Aula 19: Percursos em árvores binárias

- Percurso em pré-ordem, ordem simétrica e pós-ordem
- Algoritmo para cálculo de altura uma árvore binária
- Complexidade dos métodos

Aula 19: Percursos em árvores binárias

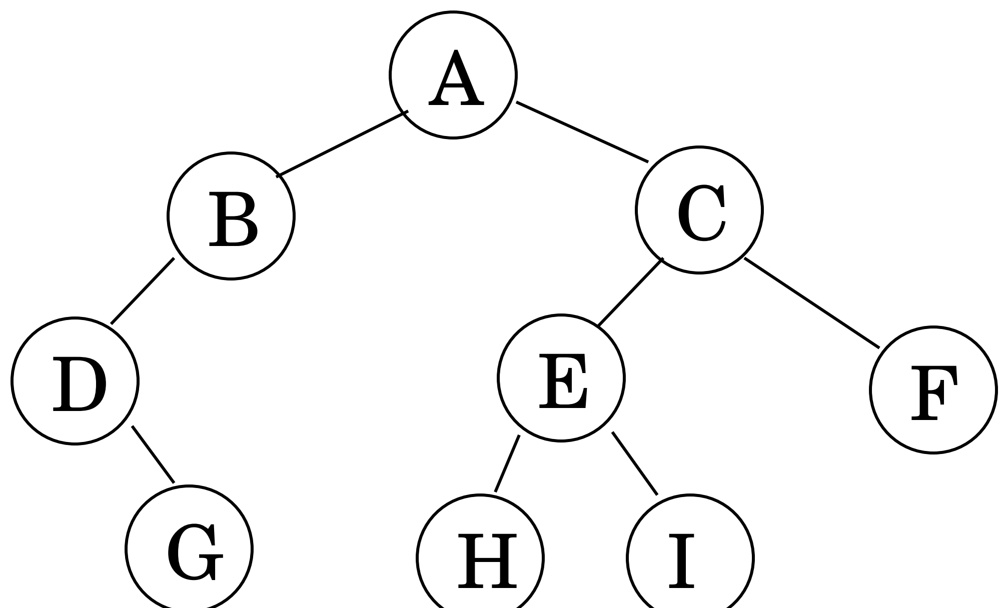
- Percurso em árvores: visita sistemática a seus nós
- Árvores são estruturas não lineares
- Como percorrer uma árvore?
- Objetivo: elaborar algoritmos para percorrer árvores binárias
- Composição dos algoritmos de percurso
  - visita a nós
  - visita às subárvores esquerda e direita dos nós

Operações básicas

- Operações básicas
  - visita a um nó v
  - visita à subárvore esquerda de v
  - visita à subárvore direita de v
- Ordem das operações básicas
  - Resta definir: em que ordem as operações básicas devem ser realizadas?
  - Suposição: as ordens das operações são idênticas, para todos os nós
- Para definir o algoritmo, basta escolher uma ordem para as três operações básicas.
  - Serão analisadas três ordens possíveis para as operações.

Percurso pré-ordem

- Pré-ordem:
  1) visitar raiz;
  2) percorrer sua subárvore esquerda, em pré-ordem;
  3) percorrer sua subárvore direita, em pré-ordem.
- Exemplo:



- Percurso pré-ordem: A B D G C E H I F

Algoritmo pré-ordem

Algoritmo: percurso em pré-ordem

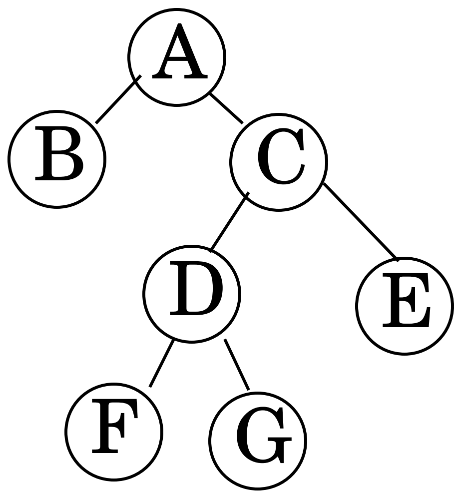
```
procedimento pre( pt )
    visita( pt )
    se pt↑.esq ≠ λ  então pre( pt↑.esq )
    se pt↑.dir ≠ λ  então pre( pt↑.dir )

se ptraiz ≠ λ então pre( ptraiz )
```

`ptraiz` = ponteiro para a raiz da árvore
`pt↑.esq` = ponteiro para o filho esquerdo do nó apontado por pt
`pt↑.dir` = ponteiro para o filho direito do nó apontado por pt
`visita( pt )` = operação da visita ao nó apontado por pt (depende da aplicação)

