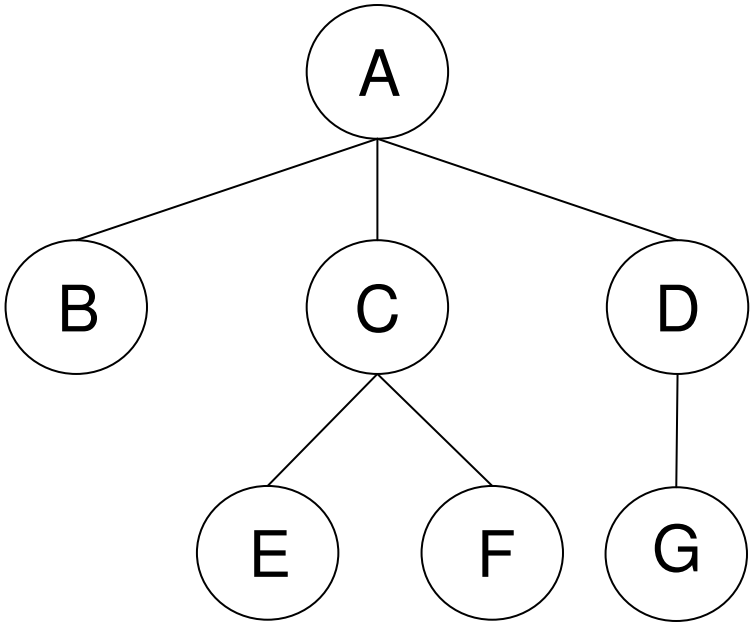
Conteúdo

1. Introdução
2. Listas
3. Pilhas e Filas
4. Árvores
5. Árvores de Pesquisa
   - Árvore Binária e Árvore AVL
   - Árvore N-ária e Árvore B
6. Tabelas de Dispersão (Hashing)
7. Métodos de Acesso a Arquivos
8. Métodos de Ordenação de Dados

Árvores

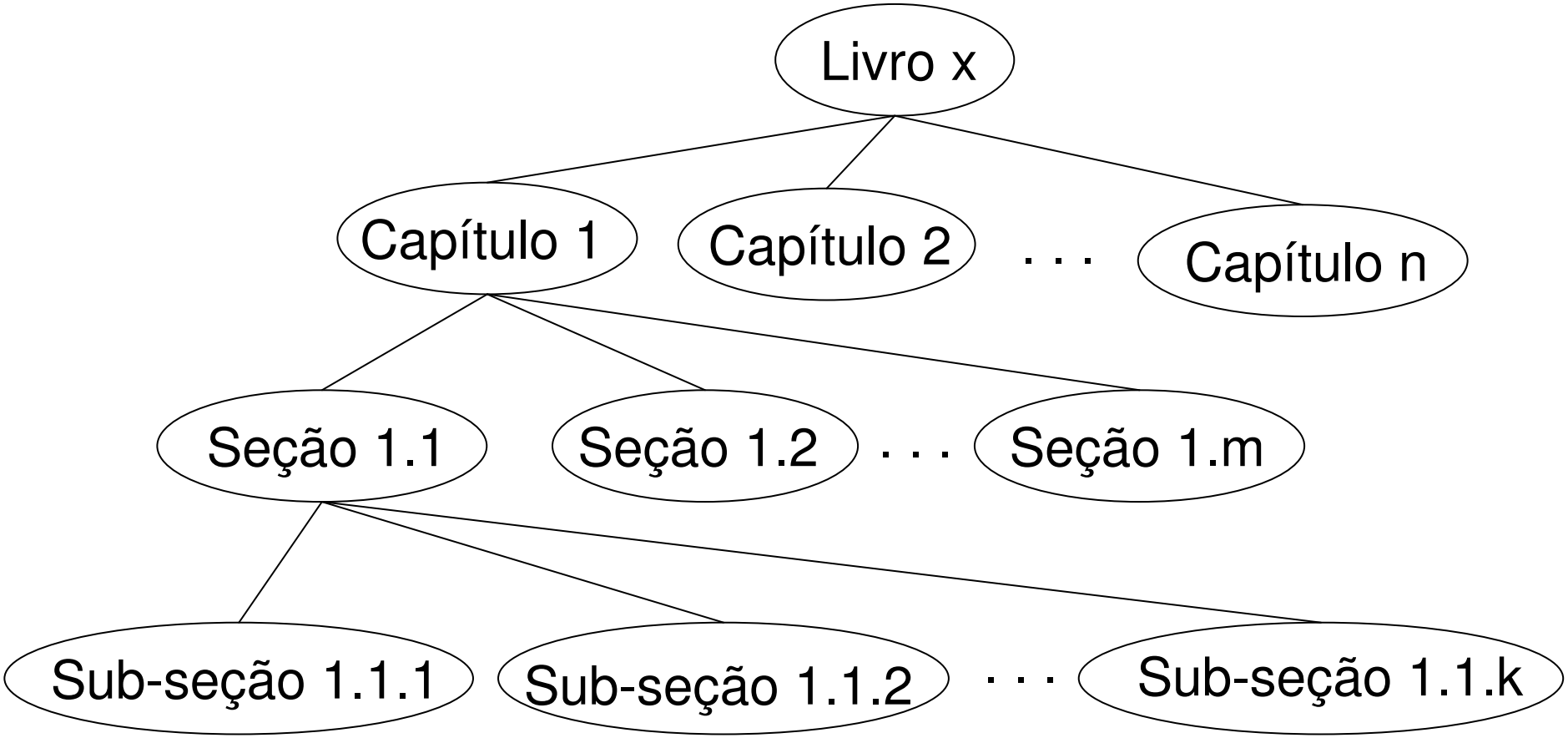
Árvore

ÁRVORE: Estrutura que armazena elementos de maneira hierárquica. Com exceção do elemento do topo, cada elemento da árvore tem um pai e zero ou mais filhos.



* o elemento topo é chamado de raiz da árvore, mas é desenhado como sendo o elemento mais alto.

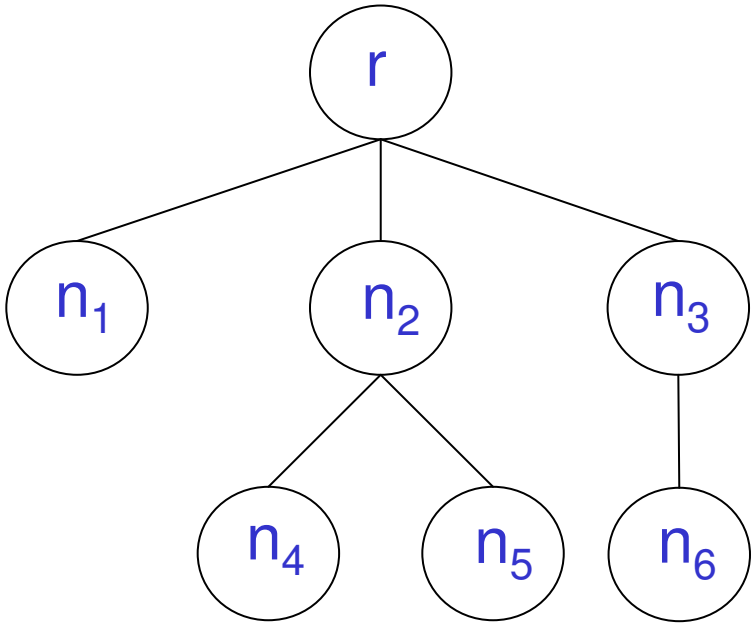
Exemplos de Árvores



Definição de Árvore

Uma árvore A é uma estrutura de dados tal que:

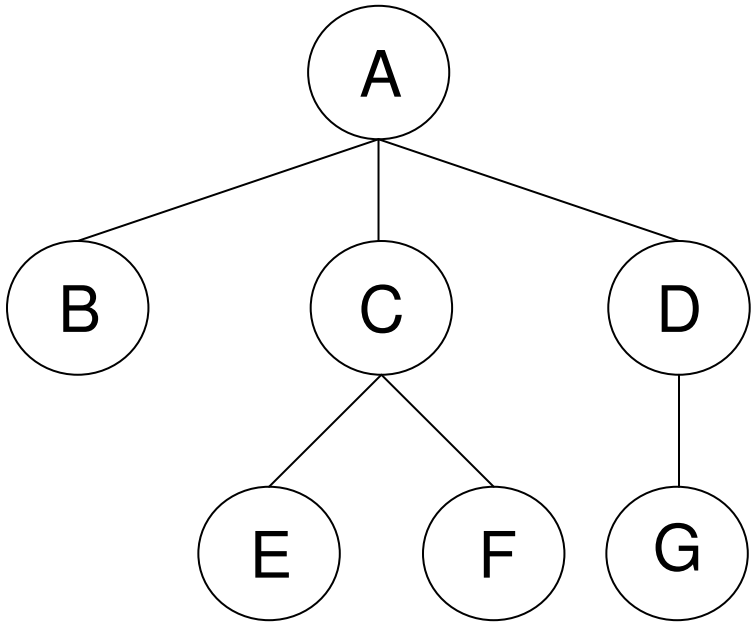
- se A não é vazia, existe um nodo raiz r;
- existem outros nodos n_1, n_2, ..., n_m, m >= 0, associados a r que são raízes de subárvores disjuntas A_1, A_2, ..., A_m.



