

Algoritmos e Estruturas de Dados I

# Árvores

Prof. Tiago Eugenio de Melo  
[tmelo@uea.edu.br](mailto:tmelo@uea.edu.br)

[www.tiagodemelo.info](http://www.tiagodemelo.info)

# Observações

- O conteúdo dessa aula é parcialmente proveniente do Capítulo 13 do livro “*Data Structure and Algorithms Using Python*”.
- As palavras com a fonte `Courier` indicam uma palavra-reservada da linguagem de programação.

# Árvores

# Introdução

# Introdução

- São exemplos de estruturas que organizam os dados de maneira linear (sequencial):

# Introdução

- São exemplos de estruturas que organizam os dados de maneira linear (sequencial):
  - Array.

# Introdução

- São exemplos de estruturas que organizam os dados de maneira linear (sequencial):
  - Array.
  - Lista.

# Introdução

- São exemplos de estruturas que organizam os dados de maneira linear (sequencial):
  - Array.
  - Lista.
  - Lista ligada.

# Introdução

- São exemplos de estruturas que organizam os dados de maneira linear (sequencial):
  - Array.
  - Lista.
  - Lista ligada.
  - Pilhas.

# Introdução

- São exemplos de estruturas que organizam os dados de maneira linear (sequencial):
  - Array.
  - Lista.
  - Lista ligada.
  - Pilhas.
  - Filas.

# Introdução

- São exemplos de estruturas que organizam os dados de maneira linear (sequencial):
  - Array.
  - Lista.
  - Lista ligada.
  - Pilhas.
  - Filas.
- Todas têm uma noção de *anterior e posterior*.

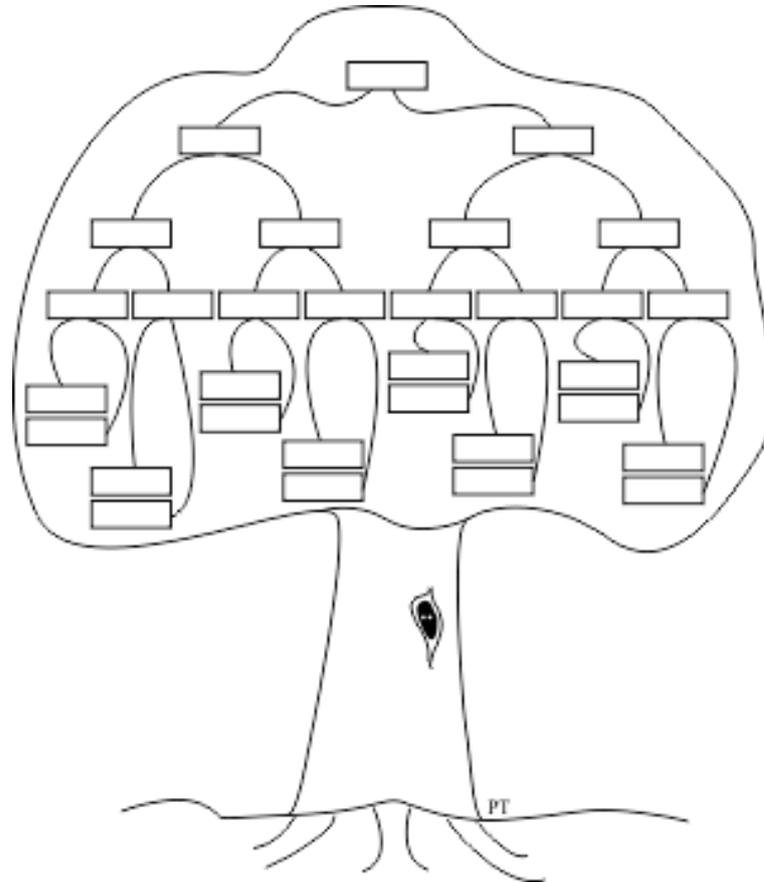
# Introdução

# Introdução

- A estrutura de dados do tipo árvore permite organizar os dados de maneira **hierárquica**.

# Introdução

- A estrutura de dados do tipo árvore permite organizar os dados de maneira **hierárquica**.



# Introdução

# Introdução

- Útil para uma série de problemas:

# Introdução

- Útil para uma série de problemas:
  - Mineração de dados.

# Introdução

- Útil para uma série de problemas:
  - Mineração de dados.
  - Sistemas de banco de dados.

# Introdução

- Útil para uma série de problemas:
  - Mineração de dados.
  - Sistemas de banco de dados.
  - Inteligência artificial.

# Introdução

- Útil para uma série de problemas:
  - Mineração de dados.
  - Sistemas de banco de dados.
  - Inteligência artificial.
  - Sistemas operacionais.

# Introdução

- Útil para uma série de problemas:
  - Mineração de dados.
  - Sistemas de banco de dados.
  - Inteligência artificial.
  - Sistemas operacionais.
  - ...

# Introdução

# Introdução

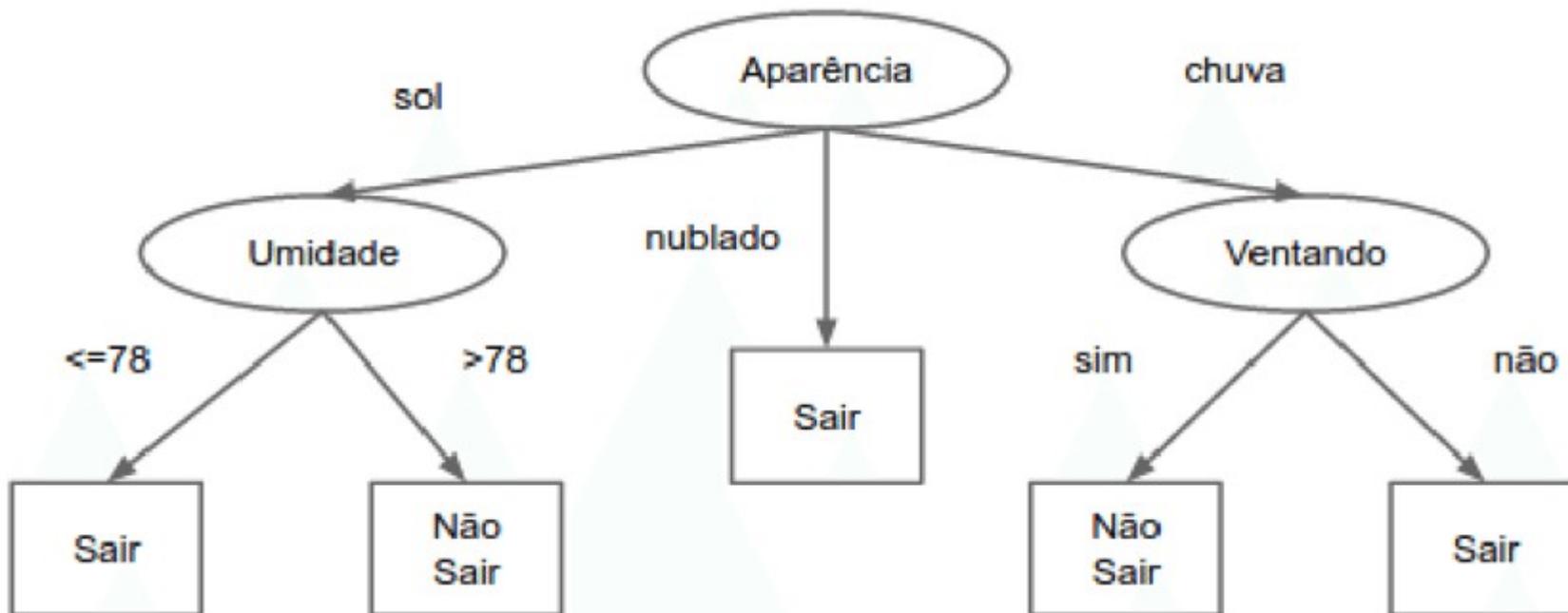
- Inteligência artificial

# Introdução

- Inteligência artificial
  - Árvores de decisão

# Introdução

- Inteligência artificial
  - Árvores de decisão



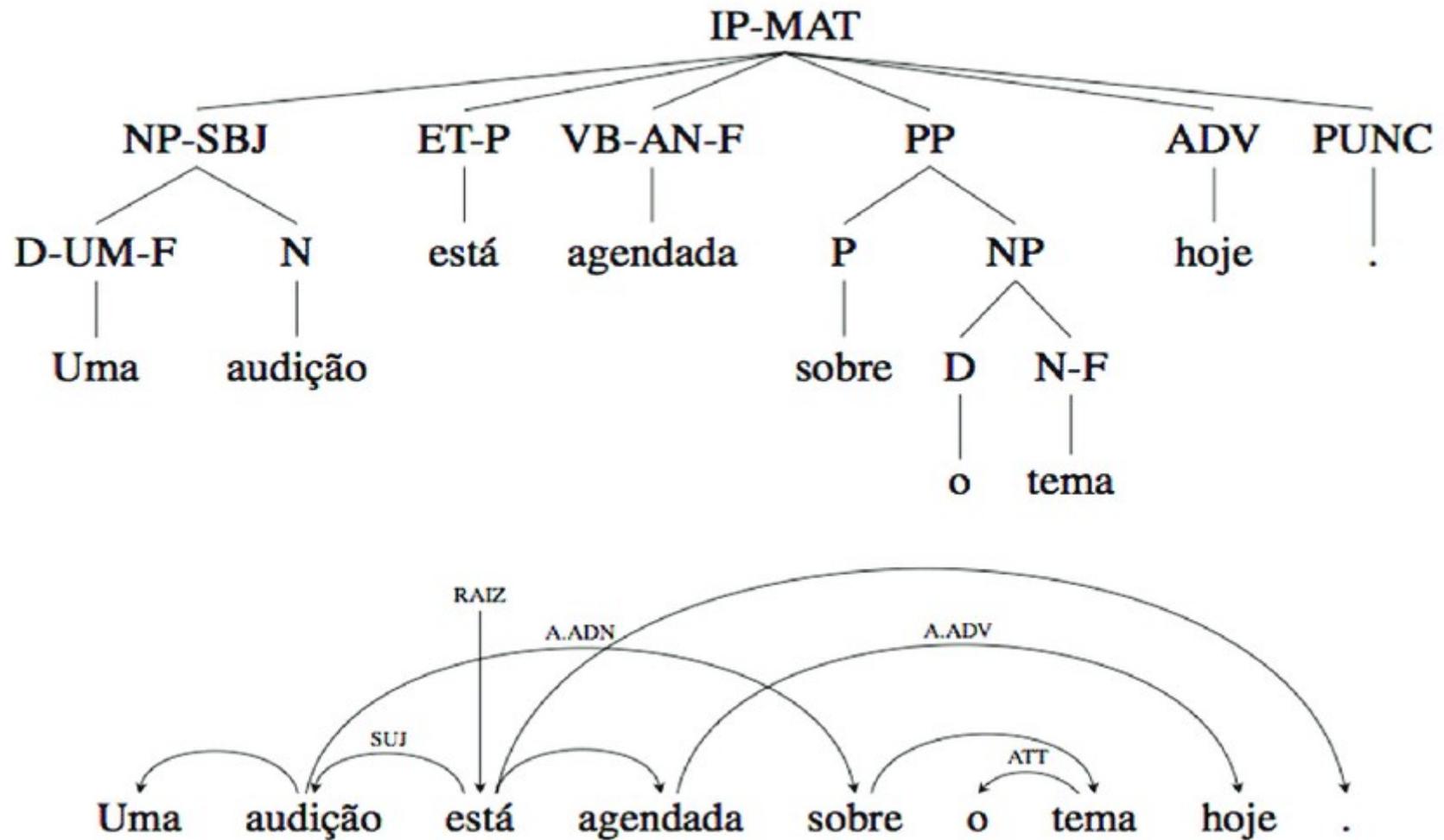
# Introdução

# Introdução

- Processamento de Linguagem Natural

# Introdução

- Processamento de Linguagem Natural



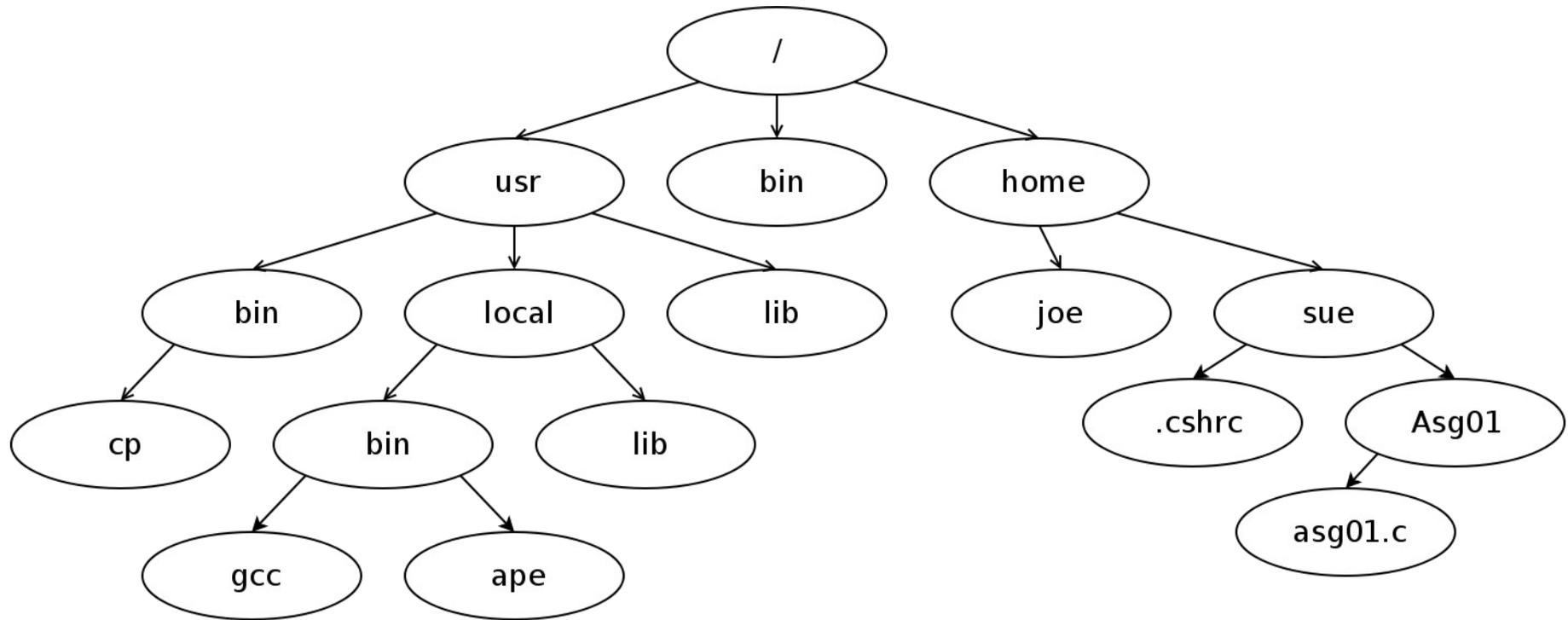
# Introdução

# Introdução

- A estrutura de diretórios e arquivos de um sistema de arquivos é um exemplo de árvore:

# Introdução

- A estrutura de diretórios e arquivos de um sistema de arquivos é um exemplo de árvore:



# Estrutura de Dados

# Estrutura de Dados

- Uma árvore é formada por:

# Estrutura de Dados

- Uma árvore é formada por:
  - Nós (*nodes*)

# Estrutura de Dados

- Uma árvore é formada por:
  - Nós (*nodes*)
    - Armazenam os dados.

# Estrutura de Dados

- Uma árvore é formada por:
  - Nós (*nodes*)
    - Armazenam os dados.
  - Vértices (*edges*)

# Estrutura de Dados

- Uma árvore é formada por:
  - Nós (*nodes*)
    - Armazenam os dados.
  - Vértices (*edges*)
    - Par de nós são ligados por vértices.

# Estrutura de Dados

# Estrutura de Dados

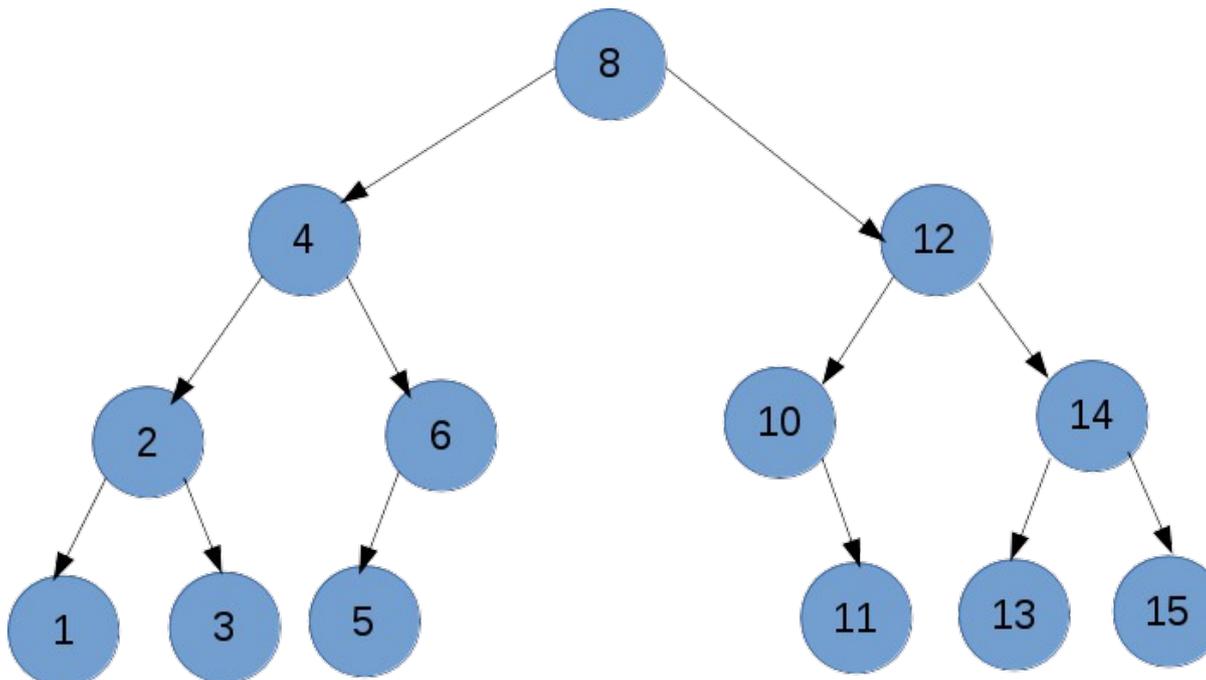
- Definição formal:

# Estrutura de Dados

- Definição formal:
  - Um conjunto de nós que pode ser vazio ou ter um nó raiz que está conectado por vértices a zero ou mais subárvores para formar uma estrutura hierárquica.

# Estrutura de Dados

- Definição formal:
  - Um conjunto de nós que pode ser vazio ou ter um nó raiz que está conectado por vértices a zero ou mais subárvores para formar uma estrutura hierárquica.



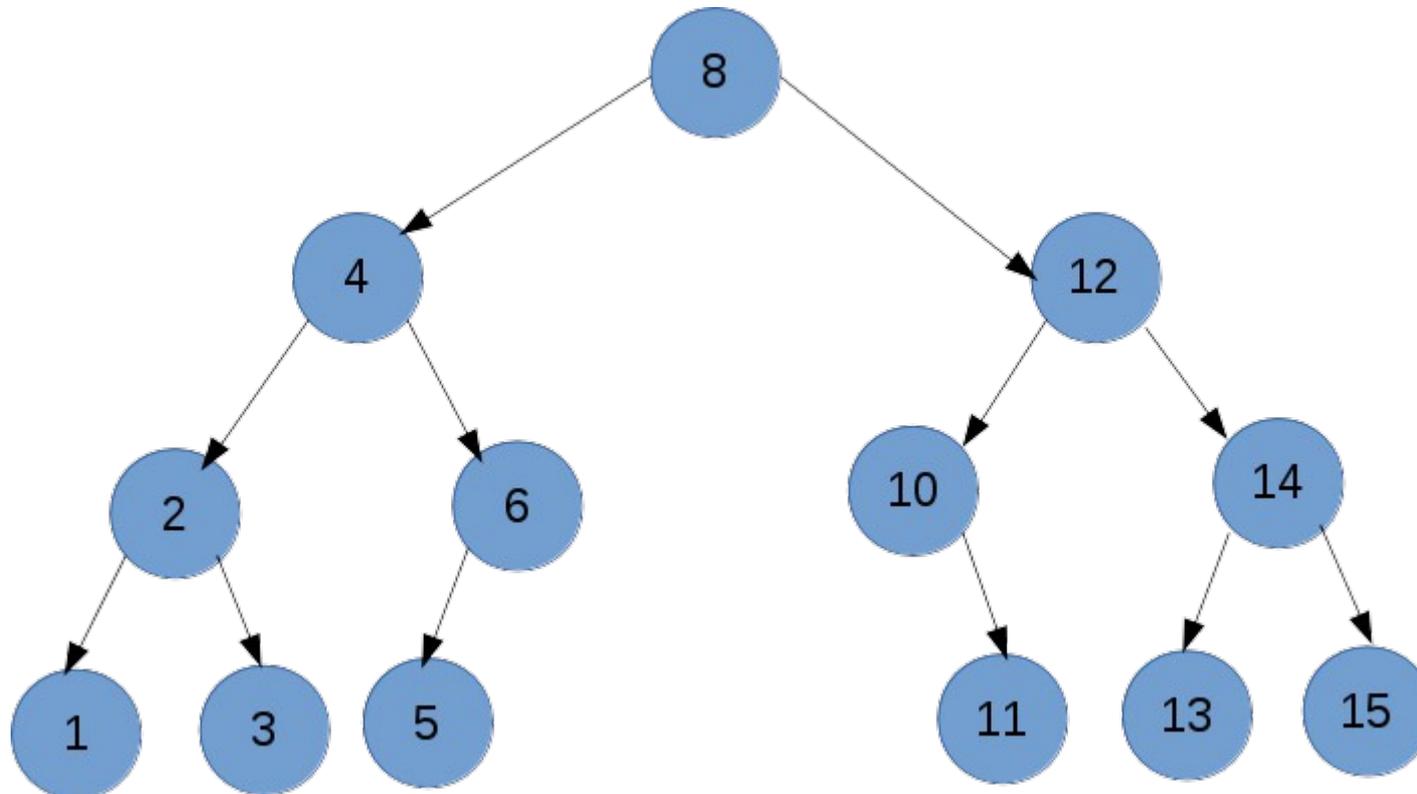
# Estrutura de Dados

# Estrutura de Dados

- Cada subárvore é também considerada como uma árvore.

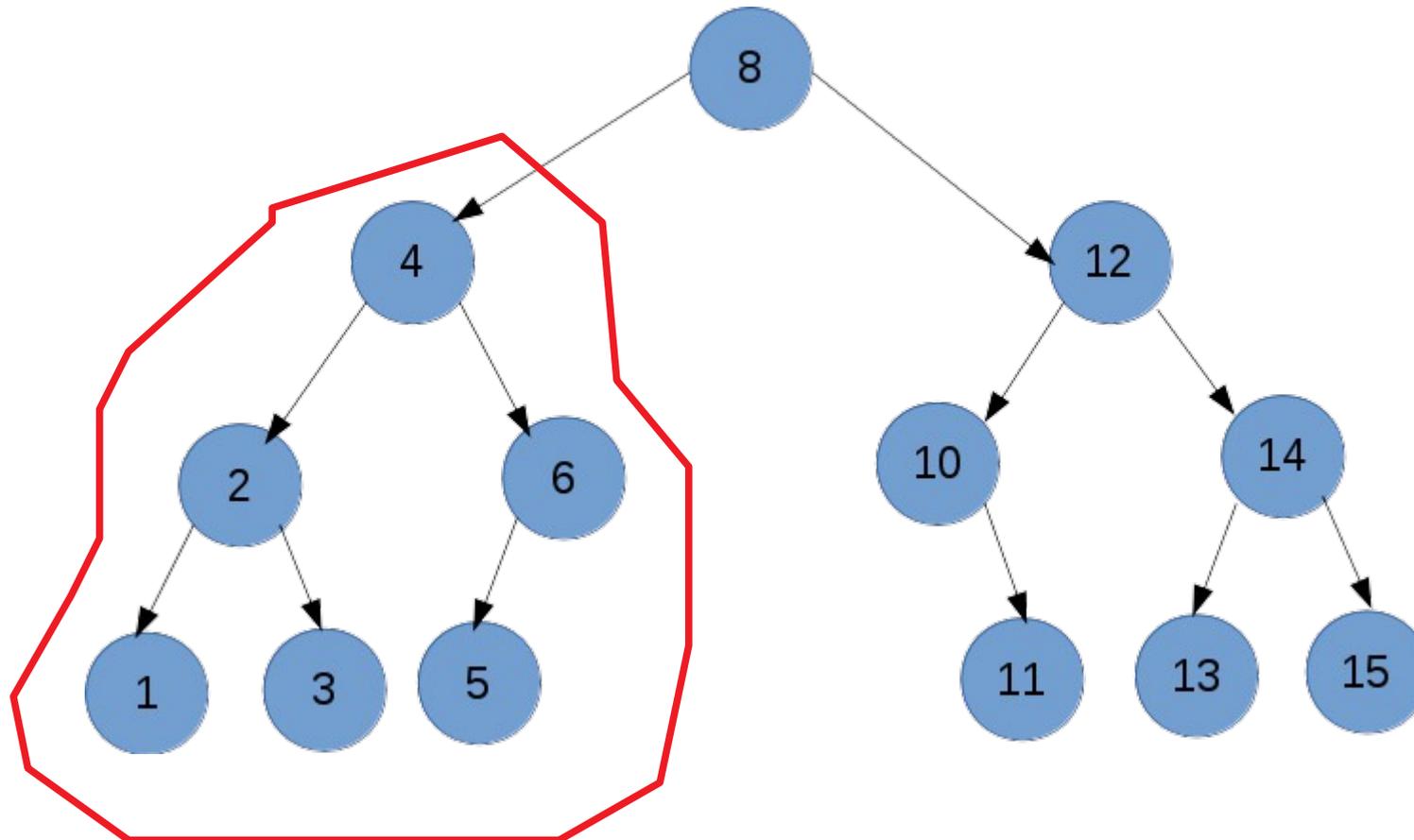
# Estrutura de Dados

- Cada subárvore é também considerada como uma árvore.



# Estrutura de Dados

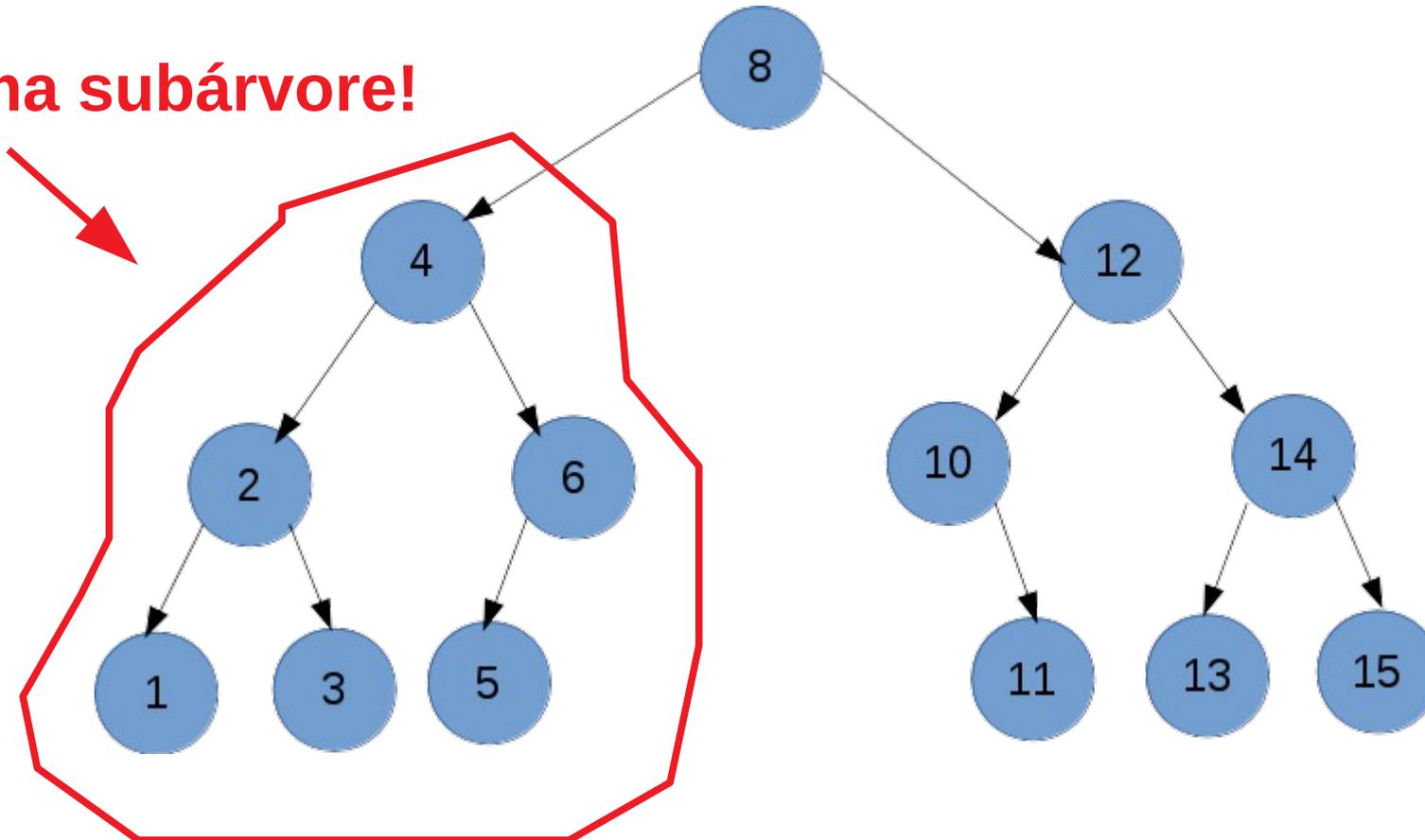
- Cada subárvore é também considerada como uma árvore.



# Estrutura de Dados

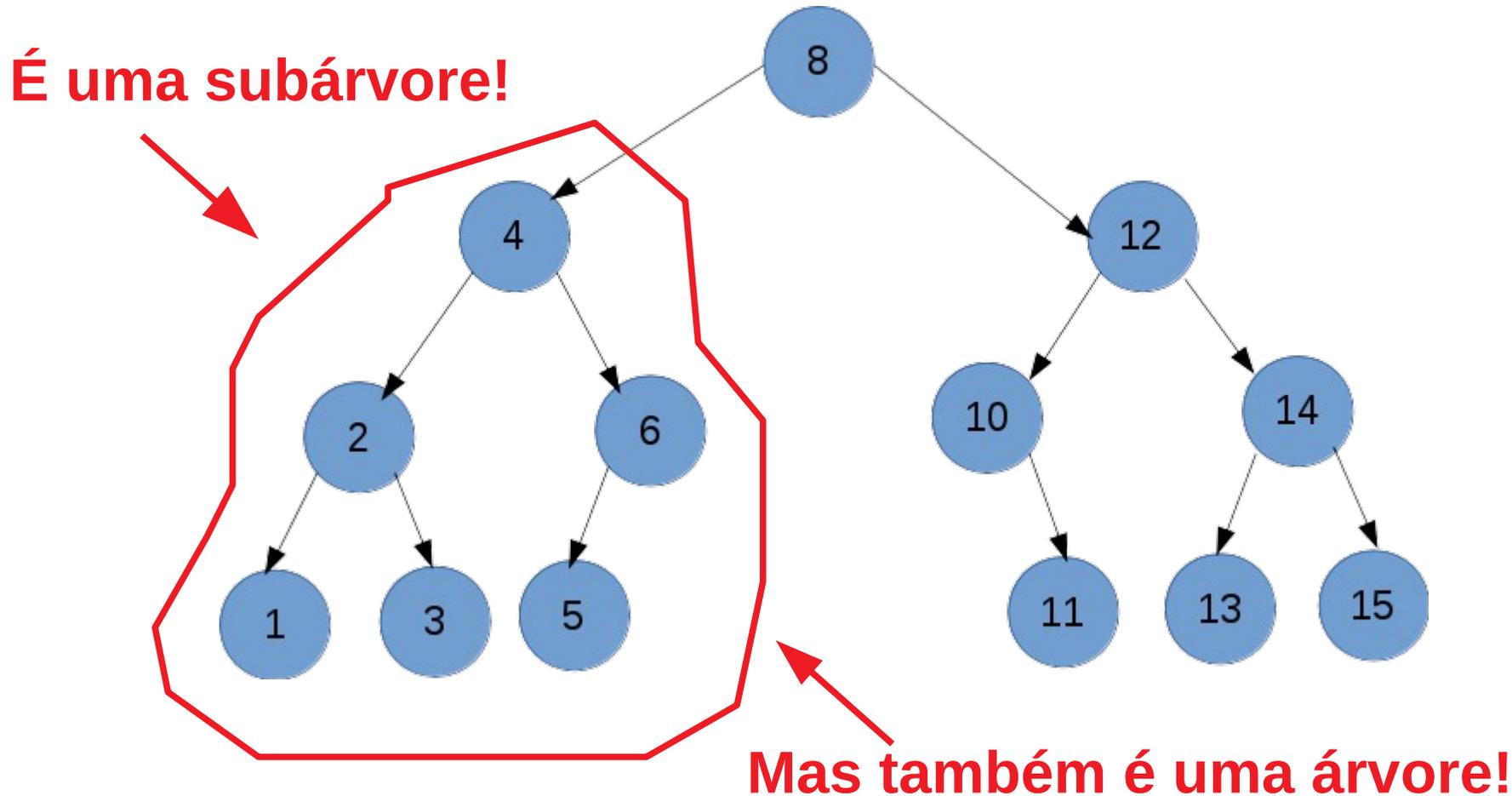
- Cada subárvore é também considerada como uma árvore.

