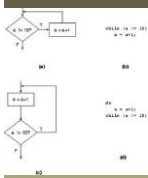
Estruturas Dinâmicas Básicas na Linguagem C (Filas e Árvores)

Edkallenn Lima

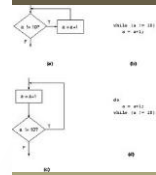
Agente de Polícia Federal

Chefe do Núcleo de Tecnologia da Informação da Polícia Federal- NTI/AC



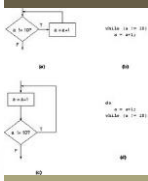
Estruturas de Dados

- Prof. Edkallenn Lima
- edkallenn@bol.com.br (somente para dúvidas)
- Blogs:
  - http://professored.wordpress.com (Computador de Papel – O conteúdo da forma)
  - http://professored.tumblr.com/ (Pensamentos Incompletos)
  - http://umcientistaporquinzena.tumblr.com/
  - http://eulinoslivros.tumblr.com/
  - http://linabiblia.tumblr.com/
- Redes Sociais:
  - http://www.facebook.com/edkallenn
  - http://twitter.com/edkallenn
  - https://plus.google.com/u/0/113248995006035389558/posts
  - Instagram: http://instagram.com/edkallenn ou @edkallenn
  - Foursquare: https://pt.foursquare.com/edkallenn
- Telefones:
  - 68 8401-2103 (VIVO) e 68 3212-1211.
- Os exercícios devem ser enviados SEMPRE para o e-mail: edkevan@gmail.com ou para o e-mail: edkallenn.lima@uninorteac.com.br



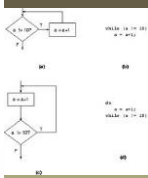
Agenda

- Filas
  - Introdução
  - Interface do tipo fila
  - Implementação de fila com vetor
  - Implementação de fila com lista
  - Fila dupla
  - Implementação de fila dupla com lista
- Árvores
  - Introdução
  - Árvores binárias
    - Representação em C
    - Ordens de percurso em árvores binárias
    - Altura de uma árvore
  - Árvores com número variável de filhos
    - Representação em C
    - Tipo abstrato de dado
    - Altura da árvore
    - Topologia binária



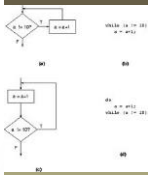
Fila - Introdução

- Fila
  - um novo elemento é inserido no final da fila e um elemento é retirado do início da fila
  - fila = “o primeiro que entra é o primeiro que sai” (FIFO)
  - pilha = “o último que entra é o primeiro que sai” (LIFO)



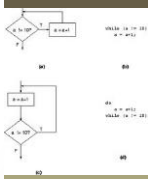
Interface do tipo fila

- Implementações:
  - usando um vetor
  - usando uma lista encadeada
  - simplificação:
    - fila armazena valores reais



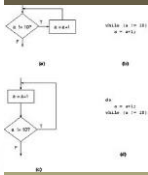
Interface do tipo fila

- Interface do tipo abstrato Fila*: fila.h*
  - função fila_cria
    - aloca dinamicamente a estrutura da fila
    - inicializa seus campos e retorna seu ponteiro
  - função fila_insere e função fila_retira
    - insere e retira, respectivamente, um valor real na fila
  - função fila_vazia
    - informa se a fila está ou não vazia
  - função fila_libera
    - destrói a fila, liberando toda a memória usada pela estrutura



```
typedef struct fila Fila;

Fila* fila_cria (void);

void fila_insere (Fila* f, float v);

float fila_retira (Fila* f);

int fila_vazia (Fila* f);

void fila_libera (Fila* f);
```

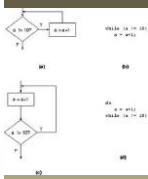
tipo Fila:
- definido na interface
- depende da implementação do struct fila



Implementação de fila com vetor

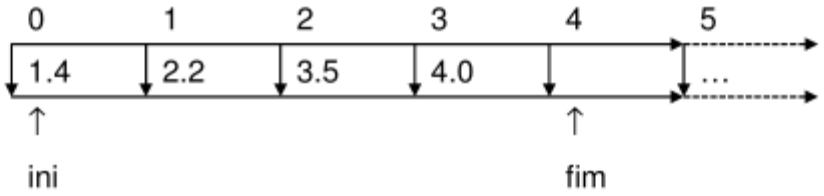
- Implementação de fila com vetor
  - vetor (vet) armazena os elementos da fila
  - estrutura de fila:

```
#define N  100      /* número máximo de elementos */

struct fila {
   int n;           /* número de elementos na fila */
   int ini;         /* posição do próximo elemento a ser retirado da fila */
   float vet[N];
};
```

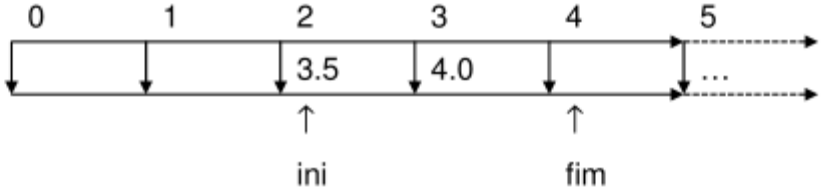


Implementação de fila com vetor

- Implementação de fila com vetor
  - processo de inserção e remoção em extremidades opostas da fila faz com que a fila "ande" no vetor
    - inserção dos elementos 1.4, 2.2, 3.5, 4.0

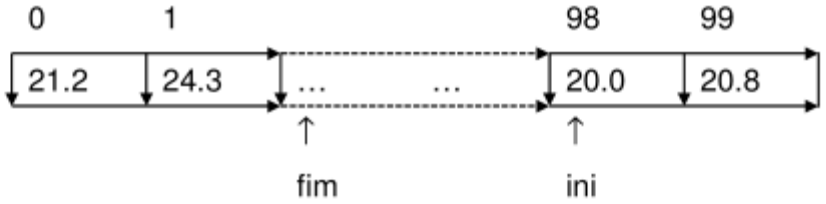


    - remoção de dois elementos



Implementação de fila com vetor

- Implementação de fila com vetor
  - incremento das posições do vetor de forma “circular”:
    - se o último elemento da fila ocupa a última posição do vetor, os novos elementos são inseridos a partir do início do vetor
    - exemplo:
      - quatro elementos, 20.0, 20.8, 21.2 e 24.3
      - distribuídos dois no fim do vetor e dois no início



Implementação de fila com vetor

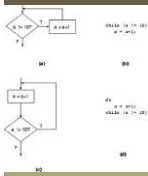
- Implementação de fila com vetor
  - incremento das posições do vetor de forma “circular”:
    - usa o operador módulo “%”
  - parâmetros da fila:

    n = número de elementos na fila

    ini = posição do próximo elemento a ser retirado da fila

    fim = posição onde será inserido o próximo elemento

```
...
fim = (ini+n)%N
...
```

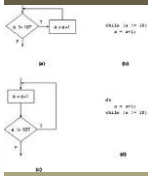


Implementação de fila com vetor

- função fila_cria
  - aloca dinamicamente um vetor
  - inicializa a fila como sendo vazia (número de elementos = 0)

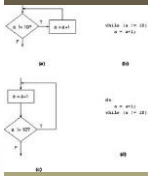
tipo Fila: definido na interface
struct fila: determina a implementação

```
Fila* fila_cria (void)
{
  Fila* f = (Fila*) malloc(sizeof(Fila));
  f->n = 0;          /* inicializa fila como vazia          */
  f->ini = 0;        /* escolhe uma posição inicial         */
  return f;
}
```

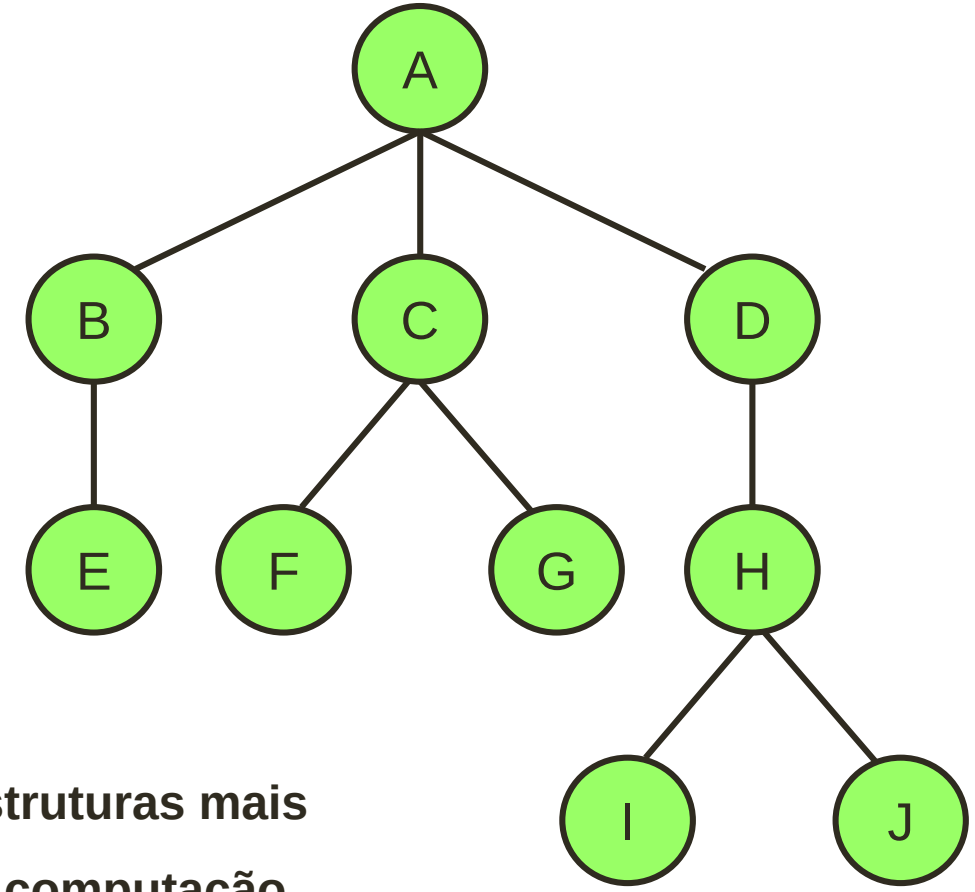


Árvores

Estudo da estrutura de dados denominada árvore, onde os relacionamentos lógicos entre os dados representam dependência de hierarquia ou composição



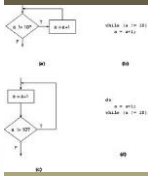
Árvores



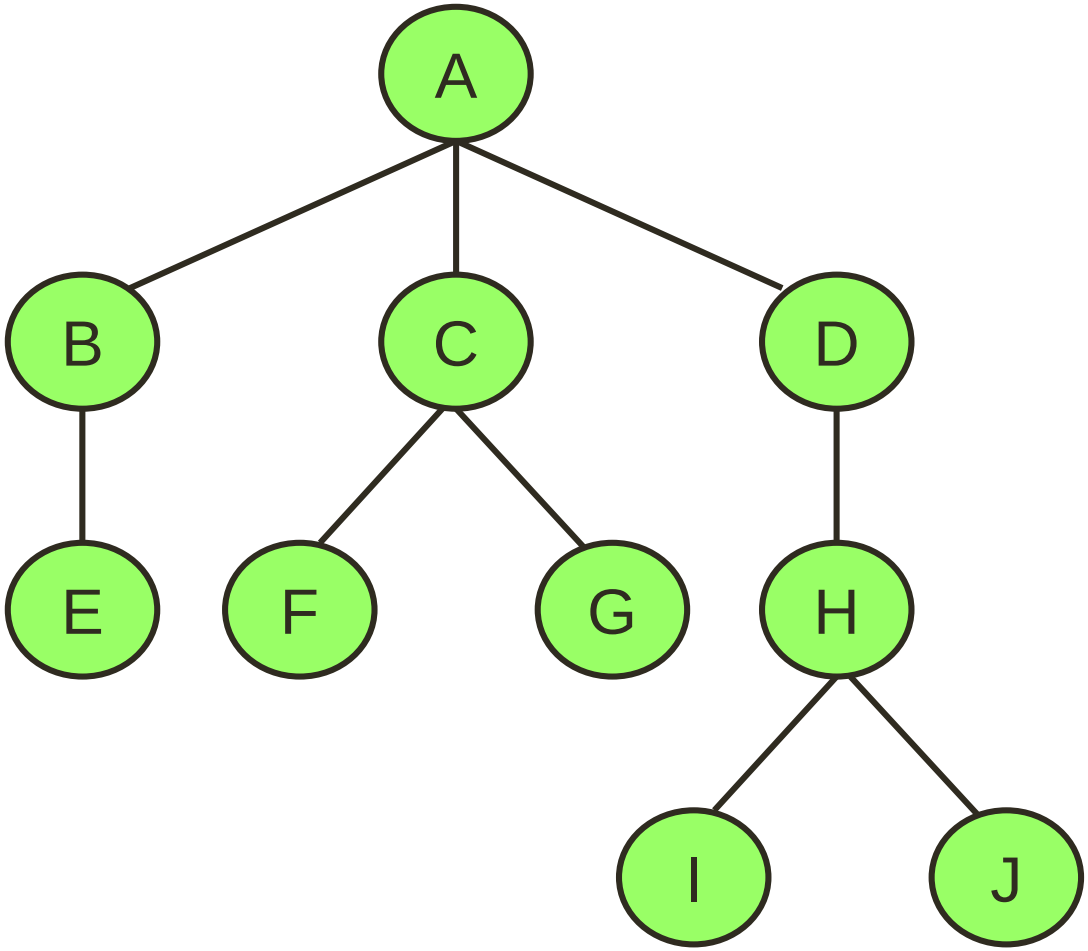
Constituem uma das estruturas mais importantes da área de computação, inclusive em aplicações

