ÁRVORES E ÁRVORES BINÁRIAS

Vanessa Braganholo

Estruturas de Dados e Seus Algoritmos

ÁRVORES

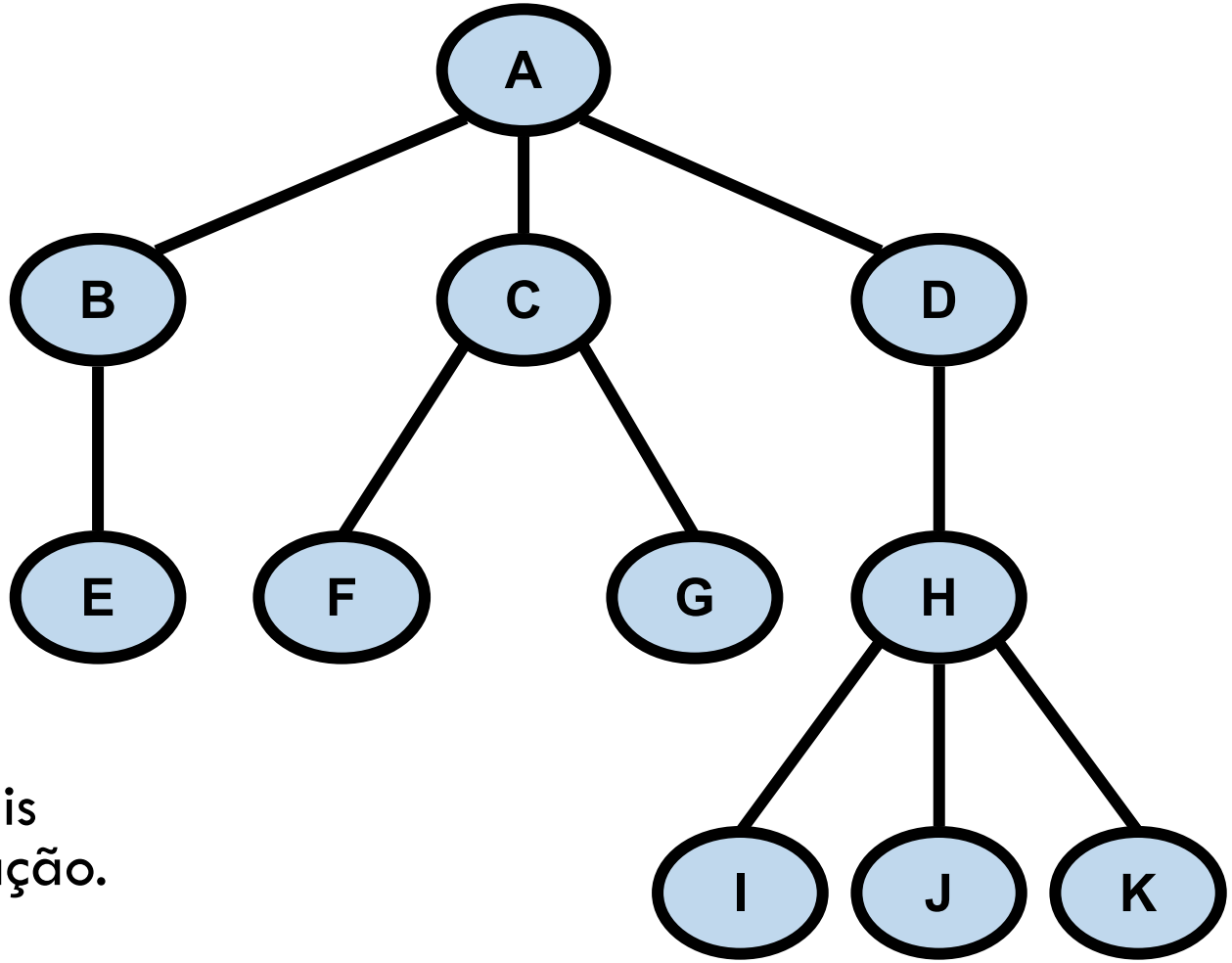
Árvores

Árvores Binárias

ÁRVORES

Fonte de consulta: Szwarcfiter, J.; Markezon, L. Estruturas de Dados e seus Algoritmos, 3a. ed. LTC. Cap. 3

ÁRVORES

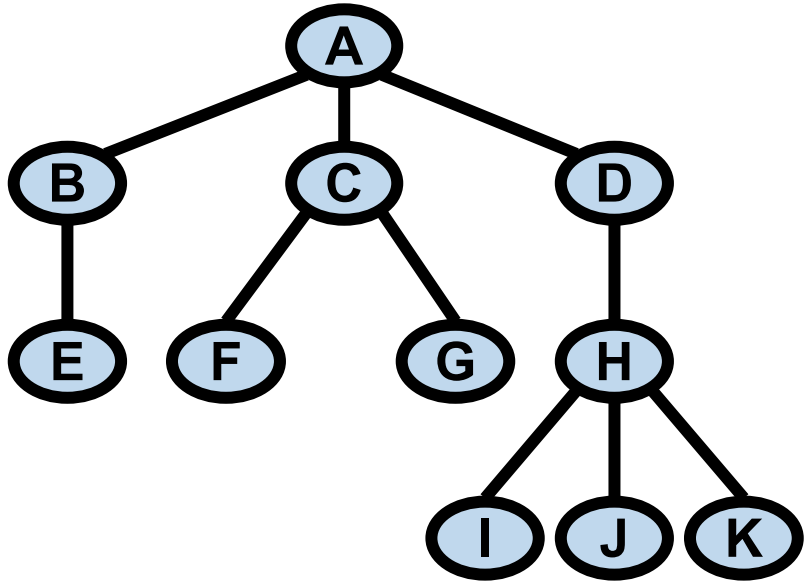


Constituem uma das estruturas mais importantes da área de computação.