É uma subárvore!



# Estrutura de Dados

- Cada subárvore é também considerada como uma árvore.



# Elementos da Estrutura Árvore

# Elementos da Estrutura Árvore

- Raiz (*root*)

# Elementos da Estrutura Árvore

- Raiz (*root*)
  - O nó no nível mais alto é chamado de **nó raiz** (*root node*).

# Elementos da Estrutura Árvore

- Raiz (*root*)
  - O nó no nível mais alto é chamado de **nó raiz** (*root node*).
  - Ele fornece o ponto de acesso para a estrutura.

# Elementos da Estrutura Árvore

- Raiz (*root*)
  - O nó no nível mais alto é chamado de **nó raiz** (*root node*).
  - Ele fornece o ponto de acesso para a estrutura.
  - É o único nó na árvore que não tem uma aresta de entrada.

# Elementos da Estrutura Árvore

- Raiz (*root*)
  - O nó no nível mais alto é chamado de **nó raiz** (*root node*).
  - Ele fornece o ponto de acesso para a estrutura.
  - É o único nó na árvore que não tem uma aresta de entrada.
  - Toda árvore não-vazia deve conter um nó raiz.

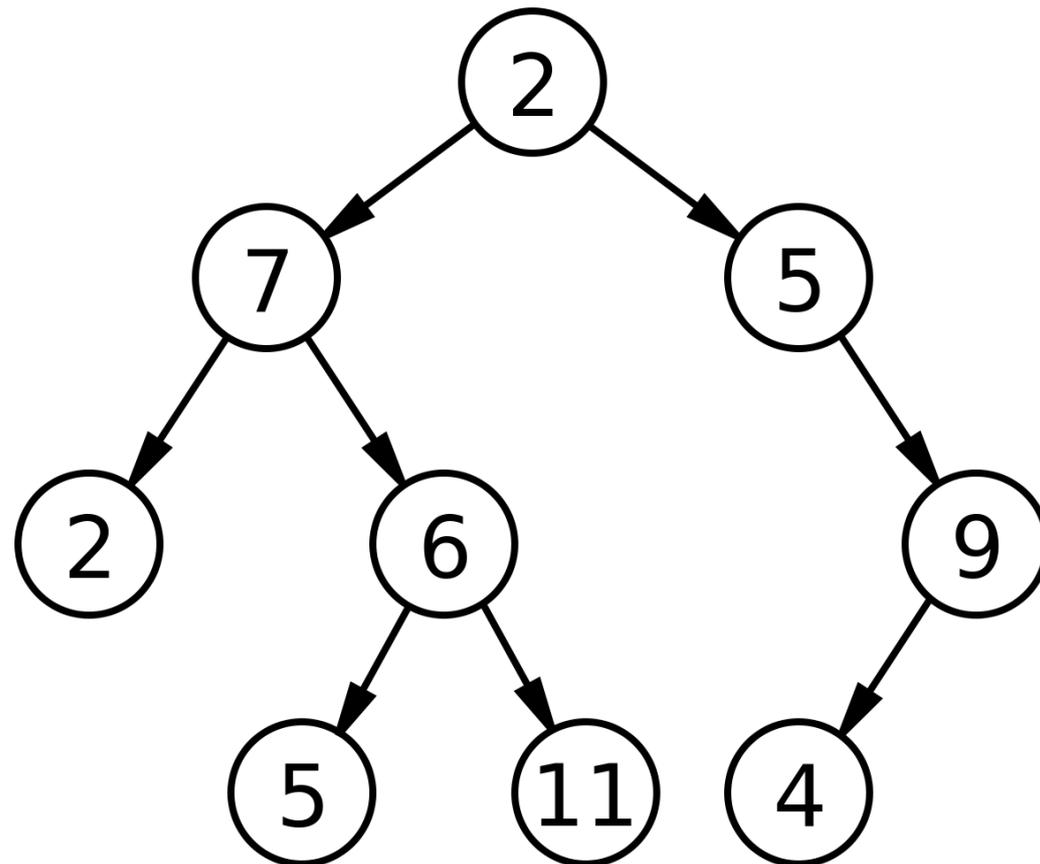
# Elementos da Estrutura Árvore

# Elementos da Estrutura Árvore

- Raiz (*root*)

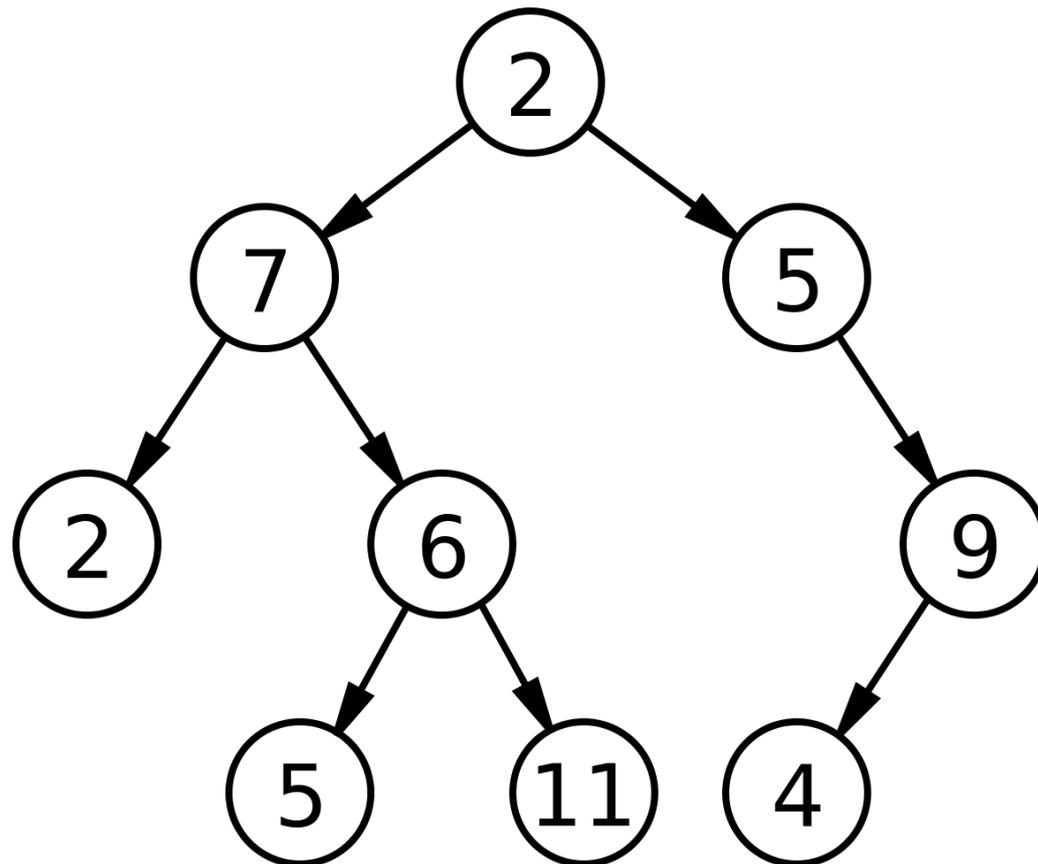
# Elementos da Estrutura Árvore

- Raiz (*root*)



# Elementos da Estrutura Árvore

- Raiz (*root*)
  - 2 é o nó raiz.



# Elementos da Estrutura Árvore

# Elementos da Estrutura Árvore

- Caminho (*path*)

# Elementos da Estrutura Árvore

- Caminho (*path*)
  - Os demais nós da árvore são acessados através dos vértices.

# Elementos da Estrutura Árvore

- Caminho (*path*)
  - Os demais nós da árvore são acessados através dos vértices.
  - Os nós encontrados ao seguir os vértices a partir do nó raiz formam o caminho (*path*).

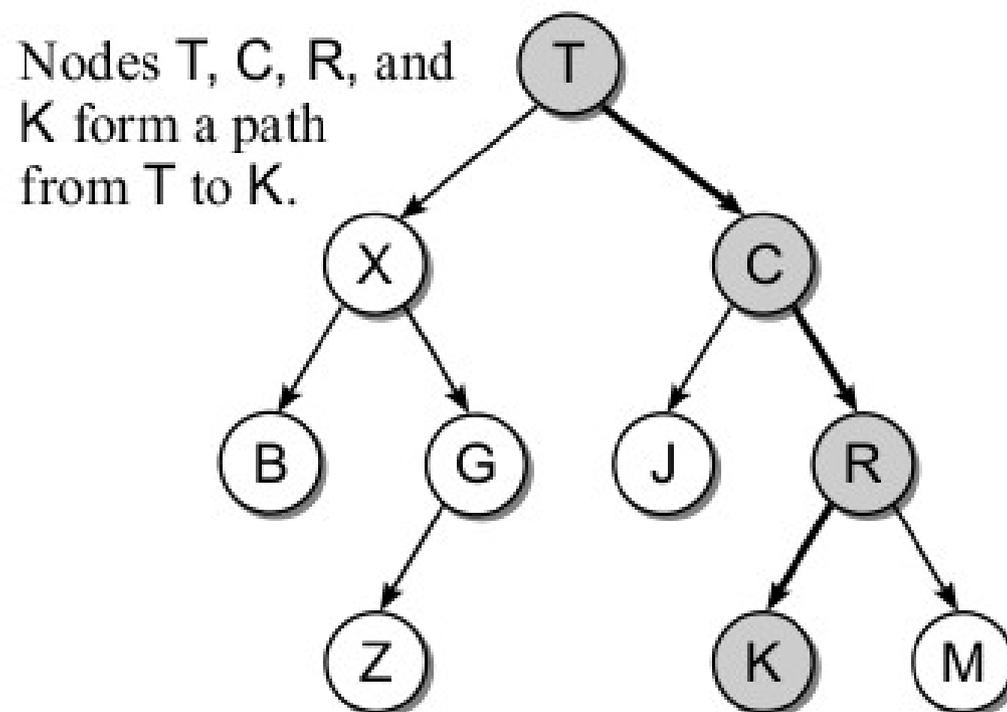
# Elementos da Estrutura Árvore

# Elementos da Estrutura Árvore

- Caminho (*path*)

# Elementos da Estrutura Árvore

- Caminho (*path*)

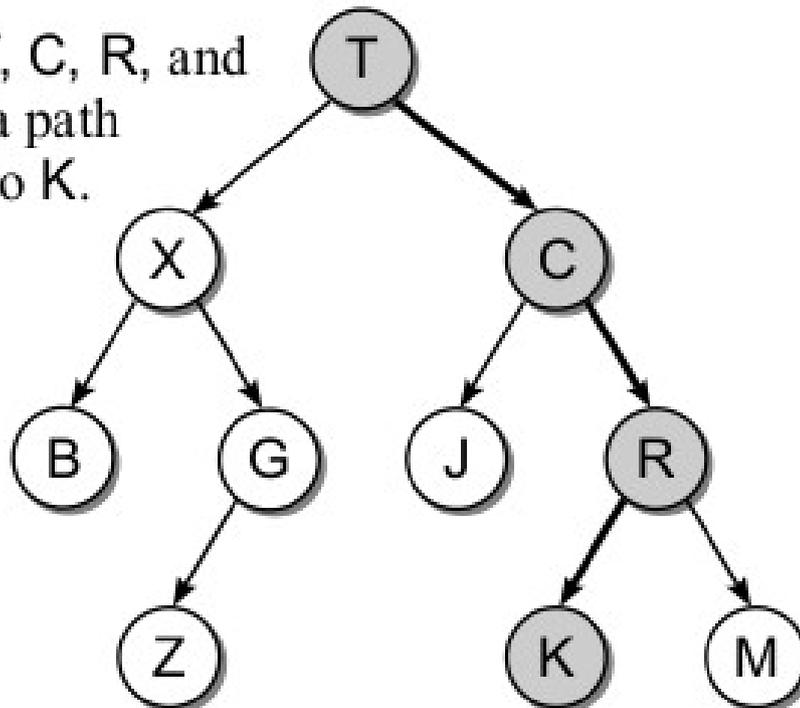


# Elementos da Estrutura Árvore

- Caminho (*path*)

- Caminho do nó T até K:  $T \rightarrow C \rightarrow R \rightarrow K$

Nodes T, C, R, and K form a path from T to K.



# Elementos da Estrutura Árvore

# Elementos da Estrutura Árvore

- Pais (*parent*)

# Elementos da Estrutura Árvore

- Pais (*parent*)
  - Cada nó, com exceção da raiz, tem um nó pai.

# Elementos da Estrutura Árvore

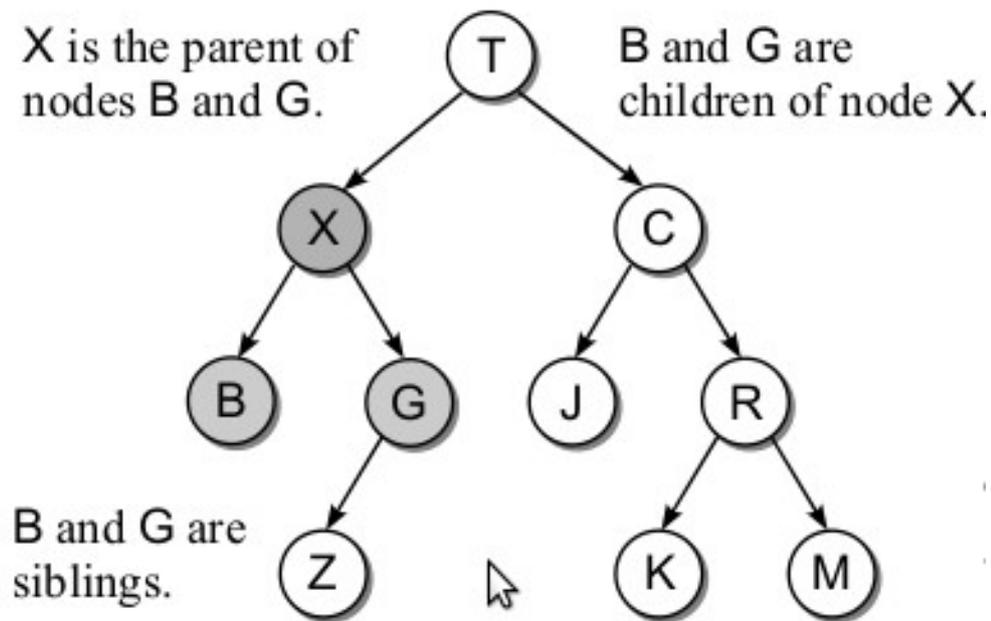
- Pais (*parent*)
  - Cada nó, com exceção da raiz, tem um nó pai.
  - Um nó pode ter apenas um nó pai.

# Elementos da Estrutura Árvore

- Pais (*parent*)
  - Cada nó, com exceção da raiz, tem um nó pai.
  - Um nó pode ter apenas um nó pai.
  - Exemplo:

# Elementos da Estrutura Árvore

- Pais (*parent*)
  - Cada nó, com exceção da raiz, tem um nó pai.
  - Um nó pode ter apenas um nó pai.
  - Exemplo:



# Elementos da Estrutura Árvore

# Elementos da Estrutura Árvore

- Filho (*children*)

# Elementos da Estrutura Árvore

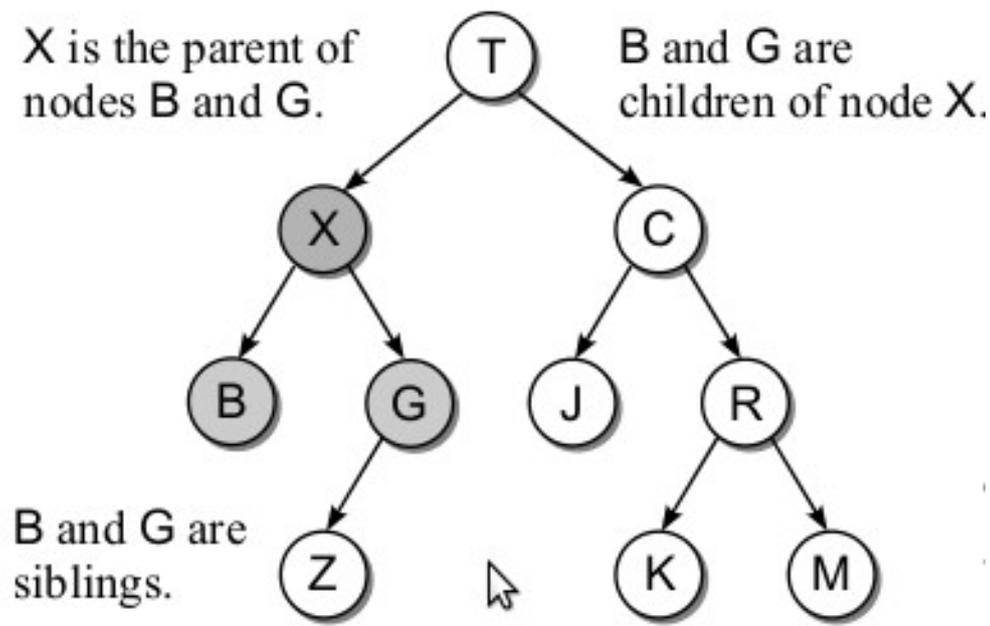
- Filho (*children*)
  - Cada nó pode ter zero, um ou mais filhos.

# Elementos da Estrutura Árvore

- Filho (*children*)
  - Cada nó pode ter zero, um ou mais filhos.
  - Exemplo:

# Elementos da Estrutura Árvore

- Filho (*children*)
  - Cada nó pode ter zero, um ou mais filhos.
  - Exemplo:



# Elementos da Estrutura Árvore

# Elementos da Estrutura Árvore

- Nós

# Elementos da Estrutura Árvore

- Nós
  - Nó que tem pelo menos 1 filho é chamado de **nó interior**.

# Elementos da Estrutura Árvore

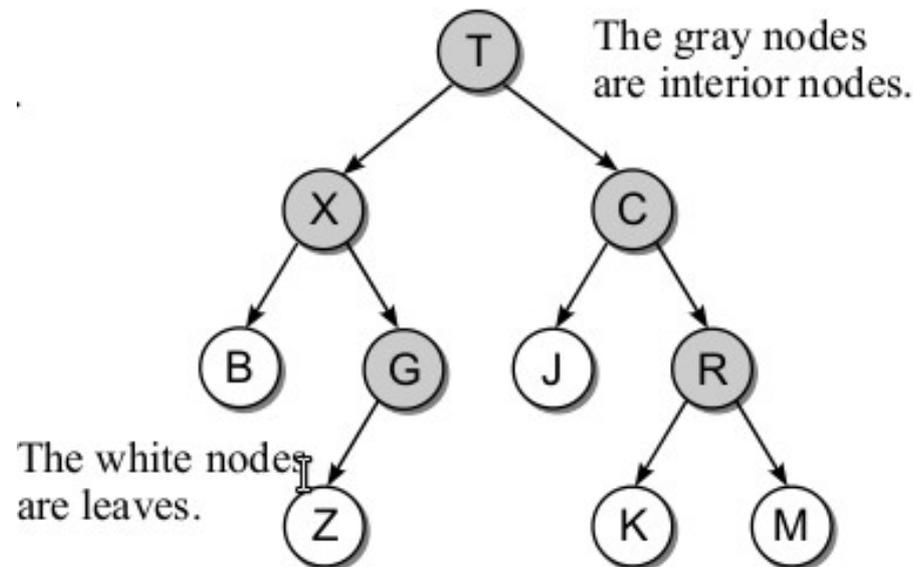
- Nós
  - Nó que tem pelo menos 1 filho é chamado de **nó interior**.
  - Nó que não tem filho é chamado de **nó folha** (*leaf*).

# Elementos da Estrutura Árvore

- Nós
  - Nó que tem pelo menos 1 filho é chamado de **nó interior**.
  - Nó que não tem filho é chamado de **nó folha** (*leaf*).
  - Exemplo:

# Elementos da Estrutura Árvore

- Nós
  - Nó que tem pelo menos 1 filho é chamado de **nó interior**.
  - Nó que não tem filho é chamado de **nó folha** (*leaf*).
  - Exemplo:



# Elementos da Estrutura Árvore

# Elementos da Estrutura Árvore

- Subárvores

# Elementos da Estrutura Árvore

- Subárvores
  - Por definição, uma árvore é uma estrutura recursiva.

# Elementos da Estrutura Árvore

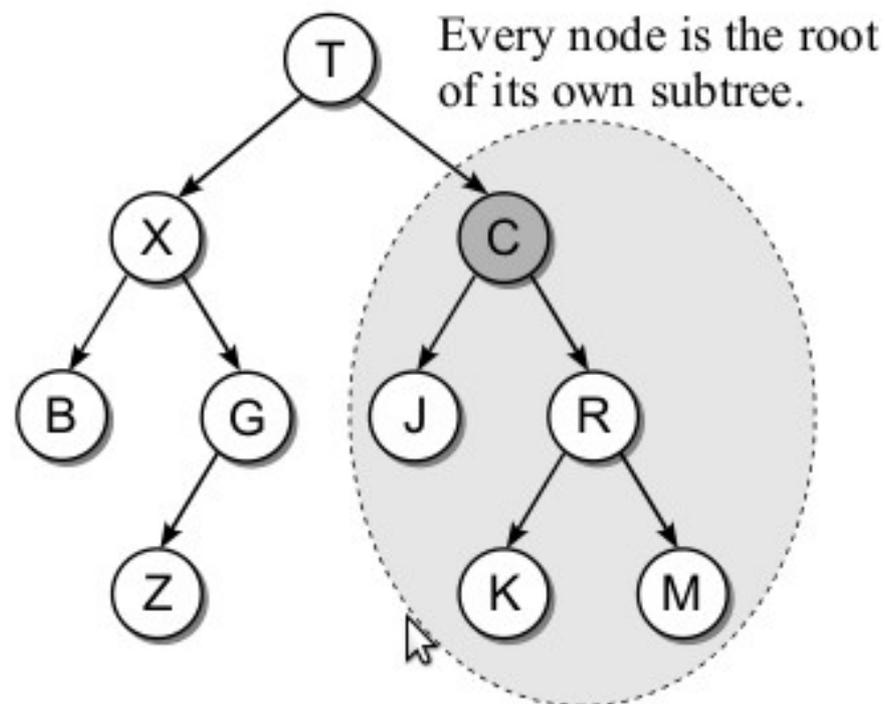
- Subárvores
  - Por definição, uma árvore é uma estrutura recursiva.
  - Todo nó pode ser raiz da sua própria subárvore.

# Elementos da Estrutura Árvore

- Subárvores
  - Por definição, uma árvore é uma estrutura recursiva.
  - Todo nó pode ser raiz da sua própria subárvore.
  - Exemplo:

# Elementos da Estrutura Árvore

- Subárvores
  - Por definição, uma árvore é uma estrutura recursiva.
  - Todo nó pode ser raiz da sua própria subárvore.
  - Exemplo:



# Elementos da Estrutura Árvore

# Elementos da Estrutura Árvore

- Grau

# Elementos da Estrutura Árvore

- Grau
  - Representa o número de subárvores de um nó.

# Elementos da Estrutura Árvore

- Grau
  - Representa o número de subárvores de um nó.
- Grau de uma árvore

# Elementos da Estrutura Árvore

- Grau
  - Representa o número de subárvores de um nó.
- Grau de uma árvore
  - É definido como sendo igual ao máximo dos graus de todos os seus nós.

# Árvores Binárias

# Árvores Binárias

# Árvores Binárias

- As árvores podem aparecer em vários formatos.

# Árvores Binárias

- As árvores podem aparecer em vários formatos.
- Elas podem variar o número de filhos permitidos por nó.

# Árvores Binárias

- As árvores podem aparecer em vários formatos.
- Elas podem variar o número de filhos permitidos por nó.
- Uma árvore bastante usada é a **árvore binária**, em que cada nó pode ter, no máximo, dois filhos por nó.

# Árvores Binárias

- As árvores podem aparecer em vários formatos.
- Elas podem variar o número de filhos permitidos por nó.
- Uma árvore bastante usada é a **árvore binária**, em que cada nó pode ter, no máximo, dois filhos por nó.
- Os filhos podem ficar à esquerda ou à direita.

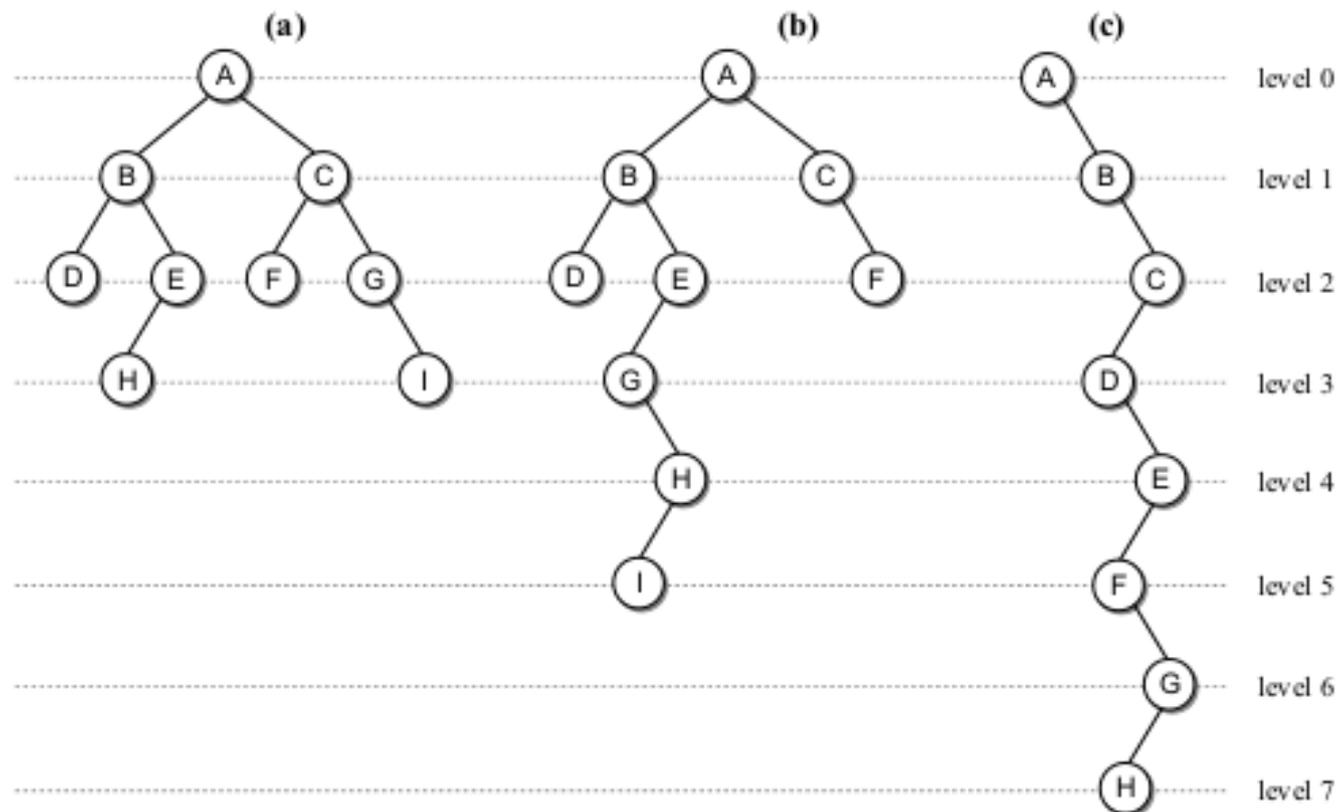
# Árvores Binárias

- Propriedades
  - Árvores binárias podem aparecer em diferentes formatos e tamanhos.

# Árvores Binárias

- Propriedades

- Exemplo de três formatos distintos para uma árvore binária com 9 nós:



# Árvores Binárias

# Árvores Binárias

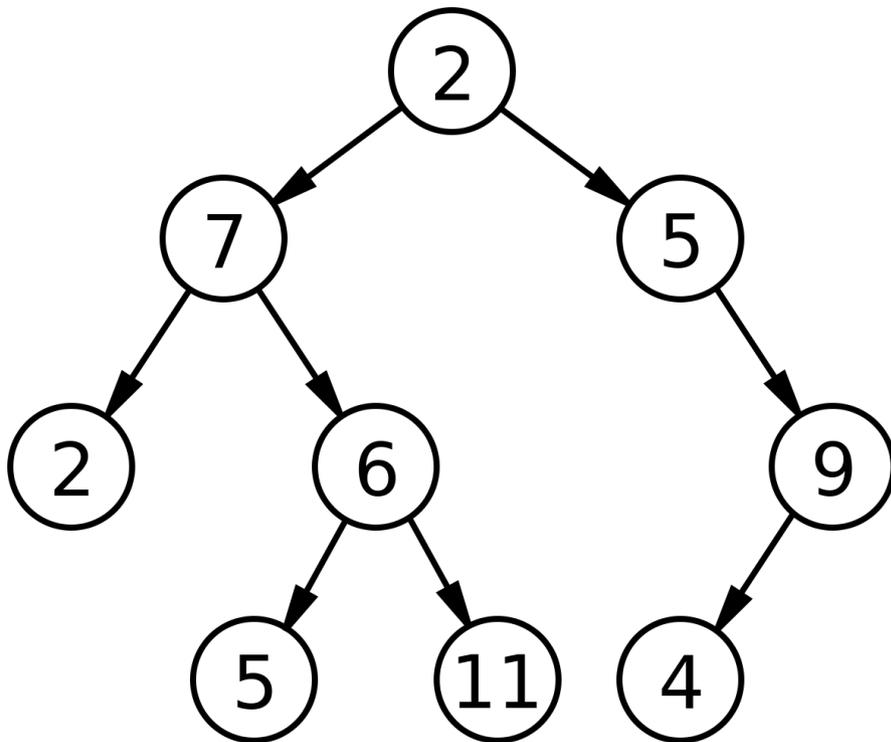
- Tamanho da árvore

# Árvores Binárias

- Tamanho da árvore
  - Os nós em uma árvore binária são organizados em níveis (*levels*), onde o nó raiz está no nível 0, os seus filhos no nível 1, etc.

