Universidade Estadual do Piauí – UESPI
Campus de Parnaíba
Bacharelado em Ciência da Computação

Estrutura de Dados

Aula 10 – Árvore Binária

Prof. Francisco Rocha

Árvore Binária

- **Árvore: motivação**

– A importância de estruturas lineares (vetores, listas, pilhas, filas, heap e deques) é inegável.

– Contudo, elas não são mais conveniente para representar dados que devem ser dispostos de maneira **hierárquica**.

– Por exemplo:
  - **diretórios** criados em um computador;
  - **sumário** de livros;
  - **pertinência** entre objetos;
  - **estrutura hierárquica** de uma empresa.

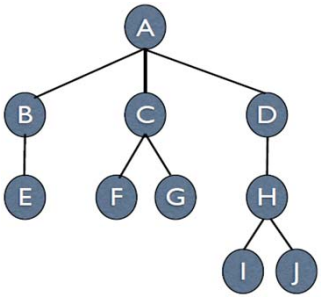
Árvore Binária

- **Árvore: motivação**

– Os relacionamentos lógicos entre os dados representam alguma dependência de hierarquia ou composição entre os nós.
  - Existe uma hierarquia de subordinação.

– Relacionamentos de subordinação, formando hierarquias, podem apresentar diferentes significados:

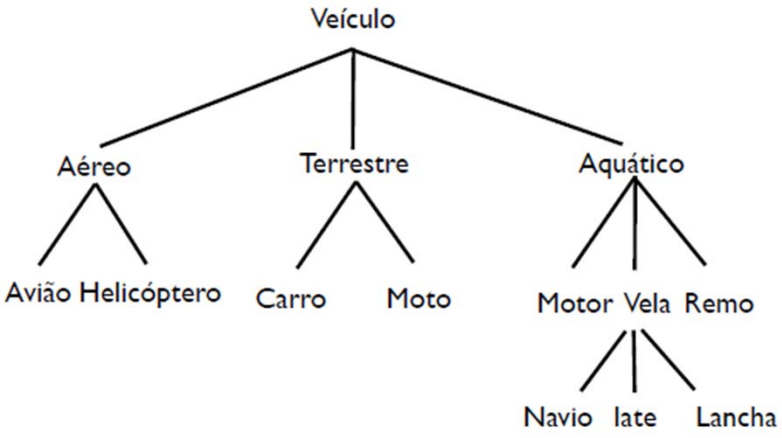
  - Hierarquias de especialização: representa classes e subclasses;
  - Hierarquias de composição: o nó é composto por partes;
  - Hierarquias de dependência: representa um organograma.



Árvore Binária

- **Árvore: motivação**

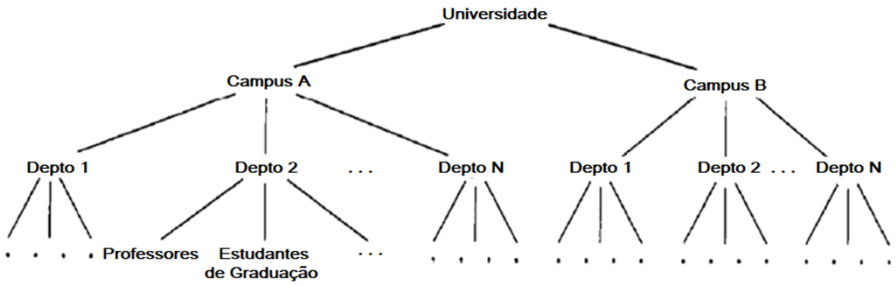
– Modelo de hierarquia:



Árvore Binária

- **Árvore: motivação**

– Estrutura hierárquica de uma universidade mostrada como uma árvore.

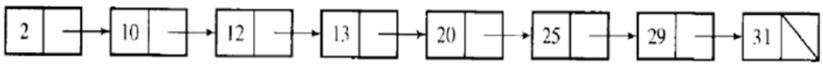


Árvore Binária

- **Árvore: motivação**

– Transformando uma lista ligada em uma árvore.

