_{i=1}^n d(v_i) = 2m = 2(n - 1),$$

onde usamos uma propriedade de árvores na última igualdade. Somando os graus dos vértices pendentes, internos e raiz tem-se $p + 3(n - p - 1) + 2 = 2(n - 1)$. Logo $p = (n + 1)/2$.

- **Exemplo:**

Quantos jogos são necessários em um torneio de tênis com 56 inscritos? Se representarmos a competição através de uma árvore binária, teremos que os vértices pendentes são os inscritos e os vértices internos mais a raiz os jogos. Assim, queremos calcular o número de vértices internos mais a raiz em uma árvore binária com 56 vértices pendentes.

Propriedades de árvores binárias

Proposição

Seja T uma árvore binária completa de altura h e p vértices pendentes. Então:

$$p \leq 2^h, \tag{1}$$

$$h \geq \lceil \log_2 p \rceil = \lceil \log_2 ((n+1)/2) \rceil = \lceil \log_2 (n+1) - 1 \rceil. \tag{2}$$

Demonstração.

Este resultado pode ser provado da seguinte forma:

- (1) princípio de indução;
- (2) aplica-se logaritmo a ambos os lados da expressão em (1).



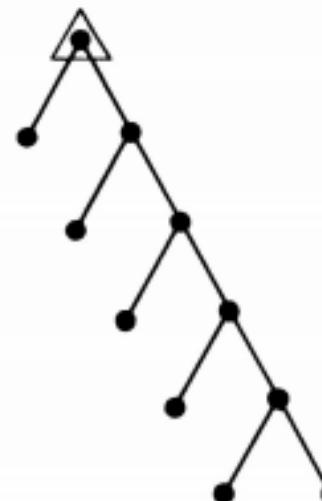
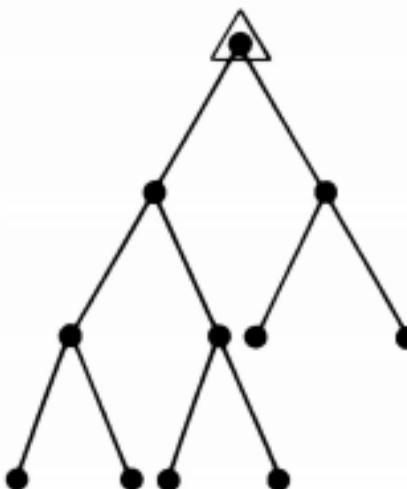
Exercício

Qual é o nível máximo de uma árvore binária com n vértices?

Observando que a maior altura da árvore será obtida com o menor número possível de vértices em cada nível, temos que em uma árvore com n vértices,

$$h \leq (n - 1)/2.$$

Abaixo temos um exemplo de níveis mínimo e máximo em uma árvore binária completa com 11 vértices.



Procedimentos de busca em árvores

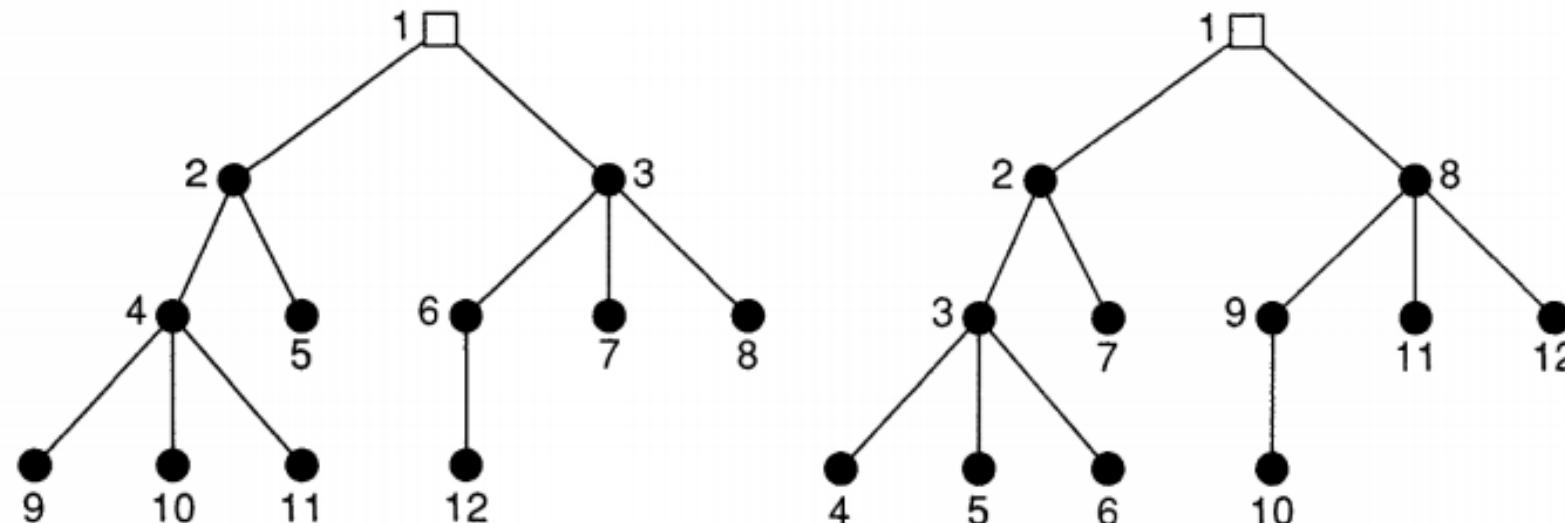
- Árvore binárias são muito utilizadas em procedimentos de busca. Considere que cada vértice da árvore representa um teste com duas respostas possíveis. Iniciando o teste no vértice raiz, a resposta ao teste nos leva a um dos dois vértices do próximo nível onde novos testes são efetuados. Quando atingimos um vértice pendente (o objetivo da busca) o procedimento de busca se encerra. Em algumas aplicações é importante construir árvores tais que os vértices pendentes estejam o mais próximo possível do vértice raiz.

Procedimentos de busca em árvores

Há dois procedimentos de busca bem conhecidos:

- Busca em profundidade (depth first);
- Busca em largura (breadth first).