São usadas em diversas aplicações

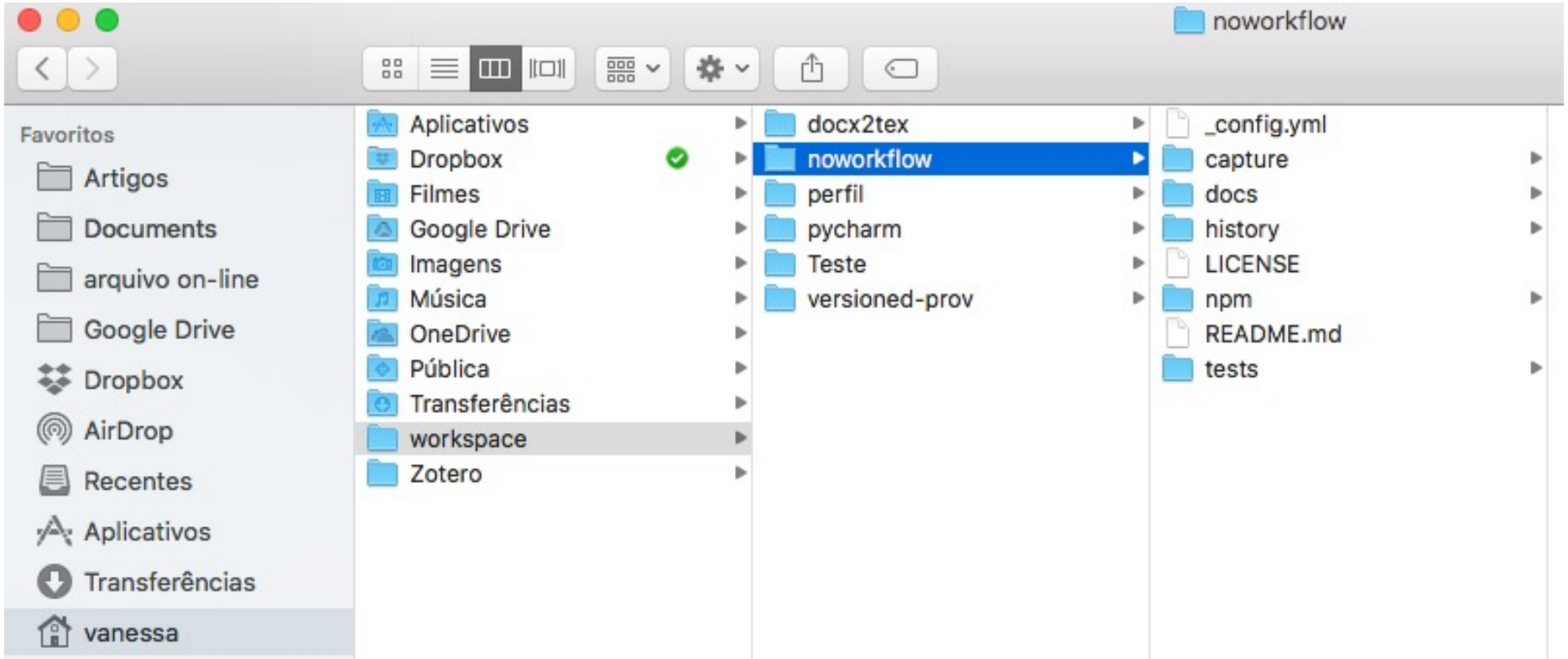
ÁRVORES

São formadas por um conjunto de nós que possuem relacionamento de **Hierarquia** ou **Subordinação**



EXEMPLOS DE APLICAÇÃO

Sistema de Arquivos



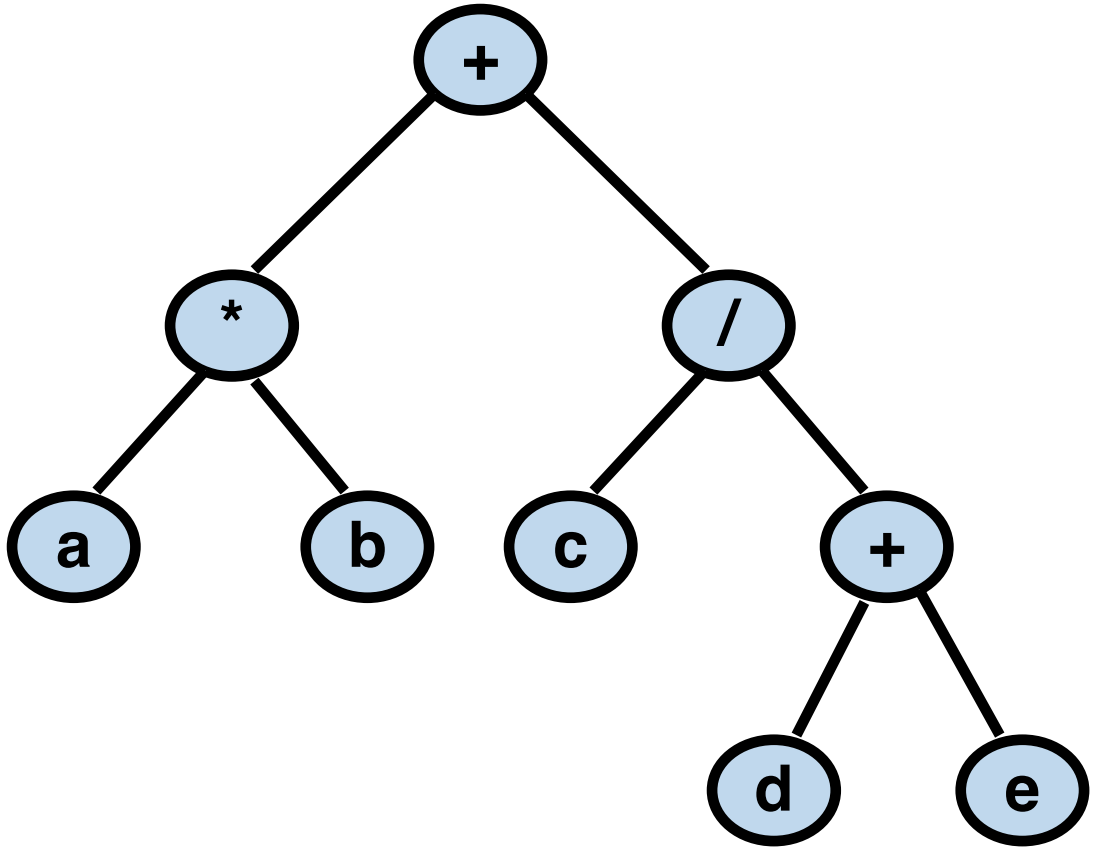
EXEMPLOS DE APLICAÇÃO

Árvore de decisão para jogar tênis

EXEMPLOS DE APLICAÇÕES

Árvore de derivação

- Usada pelos compiladores



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

DEFINIÇÃO FORMAL

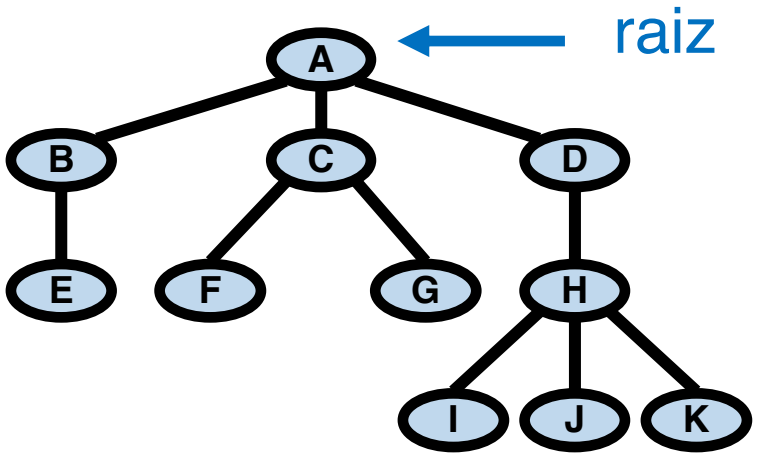
Conjunto finito de zero ou mais nós (nodos ou vértices), tal que:

Se número de nós é maior do que zero

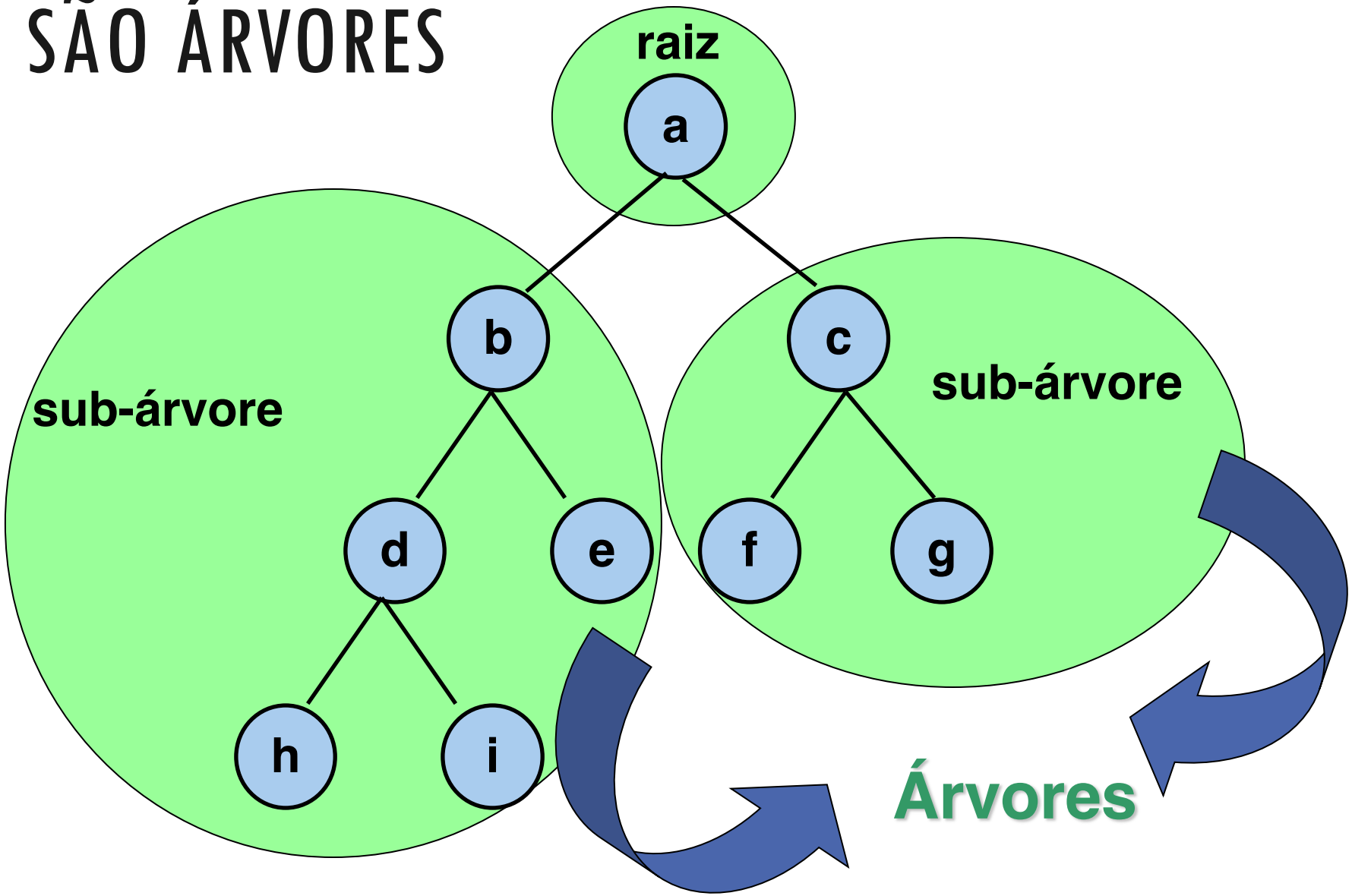
- existe um nó denominado raiz da árvore
- os demais nós formam $m \geq 0$ conjuntos disjuntos S_1, S_2, ..., S_m, onde cada um destes é uma árvore (S_i são denominadas sub-árvores)

