Árvores

Fábio Henrique Viduani Martinez

Faculdade de Computação
Universidade Federal de Mato Grosso do Sul

Estruturas de Dados

Introdução

- As estruturas de dados estudadas até o momento são sequenciais: pilhas, filas, listas, tabelas de dispersão, listas de prioridades (heaps)
- Começaremos a estudar **árvores**, estruturas de dados mais complexas, utilizadas em variadas aplicações

Introdução

- As estruturas de dados estudadas até o momento são sequenciais: pilhas, filas, listas, tabelas de dispersão, listas de prioridades (heaps)
- Começaremos a estudar **árvores**, estruturas de dados mais complexas, utilizadas em variadas aplicações

Árvores

Uma árvore T é um conjunto finito de elementos chamados *nós* ou *vértices*, de forma que:

- $T = \emptyset$, ou seja, temos uma árvore *vazia*, ou
- Existe um nó r chamado *raiz* de T, e:
  - r é o único nó de T, ou
  - os demais são divididos em até $n \geq 1$ conjuntos disjuntos não vazios, as subárvores de r, cada qual por sua vez uma árvore.

Árvores

Uma árvore T é um conjunto finito de elementos chamados *nós* ou *vértices*, de forma que:

- $T = \emptyset$, ou seja, temos uma árvore *vazia*, ou
- Existe um nó r chamado *raiz* de T, e:
  - r é o único nó de T, ou
  - os demais são divididos em até $m \geq 1$ conjuntos disjuntos não vazios, as *subárvores* de r, cada qual por sua vez uma árvore

Árvores

Uma árvore T é um conjunto finito de elementos chamados *nós* ou *vértices*, de forma que:

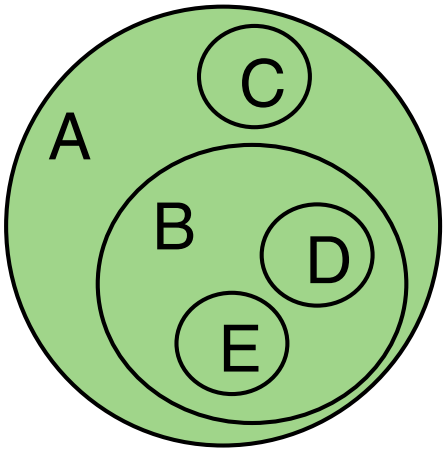
- $T = \emptyset$, ou seja, temos uma árvore *vazia*, ou
- Existe um nó r chamado *raiz* de T, e:
  - r é o único nó de T, ou
  - os demais são divididos em até $m \geq 1$ conjuntos disjuntos não vazios, as *subárvores* de r, cada qual por sua vez uma árvore

Árvores

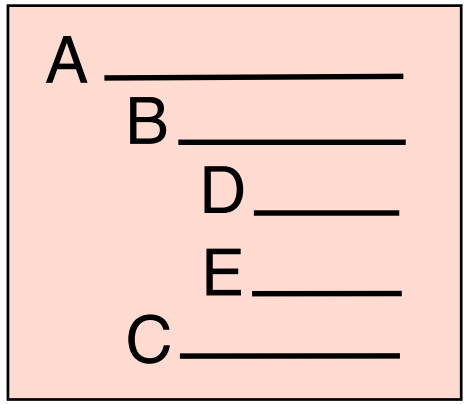
Uma árvore T é um conjunto finito de elementos chamados *nós* ou *vértices*, de forma que:

- $T = \emptyset$, ou seja, temos uma árvore *vazia*, ou
- Existe um nó r chamado *raiz* de T, e:
  - r é o único nó de T, ou
  - os demais são divididos em até $m \geq 1$ conjuntos disjuntos não vazios, as *subárvores* de r, cada qual por sua vez uma árvore

Árvores – Representações



a)



b)

