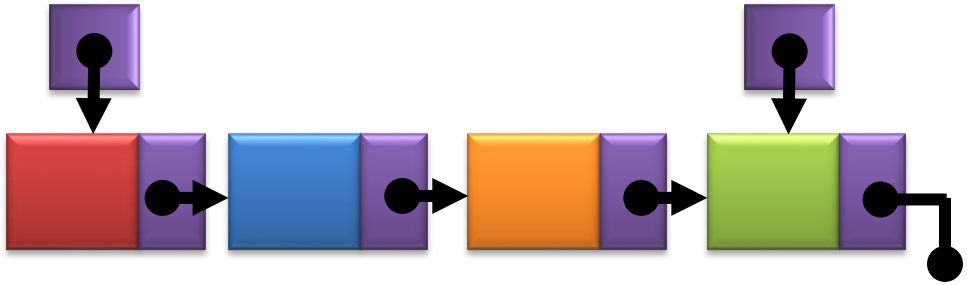
Árvores

por Rossana B Queiroz

Estruturas Lineares

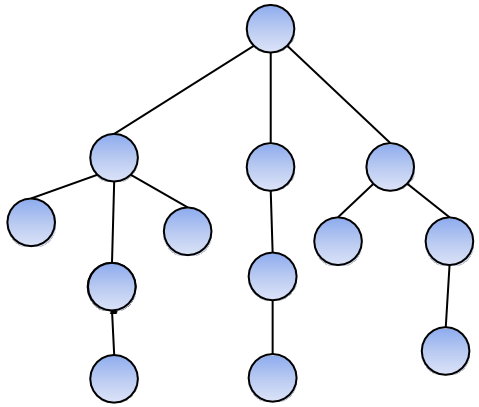
- Representam as informações organizadas de forma sequencial, formando apenas uma dimensão* ou caminho
  - Exemplo:
    - Arrays*
    - Listas
    - Pilhas
    - Filas



* Organização na memória e a forma de acesso devem ser lineares. Nesse contexto, arrays multidimensionais como uma matriz é considerada linear, pois está disposta de forma linear na memória e o acesso aos índices fornecem uma organização sequencial.

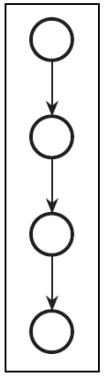
Estruturas Não Lineares

- Elementos armazenados de forma não sequencial
- Múltiplos encadeamentos, caminhos e destinos
- Podem representar relações mais complexas entre os elementos
  - Ex.:
    - Árvores
    - Grafos

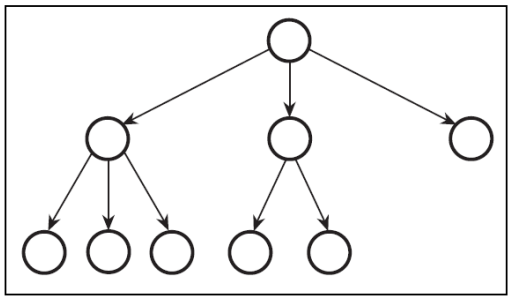


Estruturas de Dados Lineares e Não Lineares

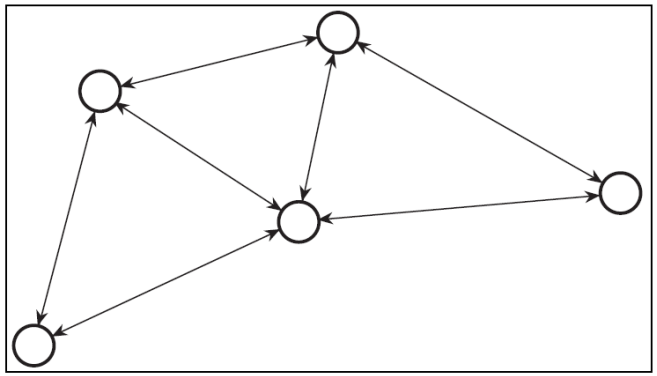
- A principal diferença entre esses dois tipos de estruturas é a **forma como os elementos estão organizados e relacionados entre si**.
  - Estruturas de dados lineares organizadas em uma única sequência
  - estruturas de dados não lineares são multidimensionais e permitem relações mais complexas entre os elementos.



Lista encadeada



Árvore

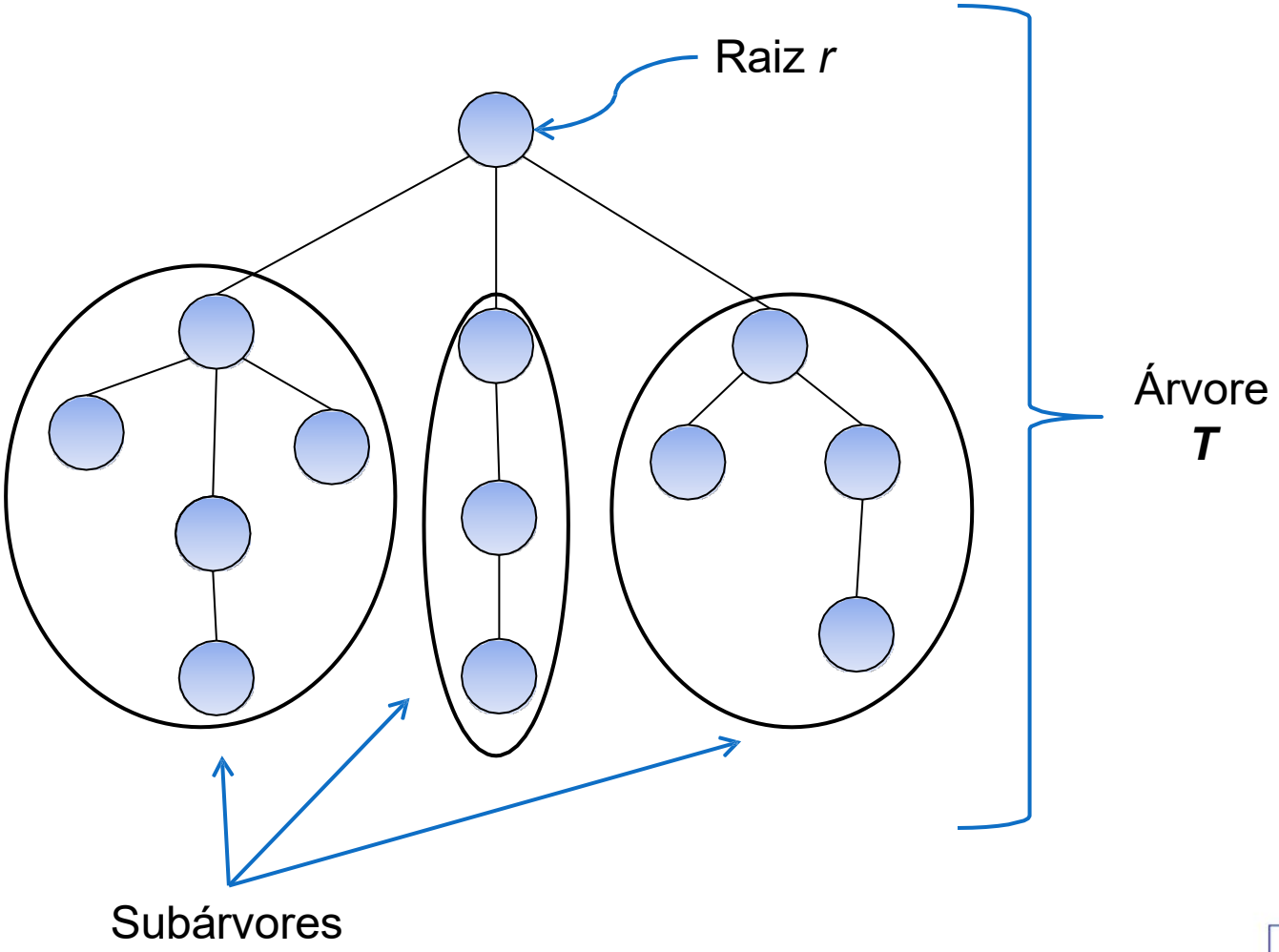


Grafo

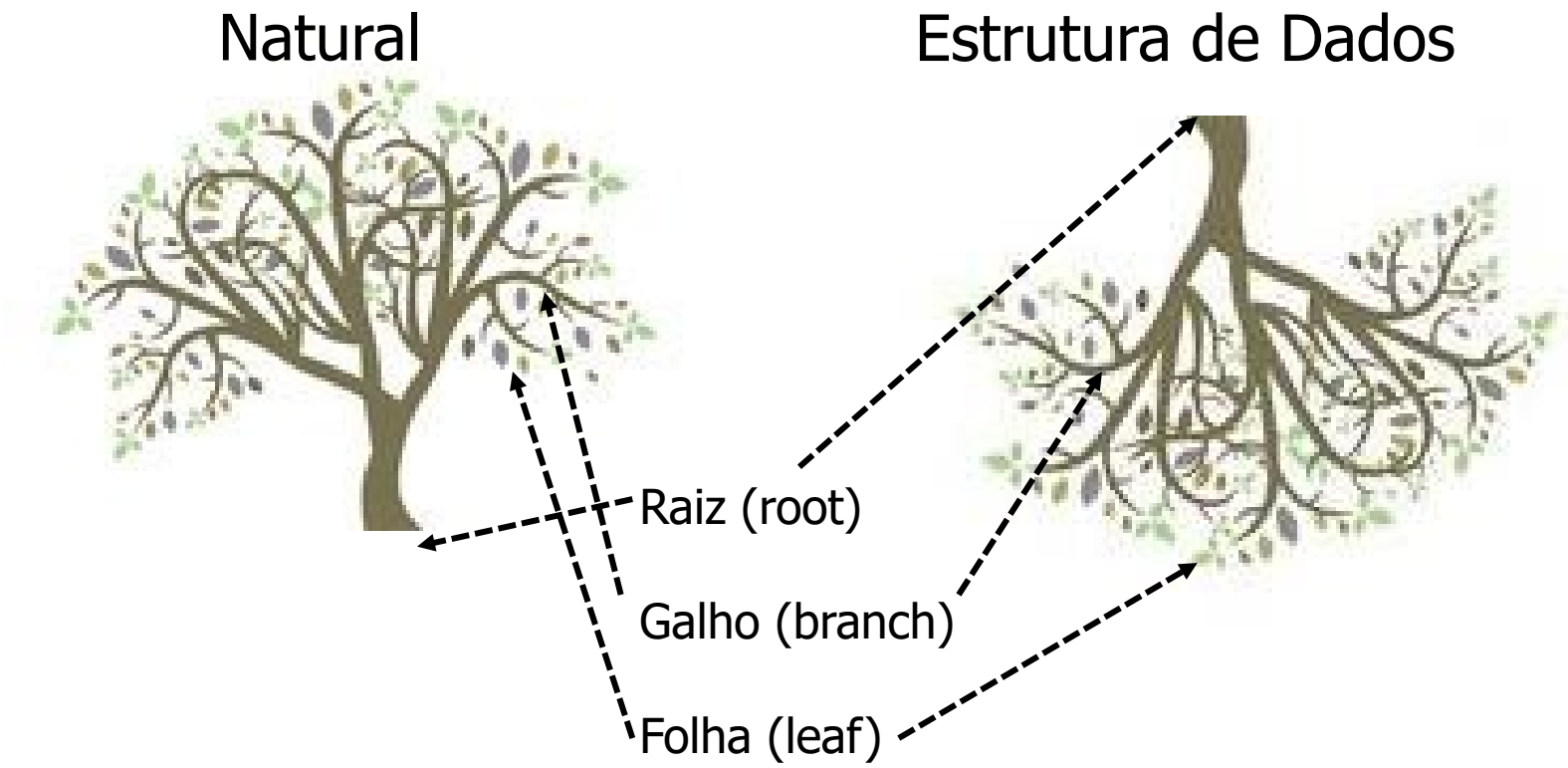
Árvores

- **Estruturas de dados** adequadas para a representação de informações de forma **hierárquica**;
- Conceitualmente, uma árvore enraizada *T*, ou simplesmente uma árvore, é um **conjunto finito** de elementos denominados nós ou nodos, tais que:
  - T = 0 é uma árvore dita vazia ou
  - existe um nó especial *r*, chamado **raiz** de T que não tem pai; os restantes constituem um único conjunto vazio ou são divididos em *m* (devendo ser maior ou igual a 1) conjuntos distintos não vazios que são as **subárvores** de *r*, onde cada subárvore é, por sua vez, uma árvore;
  - Cada nó *v* de T que não é a raiz tem apenas um pai.