Se número de nós é igual a zero

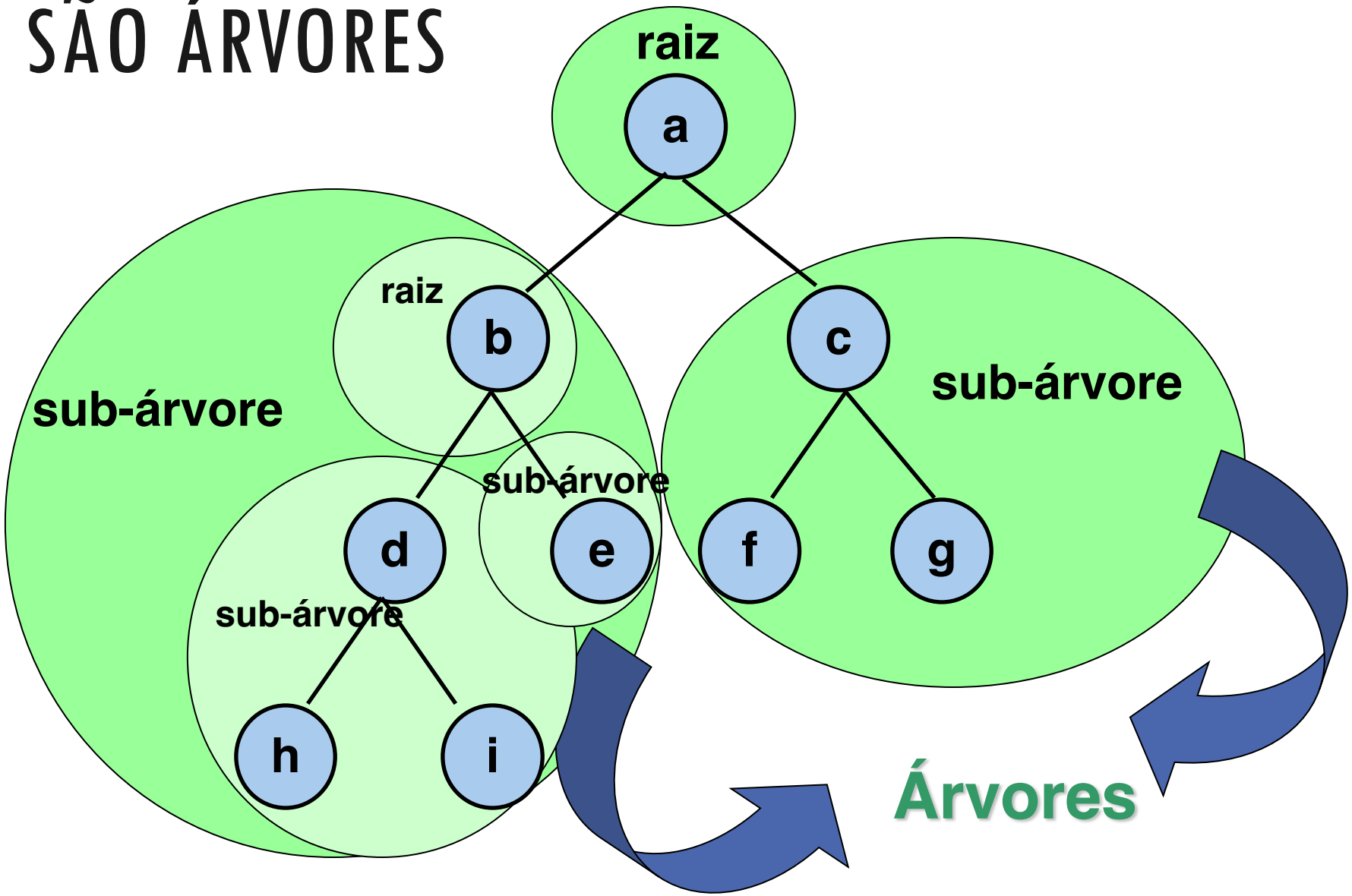
- árvore vazia



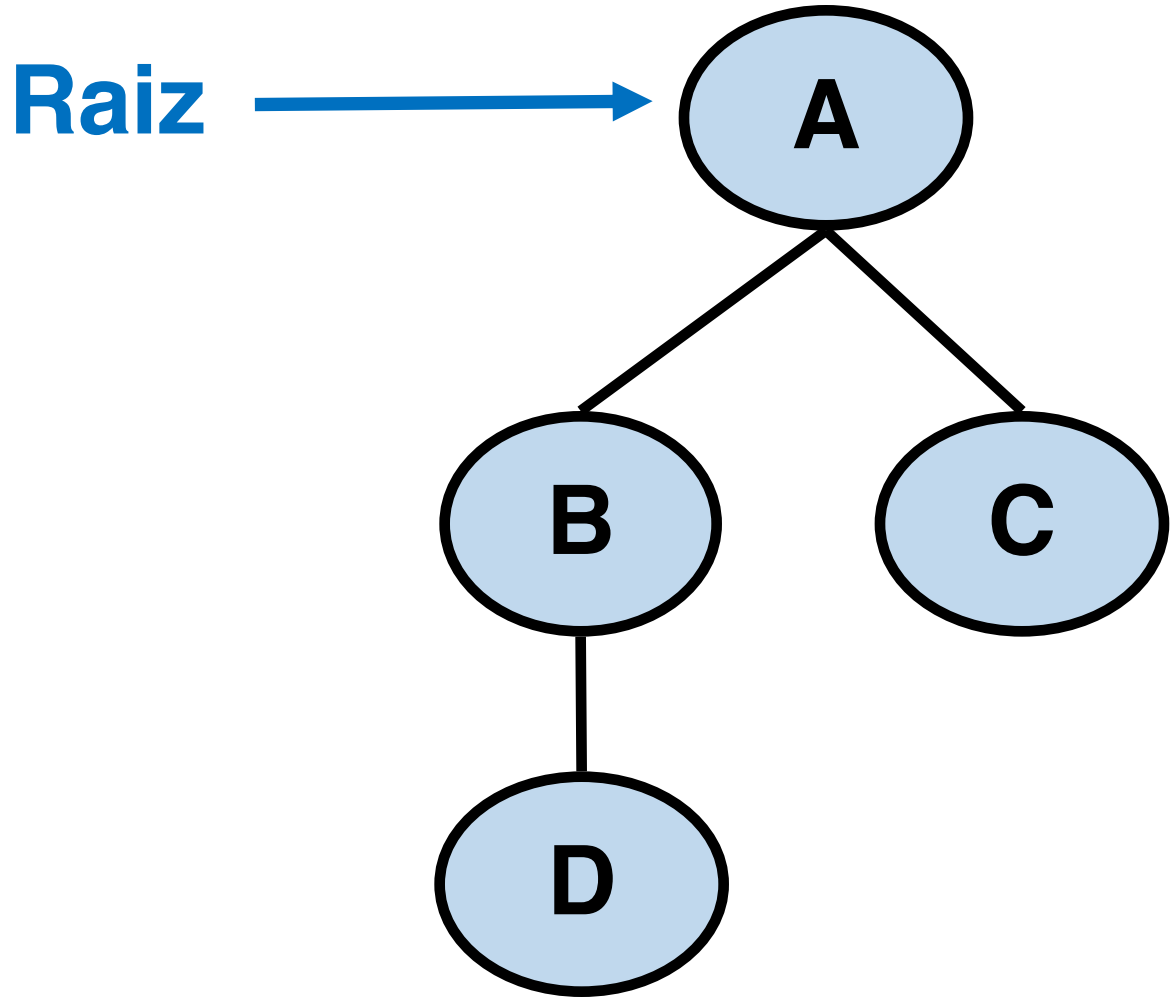
SUBÁRVORES SÃO ÁRVORES



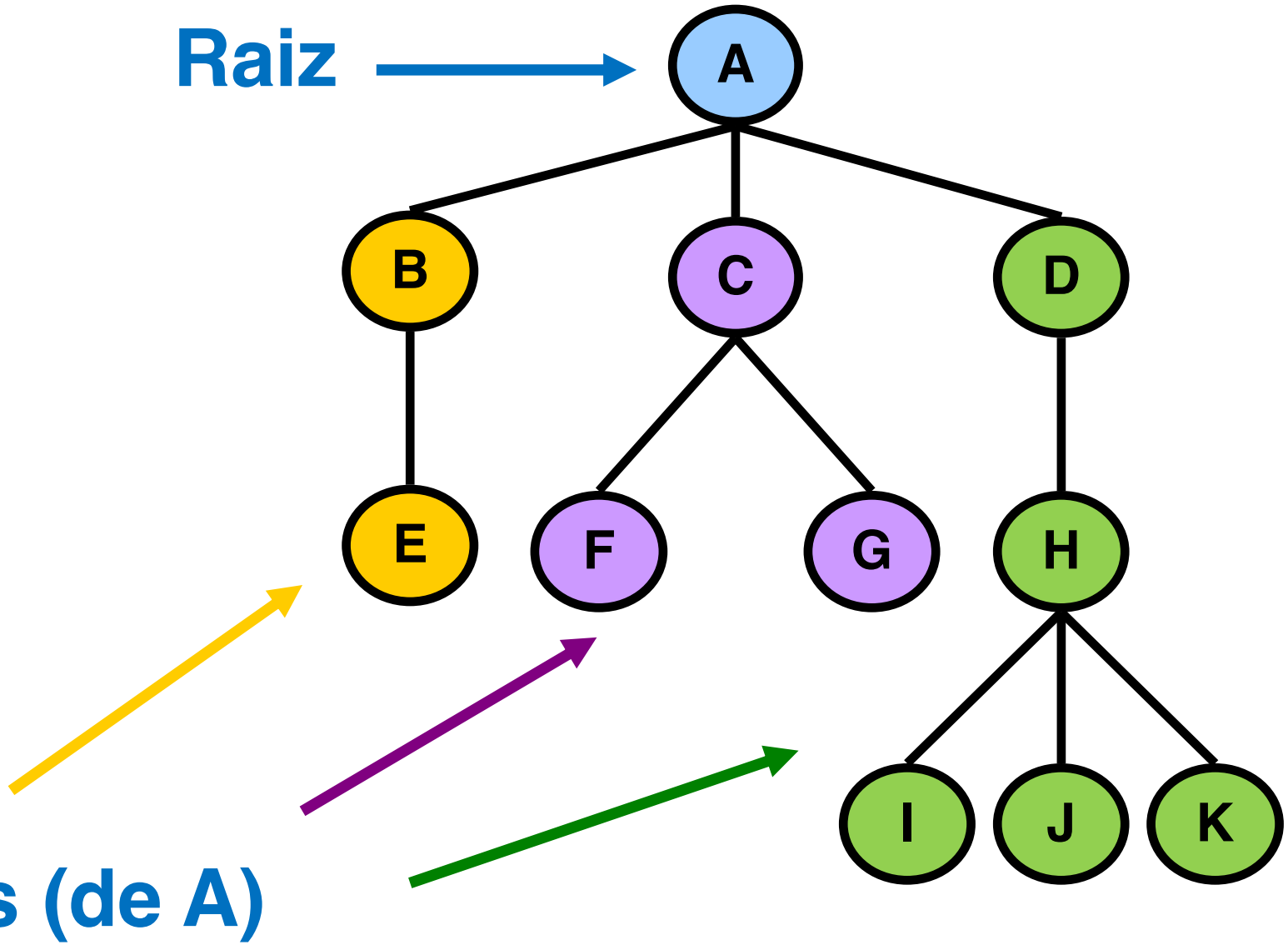
SUBÁRVORES SÃO ÁRVORES



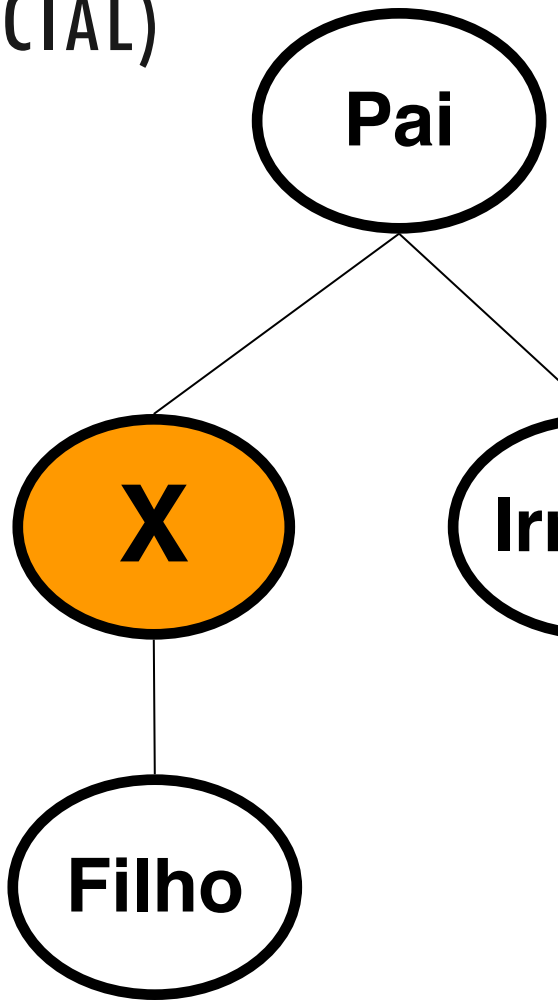
TERMINOLOGIA



TERMINOLOGIA



TERMINOLOGIA
