Capítulo 1

Motivação

As **árvores** são, sem dúvida, uma das estruturas de dados mais fascinantes e poderosas da ciência da computação. Embora, à primeira vista, possam parecer complexas, elas estão no coração de inúmeros sistemas que usamos todos os dias.

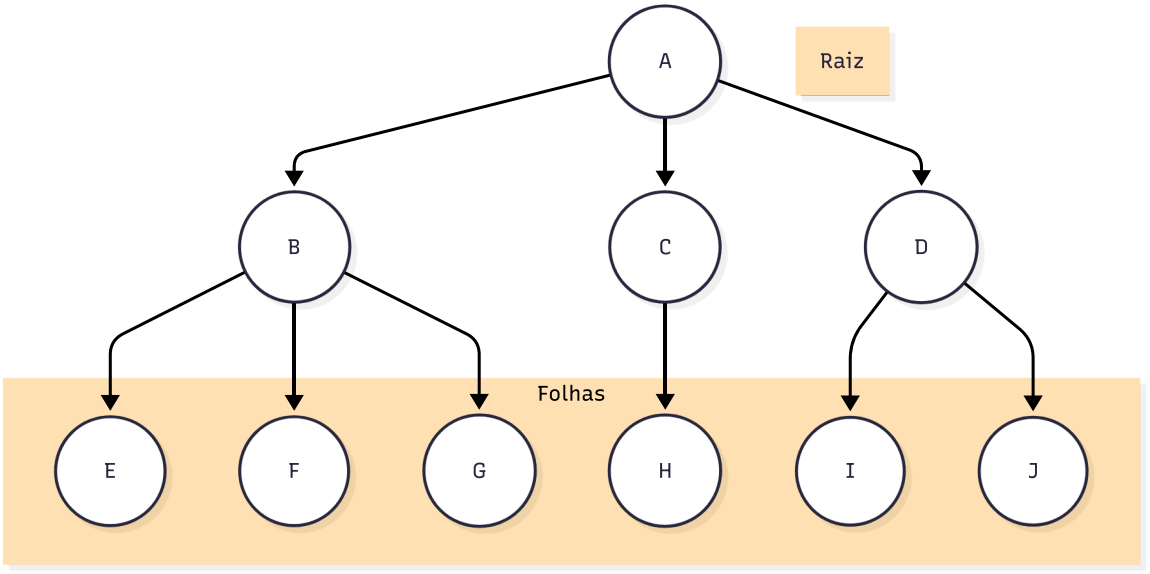
Pense em como o sistema de arquivos do seu computador organiza **pastas e arquivos**, ou como os bancos de dados otimizam a **busca por informações**, ou até mesmo como os compiladores de código processam as **linguagens de programação**. Em todos esses cenários, a estrutura por trás é uma árvore.

Neste primeira unidade do curso, vamos mergulhar no mundo das árvores para entender como elas funcionam e por que são tão eficientes. O objetivo é compreender a lógica por trás de sua **estrutura hierárquica** e como essa hierarquia nos permite resolver problemas de forma muito mais elegante e rápida do que com estruturas lineares, como listas e vetores, vistas na disciplina anterior.

Prepare-se para explorar conceitos como **nós**, **raízes**, **folhas** e as diferentes formas de **travessia** que nos permitem visitar cada elemento de uma árvore. Ao final, você não apenas terá uma nova ferramenta em seu arsenal de programação, mas também uma nova perspectiva sobre a organização e a busca de dados. Vamos começar nossa jornada pela base dessas estruturas e construir nosso conhecimento, nó por nó.

Árvores

A árvore é uma **estrutura de dados hierarquizada**, geralmente não linear. É composta por uma **coleção de nós conectados por arestas**. Cada nó pode possuir uma chave que o identifica.



Se um nó **X está conectado com um nó Y** (há uma aresta no sentido de X para Y), dizemos que **X é um nó pai** e **Y um nó filho**. Se dois nós possuem o mesmo pai, eles são ditos **irmãos**. **Um nó só pode ter no máximo um pai**.

No exemplo acima, o nó com chave C é pai do nó com chave H. Além disso, os nós com chaves I e J são irmãos. O nó raiz, cuja chave é A, não possui pai.

Você já deve ter visto estruturas desse tipo antes, como árvore genealógica ou árvore sintática. Na computação, o usuário com grau máximo de privilégios no sistema operacional Linux se chama *root* (raiz em inglês). Isso porque ele têm acesso privilegiado ao diretório raiz de onde todos os outros diretórios se ramificam.