  - Existem várias exemplos de árvores para a lista citada. Tente criar outra árvore a partir da mesma lista.

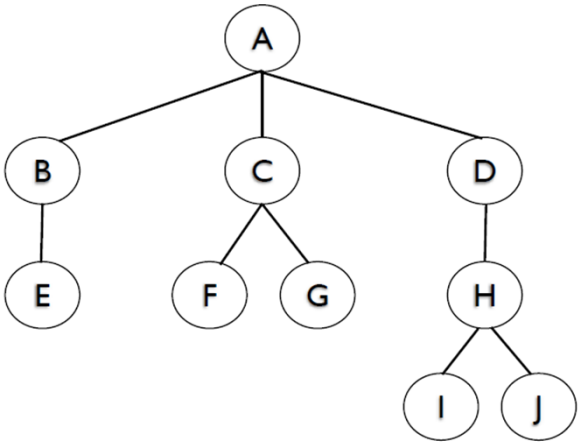


– Árvore gerada:

Árvore Binária

- **Árvore: representação**

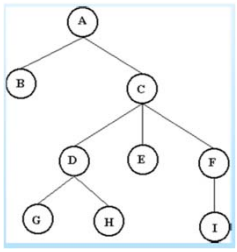
– Representação gráfica de uma árvore. Mais existem outras.

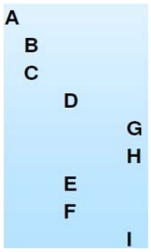


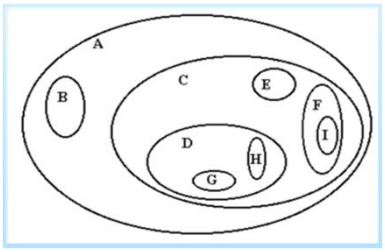
Árvore Binária

- **Árvore: representação**

– Exemplo de representação de árvore: hierárquica; alinhamento de nós; diagrama de inclusão e parênteses aninhados.







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

Árvore Binária

- **Árvore: definição**

– **Árvore** é uma estrutura de dados adequada para representar **hierarquias**.

– A forma mais natural de definirmos uma estrutura de árvore é usando **recursividade**.

1. Uma estrutura vazia é uma árvore vazia;
2. Se t_1, ..., t_k são árvores disjuntas, então a estrutura cuja raiz tem como suas filhas as raízes de t_1, ..., t_k também é uma árvore.
3. Somente estruturas geradas pelas regras 1 e 2 são árvores.

Árvore Binária

- **Árvore: terminologia**

– Uma árvore é composta por um conjunto finito de nós.

– O primeiro nó da árvore (o mais alto) é chamado de **raiz**.

– Um nó contém (abaixo) zero, uma ou mais sub-árvores.

– Um nó y abaixo de um nó x é chamado de **filho** de x.

- Neste caso, x é **pai** de y.
- Nós com o mesmo pai são **irmãos**.

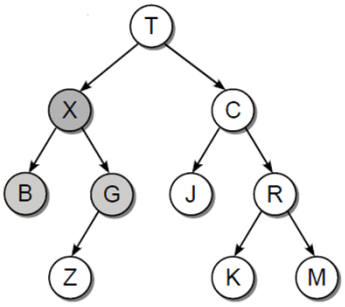
– Os nós que tem filhos são chamados de nós **internos**.

– Os nós que não têm filhos são chamados de nós **externos** ou **folhas**.

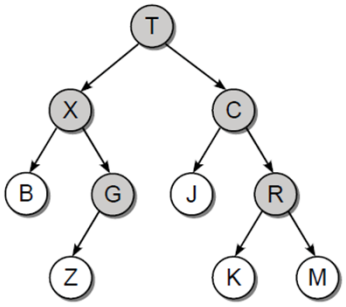
Árvore Binária

- **Árvore: terminologia**

– Exemplo de uma árvore com raiz em T:



O nó X é pai de B e G. Os nós B e G são irnãos.
Os nós B e G são filhos de X.