(USANDO O NÓ X COMO REFERENCIAL)



Pai
= mãe
= ascendente
= antecessor

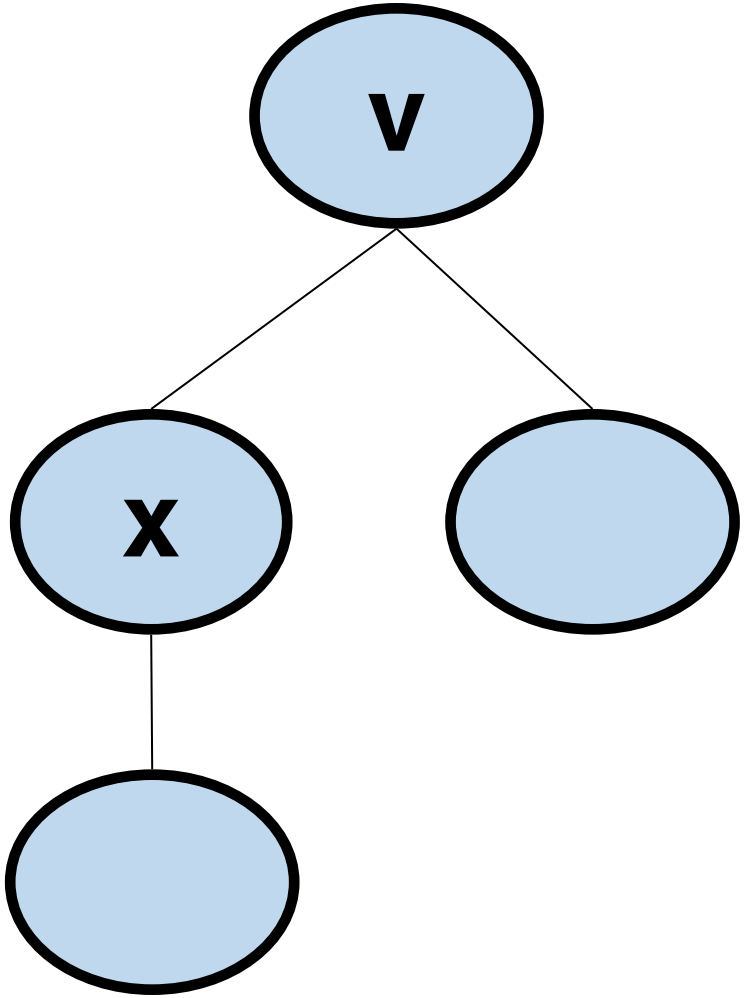
Irmão
= irmã

Filho
= filha
= descendente
= sucessor

TERMINOLOGIA

Se **x** pertence à subárvore enraizada em **v**:

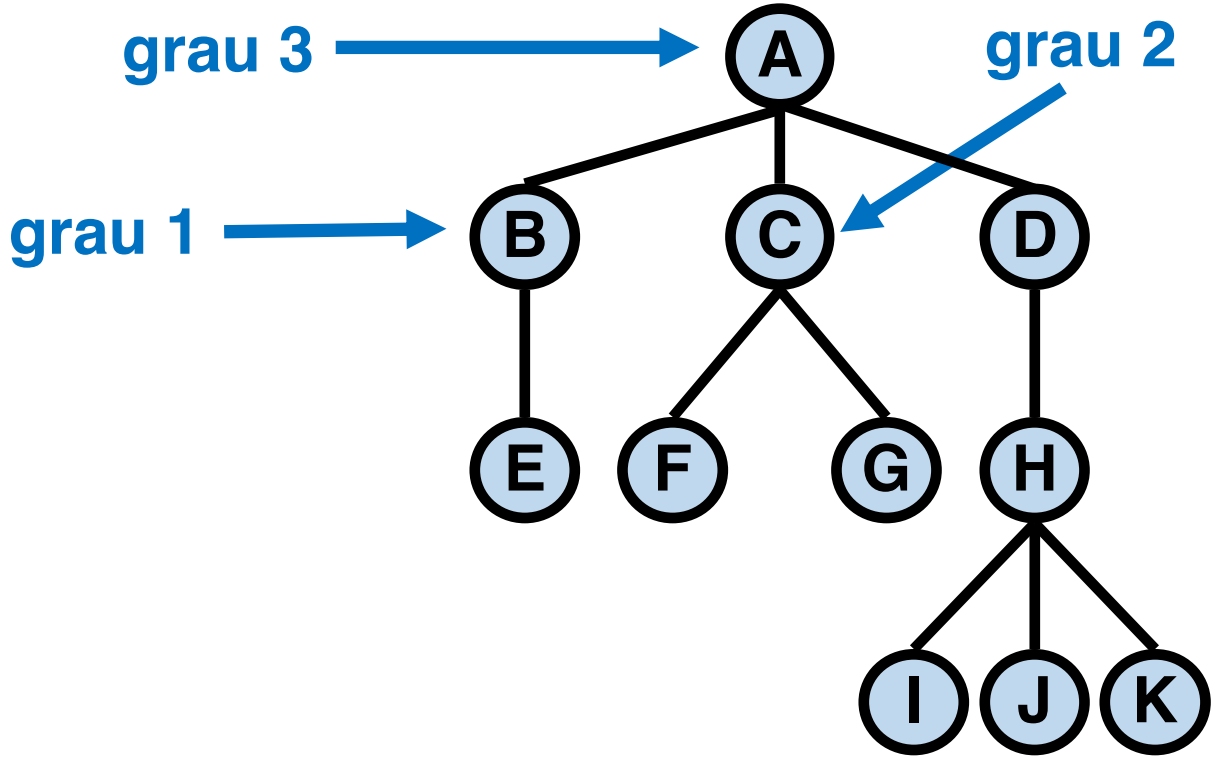
- x é **descendente** de v
- v é **ancestral** de x