Árvores



Visão de Árvore



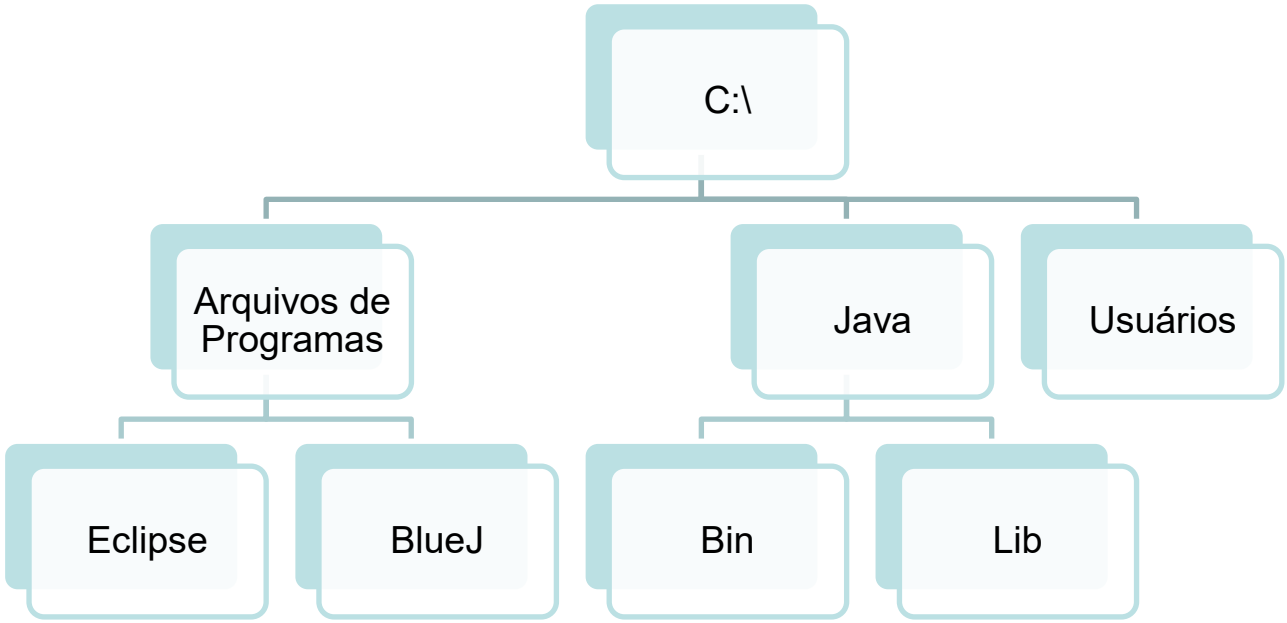
Fonte Imagens: http://cn.vector.me/browse/maps/58

Por que utilizar árvores?

- Diversas aplicações necessitam de **estruturas mais complexas** que as listas estudadas até agora;
- **Inúmeros problemas** podem ser modelados através de árvores;
- Árvores admitem **tratamento computacional eficiente** quando comparadas às estruturas mais genéricas como os grafos (os quais, por sua vez são mais flexíveis e complexos)