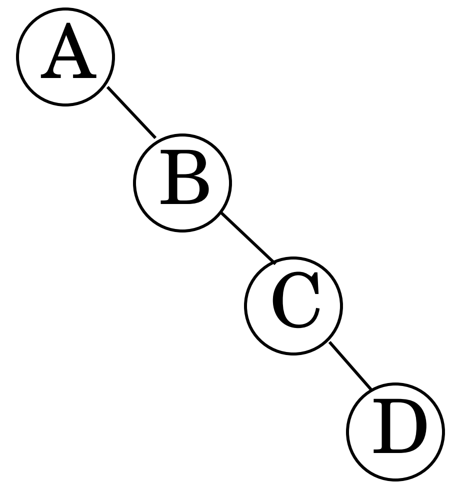
Exercício

- Escrever o percurso pré-ordem, para cada uma das árvores abaixo:

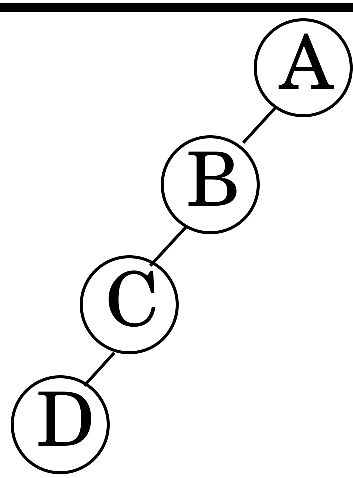
1.



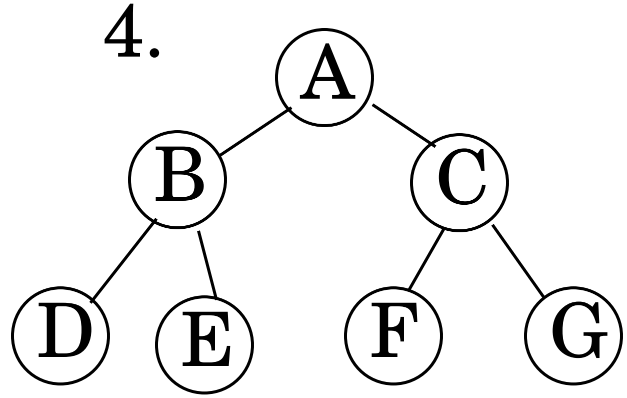
3.



2.



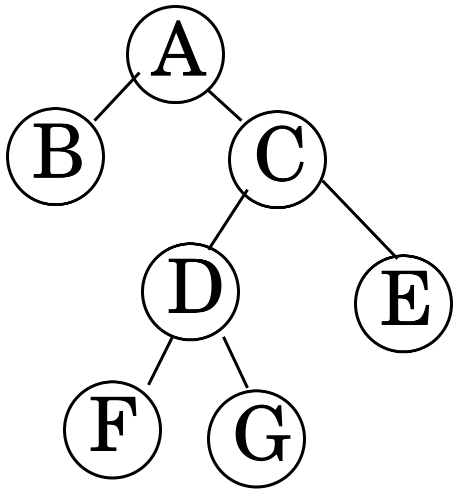
4.



Tempo: 4 minutos.

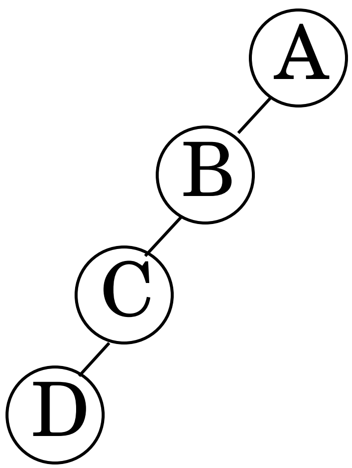
Solução

1.



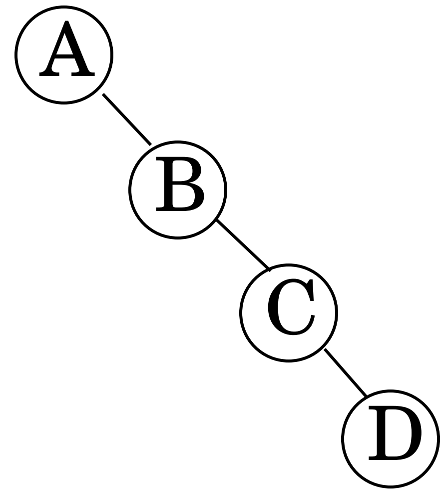
A B C D F G E

2.