TERMINOLOGIA: GRAU DE UM NÓ

Grau (ou grau de saída)

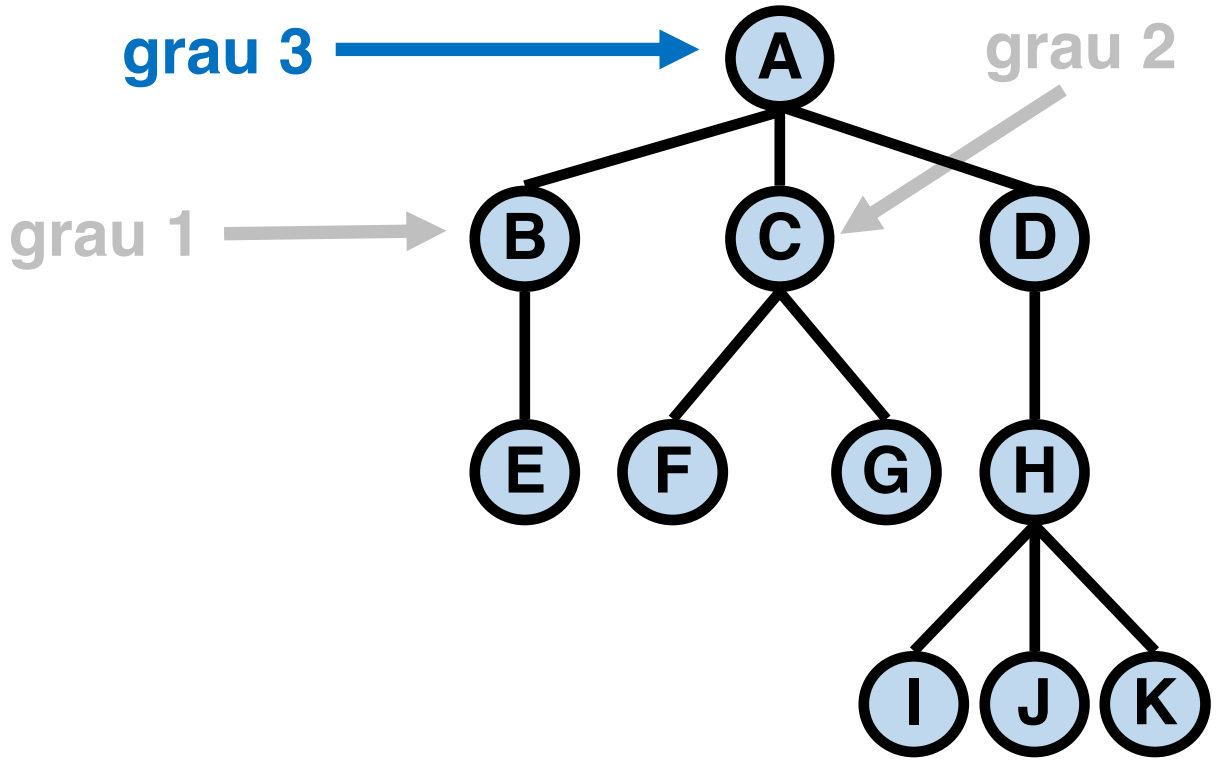
- número de sub-árvores do nó ou
- número de filhos de um nó



TERMINOLOGIA: GRAU DA ÁRVORE

Grau de uma árvore

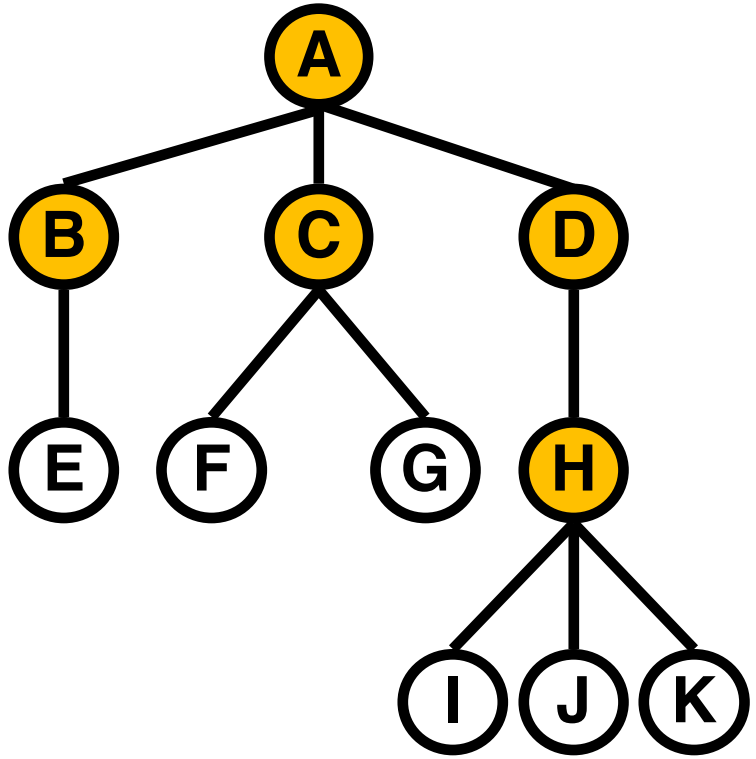
- máximo entre os graus de seus nós



TERMINOLOGIA: NÓ INTERNO

Nó interno (ou nó de derivação)

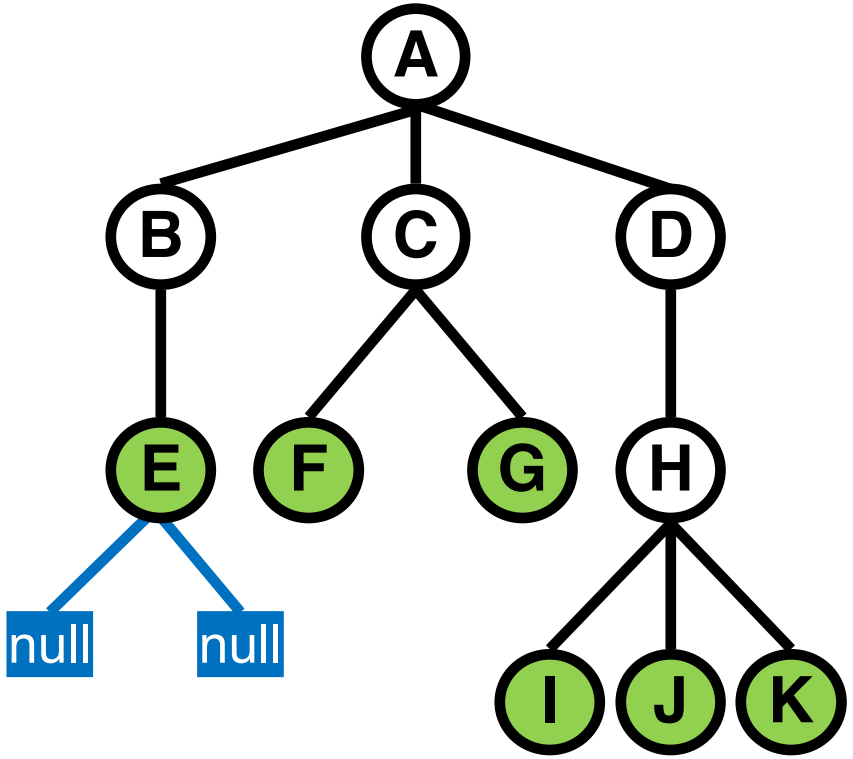
- nó com grau maior do que zero (tem pelo menos um filho)