Propriedades das Árvores

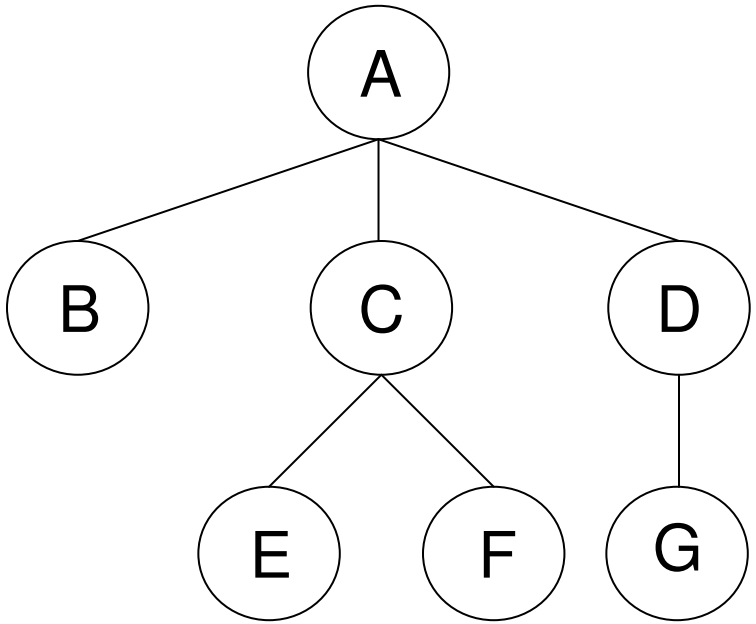
Grau de um nodo

- Denotado por *g(n)*
- É o número de subárvores de um nodo.



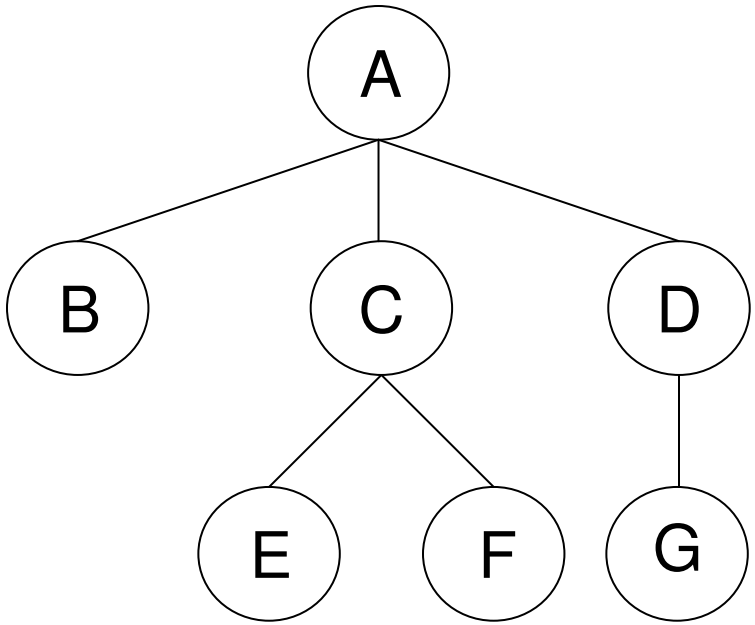
Nodo Folha e Nodo Interno

- *n* é um nodo folha ou terminal se $g(n) = 0$
- *m* é um nodo interno se $g(m) > 0$



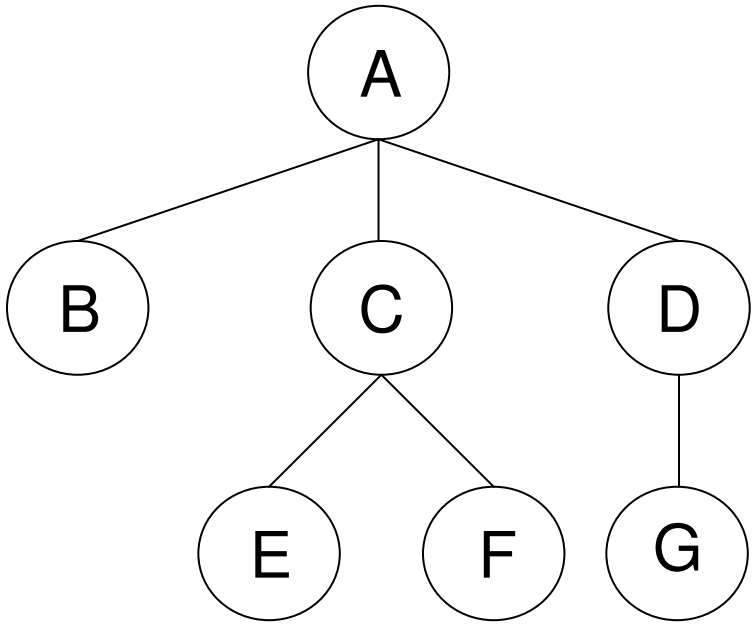
Caminho

- Um caminho *C* em uma árvore é uma seqüência de nodos relacionados na forma $C = n_1, n_2, ..., n_m, m > 0$, sendo n_i hierarquicamente superior a n_{i+1} .
- Comprimento de *C* = m – 1



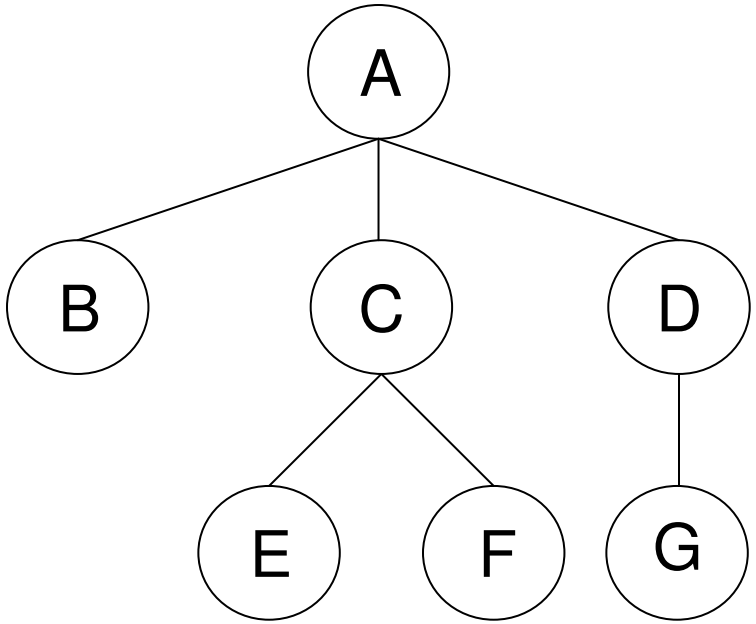
Profundidade de um Nodo

- Denotado por *p(n)*
- *p(n)* = *comprimento(C)*, sendo *C* = *r*, ..., *n*.



Nodo Pai e Nodo Filho

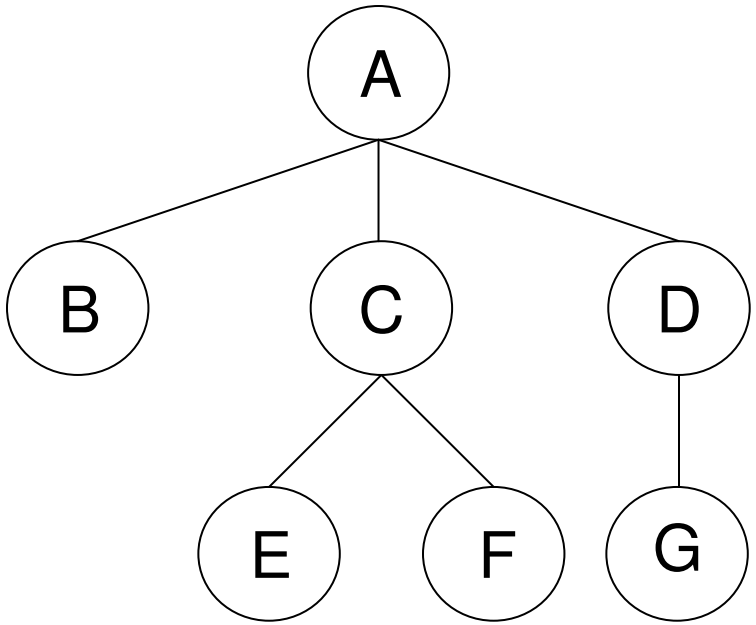
- Dados 2 nodos n_i *e* n_j, se existe um caminho a partir da raiz que passa por n_i e n_j e $p(n_j) = p(n_i) + 1$, então n_i é nodo pai de n_j e, conseqüentemente, n_j é nodo filho de n_i.



- Uma característica inerente às árvores é que todo nodo, exceto a raiz, tem um único nodo pai.

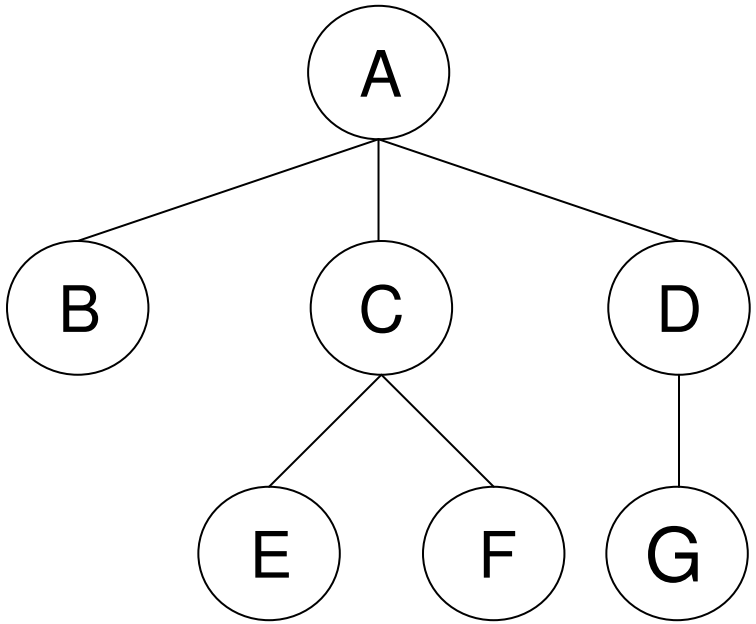
Nodo Irmão

- Dois nodos n_i *e* n_j são irmãos se possuem o mesmo nodo pai.



