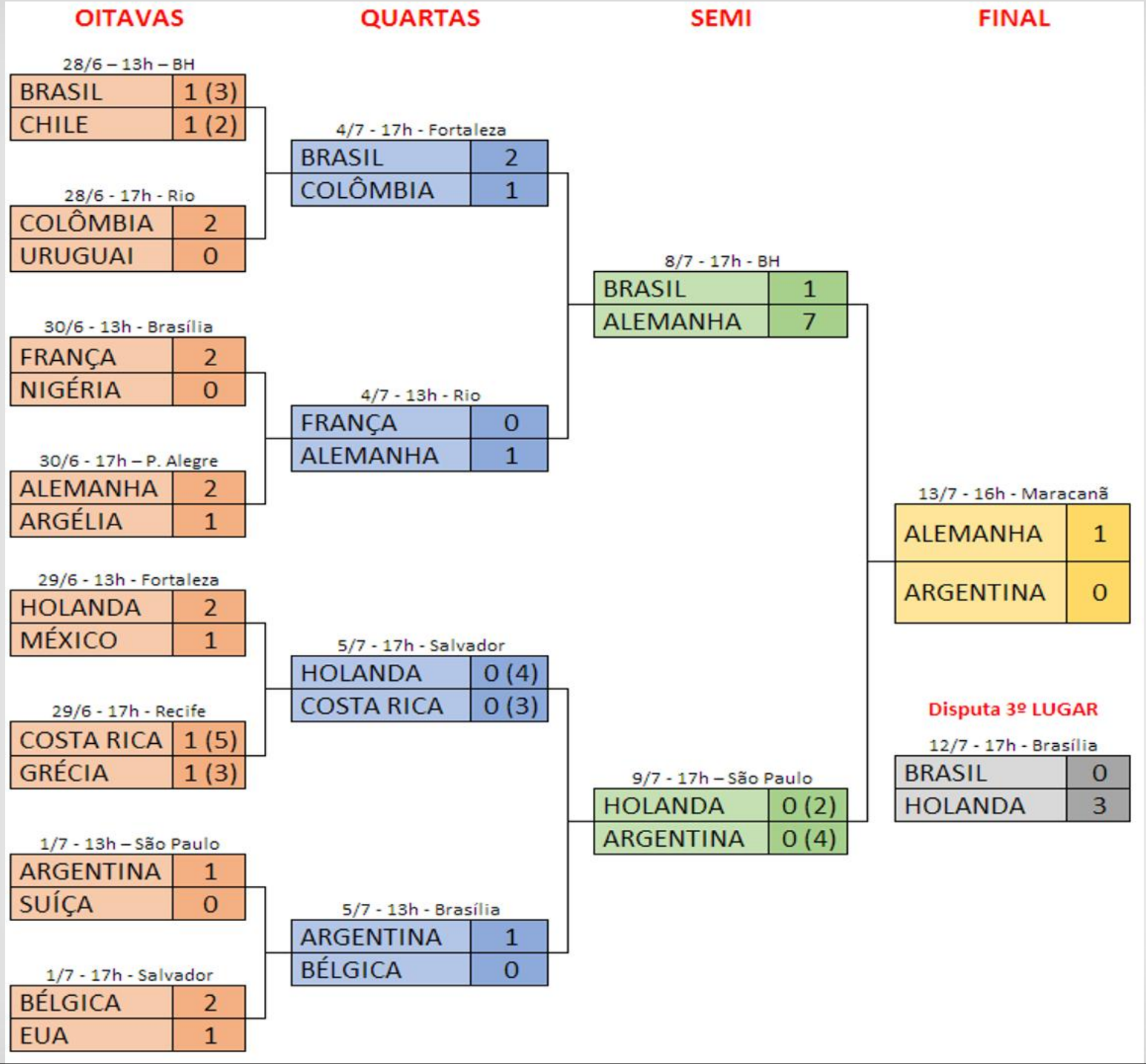
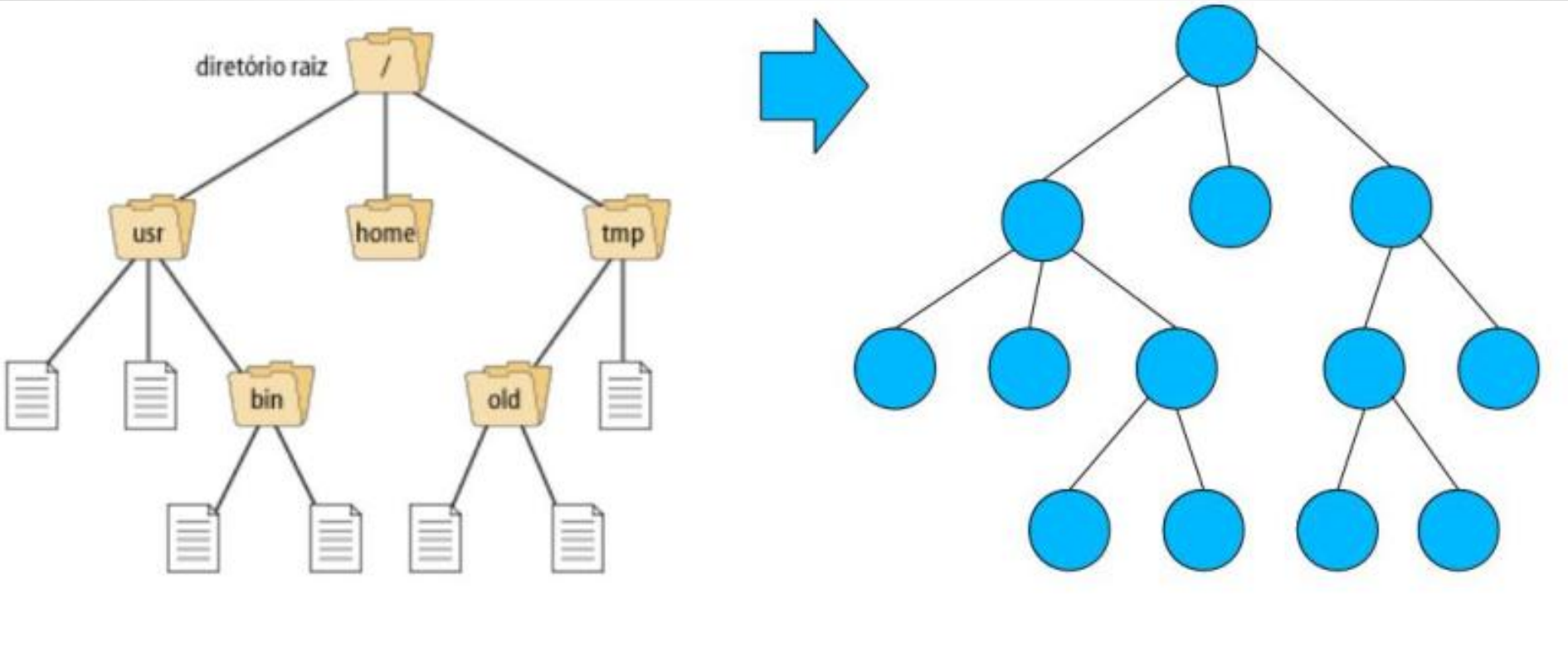
Árvores Binárias

Árvores Binárias

- Diversas aplicações necessitam que se represente um conjunto de objetos e suas relações hierárquicas;

- Várias são as aplicações de árvores:
  - Relações de descendência;
  - Diagrama hierárquico de uma organização;
  - Campeonatos e modalidades esportivas
  - Taxonomia;
  - Busca de Dados armazenados no computador;
  - Representação de espaço de soluções (ex: jogo de xadrez);
  - Modelagem de algoritmos.