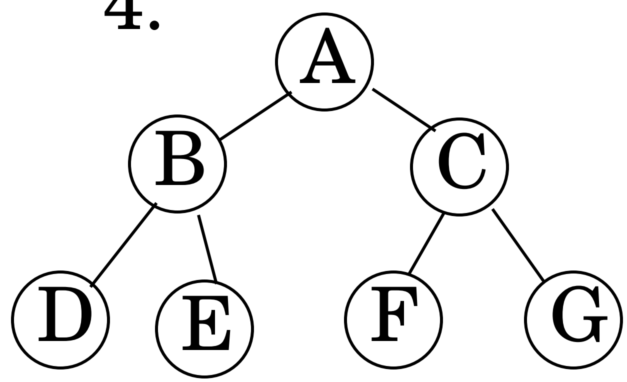
A B C D

3.



A B C D

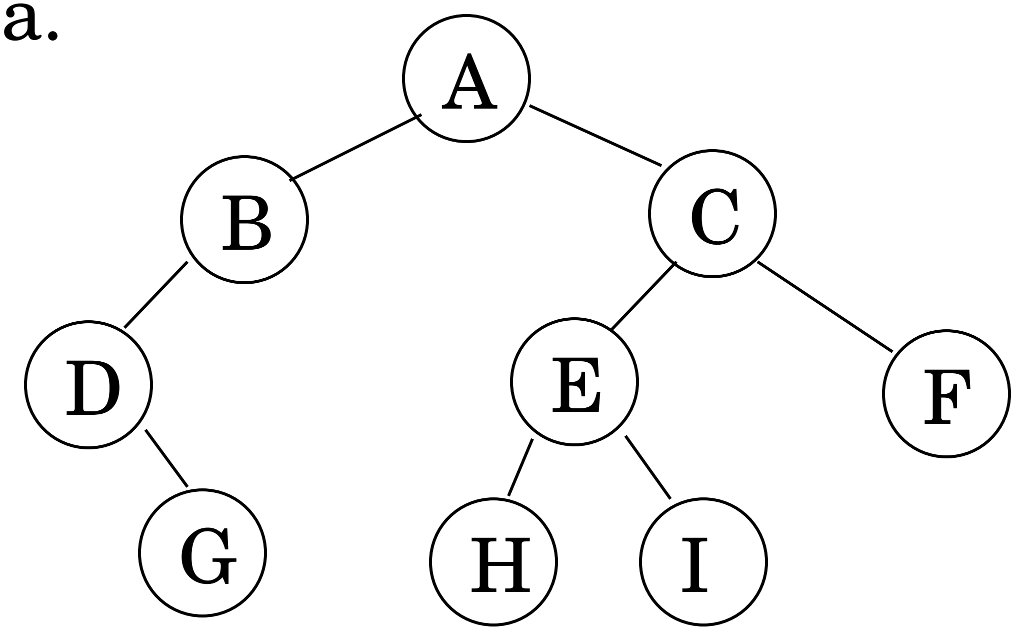
4.



A B D E C F G

Percurso em ordem simétrica

- Ordem simétrica:
  1) percorrer sua subárvore esquerda, em ordem simétrica;
  2) visitar raiz;
  3) percorrer sua subárvore direita, em ordem simétrica.
- Exemplo:



- Percurso ordem simétrica: D G B A H E I C F

Algoritmo ordem simétrica

- Algoritmo: percurso em ordem simétrica

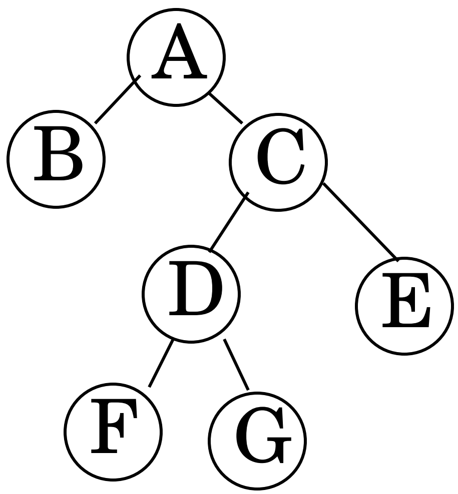
```
procedimento simet( pt )
    se pt↑.esq ≠ λ  então simet( pt↑.esq )
    visita( pt )
    se pt↑.dir ≠ λ  então simet( pt↑.dir )

se ptraiz ≠ λ então simet( ptraiz )
```