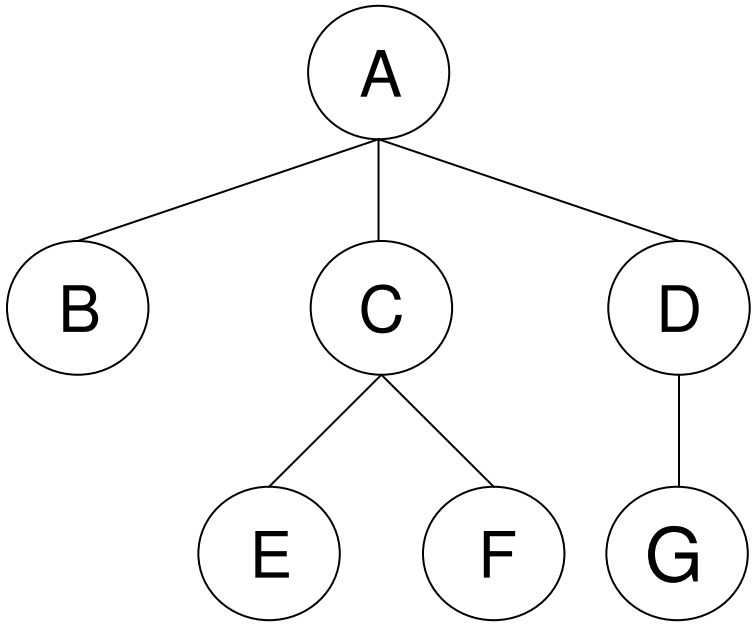
Nodo Ancestral e Descendente

- Dados 2 nodos n_i *e* n_j, se existe um caminho a partir da raiz que passa por n_i e n_j e $p(n_j) >= p(n_i)$, então n_i é ancestral de n_j e, conseqüentemente, n_j é descendente de n_i.



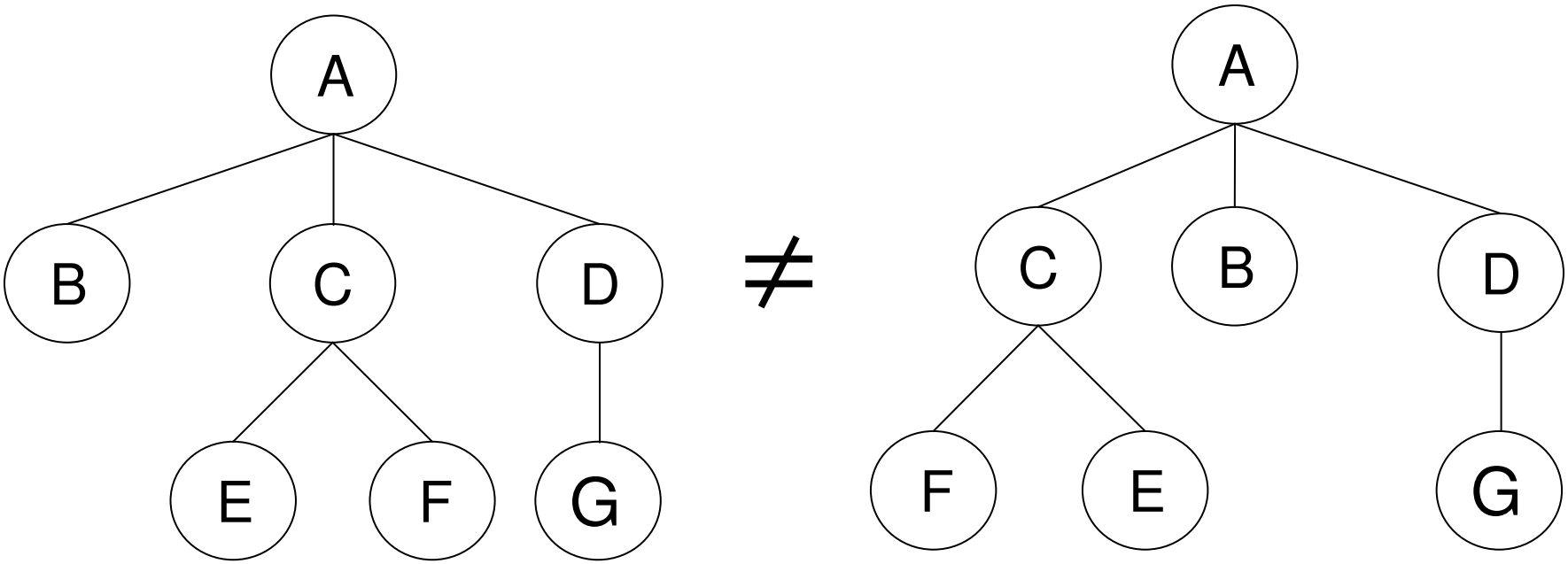
Altura de uma Árvore

- Denotado por *h(A)*
- $h(A) = \max(p(n_i))$, para $n_i \in A$.



Árvore Ordenada

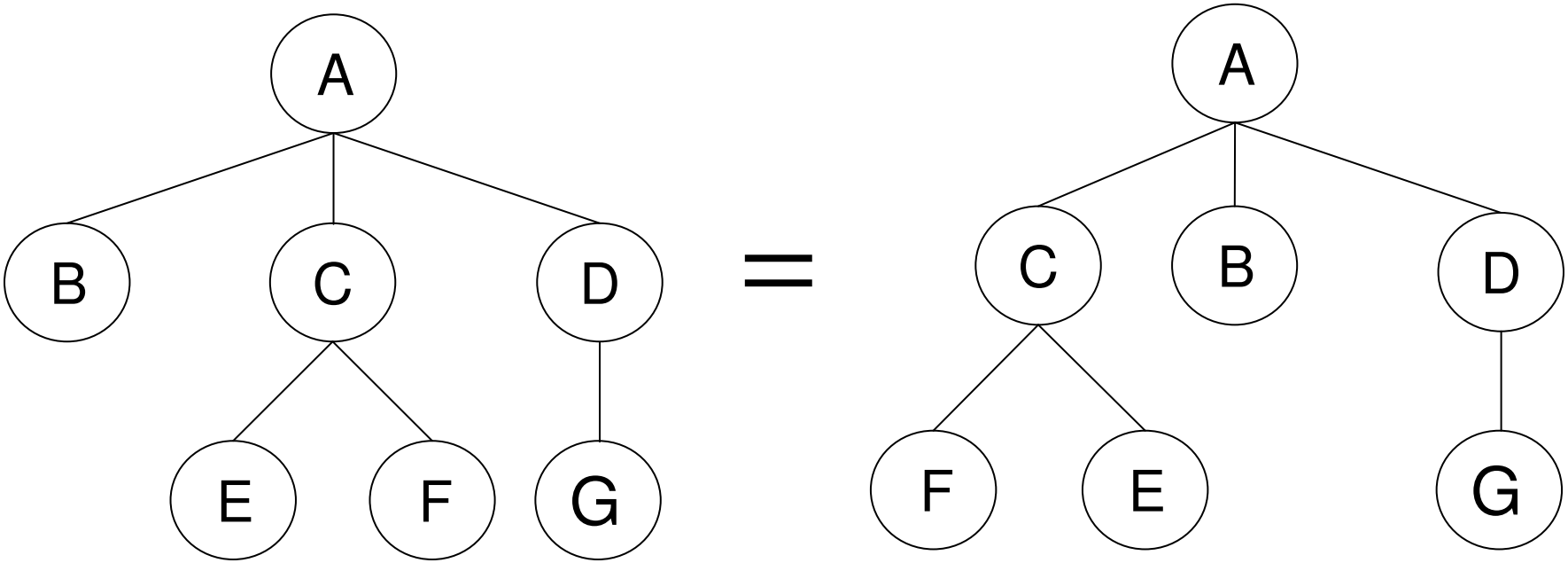
- Árvore na qual a ordem das subárvores é significativa.



- As árvores ordenadas ocorrem com maior freqüência e por isso são simplesmente denominadas árvores.

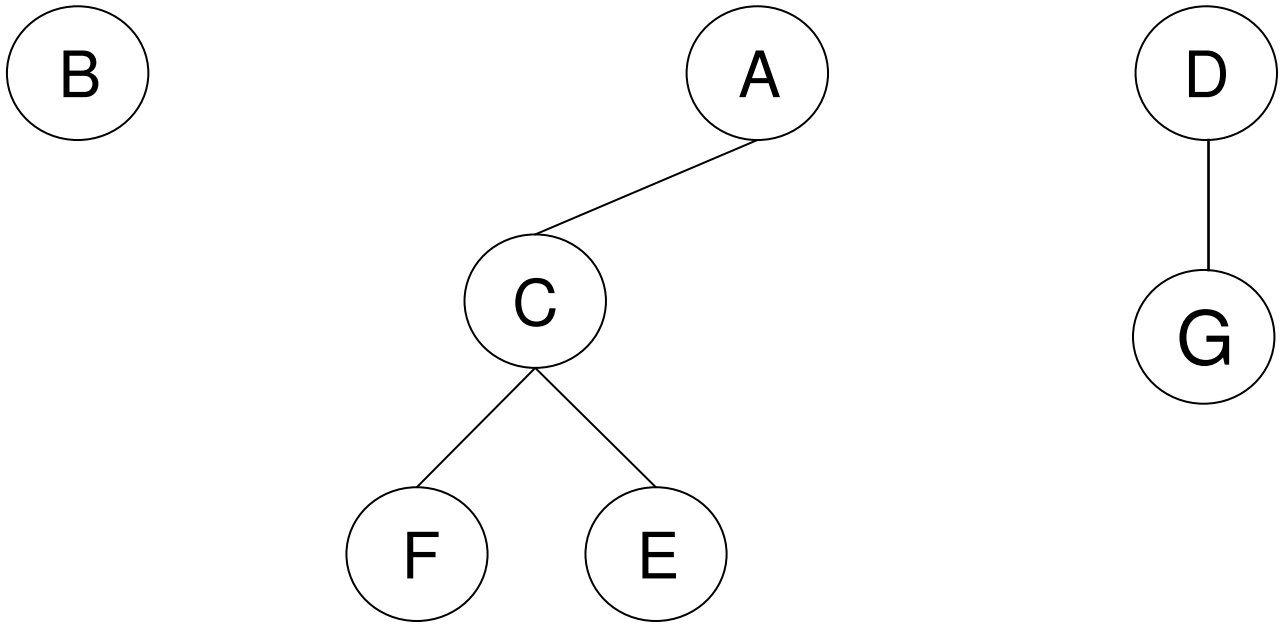
Árvore Orientada

- Árvore na qual a ordem das subárvores não é significativa.