OITAVAS
QUARTAS
SEMI
FINAL
28/6 – 13h – BH
BRASIL 1 (3)
CHILE 1 (2)
28/6 - 17h - Rio
COLÔMBIA 2
URUGUAI 0
30/6 - 13h - Brasília
FRANÇA 2
NIGÉRIA 0
30/6 - 17h – P. Alegre
ALEMANHA 2
ARGÉLIA 1
29/6 - 13h - Fortaleza
HOLANDA 2
MÉXICO 1
29/6 - 17h - Recife
COSTA RICA 1 (5)
GRÉCIA 1 (3)
1/7 - 13h – São Paulo
ARGENTINA 1
SUÍÇA 0
1/7 - 17h - Salvador
BÉLGICA 2
EUA 1
4/7 - 17h - Fortaleza
BRASIL 2
COLÔMBIA 1
4/7 - 13h - Rio
FRANÇA 0
ALEMANHA 1
5/7 - 17h - Salvador
HOLANDA 0 (4)
COSTA RICA 0 (3)
5/7 - 13h - Brasília
ARGENTINA 1
BÉLGICA 0
8/7 - 17h - BH
BRASIL 1
ALEMANHA 7
9/7 - 17h – São Paulo
HOLANDA 0 (2)
ARGENTINA 0 (4)
13/7 - 16h - Maracanã
ALEMANHA 1
ARGENTINA 0
Disputa 3º LUGAR
12/7 - 17h - Brasília
BRASIL 0
HOLANDA 3


diretório raiz
/
usr
home
tmp
bin
old

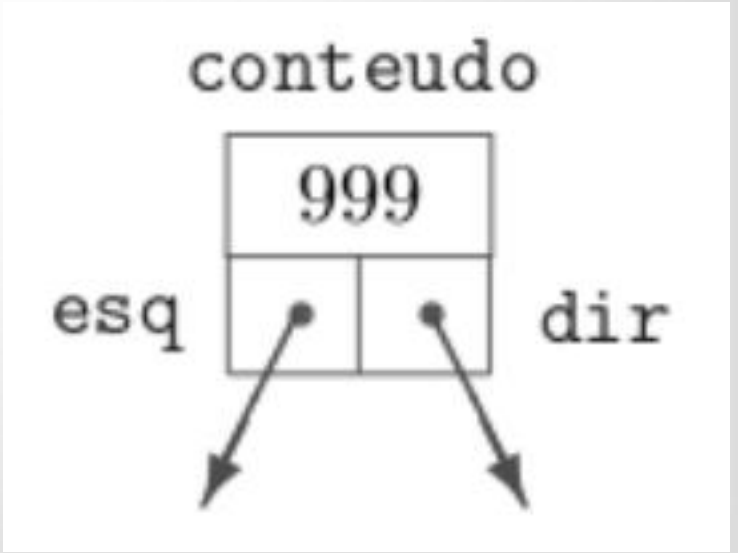
Árvore binária

- Uma árvore binária é uma árvore onde cada nó só pode ter no máximo 2 filhos.
- Uma árvore binária (= *binary tree*) é um conjunto de registros.
- Os registros serão chamados *nós*.
- Cada nó tem um endereço.
- Exemplo: cada nó tem três campos: um número inteiro e dois ponteiros para nós.

Árvore binária

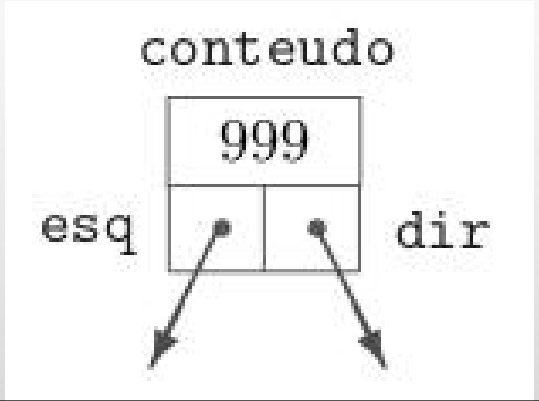
- Os nós podem, então, ser definidos assim:

```
typedef struct reg {
   int conteudo; // conteúdo
   noh *esq;
   noh *dir;
} noh; // nó
```

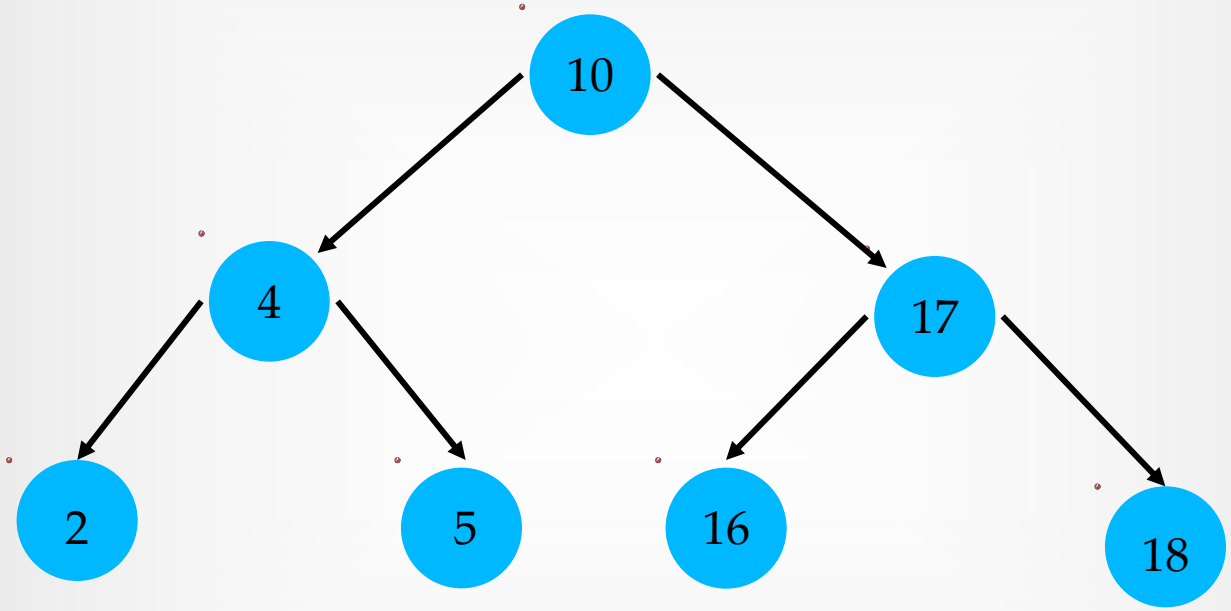


Árvore binária

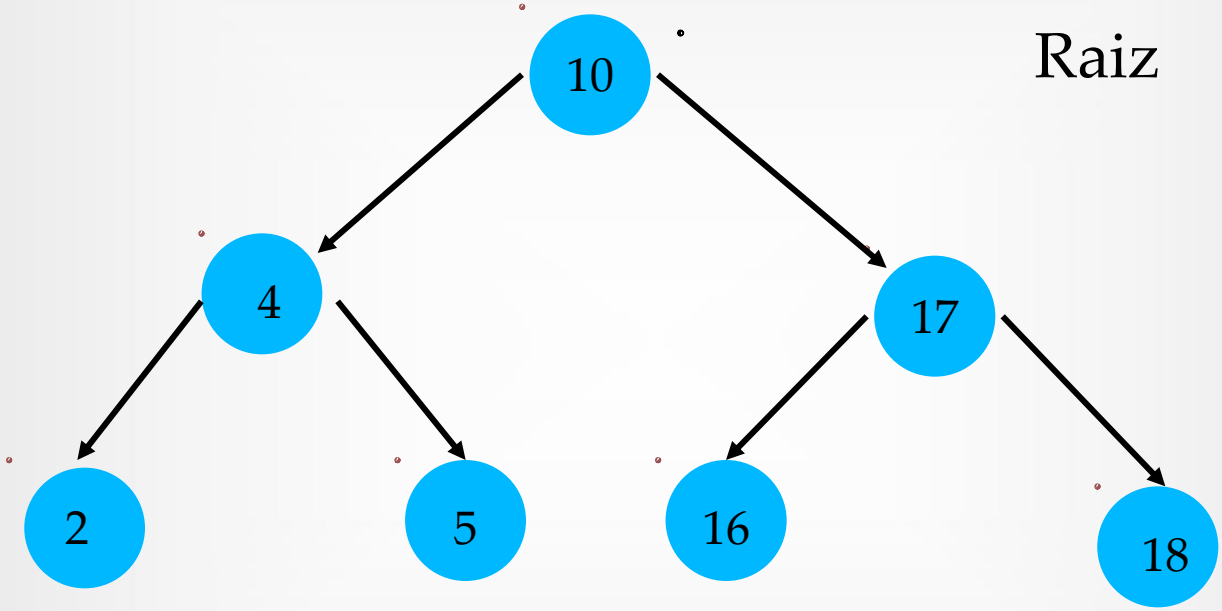
- O campo conteúdo é a carga útil do nó;
- O campo esq de cada nó contém NULL ou o endereço de outro nó.
- A mesma hipótese vale para o campo dir.