No *Windows*, o diretório raiz se chama C, pois as letras A e B eram reservadas para unidades temporárias (disquetes).

O DOM (Document Object Model) é um dos melhores exemplos de uma estrutura de árvore em programação, especificamente no desenvolvimento web. Ele é a representação em forma de árvore de um documento HTML (ou XML) que é criada pelo navegador. Pense assim:

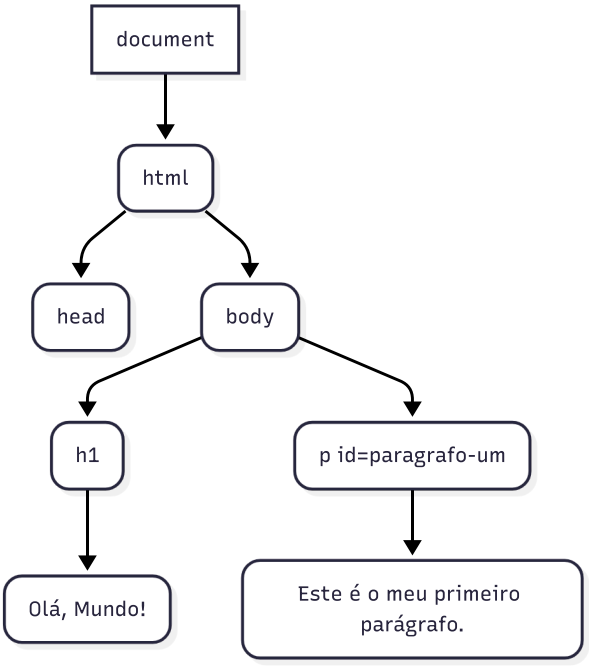
- O documento inteiro é a raiz da árvore.
- Os elementos HTML, como `<html>`, `<head>` e `<body>`, são os primeiros "galhos" que saem da raiz.
- Dentro do `<body>`, outros elementos como `<div>`, `<p>`, `<h1>`, `<a>` e `<img>` são os próximos "galhos" e "folhas".

- E assim por diante, com cada elemento contendo outros elementos, formando uma hierarquia.

Essa representação em árvore permite que linguagens de programação, principalmente o Javascript, interajam com o conteúdo de uma página web. É como se o Javascript usasse o DOM como um mapa detalhado da página para encontrar e manipular qualquer elemento. Veja um exemplo prático:

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Olá, Mundo!</h1>
    <p id="paragrafo-um">Este é o meu primeiro parágrafo.</p>
  </body>
</html>
```

O navegador cria a seguinte estrutura de árvore do DOM para ele:



Com essa estrutura, você pode usar Javascript para, por exemplo, mudar o texto do parágrafo, adicionar um novo elemento à página, ou até mesmo remover um deles. Tudo isso de forma dinâmica, sem precisar recarregar a página inteira.

Lembre-se que

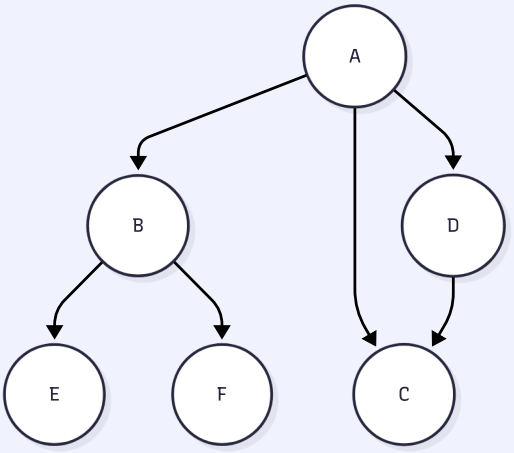
Info

- A raiz é o nó que não possui pai. Ela é única.
- As folhas, ou nós terminais, não possem filhos.
- Os demais nós são chamados de intermediários ou internos

Os exemplos a seguir não são árvores.

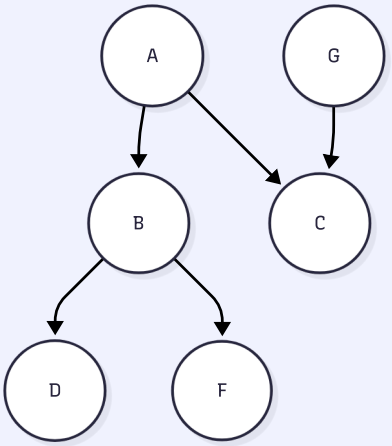
Exemplo 1.1

Abaixo, nota-se que o nó com chave C possui dois pais (A e D) e isso não é permitido.



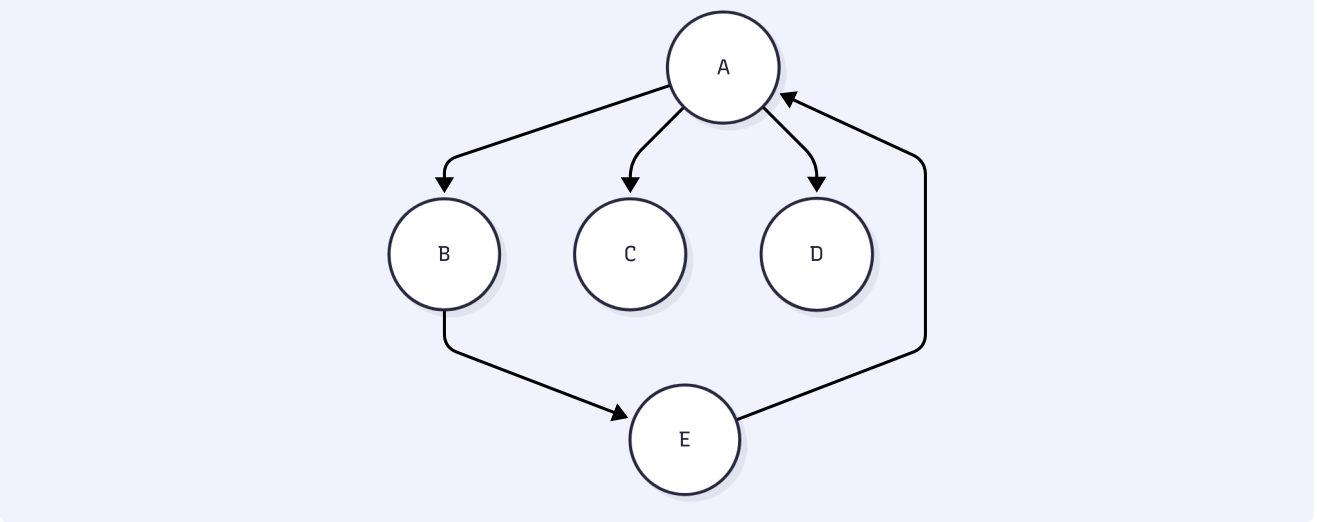
Exemplo 1.2

Neste exemplo, temos mesmo problema de dois pais (nó com chave C). Mas também temos duas raízes (A e G).



Exemplo 1.3

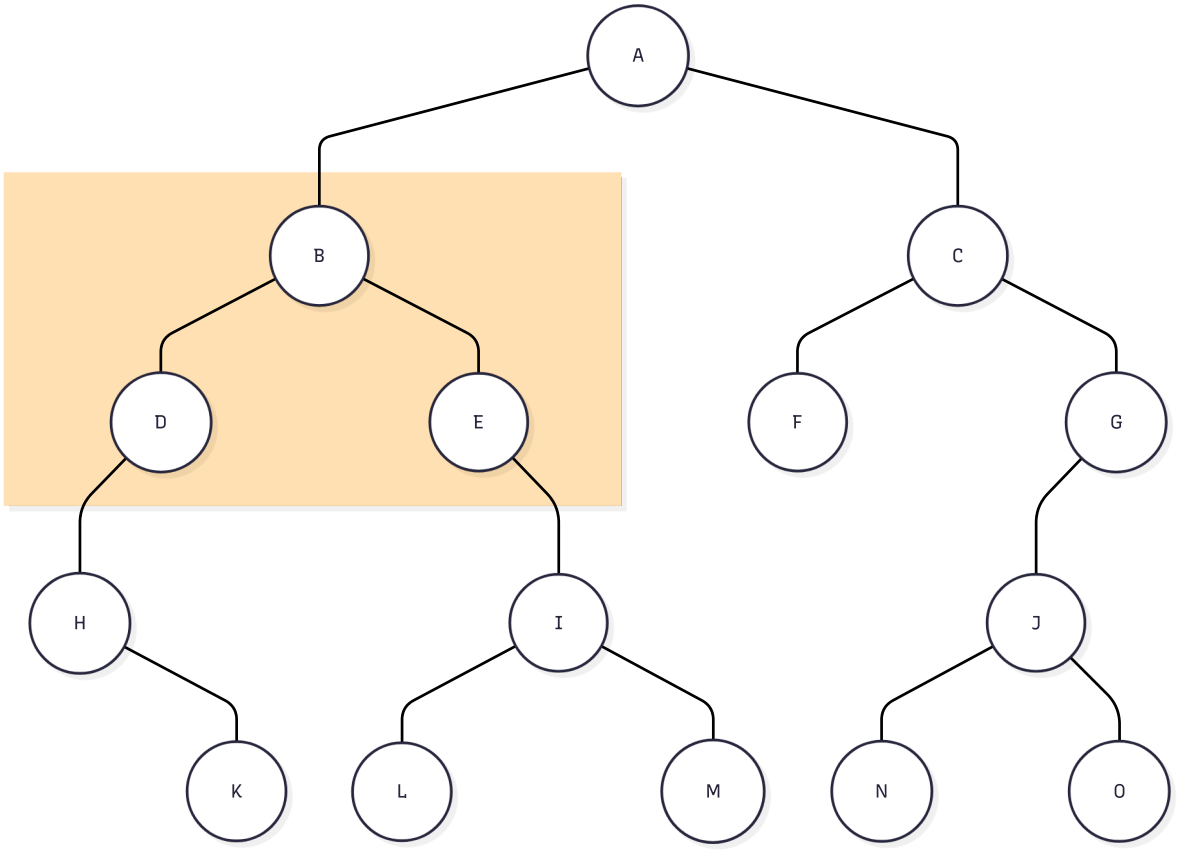
Este caso é interessante, pois todos os nós possuem um pai. Ou seja, não existe raiz e isso não é permitido.



Em uma árvore, a raiz existe e é única. Além disso, todos os nós podem possuir zero ou mais filhos.

Quanto ao número de filhos, uma árvore onde todos os nós possuem somente um filho se comporta como uma lista. Se os nós possuem no máximo dois filhos, a árvore é chamada de binária. Se três filhos, chama-se ternária e assim por diante. Uma árvore em que os nós possuem no máximo n filhos é chamada de n-ária.

Iremos estudar conceitos importante, para tanto considere a árvore T a seguir.



Note que não usamos setas para representar a relação pai/filho. Isso porque podemos admitir que a relação ocorre de cima para baixo, por exemplo E é pai de I ou O é filho de J.

ⓘ Subárvore