Floresta

- Uma floresta $F = \{A_1, A_2, ..., A_n\}$ é um conjunto de $n >= 0$ árvores disjuntas.

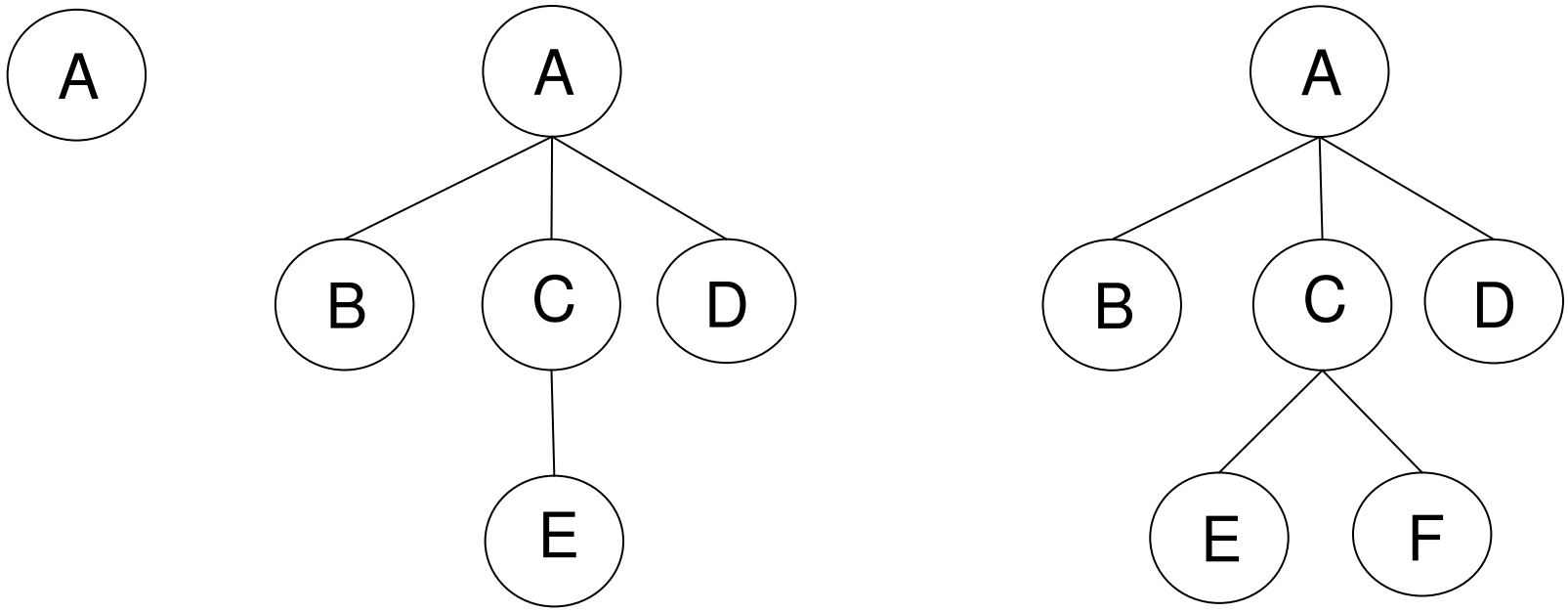


Árvore N-ária

- Uma árvore *A* é dita N-ária se

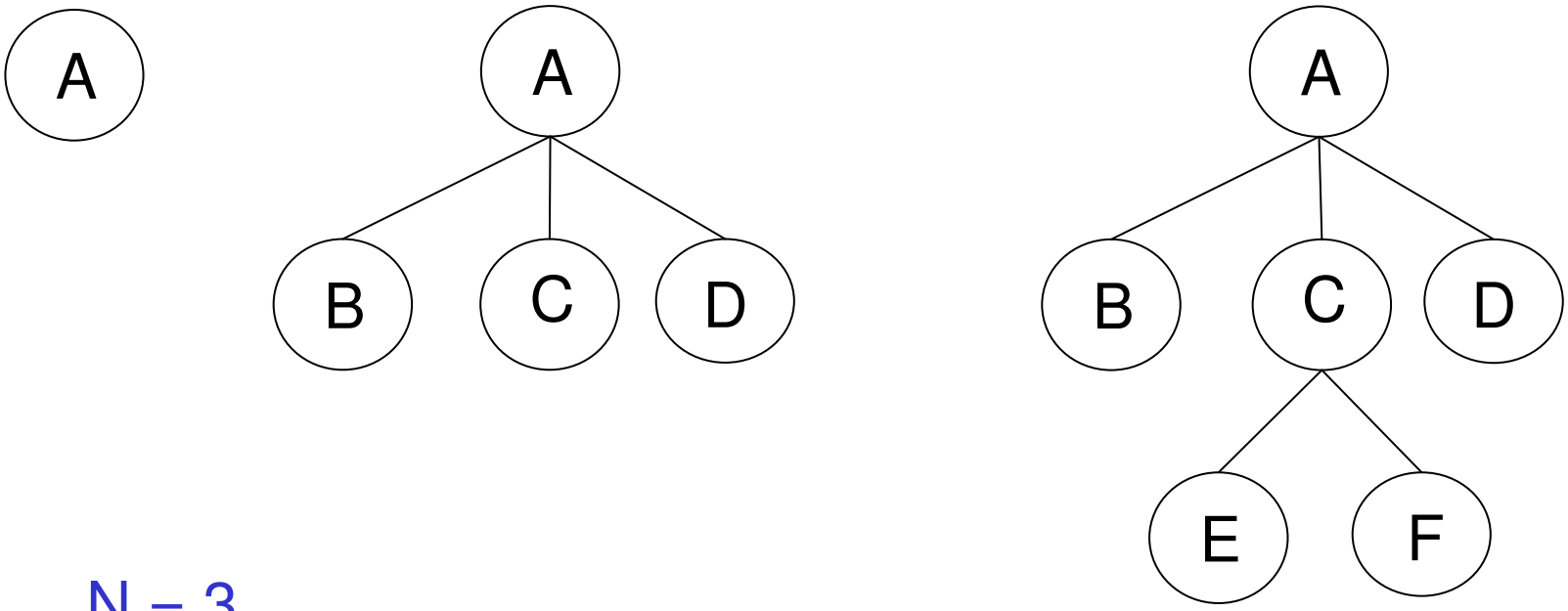
$$\forall\ n_i \in A\ (g(n_i) \in [0,N])$$

- Exemplo: N = 3.



Propriedade (P1)

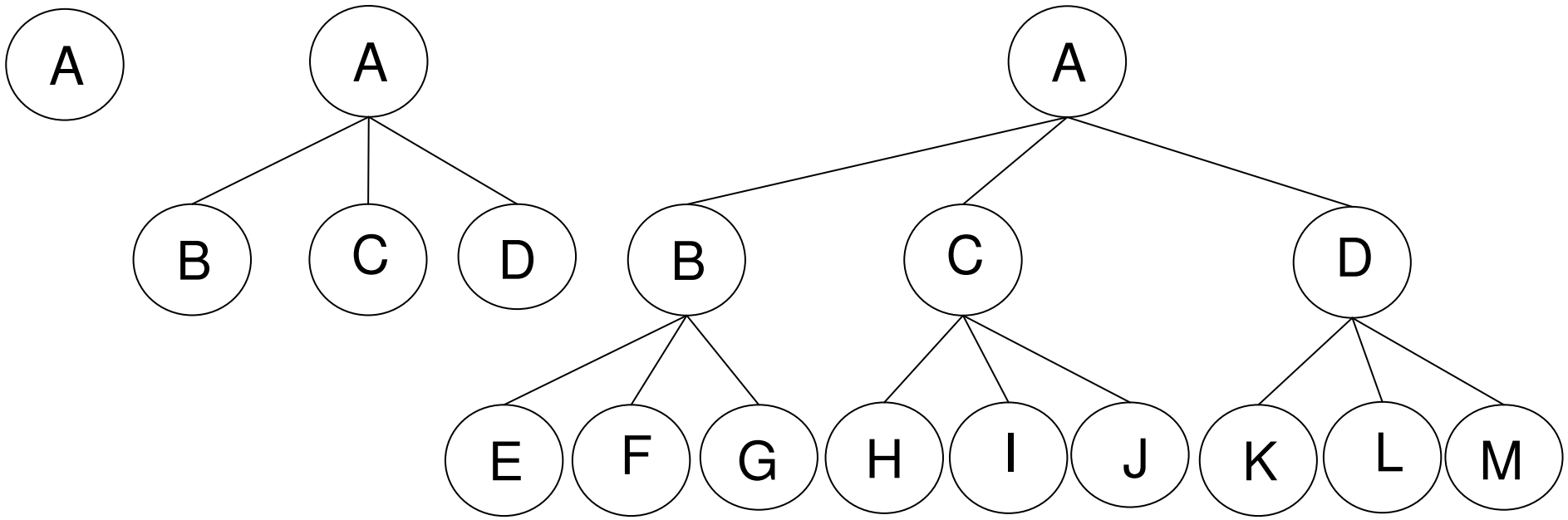
- "Em uma árvore N-ária *A* com altura *h(A)*, o número máximo de nodos folha é $N^{h(A)}$"



N = 3

Árvore Cheia

- Árvore N-ária *A* cujo número de nodos folha é $N^{h(A)}$"