Exemplo:

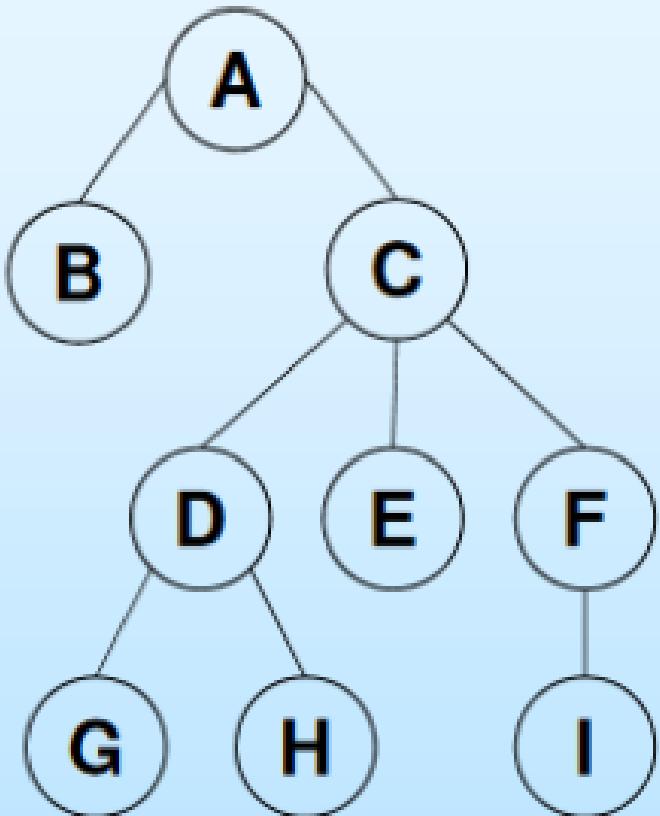


ÁRVORES

Elementos de uma árvore

- **Nó:** Elemento que contém a informação
- **Arco:** Liga dois nós
- **Pai:** nó superior de um arco
- **Filho:** nó inferior de um arco
- **Raiz:** nó topo – não tem um nó pai
- **Folhas:** nós das extremidades inferiores – não têm nós filhos.
- **Grau:** Representa o número de subárvore de um nó. Ver exemplo no próximo slide.
- **Grau de uma árvore** (aridade): é definido como sendo igual ao máximo dos graus de todos os seus nós. A árvore do próximo slide

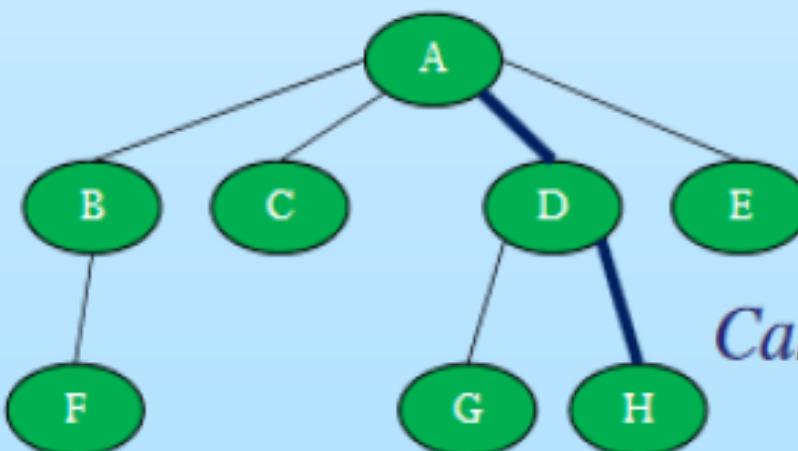
Elementos de uma árvore



- **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**

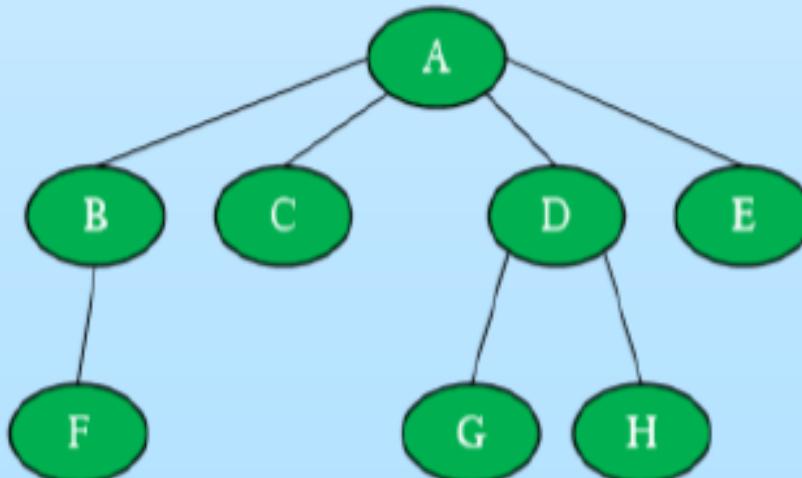
$$\text{Grau}(T) = 3$$

- Cada nó tem que ser atingível a partir da raiz através de uma sequência única de arcos, chamados de **caminho**.
- Comprimento do caminho: o número de arcos do caminho
 - O caminho de A até H tem comprimento 2
- **Nível de um nó:** é a sua distância da raiz da árvore. A raiz tem nível 0. Na fig. abaixo, o nó A tem nível 0; os nós B, C, D e E têm nível 1 e os nós F, G e H têm nível 2.

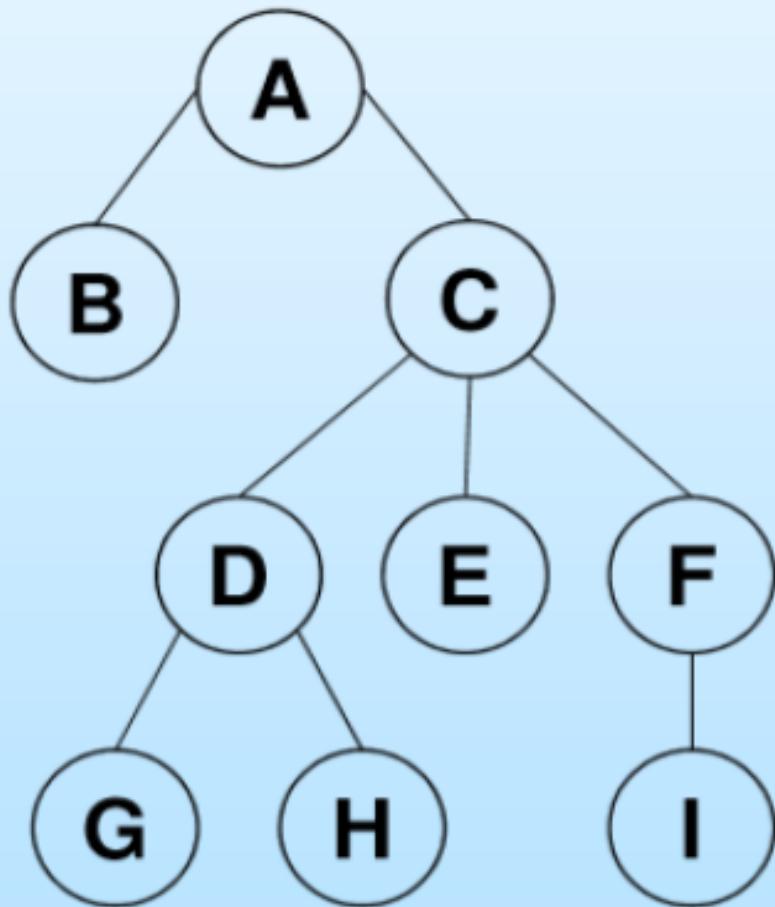


Caminho do nó A ao H.