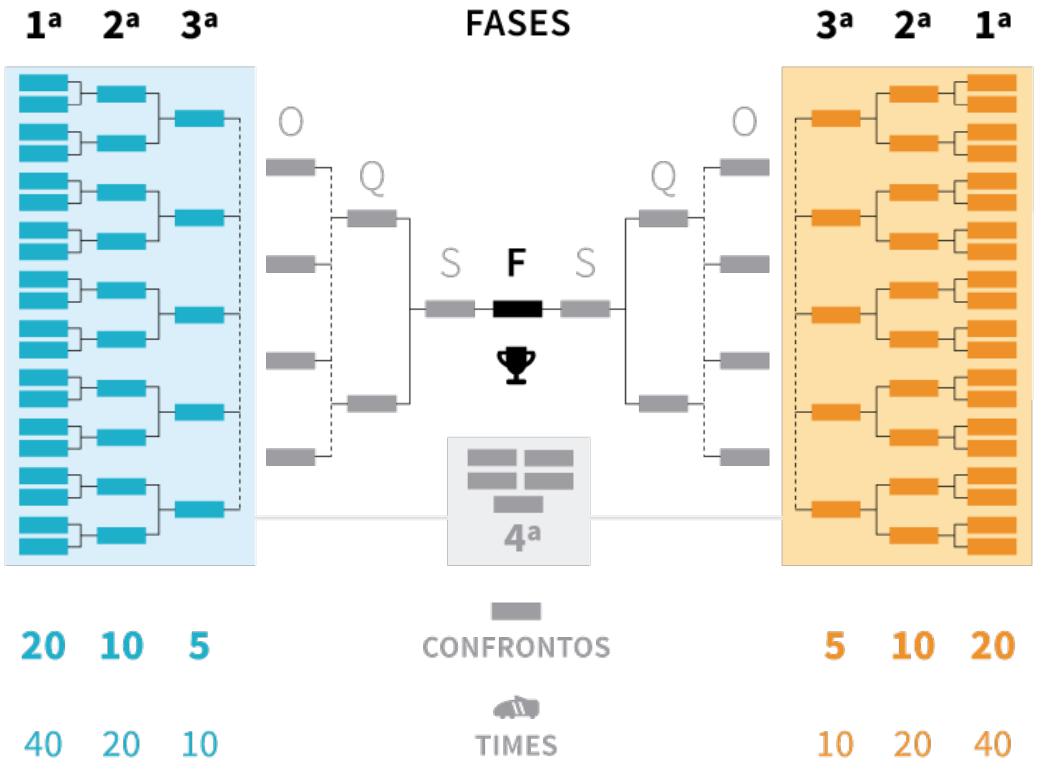
Aplicação de Estruturas em Árvore

- Diretório de um Sistema Operacional



Aplicação de Estruturas em Árvore

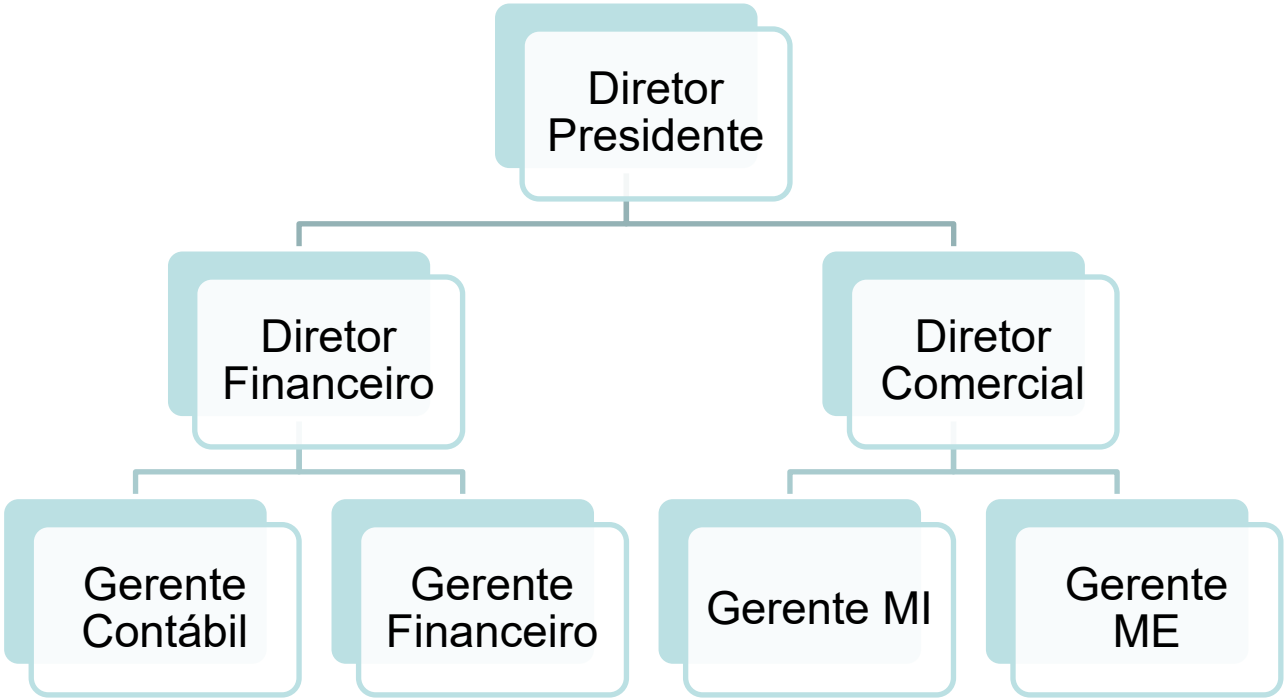
- “Mata-mata” da Libertadores



Fonte Imagens: Gaveta do Povo

Aplicação de Estruturas em Árvore

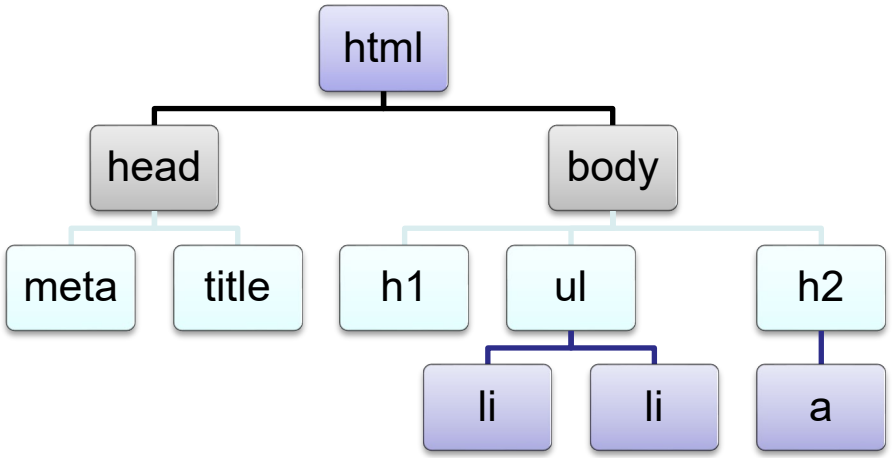
- Organograma de uma Empresa



Aplicação de Estruturas em Árvore

- Uma página HTML

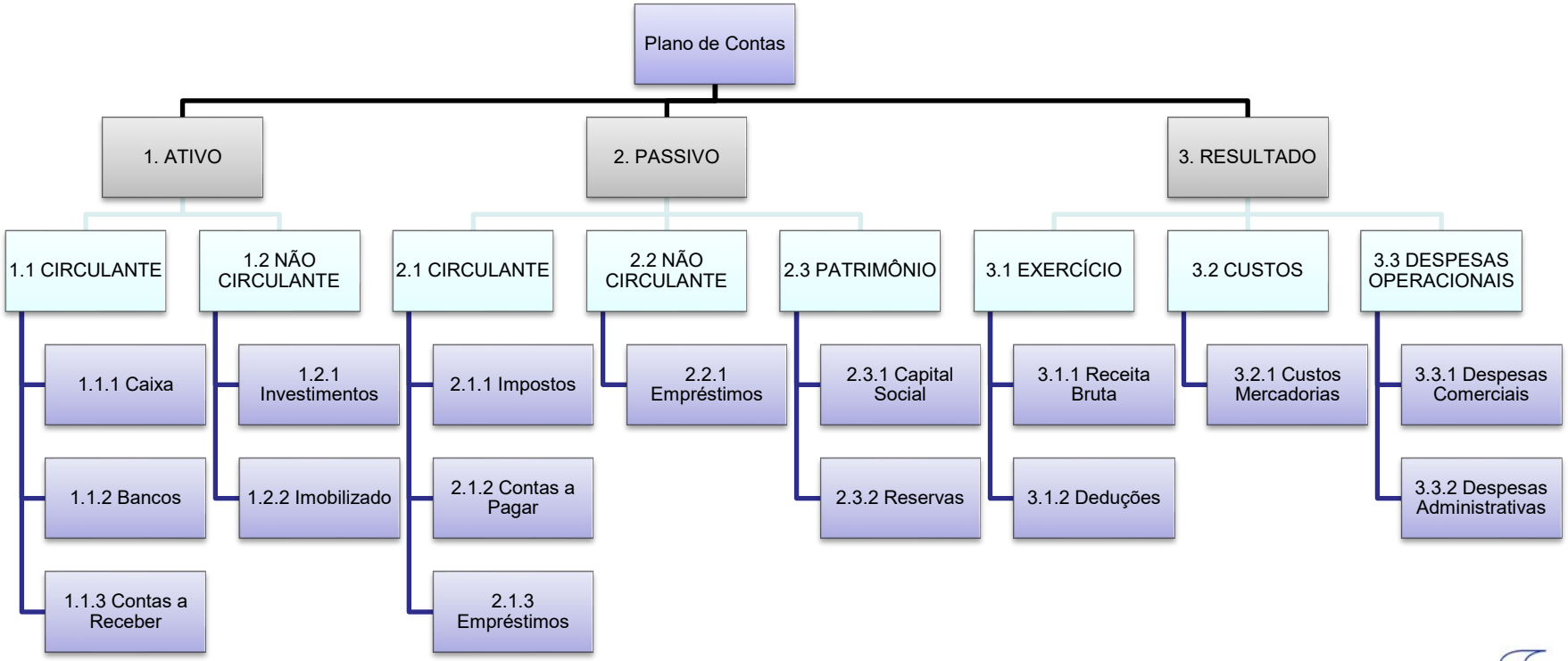
```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
  <head>
    <meta http-equiv="Content-Type"
content="text/html; charset=utf-8" />
    <title>Simples</title>
  </head>
  <body>
    <h1>Web site</h1>
     <ul> <li>Lista de itens 1</li> <li>Lista de
itens 2</li> </ul>
     <h2><a
href="http://www.unisinos.br">Unisinos</a><h2>
  </body>
</html>
```



Fonte: http://interactivepython.org/

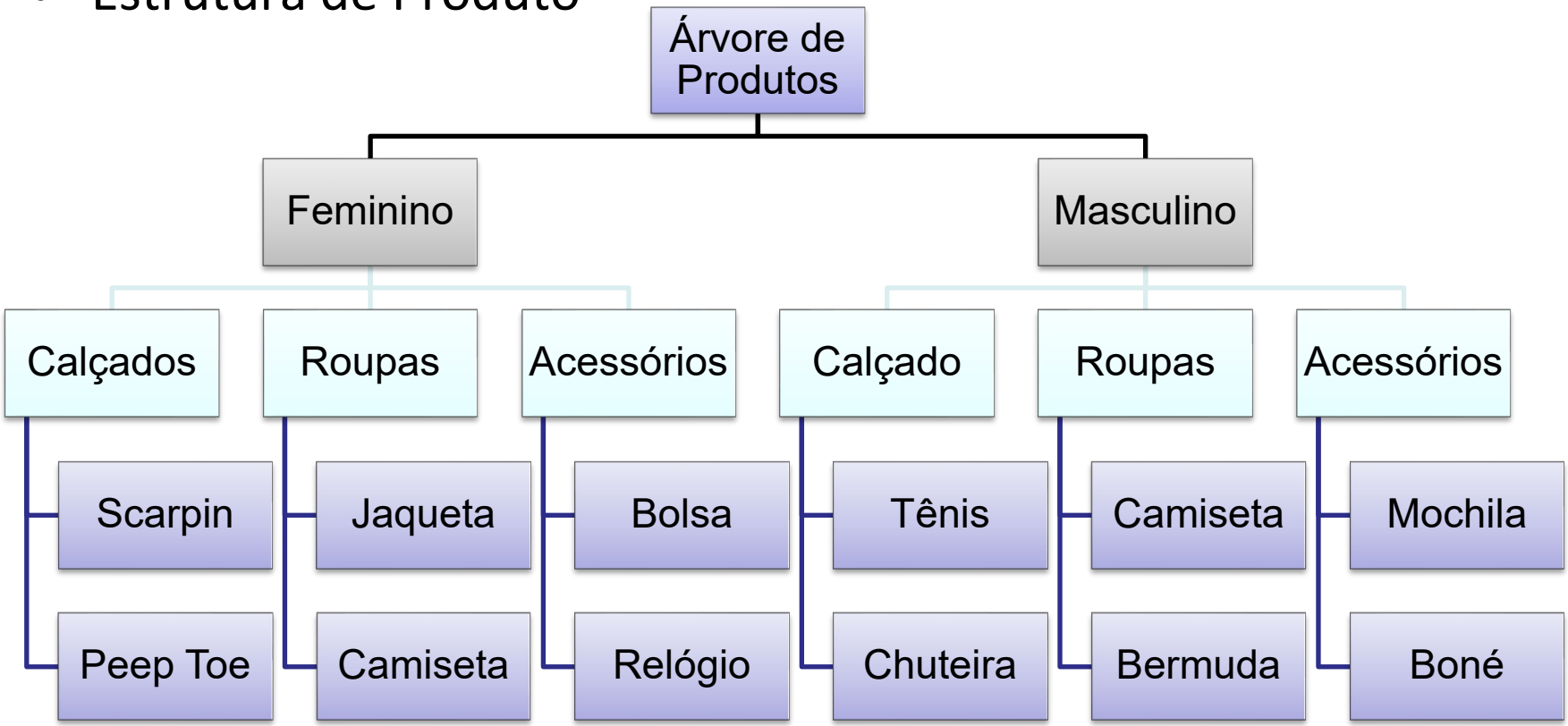
Aplicação de Estruturas em Árvore

- Plano de Contas Contábil



Aplicação de Estruturas em Árvore

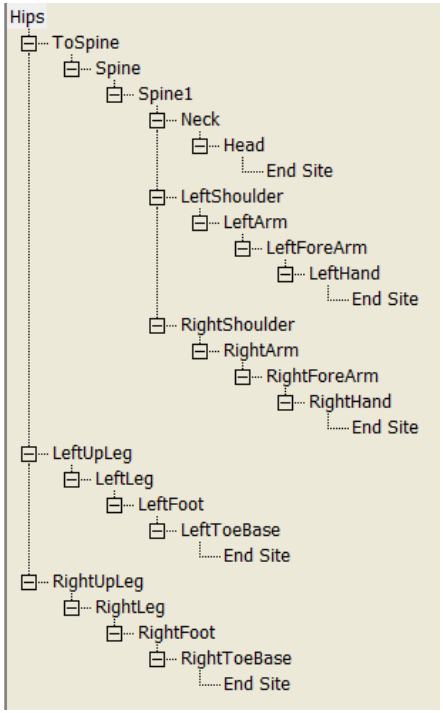
- Estrutura de Produto

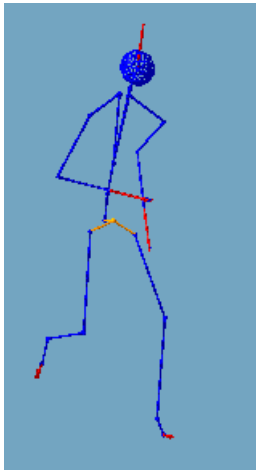


Fonte: http://www.paquetaesportes.com.br

Aplicação de Estruturas em Árvore

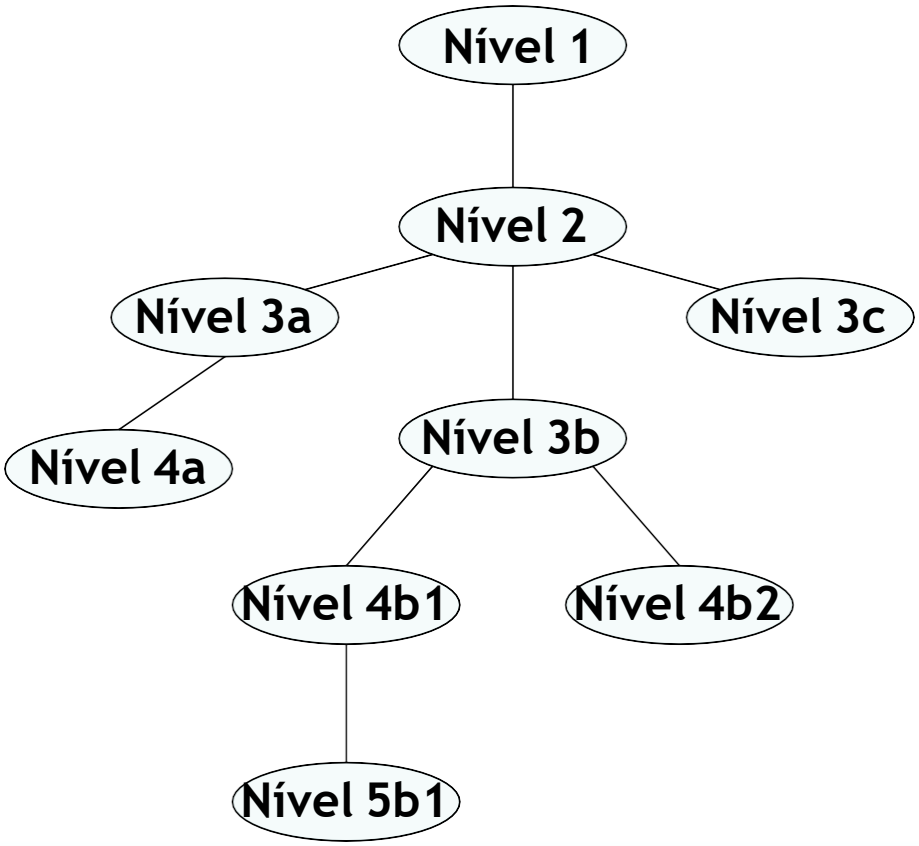
- Hierarquia de partes de um objeto
  - Ex.: esqueleto de animação