Árvores

Conceitos básicos



Representação gráfica



Outras formas de representação

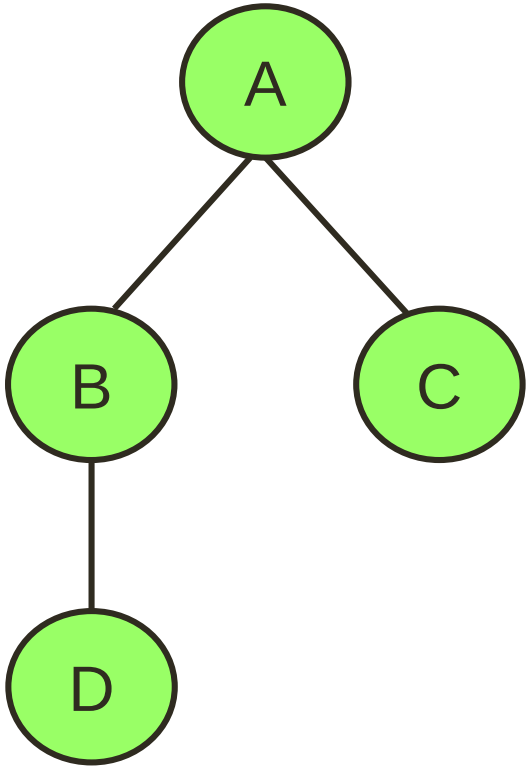
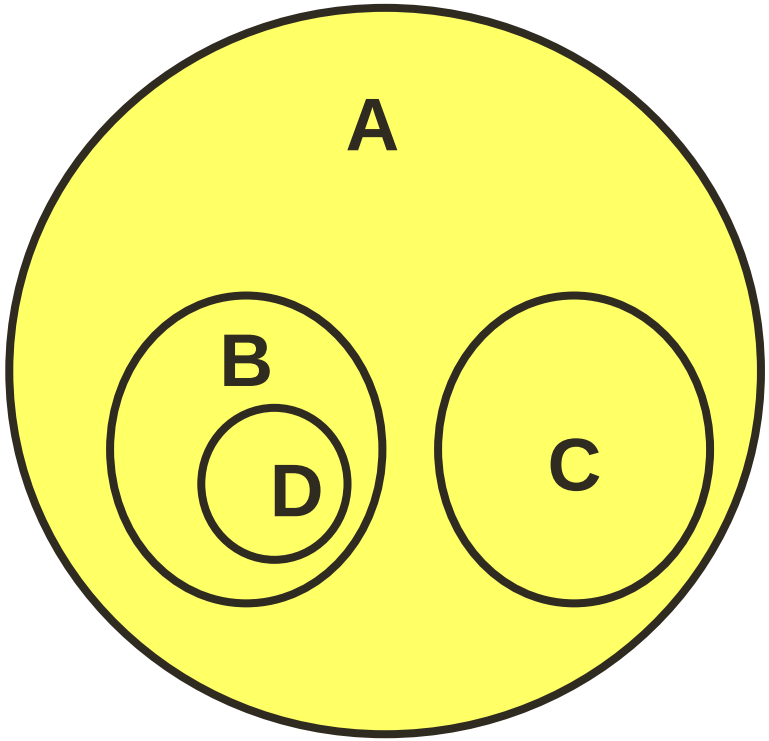
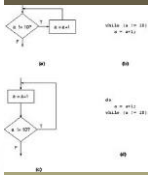


Diagrama de inclusão





Outras formas de representação

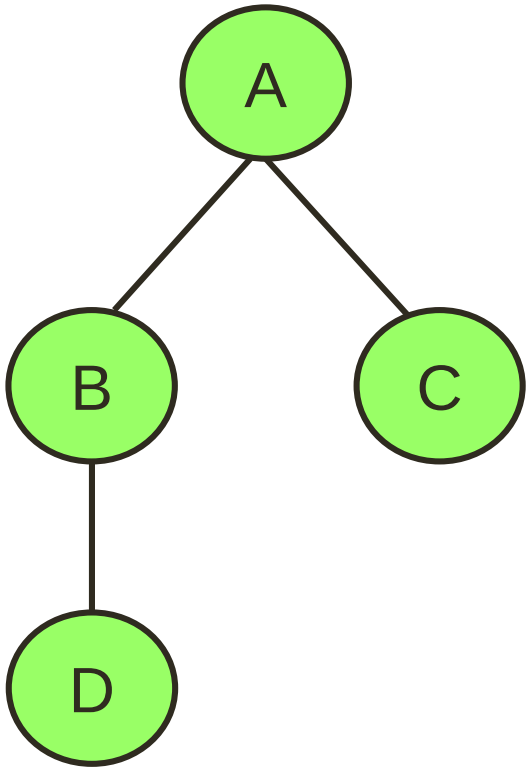
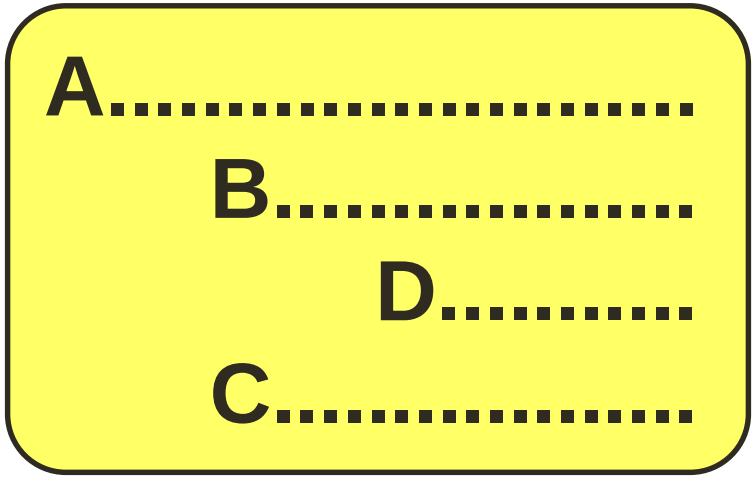
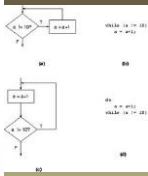
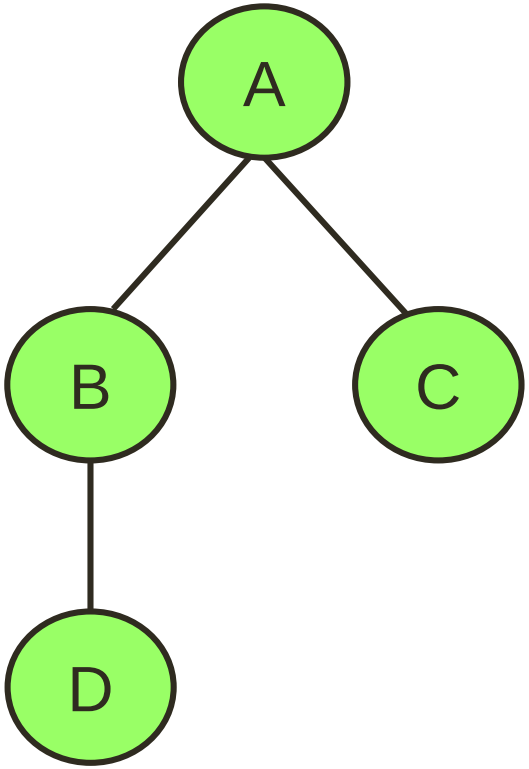


Diagrama de barras





Outras formas de representação

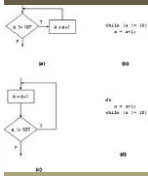


Numeração em níveis

1A; 1.1B; 1.1.1D; 1.2C

Representação aninhada

(A (B (D) , C))



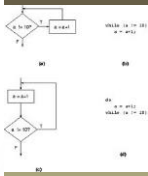
Relacionamento lógico

Hierarquia ou subordinação

onde um subconjunto dos componentes é subordinado a outro

Diferentes significados:

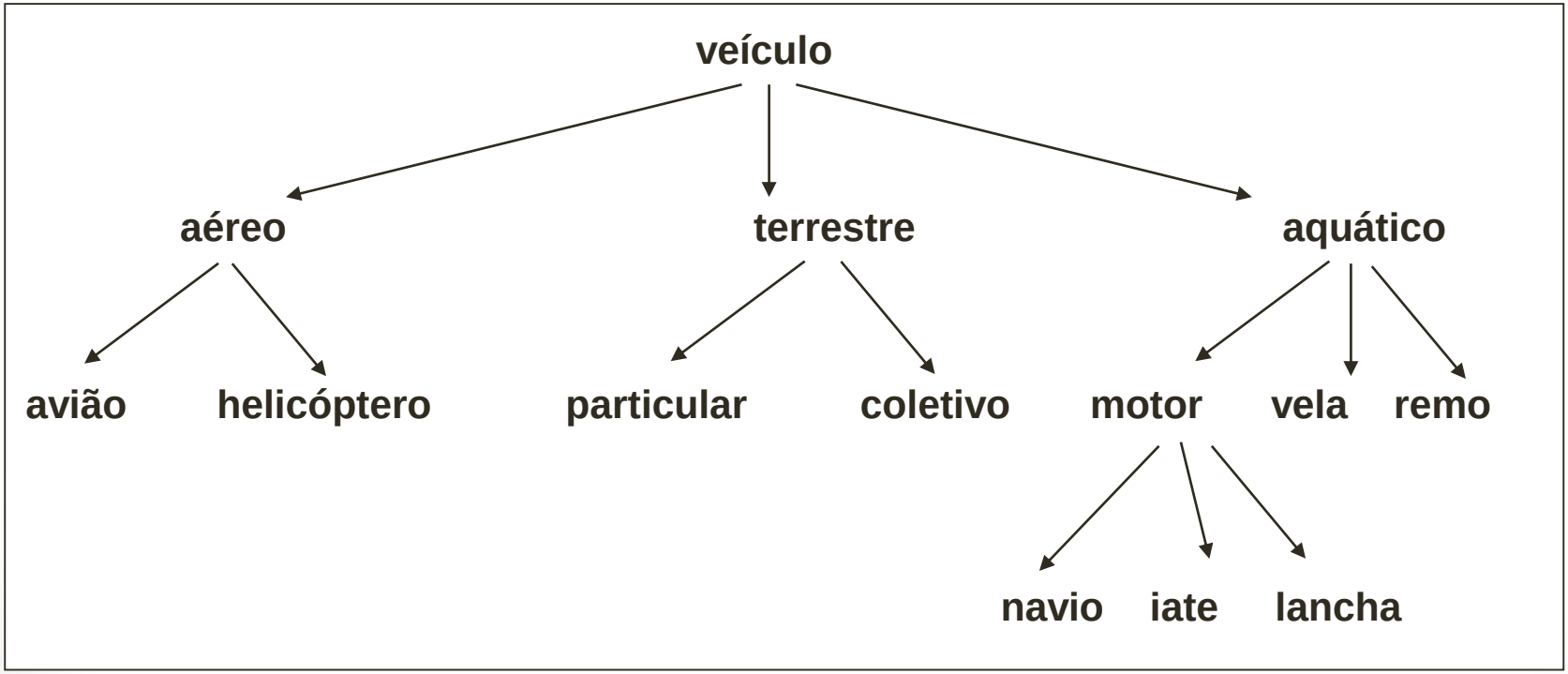
- **hierarquia de especialização – classes e subclasses**
- **hierarquia de composição**
- **hierarquia de dependência**

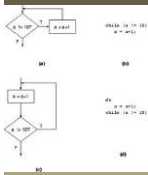


Exemplos de aplicações

Hierarquia de especialização

(classes / subclasses)