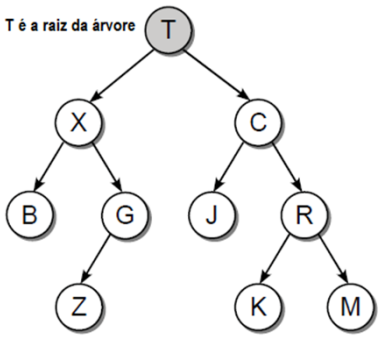


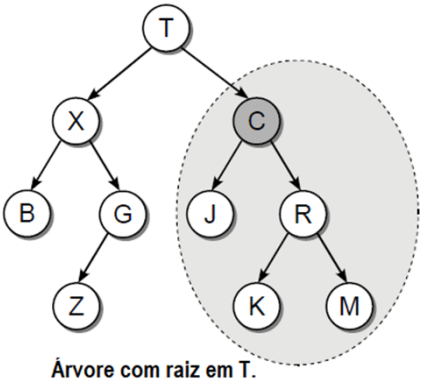
Os nós cinza são internos.
Os nós brancos são folhas.

Árvore Binária

- **Árvore: terminologia**

– Uma árvore é por definição uma estrutura recursiva. Cada nó pode ser a raiz de sua própria **subárvore**, que consiste em um subconjunto de nós e arestas da árvore maior.





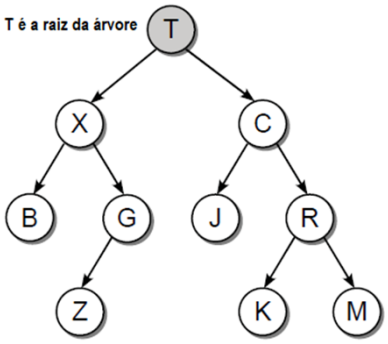
Árvore com raiz em T.
Subárvore com raiz em C.

Árvore Binária

- **Árvore: terminologia**

– O **grau de um nó** é o número de filhos de um nó.
  - O grau do nó B é 0, do nó G é 1, do nó C é 2.

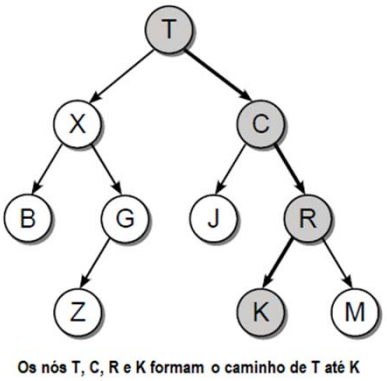
– O **grau de uma árvore** é o maior valor dentre os graus de todos os seus nós.
  - O grau da árvore é 2.



Árvore Binária

- **Árvore: terminologia**

– Todo nó deve que ser atingível a partir da raiz através de uma sequência **única** de arestas, chamado **caminho**.
  - Só existe um único caminho da raiz até qualquer nó.
  - O número de arestas em um caminho é chamado de **comprimento do caminho**.

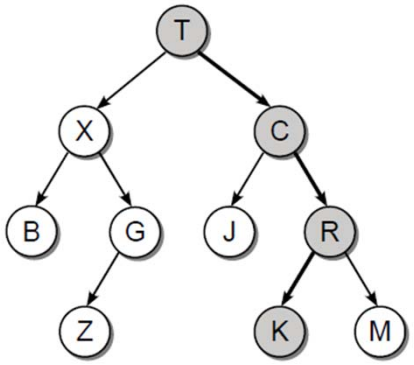


Árvore Binária

- **Árvore: terminologia**

– **Altura** da árvore é o comprimento do **caminho mais longo** da raiz até uma das folhas.

- Por definição, a altura de uma árvore de um único nó é zero.
- Qual a altura do nó raiz de uma árvore?