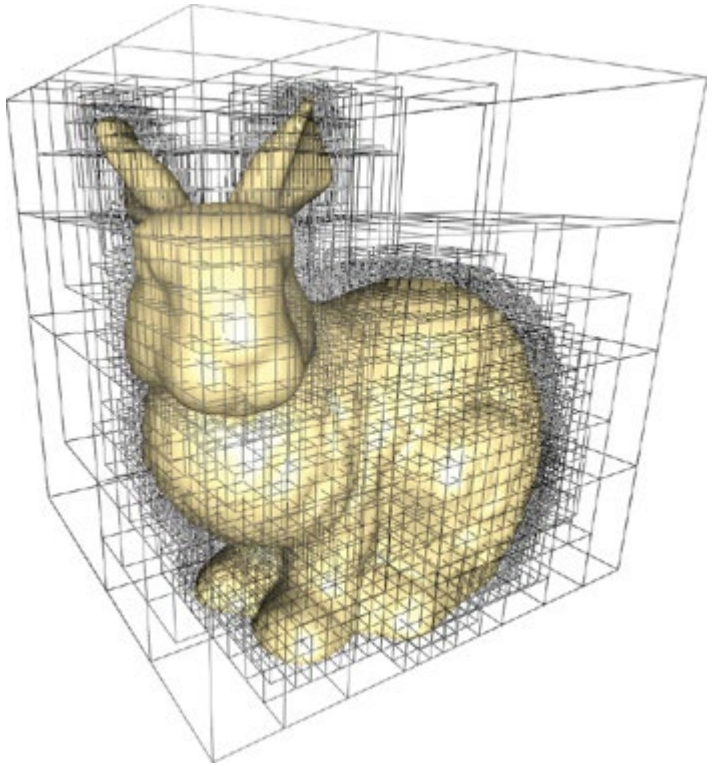
Aplicação de Estruturas em Árvore

- Hierarquia de níveis em um jogo



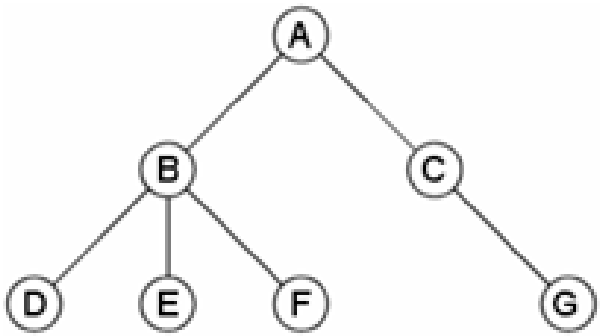
Aplicação de Estruturas em Árvore

- Estruturas de Dados Espaciais
  - Octree, quadtree



Representação de Árvores

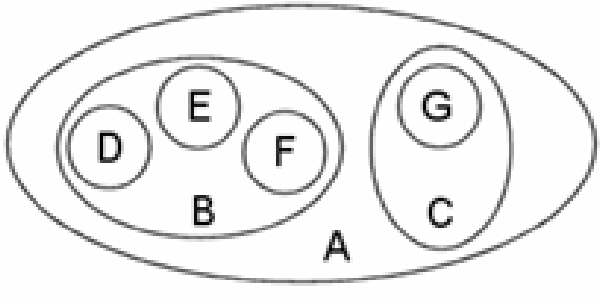
- Grafo (representação mais utilizada)



- Identação

```
A
    B
        D
        E
        F
    C
        G
```

- Diagrama de Venn (ou digrama de inclusão)



- Parênteses Aninhados

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

Conceitos Básicos – Terminologia

- **Nó ou nodo (node):** elemento da árvore;
- **Arco (edge):** ligação entre dois nós;
- **Raiz (root):** primeiro nó da árvore;
- **Subárvore (subtree)**: são os galhos da árvore; consiste de um nó e seus descendentes;
- **Grau de um nó (degree of a node)**: número de subárvores de um nó ou o número de filhos;
- **Grau de uma árvore (degree of a tree)**: é o número máximo entre os graus de seus nós;
- **Folha (ou nó terminal ou nó externo) (external node or leaf)**: um nó que possui grau zero, ou seja, não possui subárvores;
- **Nó não terminal (ou nó interno) (internal node):** é um nó que não é uma folha e é diferente da raiz (cuidado: alguns autores tratam a raiz como um nó interno, pois depende do propósito! Em nosso contexto, a raiz não faz parte!!!).

Conceitos Básicos – Terminologia

- **Pai (parent):** o pai de um nó *n* é um nó que possui caminho para o nó *n*, sendo o primeiro nó no caminho do nó *n* para a raiz;
- **Irmão (sibling):** são nós que possuem o mesmo pai;
- **Filho (child):** é a raiz da subárvore de um nó;
- **Ancestral (ancestor):** qualquer nó no caminho da raiz até o nó em questão (inclusive).
- **Descendente (descendent):** qualquer nó a partir dos caminhos possíveis do nó em questão (inclusive).

Conceitos Básicos – Terminologia

- **Caminho (path):** é a sequência única de arcos que conectam os nós;
- **Comprimento (path length):** número de arcos de um caminho;
- **Profundidade (depth):** é o número de ancestrais do nó (excluindo ele) ou o comprimento da raiz até o nó; recursão para baixo ou para cima;
- **Altura (height):** é o maior comprimento do nó até uma folha qualquer; recursão para baixo; é comum relacionar profundidade com o nó e altura com a árvore, porém, não há impedimento em aplicar as terminologias para a árvore e o nó;
- Profundidade e altura são simétricas, ou seja, a altura de uma árvore *T* é a maior profundidade de uma das folhas;
- **Nível (level):** o nível *d* é um nó em profundidade *d*.

Conceitos Básicos – Terminologia

– O número de filhos permitidos por nó e as informações armazenadas em cada nó diferenciam os **diversos tipos de árvores** existentes;

- Todos os nós **são acessíveis** a partir da raiz;
- Existe um **único caminho** entre a raiz e qualquer outro nó;
- À exceção da raiz, cada nó possui um pai.

Conceitos Básicos – Terminologia

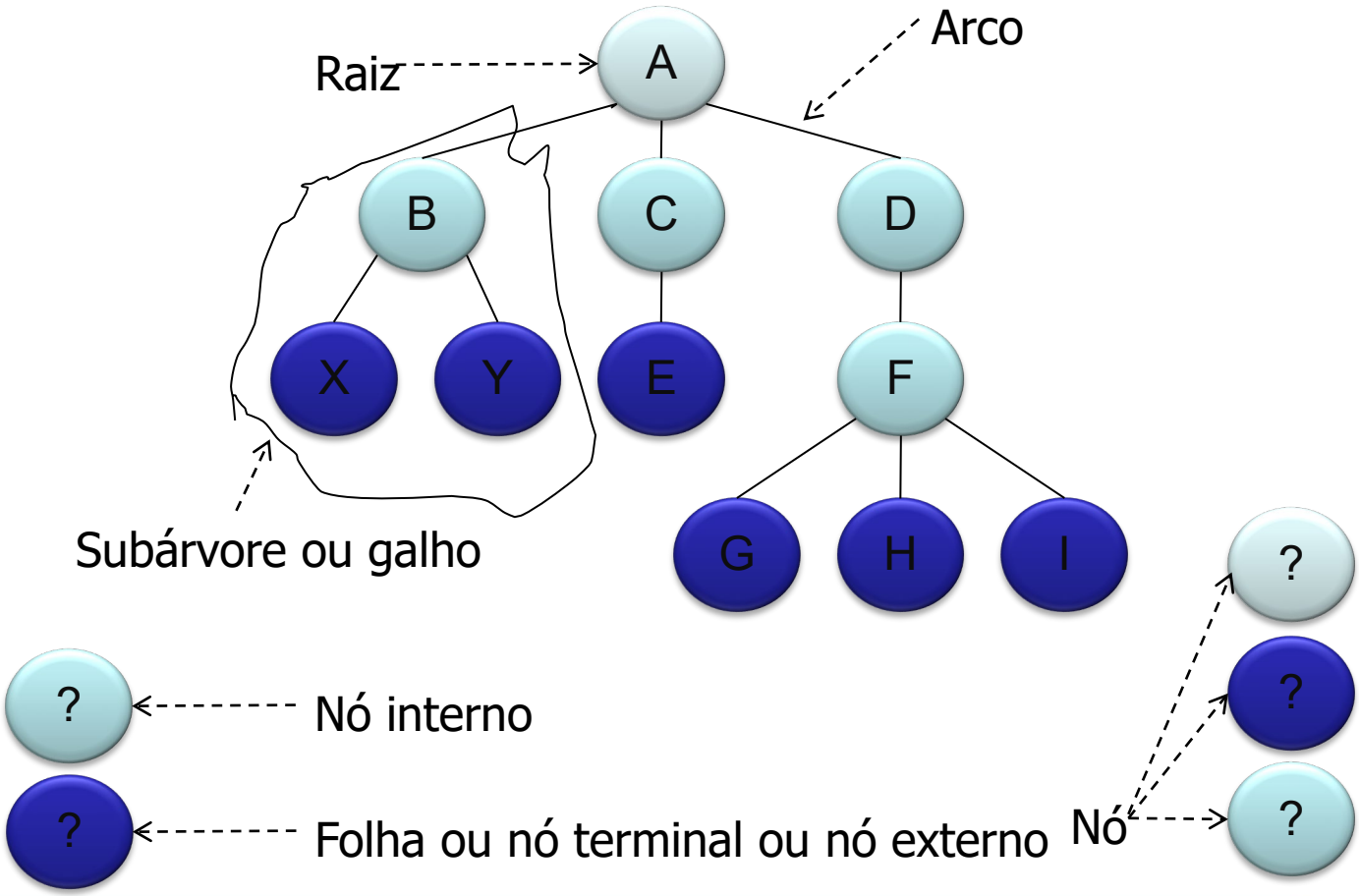
- A raiz é um nó que não possui antecessores (similar ao bullet anterior!);
- As folhas não possuem nós filhos, ou seus filhos são estruturas vazias;
- Cada nó tem que ser atingível a partir da raiz através de uma sequência única de arcos.