TERMINOLOGIA: NÓ FOLHA

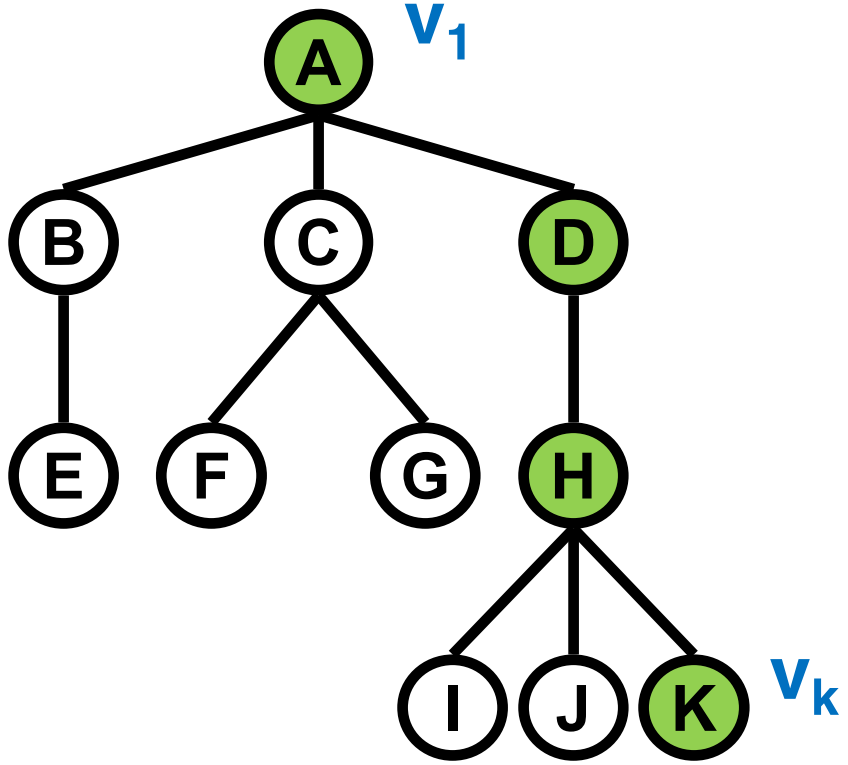
Nó folha (nó terminal ou externo)

- nó com grau igual a zero (nó sem filhos)



TERMINOLOGIA: CAMINHO

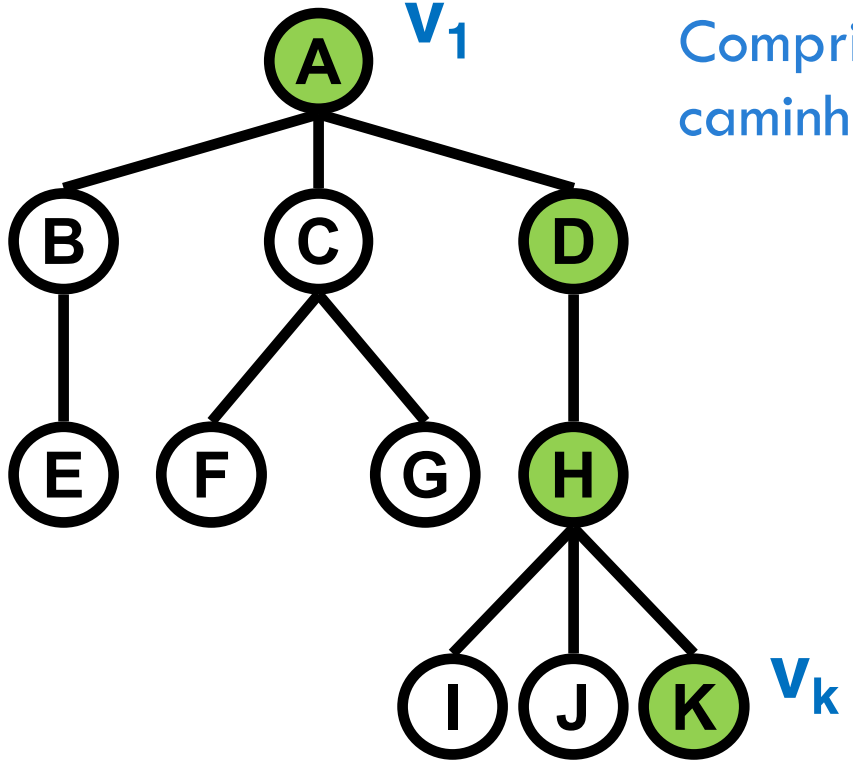
Um caminho é uma sequência de nós consecutivos distintos entre dois nós



TERMINOLOGIA: COMPRIMENTO DO CAMINHO

Comprimento do caminho

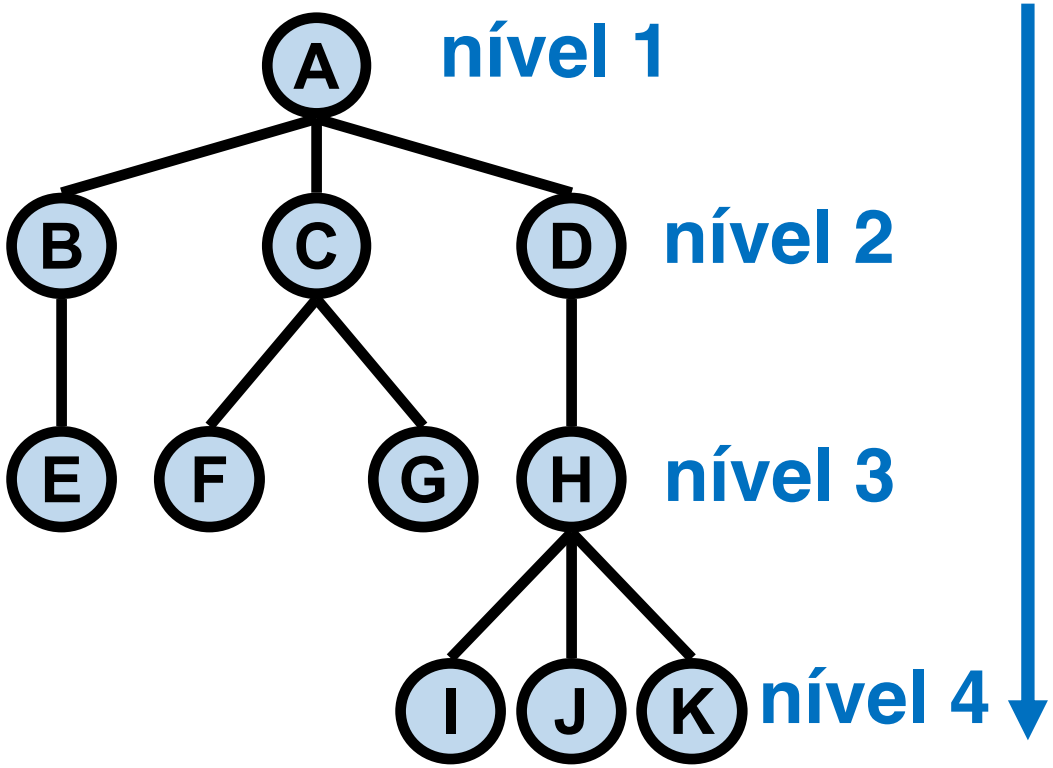
- Número de ligações entre os nós do caminho