Árvore Binária

- **Árvore: terminologia**

– O **nível do nó** é o comprimento do caminho da raiz até este nó.

- A raiz está no nível 0, seus filhos não vazios estão no nível 1, os filhos não vazios destes estão no nível 2 e assim por diante.

– Se todos os nós em todos os níveis, exceto o último, tivessem dois filhos, então haveria:

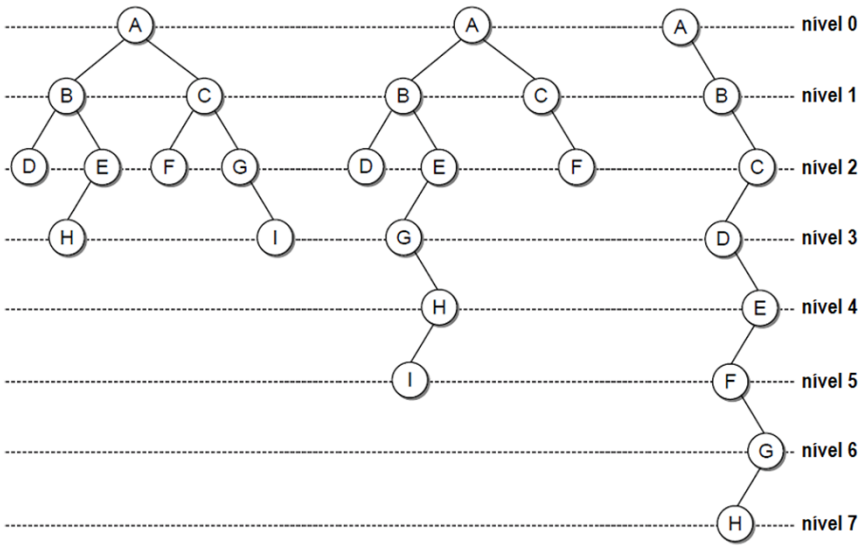
- 2^0 = 1 nó no nível 0;
- 2^1 = 2 nós no nível 1;
- 2^2 = 4 nós no nível 2,

e em uma forma geral, $\mathbf{2^i}$ nós no nível **i**.

  – Essa é a chamada **árvore binária completa**.

Árvore Binária

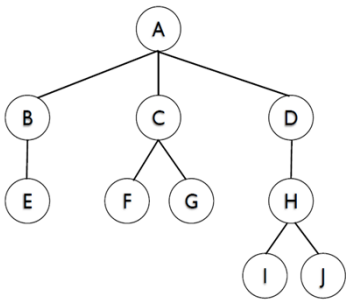
- **Árvore: terminologia**



Árvore Binária

- **Árvore: exercícios**

– Para o exemplo de árvore abaixo, responda:
  - Quantas sub-árvores existem?
  - Quais são as sub-árvores?
  - Quais nós são as raízes das sub-árvores?
  - Quais nós são considerados nós externos (folhas)?
  - Quais nós são considerados nós internos?
  - Qual o nível dos nós A, B, G, H e J ?
  - Qual a altura da árvore?



Árvore Binária

- **Árvore Binária: definição**

– Existem árvores de muitos formatos, podendo variar no número de filhos, dependendo do contexto da aplicação.

– Uma das árvores muito utilizada é a **árvore binária**.

– Uma árvore binária é uma árvore em que cada nó pode ter no máximo **dois filhos**.

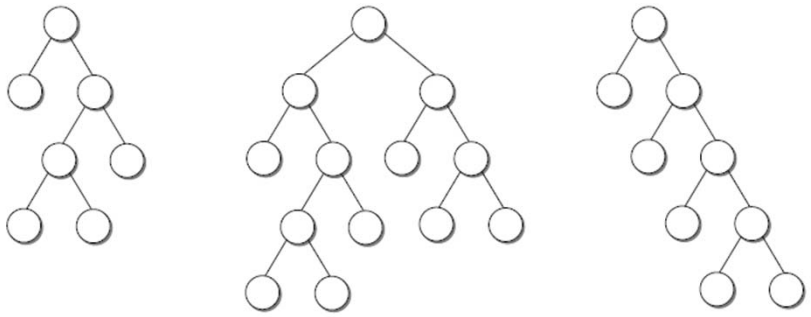
– Um dos dois filhos é identificado como **filho da esquerda** e o outro como **filho da direita**.