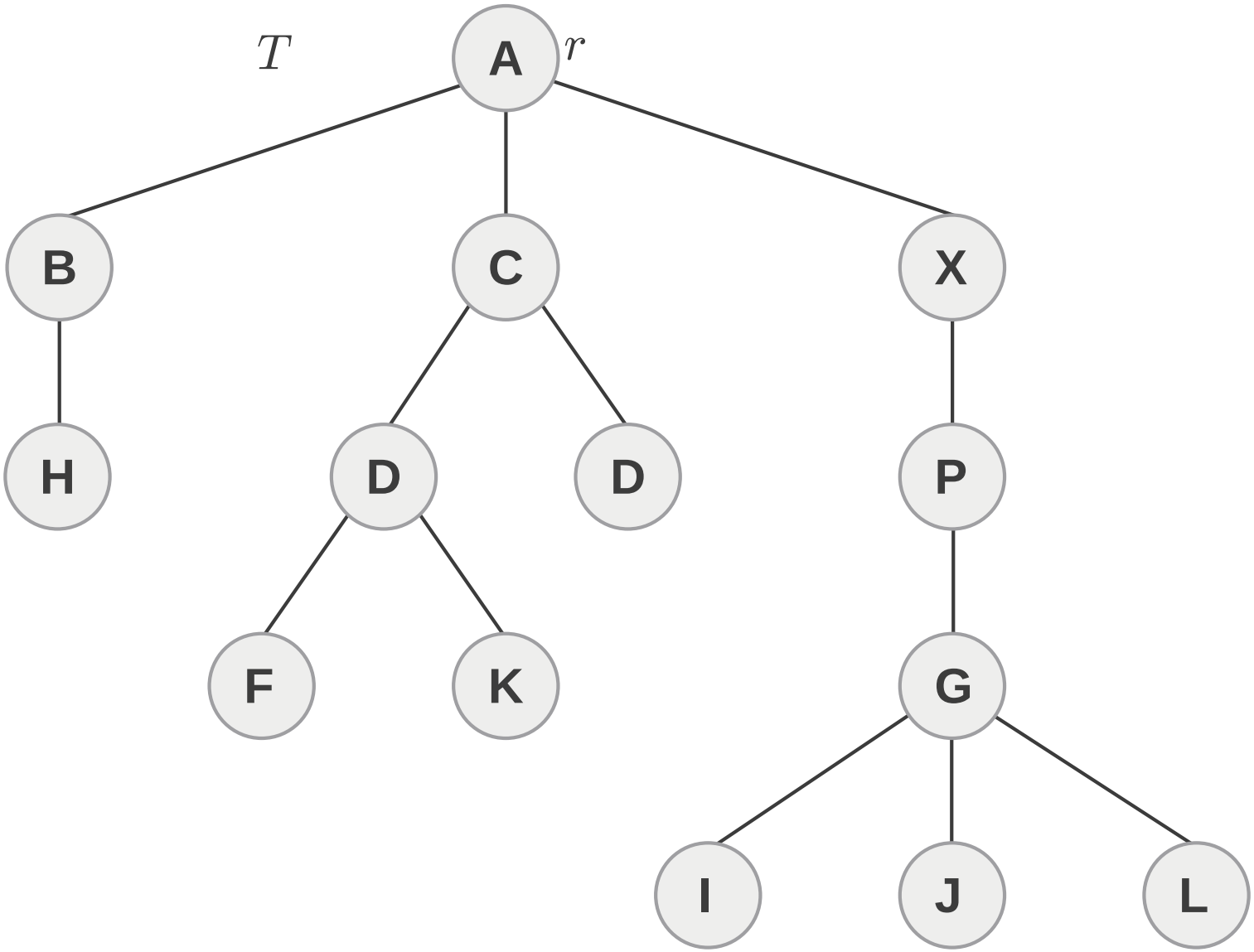
1A; 1.1B; 1.1.1D; 1.1.2E; 1.2C

c)

(A(B(D)(E))(C))

d)