- `ptraiz` = ponteiro para a raiz da árvore
  `pt↑.esq` = ponteiro para o filho esquerdo do nó apontado por pt
  `pt↑.dir` = ponteiro para o filho direito do nó apontado por pt
  `visita( pt )` = operação da visita ao nó apontado por pt (depende da aplicação)

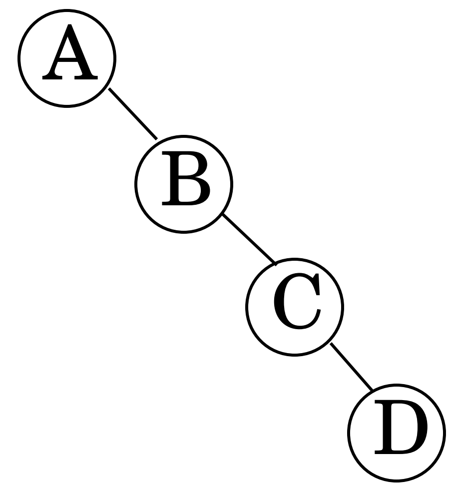
Exercício

Escrever o percurso ordem simétrica, para cada uma das árvores abaixo:

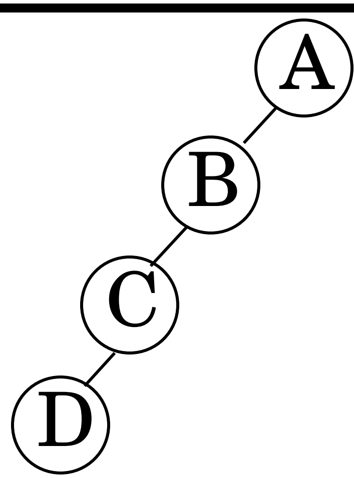
1.



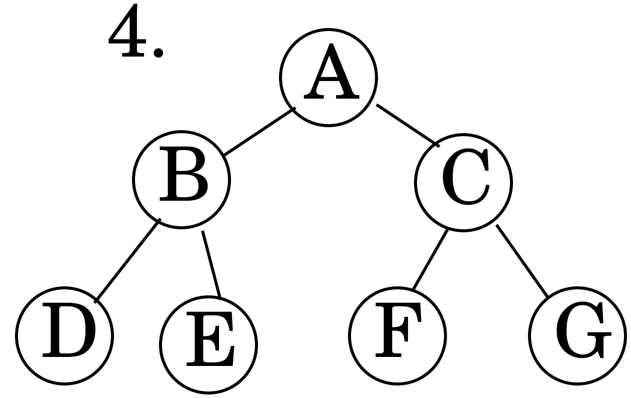
2.



3.



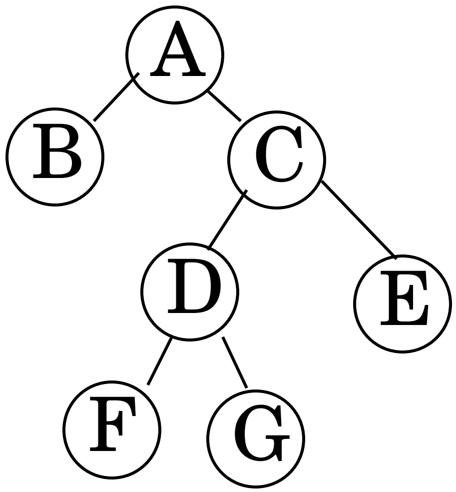
4.



Tempo: 4 minutos.

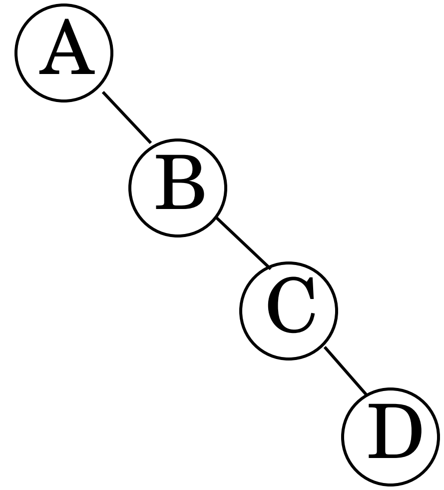
Solução

1.



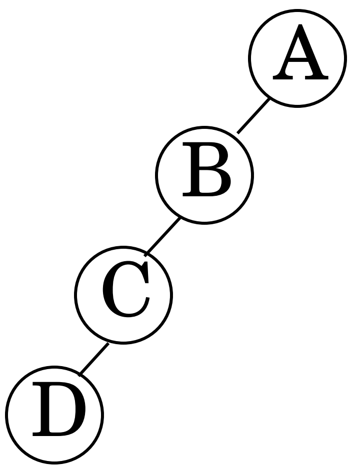
B A F D G C E

2.