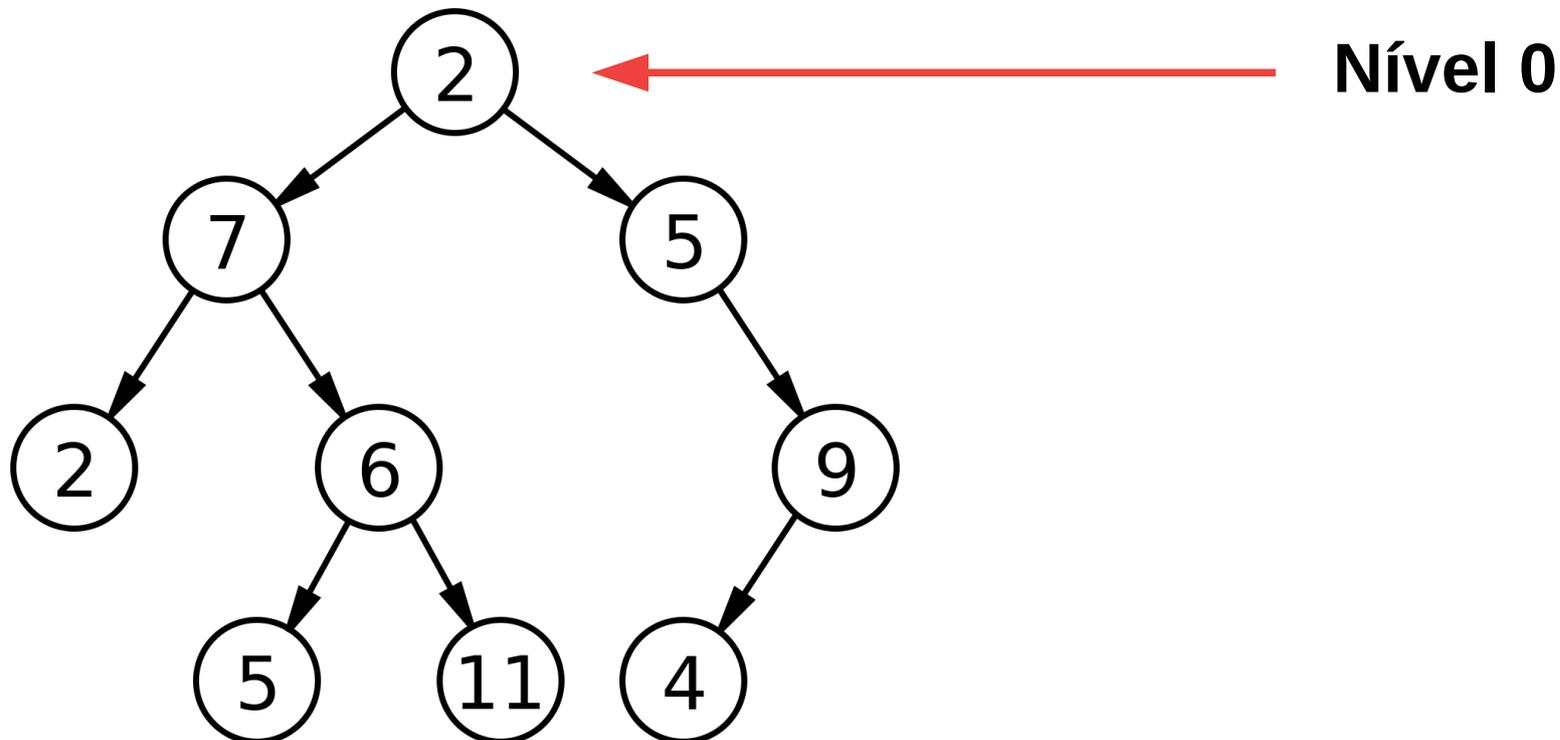
# Árvores Binárias

- Tamanho da árvore
  - Os nós em uma árvore binária são organizados em níveis (*levels*), onde o nó raiz está no nível 0, os seus filhos no nível 1, etc.



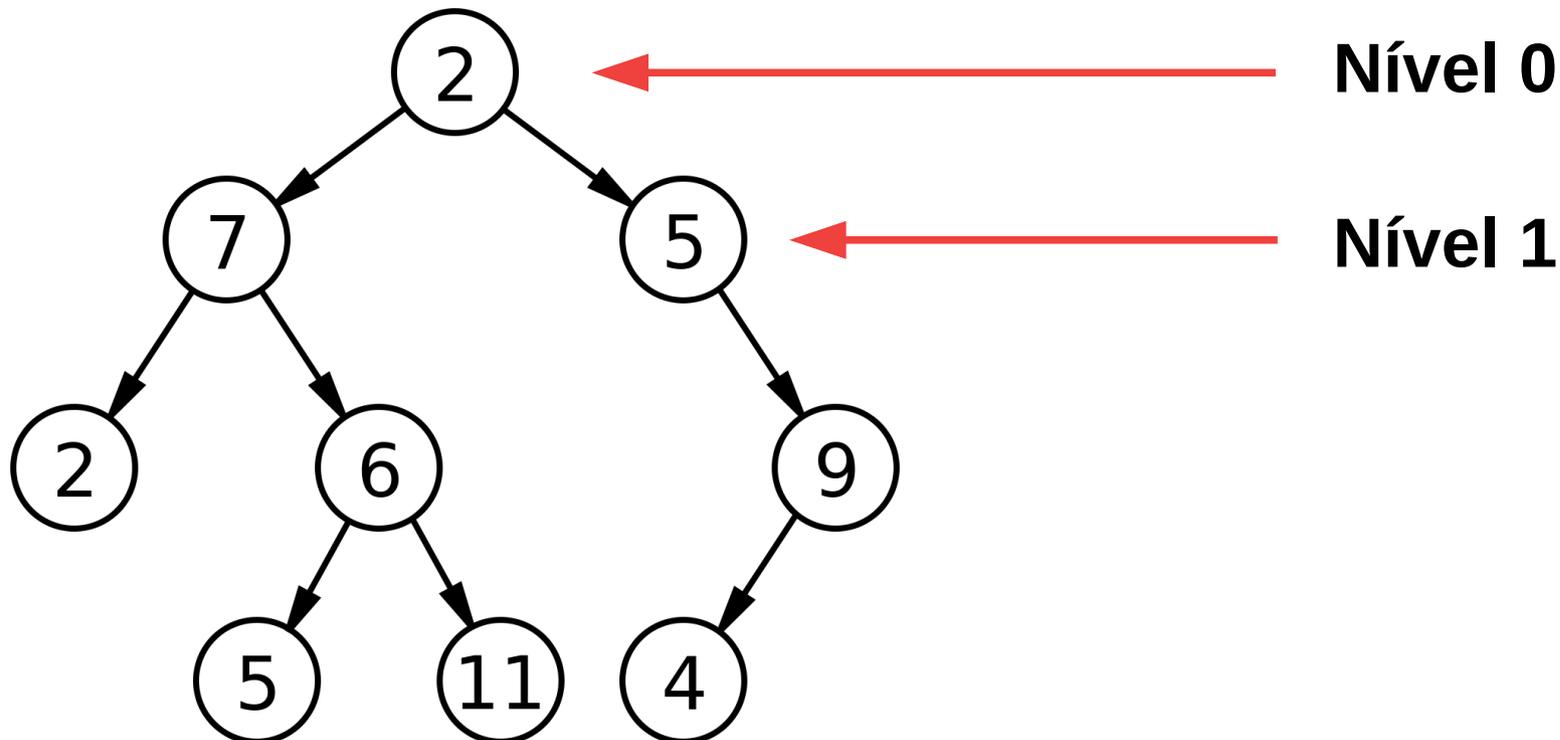
# Árvores Binárias

- Tamanho da árvore
  - Os nós em uma árvore binária são organizados em níveis (*levels*), onde o nó raiz está no nível 0, os seus filhos no nível 1, etc.



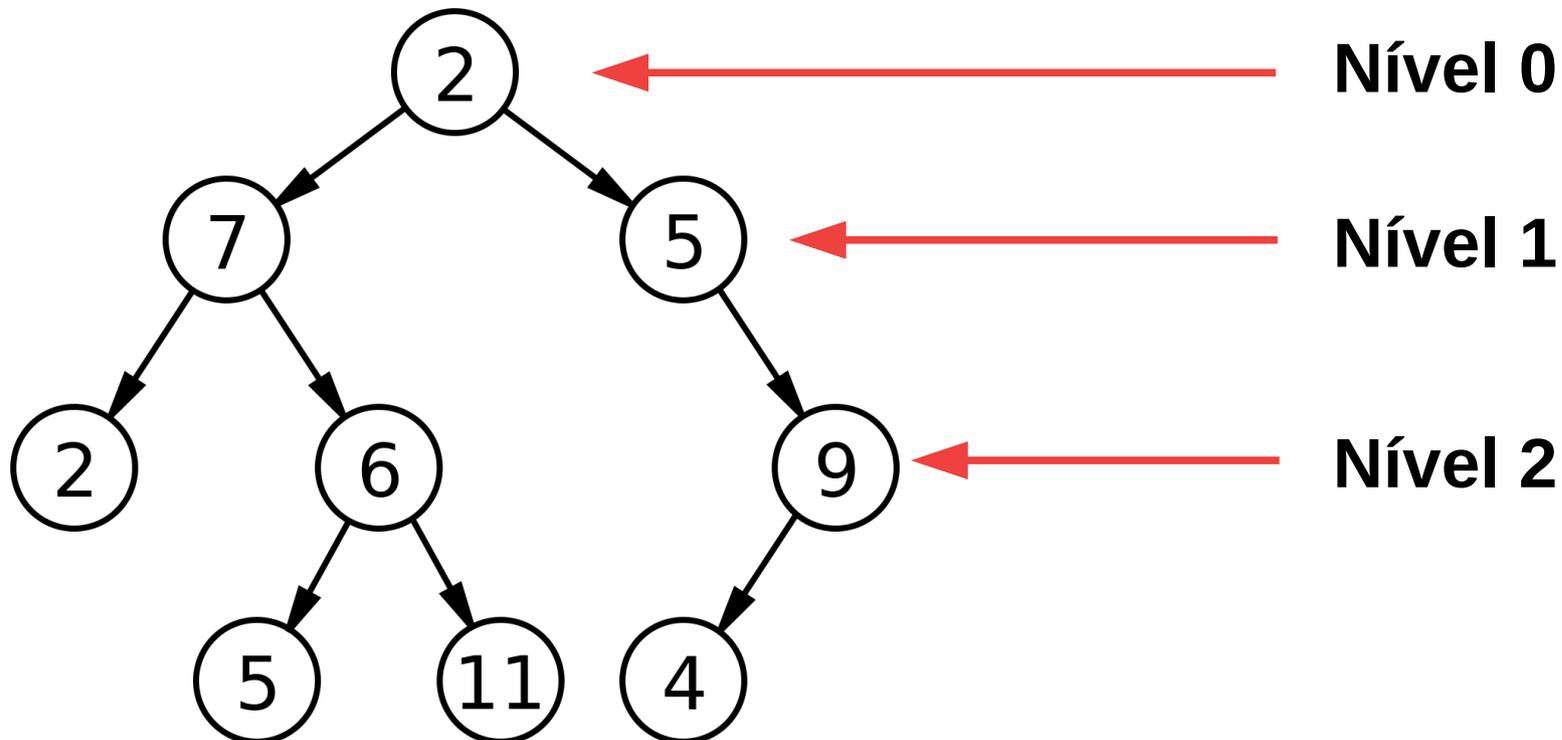
# Árvores Binárias

- Tamanho da árvore
  - Os nós em uma árvore binária são organizados em níveis (*levels*), onde o nó raiz está no nível 0, os seus filhos no nível 1, etc.



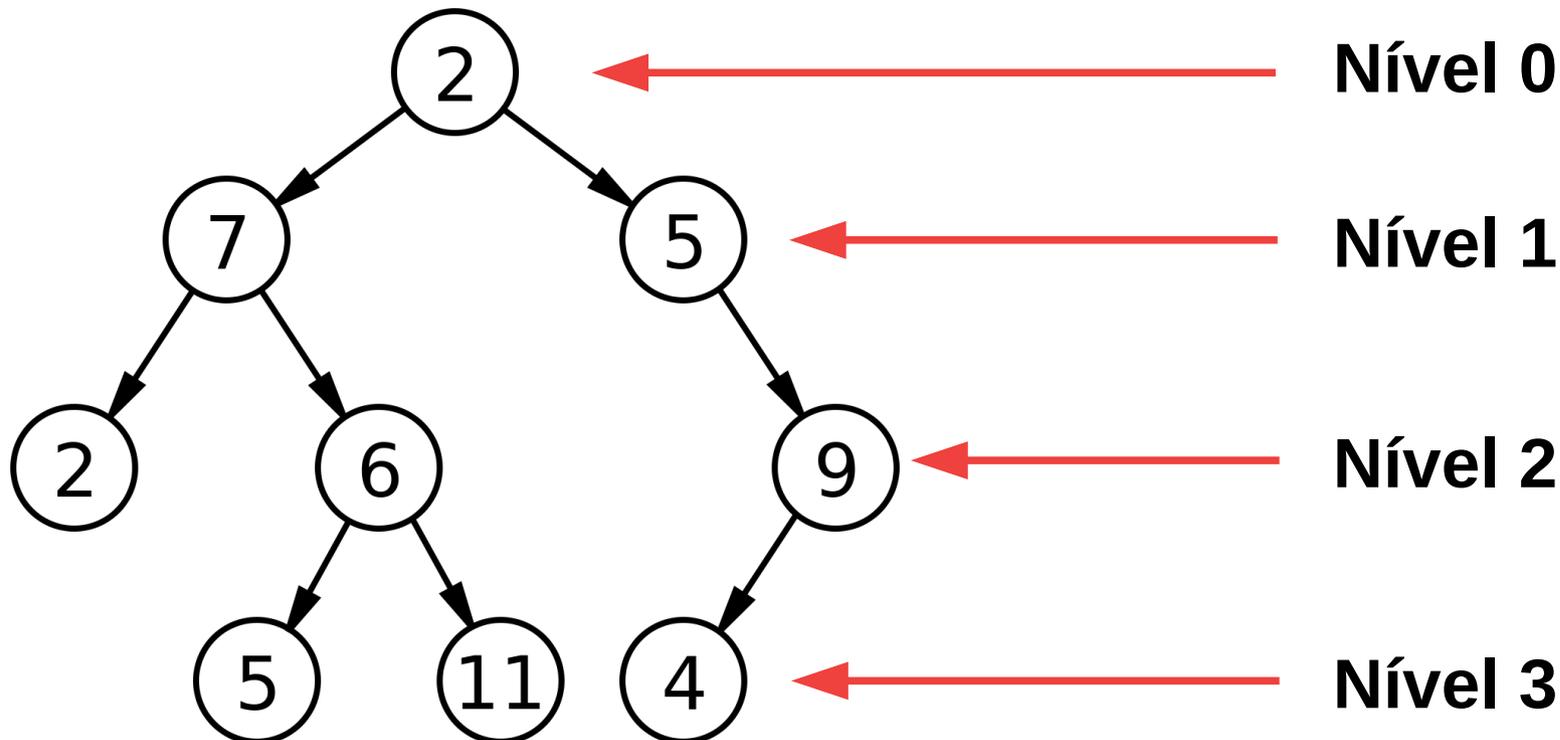
# Árvores Binárias

- Tamanho da árvore
  - Os nós em uma árvore binária são organizados em níveis (*levels*), onde o nó raiz está no nível 0, os seus filhos no nível 1, etc.



# Árvores Binárias

- Tamanho da árvore
  - Os nós em uma árvore binária são organizados em níveis (*levels*), onde o nó raiz está no nível 0, os seus filhos no nível 1, etc.



# Árvores Binárias

# Árvores Binárias

- Profundidade (*depth*)

# Árvores Binárias

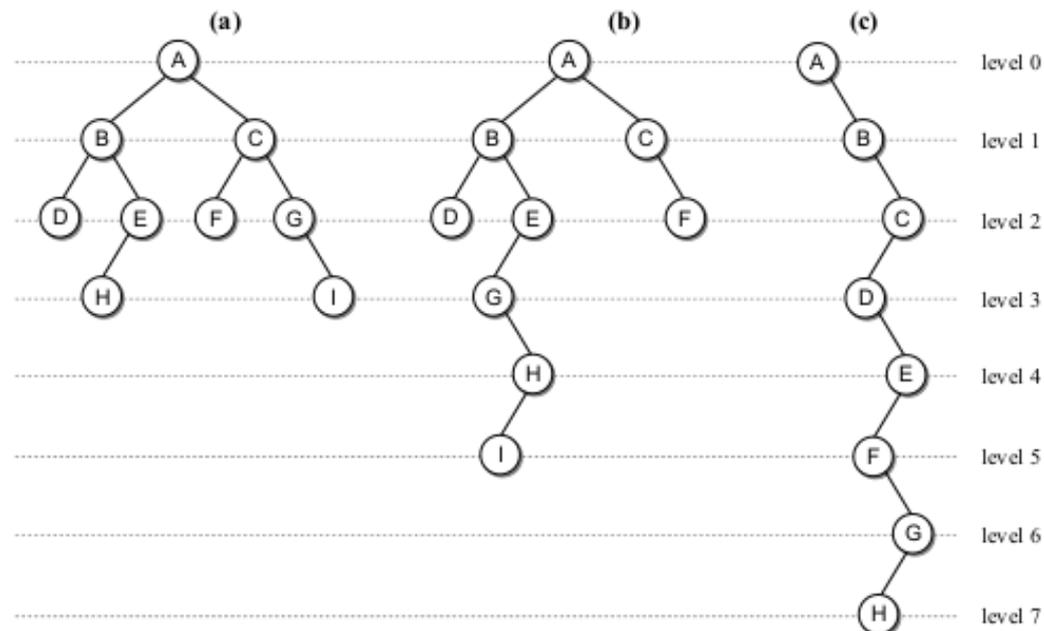
- Profundidade (*depth*)
  - Profundidade é a distância de um nó até a raiz.

# Árvores Binárias

- Profundidade (*depth*)
  - Profundidade é a distância de um nó até a raiz.
  - A profundidade do nó G:

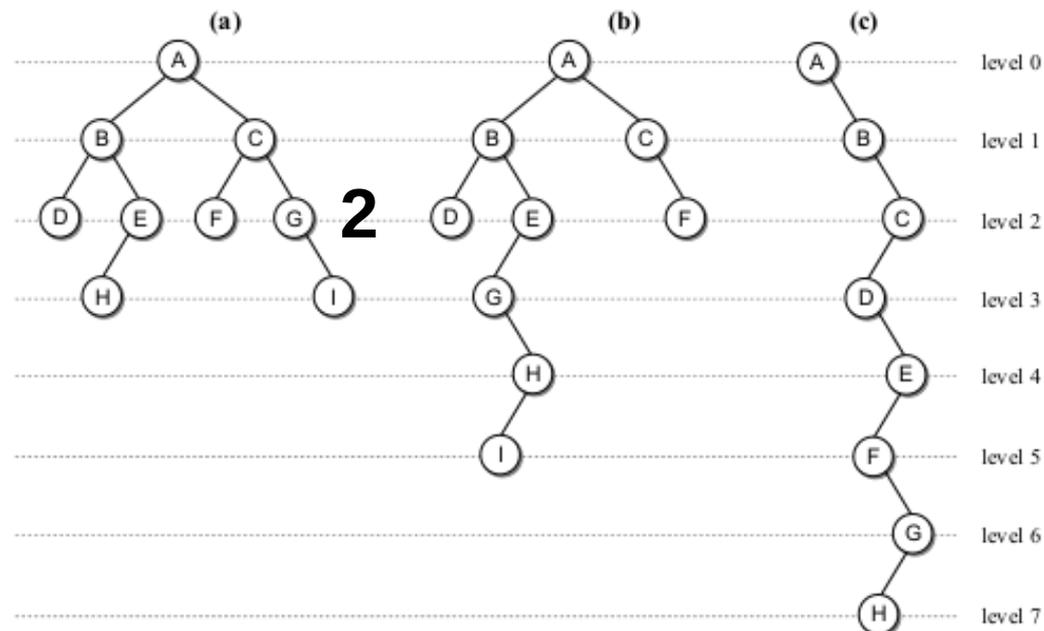
# Árvores Binárias

- Profundidade (*depth*)
  - Profundidade é a distância de um nó até a raiz.
  - A profundidade do nó G:



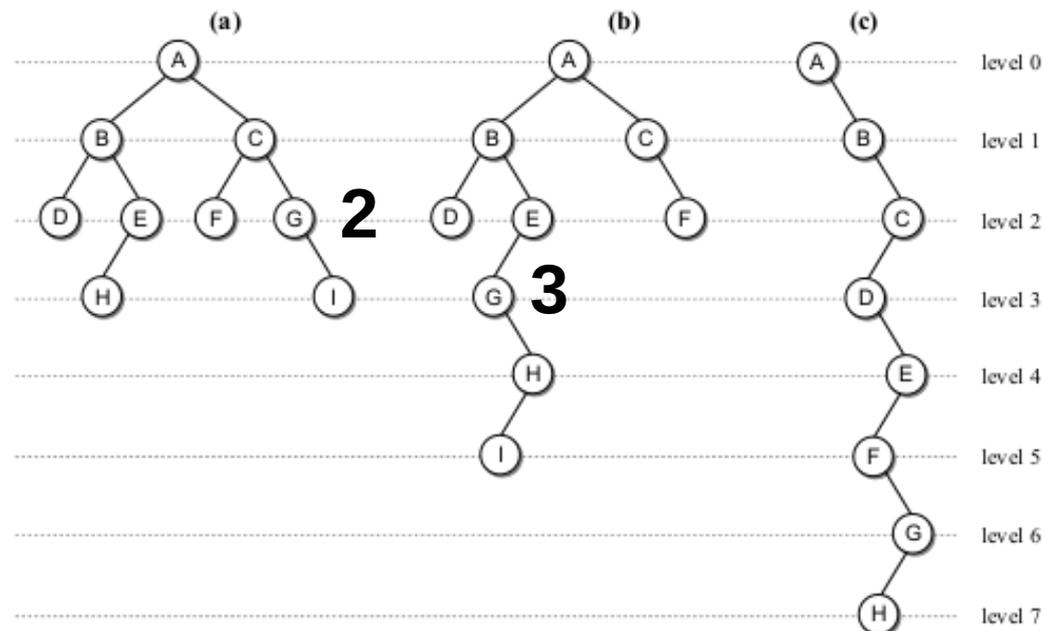
# Árvores Binárias

- Profundidade (*depth*)
  - Profundidade é a distância de um nó até a raiz.
  - A profundidade do nó G:



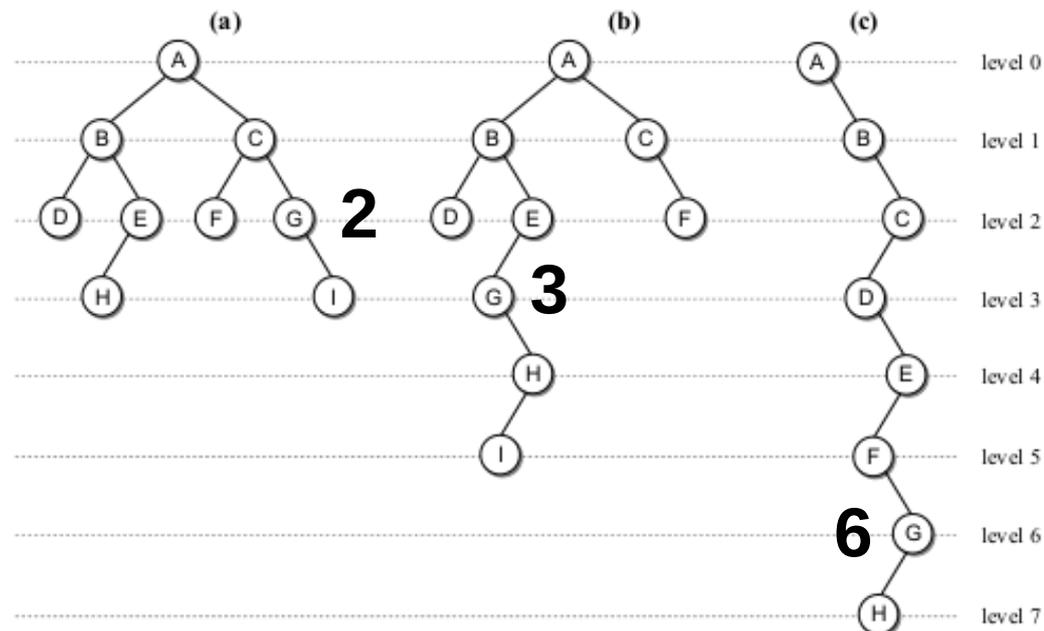
# Árvores Binárias

- Profundidade (*depth*)
  - Profundidade é a distância de um nó até a raiz.
  - A profundidade do nó G:



# Árvores Binárias

- Profundidade (*depth*)
  - Profundidade é a distância de um nó até a raiz.
  - A profundidade do nó G:



# Árvores Binárias

# Árvores Binárias

- Altura (*height*)

# Árvores Binárias

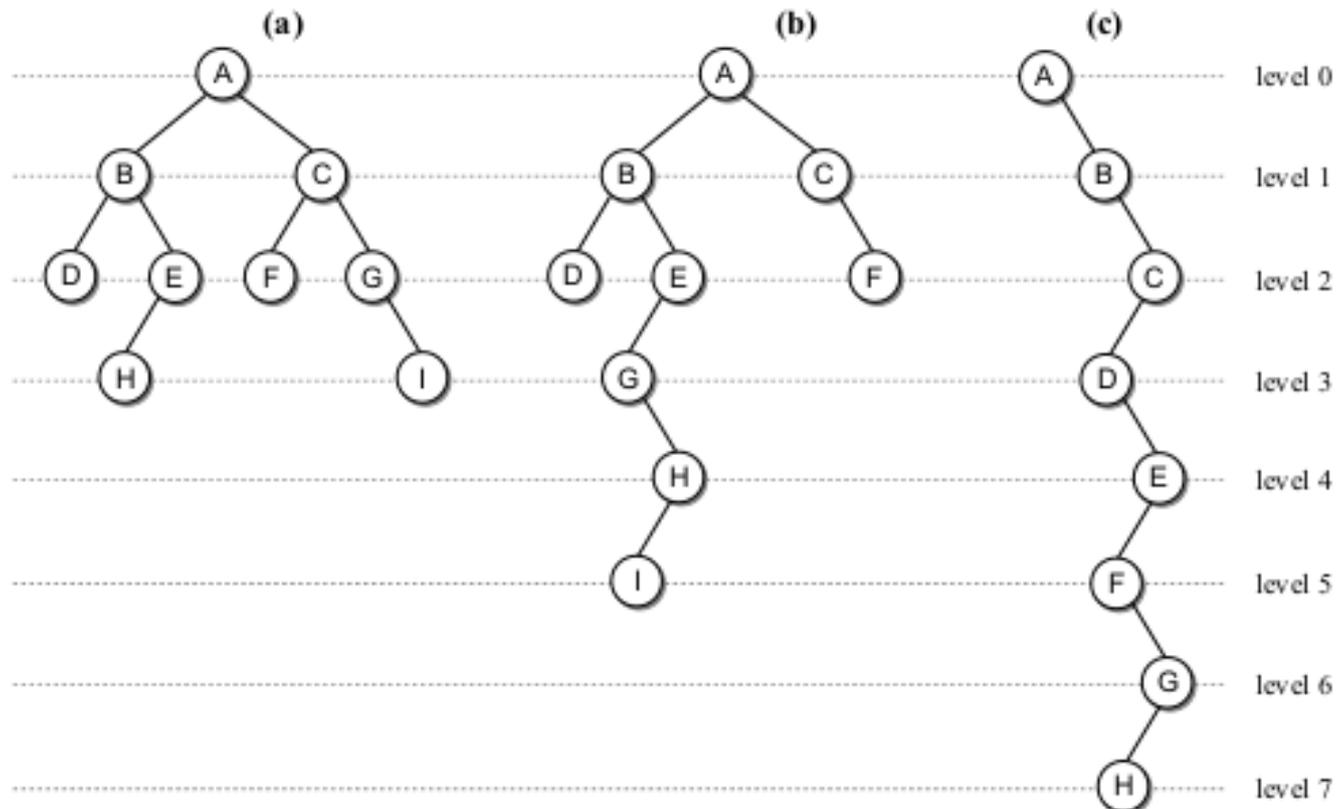
- Altura (*height*)
  - É o número de níveis de uma árvore.