N = 3

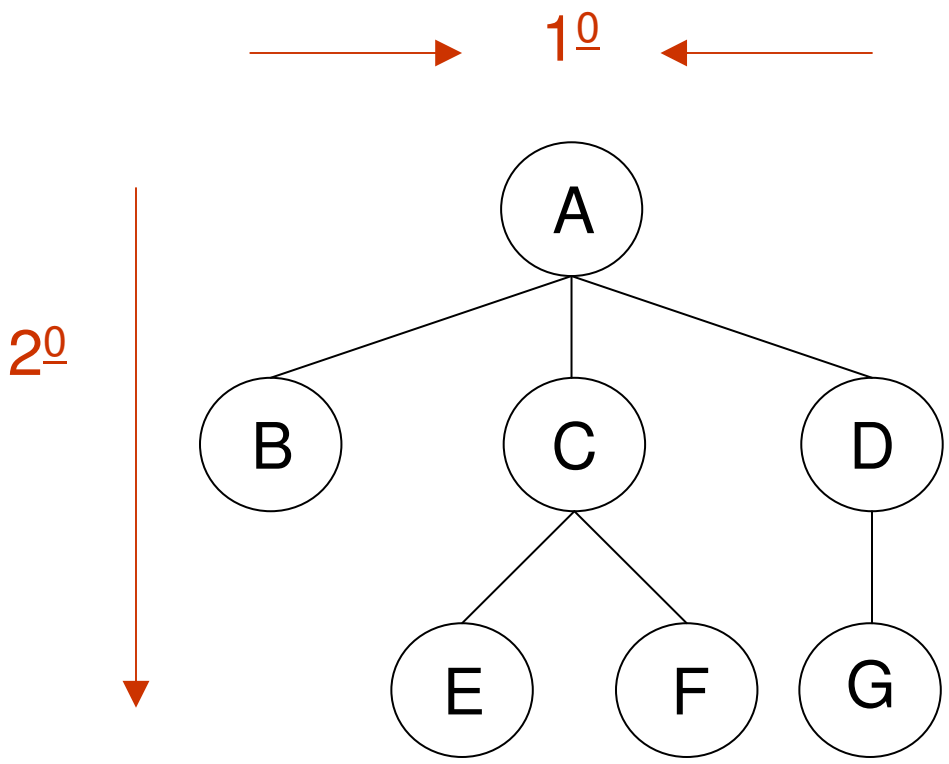
Caminhamento em Árvores

Caminhamento em Árvores

- Métodos de pesquisa em uma árvore para fins de consulta e/ou atualização de dados, de tal maneira que cada um dos nodos seja visitado exatamente uma vez
- Existem dois métodos:
  - busca em largura *(breadth-first-traversal)*
  - busca em profundidade *(depth-first-traversal)*
- O método de pesquisa a ser utilizado depende da intenção da aplicação.

Busca em Largura

- Percorre a árvore por ordem de profundidade



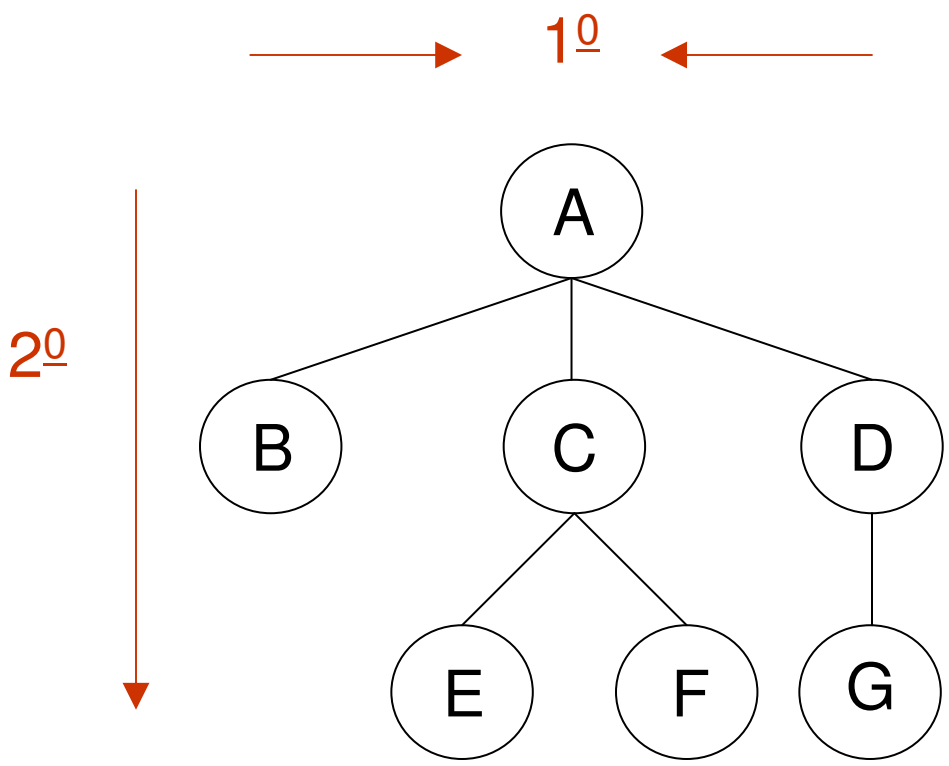
p(A) = 0

p(B) = p(C) = p(D) = 1

p(E) = p(F) = p(G) = 2

Busca em Largura

- Percorre a árvore por ordem de profundidade

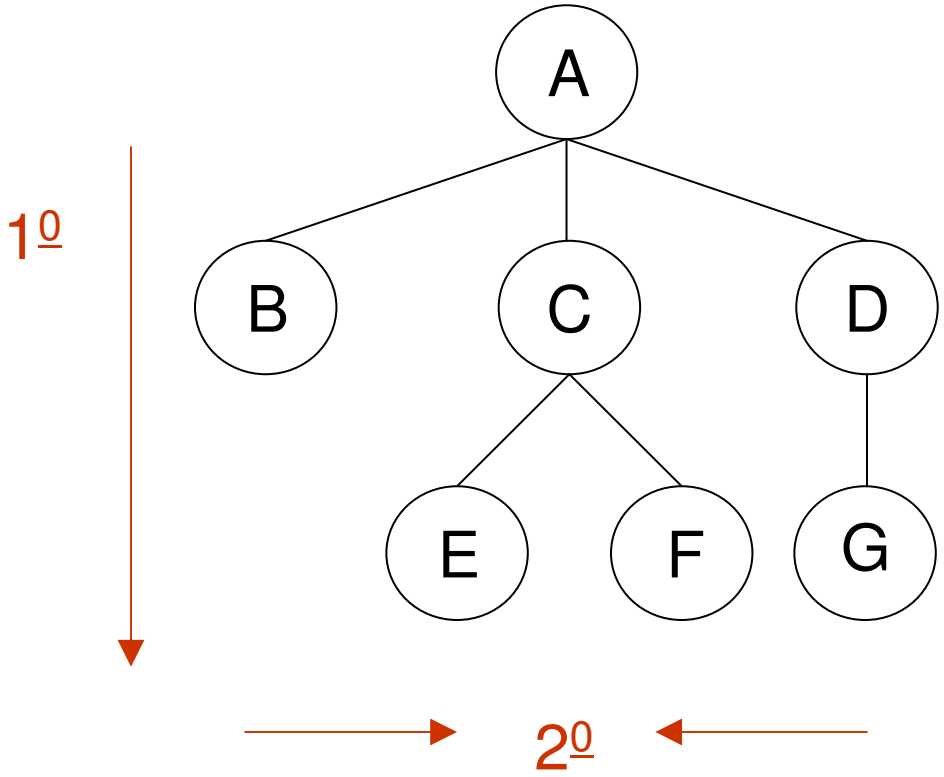


busca pela ESQ:
A-B-C-D-E-F-G

busca pela DIR:
A-D-C-B-G-F-E

Busca em Profundidade

- Percorre a árvore por ordem de sub-árvore (recursivamente)



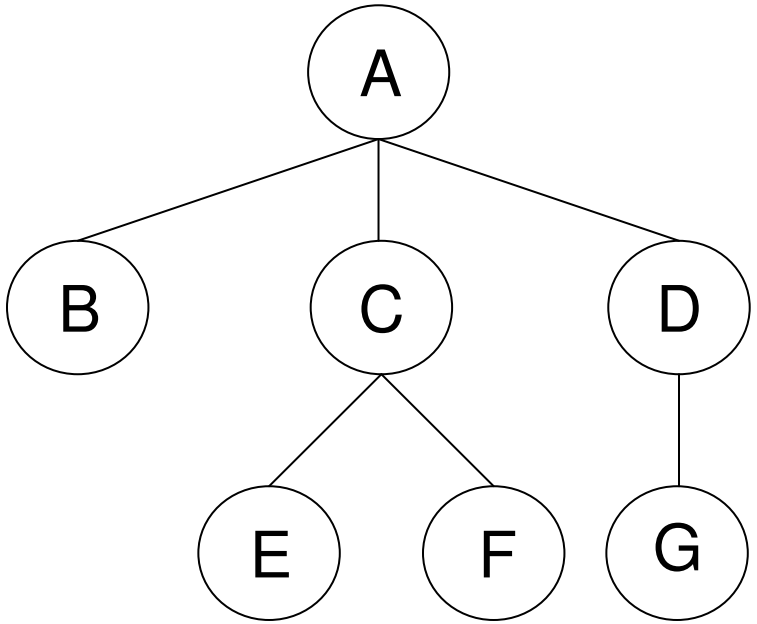
Busca em Profundidade

- Tipos de pesquisa em uma busca em profundidade:
  - pré-ordem (ou pré-fixada)
  - pós-ordem (ou pós-fixada)

Busca em Profundidade - Pré-Ordem

Passos:

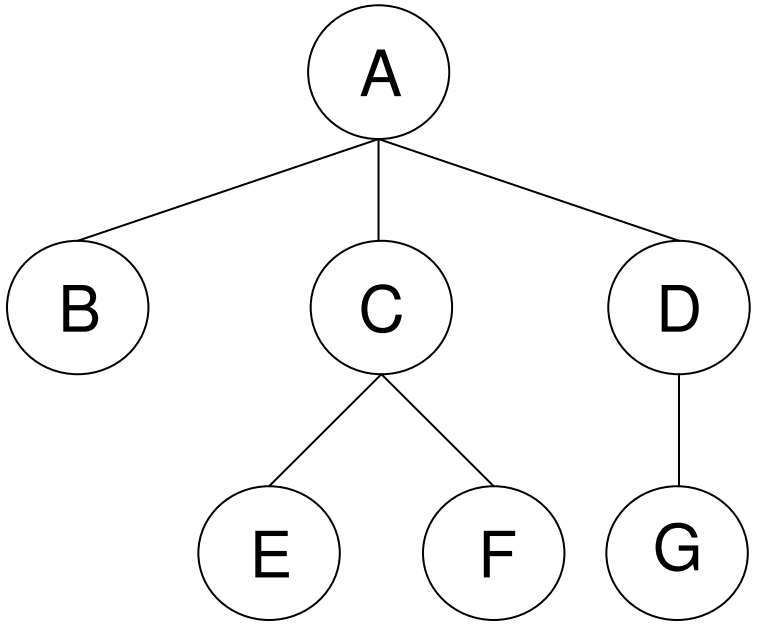
- Visita o nodo raiz
- Pesquisa em pré-ordem cada uma das subárvores (pela ESQ (*default*) ou pela DIR)