Árvores – Representação padrão

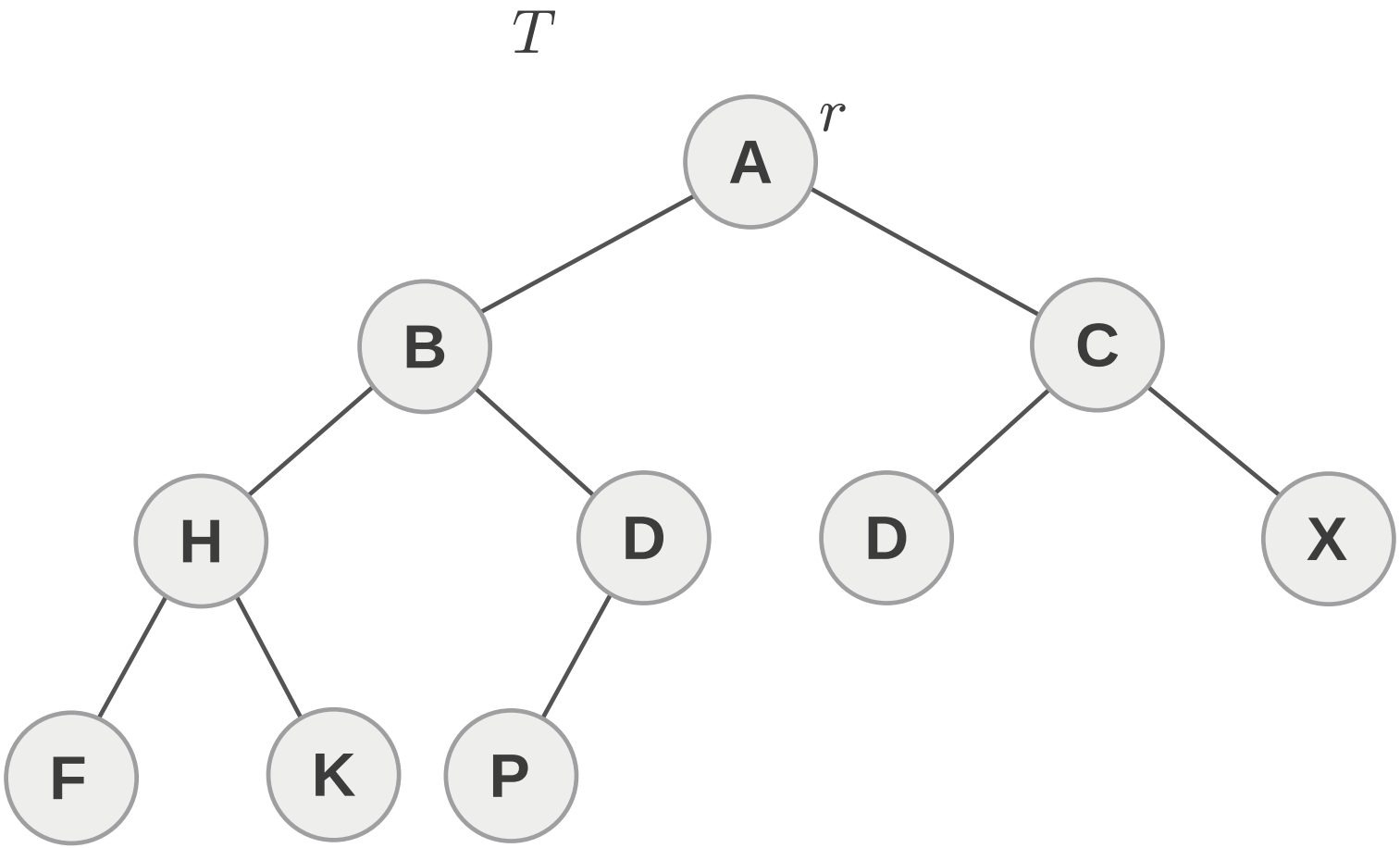


Árvores binárias

Caso especial onde $1 \leq m \leq 2$:

- $T = \emptyset$, ou seja, temos uma árvore *vazia*, ou
- Existe um nó r chamado *raiz* de T, e:
  - r é o único nó de T, ou
  - os demais são divididos em até $1 \leq m \leq 2$ conjuntos disjuntos não vazios, as *subárvores* de r, cada qual por sua vez uma árvore

Representação



Conceitos

Seja um nó v:

- T_v é a subárvore de T com raiz v
- Nós raízes u e w das subárvores de T_v (caso existam) são chamados *filhos* de v
- v é chamado *pai* de u e w
- Os nós u e w são *irmãos*

* esses e demais conceitos se estendem para árvores não binárias

Conceitos (2)

- Um nó que não tem filhos é uma *folha*
- Um *caminho* na árvore é uma sequência de nós distintos $v_1, \ldots, v_k$ tal que existe sempre entre nós consecutivos a relação "é pai de" ou "é filho de"
- Dado um nó v, nós que estão no caminho de v até a raiz são *ancestrais* ou *ascendentes* (incluindo a raiz)
- Dado um nó v, nós que estão em um caminho de v até uma folha são *descendentes* (incluindo a folha)
- Dado um nó v, T_v^E e T_v^D são suas *subárvores esquerda* e *direita*

Conceitos (2)

- Um nó que não tem filhos é uma *folha*
- Um *caminho* na árvore é uma sequência de nós distintos $v_1, \ldots, v_k$ tal que existe sempre entre nós consecutivos a relação "é pai de" ou "é filho de"
- Dado um nó v, nós que estão no caminho de v até a raiz são *ancestrais* ou *ascendentes* (incluindo a raiz)
- Dado um nó v, nós que estão em um caminho de v até uma folha são *descendentes* (incluindo a folha)
- Dado um nó v, T_v^E e T_v^D são suas *subárvores esquerda* e *direita*

Conceitos (2)

- Um nó que não tem filhos é uma *folha*
- Um *caminho* na árvore é uma sequência de nós distintos $v_1, \ldots, v_k$ tal que existe sempre entre nós consecutivos a relação "é pai de" ou "é filho de"
- Dado um nó v, nós que estão no caminho de v até a raiz são *ancestrais* ou *ascendentes* (incluindo a raiz)
- Dado um nó v, nós que estão em um caminho de v até uma folha são *descendentes* (incluindo a folha)
- Dado um nó v, T_v^E e T_v^D são suas *subárvores esquerda* e *direita*

Conceitos (2)

- Um nó que não tem filhos é uma *folha*
- Um *caminho* na árvore é uma sequência de nós distintos $v_1, \ldots, v_k$ tal que existe sempre entre nós consecutivos a relação "é pai de" ou "é filho de"
- Dado um nó v, nós que estão no caminho de v até a raiz são *ancestrais* ou *ascendentes* (incluindo a raiz)
- Dado um nó v, nós que estão em um caminho de v até uma folha são *descendentes* (incluindo a folha)
- Dado um nó v, T_v^E e T_v^D são suas *subárvores esquerda* e *direita*