Uma **subárvore** consiste em uma porção da árvore.

Os nós destacados em T (B, D e E) formam um subárvore de T. Trata-se de uma miniárvore dentro da árvore original. A subárvore possui sua própria raiz, folhas e nós internos.

Note que uma árvore pode ser considerada uma subárvore de si mesma.

ⓘ Tamanho

O **tamanho** da árvore é definido como o número total de nós.

O tamanho de T é 15, pois possui 15 nós:

- Uma raiz (A),
- Sete folhas (F, K, L, M, N e O) e
- Oito nós internos (B, C, D, E, G, J, I e J) .

ⓘ Altura

A **altura** da árvore é o comprimento do caminho mais longo da raiz até uma folha.

A altura de T é 4. O comprimento pode ser calculado como o número de arestas da raiz até a folha mais distante. Há quatro arestas de A até K, por exemplo.

A altura pode ser calculada para qualquer nó, já que ele e todos os descendentes são considerados uma (sub)árvore. Geralmente se considera a altura de uma árvore vazia como zero.

ⓘ Profundidade

A **profundidade** de um nó é o número de arestas no caminho da raiz até esse nó.

A profundidade do nó J é 3, pois há três arestas da raiz até J. Note que a profundidade da folha mais baixa é igual a altura da árvore.

A profundidade de um nó indica qual nível da árvore ele pertence. A raiz possui nível 0, o filhos da raiz nível 1, os netos da raiz nível 2 e assim sucessivamente. É fácil concluir que a altura está relacionada ao número de níveis.

ⓘ **Grau**

O **grau** de um nó é o número de filhos que ele possui.

Na árvore T, o nó B tem grau 2. Já os nós H e F possuem grau 1 e 0, respectivamente.

Todos esses conceitos são fundamentais e, portanto, devemos estar bem familiarizados com eles. No estudo das árvores como estruturas de dados, é imprescindível dominar as definições de altura, profundidade, subárvore, grau e tamanho.

No próximo capítulo, iremos estudar um tipo especial de árvore chamada binária.