– A terminologia aplicada as árvores também é aplicada as árvores binárias.

Árvore Binária

- **Árvore Binária: terminologia**

– Uma **árvore binária completa** é uma árvore binária em que cada nó interior contém dois filhos.

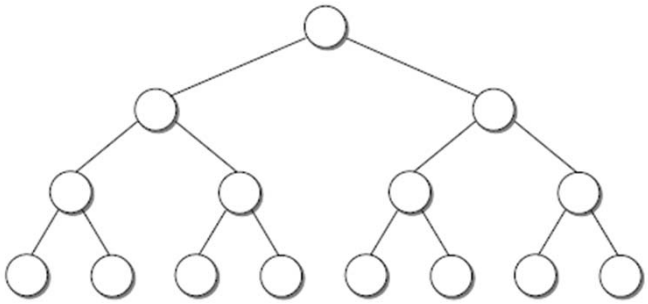


Árvore Binária

- **Árvore Binária: terminologia**

– Uma **árvore binária perfeita** é uma árvore binária completa na qual todos os nós folhas estão no mesmo nível.

– A árvore perfeita tem todas as posições possíveis para nós preenchidos de cima para baixo, sem lacunas.



Árvore Binária

- **Árvore Binária: operações**

– As principais operações sobre as árvores são:

- Criar a árvore e iniciá-la como vazia;
- Verificar se a árvore está vazia;
- Inserir um nó na árvore;
- Remover um nó (valor) da árvore;
- Informar a altura da árvore;
- Informar o nível de um nó;
- Pesquisar a ocorrência de um nó (valor) na árvore;
- Percorrer a árvore em pré-ordem, em-ordem, pós-ordem;
- Destruir a árvore.

Árvore Binária

- **Árvore Binária: operação de criação**

– A implementação do TAD Árvore usa uma estrutura ligada.

– Para tanto, cria-se primeiro a estrutura do nó. Aqui vamos criar a classe protegida **`_BinTreeNode`** com os campos **`data`**, **`left`** e **`right`**.

```
class _BinTreeNode:
   def __init__(self, data):
      self.data = data
      self.left = None
      self.right = None
```


Árvore Binária

- **Árvore Binária: operação de percurso**

– As operações realizadas em uma árvore binária dependem da aplicação. Uma das operações mais usadas é o percurso.

– A operação de percurso, caminha por uma coleção, um item por vez, para acessar ou visitar cada item.

– A operação real executada ao “visitar” um item depende da aplicação, mas pode envolver algo tão simples como imprimir o item de dados ou salvá-lo em um arquivo.

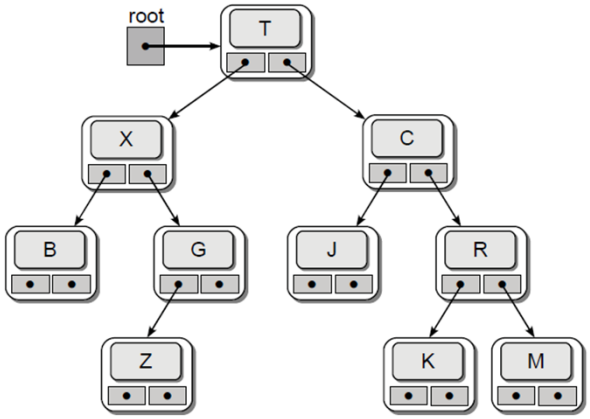
– São percursos em árvore: pré-ordem, em-ordem, pós-ordem.