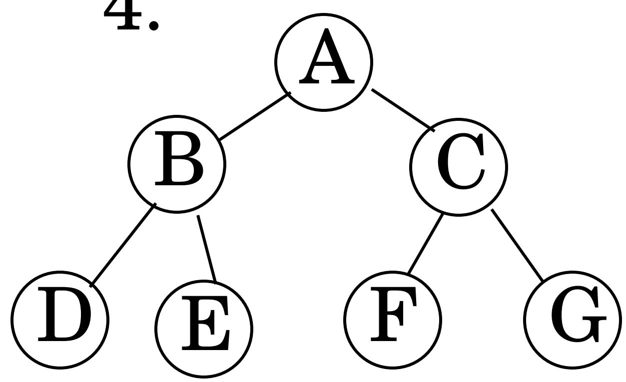
A B C D

3.



D C B A

4.



D B E A F C G

Exercício

- Escrever um algoritmo para desenhar uma árvore binária, supondo que o número de nós não exceda o limite máximo de caracteres que pode ser impresso em uma linha, por uma impressora. Cada nó corresponderá à impressão de um caracter. A saída do algoritmo deverá ser o conjunto de coordenadas (abcissa e ordenada) dos nós da árvore.
  - Sugestão: utilizar como abscissa do nó v a sua posição no percurso ordem simétrica da árvore, e como ordenada, o nível de v.

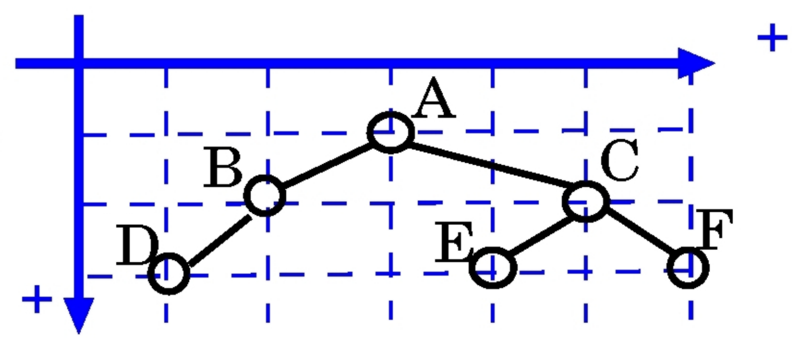
Tempo: 14 minutos

Solução

Algoritmo: impressão de uma árvore

```
procedimento coord( pt, j )
    se pt↑.esq ≠ λ  então coord( pt↑.esq, j + 1 )
    i := in+ 1
    listar ( i, j, pt↑. info )
    se pt↑.dir ≠ λ então coord( pt↑.dir, j + 1 )

i:=0
se ptraiz ≠ λ então coord( ptraiz, 1 )
```

As coordenadas dos nós das árvores correspondem ao conjunto de pares (i, j), listados pelo algoritmo.