Conceitos (2)

- Um nó que não tem filhos é uma *folha*
- Um *caminho* na árvore é uma sequência de nós distintos $v_1, \ldots, v_k$ tal que existe sempre entre nós consecutivos a relação “é pai de” ou “é filho de”
- Dado um nó v, nós que estão no caminho de v até a raiz são *ancestrais* ou *ascendentes* (incluindo a raiz)
- Dado um nó v, nós que estão em um caminho de v até uma folha são *descendentes* (incluindo a folha)
- Dado um nó v, T_v^E e T_v^D são suas *subárvores esquerda* e *direita*

Conceitos (3)

- O número de *saltos* ou *passos* de um caminho com n nós é $n - 1$
- O *nível* (ou *profundidade*) de um nó v é o número de saltos de r até v (o nível da raiz é 0)
- A *altura* $h(v)$ de um nó v é o maior número de saltos de v até uma folha descendente (a altura das folhas é 0)
- A *altura* de uma árvore T é a altura da raiz r, ou seja, $h(T) = h(r)$

Conceitos (3)

- O número de *saltos* ou *passos* de um caminho com n nós é $n - 1$
- O *nível* (ou *profundidade*) de um nó v é o número de saltos de r até v (o nível da raiz é 0)
- A *altura* $h(v)$ de um nó v é o maior número de saltos de v até uma folha descendente (a altura das folhas é 0)
- A *altura* de uma árvore T é a altura da raiz r, ou seja, $h(T) = h(r)$

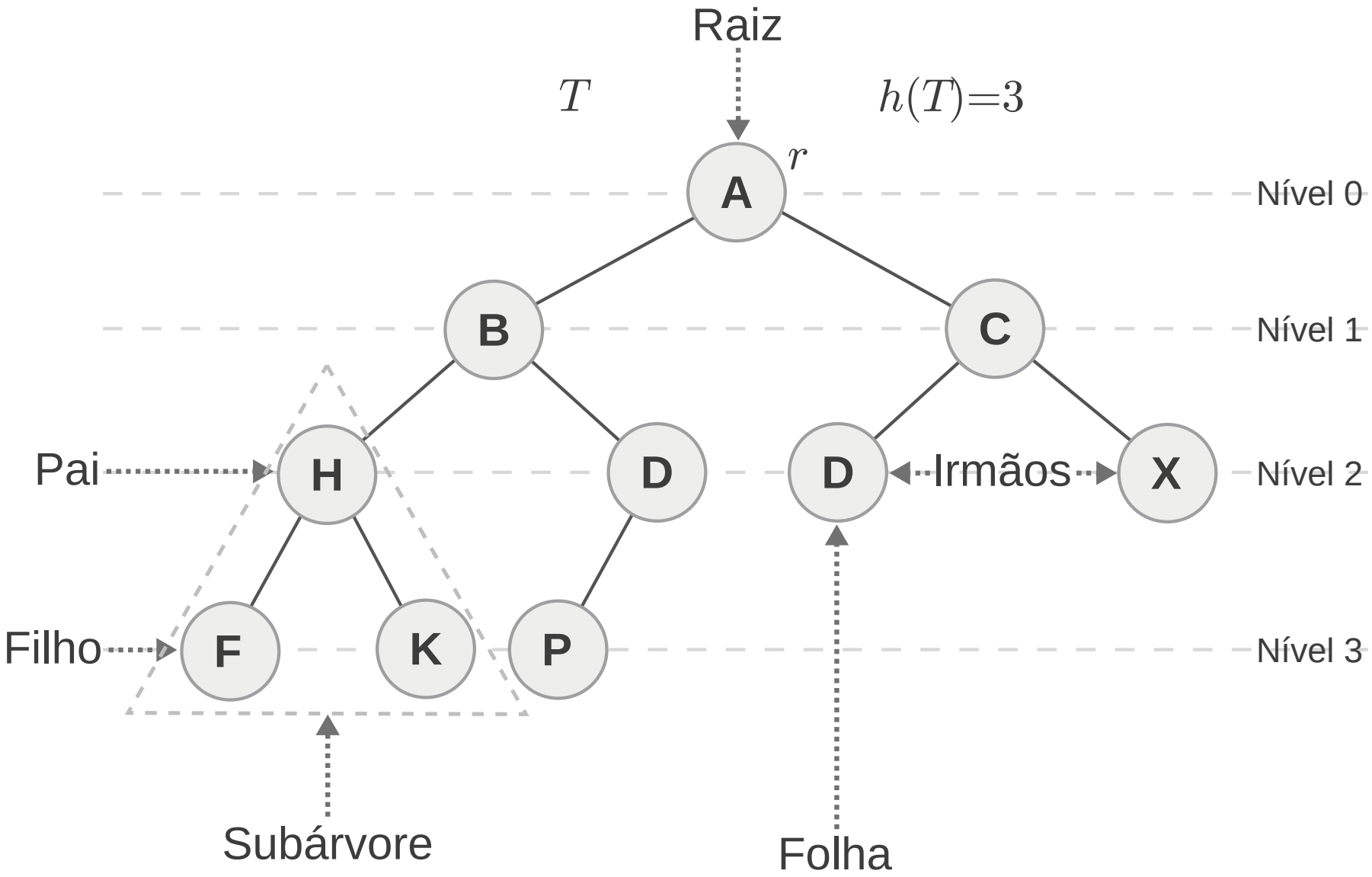
Conceitos (3)