# Árvores Binárias

- Altura (*height*)
  - É o número de níveis de uma árvore.
  - Exemplo:

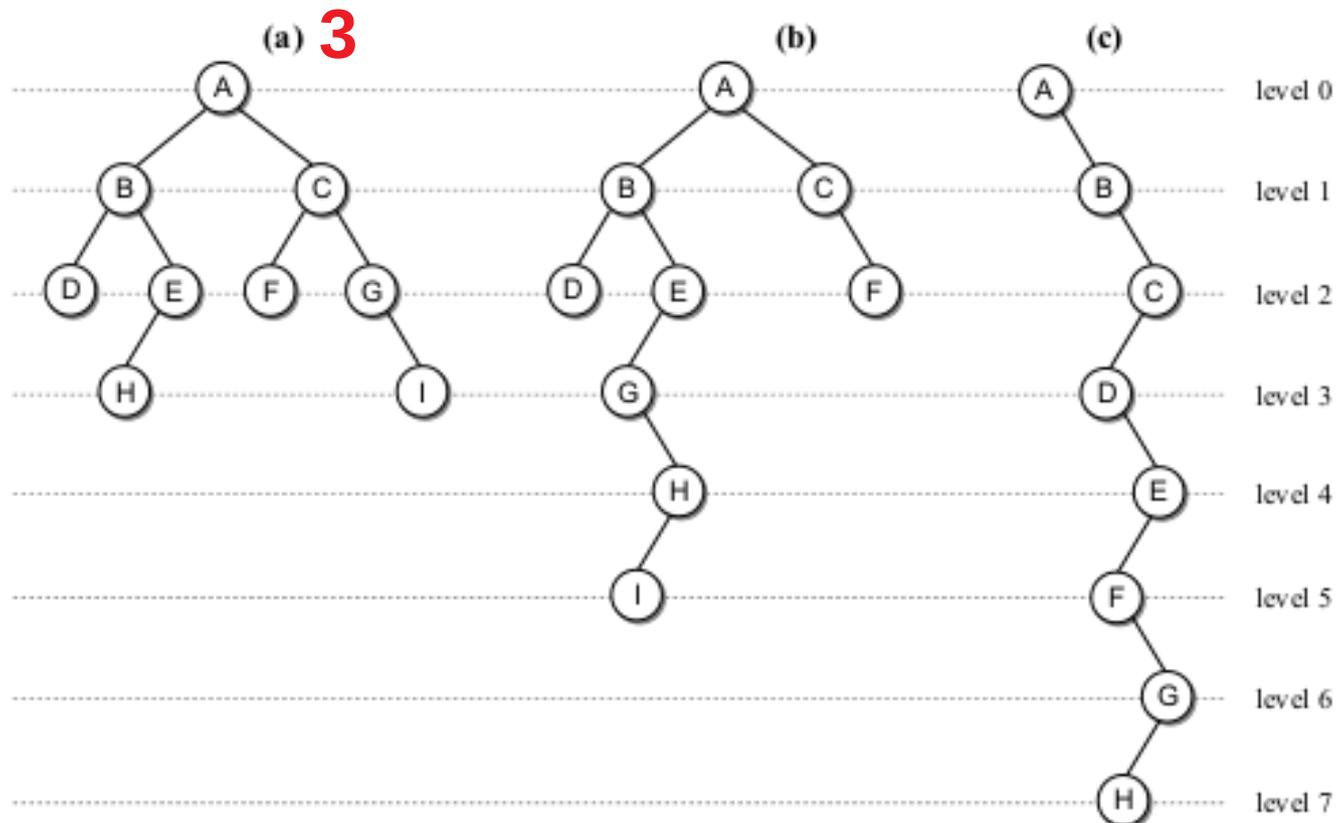
# Árvores Binárias

- *Altura (height)*
  - É o número de níveis de uma árvore.
  - Exemplo:



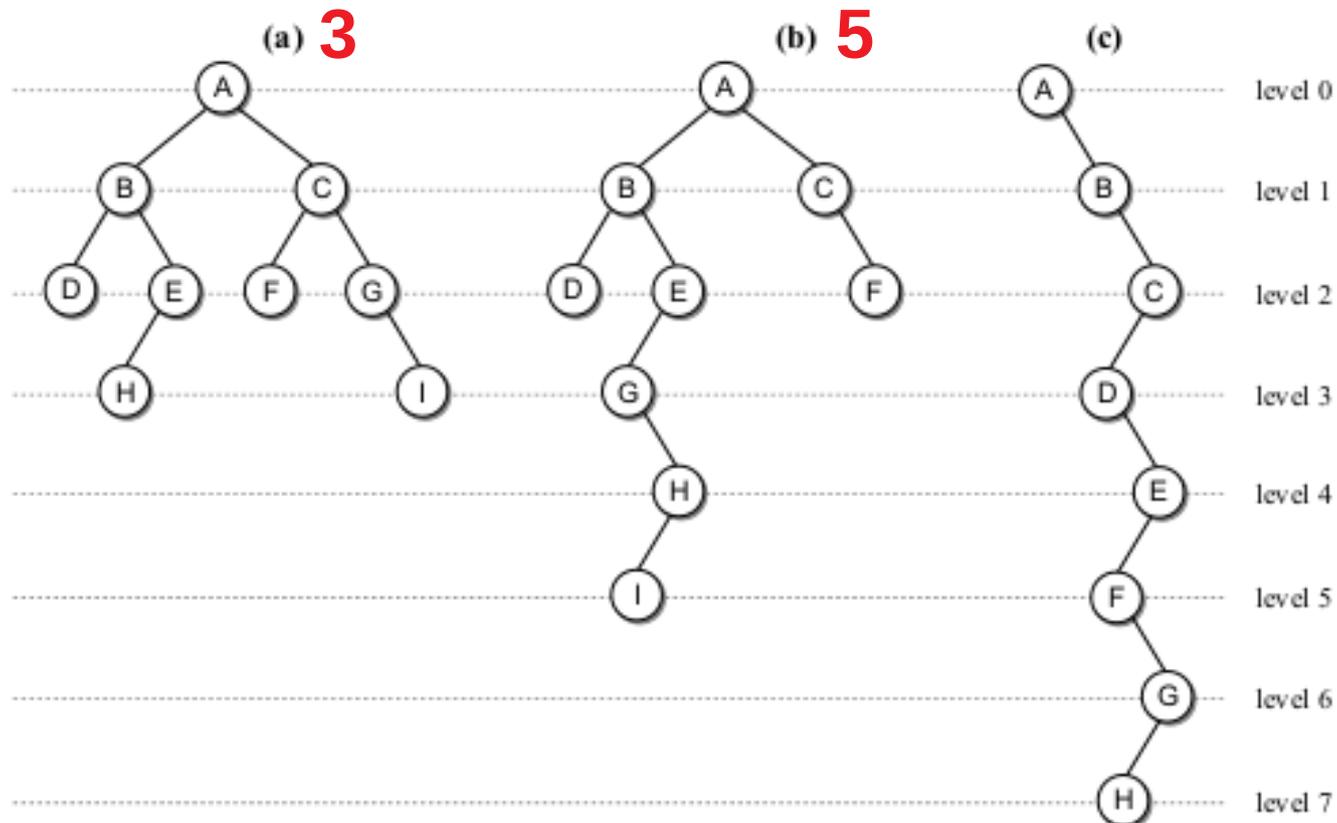
# Árvores Binárias

- *Altura (height)*
  - É o número de níveis de uma árvore.
  - Exemplo:



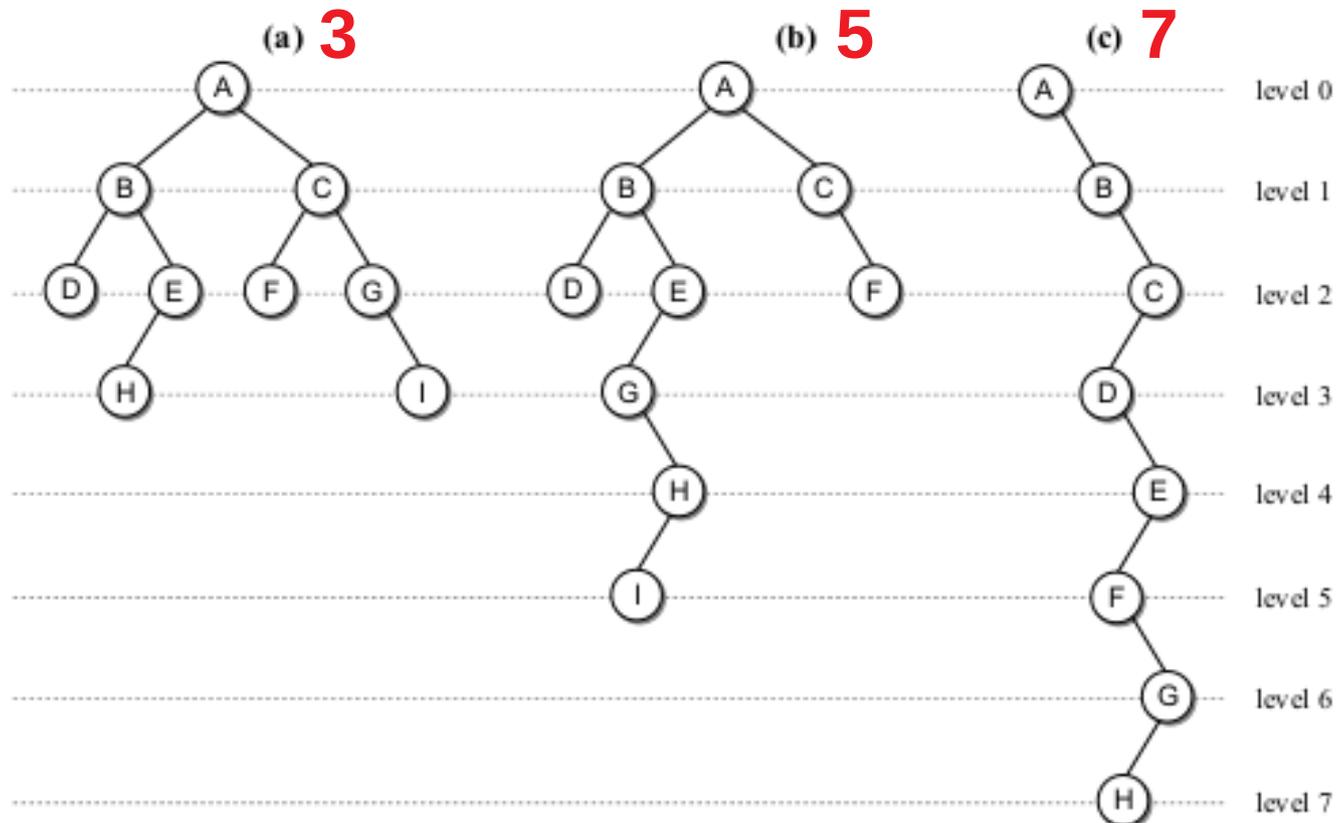
# Árvores Binárias

- Altura (*height*)
  - É o número de níveis de uma árvore.
  - Exemplo:



# Árvores Binárias

- Altura (*height*)
  - É o número de níveis de uma árvore.
  - Exemplo:



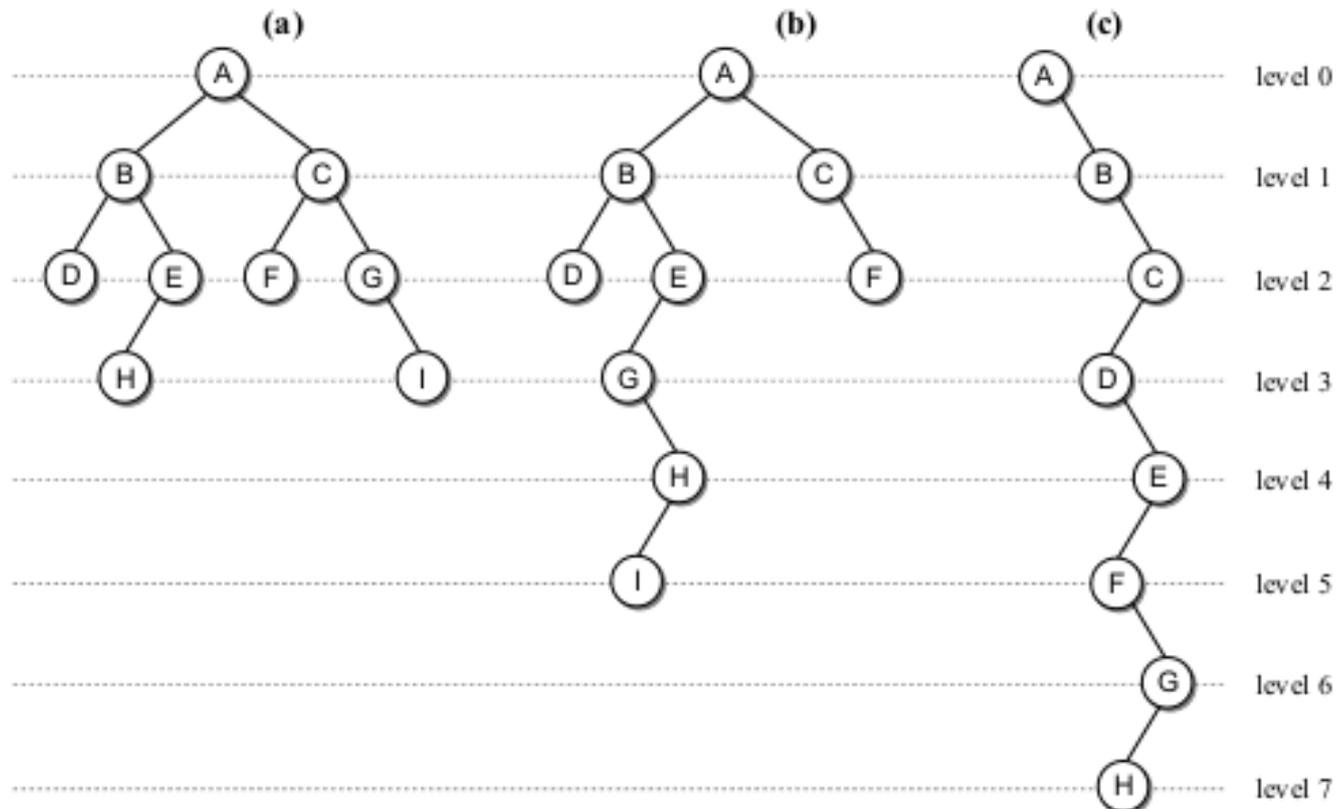
# Árvores Binárias

# Árvores Binárias

- Largura (*width*)
  - É o número de nós no nível que contém a maioria dos nós.

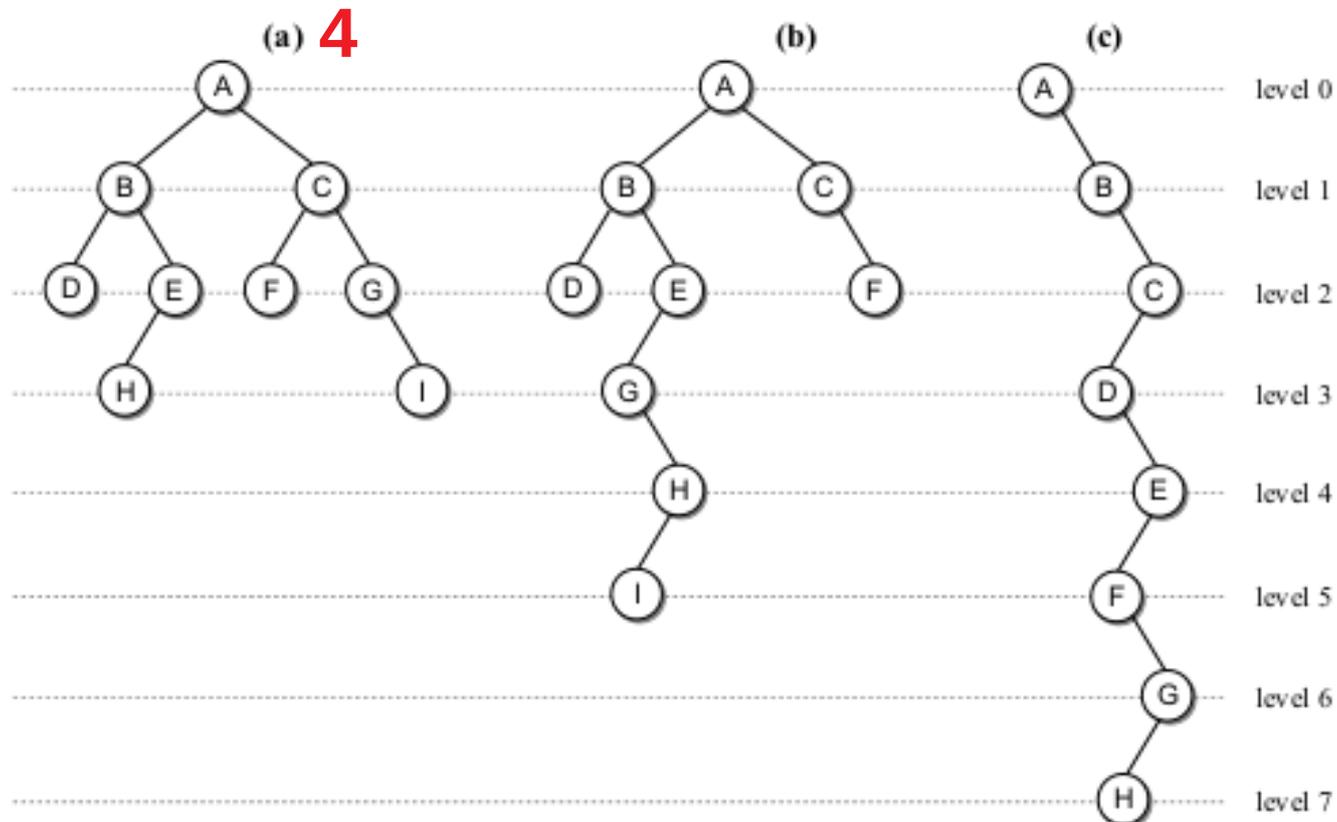
# Árvores Binárias

- Largura (*width*)
  - É o número de nós no nível que contém a maioria dos nós.



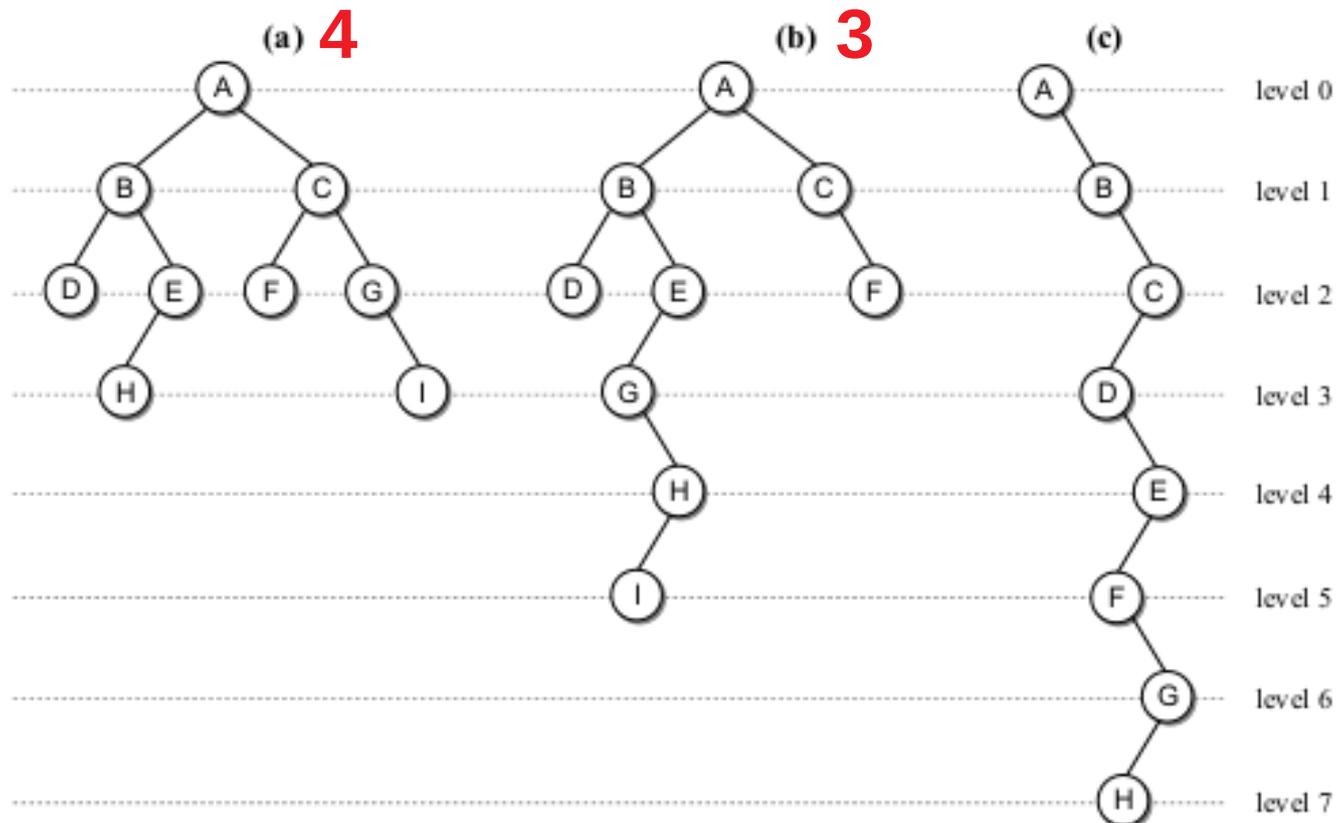
# Árvores Binárias

- Largura (*width*)
  - É o número de nós no nível que contém a maioria dos nós.



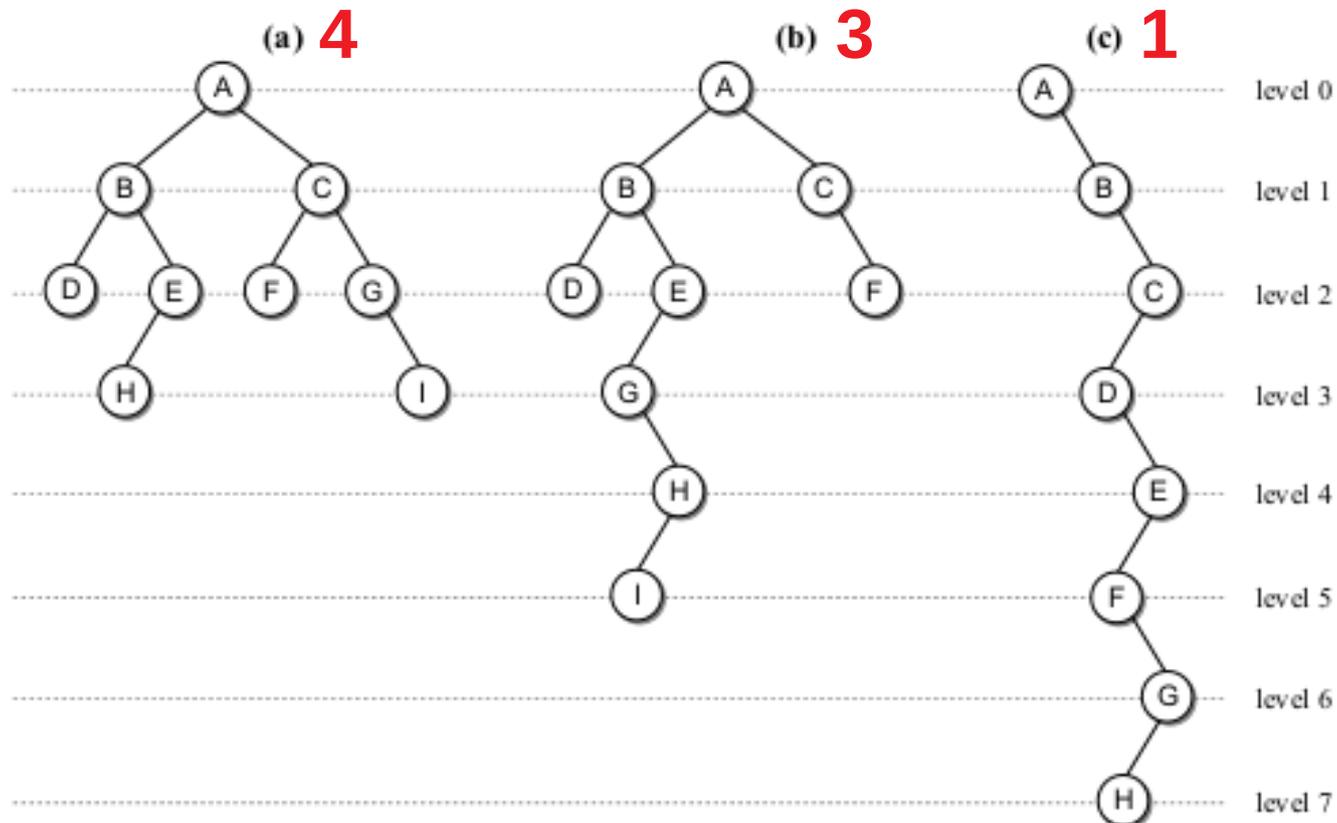
# Árvores Binárias

- Largura (*width*)
  - É o número de nós no nível que contém a maioria dos nós.



# Árvores Binárias

- Largura (*width*)
  - É o número de nós no nível que contém a maioria dos nós.



# Árvores Binárias

# Árvores Binárias

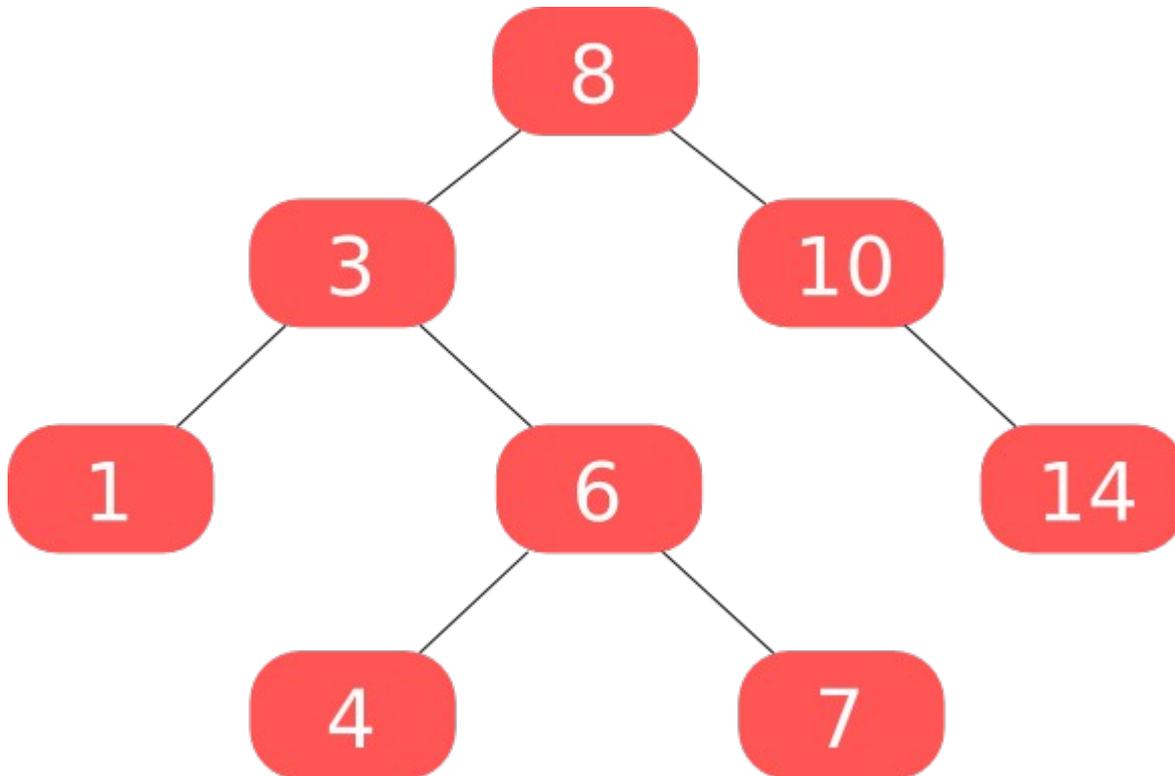
- Tamanho (*size*)

# Árvores Binárias

- Tamanho (*size*)
  - O tamanho máximo de uma árvore é o número de nós da árvore.

# Árvores Binárias

- Tamanho (*size*)
  - O tamanho máximo de uma árvore é o número de nós da árvore.



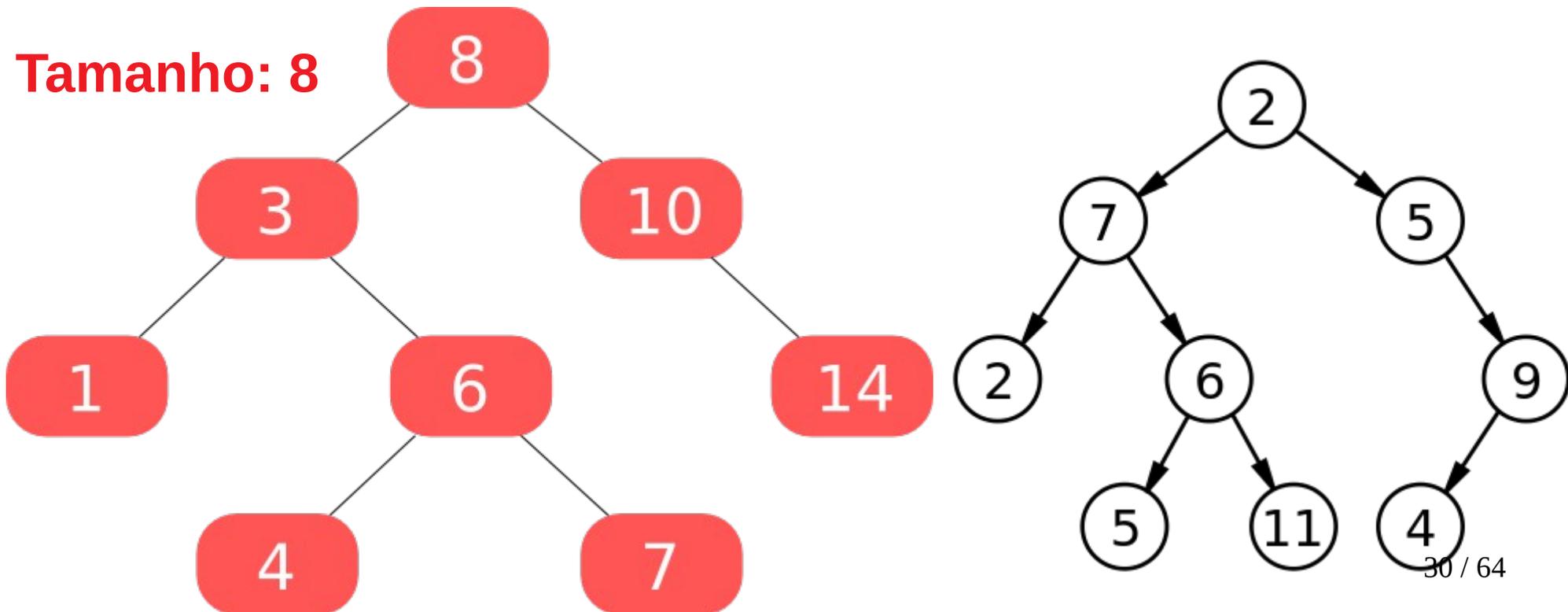
# Árvores Binárias

- Tamanho (*size*)
  - O tamanho máximo de uma árvore é o número de nós da árvore.



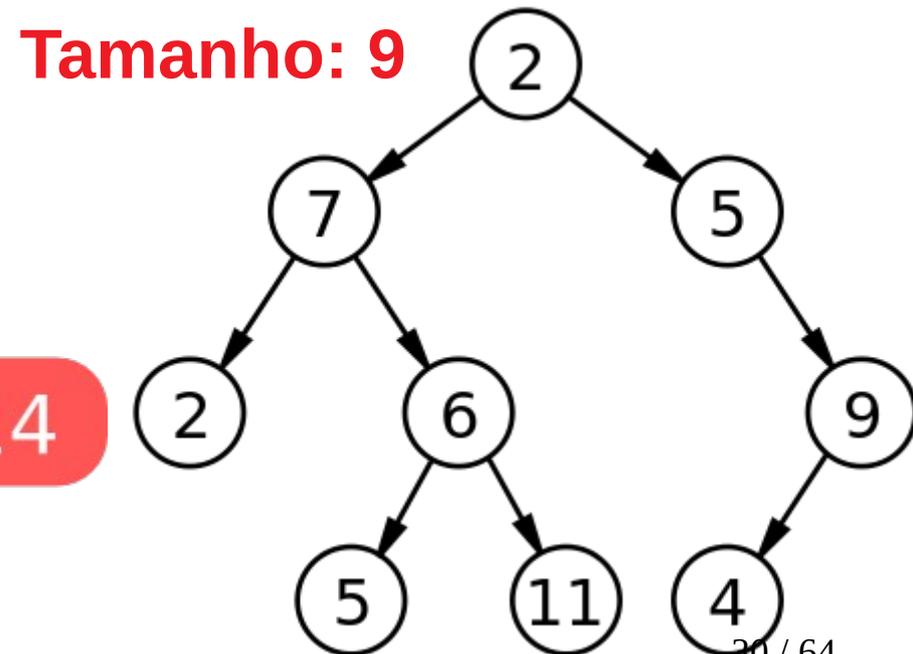
# Árvores Binárias

- Tamanho (*size*)
  - O tamanho máximo de uma árvore é o número de nós da árvore.



# Árvores Binárias

- Tamanho (*size*)
  - O tamanho máximo de uma árvore é o número de nós da árvore.



# Árvores Binárias

# Árvores Binárias

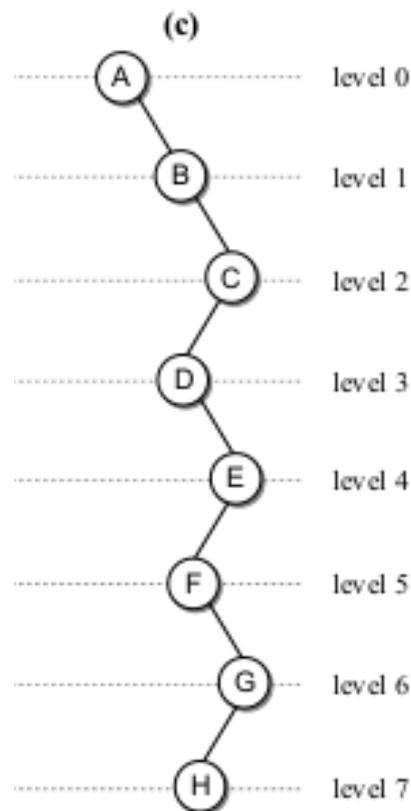
- Uma árvore binária de tamanho  $n$  pode ter uma altura máxima de  $n$ , quando existe um nó por nível.

# Árvores Binárias

- Uma árvore binária de tamanho  $n$  pode ter uma altura máxima de  $n$ , quando existe um nó por nível.
- Exemplo:

# Árvores Binárias

- Uma árvore binária de tamanho  $n$  pode ter uma altura máxima de  $n$ , quando existe um nó por nível.
- Exemplo:



# Árvores Binárias

# Árvores Binárias

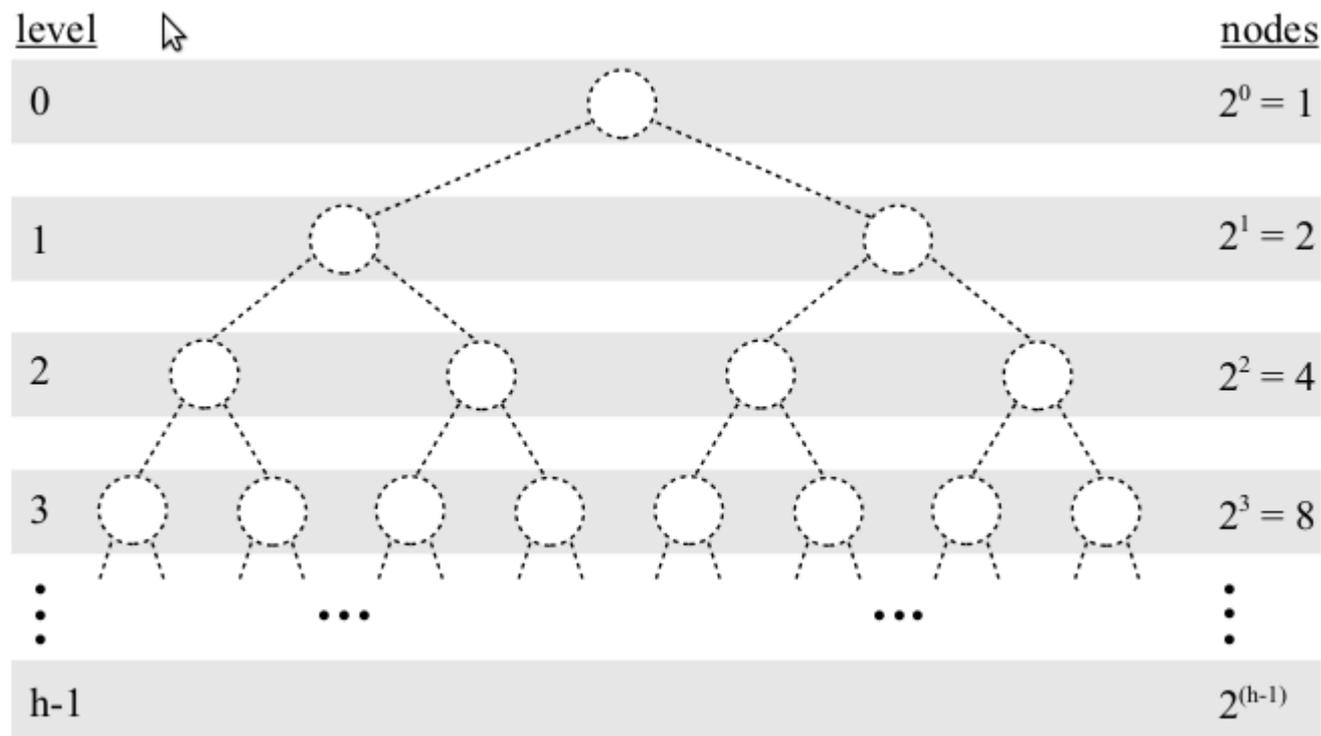
- Qual é a altura mínima de uma árvore binária com  $n$  nós?