– Um percurso em árvore deve começar com o nó raiz, pois esse é o único acesso à árvore, e segue para a **sae** (**`left`**) e para a **sad** (**`right`**), ou vice-versa.

Árvore Binária

- **Árvore Binária: operação de percurso**

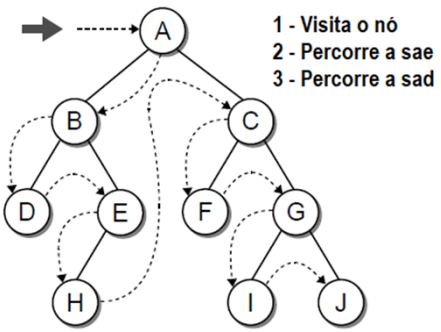
– Uma representação de uma árvore binária, com os nós e seus três campos.



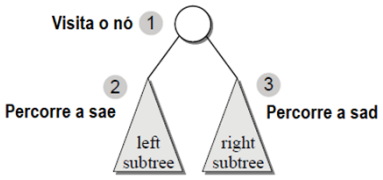
Árvore Binária

- **Árvore Binária: percurso em pré-ordem**

– O percurso em pré-ordem visita primeiro o nó e em seguida suas subárvores a esquerda e a direita, respectivamente (**RED**).



1 - Visita o nó
2 - Percorre a sae
3 - Percorre a sad



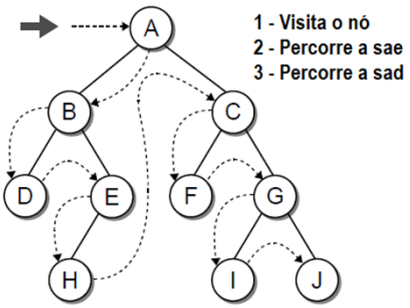
Árvore Binária

- **Árvore Binária: percurso em pré-ordem**

– Segue código do percurso em pré-ordem:

```
def preorder(subtree):
   if subtree is not None:
      print(subtree.data)
      preorder(subtree.left)
      preorder(subtree.right)
```

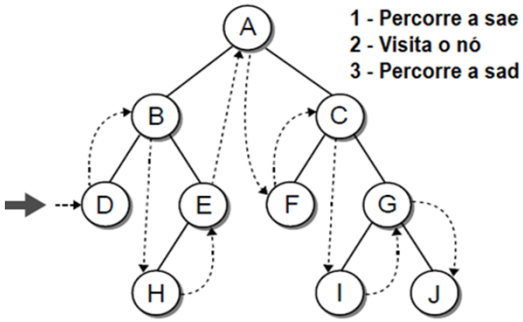
Considerando a árvore binária da figura, o percurso em pré-ordem visita os nós:
A, B, D, E, H, C, F, G, I, J.

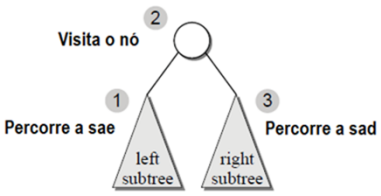


Árvore Binária

- **Árvore Binária: percurso em-ordem**

– O percurso em-ordem visita primeiro a subárvore a esquerda, visita a raiz e visita a subárvore a direita, respectivamente (**ERD**).





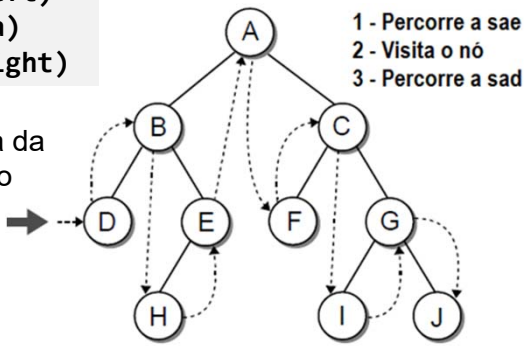
Árvore Binária

- **Árvore Binária: percurso em-ordem**

– Segue código do percurso em-ordem:

```
def inorder(subtree):
   if subtree is not None:
      inorder(subtree.left)
      print(subtree.data)
      inorder(subtree.right)
```