- **Altura (ou profundidade)** é o nível do nó folha que tem o mais longo caminho até a raiz, somando 1.
 - A altura da árvore abaixo é igual a 3.
 - A árvore vazia é uma árvore de altura -1, por definição.
 - Uma árvore com um único nó tem altura 1.
 - O nó é raiz e folha ao mesmo tempo.
- Toda árvore com $n > 1$ nós possui no mínimo 1 e no máximo $n - 1$ folhas.



Exemplo de níveis e altura da árvore)

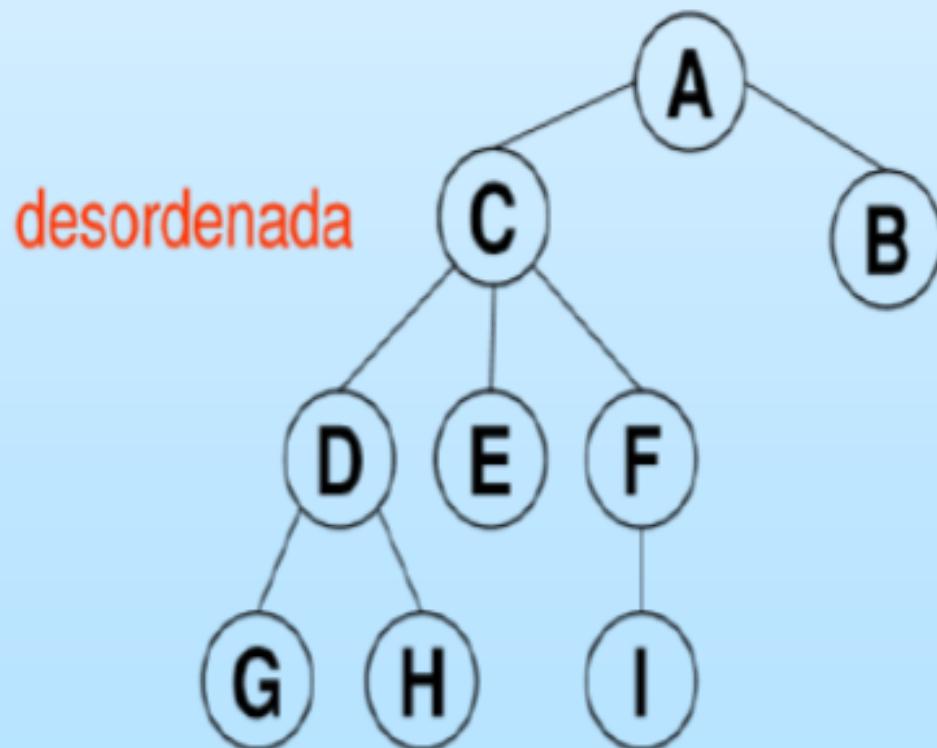
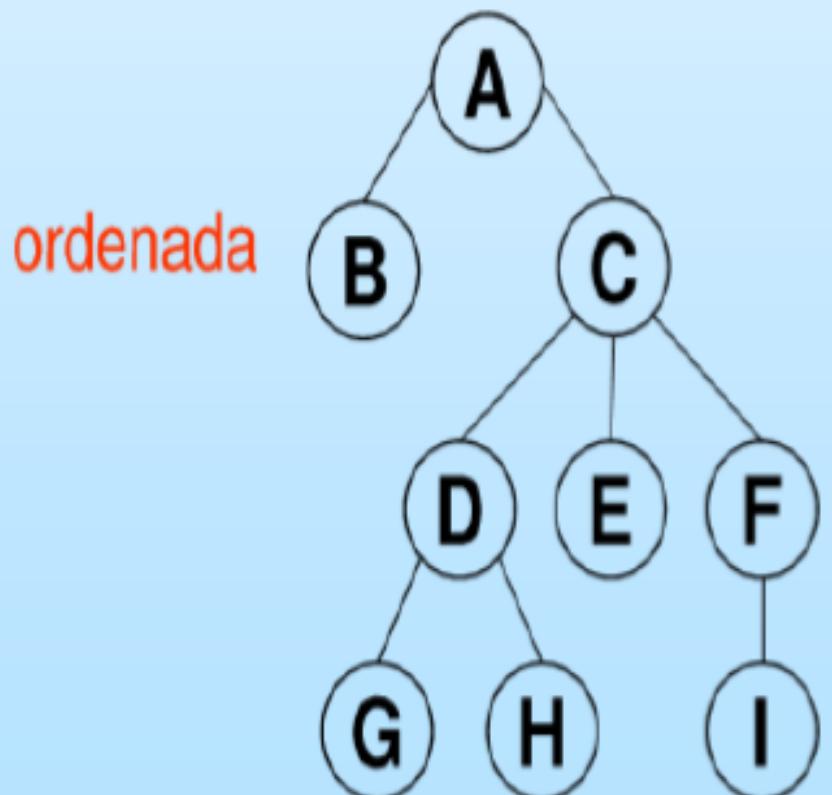