- O número de *saltos* ou *passos* de um caminho com n nós é $n - 1$
- O *nível* (ou *profundidade*) de um nó v é o número de saltos de r até v (o nível da raiz é 0)
- A *altura* $h(v)$ de um nó v é o maior número de saltos de v até uma folha descendente (a altura das folhas é 0)
- A *altura* de uma árvore T é a altura da raiz r, ou seja, $h(T) = h(r)$

Conceitos (3)

- O número de *saltos* ou *passos* de um caminho com n nós é $n - 1$
- O *nível* (ou *profundidade*) de um nó v é o número de saltos de r até v (o nível da raiz é 0)
- A *altura* $h(v)$ de um nó v é o maior número de saltos de v até uma folha descendente (a altura das folhas é 0)
- A *altura* de uma árvore T é a altura da raiz r, ou seja, $h(T) = h(r)$

Conceitos (4)


Raiz
T
$h(T)=3$
r
A
B
C
H
D
D
X
F
K
P
Nível 0
Nível 1
Nível 2
Nível 3
Pai
Irmãos
Filho
Subárvore
Folha

Árvores binárias particulares

- **Árvore estritamente binária**: cada nó possui 0 ou 2 filhos
- **Árvore binária completa**: todos os níveis têm o máximo número de elementos, exceto possivelmente o último
- **Árvore zigue-zague ou degenerada**: todo nó tem exatamente 1 filho, exceto a folha
- Dada uma quantidade n de nós, a árvore contendo n nós com a menor altura possível é a árvore completa
- Dada uma quantidade n de nós, a árvore contendo n nós com a maior altura possível é a árvore zigue-zague

Árvores binárias particulares

- **Árvore estritamente binária**: cada nó possui 0 ou 2 filhos
- **Árvore binária completa**: todos os níveis têm o máximo número de elementos, exceto possivelmente o último
- **Árvore zigue-zague ou degenerada**: todo nó tem exatamente 1 filho, exceto a folha
- Dada uma quantidade n de nós, a árvore contendo n nós com a menor altura possível é a árvore completa
- Dada uma quantidade n de nós, a árvore contendo n nós com a maior altura possível é a árvore zigue-zague

Árvores binárias particulares

- **Árvore estritamente binária**: cada nó possui 0 ou 2 filhos
- **Árvore binária completa**: todos os níveis têm o máximo número de elementos, exceto possivelmente o último
- **Árvore zigue-zague ou degenerada**: todo nó tem exatamente 1 filho, exceto a folha
- Dada uma quantidade n de nós, a árvore contendo n nós com a menor altura possível é a árvore completa
- Dada uma quantidade n de nós, a árvore contendo n nós com a maior altura possível é a árvore zigue-zague

Árvores binárias particulares

- **Árvore estritamente binária**: cada nó possui 0 ou 2 filhos
- **Árvore binária completa**: todos os níveis têm o máximo número de elementos, exceto possivelmente o último
- **Árvore zigue-zague ou degenerada**: todo nó tem exatamente 1 filho, exceto a folha
- Dada uma quantidade n de nós, a árvore contendo n nós com a menor altura possível é a árvore completa
- Dada uma quantidade n de nós, a árvore contendo n nós com a maior altura possível é a árvore zigue-zague

Árvores binárias particulares

- **Árvore estritamente binária**: cada nó possui 0 ou 2 filhos
- **Árvore binária completa**: todos os níveis têm o máximo número de elementos, exceto possivelmente o último
- **Árvore zigue-zague ou degenerada**: todo nó tem exatamente 1 filho, exceto a folha
- Dada uma quantidade n de nós, a árvore contendo n nós com a menor altura possível é a árvore completa
- Dada uma quantidade n de nós, a árvore contendo n nós com a maior altura possível é a árvore zigue-zague

Percursos em árvores binárias

- Um *percurso* é uma visitação sistemática aos nós de uma árvore
- *Visitar* um nó é acessá-lo e verificar e/ou manipular de alguma forma as informações contidas nele (p. ex., verificar ou escrever qual é o valor armazenado)
- Os percursos se caracterizam pela ordem com que os nós são visitados
- Uma árvore binária pode ser percorrida de muitas maneiras diferentes