# Árvores Binárias

- Qual é a altura mínima de uma árvore binária com  $n$  nós?
  - Cada nível  $i$  terá  $2^i$  nós.

# Árvores Binárias

- Qual é a altura mínima de uma árvore binária com  $n$  nós?
  - Cada nível  $i$  terá  $2^i$  nós.



# Árvores Binárias

# Árvores Binárias

- Qual é a altura mínima de uma árvore binária com  $n$  nós?

# Árvores Binárias

- Qual é a altura mínima de uma árvore binária com  $n$  nós?

$$n = 2^0 + 2^1 + 2^2 + \dots + 2^{d-1} + 2^d$$

# Árvores Binárias

- Qual é a altura mínima de uma árvore binária com  $n$  nós?

$$n = 2^0 + 2^1 + 2^2 + \dots + 2^{d-1} + 2^d$$

$$n = 2^{d+1} - 1$$

# Árvores Binárias

- Qual é a altura mínima de uma árvore binária com  $n$  nós?

$$n = 2^0 + 2^1 + 2^2 + \dots + 2^{d-1} + 2^d$$

$$n = 2^{d+1} - 1$$

$$d = \log(n + 1) - 1$$

# Árvores Binárias

# Árvores Binárias

- Árvore binária cheia (*full binary tree*)

# Árvores Binárias

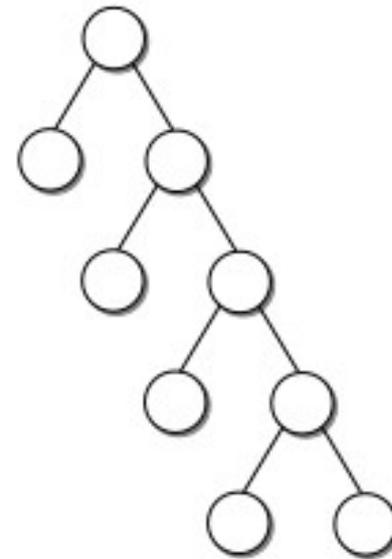
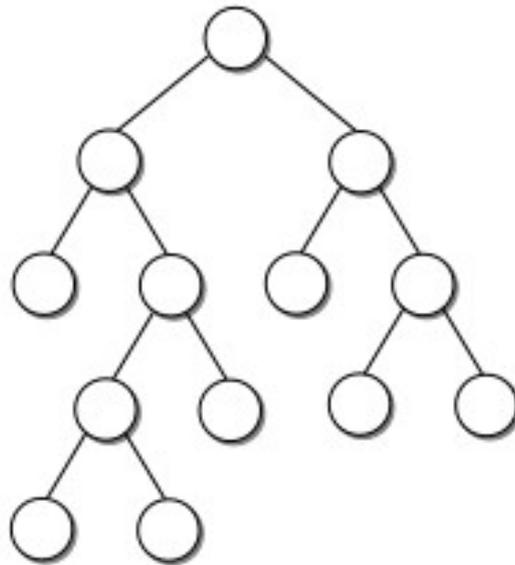
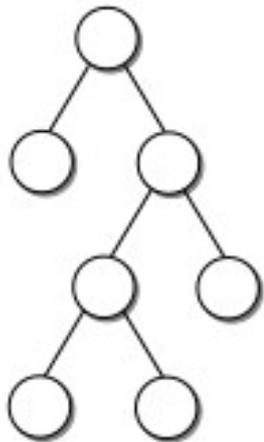
- Árvore binária cheia (*full binary tree*)
  - É uma árvore em que cada nó interior contém dois filhos.

# Árvores Binárias

- Árvore binária cheia (*full binary tree*)
  - É uma árvore em que cada nó interior contém dois filhos.
  - Exemplos:

# Árvores Binárias

- Árvore binária cheia (*full binary tree*)
  - É uma árvore em que cada nó interior contém dois filhos.
  - Exemplos:



# Árvores Binárias

# Árvores Binárias

- Árvore binária perfeita (*perfect binary tree*)

# Árvores Binárias

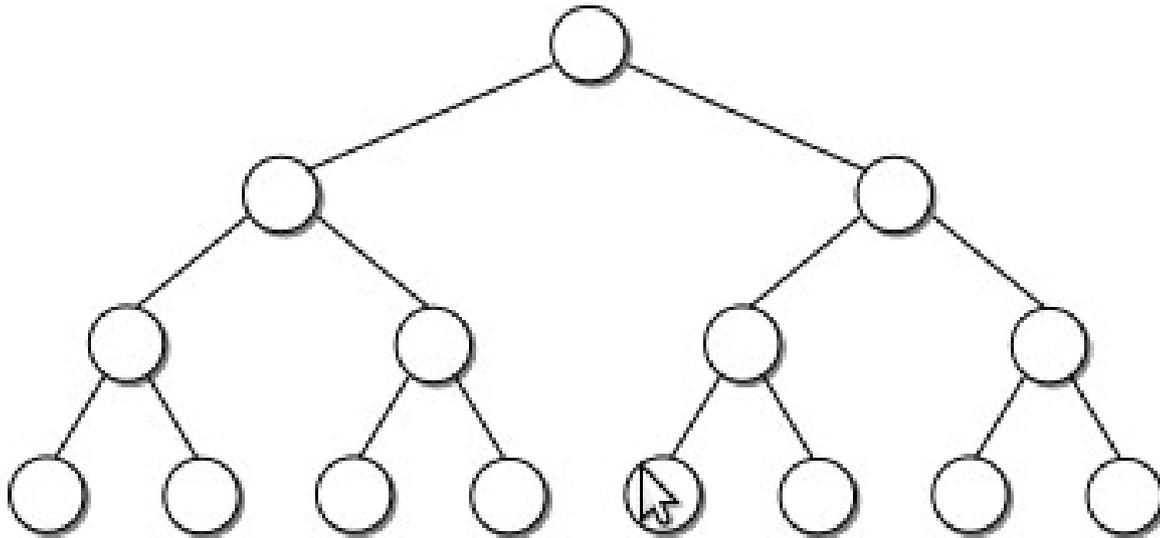
- Árvore binária perfeita (*perfect binary tree*)
  - É uma árvore binária em que todos os nós folhas estão no mesmo nível.

# Árvores Binárias

- Árvore binária perfeita (*perfect binary tree*)
  - É uma árvore binária em que todos os nós folhas estão no mesmo nível.
  - Exemplo:

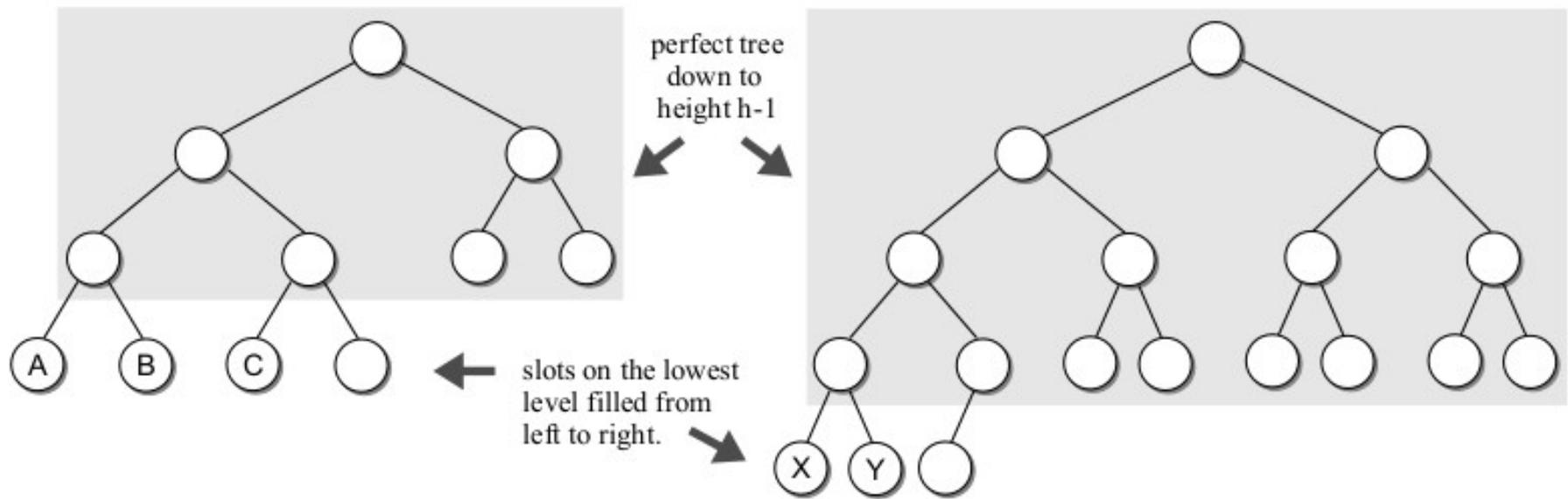
# Árvores Binárias

- Árvore binária perfeita (*perfect binary tree*)
  - É uma árvore binária em que todos os nós folhas estão no mesmo nível.
  - Exemplo:



# Árvores Binárias

- **Árvore binária completa** (*complete binary tree*)
  - É uma árvore completa em todos os níveis, exceto possivelmente o último.
  - Exemplo:



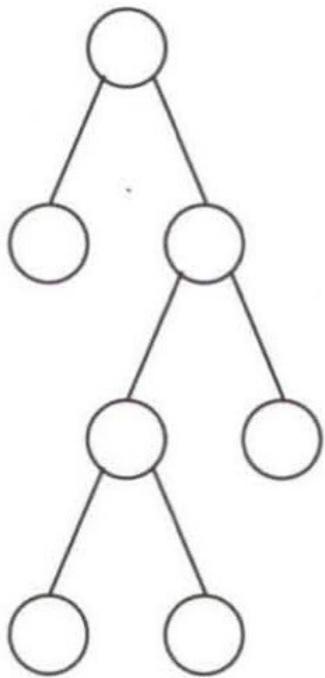
# Árvores Binárias

# Árvores Binárias

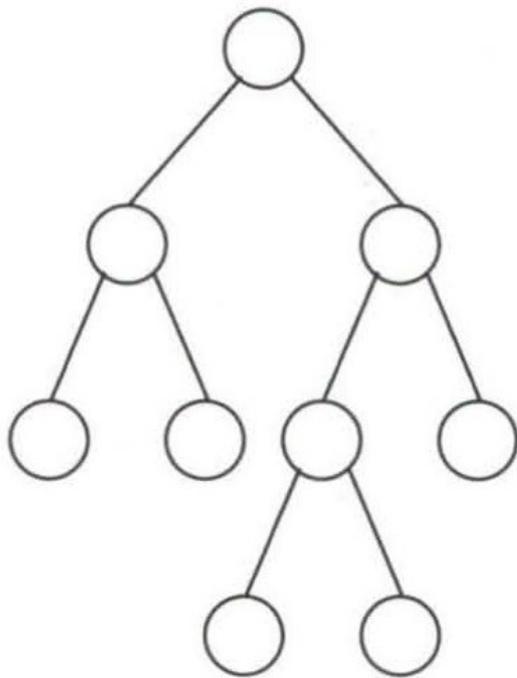
- Comparação

# Árvores Binárias

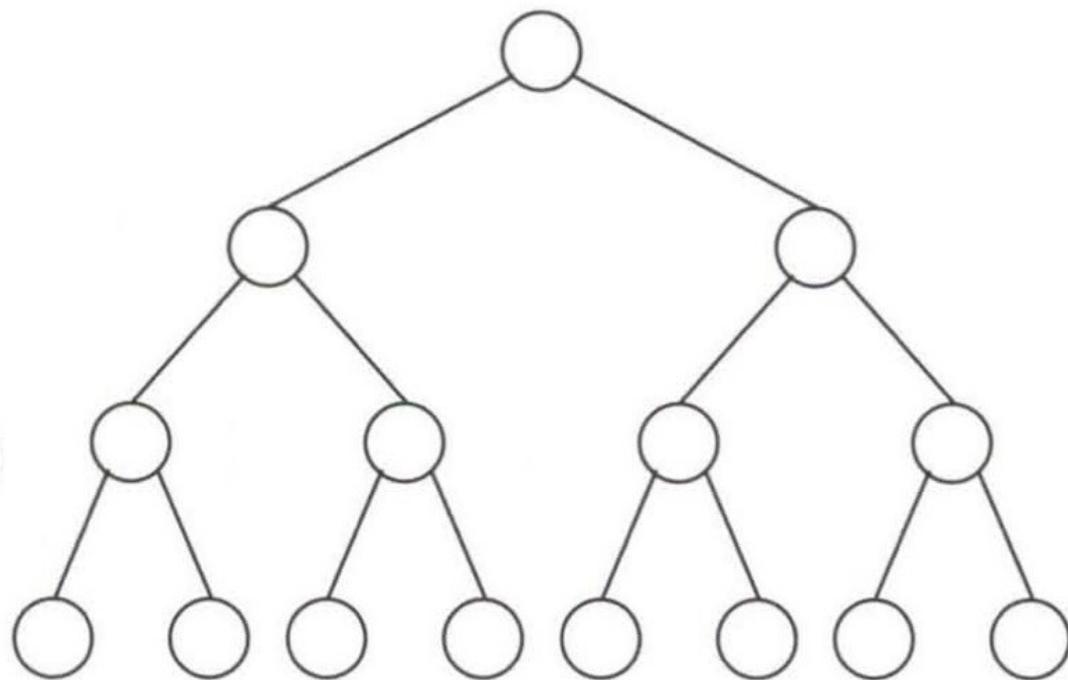
- Comparação



(a)



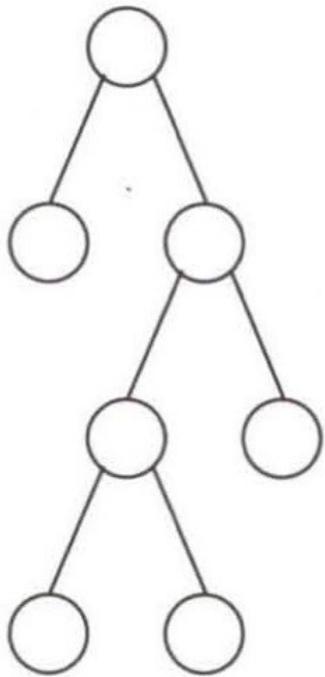
(b)



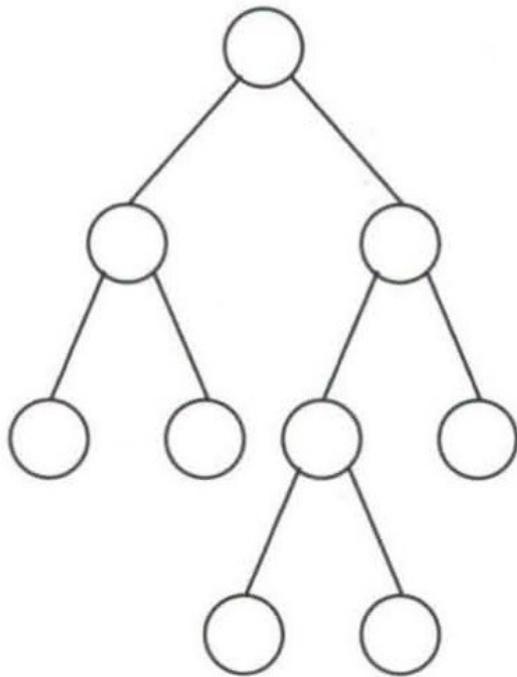
(c)

# Árvores Binárias

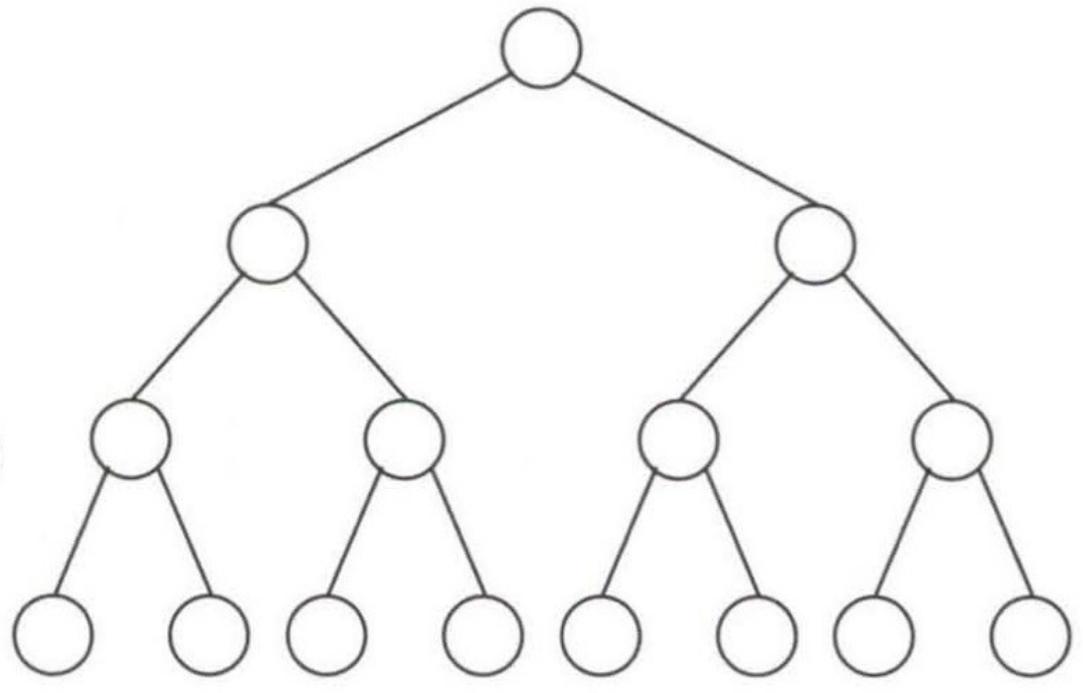
- Comparação



(a)



(b)



(c)

**estritamente binária**





# Árvores Binárias

# Árvores Binárias

- Árvores binárias são comumente implementadas como estruturas dinâmicas, assim como fizemos com as listas ligadas.

# Árvores Binárias

- Árvores binárias são comumente implementadas como estruturas dinâmicas, assim como fizemos com as listas ligadas.
- Uma árvore binária é uma estrutura de dados que pode ser usada para implementar diferentes tipos abstratos de dados (TAD).

# Árvores Binárias

- Árvores binárias são comumente implementadas como estruturas dinâmicas, assim como fizemos com as listas ligadas.
- Uma árvore binária é uma estrutura de dados que pode ser usada para implementar diferentes tipos abstratos de dados (TAD).
- Para a sua implementação, nós devemos explicitamente armazenar em cada nó as ligações (*links*) para os dois nós filhos juntamente com os dados armazenados em cada nó.

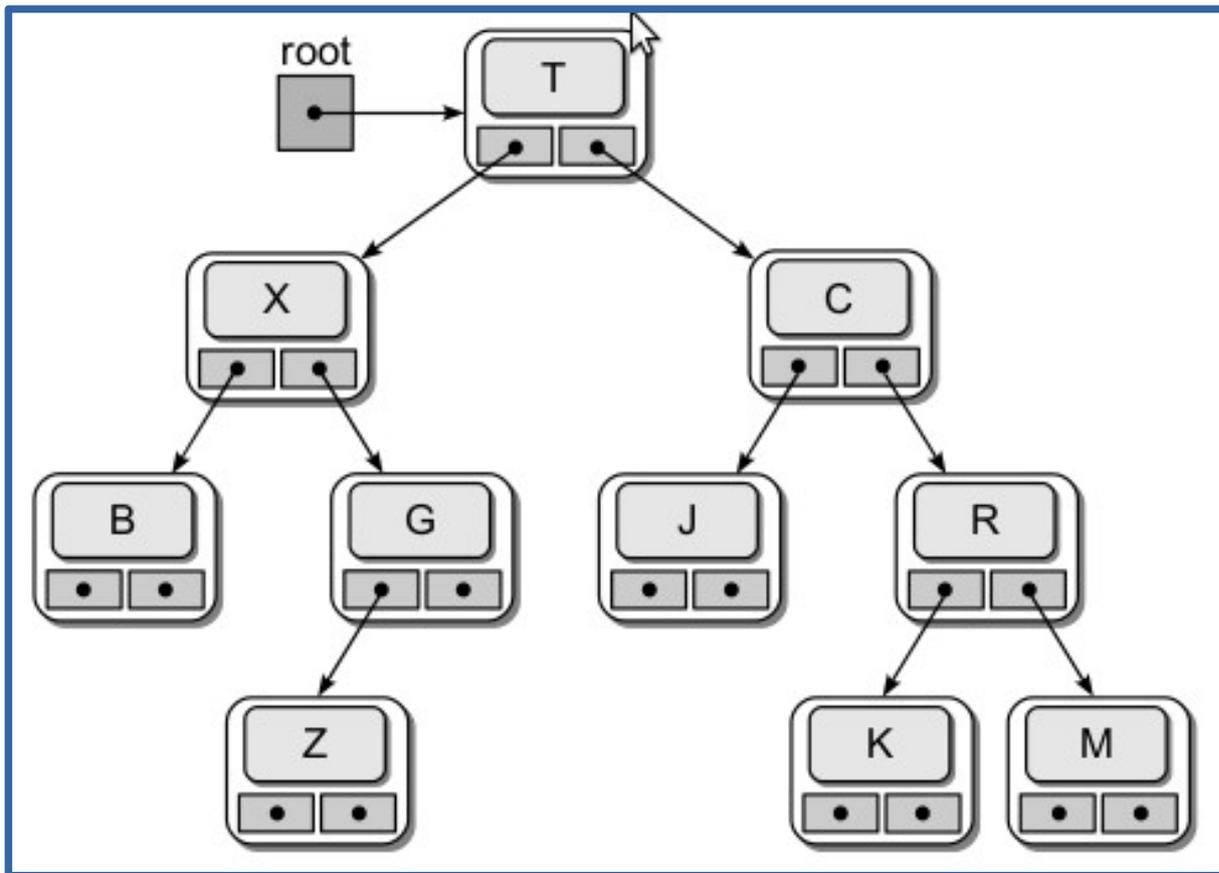
# Árvores Binárias

- Classe Nó

```
1 # The storage class for creating binary tree nodes.
2 class _BinTreeNode :
3     def __init__( self, data ):
4         self.data = data
5         self.left = None
6         self.right = None
```

# Árvores Binárias

- Exemplo de implementação física da AB:



# Árvores Binárias - Varredura

# Árvores Binárias - Varredura

- A operação de varredura é uma das mais importantes na manipulação de coleções de dados.

# Árvores Binárias - Varredura

- A operação de varredura é uma das mais importantes na manipulação de coleções de dados.
- O objetivo aqui é percorrer toda a coleção, acessando um elemento de cada vez.

# Árvores Binárias - Varredura

# Árvores Binárias - Varredura

- Como seria a busca em uma lista ligada?

# Árvores Binárias - Varredura

- Como seria a busca em uma lista ligada?
  - Bastaria iniciar do primeiro nó e percorrer a lista, seguindo as ligações, até o último nó.

# Árvores Binárias - Varredura

- Como seria a busca em uma lista ligada?
  - Bastaria iniciar do primeiro nó e percorrer a lista, seguindo as ligações, até o último nó.
- Mas seria possível visitar cada nó em uma árvore binária?

# Árvores Binárias - Varredura

- Como seria a busca em uma lista ligada?
  - Bastaria iniciar do primeiro nó e percorrer a lista, seguindo as ligações, até o último nó.
- Mas seria possível visitar cada nó em uma árvore binária?
  - Não existe um único caminho que parta da raiz e consiga visitar todos os nós.

# Árvores Binárias - Varredura

- Como seria a busca em uma lista ligada?
  - Bastaria iniciar do primeiro nó e percorrer a lista, seguindo as ligações, até o último nó.
- Mas seria possível visitar cada nó em uma árvore binária?
  - Não existe um único caminho que parta da raiz e consiga visitar todos os nós.
  - As árvores podem ser percorridas de várias formas.

# Árvores Binárias - Varredura

- Existem vários métodos de varredura (caminhamento) em árvores, que permitem percorrê-la de forma sistemática e de tal modo que cada nó seja visitado apenas uma vez.

# Árvores Binárias - Varredura

# Árvores Binárias - Varredura

- Existem 3 principais ordens de caminhamento:

# Árvores Binárias - Varredura

- Existem 3 principais ordens de caminhamento:
  - Pré-fixado

# Árvores Binárias - Varredura

- Existem 3 principais ordens de caminhamento:
  - Pré-fixado
  - Central

# Árvores Binárias - Varredura

- Existem 3 principais ordens de caminhamento:
  - Pré-fixado
  - Central
  - Pós-fixado

# Árvores Binárias - Varredura

# Árvores Binárias - Varredura

- Pré-fixado

# Árvores Binárias - Varredura

- Pré-fixado
  - Visita a raiz.

# Árvores Binárias - Varredura

- Pré-fixado
  - Visita a raiz.
  - Percorre a subárvore da esquerda.

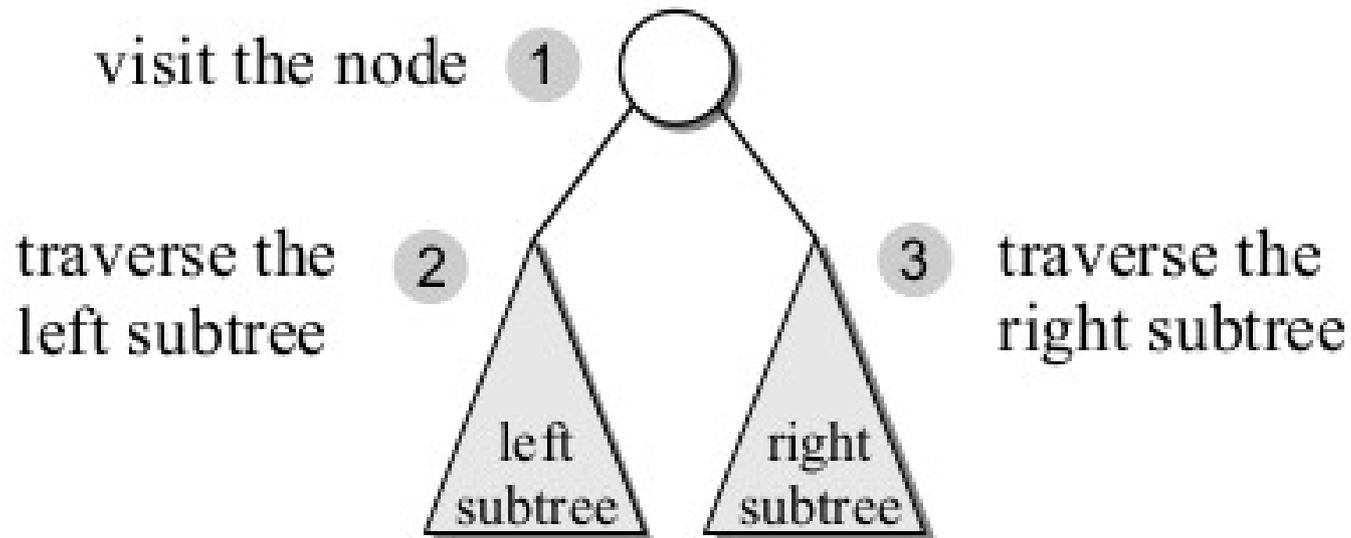
# Árvores Binárias - Varredura

- Pré-fixado
  - Visita a raiz.
  - Percorre a subárvore da esquerda.
  - Percorre a subárvore da direita.

# Árvores Binárias - Varredura

- Pré-fixado

- Visita a raiz.
- Percorre a subárvore da esquerda.
- Percorre a subárvore da direita.