NÍVEIS	
A	0
B, C	1
D, E, F	2
G, H, I	3

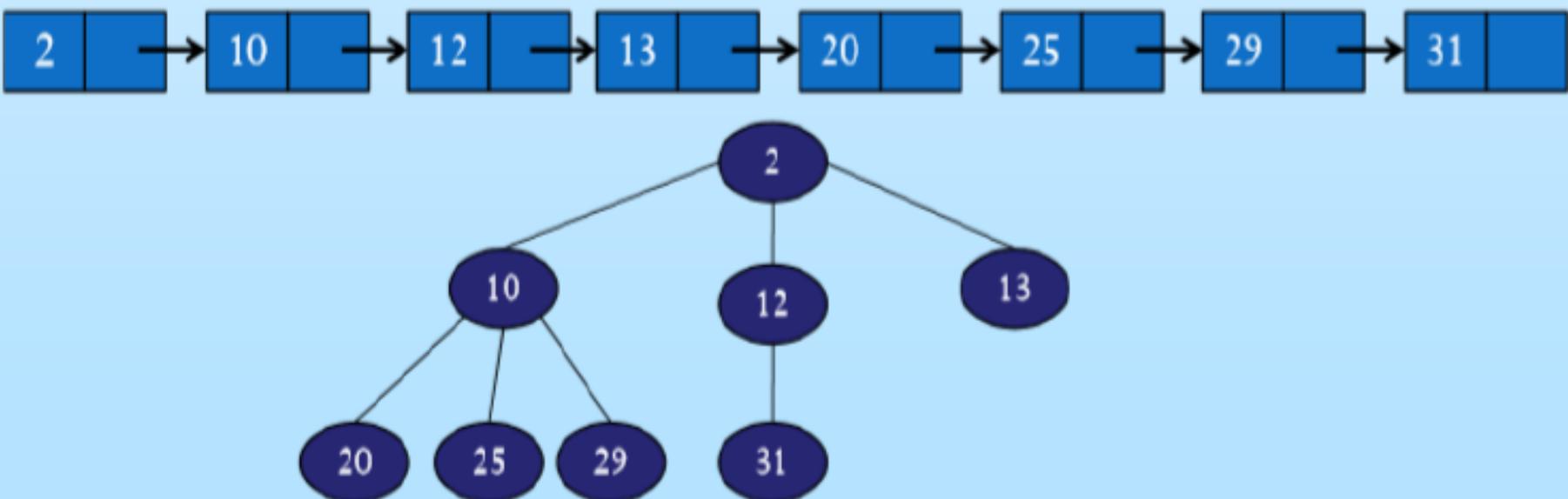
$$h(T) = 4$$

- Árvore Ordenada

- Os filhos de cada nó estão ordenados (assume-se ordenação da esquerda para a direita)

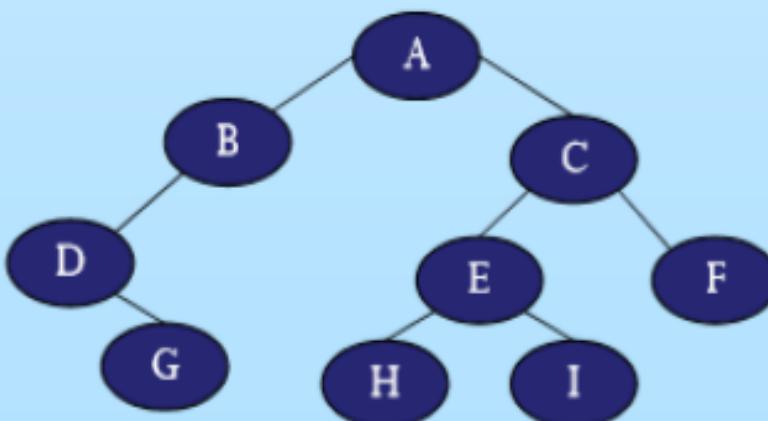


- A definição de árvore não impõe qualquer condição sobre o número de filhos de um nó:
 - Pode variar de 0 a qualquer inteiro
- Árvores são muito utilizadas em sistemas gerenciadores de banco de dados.
- Considerando a lista encadeada e a árvore abaixo, qual pesquisa é mais rápida para achar um valor (chave)?

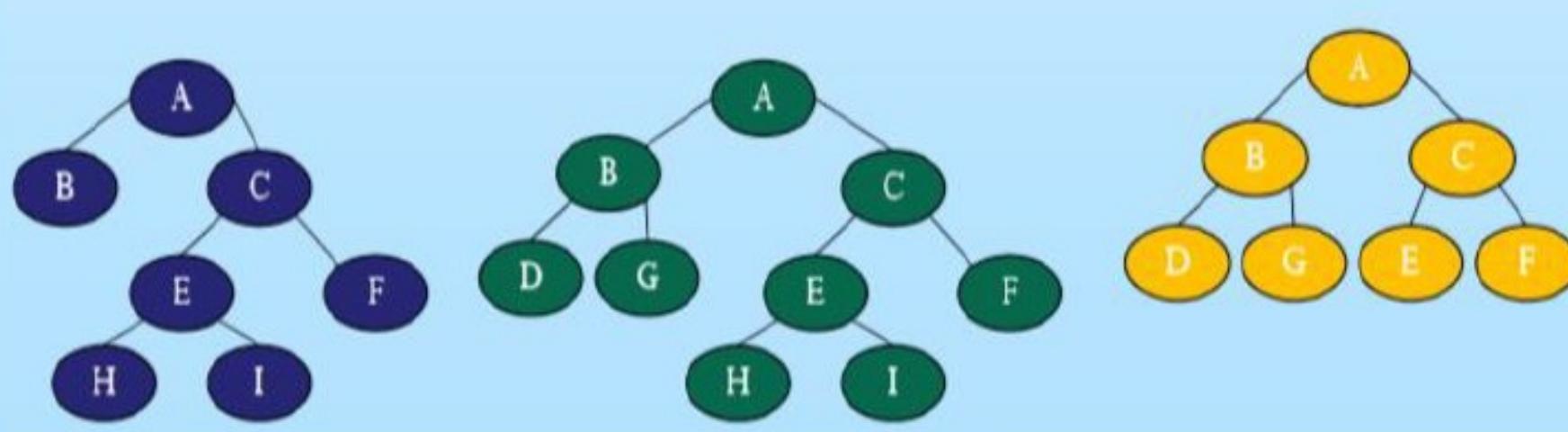


• Definição

- Uma árvore binária T é um conjunto finito de elementos denominados *nós* ou vértices, tal que:
 - $T = \emptyset$ e a árvore é dita vazia, ou
 - Existe um nó especial r , chamado *raiz* de T , e os restantes podem ser divididos em dois subconjuntos disjuntos, T_r^L e T_r^R a subárvore esquerda e a direita de r , respectivamente, as quais são também árvores binárias.