Busca em Profundidade - Pré-Ordem

Passos:

- Visita o nodo raiz
- Pesquisa em pré-ordem cada uma das subárvores (pela ESQ (*default*) ou pela DIR)



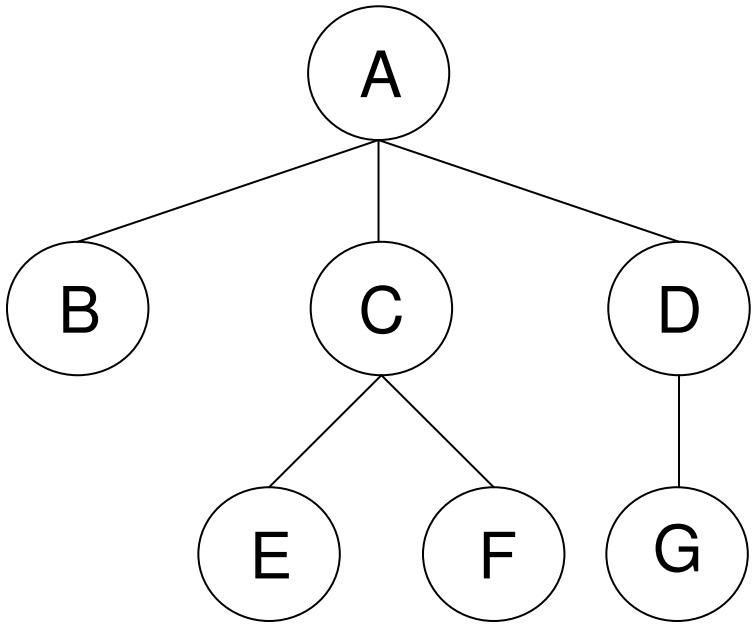
pré-ordem pela ESQ:
A-B-C-E-F-D-G

pré-ordem pela DIR:
A-D-G-C-F-E-B

Busca em Profundidade - Pós-Ordem

Passos:

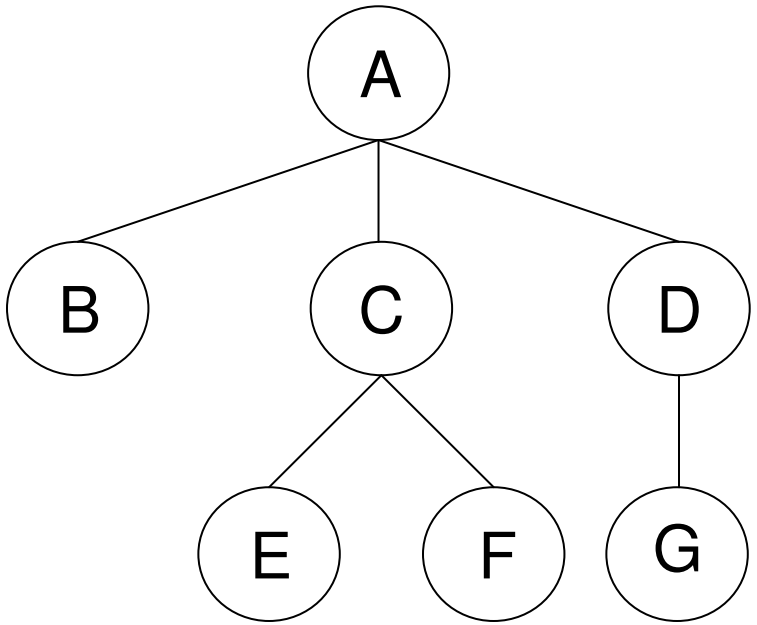
- Pesquisa em pós-ordem cada uma das subárvores (pela ESQ (*default*) ou pela DIR)
- Visita o nodo raiz



Busca em Profundidade - Pós-Ordem

Passos:

- Pesquisa em pós-ordem cada uma das subárvores (pela ESQ (*default*) ou pela DIR)
- Visita o nodo raiz

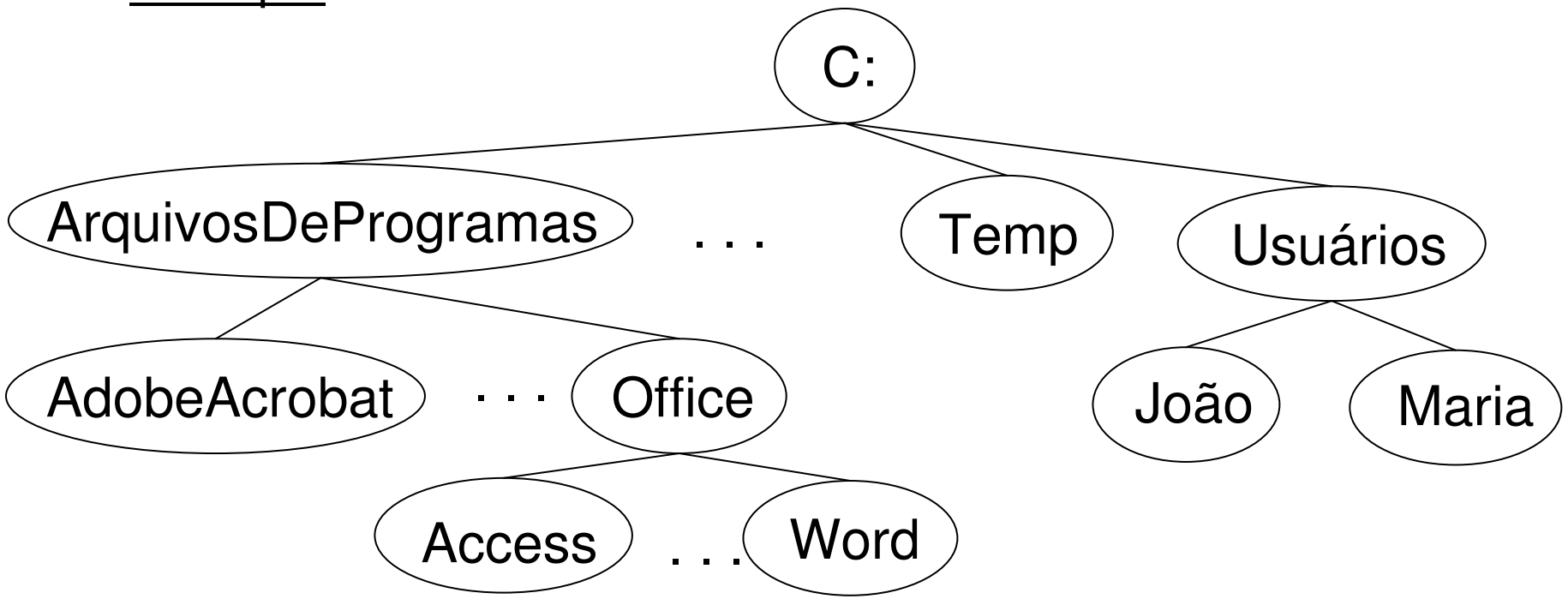


pós-ordem pela ESQ:
B-E-F-C-G-D-A

pós-ordem pela DIR:
G-D-F-E-C-B-A

Pesquisa em Árvores

- Exemplo:

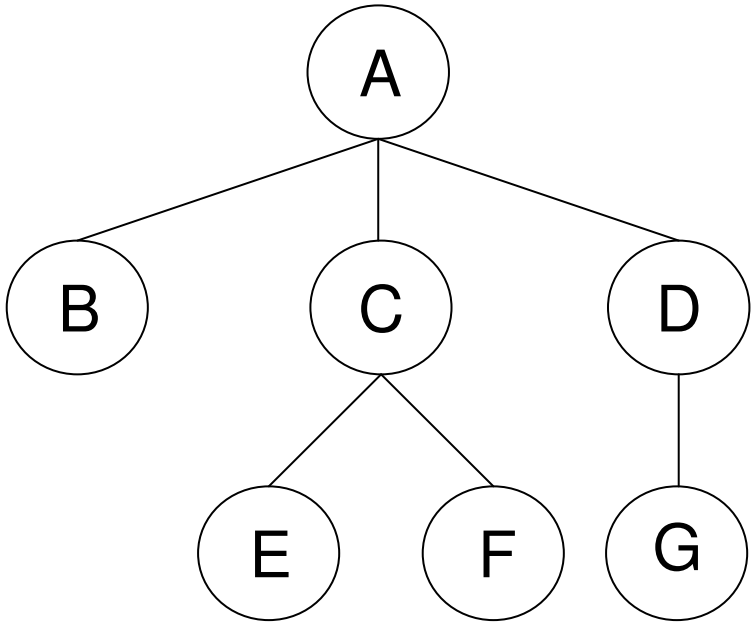


C: (ArquivosDePrograma (AdobeAcrobat; ...; Office (Access; ...; Word)); Temp; Usuários (João; Maria))

↳ Pesquisa em Pré-Ordem!