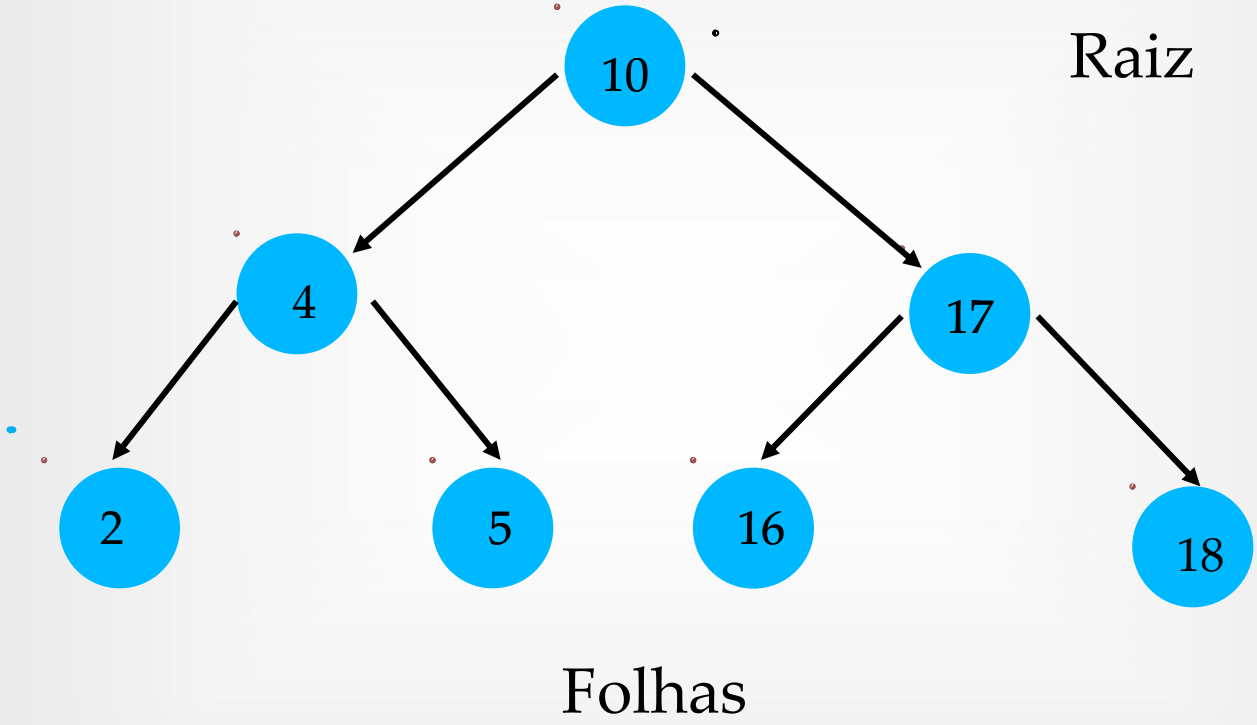
Árvores binárias



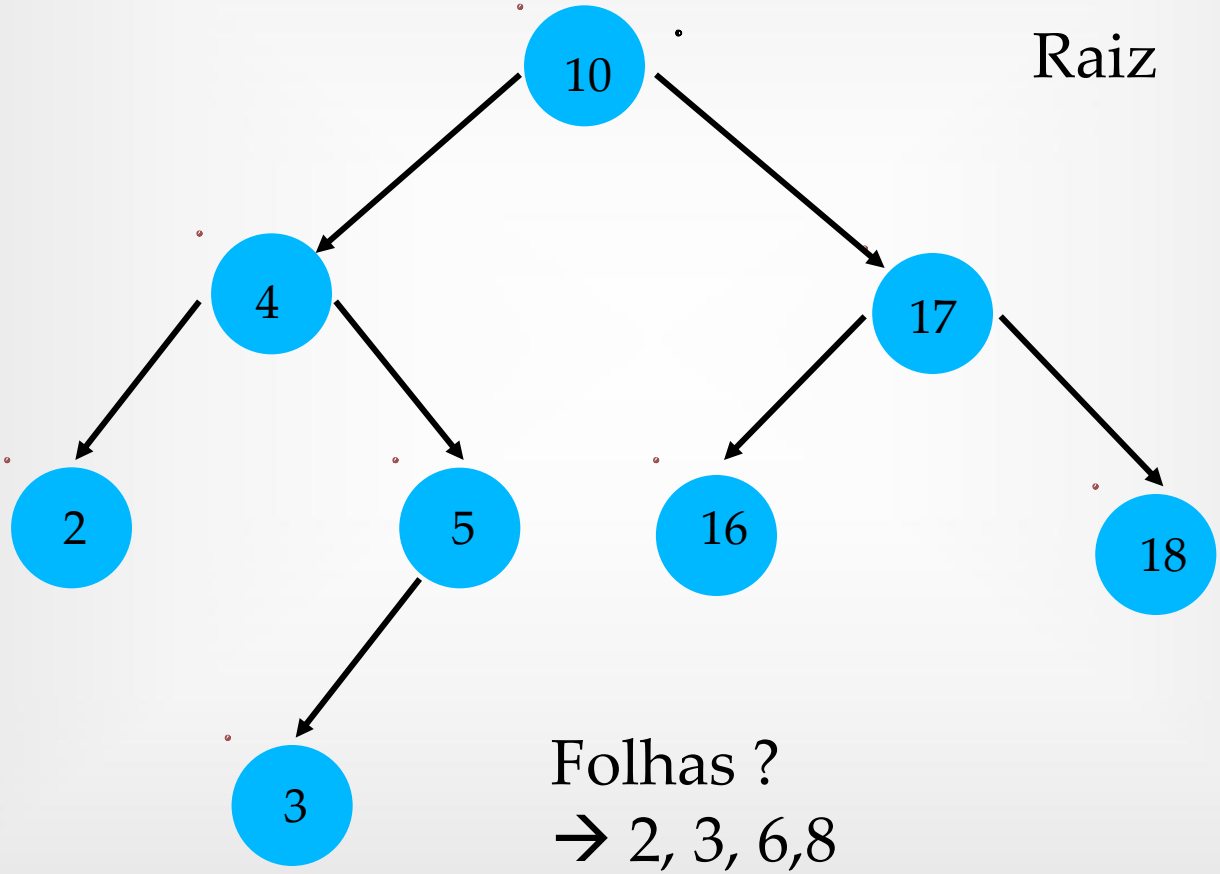
Árvores binárias



Árvores binárias



Árvores binárias

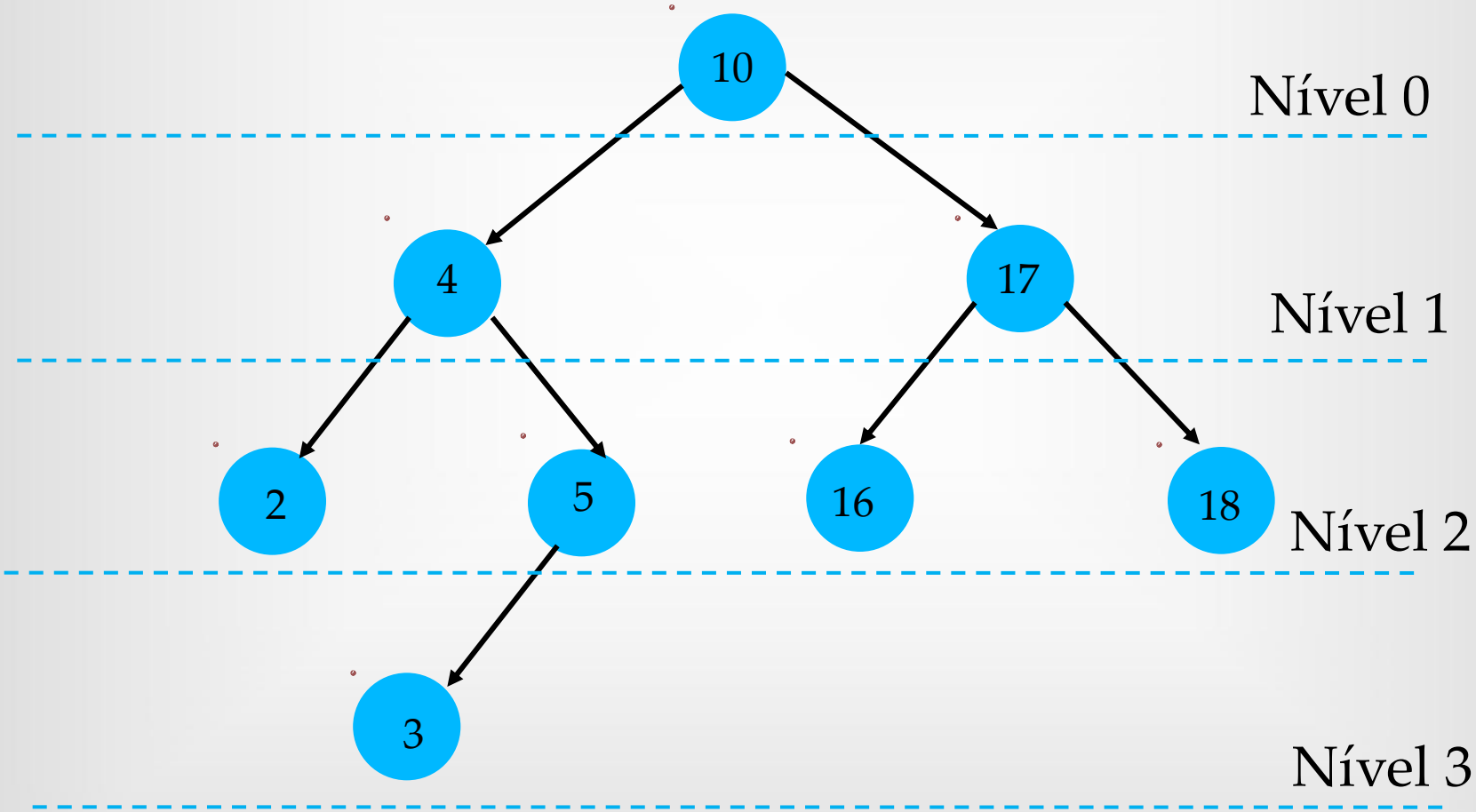


Folhas ?