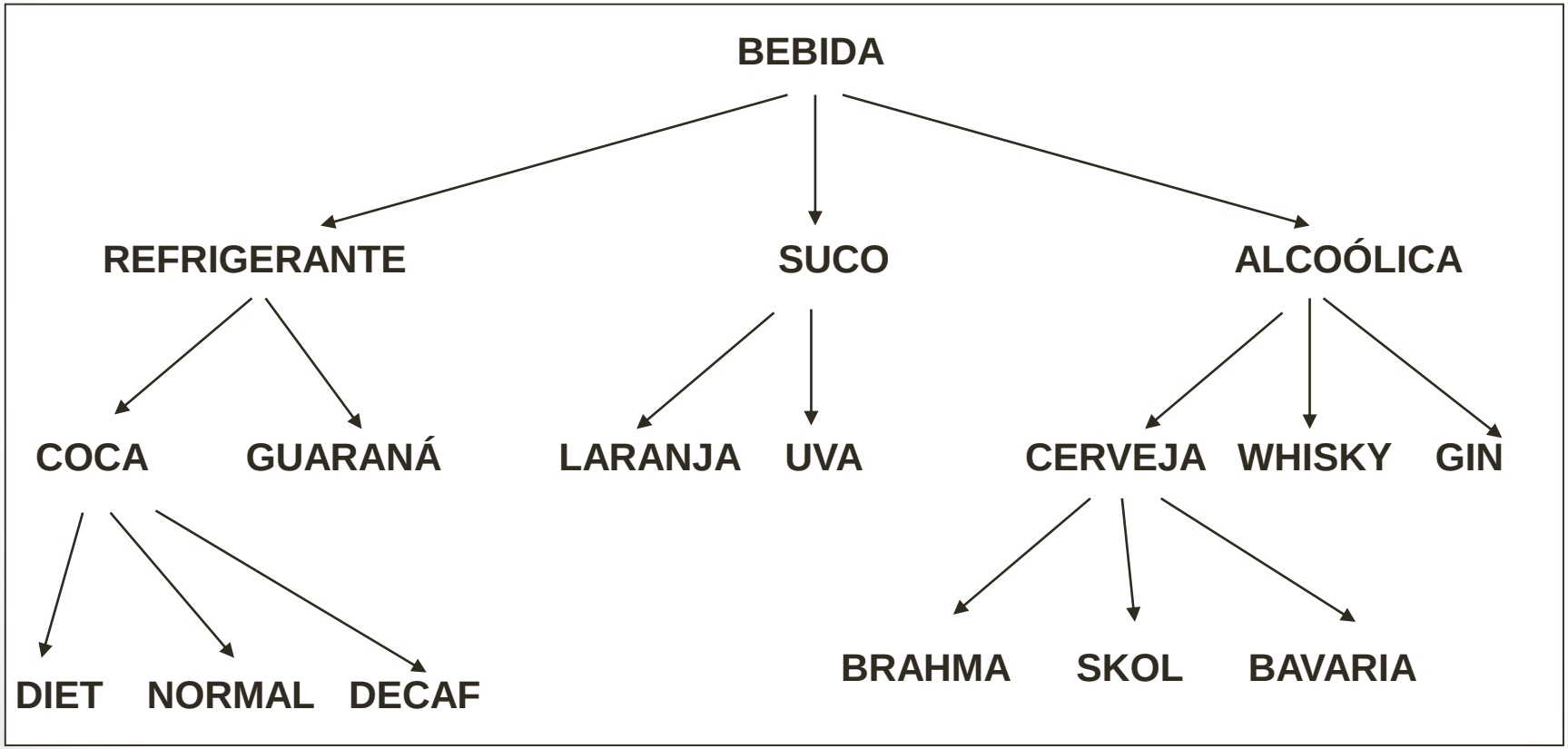




Exemplos de aplicações

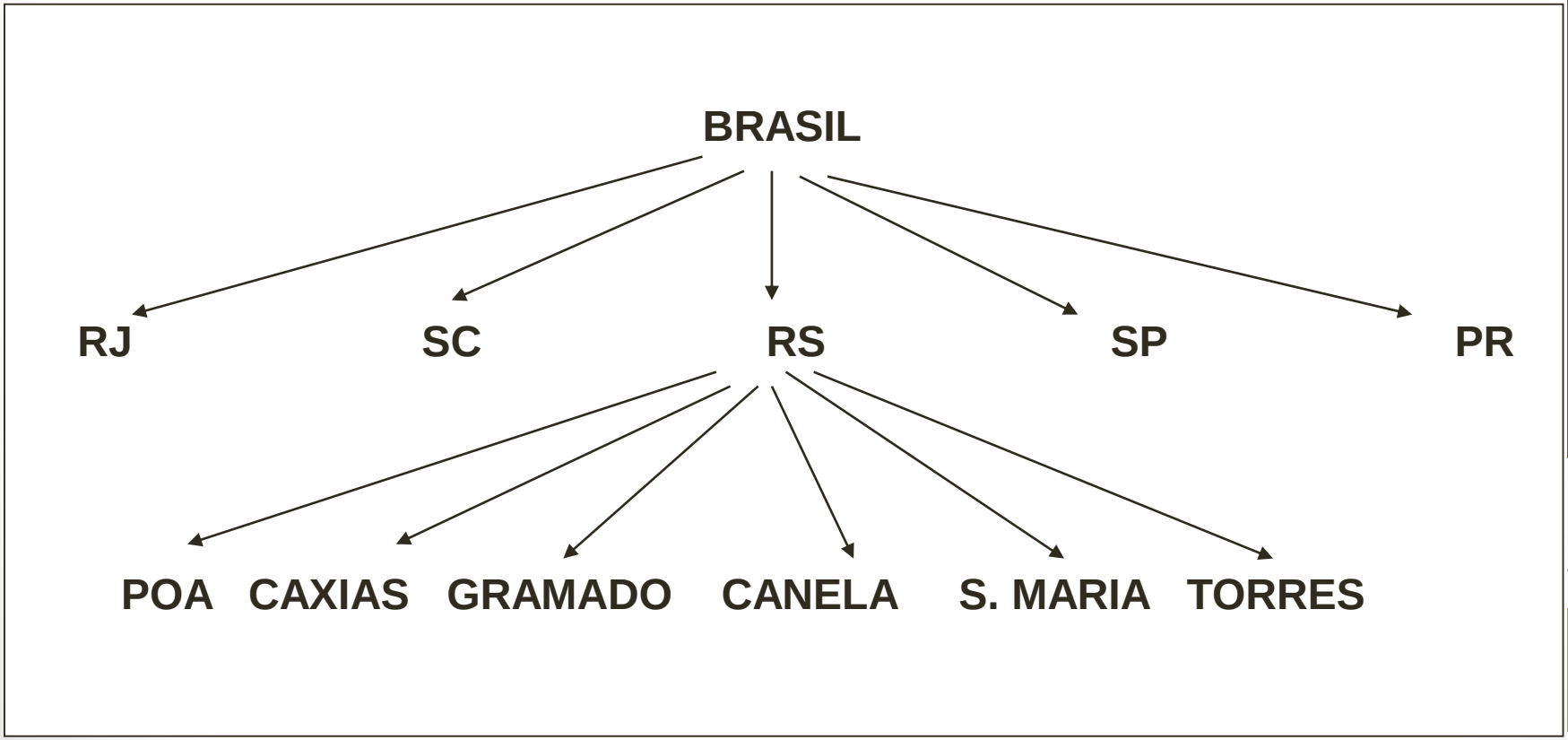
Hierarquia de especialização

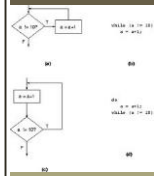
(classes / subclasses)