Conceitos Básicos – Terminologia

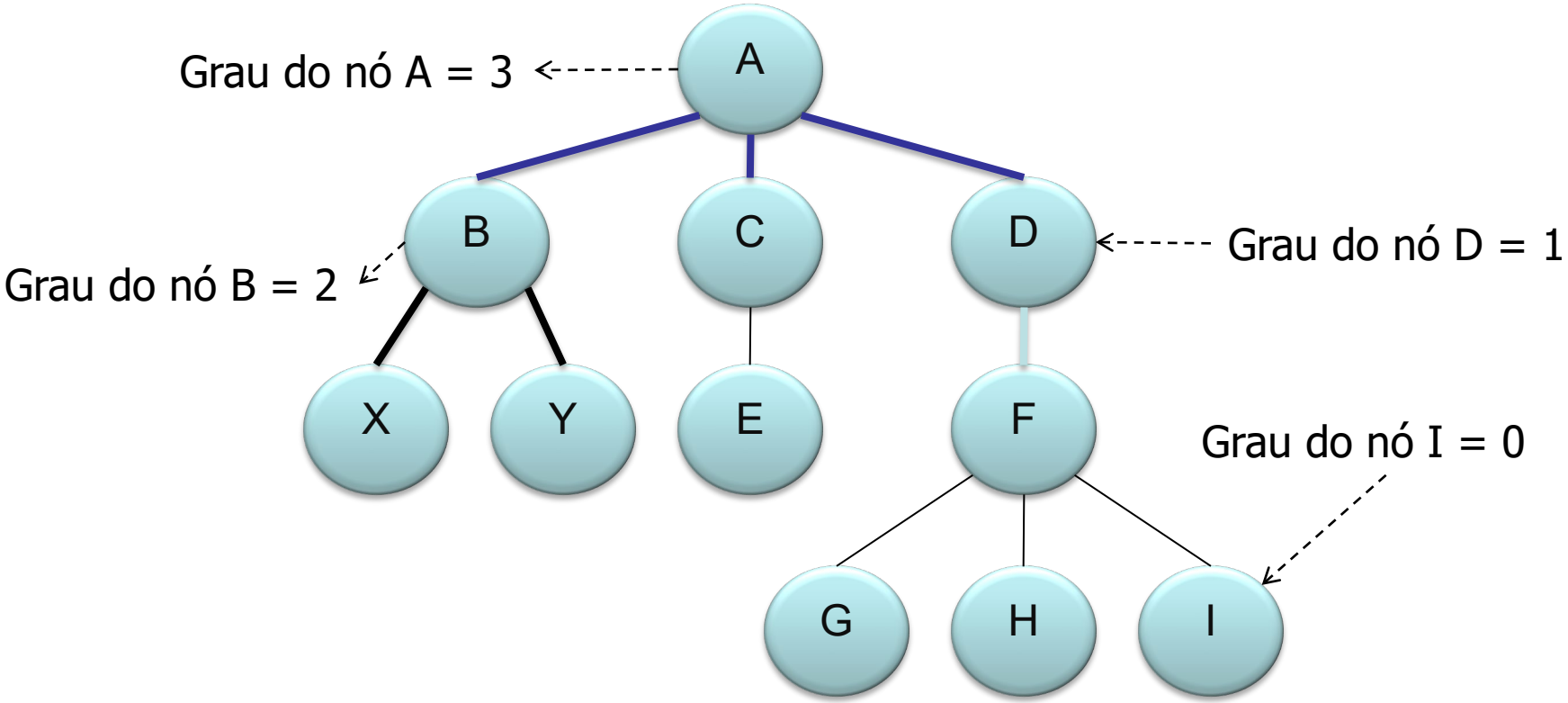
- **Árvore Genérica**: cada nó poderá ter *n* subárvores, ou seja, não tem um número máximo de subárvores;

- **Árvore N-ária:** cada nó tem no máximo *n* subárvores. Há um caso especial que estudaremos na próxima aula onde n = 2.

Aplicação dos Conceitos

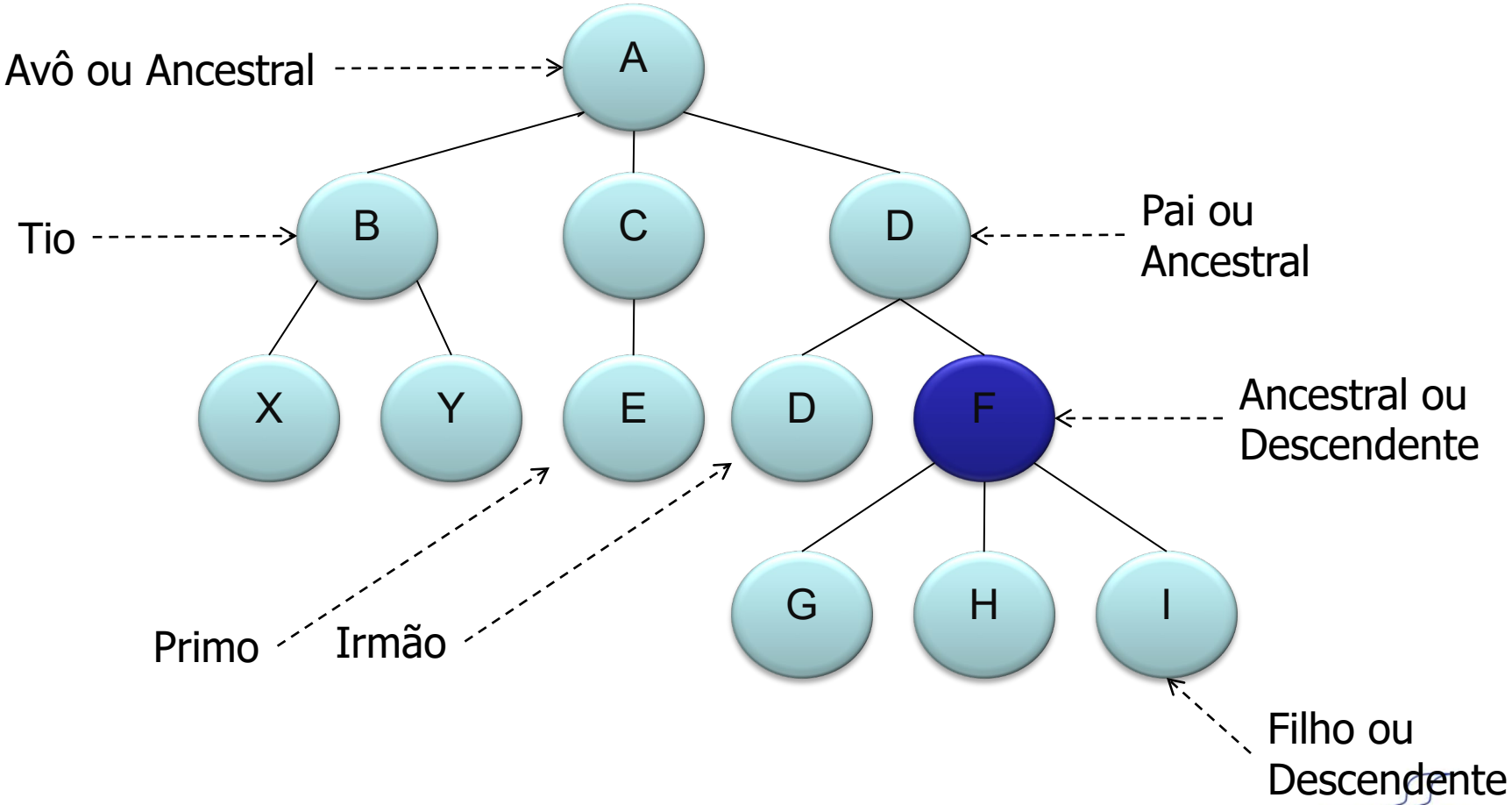


Aplicação dos Conceitos - Grau

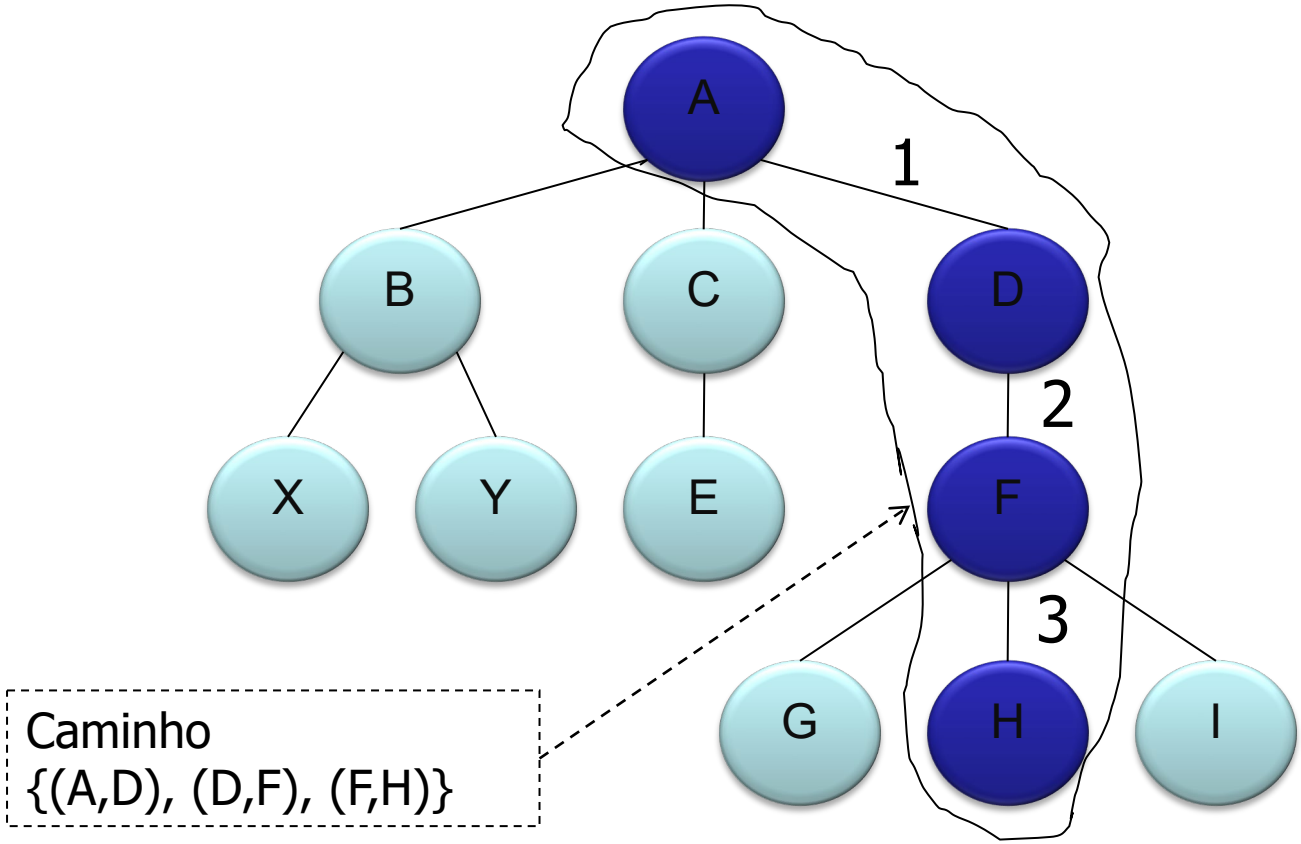


Grau da árvore = 3 (A e F)