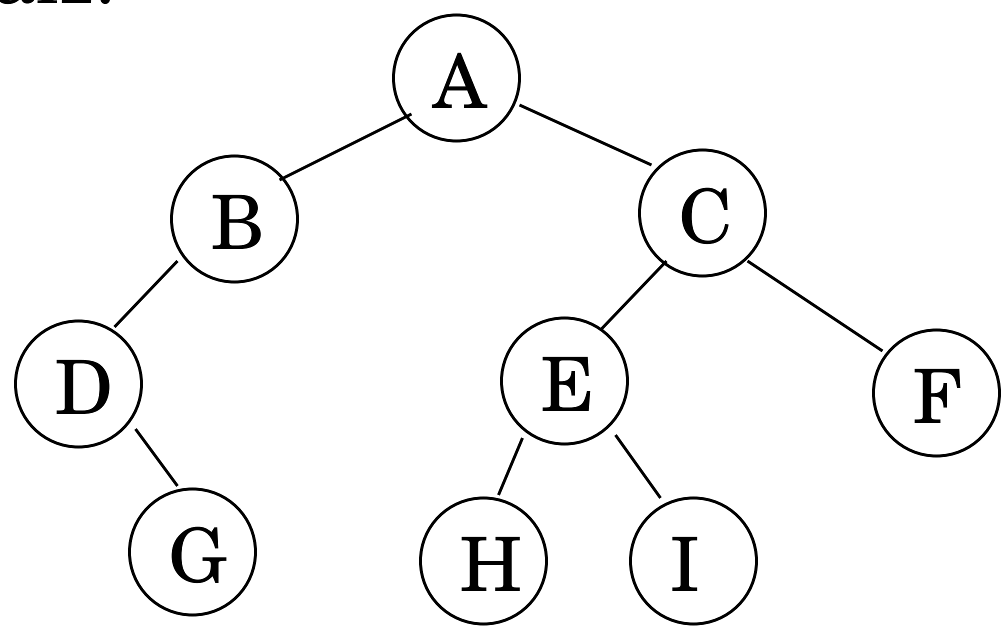


D: (1 , 3)   B: (2 , 2)   A: (3 , 1)
E: (4 , 3)   C: (5 , 2)   F: (6 , 3)

Percurso pós-ordem

- Pós ordem:
  1) percorrer sua subárvore esquerda, em pós-ordem;
  2) percorrer sua subárvore direita, em pós-ordem;
  3) visitar raiz.

- Exemplo:



- Percurso pós ordem: G D B H I E F C A

Algoritmo pós-ordem

Algoritmo: percurso em pós-ordem

```
procedimento pos( pt )
   se pt↑.esq ≠ λ então pos( pt↑.esq )
   se pt↑.dir ≠ λ então pos( pt↑.dir )
   visita( pt )
se ptraiz≠λ então pos( ptraiz )
```

`ptraiz` = ponteiro para a raiz da árvore

`pt↑.esq` = ponteiro para o filho esquerdo do nó apontado por pt

`pt↑.dir` = ponteiro para o filho direito do nó apontado por pt

`visita( pt )` = operação da visita ao nó apontado por pt (depende da aplicação)

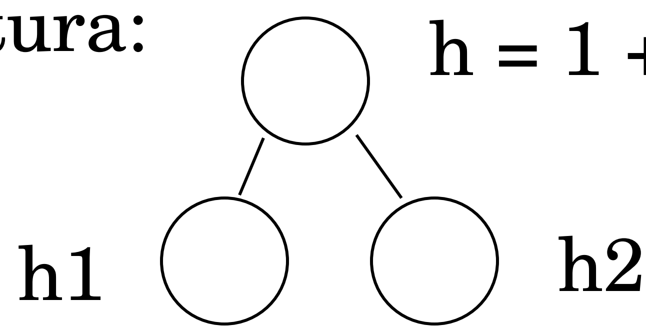
Complexidade do algoritmo de percurso

- Há exatamente uma chamada do procedimento, para cada nó da árvore.
- Em cada chamada, as operações envolvidas podem ser realizadas em tempo constate.
- Consequüência: o algoritmo executa exatamente n passos, onde n é o número de nós da árvore.
- Complexidade: $\Theta(n)$

Aplicação

- Aplicação: determinar a altura de cada nó de uma árvore binária T
- Para determinar a altura de um nó v, é necessário conhecer o comprimento do maior caminho de v, até um de seus descendentes.
- A altura de v somente pode ser calculada após a visita a todos os descendentes de v.
- O percurso pós ordem é o indicado.
- Cálculo da altura:

$$h = 1 + \max \{ h1, h2 \}$$



Aplicação

- Algoritmo: cálculo da altura de um dado nó v de T

```
procedimento altura( pt )
    se pt↑.esq ≠ λ  então h1 := altura ( pt↑.esq )
    senão h1 := 0
    se pt↑.dir ≠ λ então h2 := altura ( pt↑.dir )
    senão h2 := 0
    retornar 1 + max{ h1, h2 }
```

- `pt` = ponteiro para o nó v de T
- A função `altura( pt )` computa o valor da altura do nó apontado por v.
- Complexidade: O(n)

Exercício

- Modificar o algoritmo anterior, de modo a calcular a altura de todos os nós da árvore T.

- Determinar o pior e o melhor caso do algoritmo de cálculo da altura v.

Tempo: 5 minutos.

Solução

- A função altura(`pt`) do algoritmo de cálculo da altura do nó `v`, apontado por pt, determina a altura de todos os nós, quando `v` for a raiz de `T`.
  - Chamada externa: altura(`ptraiz`) onde `ptraiz` é o ponteiro para a raiz de `T`.