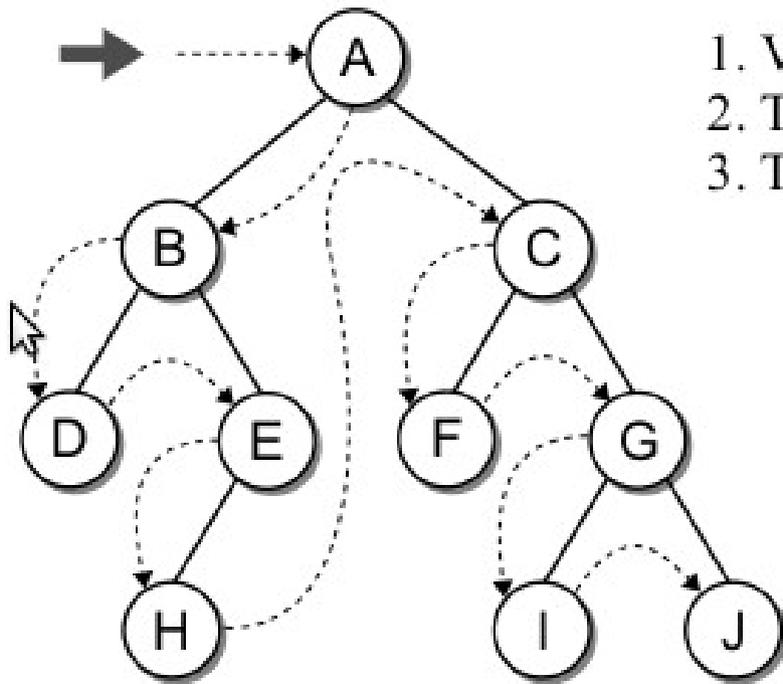
# Árvores Binárias - Varredura

# Árvores Binárias - Varredura

- Exemplo de pré-fixado:

# Árvores Binárias - Varredura

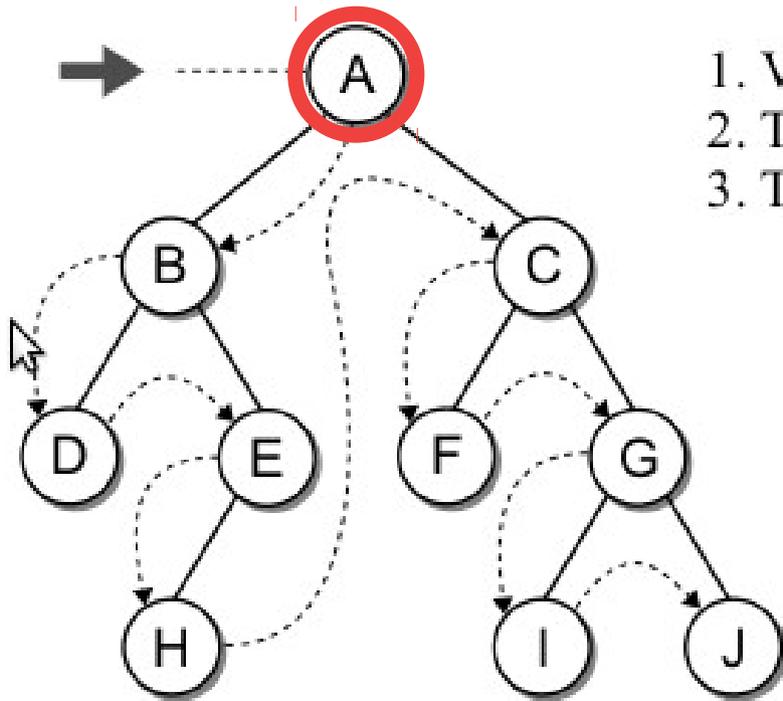
- Exemplo de pré-fixado:



1. Visit the node.
2. Traverse the left subtree.
3. Traverse the right subtree.

# Árvores Binárias - Varredura

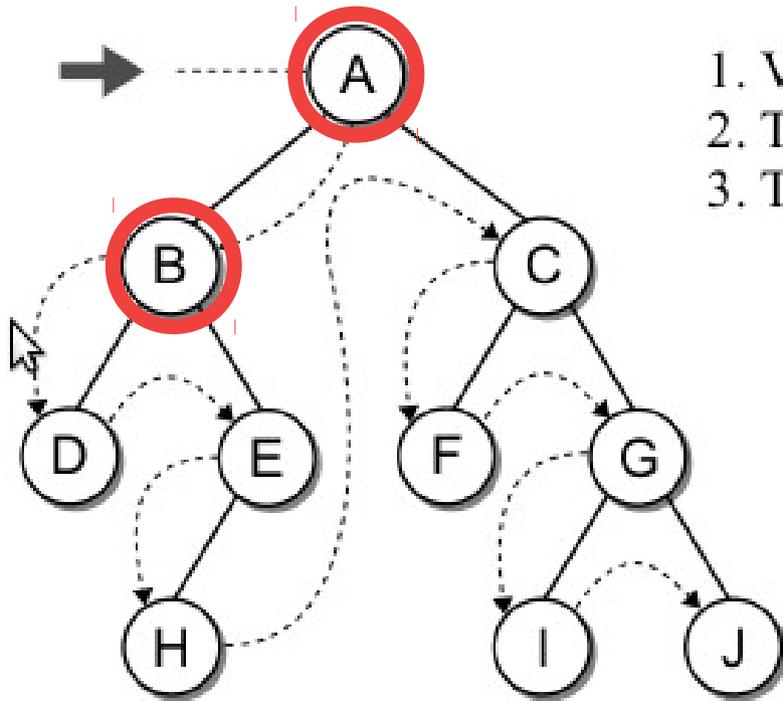
- Exemplo de pré-fixado:



1. Visit the node.
2. Traverse the left subtree.
3. Traverse the right subtree.

# Árvores Binárias - Varredura

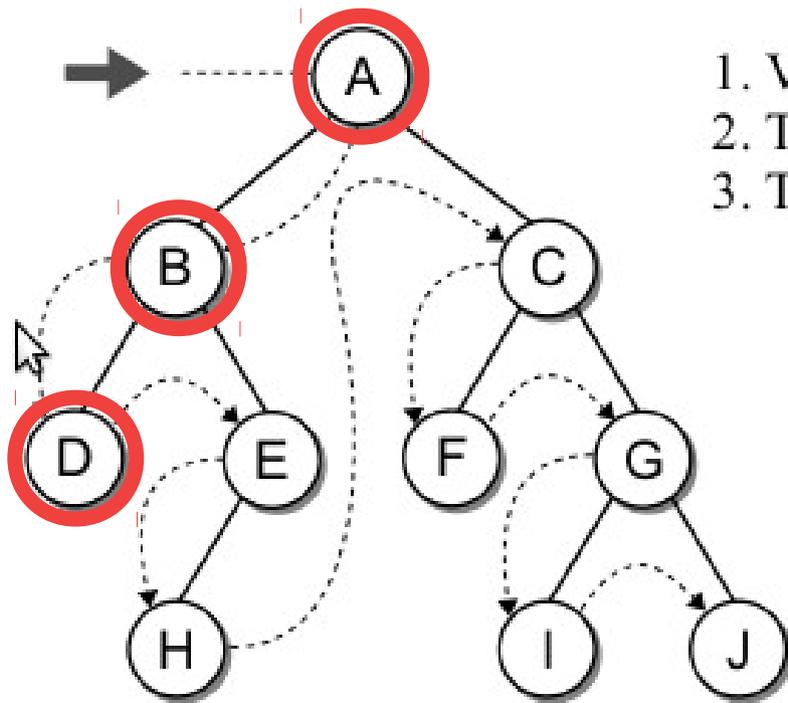
- Exemplo de pré-fixado:



1. Visit the node.
2. Traverse the left subtree.
3. Traverse the right subtree.

# Árvores Binárias - Varredura

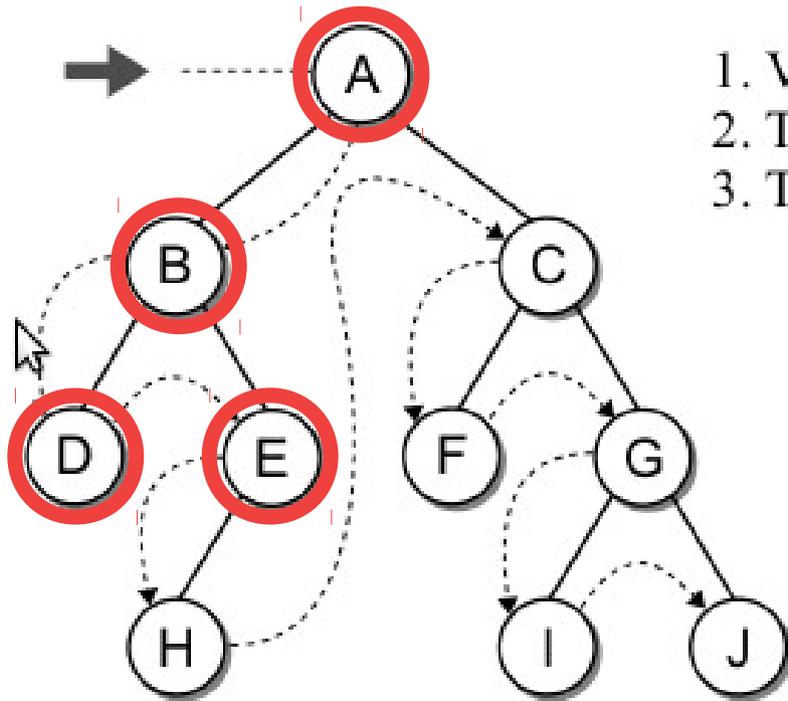
- Exemplo de pré-fixado:



1. Visit the node.
2. Traverse the left subtree.
3. Traverse the right subtree.

# Árvores Binárias - Varredura

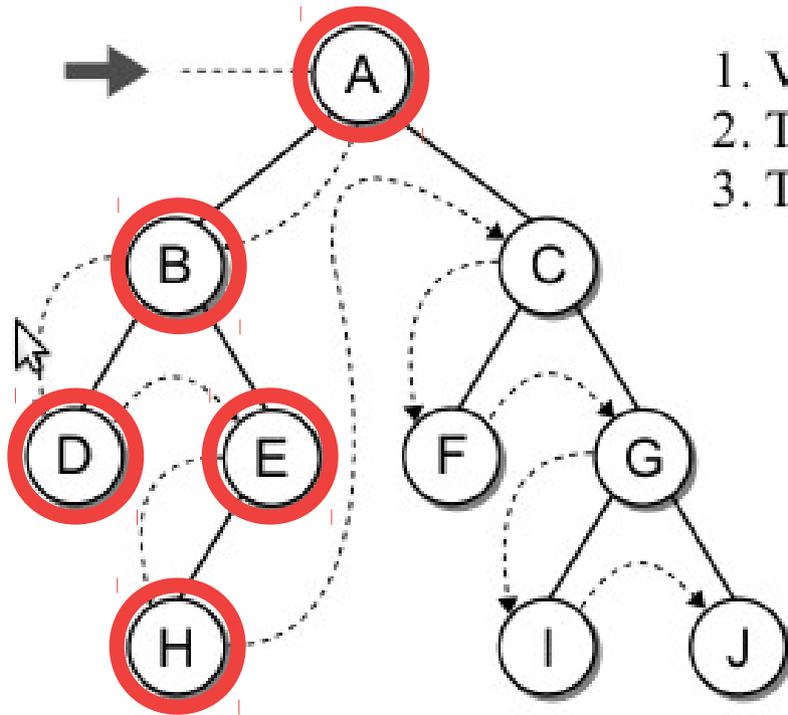
- Exemplo de pré-fixado:



1. Visit the node.
2. Traverse the left subtree.
3. Traverse the right subtree.

# Árvores Binárias - Varredura

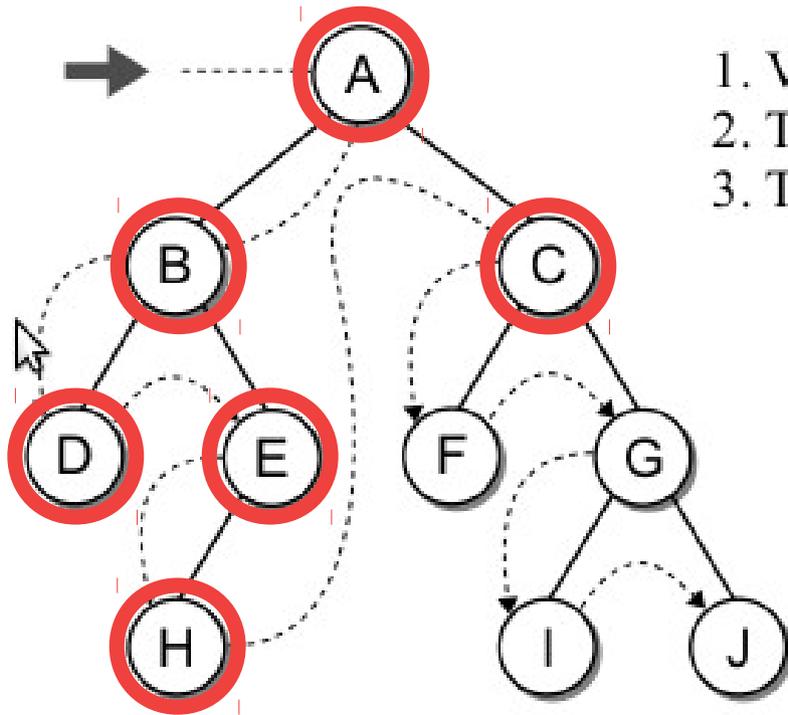
- Exemplo de pré-fixado:



1. Visit the node.
2. Traverse the left subtree.
3. Traverse the right subtree.

# Árvores Binárias - Varredura

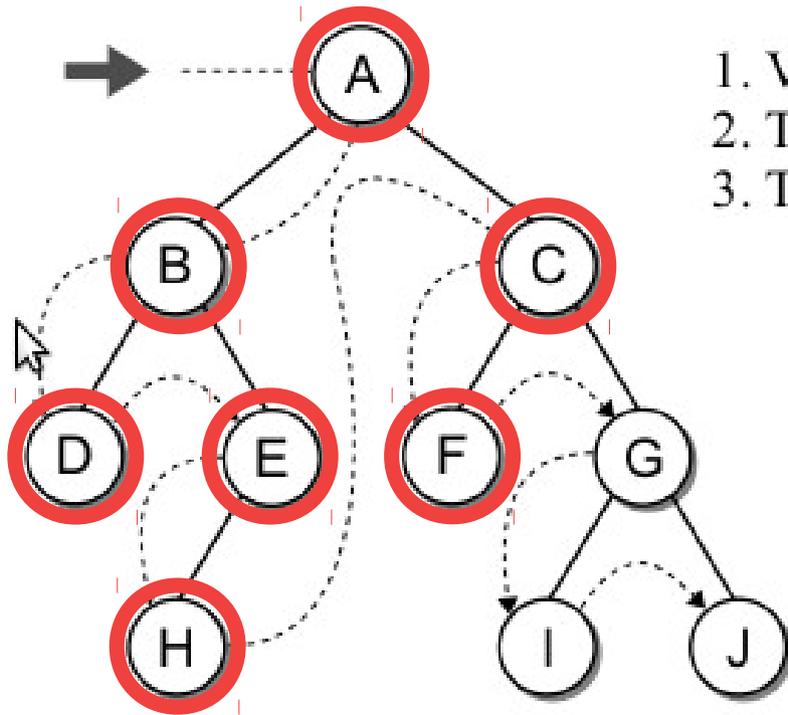
- Exemplo de pré-fixado:



1. Visit the node.
2. Traverse the left subtree.
3. Traverse the right subtree.

# Árvores Binárias - Varredura

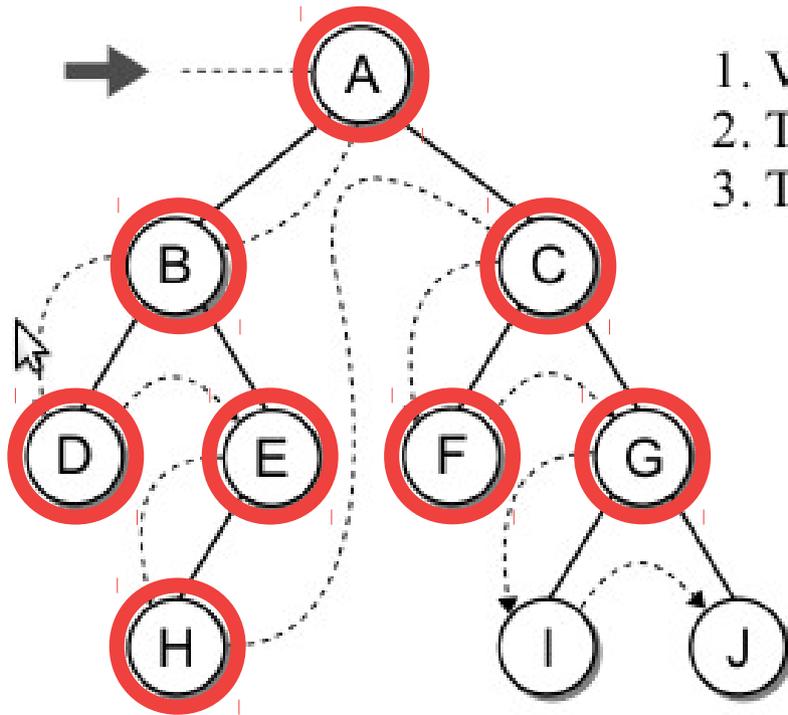
- Exemplo de pré-fixado:



1. Visit the node.
2. Traverse the left subtree.
3. Traverse the right subtree.

# Árvores Binárias - Varredura

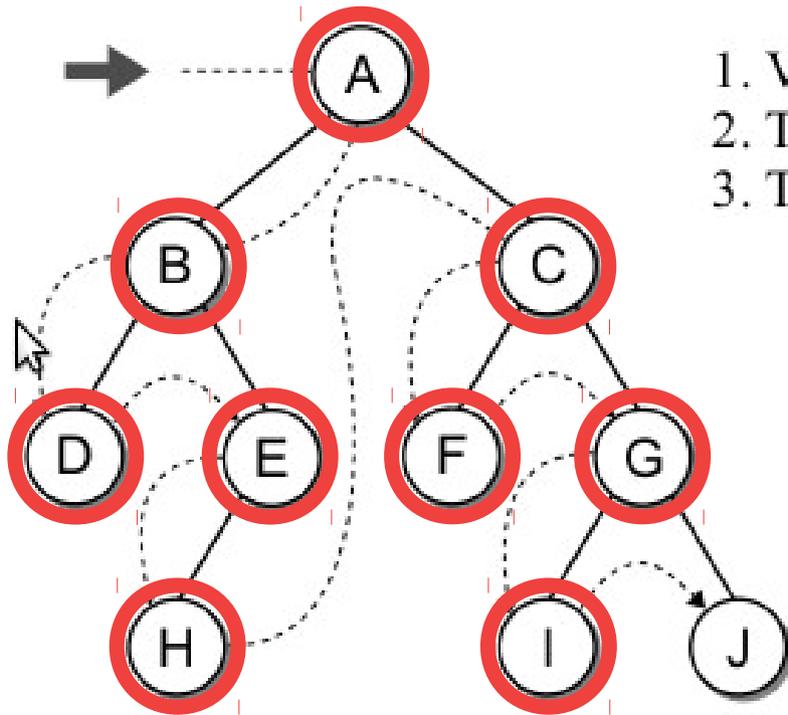
- Exemplo de pré-fixado:



1. Visit the node.
2. Traverse the left subtree.
3. Traverse the right subtree.

# Árvores Binárias - Varredura

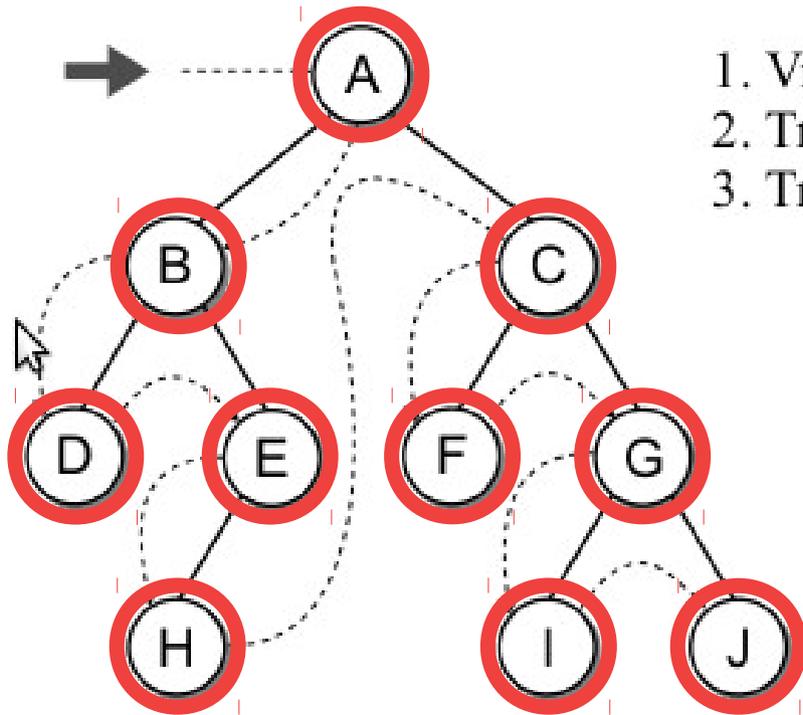
- Exemplo de pré-fixado:



1. Visit the node.
2. Traverse the left subtree.
3. Traverse the right subtree.

# Árvores Binárias - Varredura

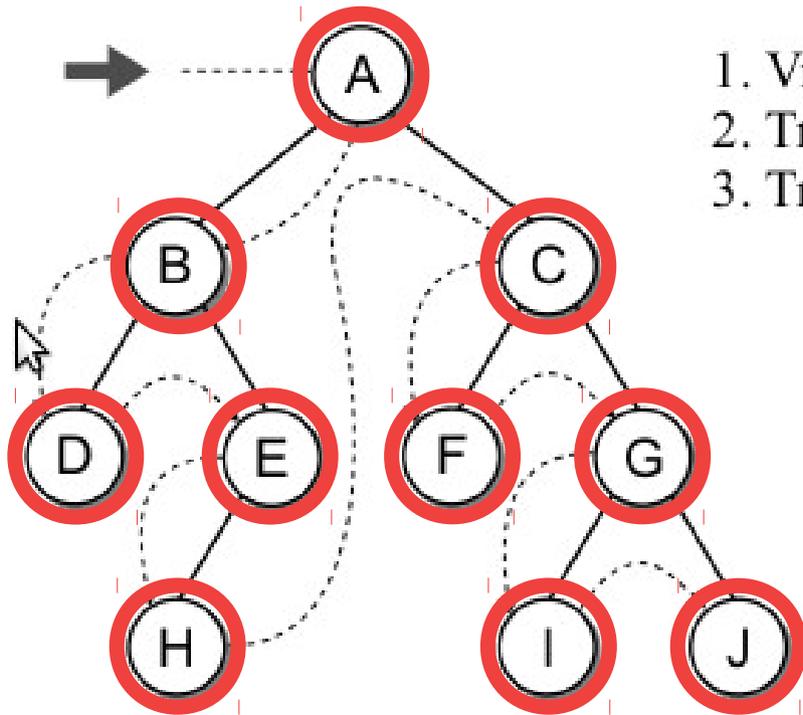
- Exemplo de pré-fixado:



1. Visit the node.
2. Traverse the left subtree.
3. Traverse the right subtree.

# Árvores Binárias - Varredura

- Exemplo de pré-fixado:



1. Visit the node.
2. Traverse the left subtree.
3. Traverse the right subtree.

A → B → D → E → H → C → F → G → I → J

# Árvores Binárias - Varredura

# Árvores Binárias - Varredura

- A função para caminho pré-fixado:

# Árvores Binárias - Varredura

- A função para caminho pré-fixado:
  - Função recursiva.

# Árvores Binárias - Varredura

- A função para caminho pré-fixado:
  - Função recursiva.

```
1  def preorderTrav( subtree ):  
2      if subtree is not None :  
3          print( subtree.data )  
4          preorderTrav( subtree.left )  
5          preorderTrav( subtree.right )
```

# Árvores Binárias - Varredura

- A função para caminho pré-fixado:
  - Função recursiva.
  - O parâmetro será uma subárvore:

```
1  def preorderTrav( subtree ):  
2      if subtree is not None :  
3          print( subtree.data )  
4          preorderTrav( subtree.left )  
5          preorderTrav( subtree.right )
```

# Árvores Binárias - Varredura

- A função para caminho pré-fixado:
  - Função recursiva.
  - O parâmetro será uma subárvore:
    - Referência null (None) ou

```
1 def preorderTrav( subtree ):  
2     if subtree is not None :  
3         print( subtree.data )  
4         preorderTrav( subtree.left )  
5         preorderTrav( subtree.right )
```

# Árvores Binárias - Varredura

- A função para caminho pré-fixado:
  - Função recursiva.
  - O parâmetro será uma subárvore:
    - Referência null (None) ou
    - Referência para o nó raiz de uma subárvore.

```
1 def preorderTrav( subtree ):  
2     if subtree is not None :  
3         print( subtree.data )  
4         preorderTrav( subtree.left )  
5         preorderTrav( subtree.right )
```

# Árvores Binárias - Varredura

# Árvores Binárias - Varredura

- Caminhamento pré-fixado

# Árvores Binárias - Varredura

- Caminhamento pré-fixado
  - Dada uma árvore binária de tamanho  $n$ , o caminhamento completo dessa árvore visitará cada nó uma única vez.

# Árvores Binárias - Varredura

- Caminhamento pré-fixado
  - Dada uma árvore binária de tamanho  $n$ , o caminhamento completo dessa árvore visitará cada nó uma única vez.
  - Se a operação de visita requerer tempo constante, então o tempo de caminhamento será feito em  $O(n)$ .

# Árvores Binárias - Varredura

# Árvores Binárias - Varredura

- Central

# Árvores Binárias - Varredura

- Central
  - Percorre a subárvore da esquerda.

# Árvores Binárias - Varredura

- Central
  - Percorre a subárvore da esquerda.
  - Visita a raiz.

# Árvores Binárias - Varredura

- Central
  - Percorre a subárvore da esquerda.
  - Visita a raiz.
  - Percorre a subárvore da direita.