Modelagem Física de Árvores

- Alternativas de implementação:
- Array
- Encadeamento



Árvore com Encadeamento

Árvore com Encadeamento

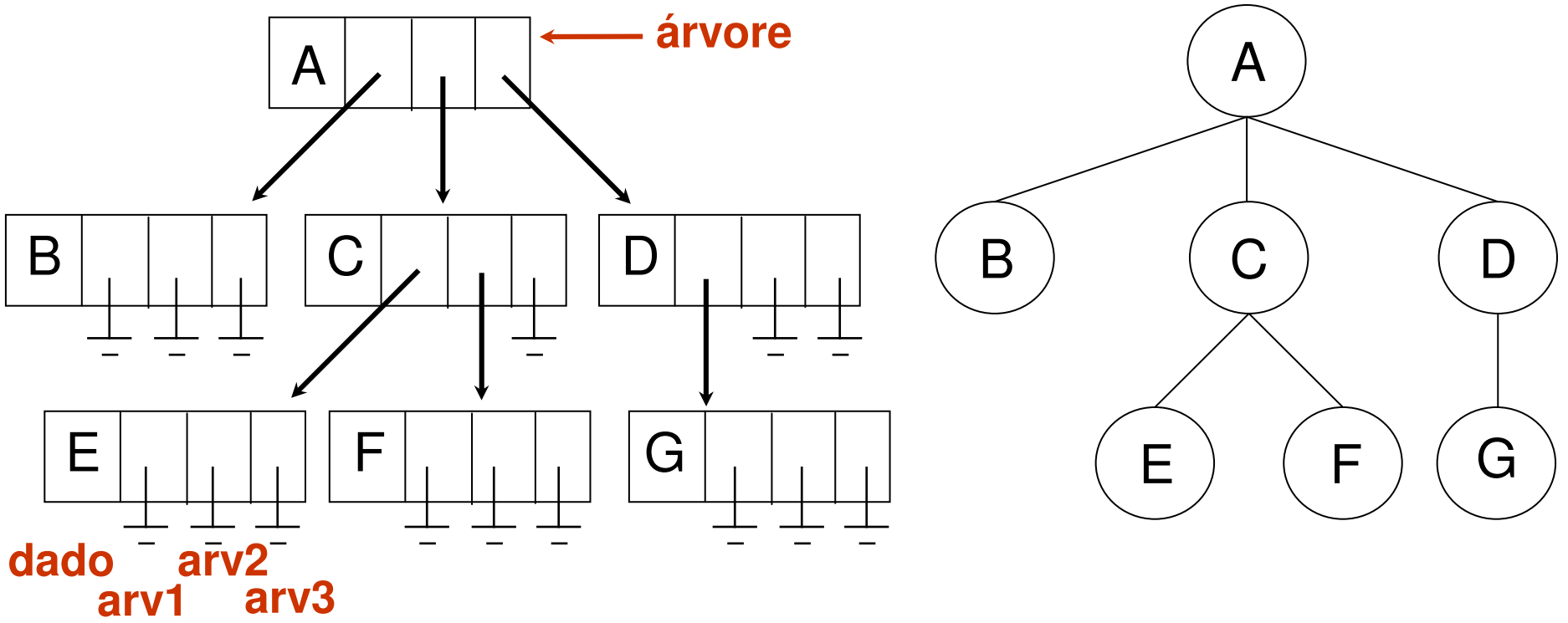
- Maior flexibilidade de definição da estrutura.

- Menor complexidade nas operações de manipulação dos dados.

Árvore com Encadeamento

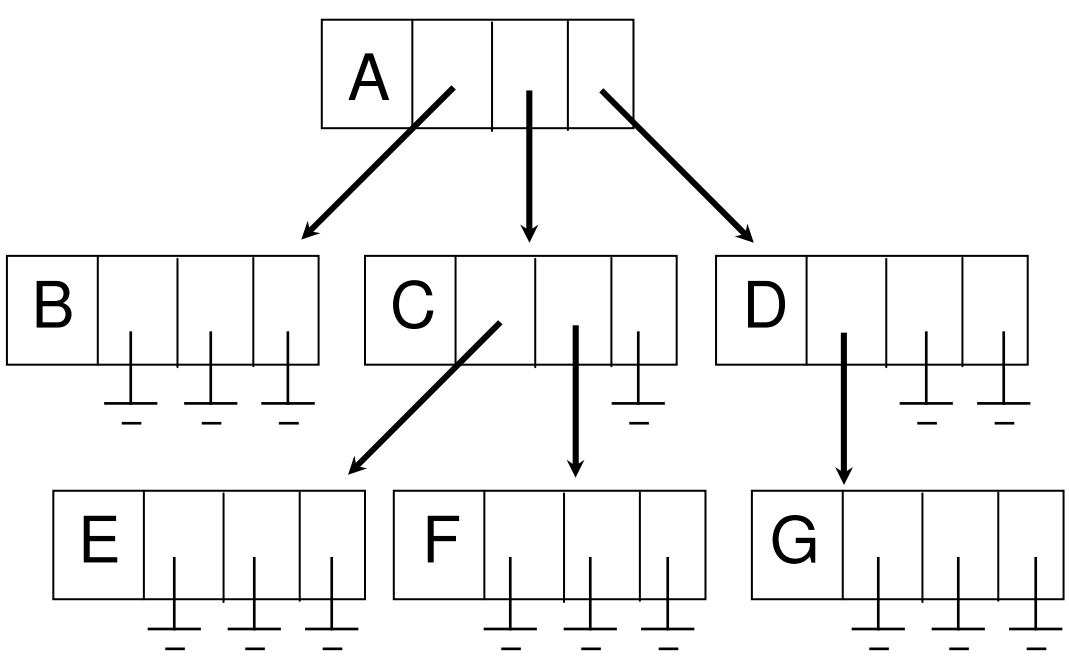
Alternativa 1: representação de árvores N-árias

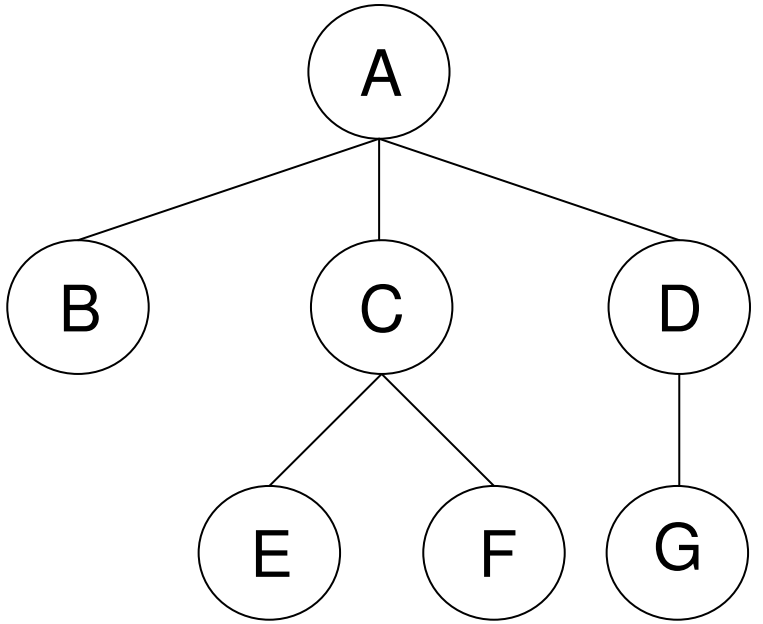
Exemplo para N = 3



Árvore com Encadeamento

Alternativa 1: representação de árvores N-árias

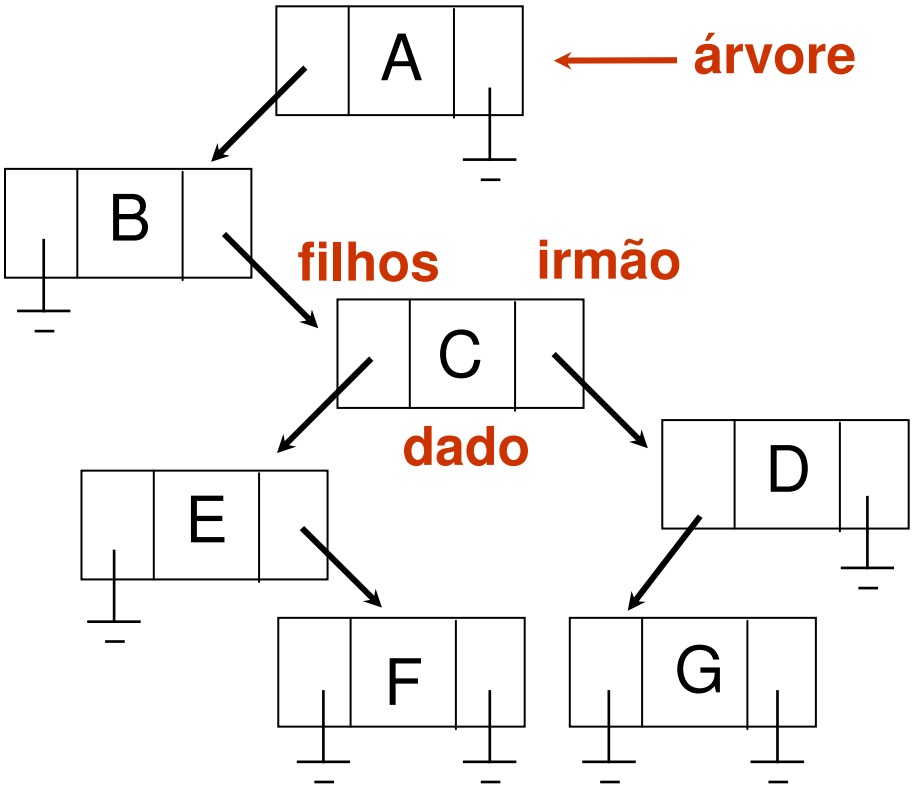


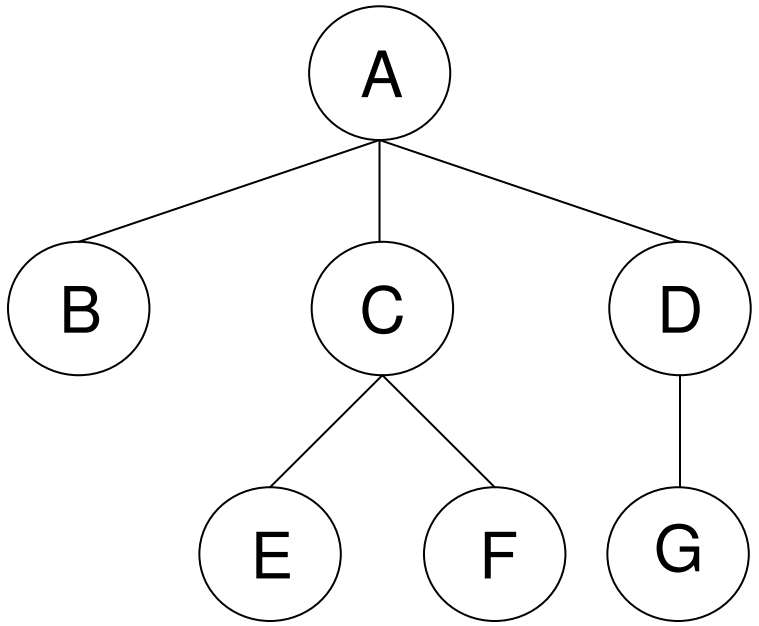


Desvantagem: *N* atributos de referência, sendo que vários deles podem não ter valor associado.