Exemplos de aplicações

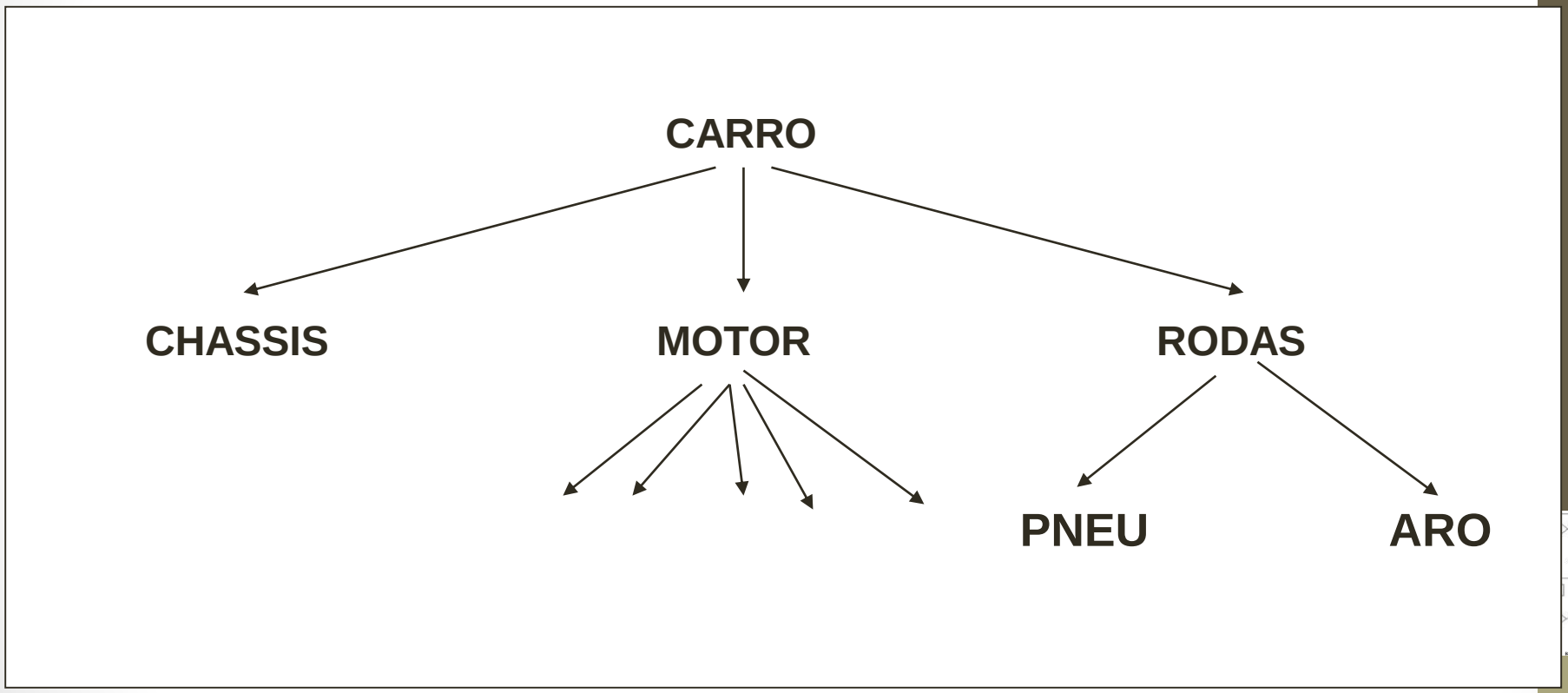
Abstração de Composição

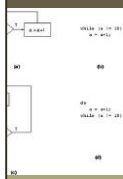




Exemplos de aplicações

Abstração de Composição

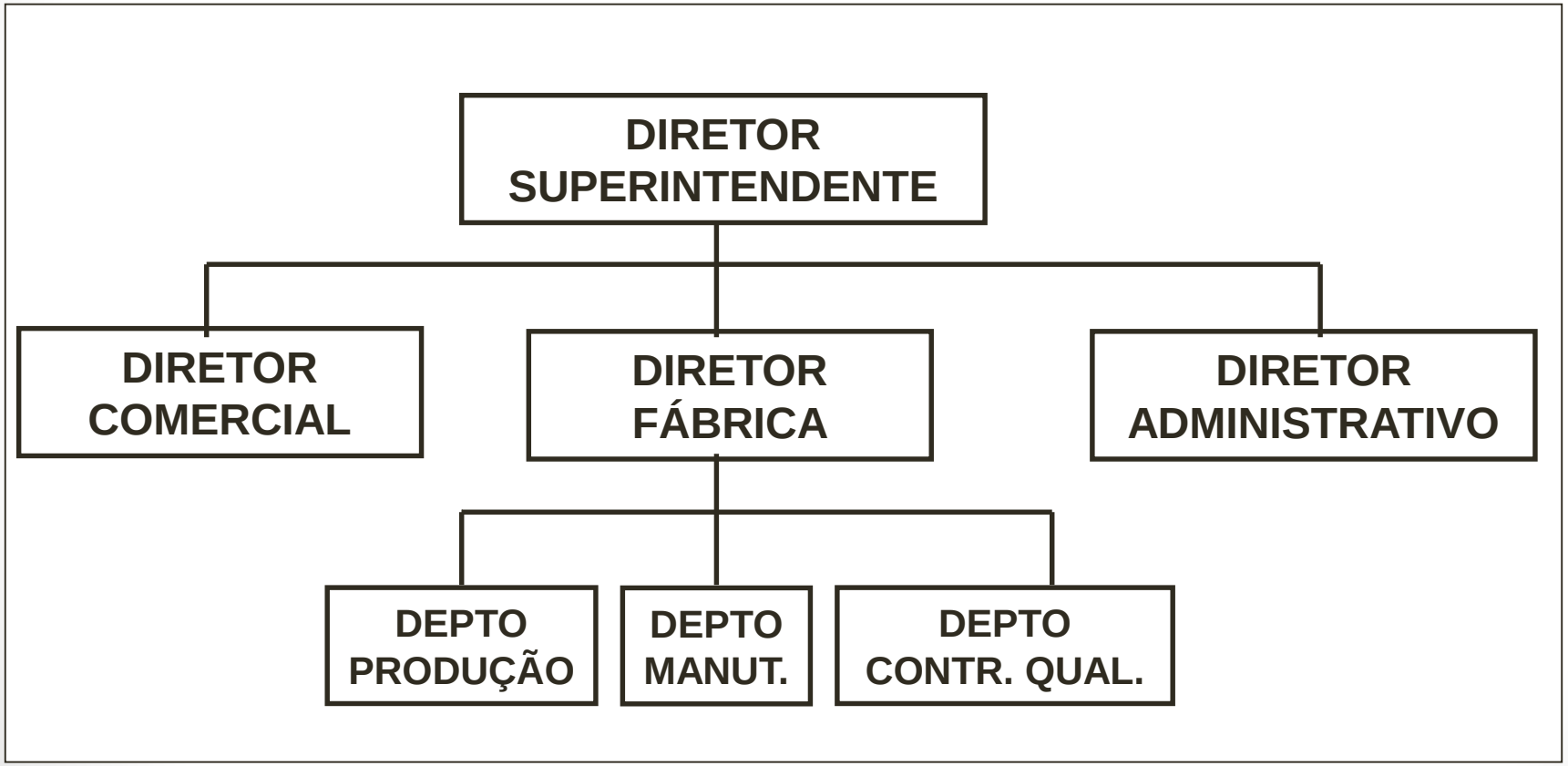




Exemplos de aplicações

Hierarquia de dependência

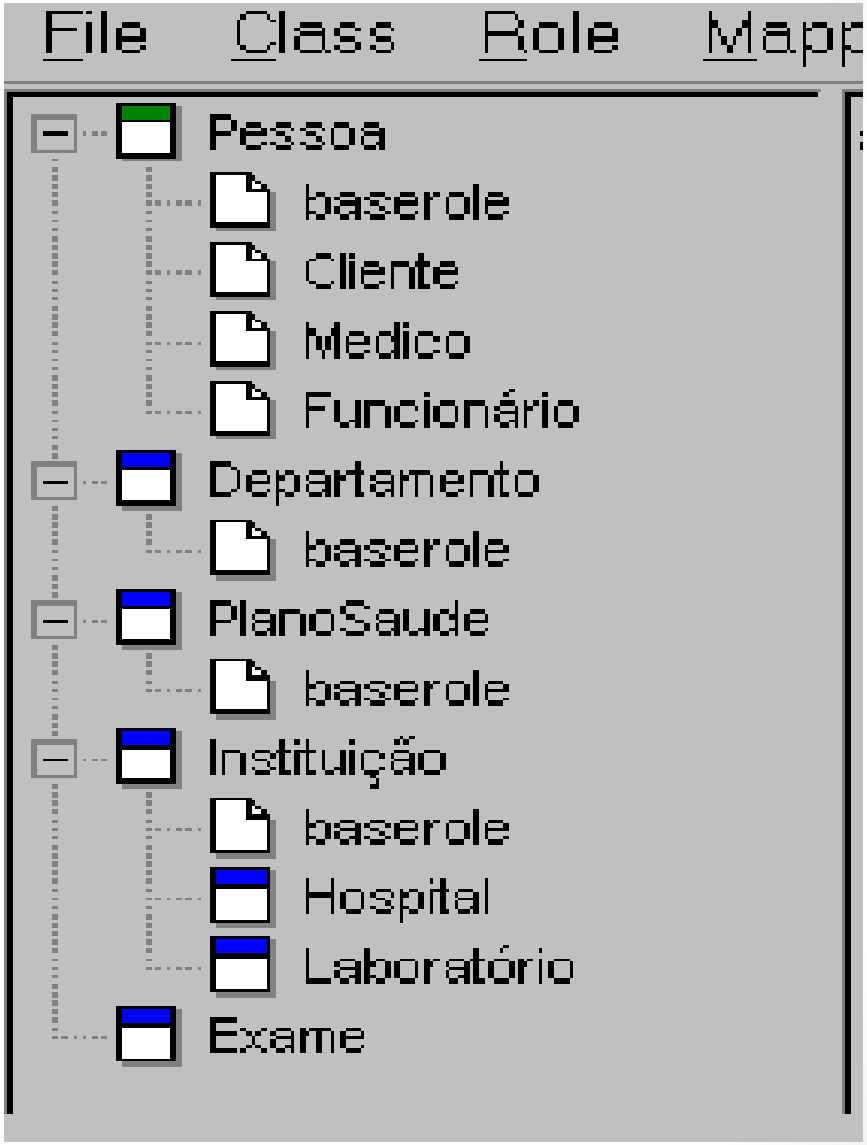
Organograma de empresa



Exemplos de aplicações

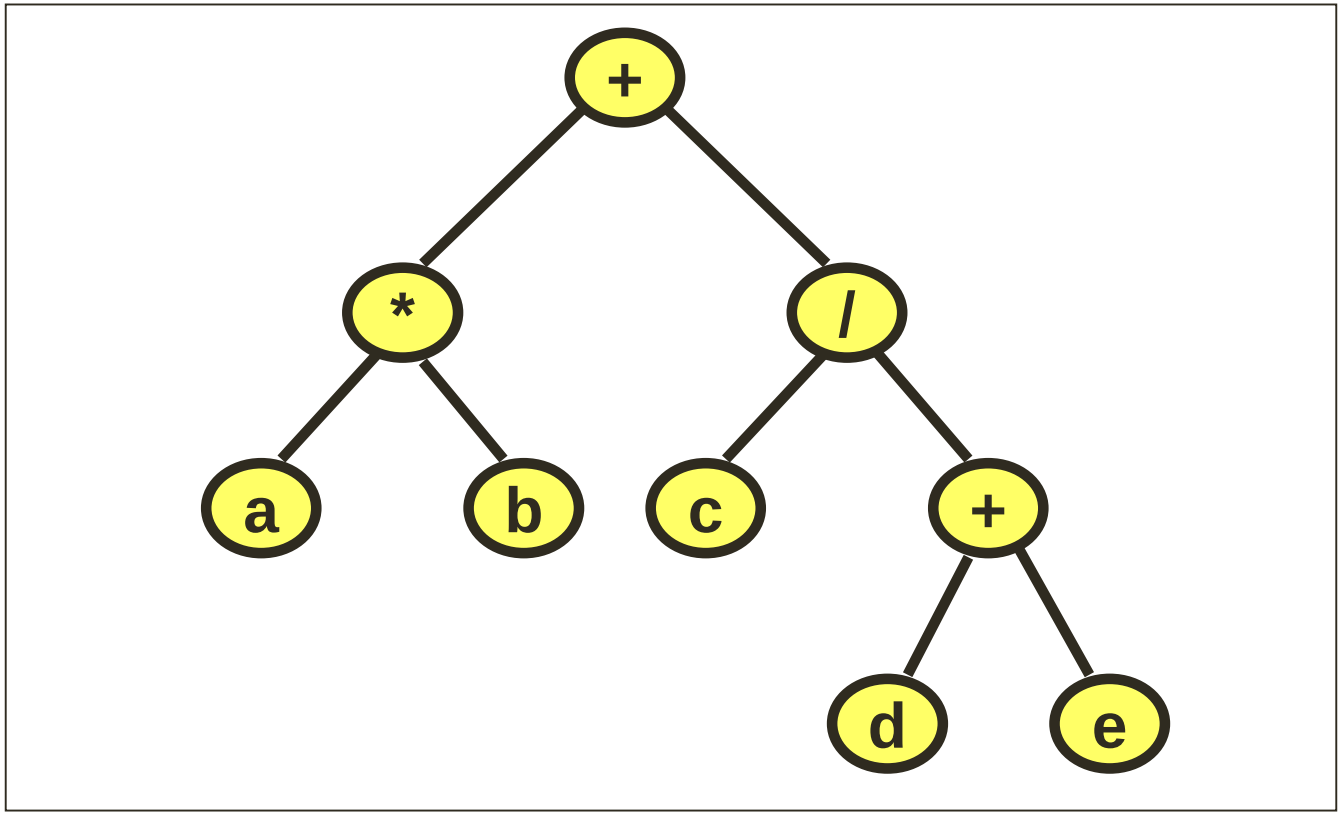
Hierarquia de dependência

Diretórios de arquivos

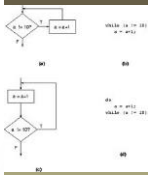


Exemplos de aplicações

Árvore de derivação - compilador

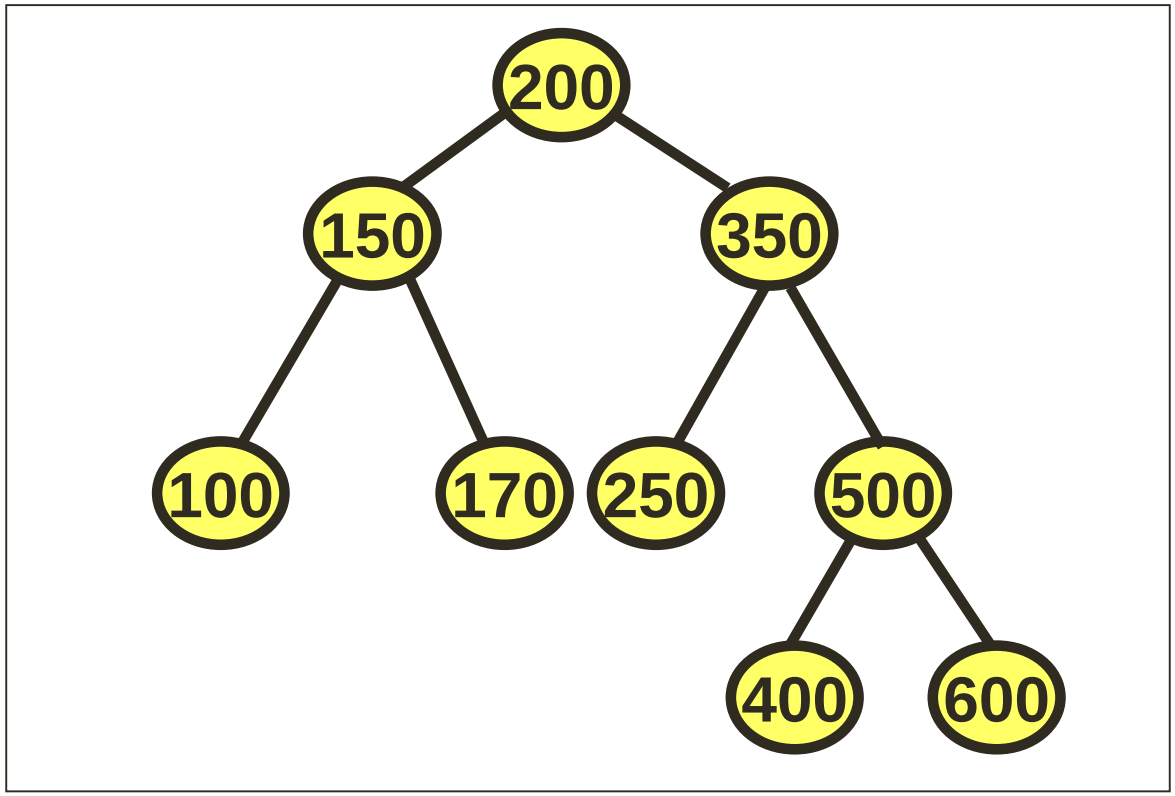


Expressão aritmética: (a * b) + (c / (d + e))