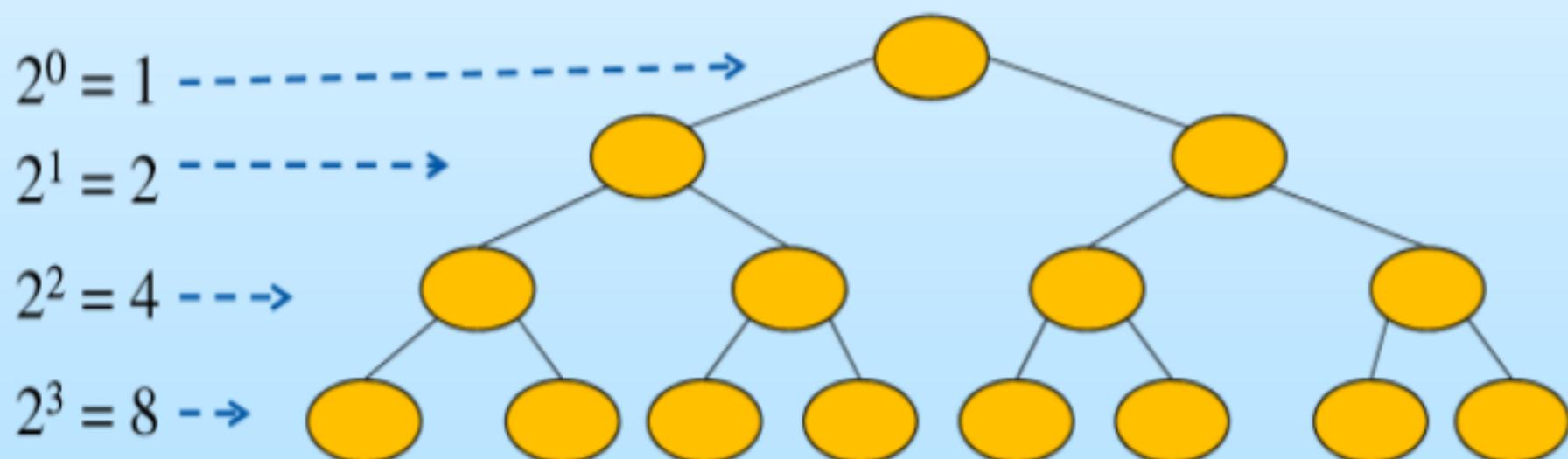


- **Árvore estritamente binária**
 - Cada nó possui 0 ou 2 filhos.
- **Árvore binária completa** apresenta a seguinte propriedade
 - Se v é um nó tal que alguma subárvore de v é vazia, então v se localiza ou no último (maior) ou no penúltimo nível da árvore.
- **Árvore binária cheia** apresenta a seguinte propriedade:
 - Se v é um nó tal que alguma subárvore de v é vazia, então v se localiza no último (maior) nível da árvore. v é um nó folha.



Árvore binária

- Em árvore binária cheia o número de nós do nível i é igual a 2^i .
- Consequentemente, em qualquer árvore binária existe no máximo 2^i nós no nível i .



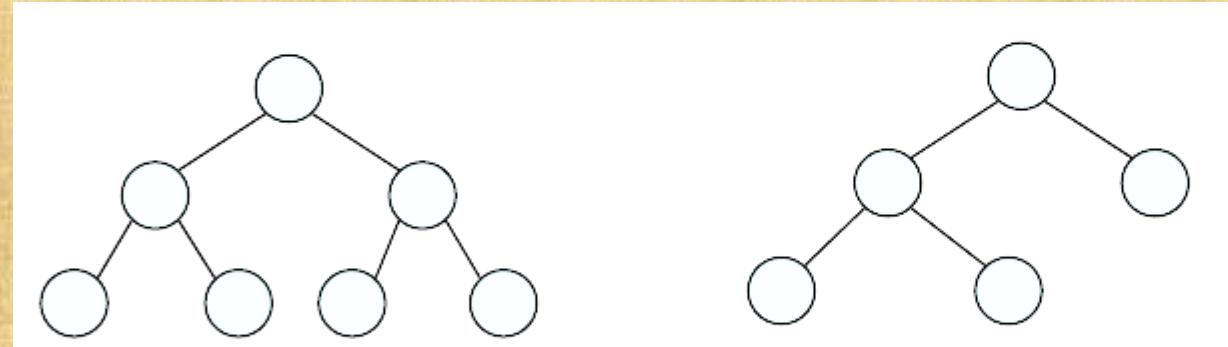
ÁRVORES AVL

ÁRVORES BALANCEADAS

- As árvores binárias de pesquisa são, em alguns casos, pouco recomendáveis para as operações básicas (remoção e busca);
- Árvores binárias de pesquisa degeneradas tornam as operações básicas lentas $O(n)$;

Árvore binária completamente balanceada

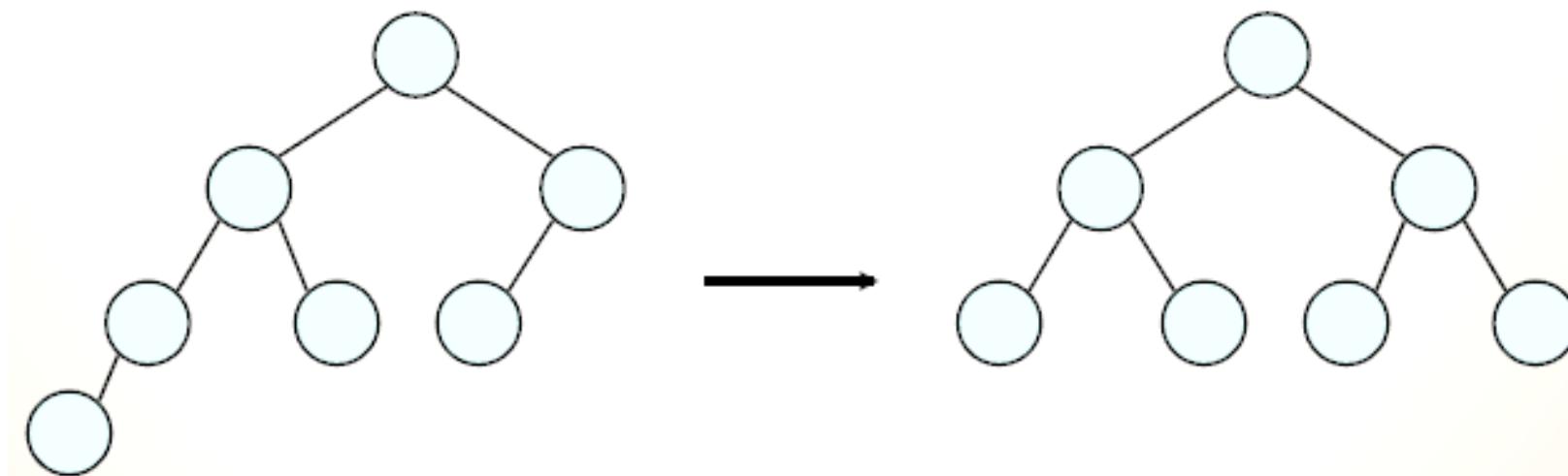
* Ocorre quando a árvore está cheia ou quase cheia com o nível $n - 1$ completo;



*Uma árvore binária completa leva um tempo na ordem de $O(\log n)$ para operações de inserção, remoção e pesquisa. O que é, sem dúvida, muito bom;