Aplicação dos Conceitos - Parentesco

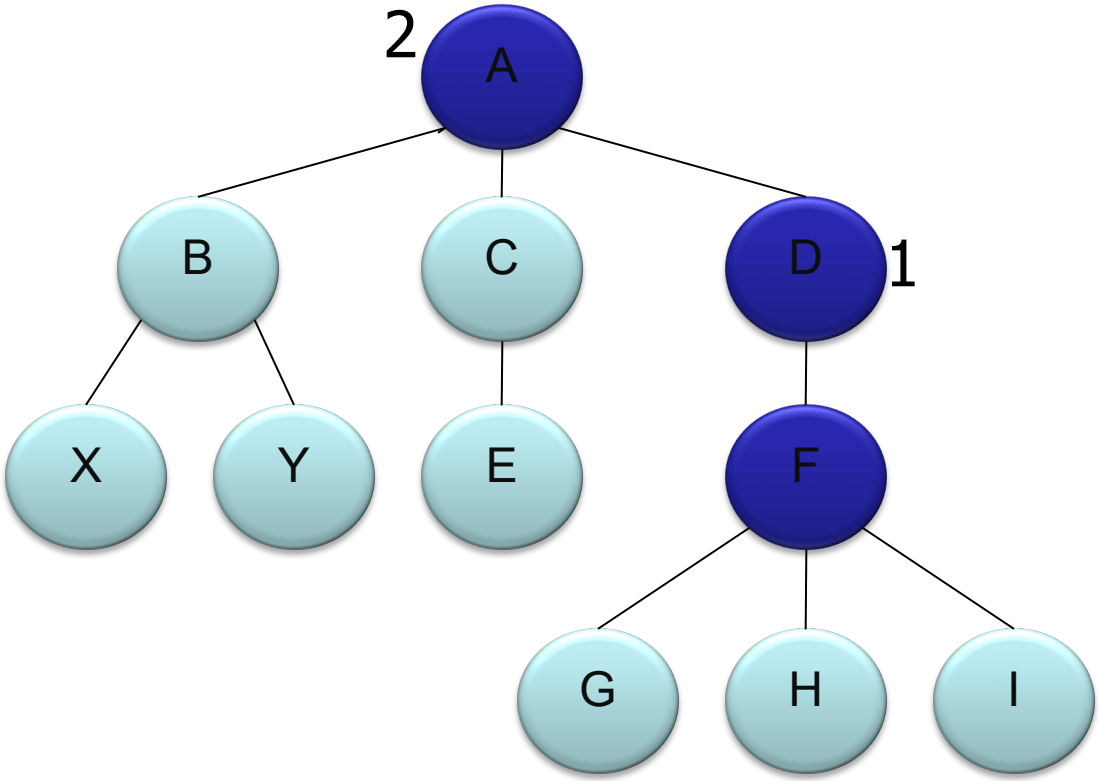


Aplicação dos Conceitos - Caminho



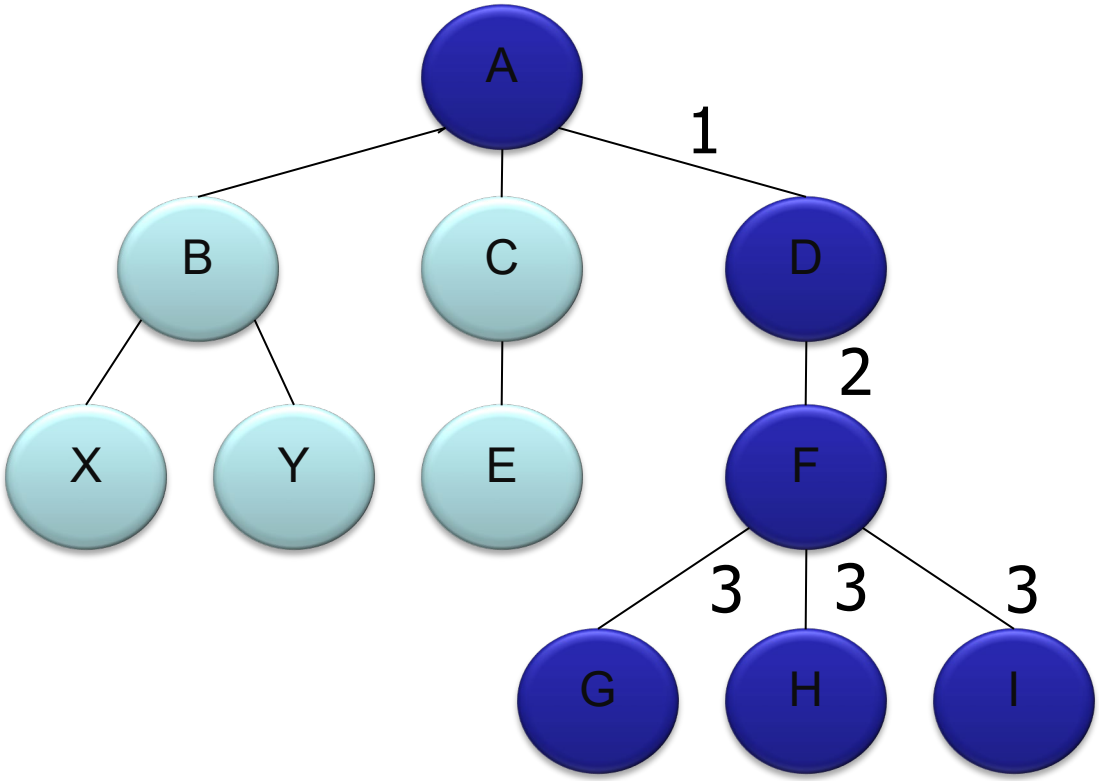
Comprimento do caminho (A,H) é 3

Aplicação dos Conceitos - Profundidade



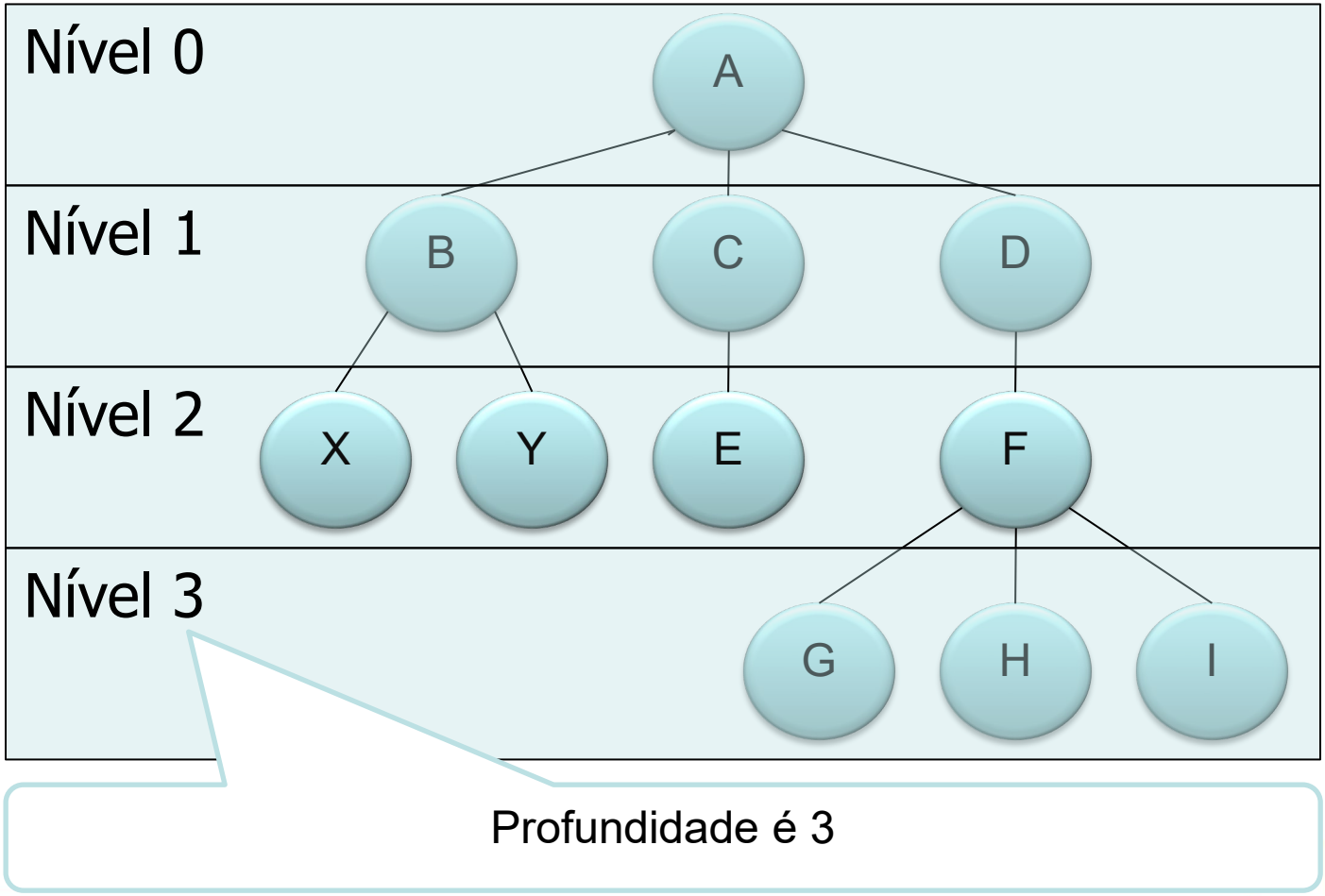
Profundidade do nó F é 2

Aplicação dos Conceitos - Altura



Altura da árvore é 3

Aplicação dos Conceitos – Nível



O que iremos trabalhar em árvores?

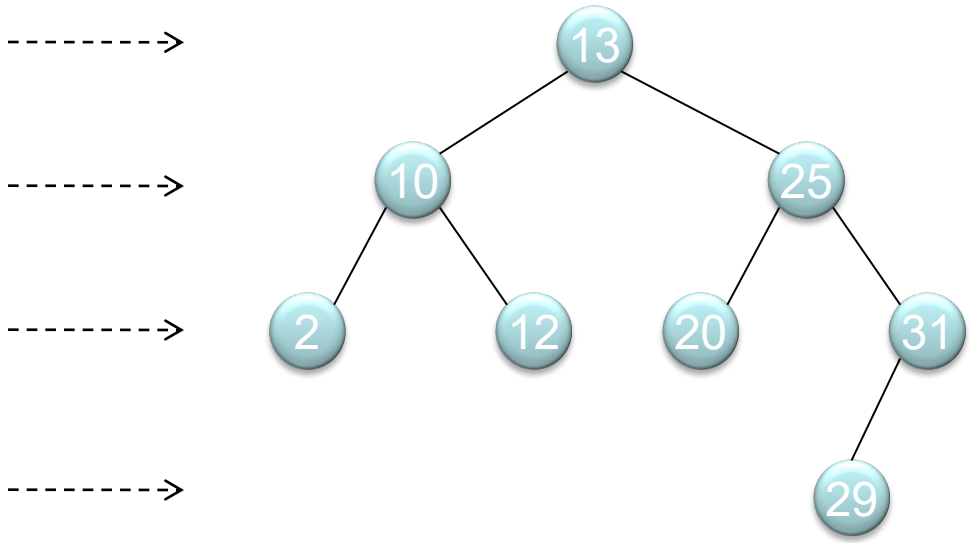
- Criação da árvore (estrutura de dados)
- Inserção de um nó
- Exclusão de um nó
- Acesso ao nó
  - Tipos de caminhamento (**percurso**)
- Exclusão da árvore

Percurso em Árvores

- O percurso em árvores é o processo de visitar cada nó da árvore;
- O percurso pode ser interpretado como colocar todos os nós em uma linha;
- Mas qual a ordem? Existem n! percursos diferentes, quase todos caóticos; depende da aplicação;
- Os básicos são: percurso em **profundidade** e percurso em **amplitude**.