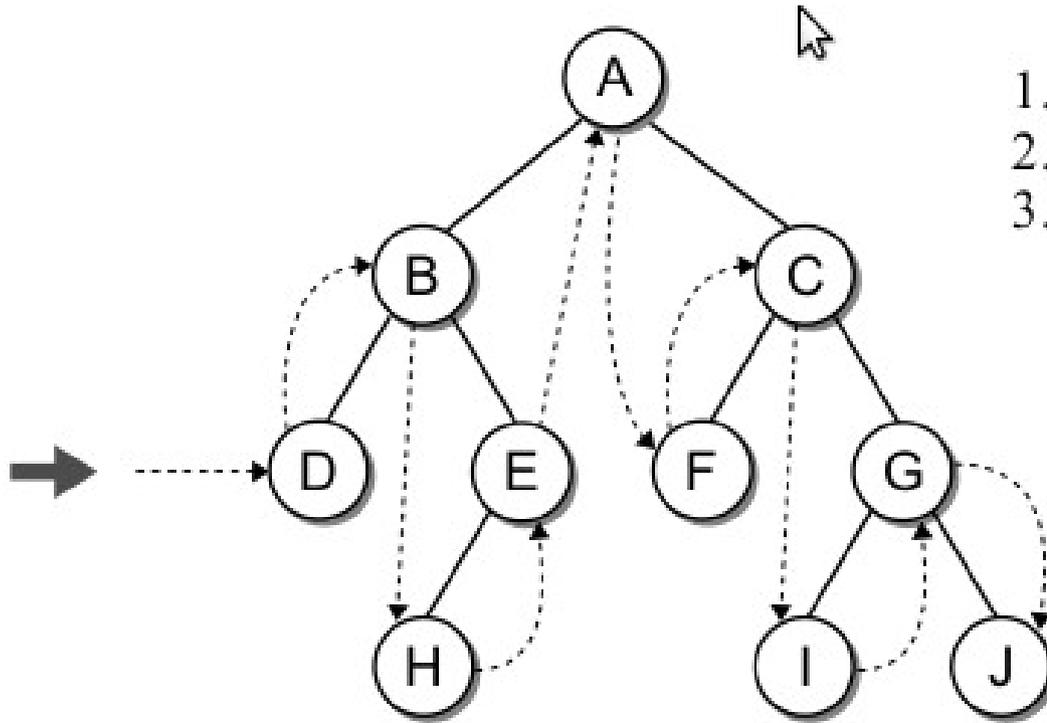
# Árvores Binárias - Varredura

# Árvores Binárias - Varredura

- Exemplo de caminhamento central

# Árvores Binárias - Varredura

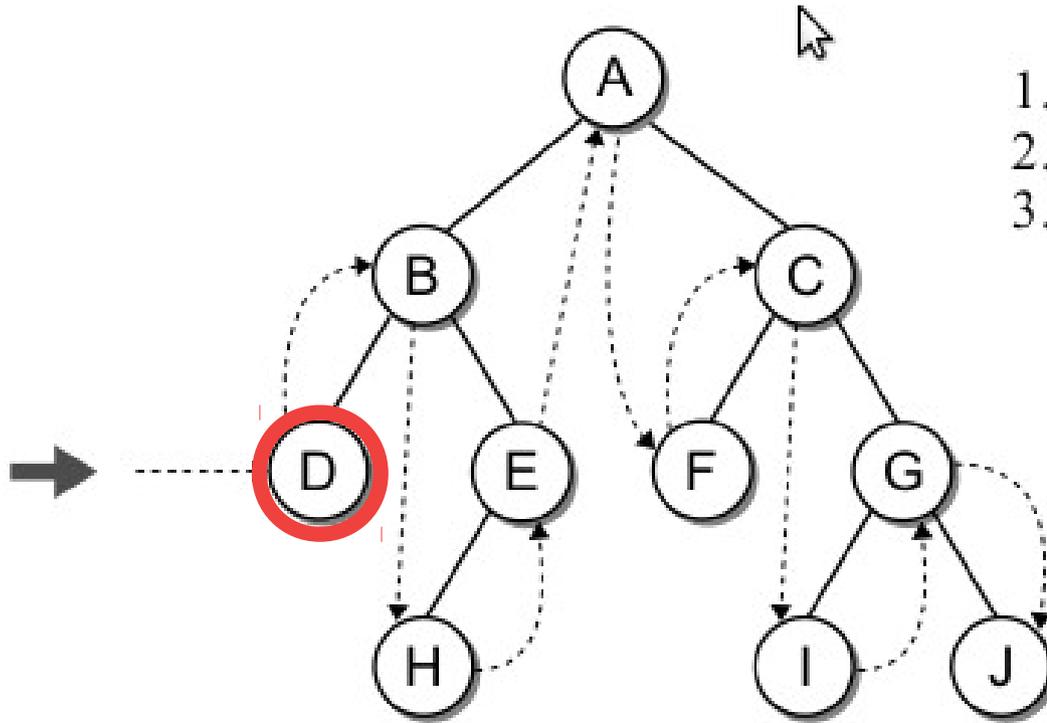
- Exemplo de caminhamento central



1. Traverse the left subtree.
2. Visit the node.
3. Traverse the right subtree.

# Árvores Binárias - Varredura

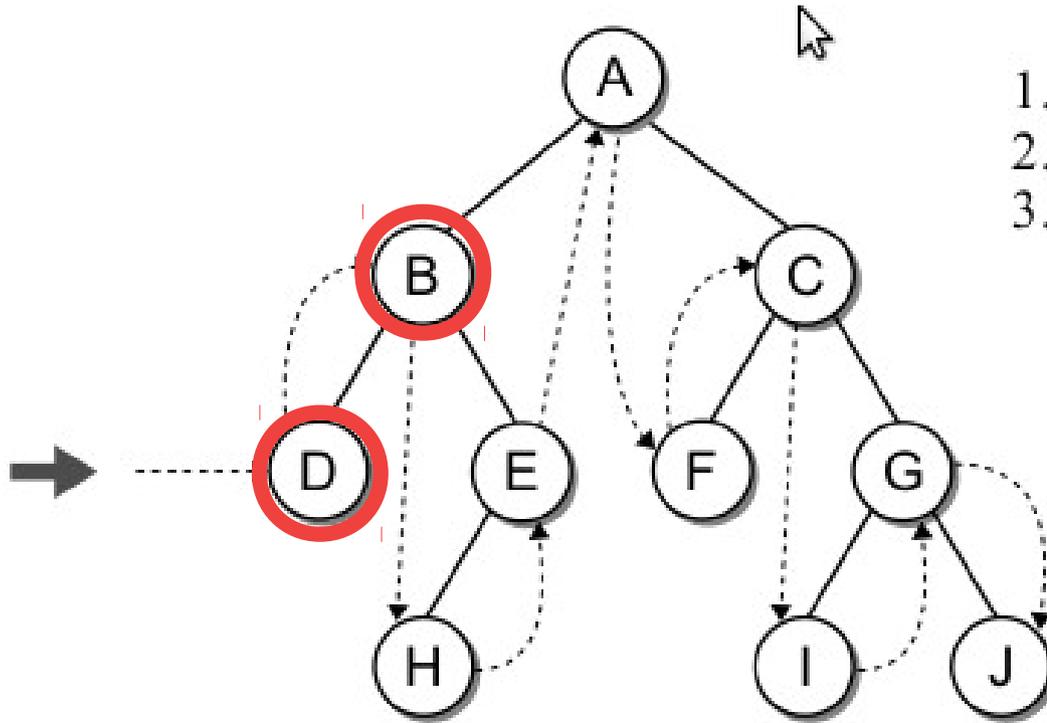
- Exemplo de caminhamento central



1. Traverse the left subtree.
2. Visit the node.
3. Traverse the right subtree.

# Árvores Binárias - Varredura

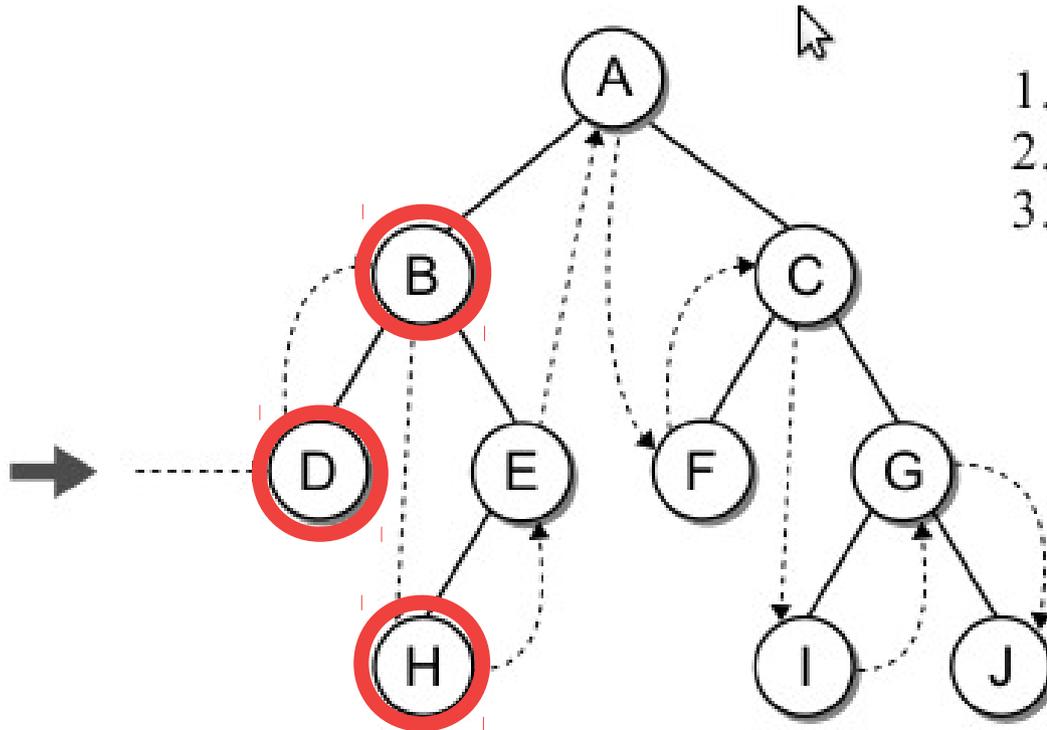
- Exemplo de caminhamento central



1. Traverse the left subtree.
2. Visit the node.
3. Traverse the right subtree.

# Árvores Binárias - Varredura

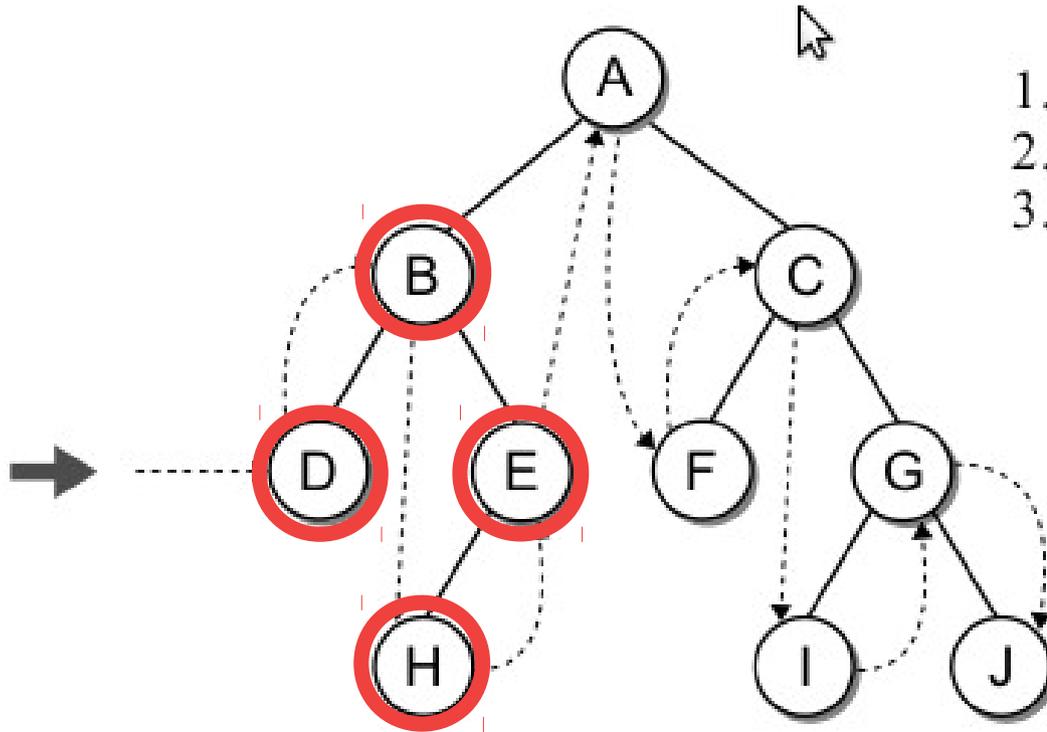
- Exemplo de caminhamento central



1. Traverse the left subtree.
2. Visit the node.
3. Traverse the right subtree.

# Árvores Binárias - Varredura

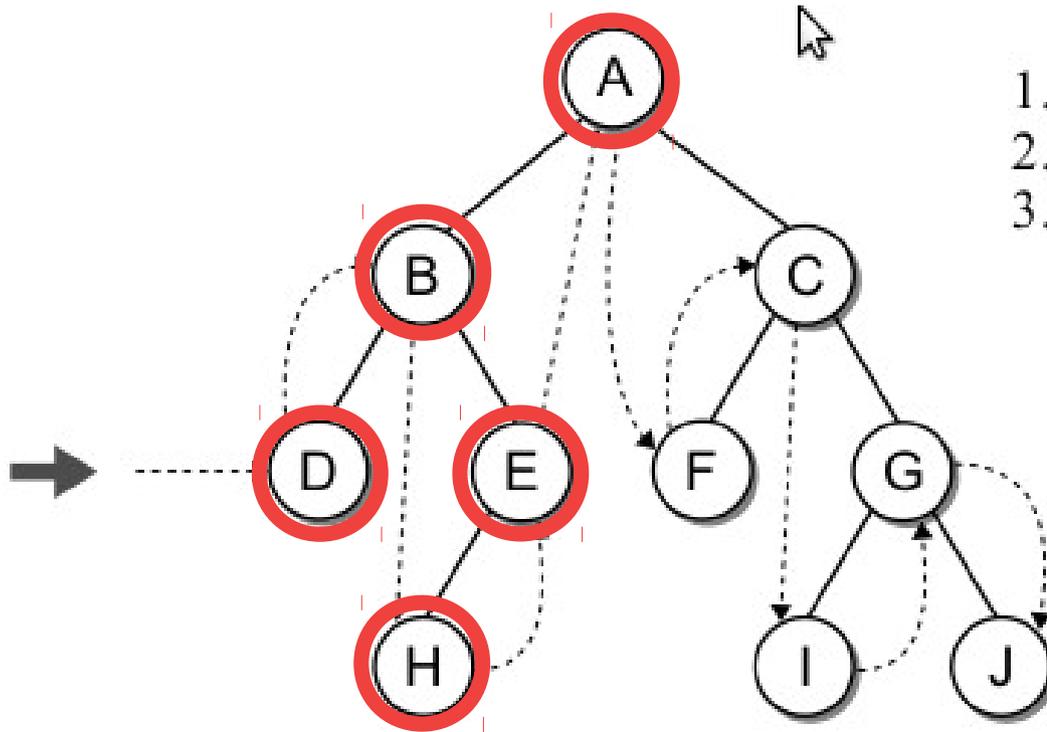
- Exemplo de caminhamento central



1. Traverse the left subtree.
2. Visit the node.
3. Traverse the right subtree.

# Árvores Binárias - Varredura

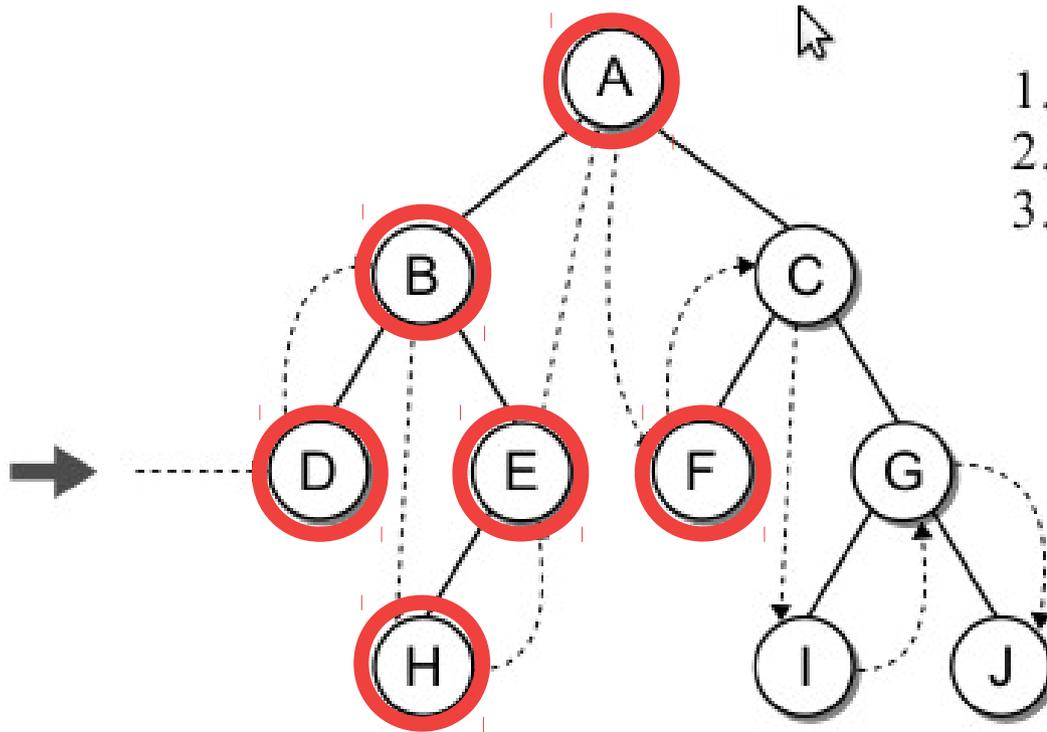
- Exemplo de caminhamento central



1. Traverse the left subtree.
2. Visit the node.
3. Traverse the right subtree.

# Árvores Binárias - Varredura

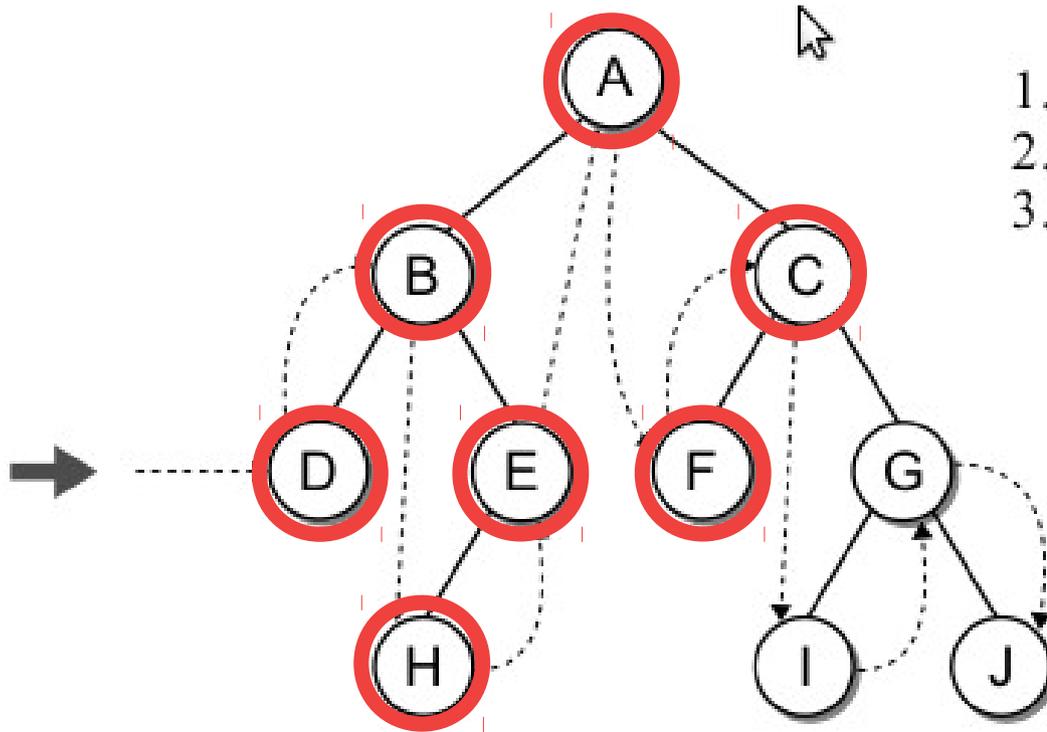
- Exemplo de caminhamento central



1. Traverse the left subtree.
2. Visit the node.
3. Traverse the right subtree.

# Árvores Binárias - Varredura

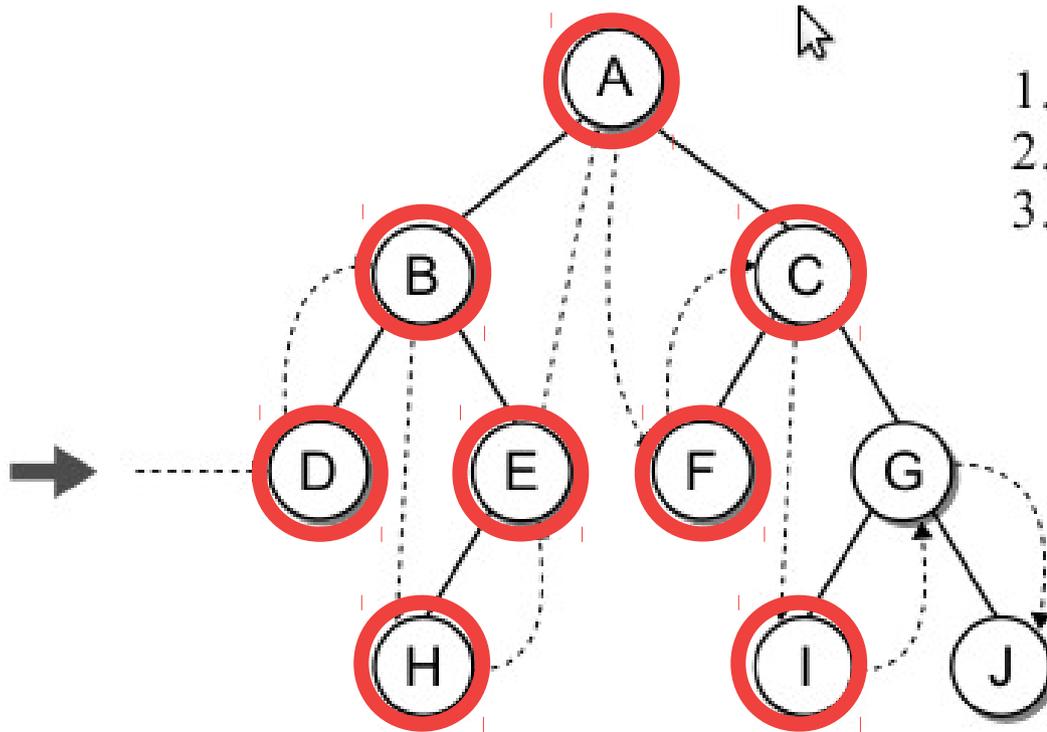
- Exemplo de caminhamento central



1. Traverse the left subtree.
2. Visit the node.
3. Traverse the right subtree.

# Árvores Binárias - Varredura

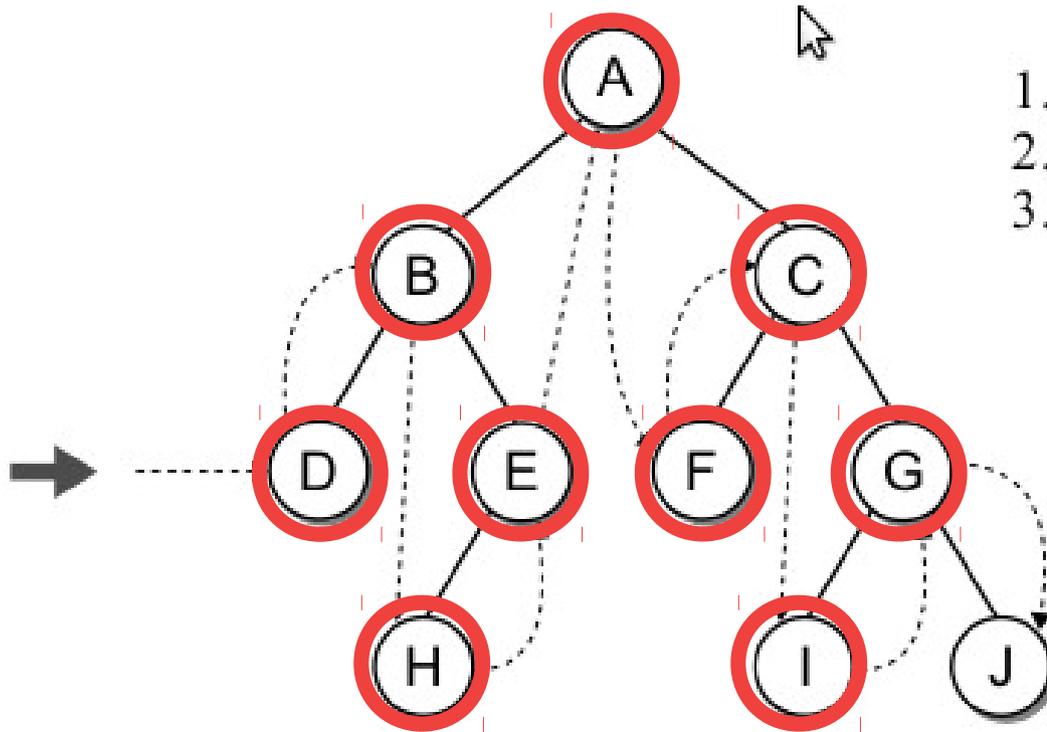
- Exemplo de caminhamento central



1. Traverse the left subtree.
2. Visit the node.
3. Traverse the right subtree.

# Árvores Binárias - Varredura

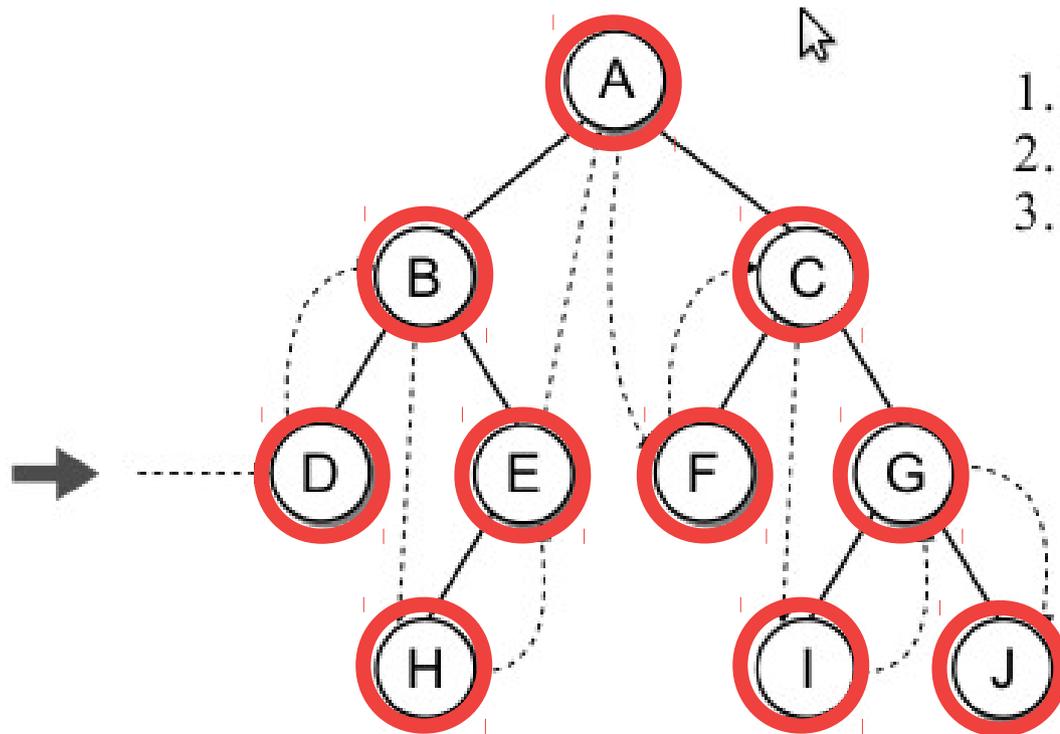
- Exemplo de caminhamento central



1. Traverse the left subtree.
2. Visit the node.
3. Traverse the right subtree.

# Árvores Binárias - Varredura

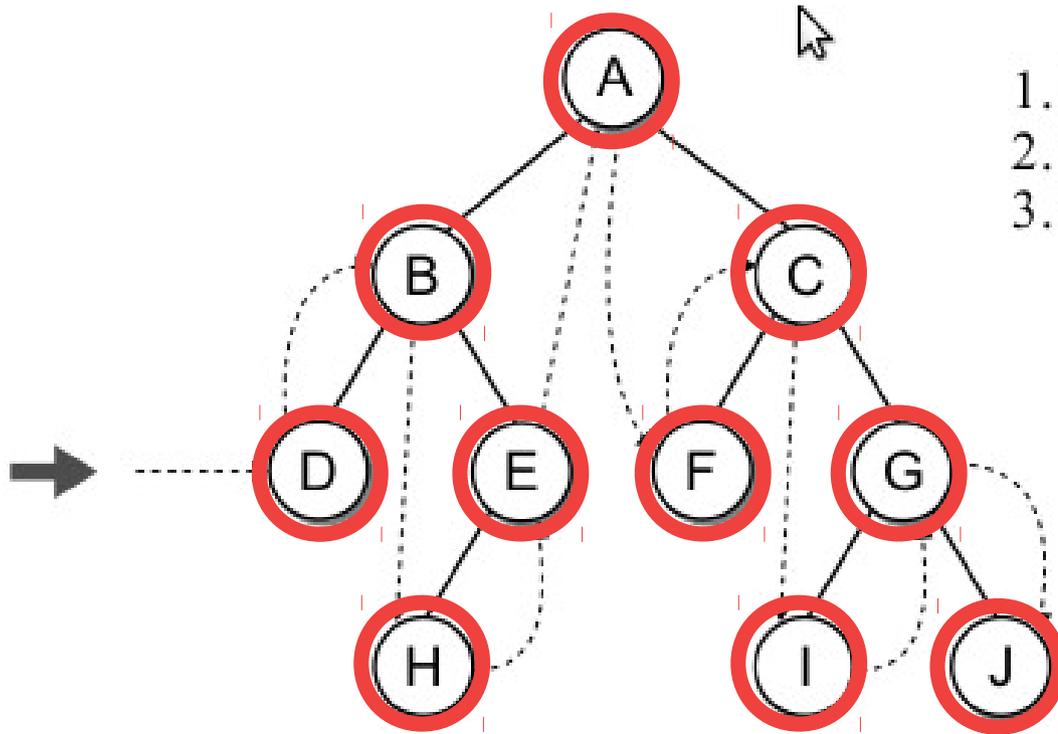
- Exemplo de caminhamento central



1. Traverse the left subtree.
2. Visit the node.
3. Traverse the right subtree.

# Árvores Binárias - Varredura

- Exemplo de caminhamento central



1. Traverse the left subtree.
2. Visit the node.
3. Traverse the right subtree.

D → B → H → E → A → F → C → I → G → J

# Árvores Binárias - Varredura

# Árvores Binárias - Varredura

- A função para caminho central:

# Árvores Binárias - Varredura

- A função para caminho central:
  - Função recursiva.

# Árvores Binárias - Varredura

- A função para caminho central:
  - Função recursiva.

```
1  def inorderTrav( subtree ) :  
2      if subtree is not None :  
3          inorderTrav( subtree.left )  
4          print( subtree.data )  
5          inorderTrav( subtree.right )
```

# Árvores Binárias - Varredura

- A função para caminho central:
  - Função recursiva.
  - O parâmetro será uma subárvore:

```
1  def inorderTrav( subtree ):  
2      if subtree is not None :  
3          inorderTrav( subtree.left )  
4          print( subtree.data )  
5          inorderTrav( subtree.right )
```

# Árvores Binárias - Varredura

- A função para caminho central:
  - Função recursiva.
  - O parâmetro será uma subárvore:
    - Referência null (None) ou

```
1  def inorderTrav( subtree ):  
2      if subtree is not None :  
3          inorderTrav( subtree.left )  
4          print( subtree.data )  
5          inorderTrav( subtree.right )
```

# Árvores Binárias - Varredura

- A função para caminho central:
  - Função recursiva.
  - O parâmetro será uma subárvore:
    - Referência null (None) ou
    - Referência para o nó raiz de uma subárvore.

```
1  def inorderTrav( subtree ) :
2      if subtree is not None :
3          inorderTrav( subtree.left )
4          print( subtree.data )
5          inorderTrav( subtree.right )
```

# Árvores Binárias - Varredura

# Árvores Binárias - Varredura

- Pós-fixado

# Árvores Binárias - Varredura

- Pós-fixado
  - Percorre a subárvore da esquerda.

# Árvores Binárias - Varredura

- Pós-fixado
  - Percorre a subárvore da esquerda.
  - Percorre a subárvore da direita.

# Árvores Binárias - Varredura

- Pós-fixado
  - Percorre a subárvore da esquerda.
  - Percorre a subárvore da direita.
  - Visita a raiz.