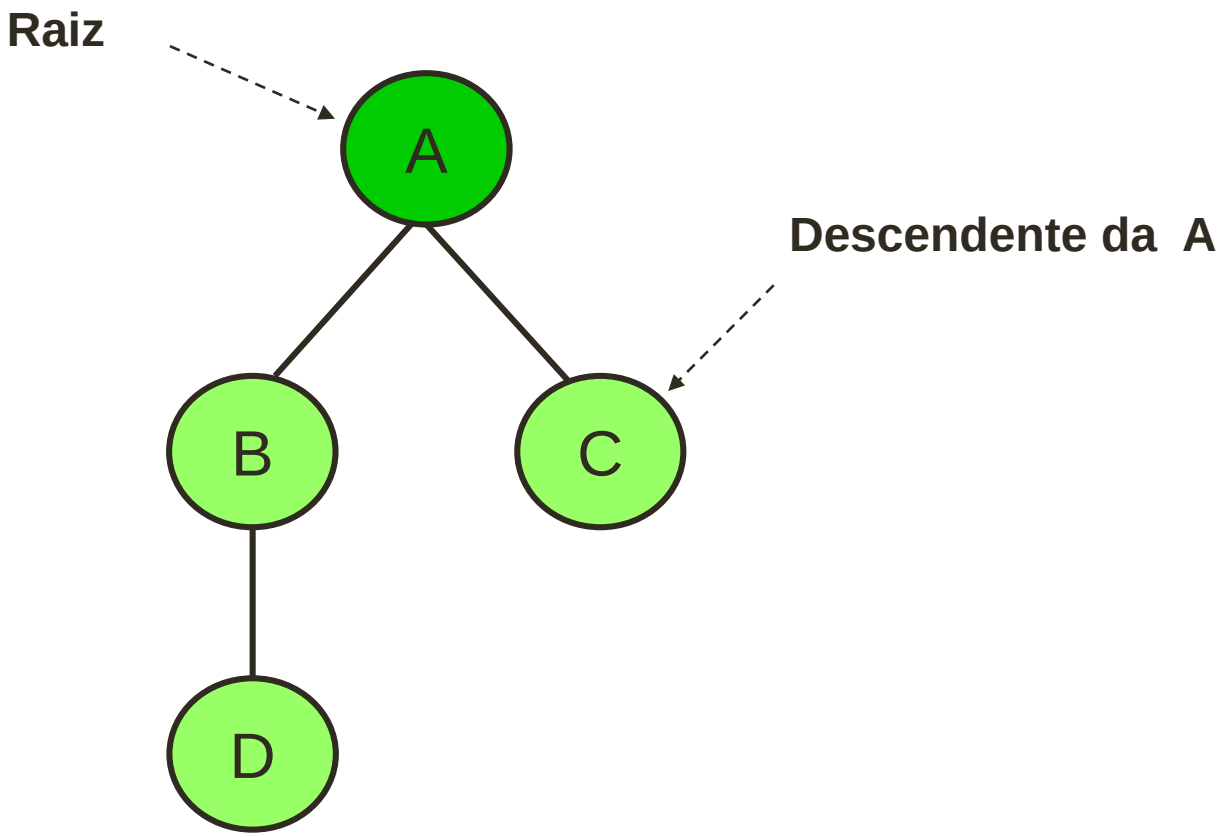
Exemplos de aplicações

Ordenar valores



Árvore ordenada - esquerda / raiz / direita

Terminologia



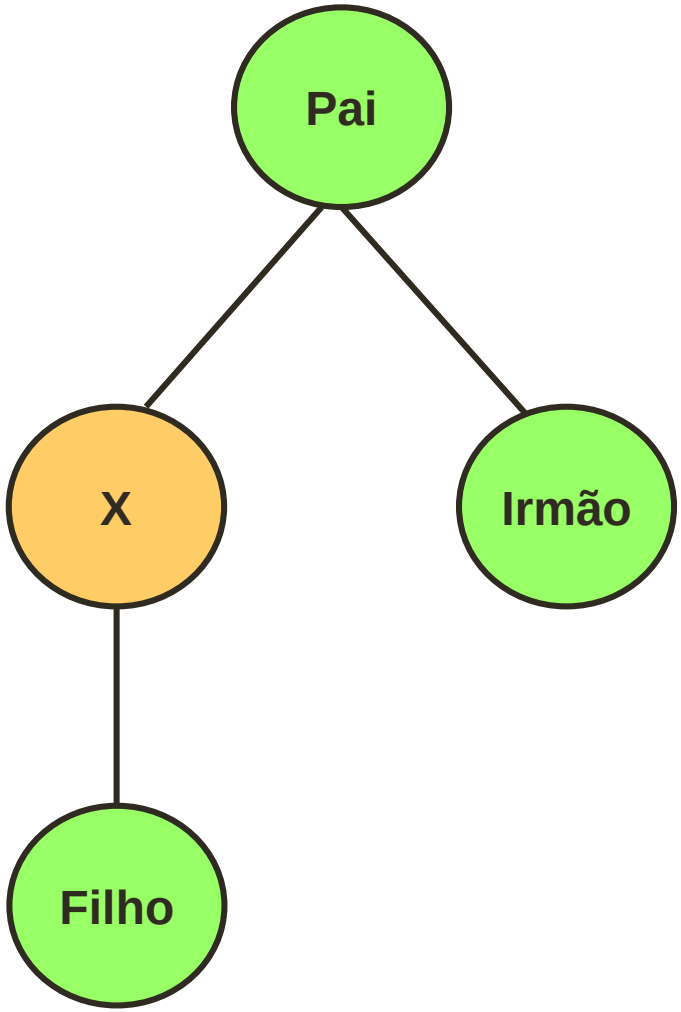
Nodos descendentes

Pai

= mãe

= ascendente

= antecessor



Irmão

= irmã

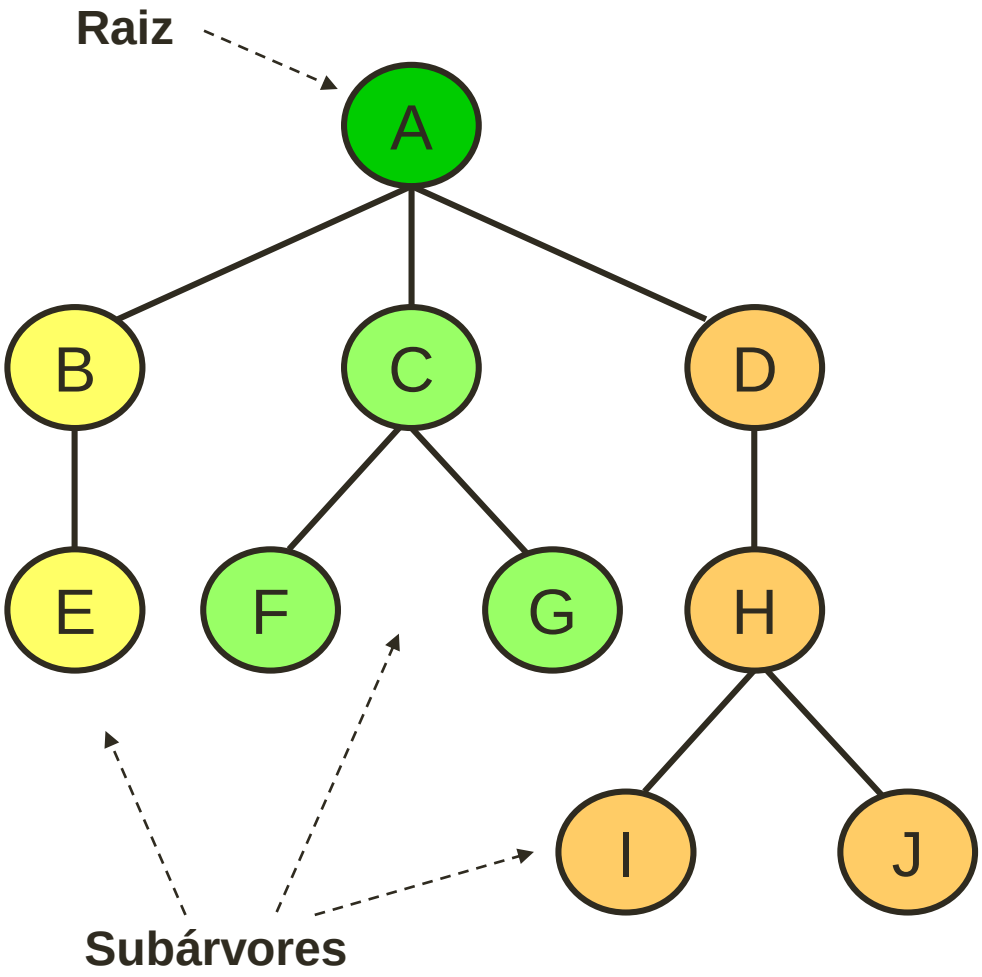
Filho

= filha

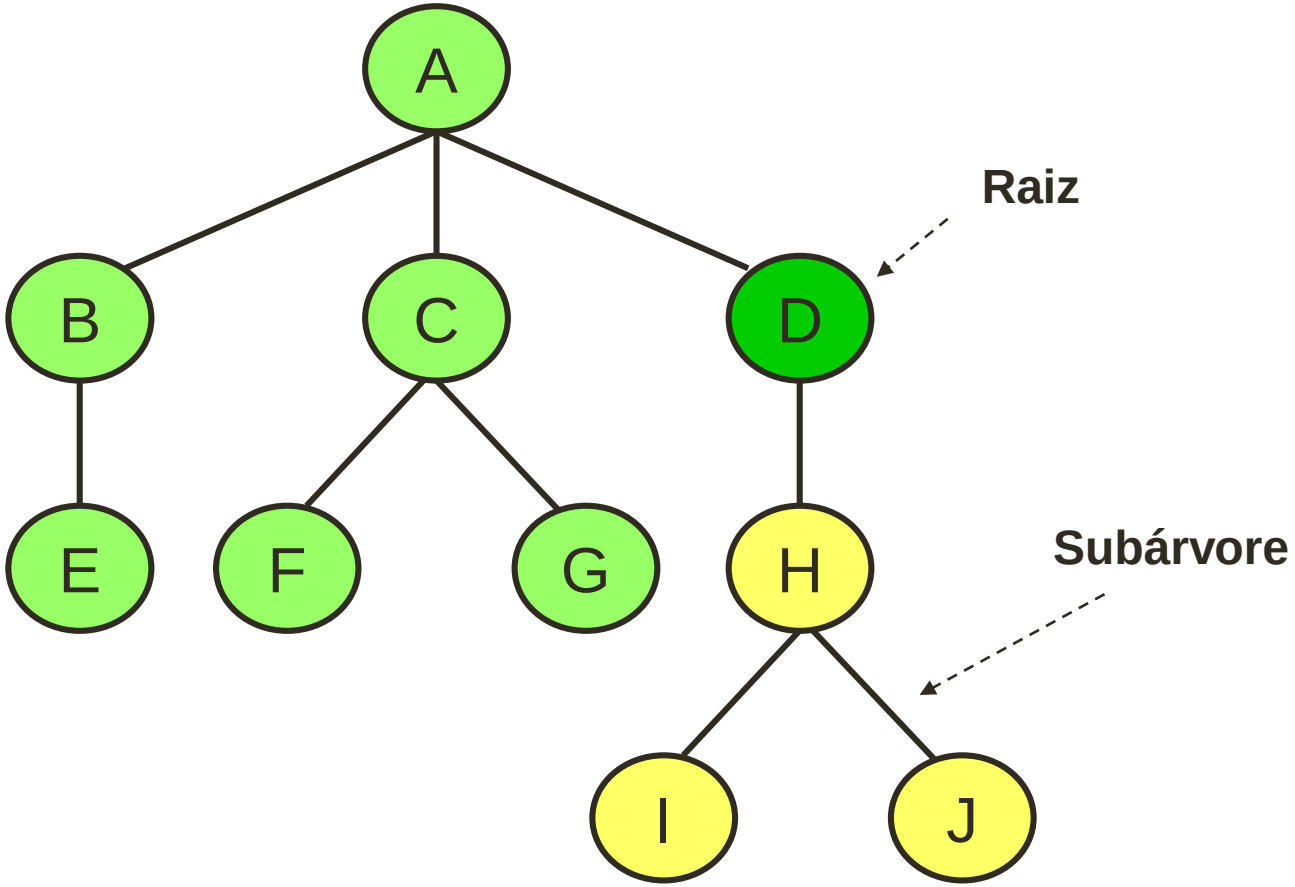
= descendente

= sucessor

Subárvore

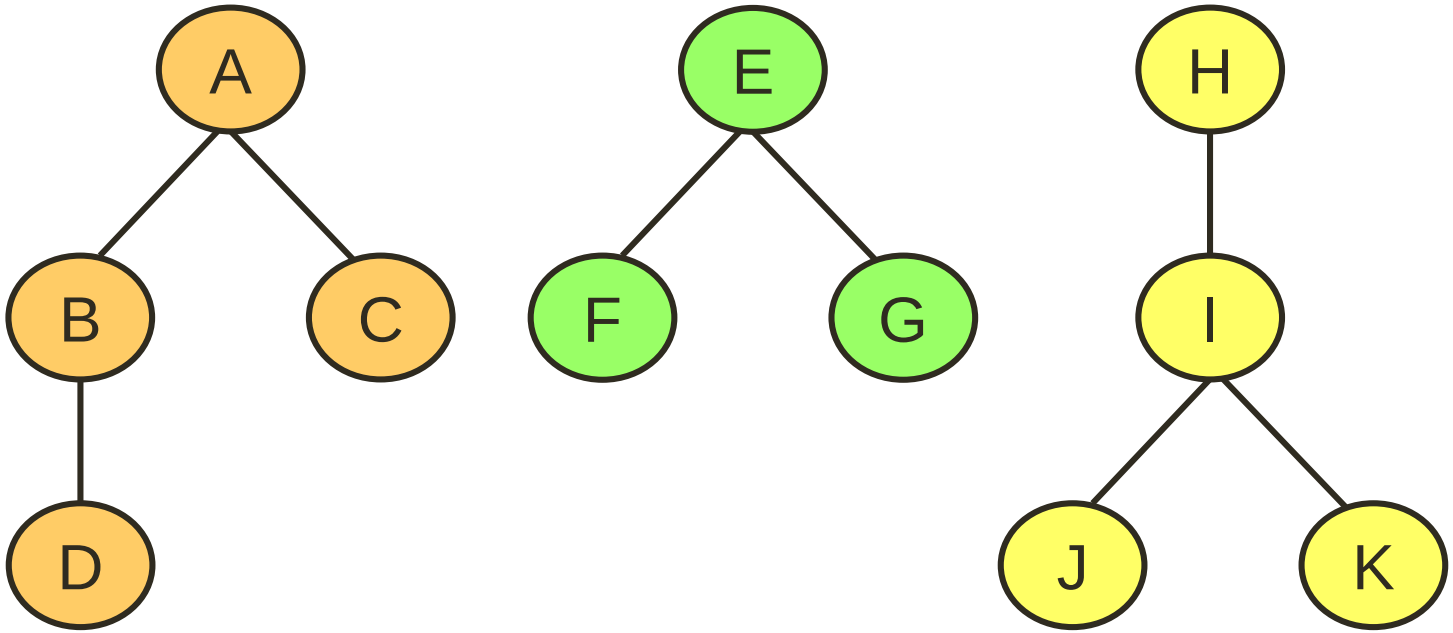


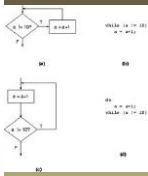
Subárvore



Floresta

Conjunto de zero ou mais árvores disjuntas

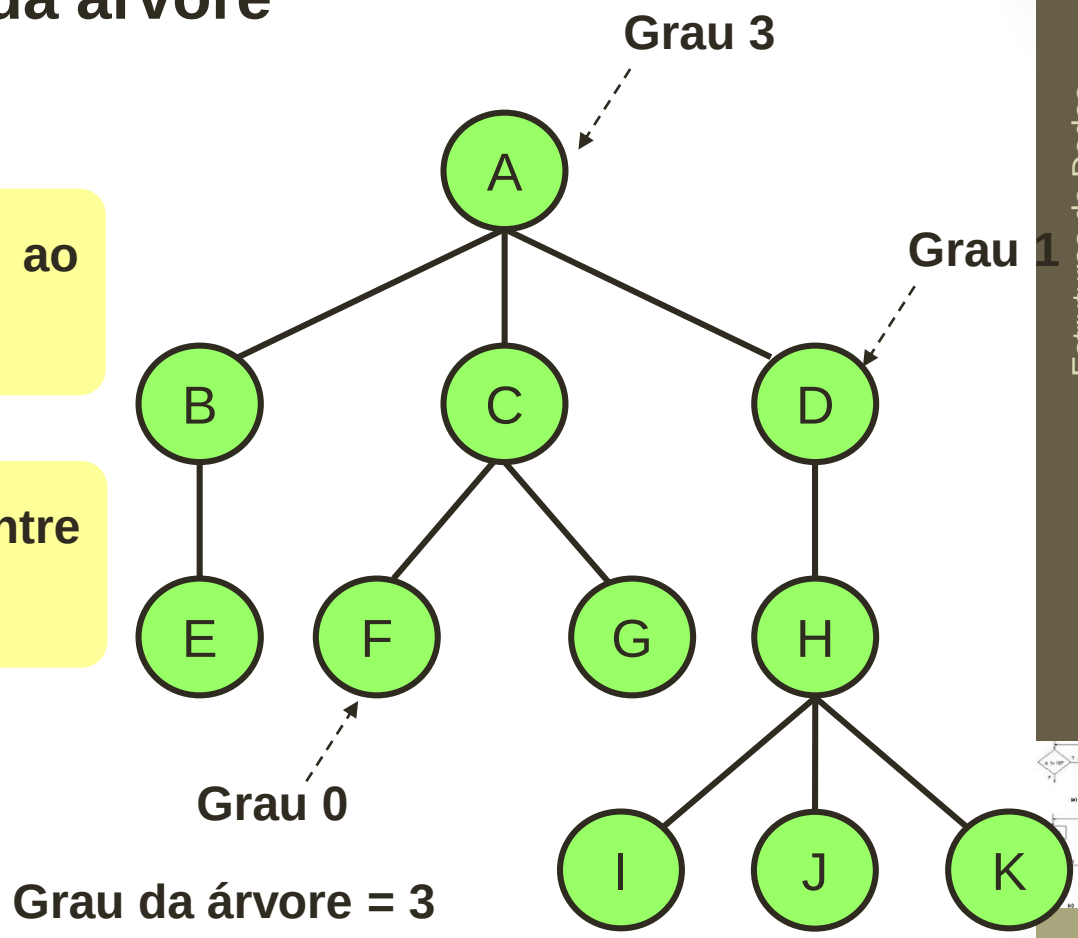




Grau dos nodos e da árvore

Grau de um nodo é igual ao número de suas subárvores

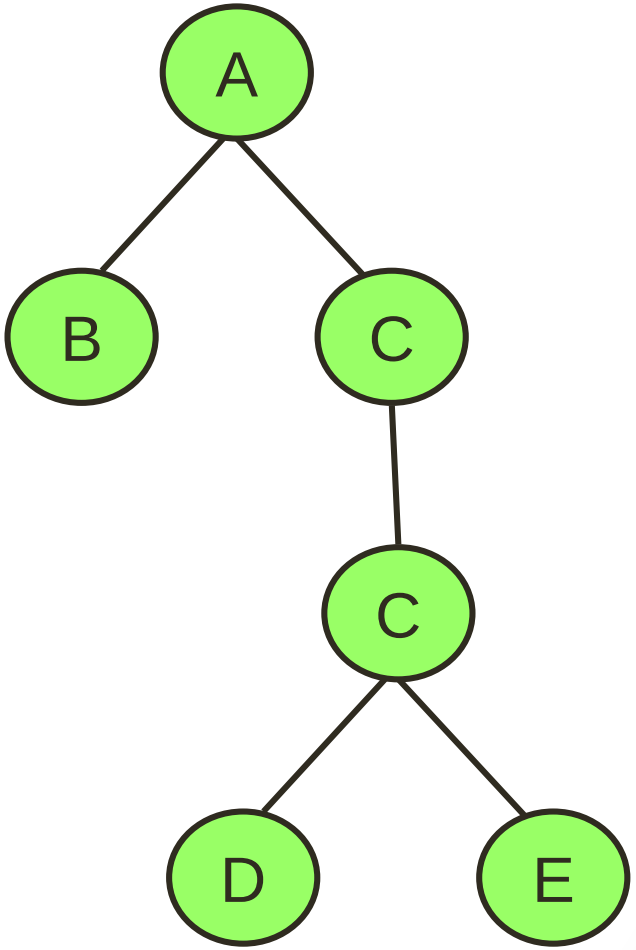
Grau de uma árvore maior entre os graus de seus nodos