Árvore com Encadeamento

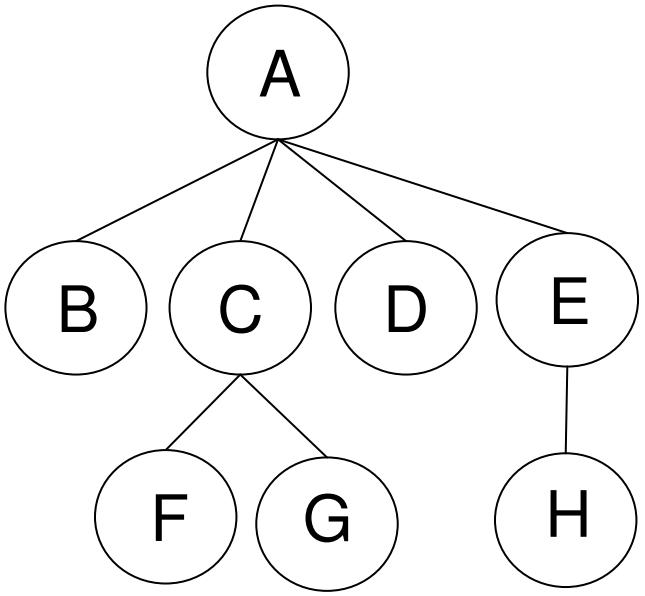
Alternativa 2: representação estilo árvore binária.





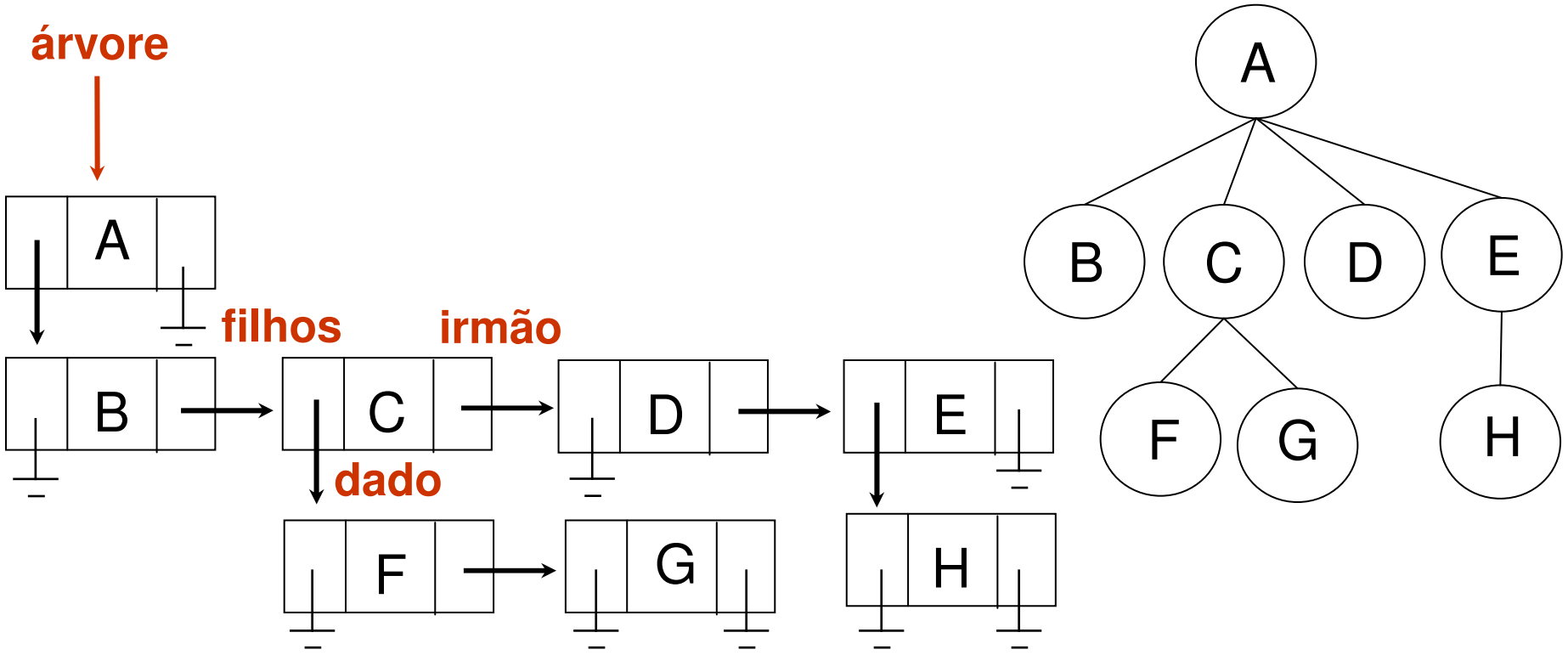
Árvore com Encadeamento

Alternativa 2: representação estilo árvore binária.



Árvore com Encadeamento

Alternativa 2: representação estilo árvore binária.



Árvore com Array

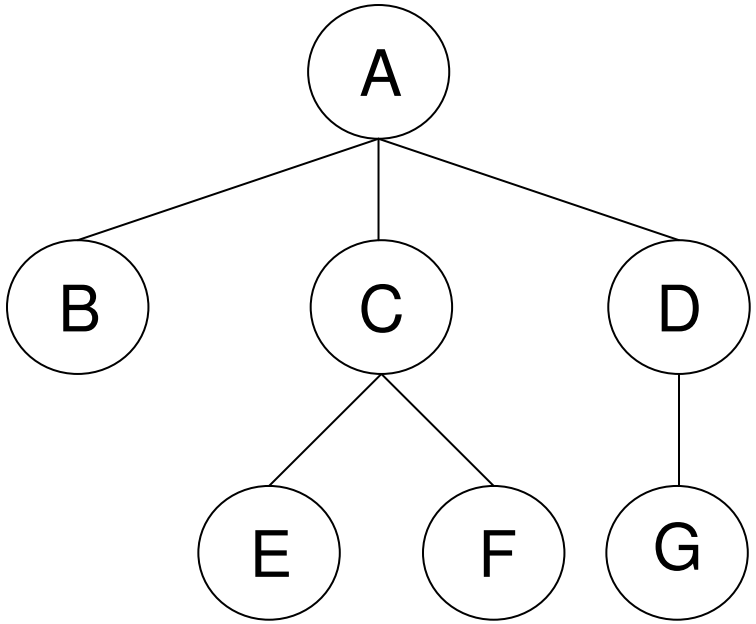
Árvore com Array

- Alternativa geralmente não adotada
  - array é bom para manter uma seqüência de dados, não uma hierarquia de dados.
- Operações de inserção, exclusão e certas consultas são mais complexas do que na alternativa por encadeamento

Árvore com Array

Alternativa 1: (nodo, número de filhos)

A	3	B	0	C	2	E	0	F	0	D	1	G	0



Árvore com Array

Alternativa 1: (nodo, número de filhos)

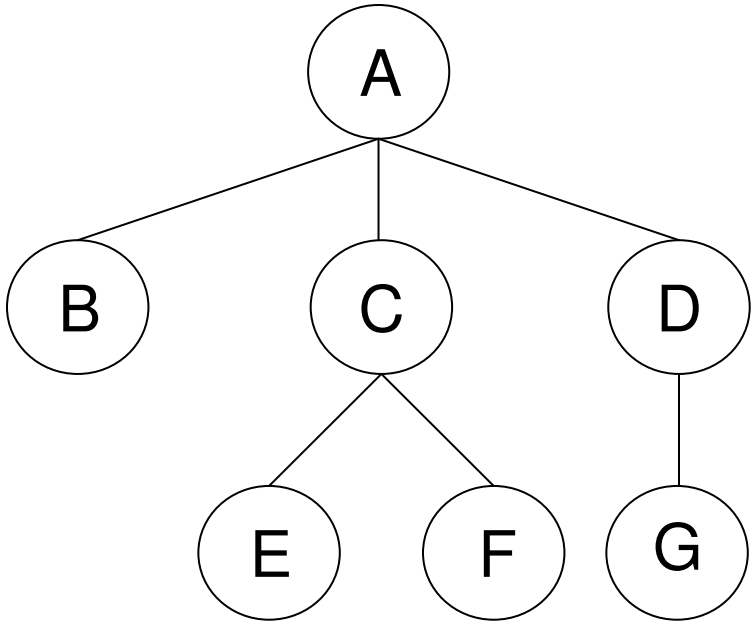
A	3	B	0	C	2	E	0	F	0	D	1	G	0

- Desvantagens:
  - dado adicional (número filhos)
  - inserção / remoção:
    - achar a posição correta para inserir/remover um nodo
    - deslocamento
  - consultar o pais de um nodo
  - consultar os filhos de um nodo

Árvore com Array

Alternativa 2: (nodo, posição do nodo pai)

0	1	2	3	4	5	6	7	8	9	10	11	12	13
A	-1	B	0	C	0	E	4	F	4	D	0	G	10



Árvore com Array

Alternativa 2: (nodo, posição do nodo pai)

0	1	2	3	4	5	6	7	8	9	10	11	12	13
A	-1	B	0	C	0	E	4	F	4	D	0	G	10

- Desvantagens:
  - dado adicional (posição pai)
  - inserção:
    - buscar pai para descobrir o valor da sua posição
  - inserção / exclusão:
    - deslocamento e atualização de referências aos nodos pais no vetor
  - consultar os filhos de um nodo