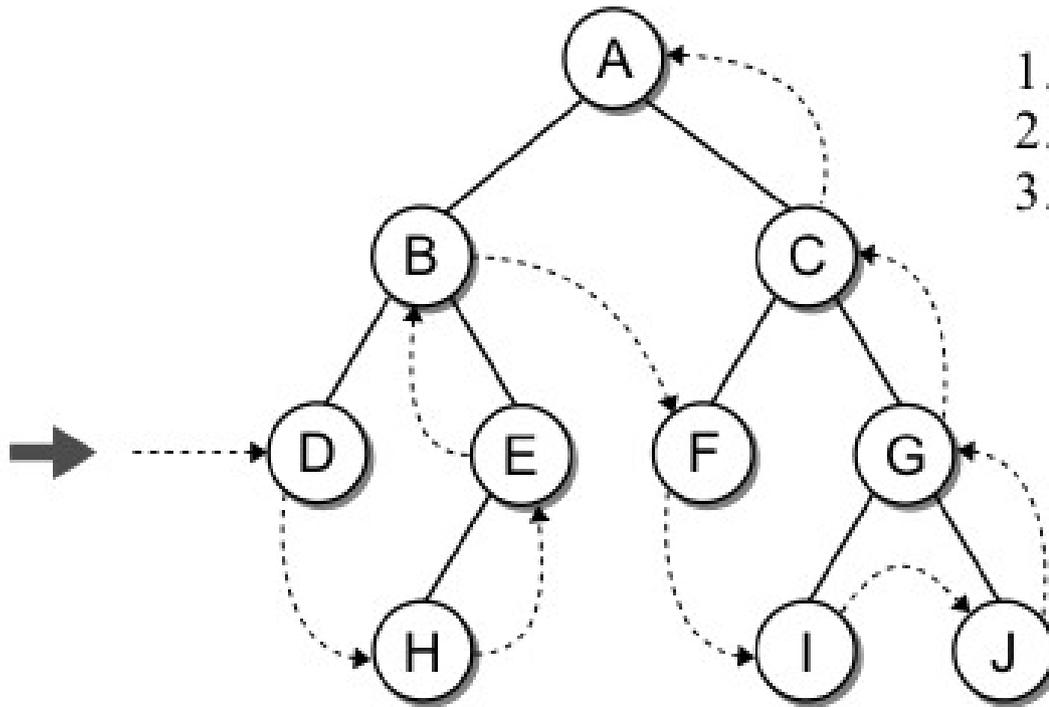
# Árvores Binárias - Varredura

# Árvores Binárias - Varredura

- Exemplo de caminhamento pós-fixado

# Árvores Binárias - Varredura

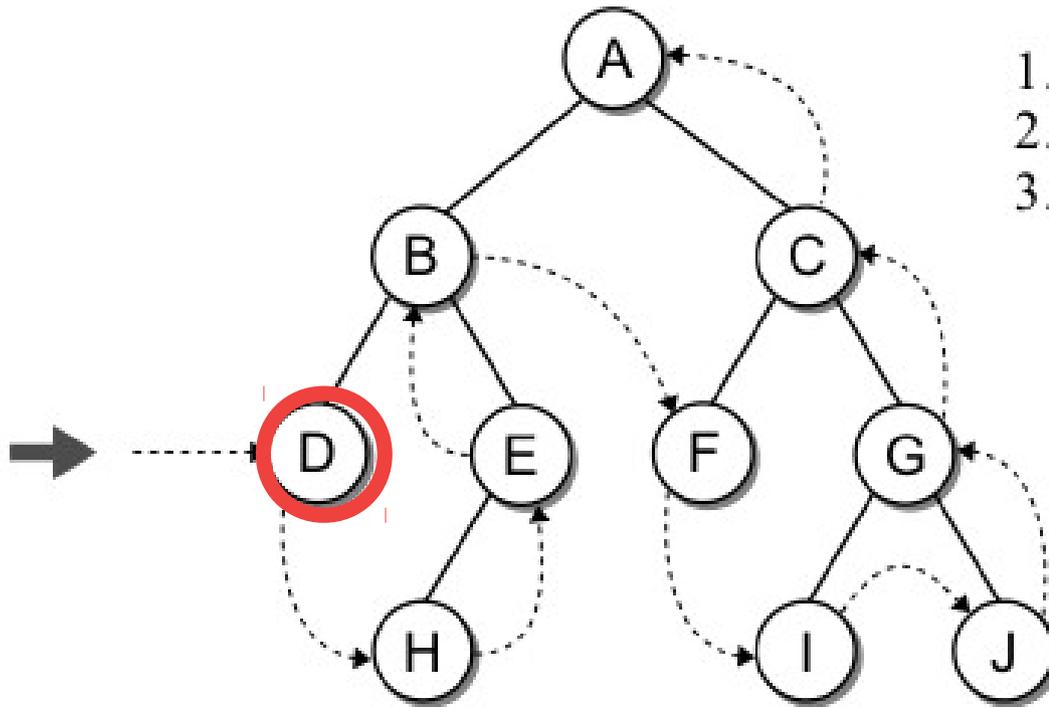
- Exemplo de caminhamento pós-fixado



1. Traverse the left subtree.
2. Traverse the right subtree.
3. Visit the node.

# Árvores Binárias - Varredura

- Exemplo de caminhamento pós-fixado

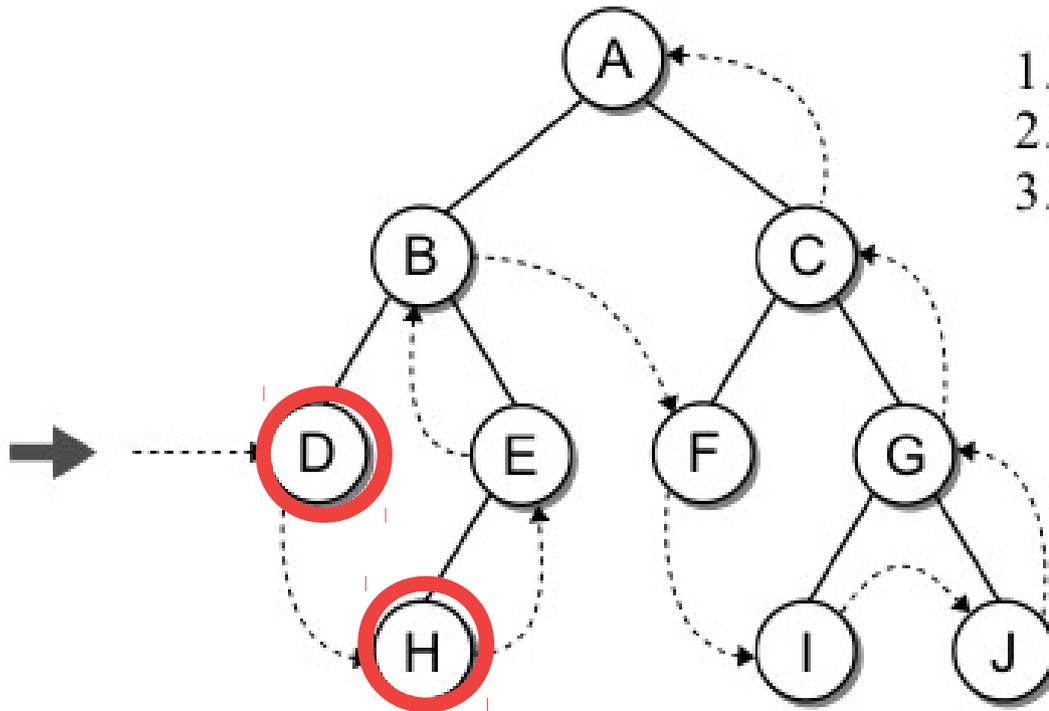


1. Traverse the left subtree.
2. Traverse the right subtree.
3. Visit the node.



# Árvores Binárias - Varredura

- Exemplo de caminhamento pós-fixado

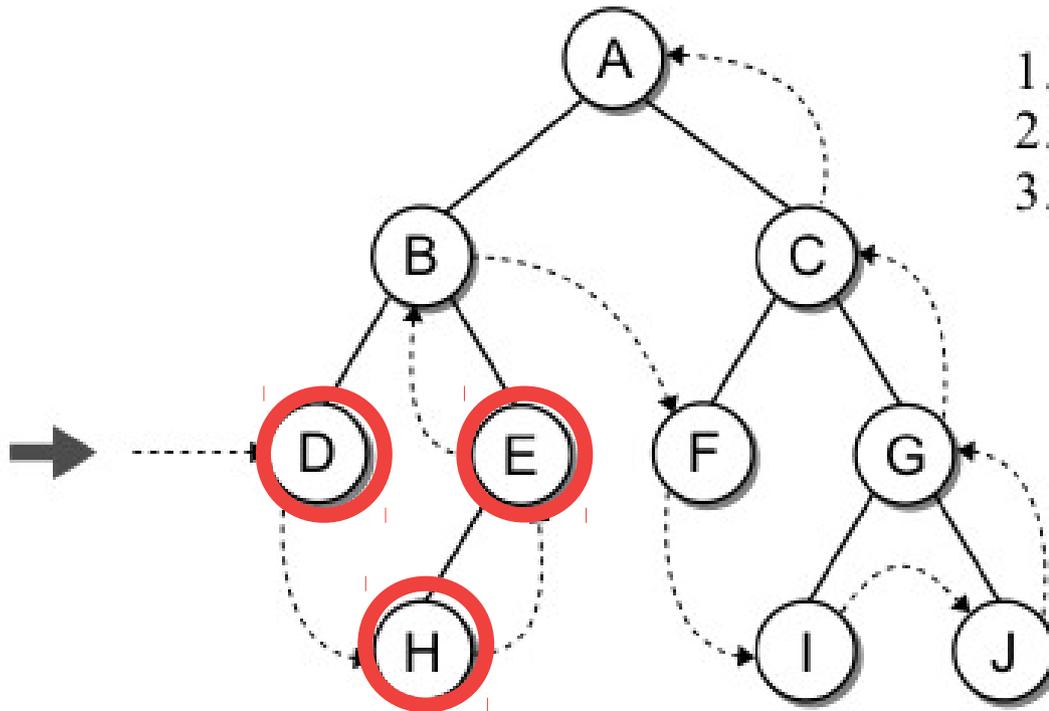


1. Traverse the left subtree.
2. Traverse the right subtree.
3. Visit the node.



# Árvores Binárias - Varredura

- Exemplo de caminhamento pós-fixado

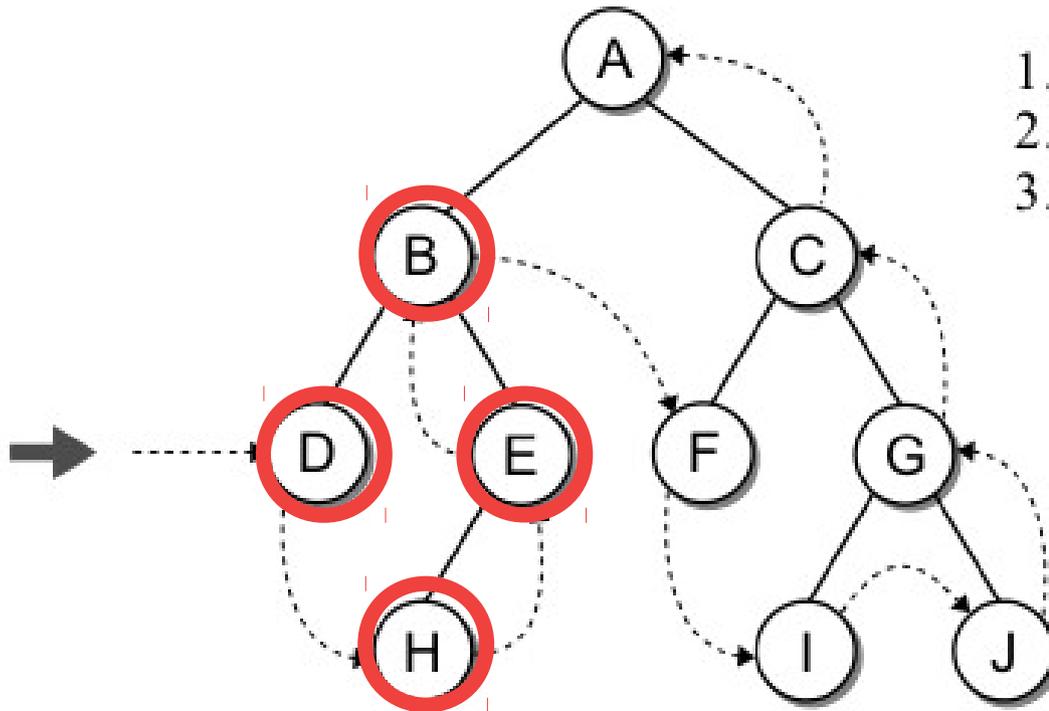


1. Traverse the left subtree.
2. Traverse the right subtree.
3. Visit the node.



# Árvores Binárias - Varredura

- Exemplo de caminhamento pós-fixado

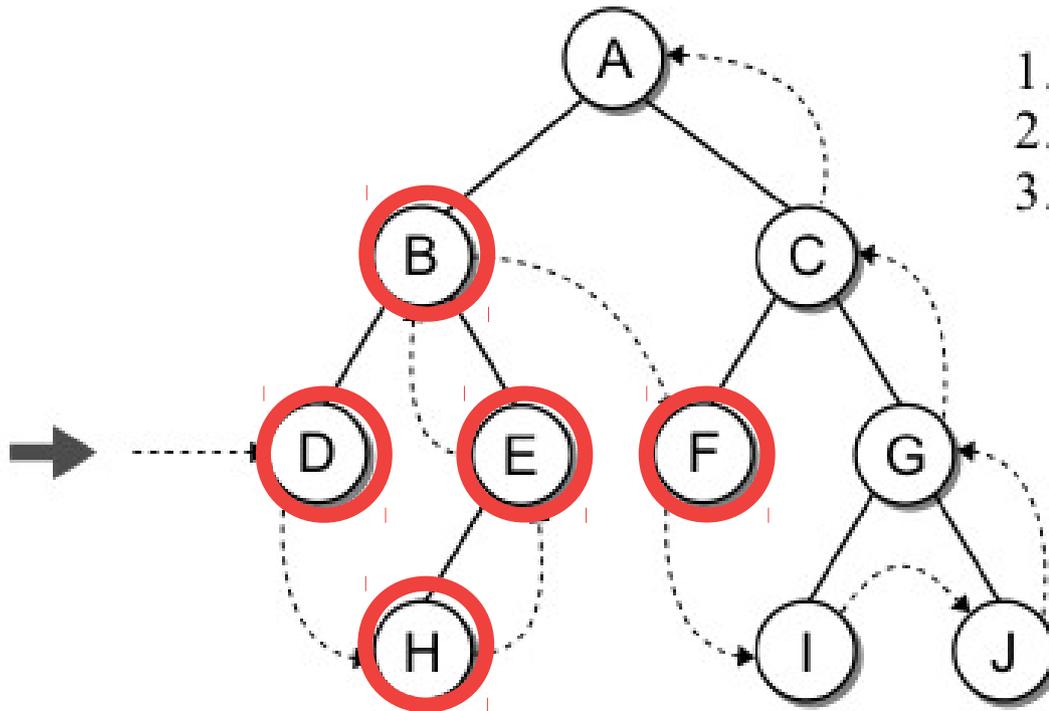


1. Traverse the left subtree.
2. Traverse the right subtree.
3. Visit the node.



# Árvores Binárias - Varredura

- Exemplo de caminhamento pós-fixado

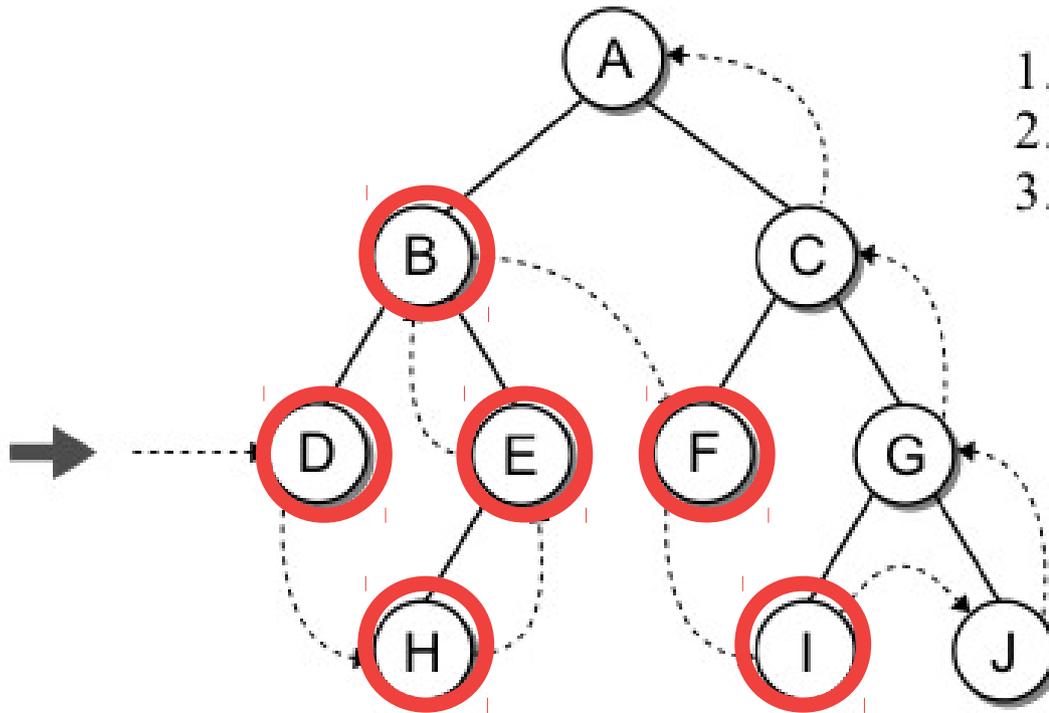


1. Traverse the left subtree.
2. Traverse the right subtree.
3. Visit the node.



# Árvores Binárias - Varredura

- Exemplo de caminhamento pós-fixado

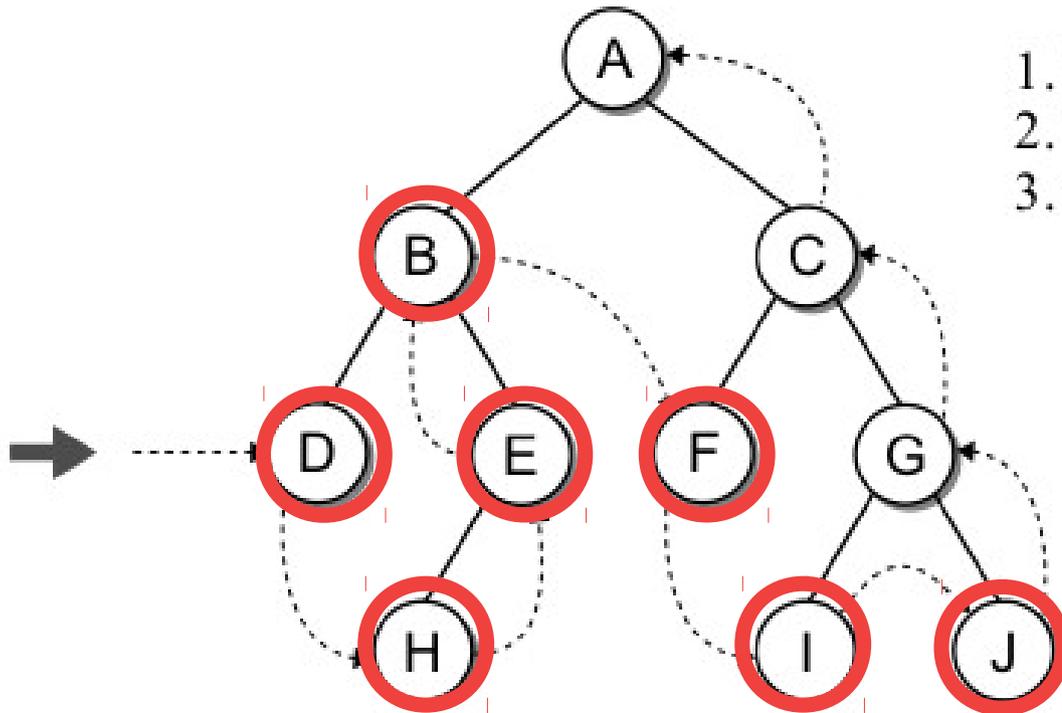


1. Traverse the left subtree.
2. Traverse the right subtree.
3. Visit the node.



# Árvores Binárias - Varredura

- Exemplo de caminhamento pós-fixado

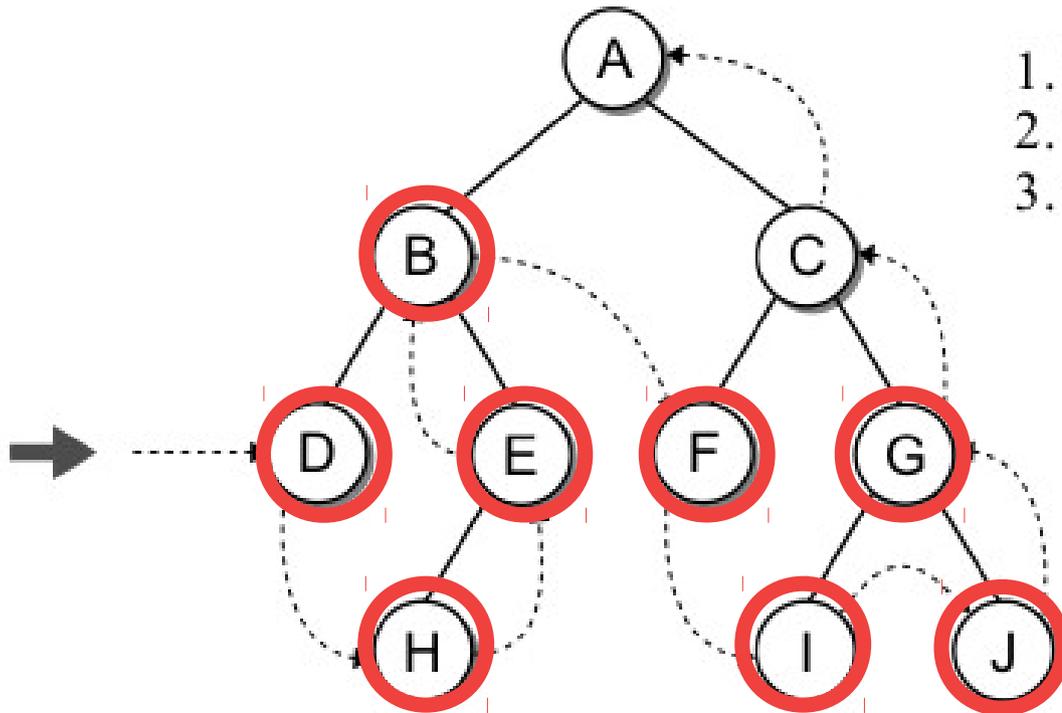


1. Traverse the left subtree.
2. Traverse the right subtree.
3. Visit the node.



# Árvores Binárias - Varredura

- Exemplo de caminhamento pós-fixado

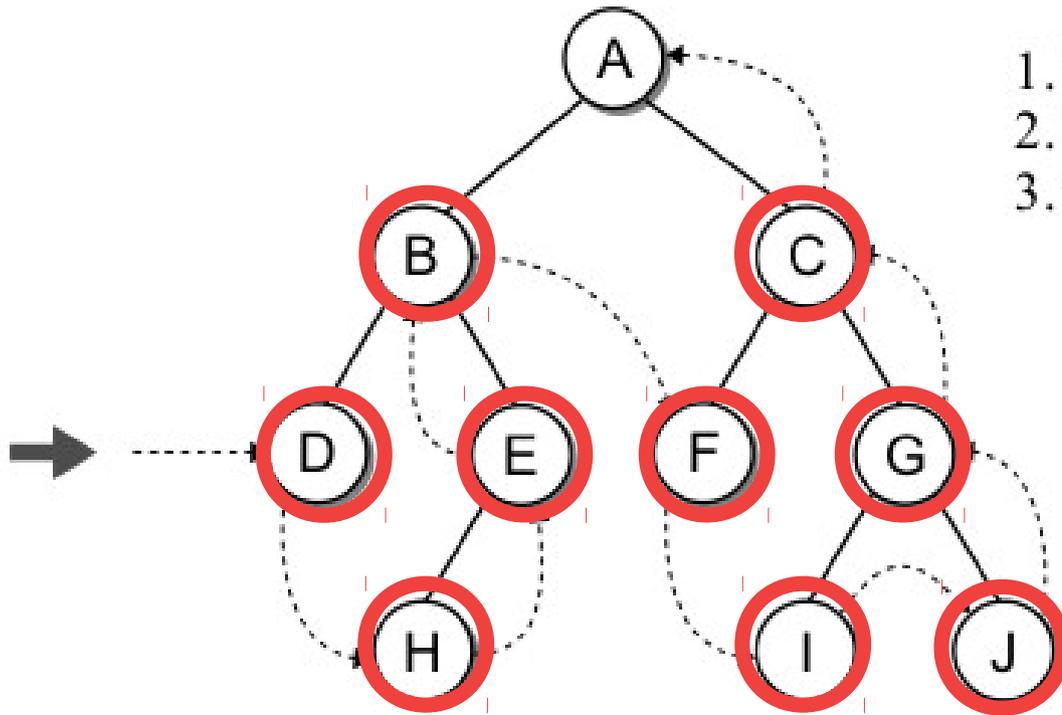


1. Traverse the left subtree.
2. Traverse the right subtree.
3. Visit the node.



# Árvores Binárias - Varredura

- Exemplo de caminhamento pós-fixado

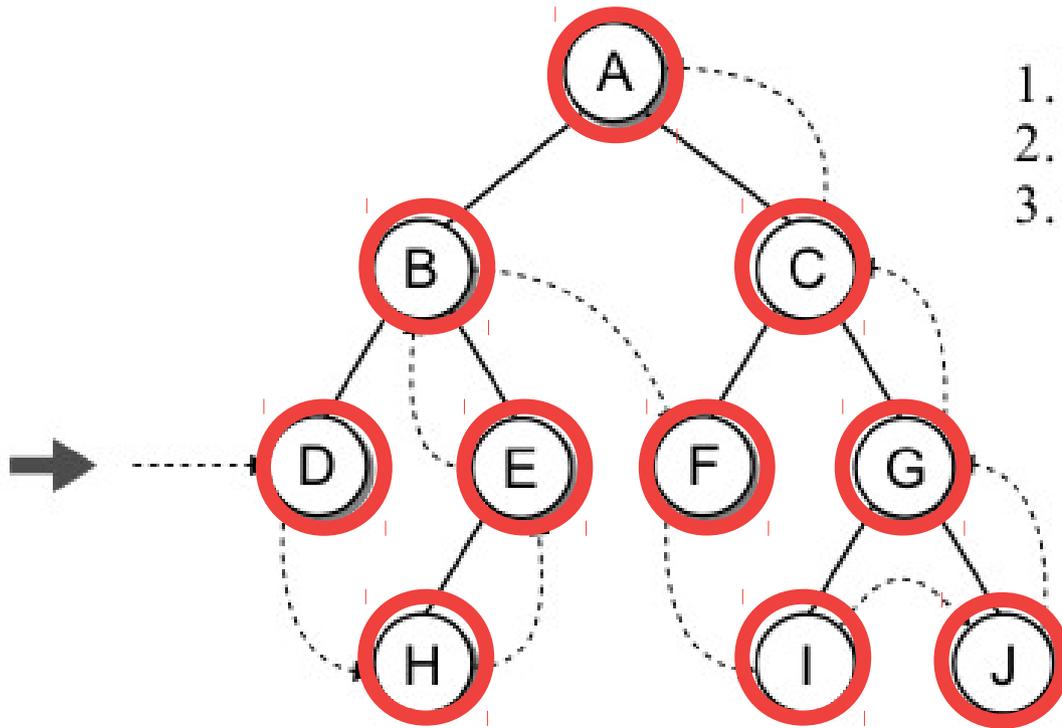


1. Traverse the left subtree.
2. Traverse the right subtree.
3. Visit the node.



# Árvores Binárias - Varredura

- Exemplo de caminhamento pós-fixado

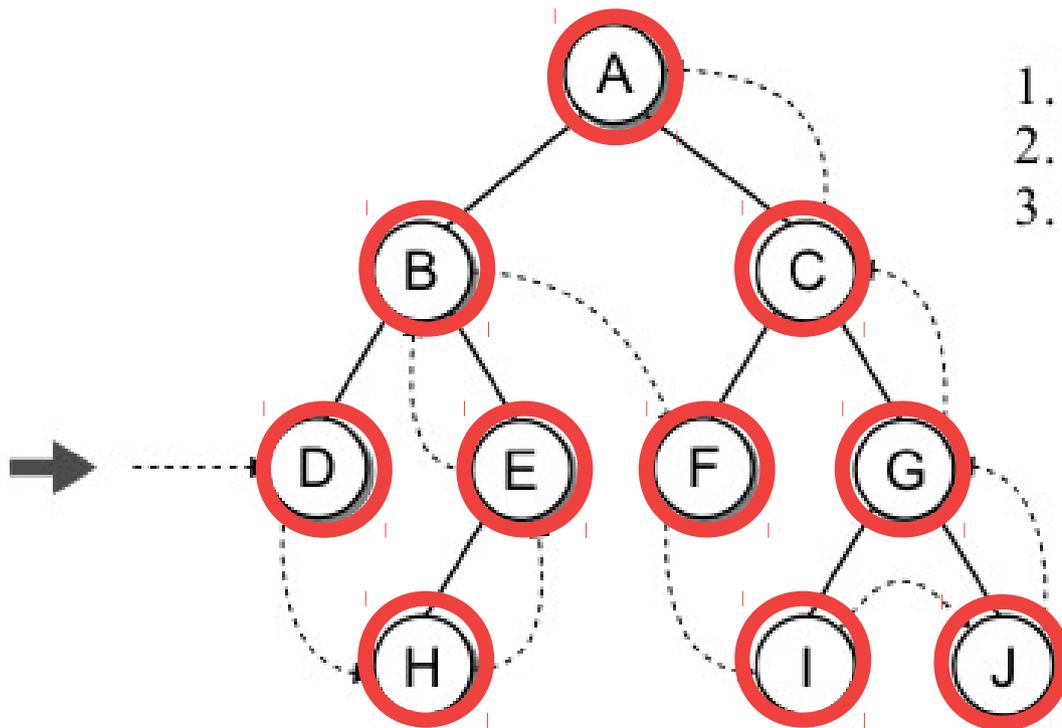


1. Traverse the left subtree.
2. Traverse the right subtree.
3. Visit the node.



# Árvores Binárias - Varredura

- Exemplo de caminhamento pós-fixado



1. Traverse the left subtree.
2. Traverse the right subtree.
3. Visit the node.

D → H → E → B → F → I → J → G → C → A

# Árvores Binárias - Varredura

# Árvores Binárias - Varredura

- A função para caminho pós-fixado:

# Árvores Binárias - Varredura

- A função para caminho pós-fixado:
  - Função recursiva.

# Árvores Binárias - Varredura

- A função para caminho pós-fixado:
  - Função recursiva.

```
1  def postorderTrav( subtree ):
2      if subtree is not None :
3          postorderTrav( subtree.left )
4          postorderTrav( subtree.right )
5          print( subtree.data )
```

# Árvores Binárias - Varredura

- A função para caminho pós-fixado:
  - Função recursiva.
  - O parâmetro será uma subárvore:

```
1  def postorderTrav( subtree ):
2      if subtree is not None :
3          postorderTrav( subtree.left )
4          postorderTrav( subtree.right )
5          print( subtree.data )
```

# Árvores Binárias - Varredura

- A função para caminho pós-fixado:
  - Função recursiva.
  - O parâmetro será uma subárvore:
    - Referência null (None) ou

```
1  def postorderTrav( subtree ):
2      if subtree is not None :
3          postorderTrav( subtree.left )
4          postorderTrav( subtree.right )
5          print( subtree.data )
```

# Árvores Binárias - Varredura

- A função para caminho pós-fixado:
  - Função recursiva.
  - O parâmetro será uma subárvore:
    - Referência null (None) ou
    - Referência para o nó raiz de uma subárvore.

```
1 def postorderTrav( subtree ):
2     if subtree is not None :
3         postorderTrav( subtree.left )
4         postorderTrav( subtree.right )
5     print( subtree.data )
```

# Árvores Binárias - Varredura

- A função para caminho pós-fixado:
  - Função recursiva.
  - O parâmetro será uma subárvore:
    - Referência null (None) ou
    - Referência para o nó raiz de uma subárvore.
  - O nó raiz é sempre visitado por último.

```
1 def postorderTrav( subtree ):  
2     if subtree is not None :  
3         postorderTrav( subtree.left )  
4         postorderTrav( subtree.right )  
5         print( subtree.data )
```

# Caminhamento em Largura

# Caminhamento em Largura

- Os caminhamentos pré-fixado, central e pós-fixado são exemplos de busca em profundidade.

# Caminhamento em Largura

- Os caminhamentos pré-fixado, central e pós-fixado são exemplos de busca em profundidade.
- Outra forma de pesquisa é o **caminhamento em largura** (*breadth-first*).

# Caminhamento em Largura

# Caminhamento em Largura

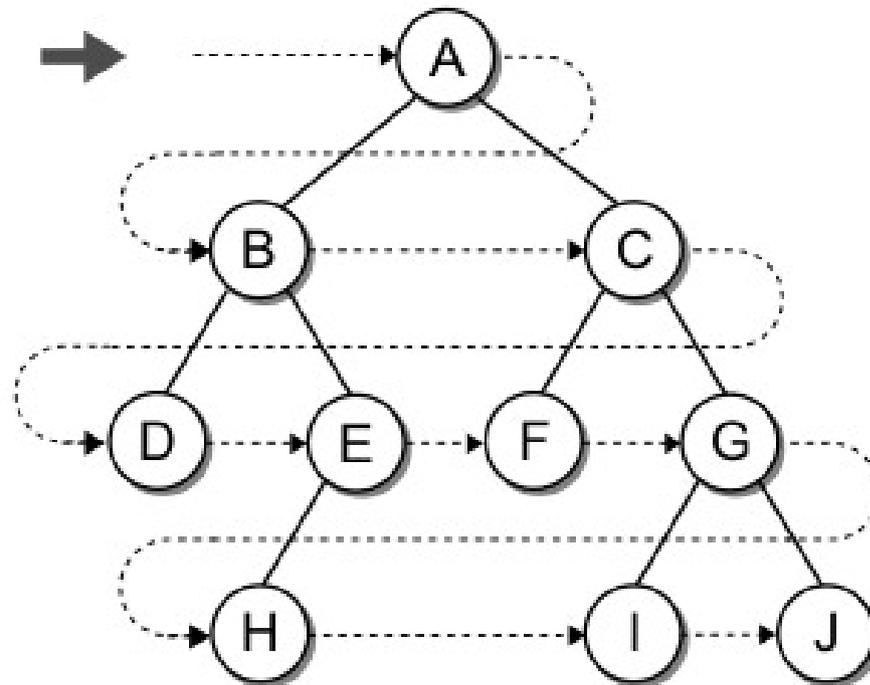
- No caminhamento em largura, os nós são visitados por nível, da esquerda para a direita.

# Caminhamento em Largura

- No caminhamento em largura, os nós são visitados por nível, da esquerda para a direita.
- Exemplo:

# Caminhamento em Largura

- No caminhamento em largura, os nós são visitados por nível, da esquerda para a direita.
- Exemplo:



# Caminhamento em Largura

# Caminhamento em Largura

- Não é possível usar funções recursivas nesse tipo de caminhamento.

# Caminhamento em Largura

- Não é possível usar funções recursivas nesse tipo de caminhamento.
- Uma alternativa é usar uma estrutura do tipo Fila.

# Caminhamento em Largura

- Não é possível usar funções recursivas nesse tipo de caminhamento.
- Uma alternativa é usar uma estrutura do tipo Fila.
- Algoritmo:

# Caminhamento em Largura

- Não é possível usar funções recursivas nesse tipo de caminhamento.
- Uma alternativa é usar uma estrutura do tipo Fila.
- Algoritmo:
  - Inicia pelo nó raiz.

# Caminhamento em Largura

- Não é possível usar funções recursivas nesse tipo de caminhamento.
- Uma alternativa é usar uma estrutura do tipo Fila.
- Algoritmo:
  - Inicia pelo nó raiz.
  - Durante cada iteração, nós removemos o nó da fila, visitamos o nó, e então adicionamos os seus filhos à fila.

# Caminhamento em Largura

- Não é possível usar funções recursivas nesse tipo de caminhamento.
- Uma alternativa é usar uma estrutura do tipo Fila.
- Algoritmo:
  - Inicia pelo nó raiz.
  - Durante cada iteração, nós removemos o nó da fila, visitamos o nó, e então adicionamos os seus filhos à fila.
  - O algoritmo termina quando todos os nós tiverem sido visitados.

# Caminhamento em Largura

# Caminhamento em Largura

- Exemplo de código:

# Caminhamento em Largura

- Exemplo de código:

```
1  def breadthFirstTrav( bintree ):  
2      # Create a queue and add the root node to it.  
3      Queue q  
4      q.enqueue( bintree )  
5  
6      # Visit each node in the tree.  
7      while not q.isEmpty() :  
8          # Remove the next node from the queue and visit it.  
9          node = q.dequeue()  
10         print( node.data )  
11  
12         # Add the two children to the queue.  
13         if node.left is not None :  
14             q.enqueue( node.left )  
15         if node.right is not None :  
16             q.enqueue( node.right )
```

# Para pensar...

Como seria possível representar uma expressão tal como  $(9 + 3) * (8 - 4)$  em uma árvore binária?

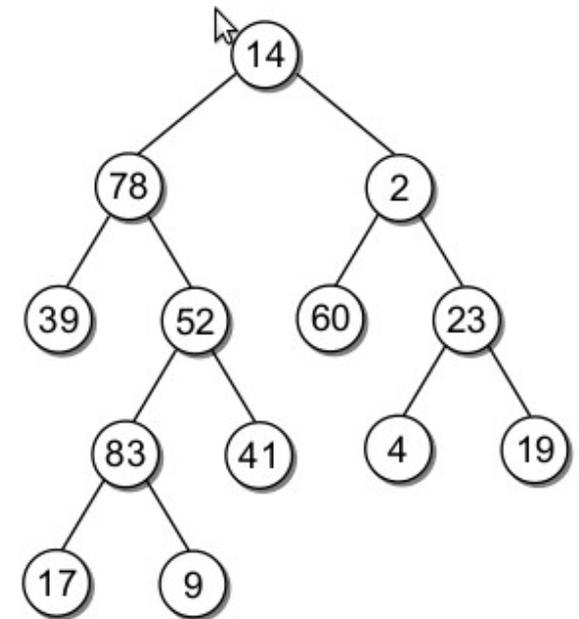


# Exercícios

- Dada uma árvore binária de tamanho 76, qual é o número mínimo de níveis que ela pode ter? E qual seria o número máximo de níveis?
- Qual é o número máximo de nós de uma árvore binária de 5 níveis?
- Qual é a altura mínima de uma árvore binária com 31 nós? E a altura máxima?
- Considere uma árvore binária de 15 nós. Qual é o número máximo de folhas que essa árvore pode ter?
- Uma árvore binária de altura 4 pode ter no máximo quantos nós? Justifique sua resposta.

# Exercícios

- Considere a árvore abaixo e faça o que se pede:
  - Mostre a ordem em que os nós seria visitados para cada caminhamento visto em sala de aula.
  - Identifique os nós folhas.
  - Identifique os nós internos.
  - Liste todos os nós do nível 4.
  - Identifique a profundidade do nó 2.
  - Quem são as ascendentes do nó 4?

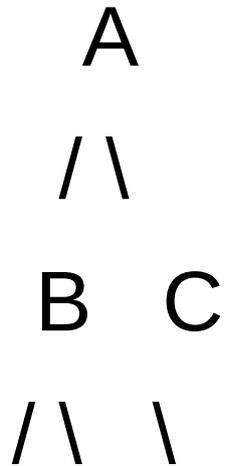


# Exercícios

- Considere as árvores binárias abaixo e responda ao que se pede:
  - Elas são balanceadas?
  - Elas são perfeitamente balanceadas?
  - Liste os nós conforme os vários tipos de caminhamento vistos em sala de aula.
    - a) (1 (2 (4) (5)) (3 (6) (7)));
    - b) (A (B (D (F)) (E)) (C (G (H))));

# Exercícios

- Dado a árvore a seguir, liste a ordem dos nós visitados em cada um dos percursos estudados:



# Exercícios

- Implemente a função `treeSize(root)` para calcular o número de nós de uma árvore binária.
- Implemente a função `treeHeight(root)` para calcular a altura de uma árvore binária.