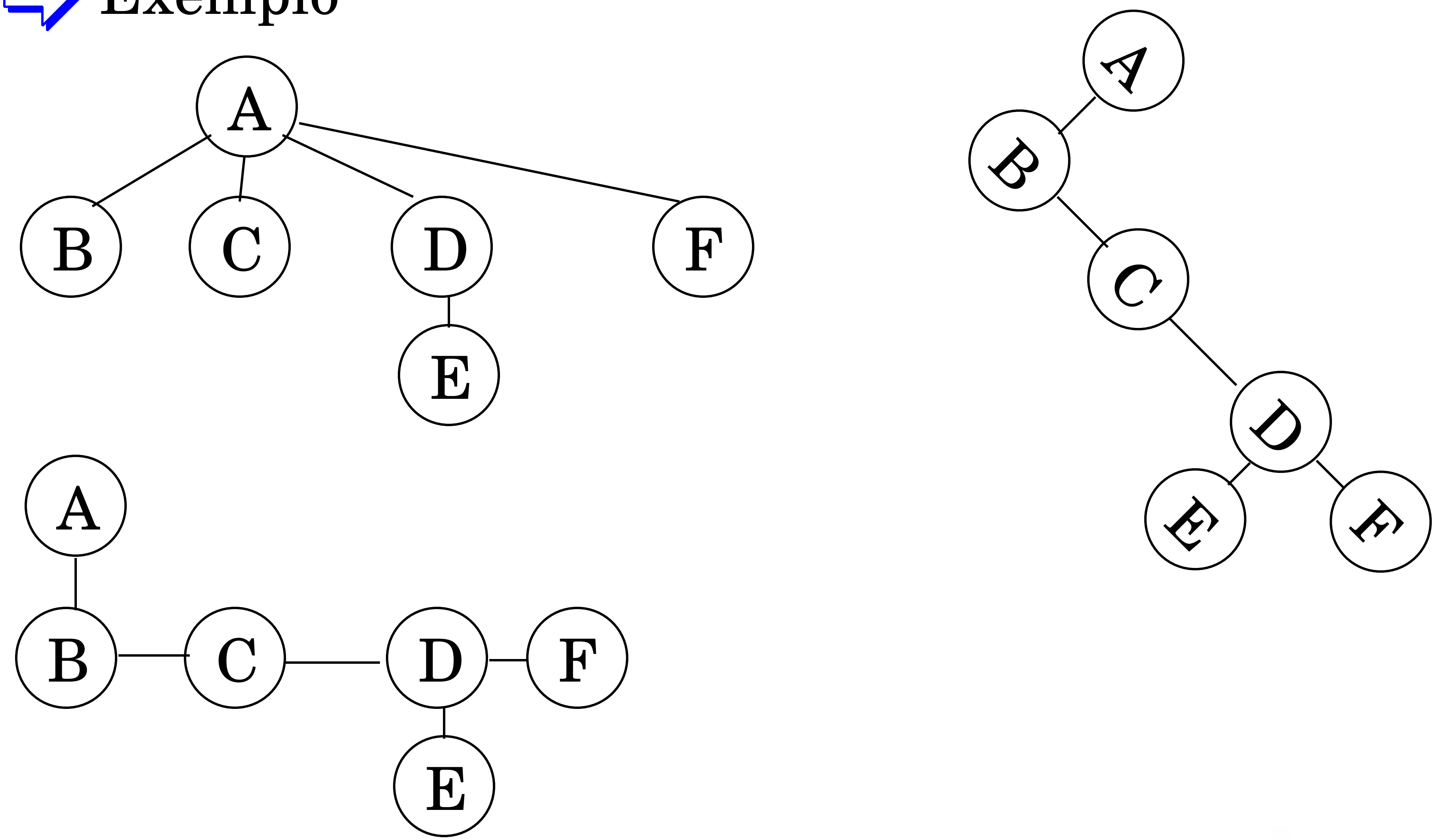
- Melhor caso:
  `v` folha => 1 chamada de altura
  Pior caso:
  `v` raiz => n chamadas de altura

Conversão de uma floresta

- Como representar uma árvore (não binária) T?
  - Idéia básica: Converter a árvore dada T em uma árvore binária B(T).
    Usar a representação conhecida para B(T).
- Determinação de B(T):
  - B(T) possui um nó B(v) para cada nó v de T.
  - O filho esquerdo de B(v) em B(T) corresponde ao primeiro filho de v em T, se existir. Se v for folha, B(v) não possui filho esquerdo.
  - O filho direito de B(v) corresponde ao irmão de v em T, localizado imediatamente a sua direita, caso exista. Se não existir, B(v) não possui filho direito.