Percursos em árvores binárias

- Um *percurso* é uma visitação sistemática aos nós de uma árvore
- *Visitar* um nó é acessá-lo e verificar e/ou manipular de alguma forma as informações contidas nele (p. ex., verificar ou escrever qual é o valor armazenado)
- Os percursos se caracterizam pela ordem com que os nós são visitados
- Uma árvore binária pode ser percorrida de muitas maneiras diferentes

Percursos em árvores binárias

- Um *percurso* é uma visitação sistemática aos nós de uma árvore
- *Visitar* um nó é acessá-lo e verificar e/ou manipular de alguma forma as informações contidas nele (p. ex., verificar ou escrever qual é o valor armazenado)
- Os percursos se caracterizam pela ordem com que os nós são visitados
- Uma árvore binária pode ser percorrida de muitas maneiras diferentes

Percursos em árvores binárias

- Um *percurso* é uma visitação sistemática aos nós de uma árvore
- *Visitar* um nó é acessá-lo e verificar e/ou manipular de alguma forma as informações contidas nele (p. ex., verificar ou escrever qual é o valor armazenado)
- Os percursos se caracterizam pela ordem com que os nós são visitados
- Uma árvore binária pode ser percorrida de muitas maneiras diferentes

Percursos em árvores binárias

Dada a raiz r de uma (sub)árvore:

- Percurso *pré-ordem* ou r-e-d ●:
  visita $r \rightarrow$ percorre T_r^E pré-ordem $\rightarrow$ percorre T_r^D pré-ordem
- Percurso *pós-ordem* ou e-d-r ●:
  percorre T_r^E pós-ordem $\rightarrow$ percorre T_r^D pós-ordem $\rightarrow$ visita r
- Percurso *em-ordem* ou e-r-d ●:
  percorre T_r^E em-ordem $\rightarrow$ visita $r \rightarrow$ percorre T_r^D em-ordem
- Note que, dado um nó v, percorremos sempre T_v^E e posteriormente T_v^D
- A diferença entre os percursos é **quando visitamos** v:

Percursos em árvores binárias

Dada a raiz r de uma (sub)árvore:

- Percurso *pré-ordem* ou r-e-d ●:
  visita $r \rightarrow$ percorre T_r^E pré-ordem $\rightarrow$ percorre T_r^D pré-ordem
- Percurso *pós-ordem* ou e-d-r ●:
  percorre T_r^E pós-ordem $\rightarrow$ percorre T_r^D pós-ordem $\rightarrow$ visita r
- Percurso *em-ordem* ou e-r-d ●:
  percorre T_r^E em-ordem $\rightarrow$ visita $r \rightarrow$ percorre T_r^D em-ordem
- Note que, dado um nó v, percorremos sempre T_v^E e posteriormente T_v^D
- A diferença entre os percursos é **quando visitamos** v:

Percursos em árvores binárias

Dada a raiz r de uma (sub)árvore:

- Percurso *pré-ordem* ou r-e-d ●:
  visita $r \rightarrow$ percorre T_r^E pré-ordem $\rightarrow$ percorre T_r^D pré-ordem
- Percurso *pós-ordem* ou e-d-r ●:
  percorre T_r^E pós-ordem $\rightarrow$ percorre T_r^D pós-ordem $\rightarrow$ visita r
- Percurso *em-ordem* ou e-r-d ●:
  percorre T_r^E em-ordem $\rightarrow$ visita $r \rightarrow$ percorre T_r^D em-ordem
- Note que, dado um nó v, percorremos sempre T_v^E e posteriormente T_v^D
- A diferença entre os percursos é **quando visitamos** v:

Percursos em árvores binárias

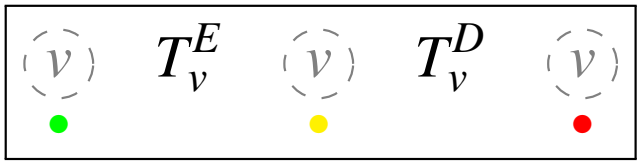
Dada a raiz r de uma (sub)árvore:

- Percurso *pré-ordem* ou r-e-d ●:
  visita $r \rightarrow$ percorre T_r^E pré-ordem $\rightarrow$ percorre T_r^D pré-ordem
- Percurso *pós-ordem* ou e-d-r ●:
  percorre T_r^E pós-ordem $\rightarrow$ percorre T_r^D pós-ordem $\rightarrow$ visita r
- Percurso *em-ordem* ou e-r-d ●:
  percorre T_r^E em-ordem $\rightarrow$ visita $r \rightarrow$ percorre T_r^D em-ordem
- Note que, dado um nó v, percorremos sempre T_v^E e posteriormente T_v^D
- A diferença entre os percursos é **quando visitamos** v:

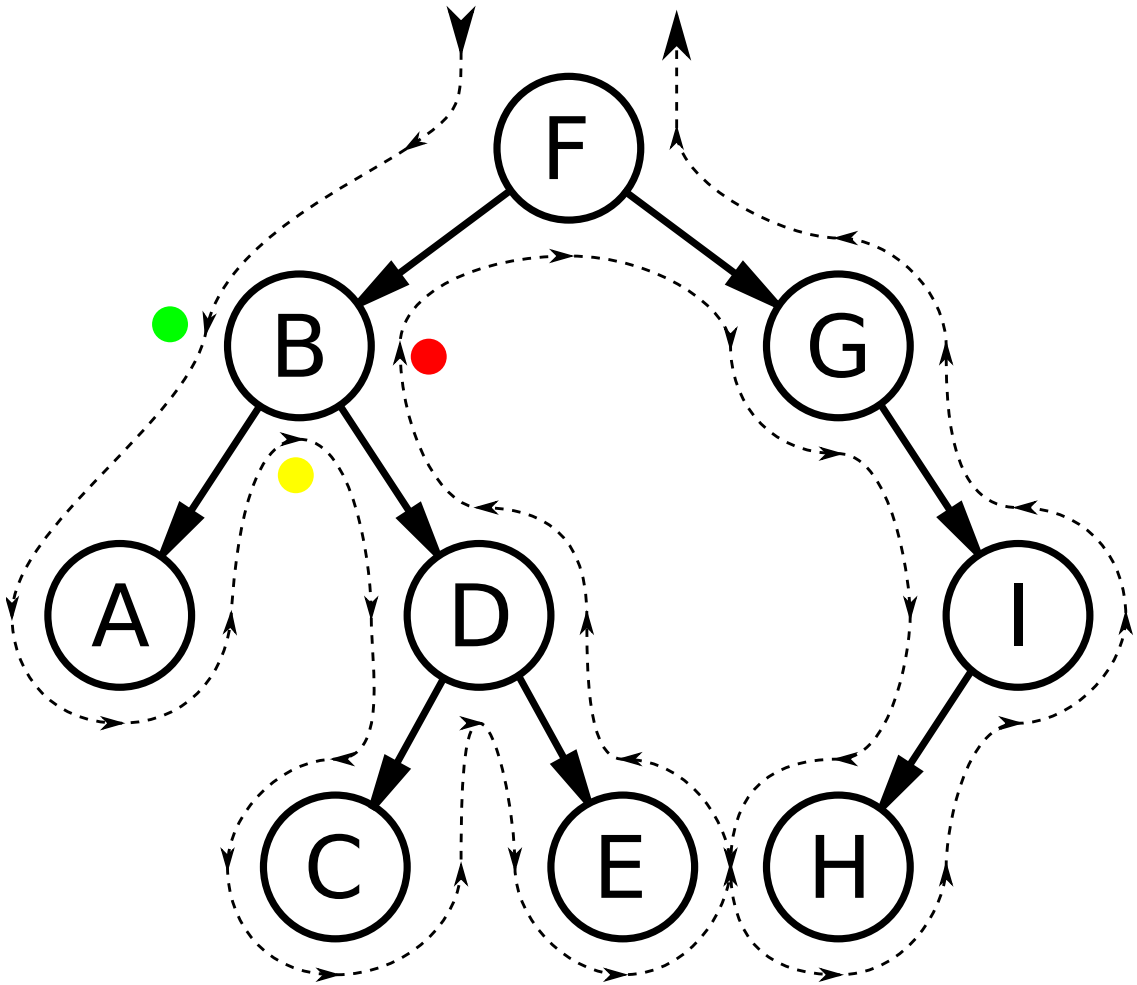
Percursos em árvores binárias

Dada a raiz r de uma (sub)árvore:

- Percurso *pré-ordem* ou r-e-d ●:
  visita $r \rightarrow$ percorre T_r^E pré-ordem $\rightarrow$ percorre T_r^D pré-ordem
- Percurso *pós-ordem* ou e-d-r ●:
  percorre T_r^E pós-ordem $\rightarrow$ percorre T_r^D pós-ordem $\rightarrow$ visita r
- Percurso *em-ordem* ou e-r-d ●:
  percorre T_r^E em-ordem $\rightarrow$ visita $r \rightarrow$ percorre T_r^D em-ordem
- Note que, dado um nó v, percorremos sempre T_v^E e posteriormente T_v^D
- A diferença entre os percursos é **quando visitamos** v:



Percursos em árvores binárias



Exercícios

4.1 Mostre que toda árvore pode ser representada por uma sequência binária. Quantos 0's e 1's possui essa sequência?

4.2 Representar, através de uma árvore, a seguinte expressão aritmética:

$$[(a+b)(c+d)/e] - [(f+g)h] \, .$$

4.3 Prove ou dê um contraexemplo. Se v é o pai de um nó w de uma árvore T, então:

(1) $\mathit{nível}(v) = \mathit{nível}(w) + 1$;
(2) $\mathit{altura}(v) = \mathit{altura}(w) + 1$;
(3) $\max_{v \in T}\{\mathit{altura}(v)\} = \max_{v \in T}\{\mathit{nível}(v)\}$.