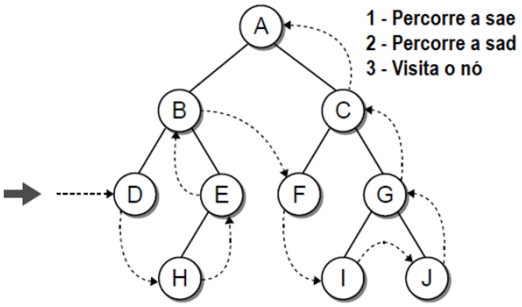
Considerando a árvore binária da figura, a execução do percurso em-ordem visita os nós:
D, B, H, E, A, F, C, I, G, C.

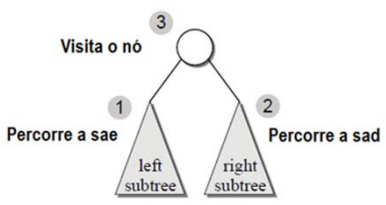


Árvore Binária

- **Árvore Binária: percurso pós-ordem**

– O percurso pós-ordem visita primeiro a subárvore a esquerda, segue para a subárvore a direita e por fim a raiz, respectivamente (**EDR**).





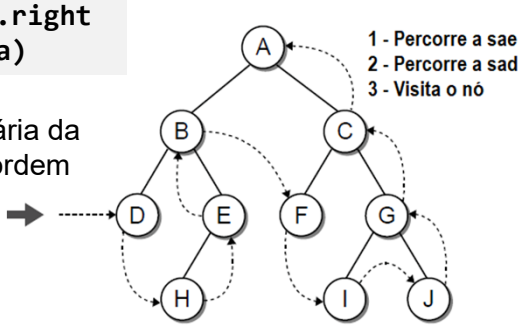
Árvore Binária

- **Árvore Binária: percurso pós-ordem**

– Segue código do percurso pós-ordem:

```
def postorder(subtree):
  if subtree is not None:
    postorder(subtree.left)
    postorder(subtree.right
    print(subtree.data)
```

Considerando a árvore binária da figura, o percurso em pós-ordem visita os nós:
D, H, E, B, F, I, J, G, C, A.



Árvore Binária

- **Árvore Binária: percurso em largura**

– Os percursos em pré-ordem, em-ordem e pós-ordem são exemplos de **percursos em profundidade**.
  - Ou seja, os nós são percorridos mais profundamente na árvore antes de retornar aos nós de nível superior.

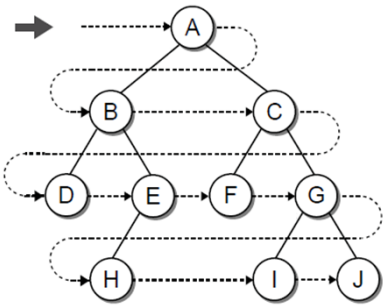
– Outro tipo de percurso que pode ser realizada em uma árvore binária é o **percurso em largura**.

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

Árvore Binária

- **Árvore Binária: percurso em largura**

– A figura abaixo mostra a ordenação lógica dos nós no percurso em largura numa árvore.



– A recursão não pode ser usada para implementar um percurso em largura, pois as chamadas recursivas devem seguir os links que levam mais fundo na árvore.

Árvore Binária

- **Árvore Binária: percurso em largura**

```
def breadthFirst(bintree):
  # Cria uma fila e enfileira a raiz.
  q = Queue()
  q.enqueue(bintree)
  # Visita cada nó na árvore.
  while not q.isEmpty():
    # Remove o próximo nó da fila e o visita.
    node = q.dequeue()
    print(node.data)
    # Enfileira os dois filhos.
    if node.left is not None:
      q.enqueue(node.left)
    if node.right is not None:
      q.enqueue( node.right )
```