→ 2, 3, 6,8

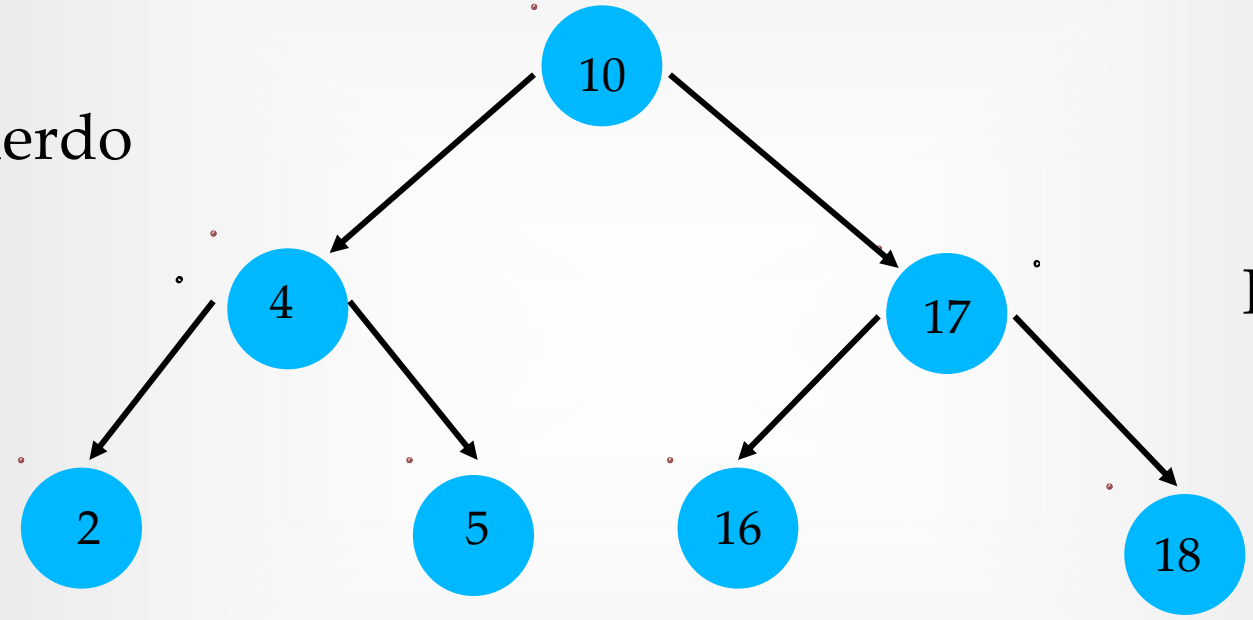
Árvores binárias

Profundidade



Árvores binárias

Filho esquerdo



Filho direito

Exercícios

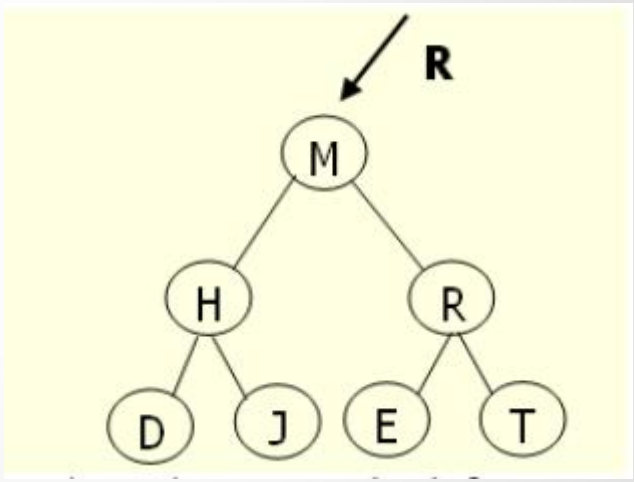
Dada a árvore ao lado, indique:

a) os nós folha

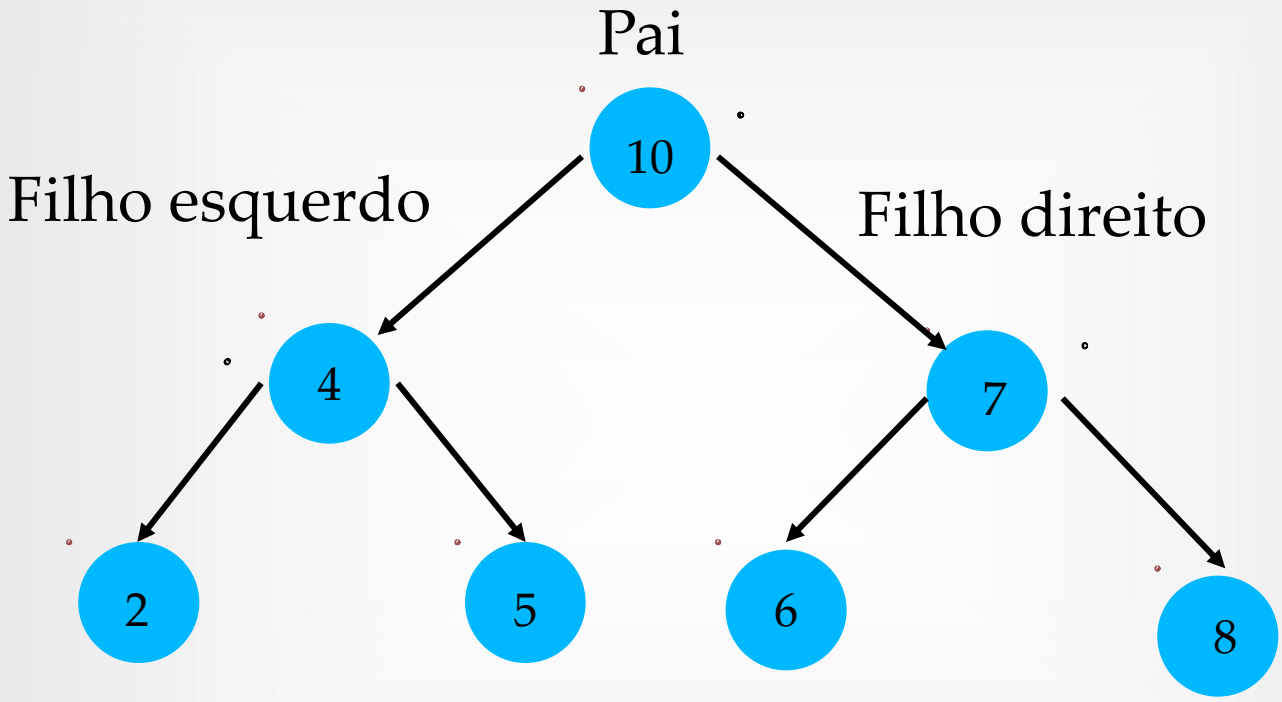
b) o grau da árvore

c) o altura da árvore

d) os descendentes do nó H

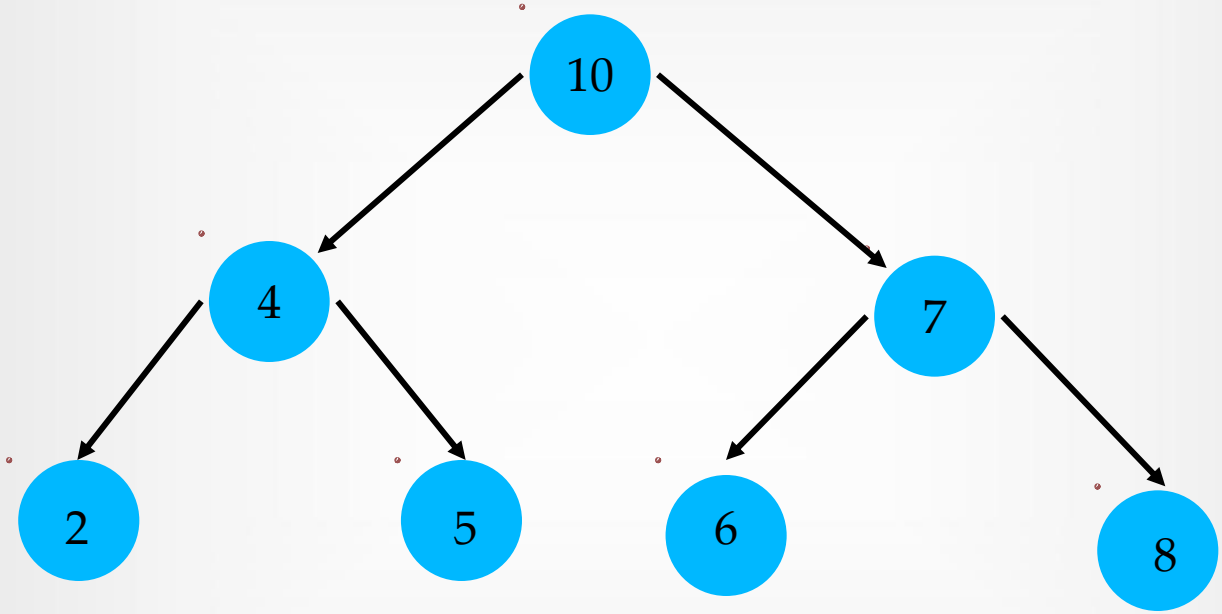


Árvores binárias



Filho dir de 10? → 17
Filho esq de 10? → 4
Pai do 4? → 10
Pai do 7 → 10

Árvores binárias



Filho dir de 17? → ?
Filho esq de 5? → ?
Pai do 2? → ?
Pai do 10 → ?

Árvores Binárias

- **Nó:** são todos os itens guardados na árvore;
- **Raiz:** é o item do topo da árvore;
- **Filho:** são os itens logo abaixo da raiz;
- **Folha:** é um nó que não tem filho, é o último item da árvore.

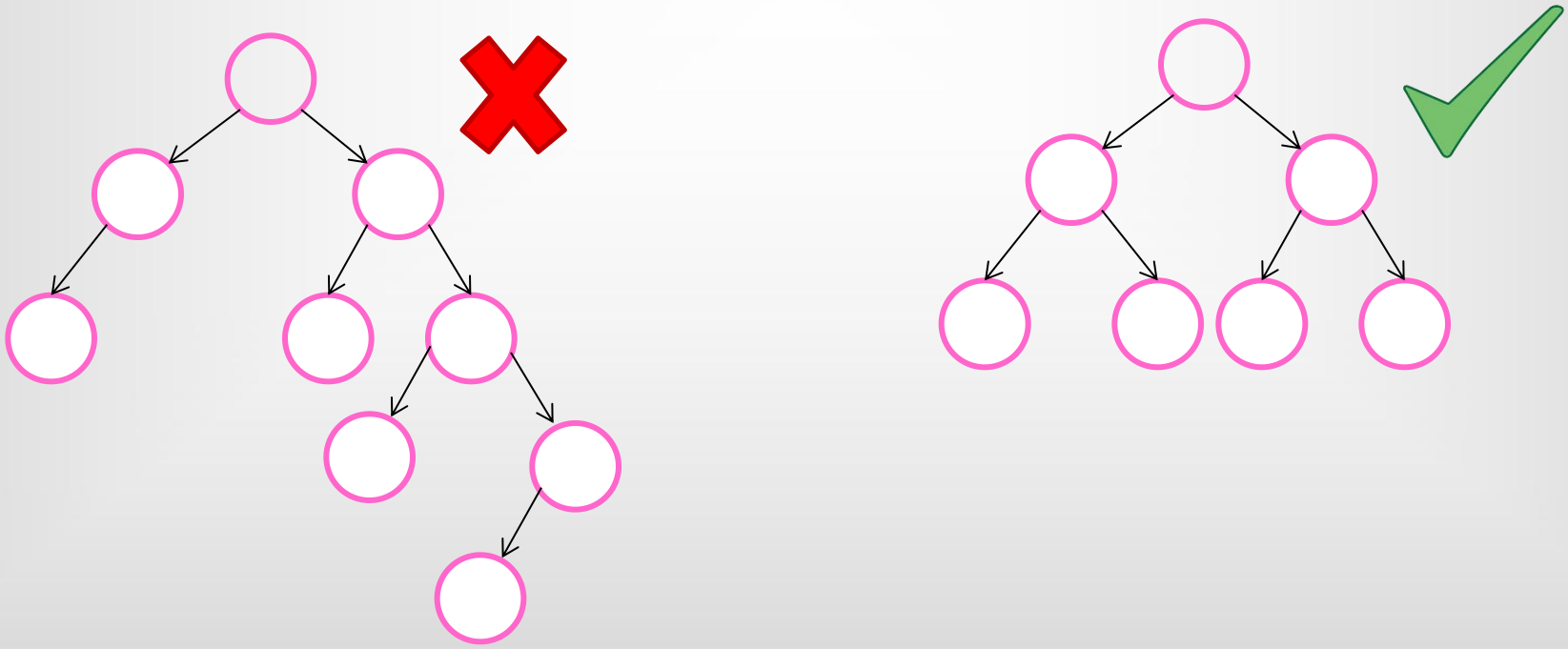
Exercícios

1. Num diagrama convencional de árvore (raiz no topo), se o nó X tem nível maior que o nó Y, então X aparece abaixo de Y no diagrama?

2. Se o nó A tem 3 irmãos e B é pai de A, qual o grau de B?

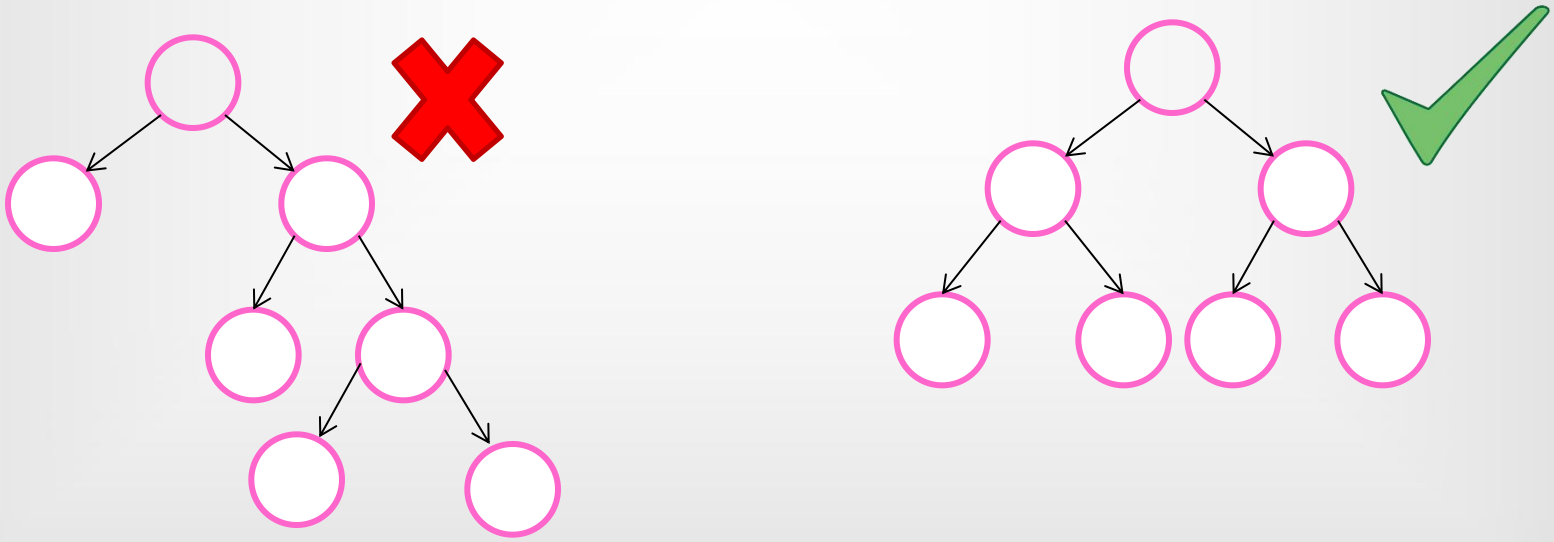
Terminologia

Árvore estritamente binária: cada nó tem grau 0 ou 2, ou seja, todo nó tem 0 ou 2 filhos.