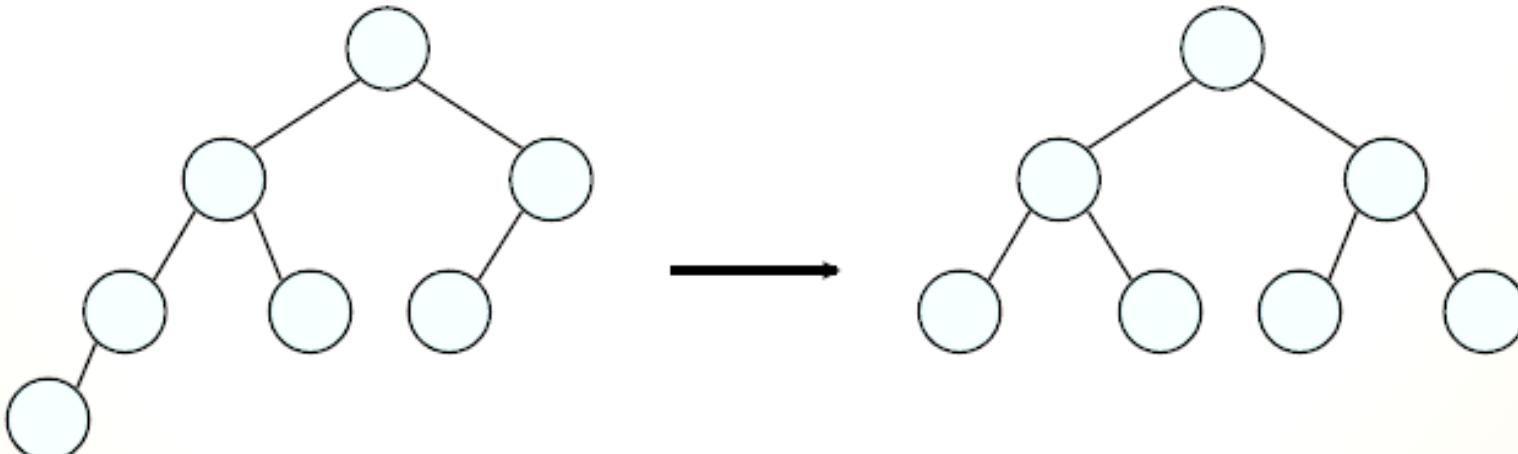
Árvore binária completamente balanceada

Após uma inserção ou remoção a árvore pode deixar de ser completa. A solução seria aplicar um algoritmo que tornasse a árvore novamente completa, porém o custo para realizar esta operação seria de $O(n)$.



Árvore binária completamente balanceada

- Percebe-se que todos os nós tiveram sua posição na estrutura alterados.
- Na maioria dos casos, utiliza-se árvores quase平衡adas.



CRITÉRIOS PARA DEFINIR BALANCEAMENTO

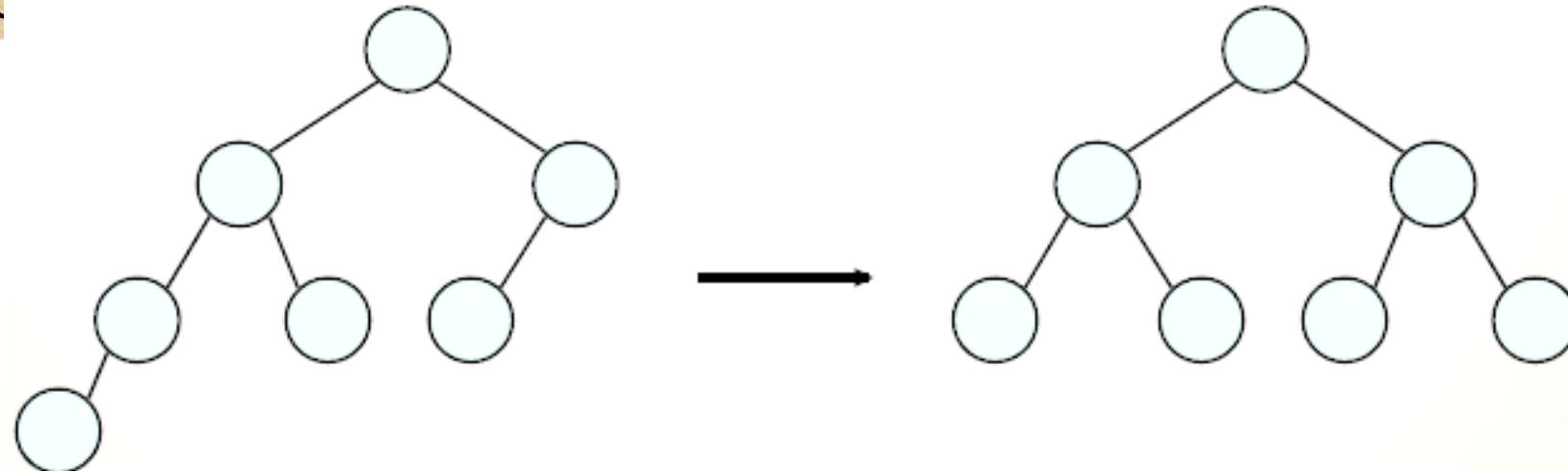
- Vários são os critérios (métodos) para definir balanceamento.

Alguns são:

- Restrições imposta na diferença das alturas das sub árvores de cada nó.