Percurso em Árvores – Amplitude – Largura

- Um percurso em amplitude (extensão ou nível) consiste em visitar cada nó começando do menor nível (raiz) movendo-se para os níveis mais altos, nível após nível, visitando cada nó da esquerda para a direita:
  - Breadth First Search (BFS)



Fila: 13

Fila: 10, 25

Fila: 25, 2, 12

Fila: 2, 12, 20, 31

Fila: 12, 20, 31

Fila: 20, 31

Fila: 31

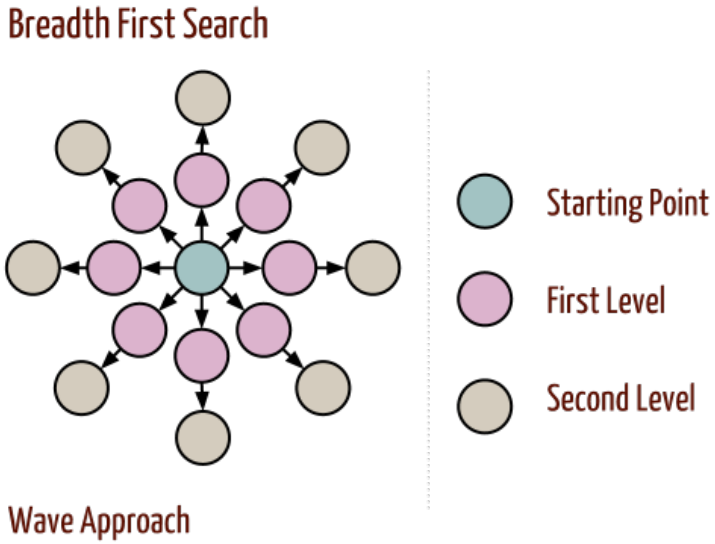
Fila: 29

BFS: 13, 10, 25, 2, 12, 20, 31, 29

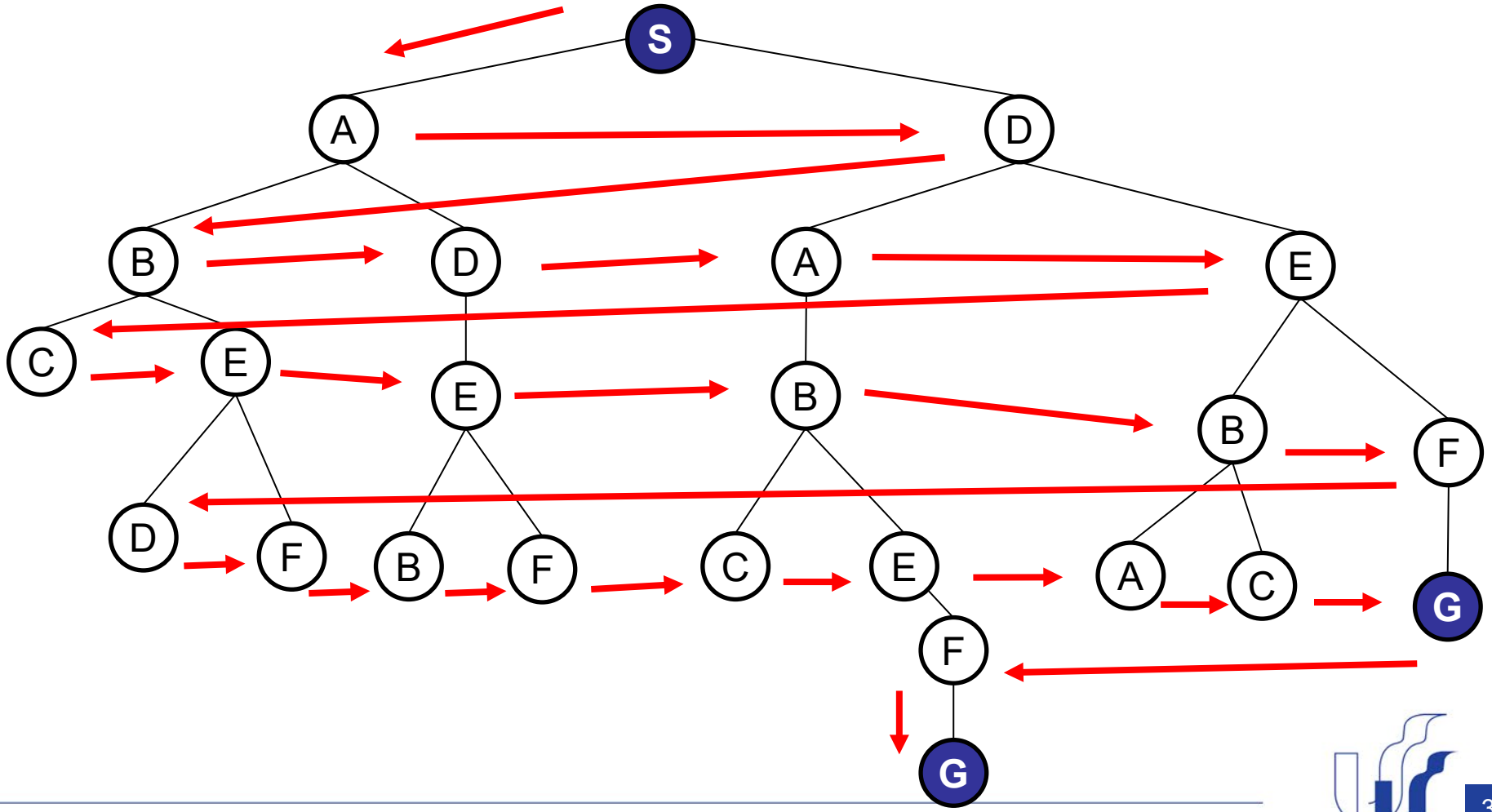
Percurso em Largura

- Pseudo código

```
BreadthFirst( Node )
     Queue.Enqueue( Node )
     Mark( Node )
     While( Queue.IsNotEmpty )
          Process( Queue.Front )
          For Each Child of Queue.Front
               if NotMarked( Child )
                    Queue.Enqueue( Child )
                    Mark( Child )
               end if
          end For
          Queue.Dequeue()
     End While
End Function
```



Percurso em Largura



Percurso em Largura

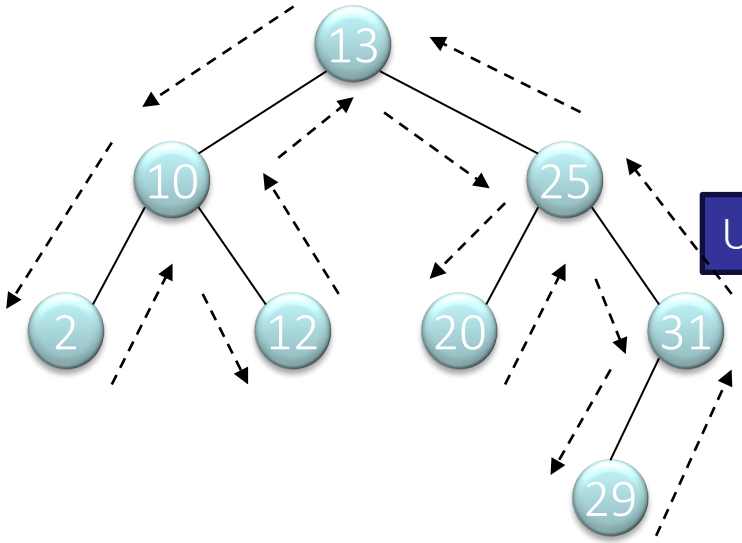
Possível implementação

```
Visite um nodo arbitrário, marque-o e coloque-o em uma fila Q

Enquanto a fila Q não estiver vazia
        Retire um elemento N de Q
        Para cada nodo M (não marcado) adjacente a N
                Visite M
                Coloque M na fila Q
                Marque M
```

Percurso em Árvores - Profundidade

- O percurso em profundidade prossegue tanto quanto possível à esquerda (ou direita), então se move para trás até a primeira encruzilhada, vai um passo para a direita (ou esquerda) e novamente, tanto quanto possível, para a esquerda (ou direita).

  - Depth First Search (DFS)

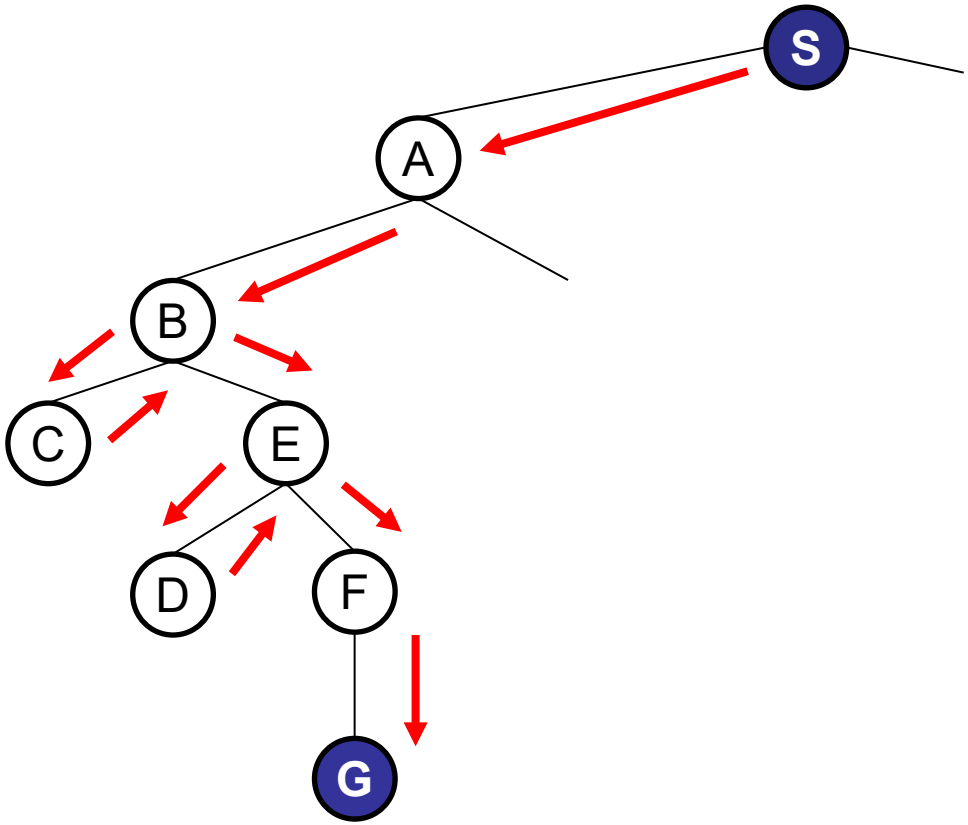


V – Visitar um nó
L – Percorrer à esquerda
R – Percorrer à direita

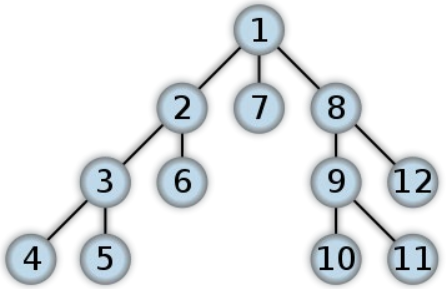
VLR VRL LVR RVL LRV RLV

Um percurso possível: 2, 12, 10, 20, 29, 31, 25, 13

Busca em Profundidade



- Permite incluir uma profundidade máxima
- Implementado com recursividade ou com uma pilha
- Critério de parada:
  - Achou a solução
  - Atingiu a profundidade máxima