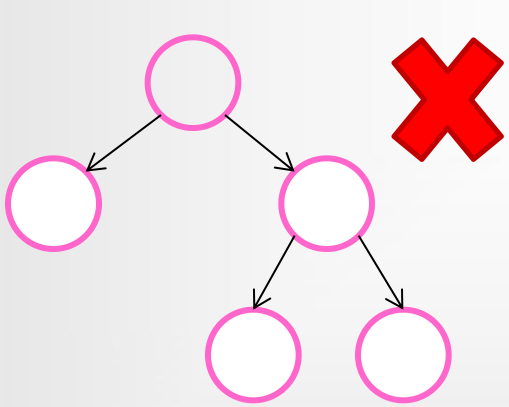
Terminologia

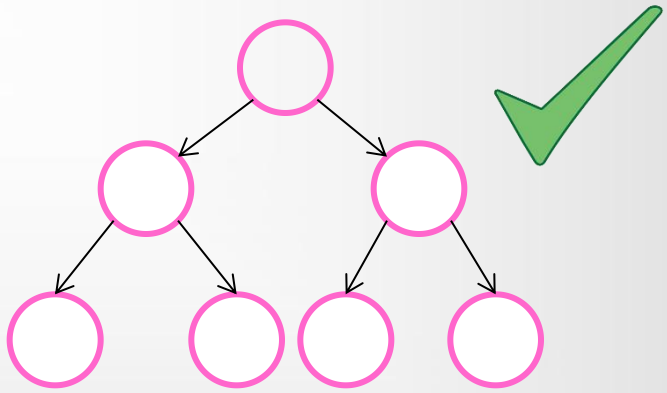
Árvore binária completa: é uma árvore estritamente binária na qual todo nó que apresente alguma subárvore vazia está localizada no último ou no penúltimo nível da árvore.



Terminologia

Árvore binária cheia: quando todos os nós internos tem grau 2 e todas as folhas estão no mesmo nível.





Árvores e subárvores

Suponha que x é um nó.

Um *descendente* de x é qualquer nó que possa ser alcançado pela iteração das instruções:

```
x = x->esq   e   x = x->dir
```

Árvores e subárvores

Um nó `x` juntamente com todos os seus descendentes é uma *árvore binária*.

Dizemos que `x` é a *raiz* (*root*) da árvore.

Se `x` tiver um `pai`, essa árvore é uma `subárvore` de alguma árvore maior.

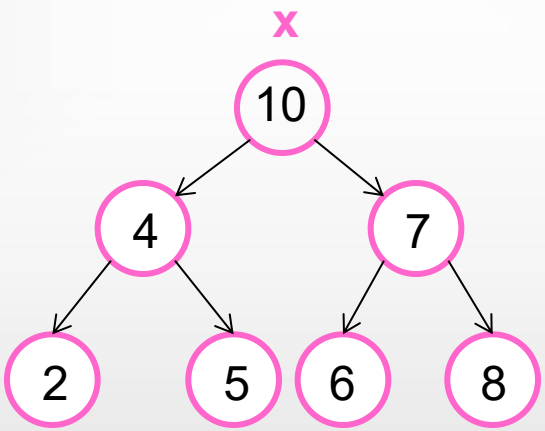
Se `x` é `NULL`, a árvore é `vazia`.

Árvores e subárvores

Para qualquer nó x:

o nó x->esq é a raiz da *subárvore esquerda* de x

x->dir é a raiz da *subárvore direita* de x.

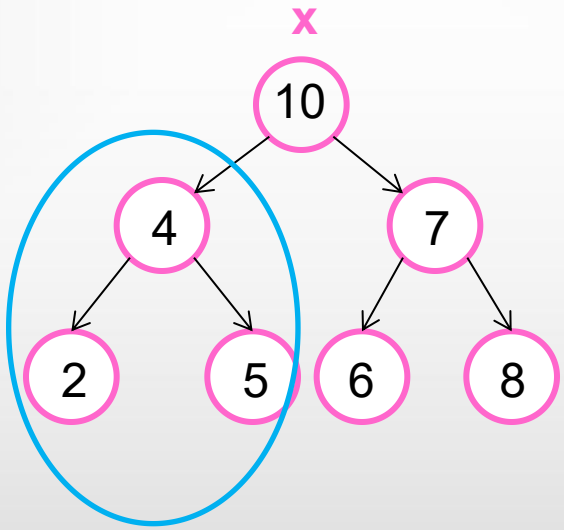


Árvores e subárvores

Para qualquer nó x:

o nó x->esq é a raiz da *subárvore esquerda* de x

x->dir é a raiz da *subárvore direita* de x.



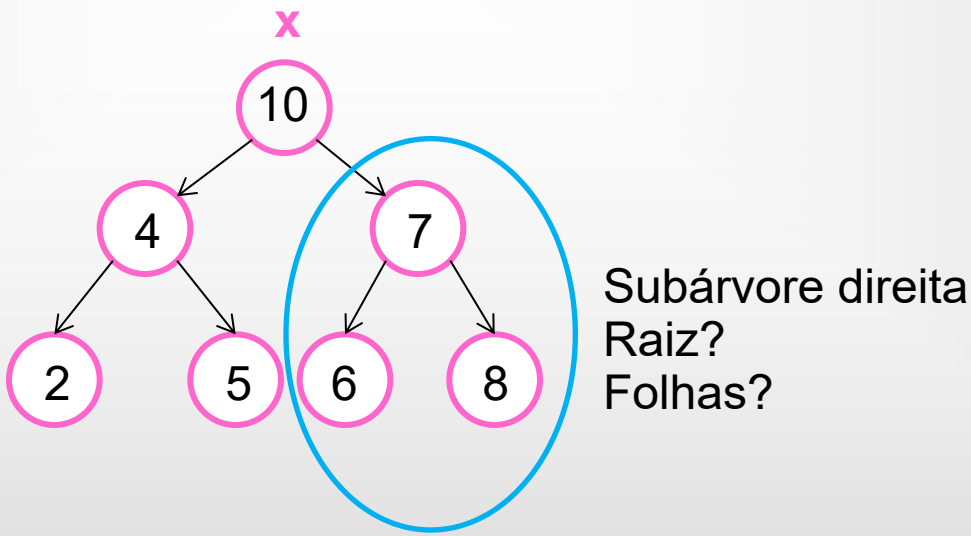
Subárvore esquerda
Raiz?
Folhas?

Árvores e subárvores

Para qualquer nó x:

o nó x->esq é a raiz da *subárvore esquerda* de x

x->dir é a raiz da *subárvore direita* de x.



Percorrer árvores

Os elementos de uma árvore (binária) podem ser enumerados por quatro ordens diferentes. As três primeiras definem-se recursivamente:

Pré-ordem: Primeiro a raiz, depois a sub-árvore esquerda, e finalmente a sub-árvore direita.

Em-ordem: Primeiro a sub-árvore esquerda, depois a raiz, e finalmente a sub-árvore direita.