Ex. AVL

- Todos os

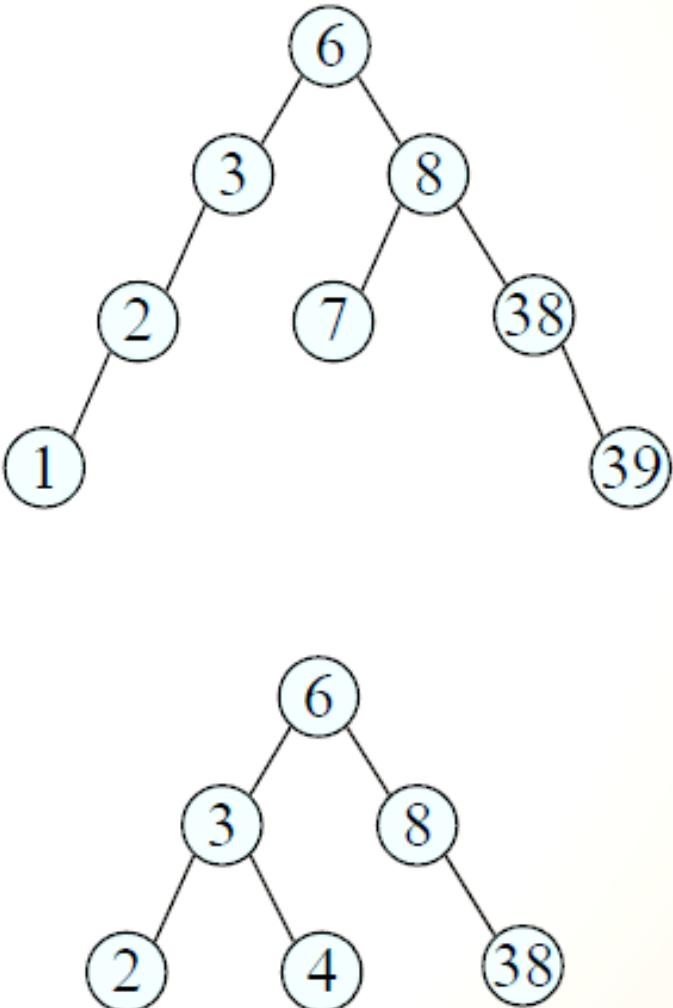
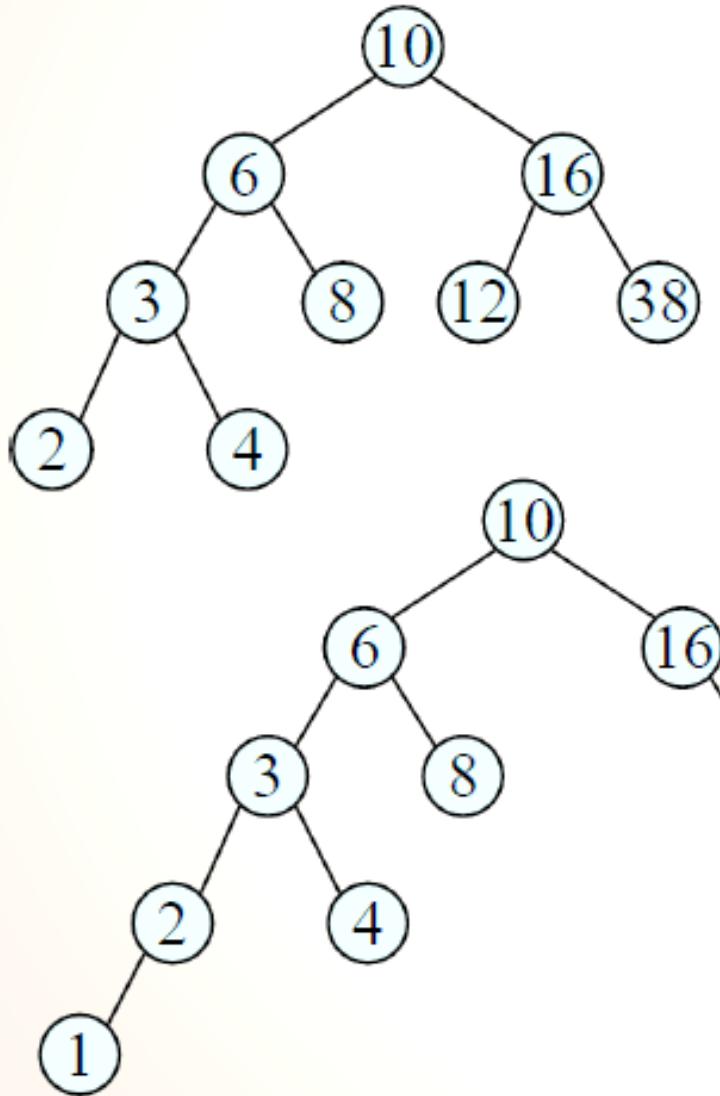


ÁRVORES AVL

- Foram introduzidas por Adelson-Velskii e Landis em 1962;
- São baseadas em árvore binárias de pesquisa
- A medida em que as operações de inserção e remoção são efetuadas a árvore é balanceada.

- Definição:
 - Uma árvore binária T é dita AVL quando, para qualquer nó v de T , a diferença entre a altura das subárvore esquerda $h_e(v)$ e direita $h_d(v)$ é no máximo em módulo igual a 1.

ÁRVORES AVL



OBS.: se uma árvore T é dita AVL, então todas as suas sub árvores também são AVL;

- Balanceamento de um nó
 - O fator de balanceamento:
 - É dado pela altura da subárvore da esquerda $h_e(v)$ menos a altura da subárvore da direita $h_d(v)$.

$$FB(v) = h_e(v) - h_d(v)$$

ÁRVORES AVL

- Nós balanceados
 - São aqueles onde os valores de FB são -1, 0 ou 1
 - $FB(v)$:
 - +1: subárvore esquerda mais alta que a direita
 - 0: subárvore esquerda igual a direita
 - 1: subárvore direita mais alta do que a esquerda

ÁRVORES AVL

- Balanceamento de um nó
 - O fator de balanceamento:
 - É dado pela altura da subárvore da esquerda $h_e(v)$ menos a altura da subárvore da direita $h_d(v)$.

$$FB(v) = h_e(v) - h_d(v)$$

ÁRVORES AVL

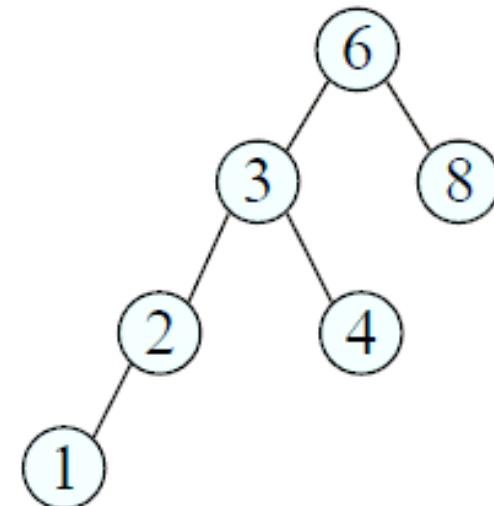
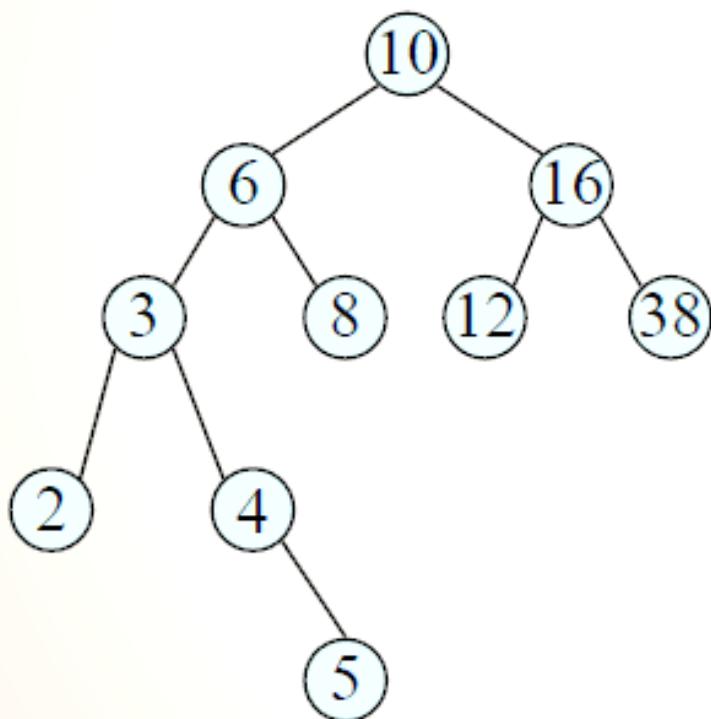
- Nós balanceados
 - São aqueles onde os valores de FB são -1, 0 ou 1
- $FB(v)$:
 - +1: subárvore esquerda mais alta que a direita
 - 0: subárvore esquerda igual a direita
 - 1: subárvore direita mais alta do que a esquerda