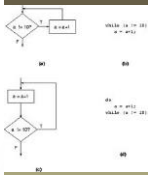


Árvore binária e n-ária

Árvore binária apresenta no máximo grau 2 em cada nodo

Árvore n-ária apresenta no máximo grau "n" em cada nodo

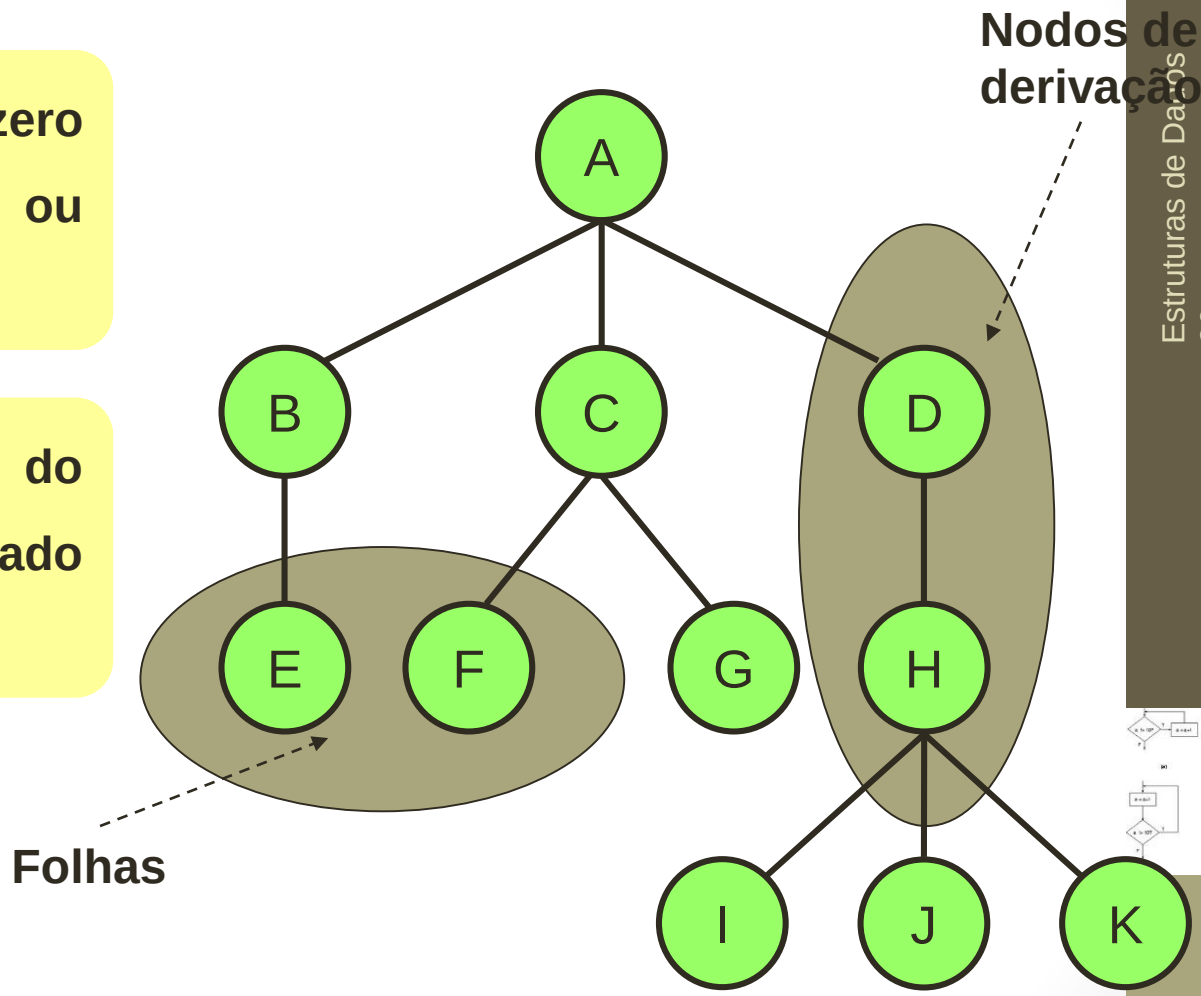




Nodos de derivação e folhas

Nodo com grau igual a zero é denominado **folha** ou **terminal**

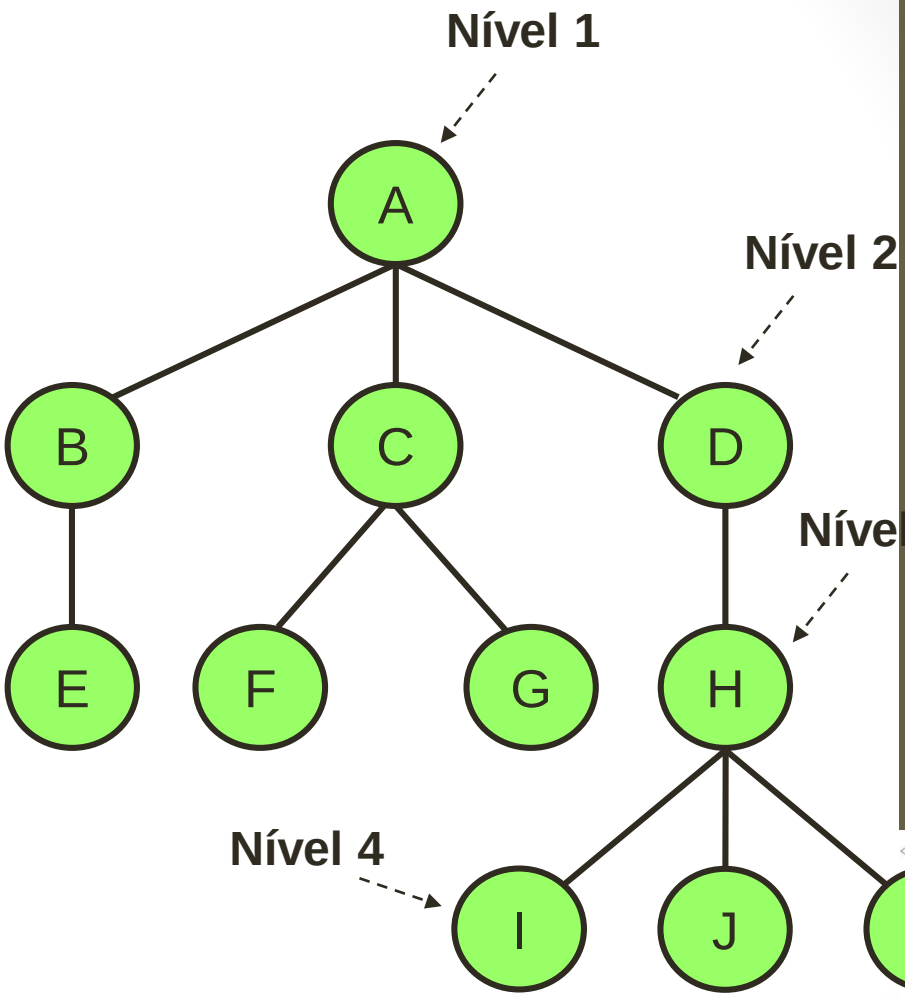
Nodo com grau maior do que zero é denominado **nodo de derivação**



Nível e altura

Nível de um nodo é igual ao número de linhas entre ele e a raiz, acrescido de uma unidade

Altura ou **profundidade** de uma árvore é igual ao seu maior nível

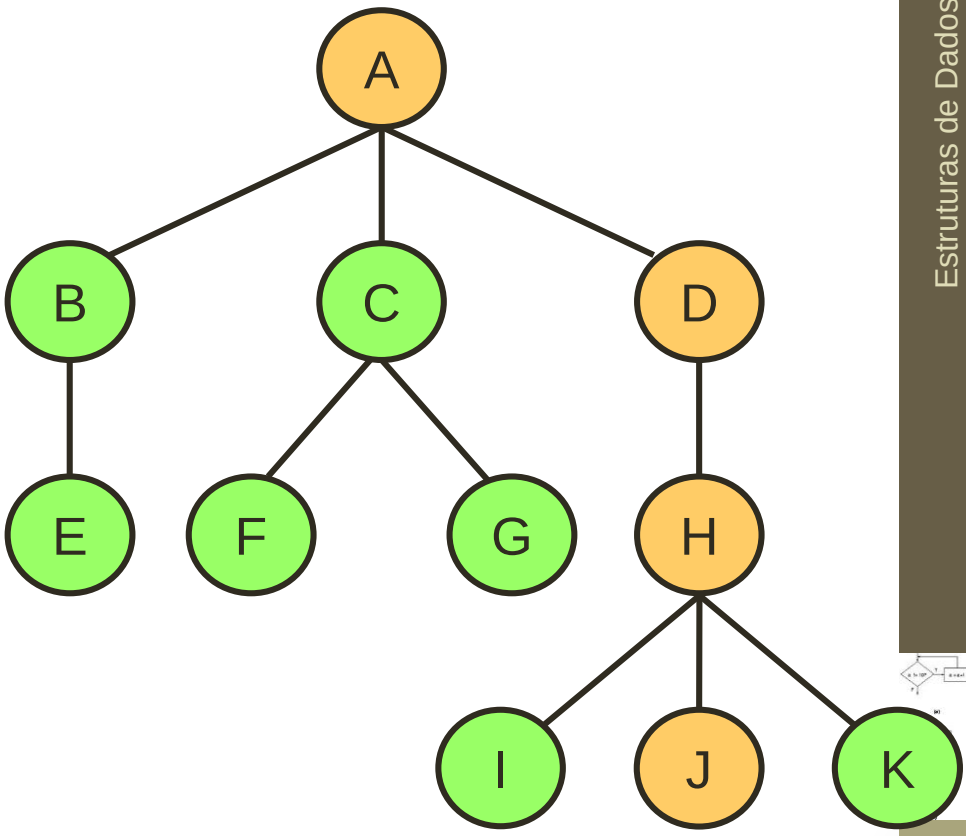


Altura da árvore = 4

Caminho

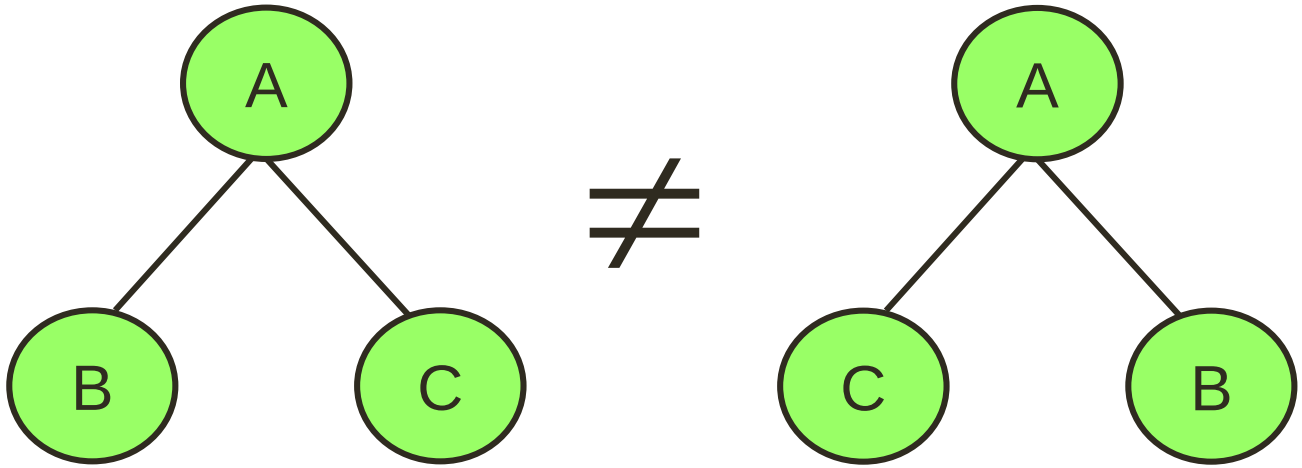
Caminho sequência de nodos consecutivos distintos entre 2 nodos

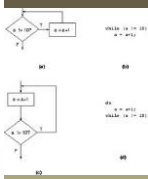
Comprimento do caminho entre 2 nodos é dado pelo número de níveis entre estes 2 nodos, diminuído de 1 unidade



Árvore ordenada

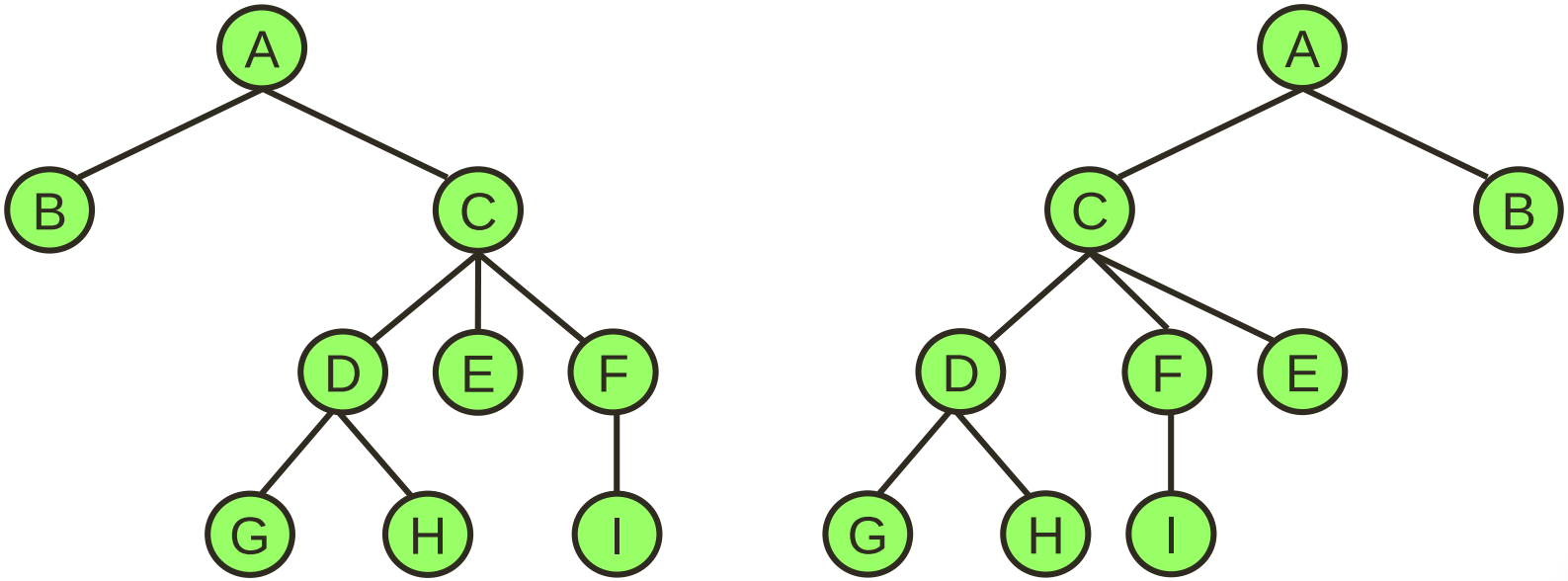
Quando a ordem de suas subárvores é relevante





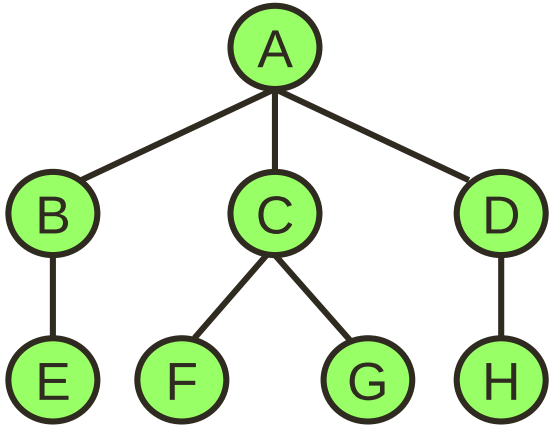
Árvores isomorfas

Duas árvores são isomorfas quando puderem se tornar coincidentes através de uma permutação na ordem das sub-árvores de seus nodos.

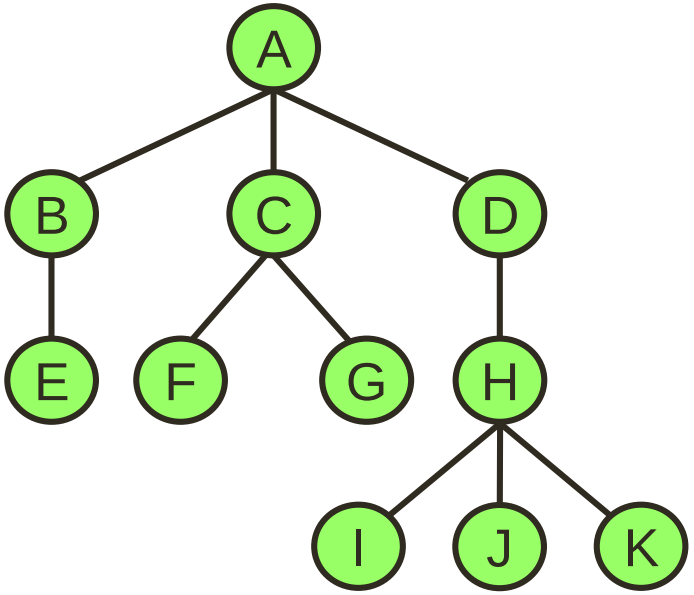


Árvore balanceada

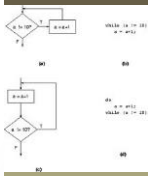
Uma árvore é balanceada quando todas as folhas ficam à mesma distância da raiz



Balanceada



Não balanceada



Definição formal de árvore

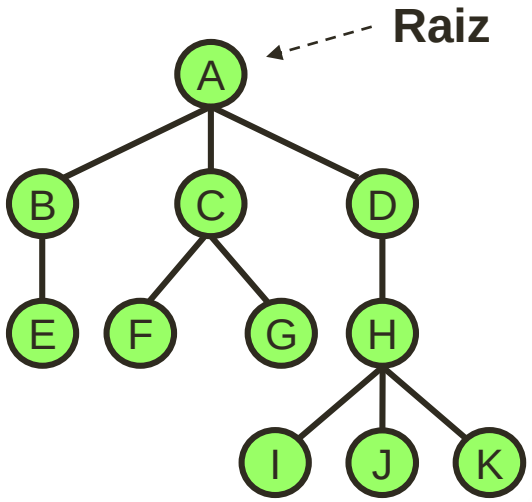
Uma árvore é um conjunto finito *T* de zero ou mais nodos, tal que:

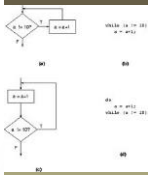
(1) número de nodos diferente de zero:

- **existe um nodo denominado raiz da árvore**
- **os demais nodos formam m > 0 conjuntos disjuntos**