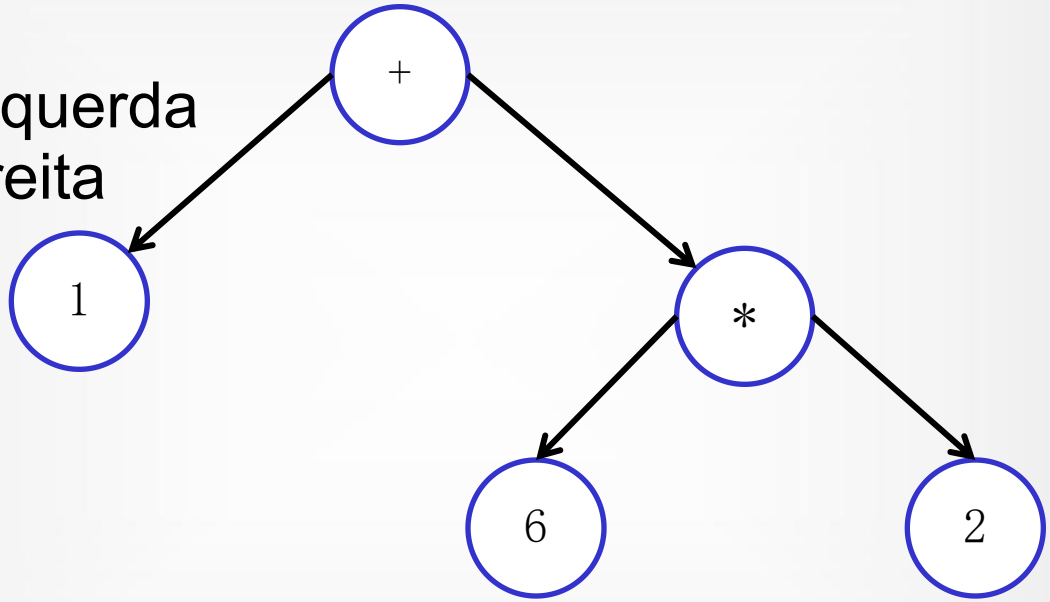
Percorrer árvores

Pós-ordem: Primeiro a sub-árvore esquerda, depois a sub-árvore direita, e finalmente a raiz

Percorrer árvores

Pré-ordem

→ Raiz
→ Subárvore esquerda
→ Subárvore direita

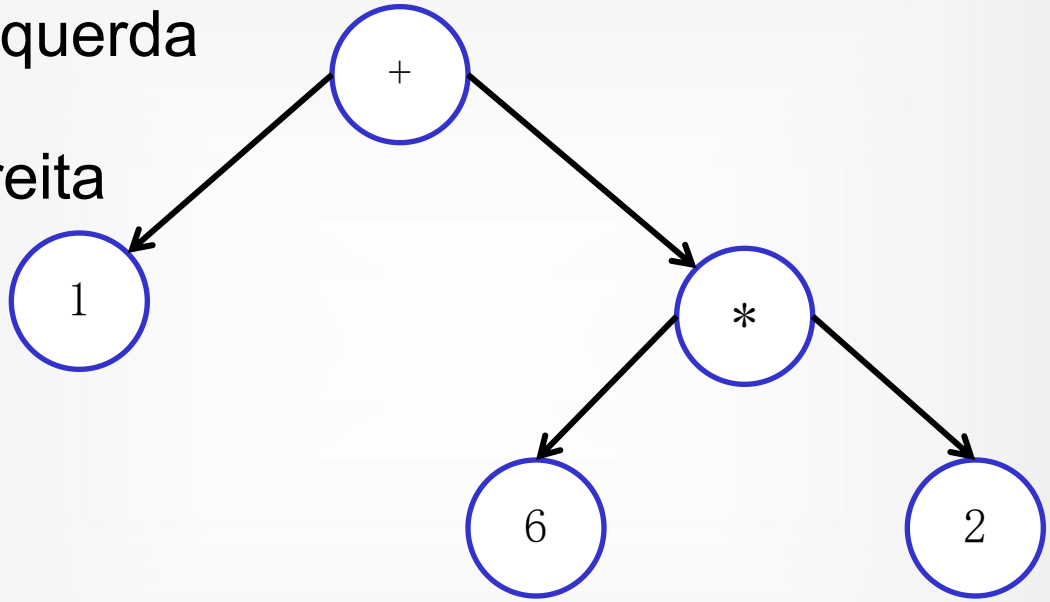


+1*62

Percorrer árvores

Em-ordem

→ Subárvore esquerda
→ Raiz
→ Subárvore direita

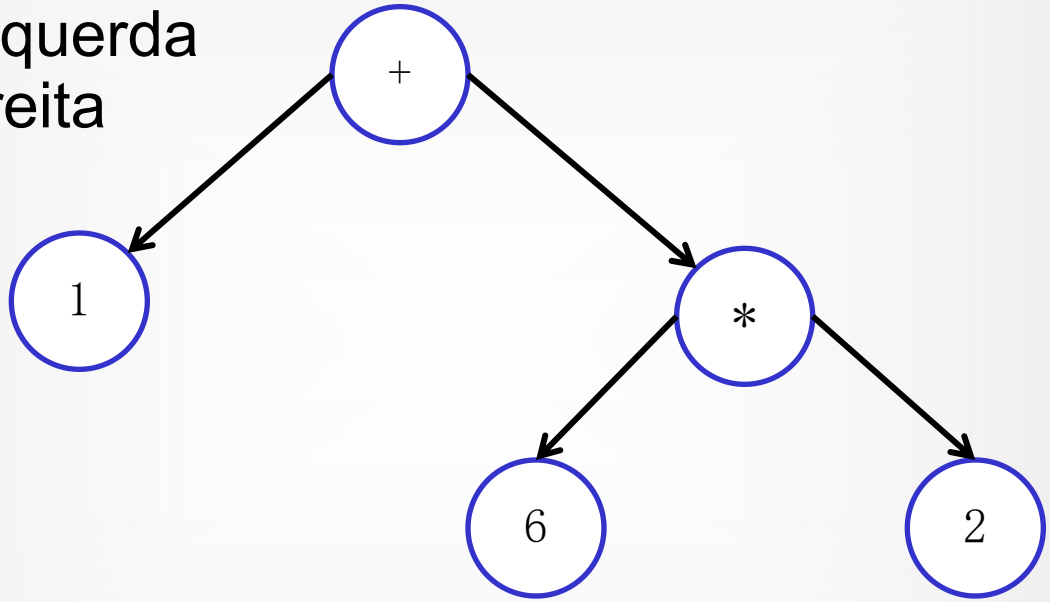


1+6*2

Percorrer árvores

Pós-ordem

→ Subárvore esquerda
→ Subárvore direita
→ Raiz



162*+

Exercício

1) Desenha cada uma das árvores abaixo e iga, e liste seus nós em (i) pré-ordem, (ii) in-ordem e (iii) pós-ordem:

a) (1 (2 (4) (5)) (3 (6) (7)));

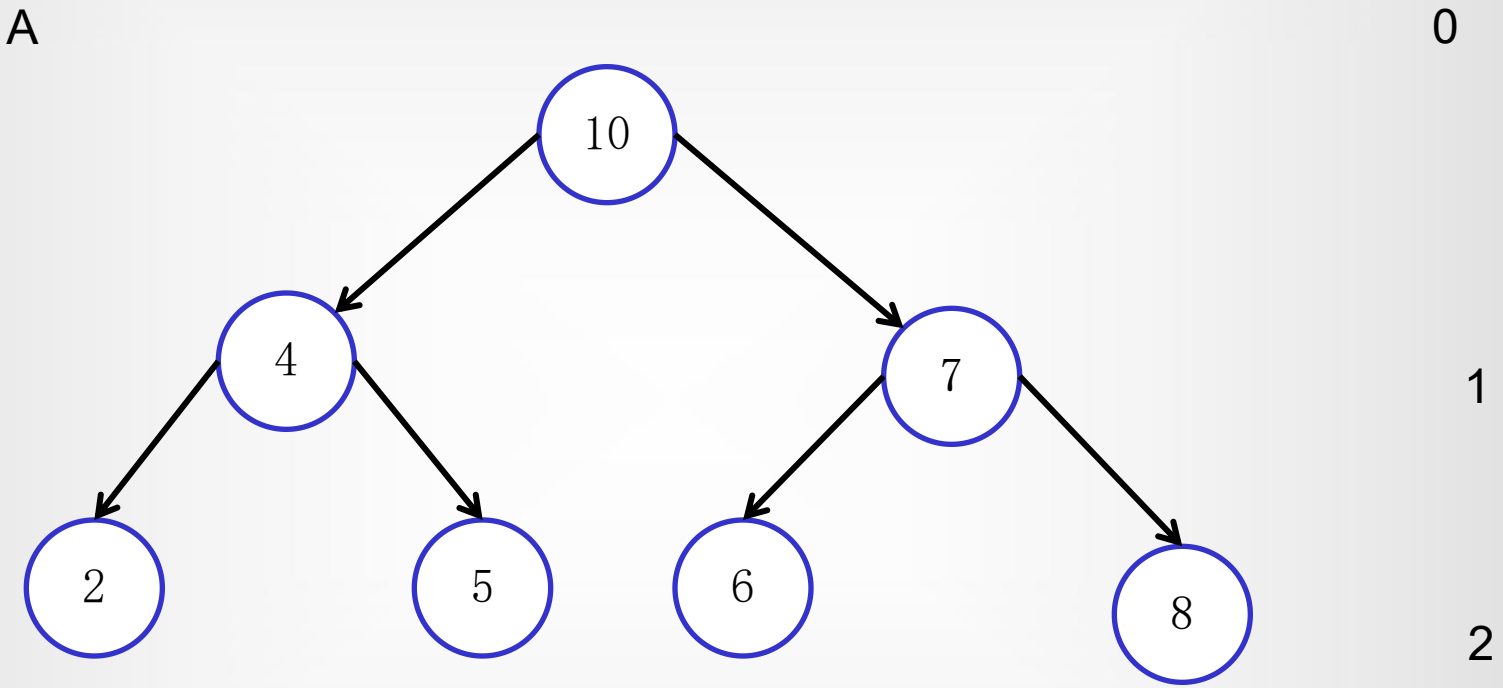
b) (A (B (D (F)) (E)) (C (G (H)))).