4.4 Mostre que toda árvore com $n > 1$ nós possui no mínimo 1 e no máximo $n - 1$ folhas.

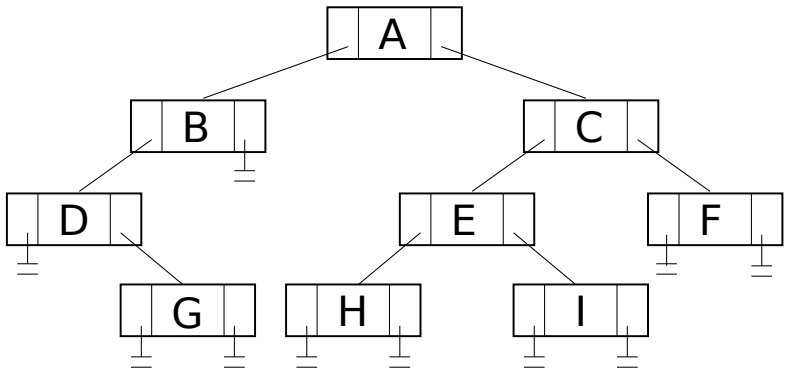
4.5 Prove a afirmação a seguir ou dê um contraexemplo. Uma árvore binária é completa se e somente se ela possuir altura mínima para um dado número de nós.

Exercícios

4.6 Seja um percurso definido pelas seguintes operações:

- Ordem A
  - visitar a raiz;
  - percorrer a subárvore esquerda de v na ordem A;
  - percorrer a subárvore direita de v na ordem B.
- Ordem B
  - percorrer a subárvore esquerda de v na ordem B;
  - visitar a raiz;
  - percorrer a subárvore direita de v na ordem A.

Supondo que o processo se inicie pela raiz da árvore, em ordem A, escreva o percurso final obtido quando o algoritmo for aplicado à árvore da Figura 1.



Exercícios

4.7 Uma expressão aritmética, em *notação polonesa*, é definida recursivamente da seguinte maneira: Uma expressão, em notação polonesa, consiste de um operando ou, então, de um operador seguido por duas expressões em notação polonesa. Em vista disso, toda expressão aritmética pode ser escrita de forma não ambígua em notação polonesa, dispensando-se o uso de parênteses.

(1) Converta a expressão aritmética $(A + B)/(C + D)$ para notação polonesa.

(2) Escreva um algoritmo para transformar uma dada expressão aritmética em notação polonesa.

4.8 Dada a árvore binária que representa uma expressão aritmética (considerando-se apenas operações binárias), gere a mesma expressão em notação completamente parentizada. *Sugestão*: use o percurso em ordem.

Exercícios

4.9 Escreva um algoritmo para determinar o número de nós das subárvores de v, para cada nó v de uma árvore binária.

4.10 Escreva um algoritmo que, dada uma árvore binária T, calcule e devolva o número de nós de T.

4.11 Escreva um algoritmo para percorrer em nível uma árvore binária. *Sugestão*: use uma fila.

4.12 O percurso de uma árvore em pré-ordem resultou na impressão da sequência

A B C F H D L M P N E G I

e o percurso da mesma árvore em ordem resultou em

F C H B D L P M N A I G E

Construa uma árvore que satisfaça esses percursos. Ela é única?

Exercícios

4.13 Prove ou dê um contraexemplo. Uma árvore binária pode ser construída, de forma única, a partir das seguintes informações:

(1) Os percursos em pré-ordem e em ordem.
(2) Os percursos em pré-ordem e pós-ordem.
(3) Os percursos em pré-ordem e em nível.
(4) Os percursos em ordem e em nível.
(5) O percurso em pré-ordem e a informação do número de nós em cada subárvore.
(6) O percurso em pós-ordem e a informação do número de nós em cada subárvore.
(7) O percurso em nível e a informação do número de nós em cada subárvore.

4.14 Descreva três algoritmos PRÉ-ORDEM, PÓS-ORDEM e EM-ORDEM tais que, dada uma árvore binária T, escreva os elementos na ordem em que são visitados em cada um dos percursos.
Sugestão: escreva algoritmos recursivos.

Exercícios

4.15 Escreva um algoritmo que, dada uma árvore binária T, imprima em um percurso em-ordem (e-r-d) os conteúdos das folhas de T.

4.16 Escreva um algoritmo que, dada uma árvore binária T e um valor x, devolva VERDADEIRO se T possui algum nó com valor x, e FALSO caso contrário.

4.17 Escreva uma versão não recursiva do algoritmo PRÉ-ORDEM de percurso na árvore. *Sugestão*: use uma pilha.

4.18 Versões não recursivas dos algoritmos PÓS-ORDEM e EM-ORDEM não são triviais: investigue como eles podem ser implementadas.