S_1, S_2, ..., S_m, onde cada um destes é uma árvore (S_i são denominadas subárvores)

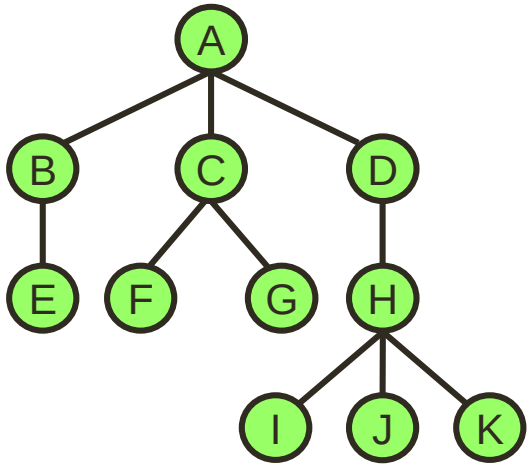
(2) número de nodos zero: árvore vazia

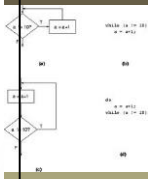




Operações básicas

- criação de uma árvore
- inserção de um novo nodo
  - a raiz
  - como folha
  - em posição intermediária
- exclusão de um determinado nodo
- acesso a um nodo
  - determinar forma de percorrer a árvore
- destruição da árvore





Outras operações

- **Raiz**

  retorna a raiz da árvore

- **Pai**

  retorna o nodo pai de um determinado nodo

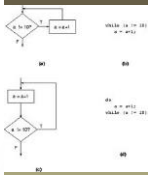
- **Filho**

  retorna o primeiro filho de um determinado nodo

- **Tamanho**

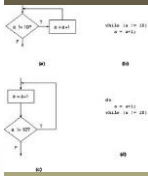
  retorna o número total de nodos de uma árvore

  ou retorna a altura de uma árvore



Árvores

Árvores implementadas através de contiguidade física



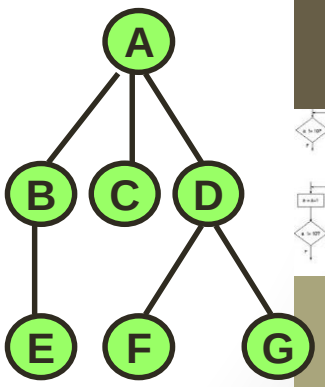
Árvores em contiguidade física

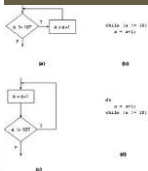
- **Representação não é intuitiva**
- **Estratégia de representação que preserve hierarquia**
- **Informações adicionais (ex: número de descendentes do nodo)**
- **Operações**

  Inserção e remoção – deslocamento de nodos

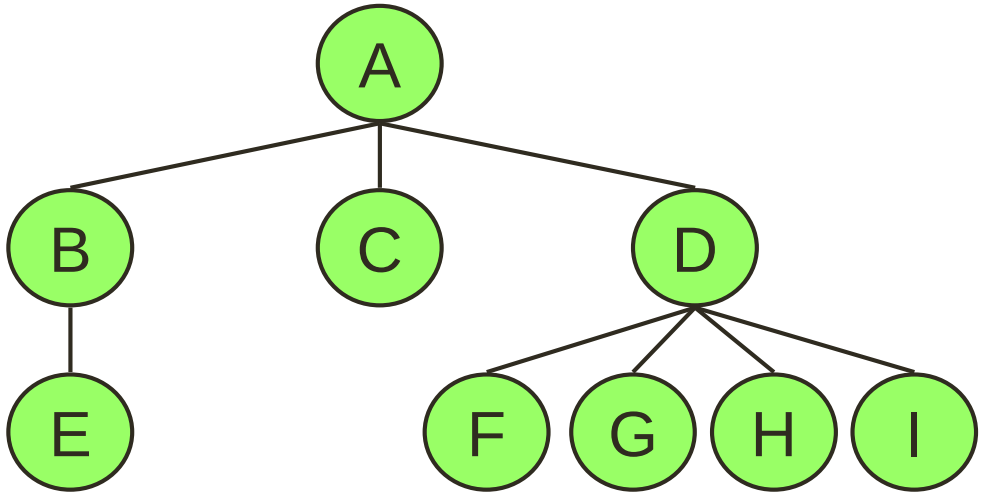
  Acesso – respeitar hierarquia

1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	3	B	1	E	0	C	0	D	2	F	0	G	0

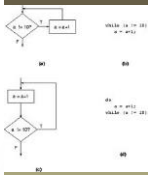




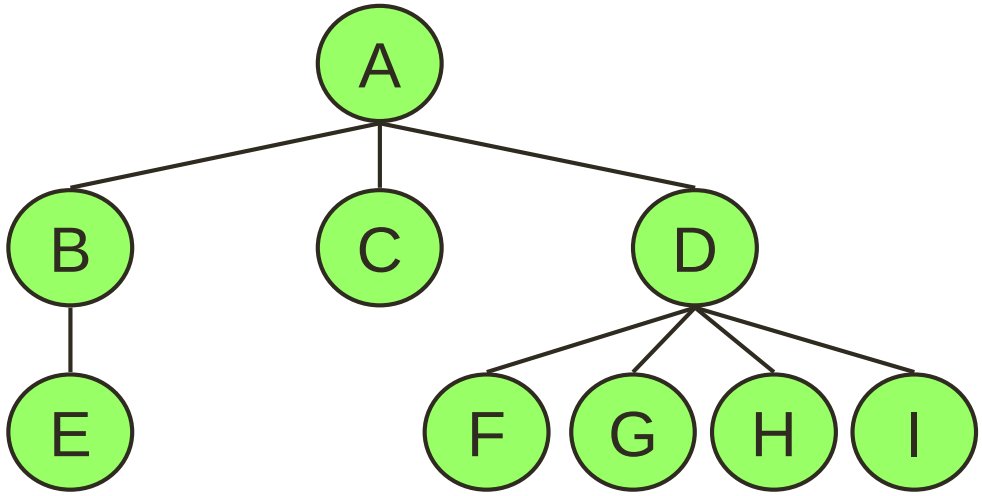
Implementação por níveis



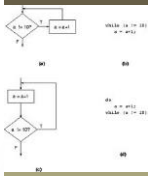
1	2	3	4	5	6	7	8	9	10
A(3)	B(1)	C(0)	D(4)	E(0)	F(0)	G(0)	H(0)	I(0)	



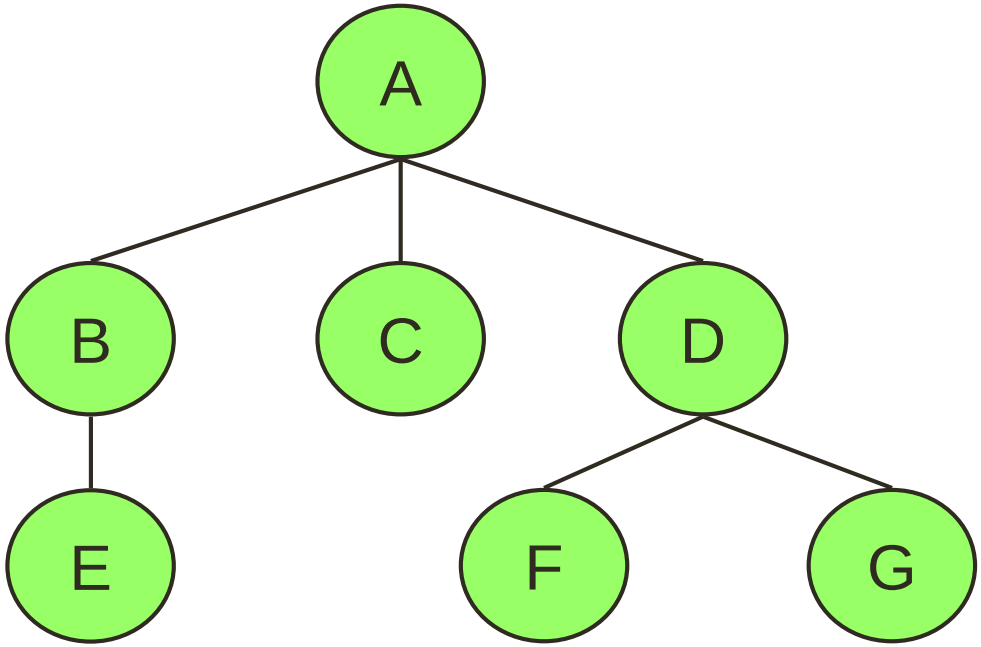
Implementação em profundidade



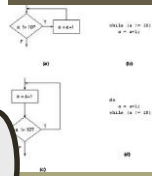
1	2	3	4	5	6	7	8	9	10
A(3)	B(1)	E(0)	C(0)	D(4)	F(0)	G(0)	H(0)	I(0)	



Árvore ternária

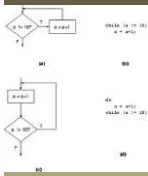


1	2	3	4	5	6	7	8	9	10	11	12	13
A	B	C	D	λ	E	λ	λ	λ	λ	F	λ	G



Análise da implementação de árvores por contiguidade

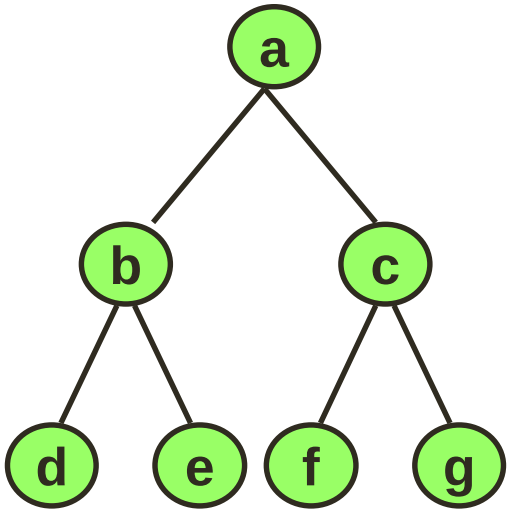
- **Não constitui, em geral, uma boa solução**
- **Dificuldades para manipulação da estrutura (hierarquia)**
- **Geralmente eficiente em termos de espaço ocupado**

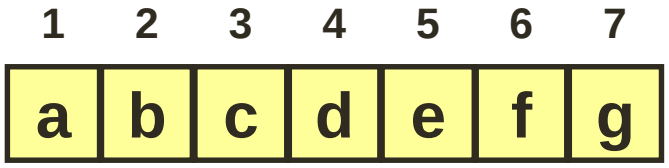


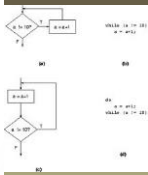
Análise da implementação de árvores por contiguidade

Exceção 1

Quando os nodos são processados exatamente na mesma ordem em que são armazenados



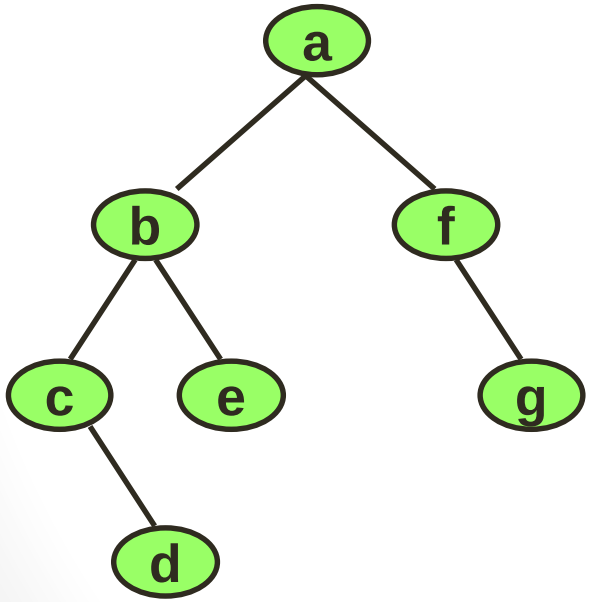


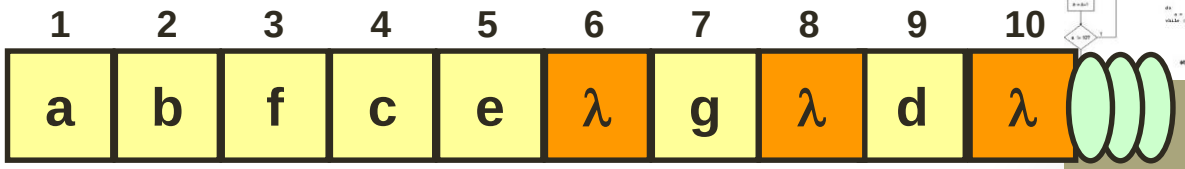


Análise da implementação de árvores por contiguidade

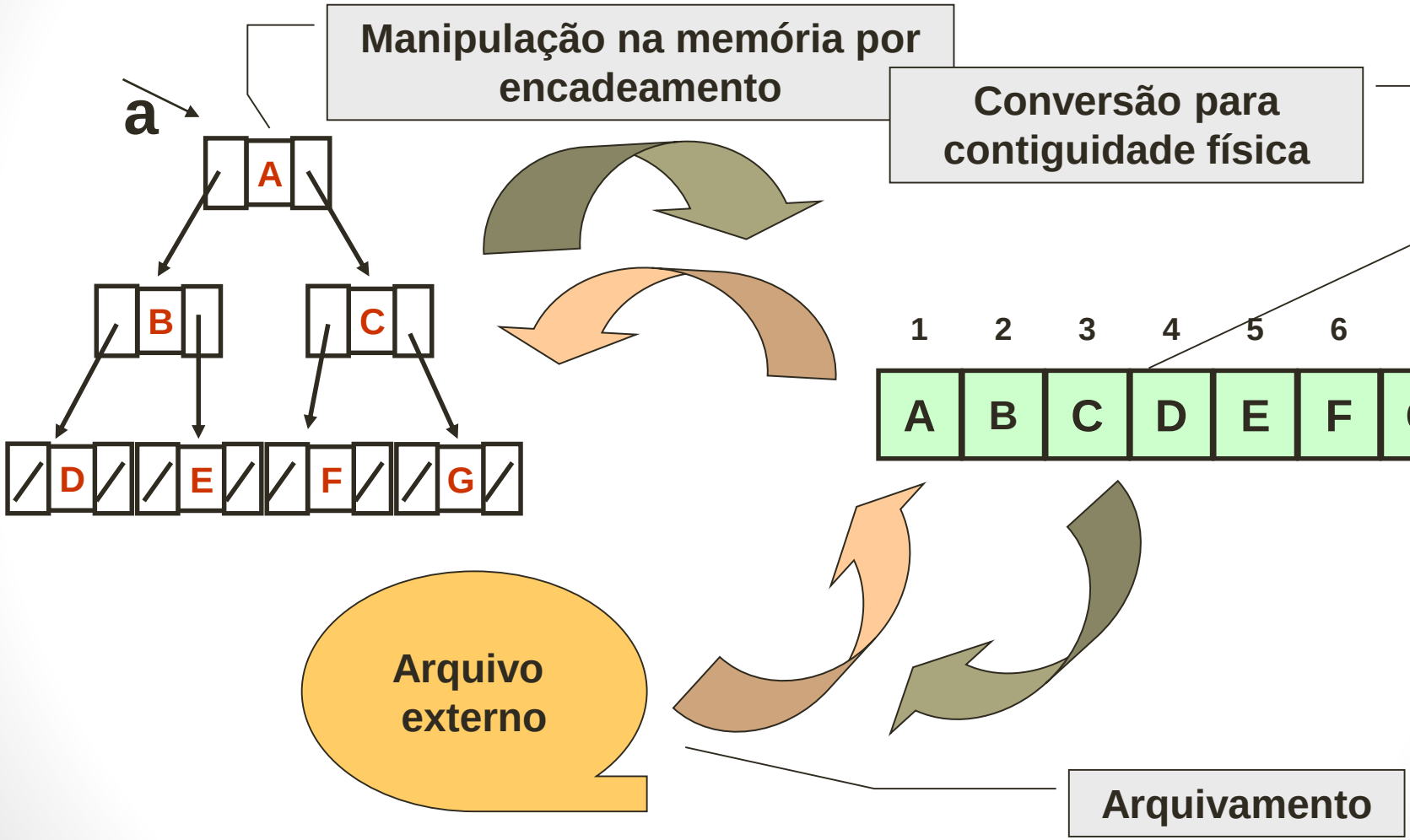
Exceção 2

Quando, excetuando-se as folhas, os demais nodos têm graus iguais ou muito próximos