Altura e profundidade

A *altura de um nó* x em uma árvore binária é a distância entre x e o seu descendente mais afastado.

A altura de x é o número de passos do mais longo caminho que leva de x até uma folha.

Altura e profundidade



Altura(A) = ?

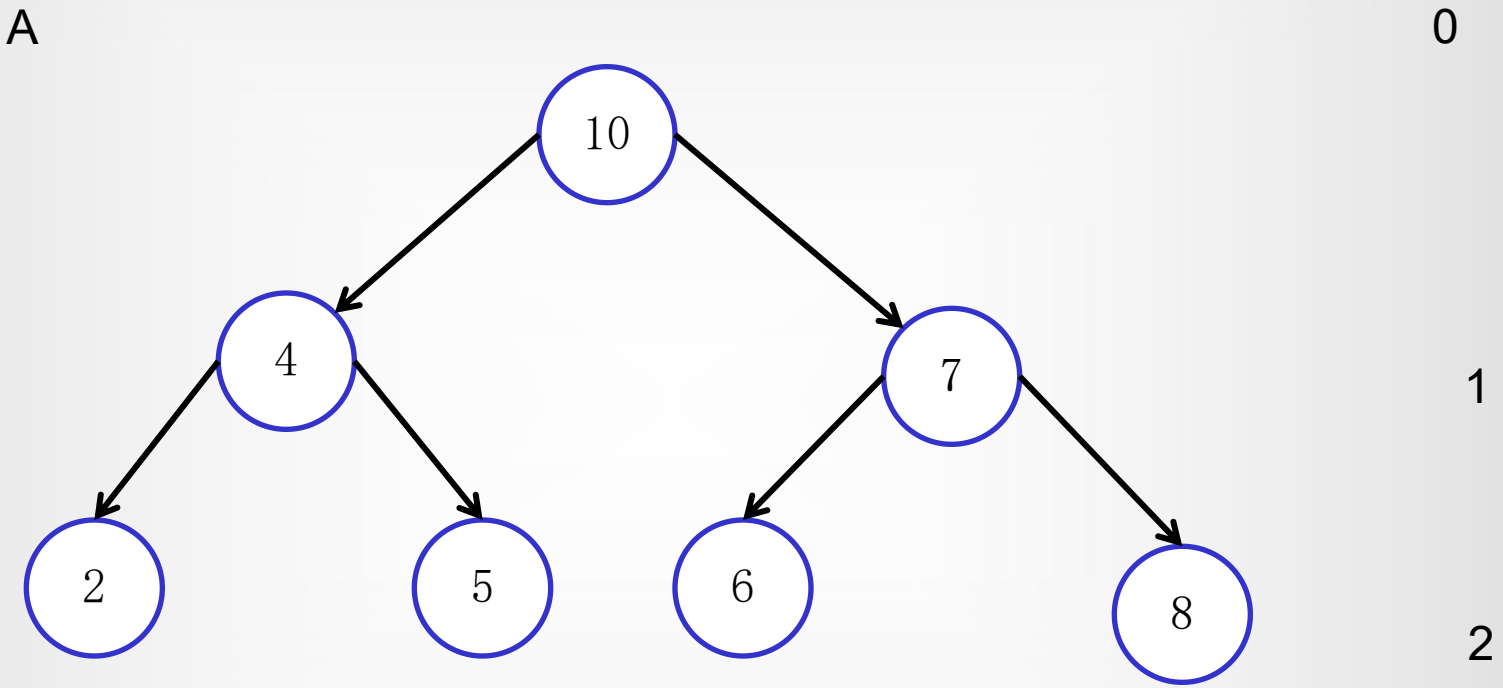
A *altura de uma árvore* é a altura da raiz da árvore.

Altura e profundidade

Uma árvore binária é *balanceada* se, em cada um de seus nós, as subárvores esquerda e direita tiverem aproximadamente a mesma altura.

A *profundidade* (*depth*) de um nó `s` em uma árvore binária com raiz `r` é a distância de `r` a `s`. Mais precisamente, a profundidade de `s` é o comprimento do (único) caminho que vai de `r` até `s`.

Altura e profundidade



Profundidade(2) = ?

Uma árvore é balanceada se todas as suas folhas têm aproximadamente a mesma profundidade.

Exercícios

1) Desenhe árvores binárias de pesquisa de alturas 2, 3, 4, 5 e 6 dado o conjunto de chaves {1, 4, 5, 10, 16, 17, 21}.

2) Monte uma árvore binária de pesquisa com a menor altura possível com o seguinte conjunto de dados:

{ 1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31, 34, 37, 40, 43 }.

TDA de uma árvore binária

Vazia: () → A

Noh: E // Elemento

AE // subárvore da esquerda

AD // subárvore da direita

Exemplo:

A = noh(10, noh(4, noh(2,vazia,vazia),noh(5, vazia, vazia)),noh(7, noh(6, vazia, vazia), noh(8,vazia,vazia)))

TDA de uma árvore binária

Exemplo:

```
A = noh(10, noh(4,
        noh(2,vazia,vazia),noh(5, vazia, vazia)
        ),
         noh(7,
          noh(6, vazia, vazia), noh(8,vazia,vazia)
         ))
```

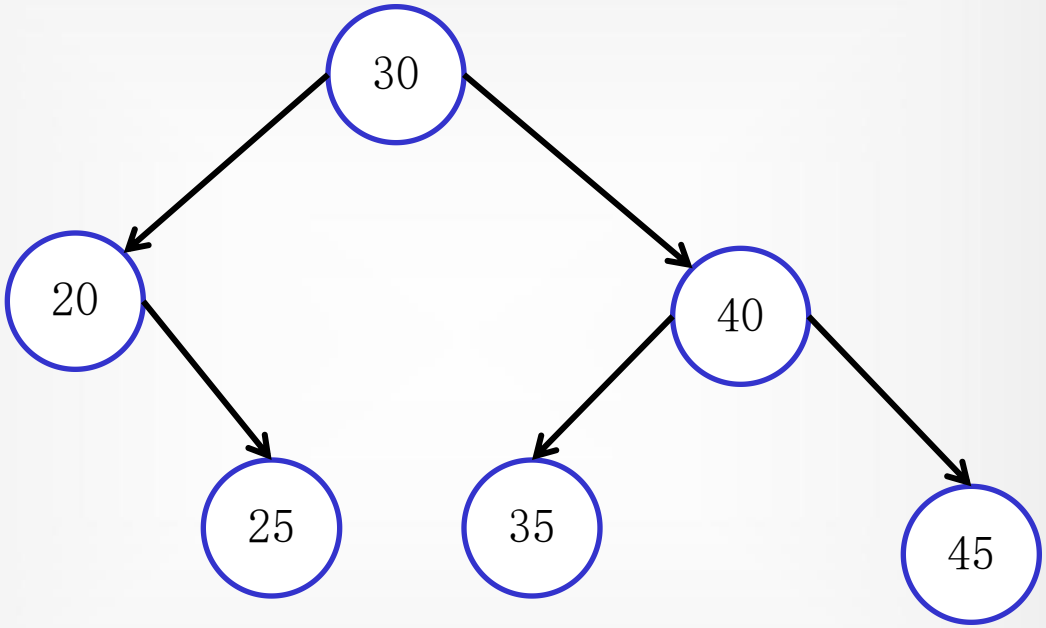
TDA de uma árvore binária

Exemplo:

```
A = noh(5, vazia, noh(8, noh(4, vazia, vazia), vazia)
```

TDA de uma árvore binária

Exemplo:



Fazer o inverso...

Exercício

Desenhe uma árvore binária que com 17 nós que tenha a menor altura possível. Repita com a maior altura possível.

Escreva uma função iterativa para calcular a altura de uma árvore binária.