Busca em Profundidade

- Pseudo código (recursivo)

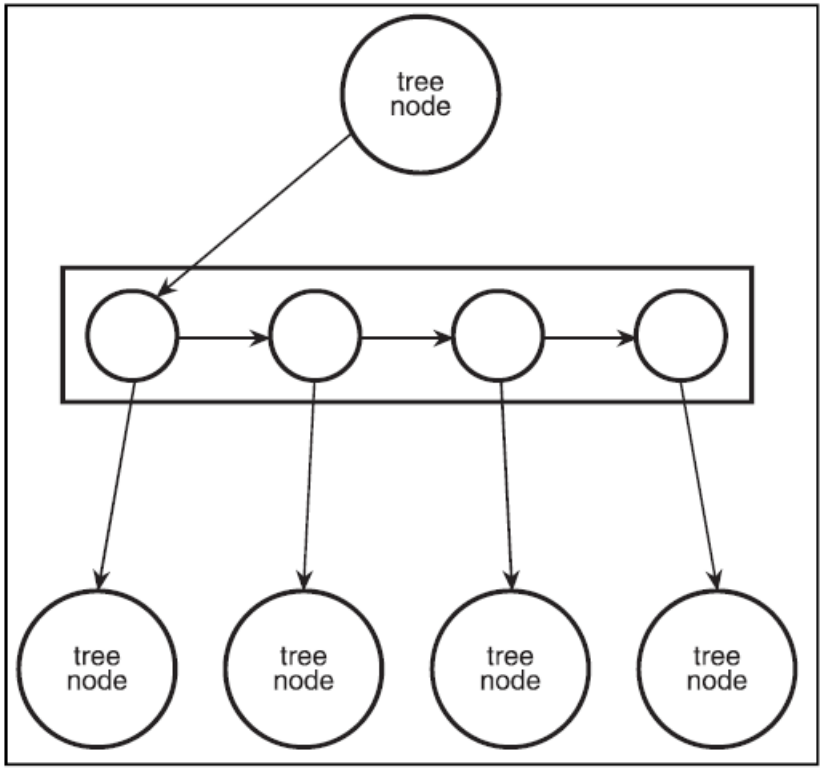
```
DepthFirst( Node )
    Process( Node )
    Mark( Node )
    For Every Child of Node
        If NotMarked( Child )
            DepthFirst( Child )
        End If
    End For
End Function
```

Busca em Profundidade

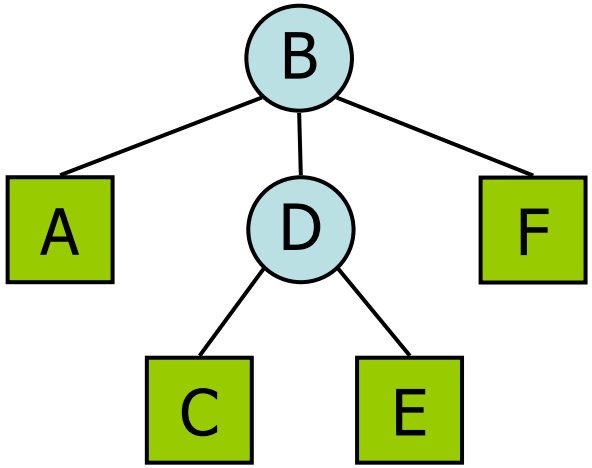
- Possível implementação

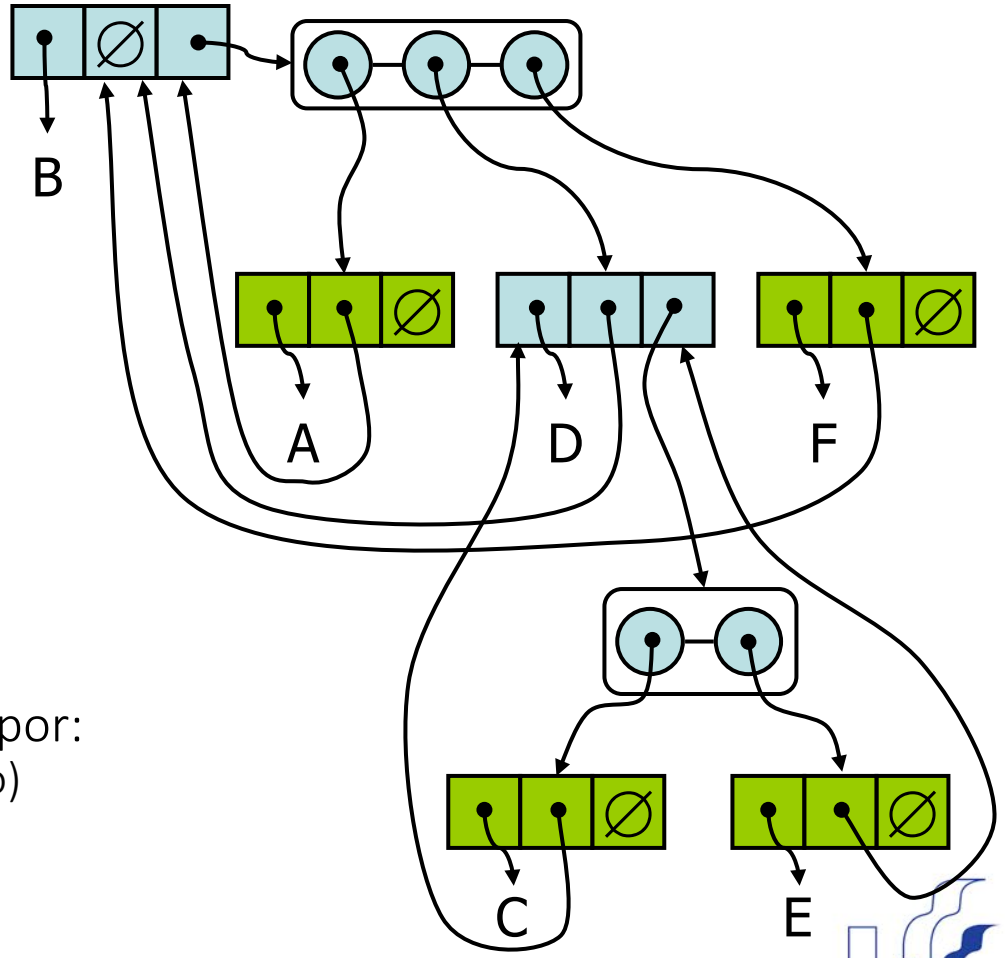
```
Visite um nodo arbitrário, marque-o e coloque-o em uma pilha S

Enquanto a pilha S não estiver vazia
        Retire um elemento N de S
        Para cada nodo M (não marcado) adjacente a N
                Visite M
                Coloque N na pilha S
                Marque M
                Faça N = M
```

Estrutura de Dados – Árvore Genérica



Estrutura de Dados – Árvore Genérica

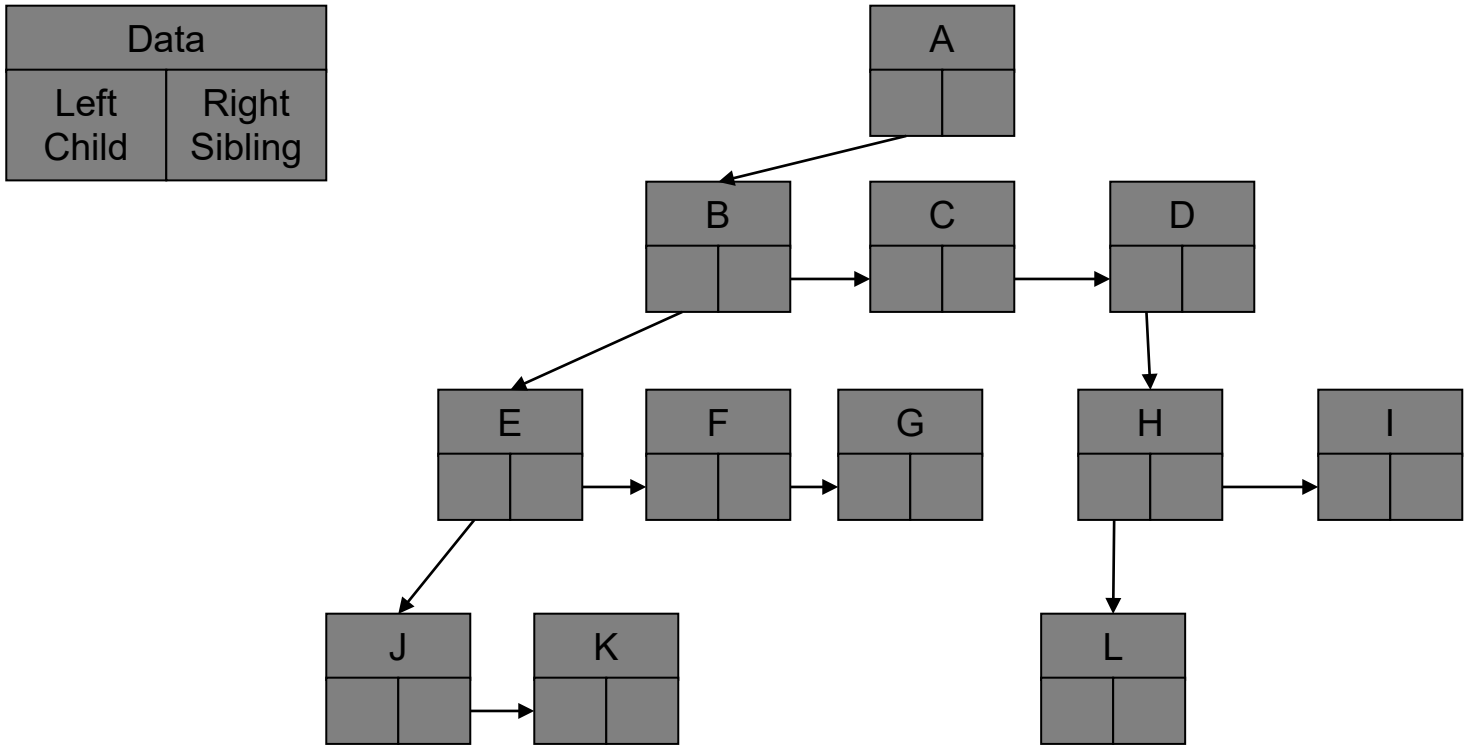




O nó é representado por:

- Elemento (dado)
- Nó pai
- Lista de filhos

Estrutura de Dados – Representação Filho Esquerdo e Irmão Direito



Créditos e Bibliografia

- Materiais prof. Gilberto Irajá Müller
- Materiais profa. Rossana B Queiroz