ÁRVORES AVL

- Nós desregulados ou desbalanceados
 - São aqueles onde os valores de FB são diferentes de -1, 0 ou 1
 - $FB(v)$:
 - >1 : subárvore esquerda está desbalanceando o nó v
 - <-1 : subárvore direita está desbalanceando o nó v

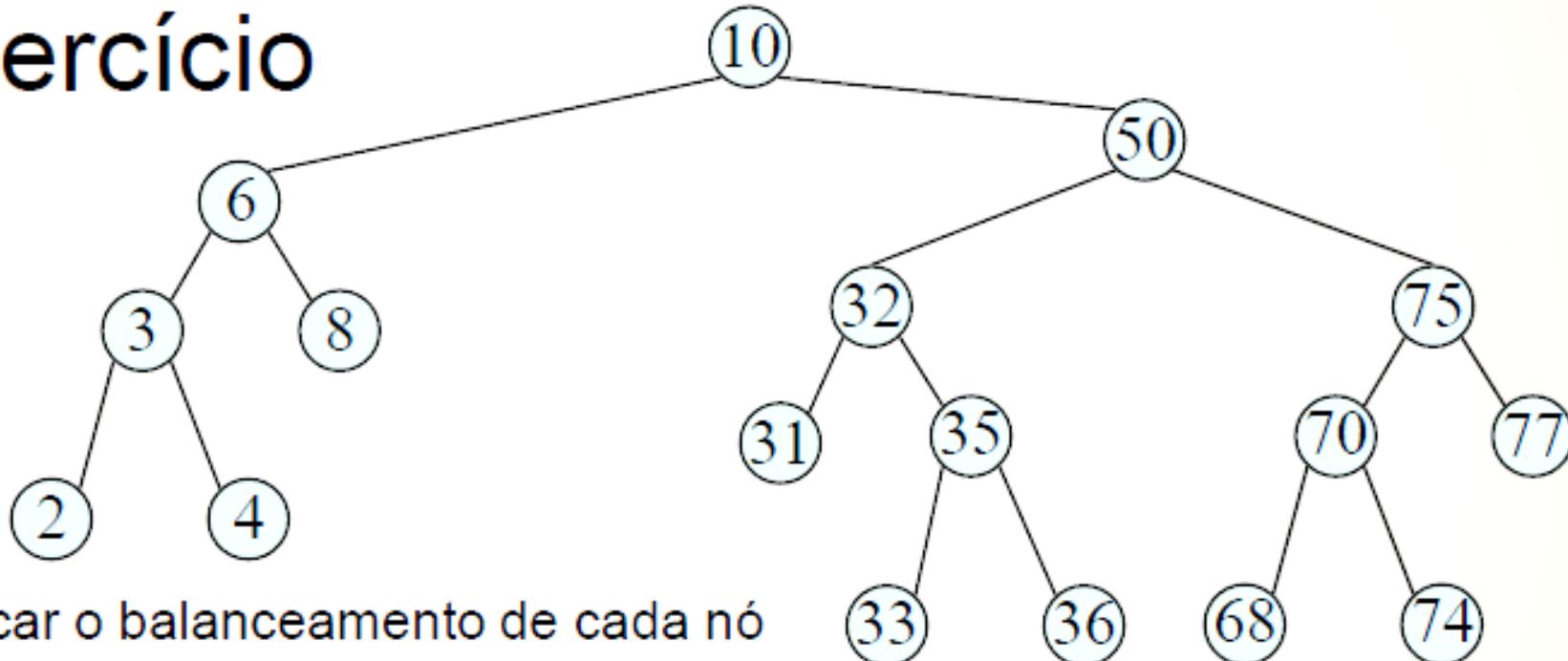
ÁRVORES AVL

- Exemplos



ÁRVORES AVL

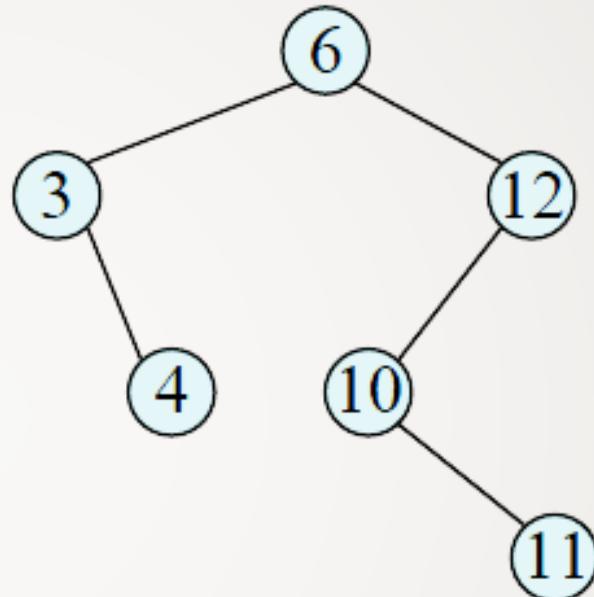
- Exercício



- Colocar o balanceamento de cada nó
- Dizer se a árvore é AVL
- Verificar quais as possíveis posições para a inserção de elementos e em quais posições de inserção, a árvore é AVL

ÁRVORES AVL

- Exercício 2



- Colocar o balanceamento de cada nó
- Dizer se a árvore é AVL
- Verificar quais as possíveis posições para a inserção de elementos e em quais posições de inserção, a árvore é AVL