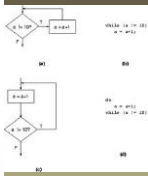


Arquivamento permanente de árvores

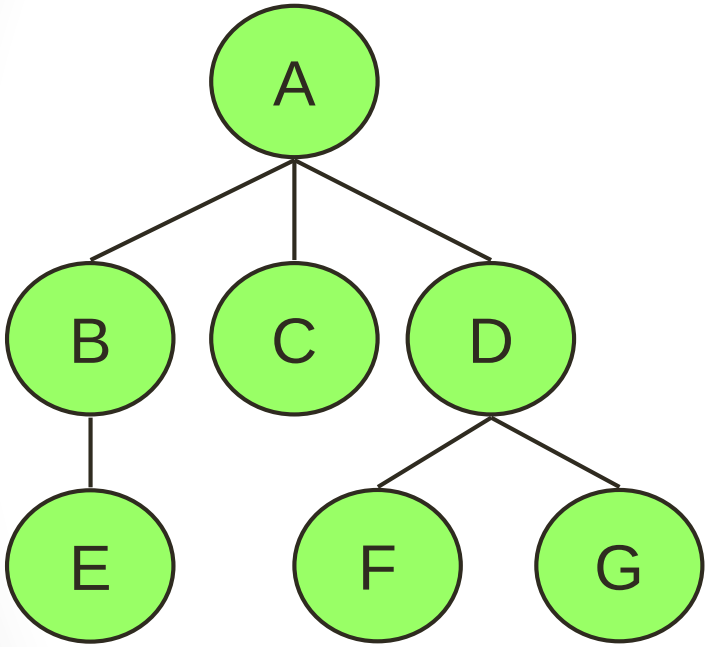


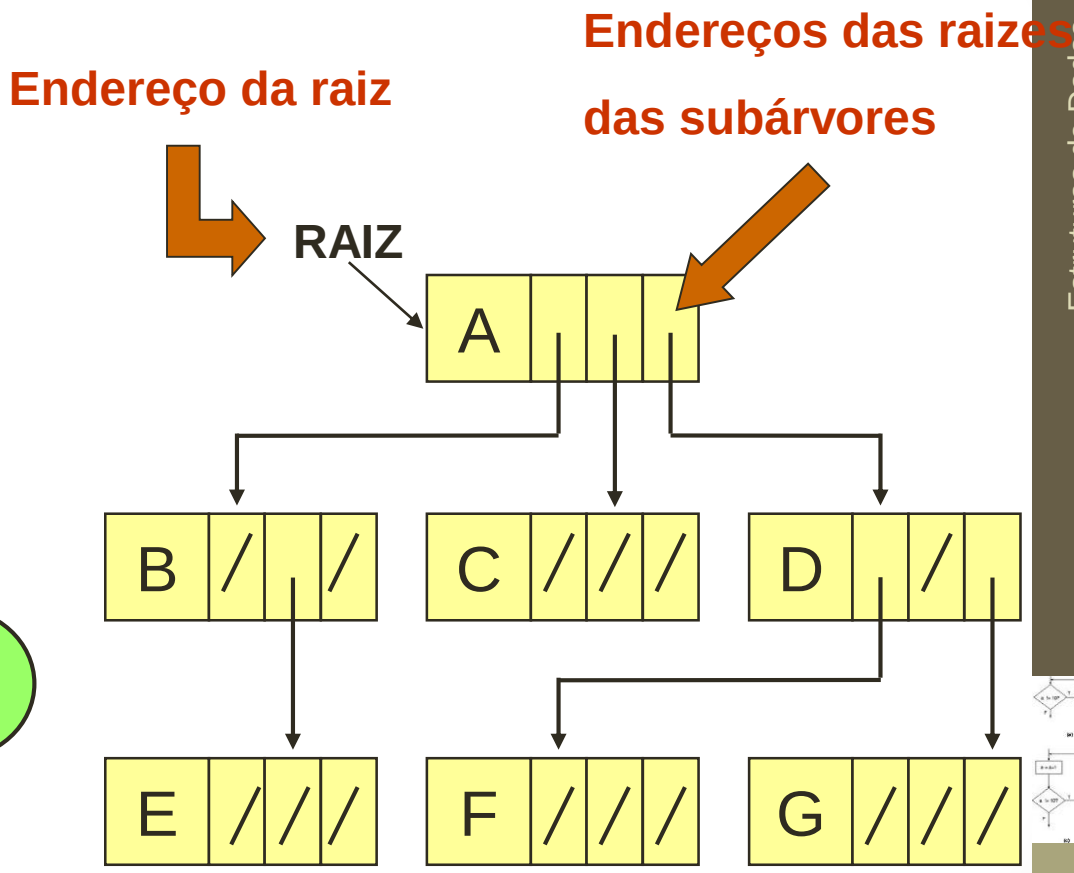
Árvores

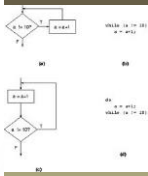
Árvores implementadas por encadeamento



Árvore implementada por encadeamento

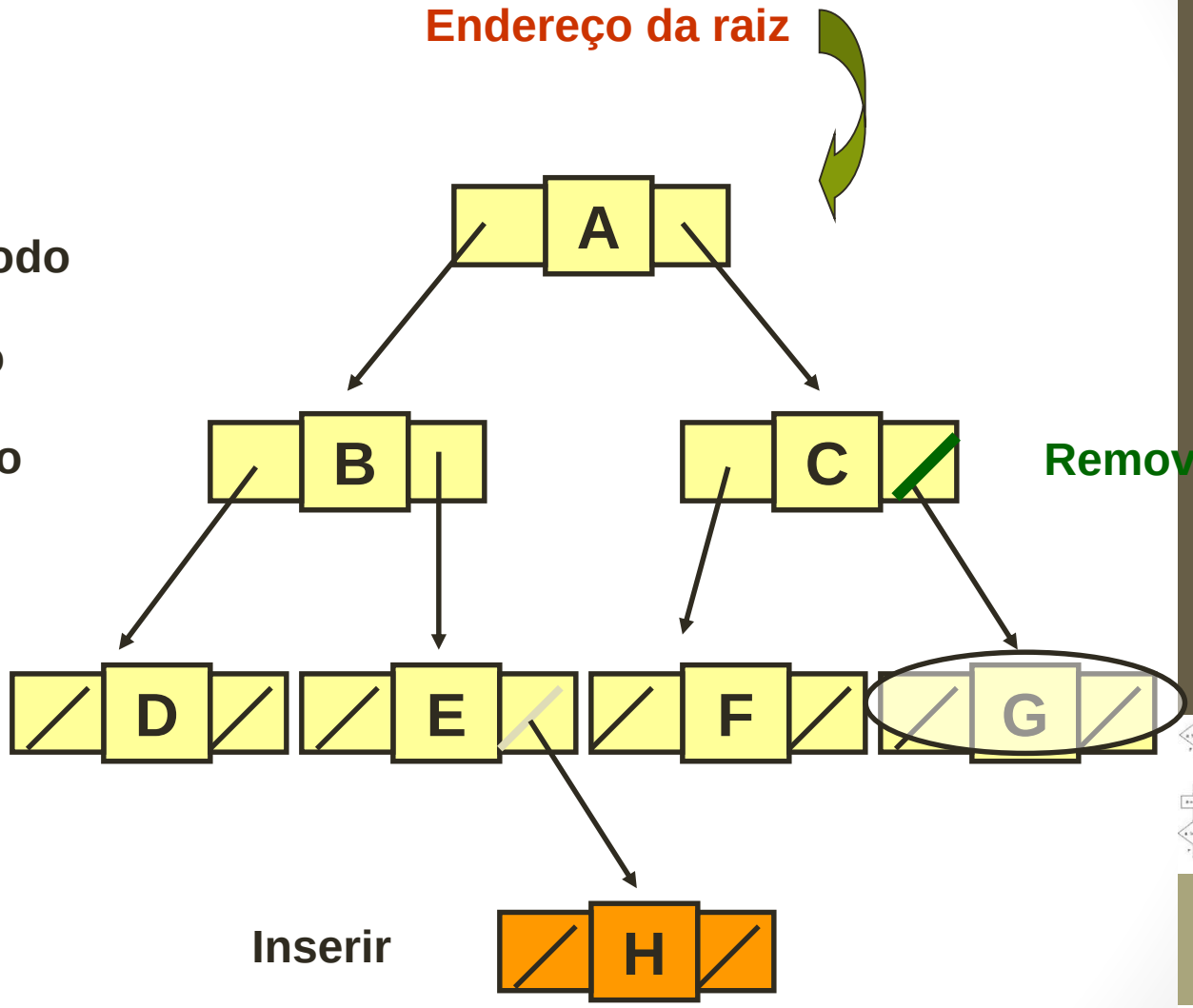


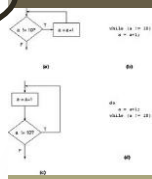




Operações

- Inserir novo nodo
- Remover nodo
- Consultar nodo



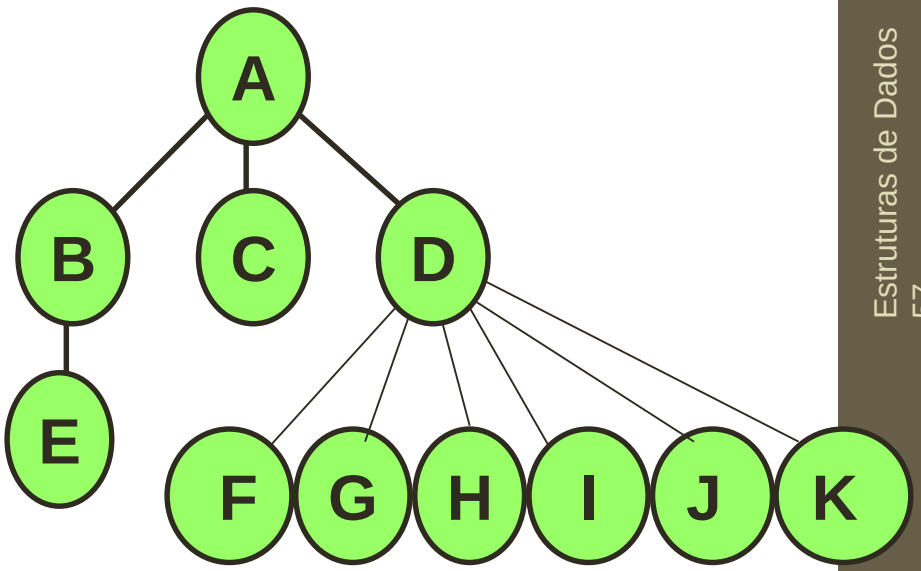


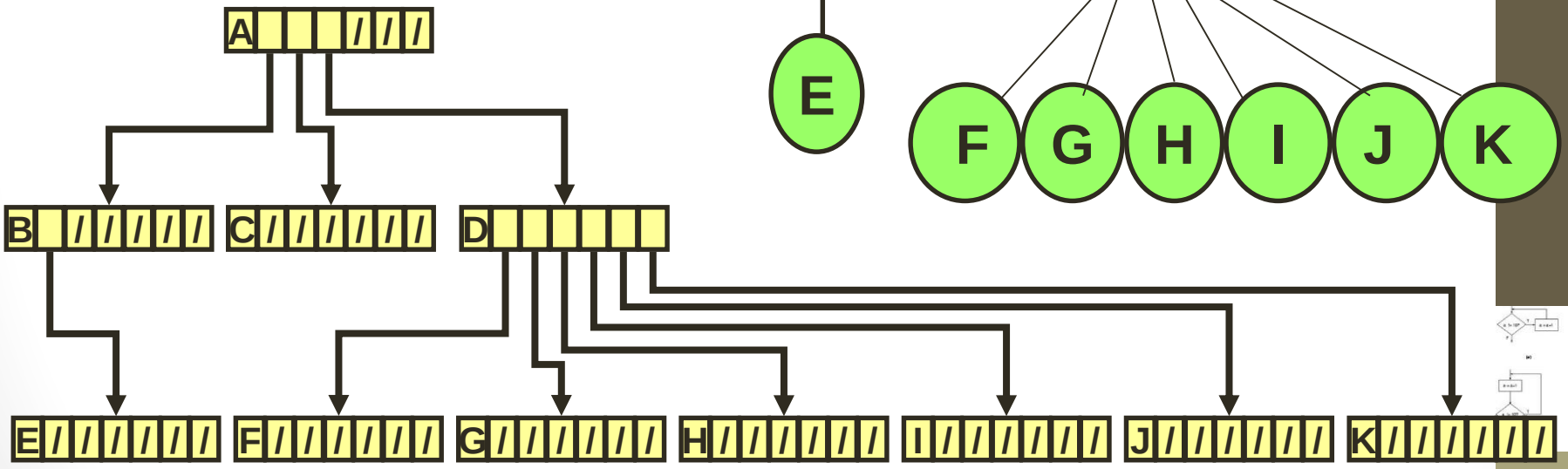
Análise de árvore por encadeamento

Problema:

Árvore de grau arbitrário tem

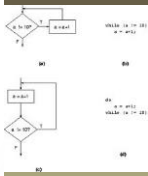
nr. elevado de campos ociosos





Análise de árvore por encadeamento

- Problemas quando grau dos nodos é variado - muitos elos vazios
- Acesso somente através de raiz - disciplina para percorrer árvore
- Inserção e remoção são simplificadas
- Hierarquia é intuitiva → esta forma de implementação é adotada a partir de agora



VER A LISTA DE EXERCÍCIOS QUE ESTARÁ DISPONÍVEL NO BLOG E NO DROPBOX.