Exemplo

- Exemplo

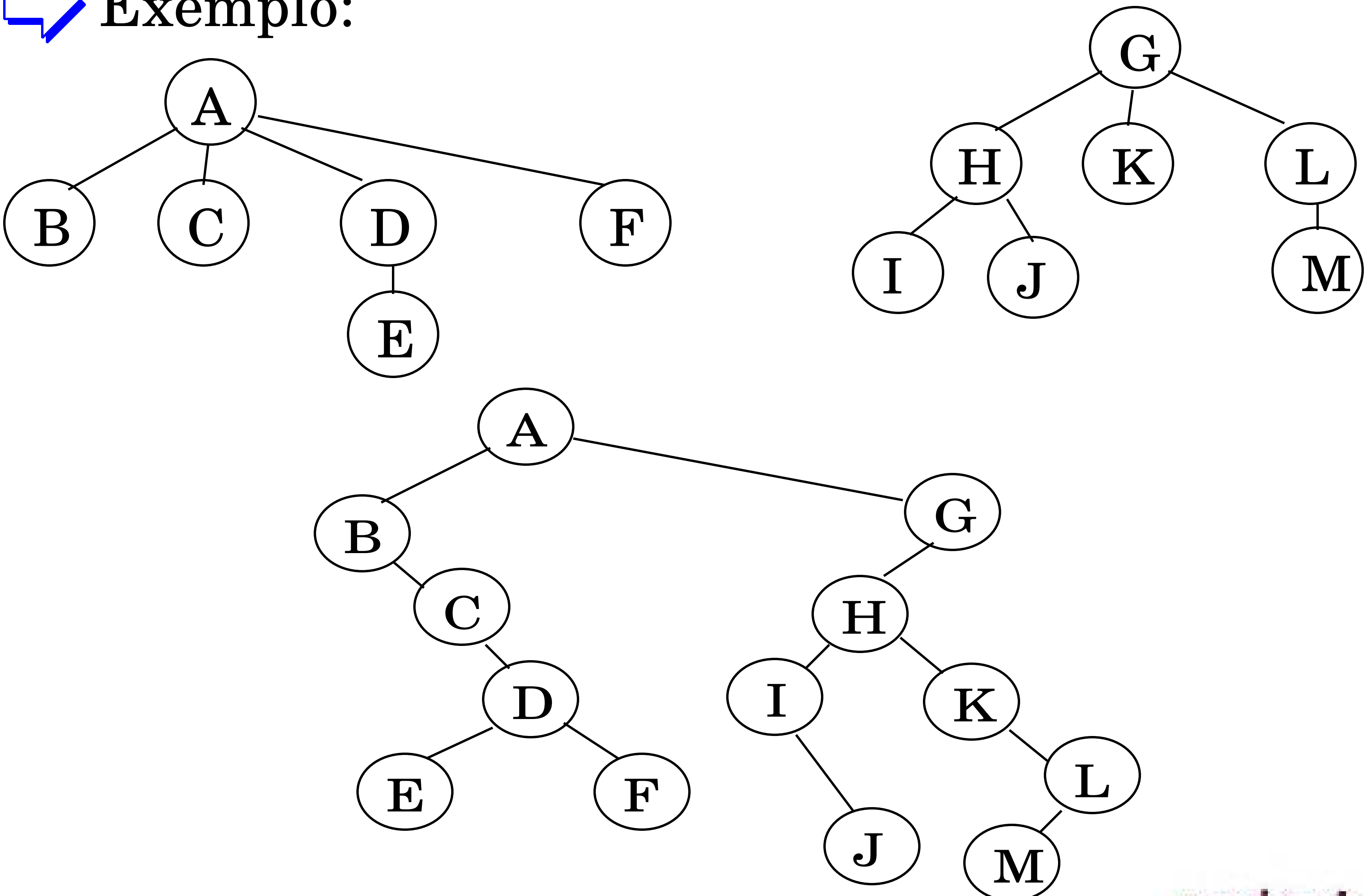


Generalização para florestas

- Conversão de uma floresta: considerar as raízes das árvores das florestas, como nós irmãos, e aplicar o método de conversão de uma árvore.

Exemplo

- Exemplo:



Exercícios finais

- Exercícios finais:
  - Escrever um algoritmo para determinar o pai de um nó v de uma árvore binária T.
  - O percurso em nível de uma árvore binária T é aquele em que os nós são dispostos em ordem não decrescente de seus níveis. Escrever um algoritmo para efetuar o percurso em nível da árvore T.
    Sugestão: utilizar uma fila.
  - Escrever um algoritmo não recursivo para o percurso em pré-ordem de uma árvore binária.
    Sugestão: utilizar uma pilha.
  - Utilizando a representação de uma expressão aritmética através de uma árvore binária, escrever um algoritmo simples para determinar a notação polonesa da expressão aritmética.