Comprimento do caminho= 3

TERMINOLOGIA: NÍVEL

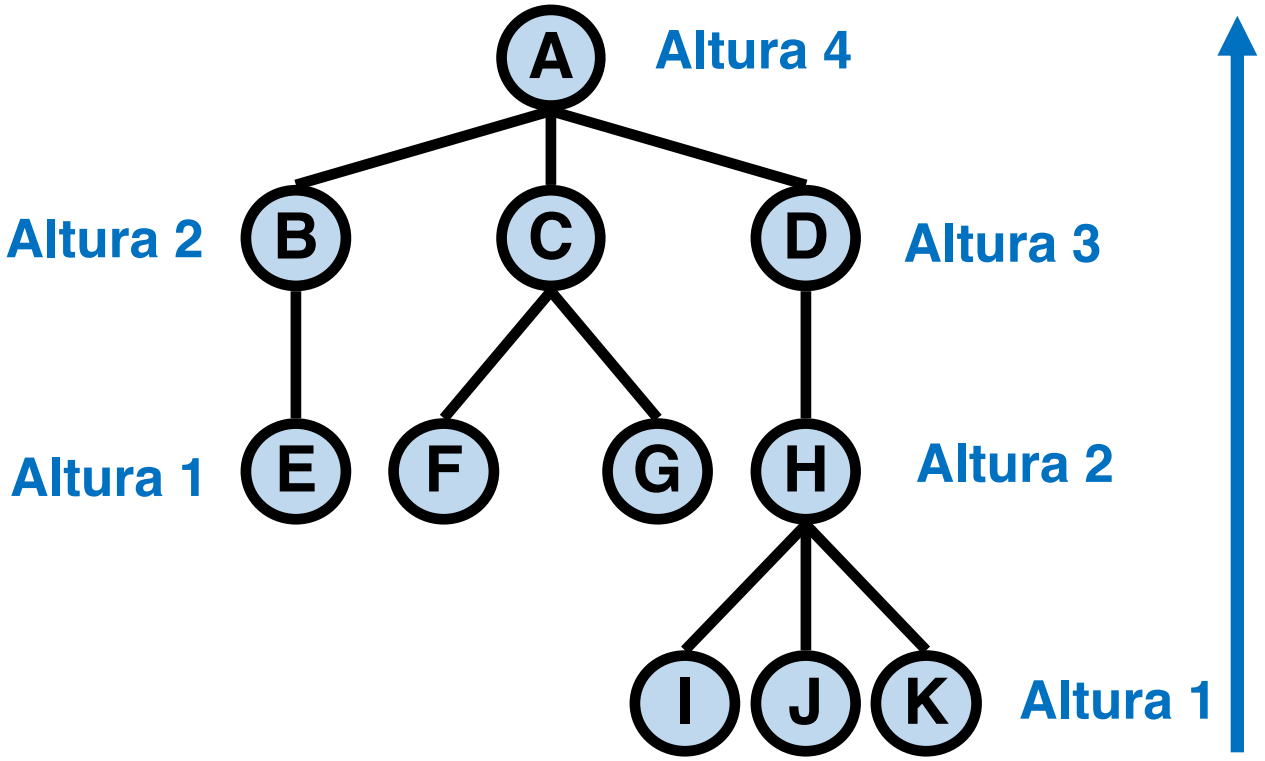
Nível = número de ligações entre **a raiz e o nó**, acrescido de uma unidade



TERMINOLOGIA: ALTURA DE UM NÓ

Altura (profundidade) de um nó

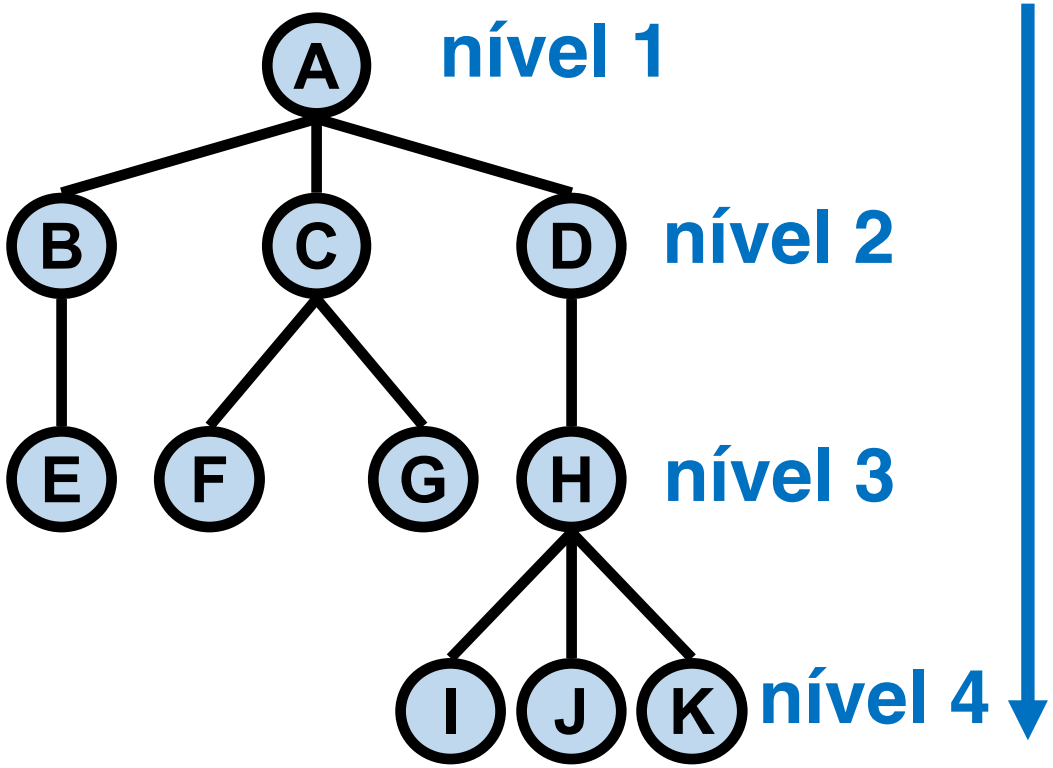
- número de ligações entre **o nó e o nó folha (descendente dele) de maior nível,** acrescido de uma unidade
- Altura de nó folha é 1



TERMINOLOGIA: ALTURA DA ÁRVORE

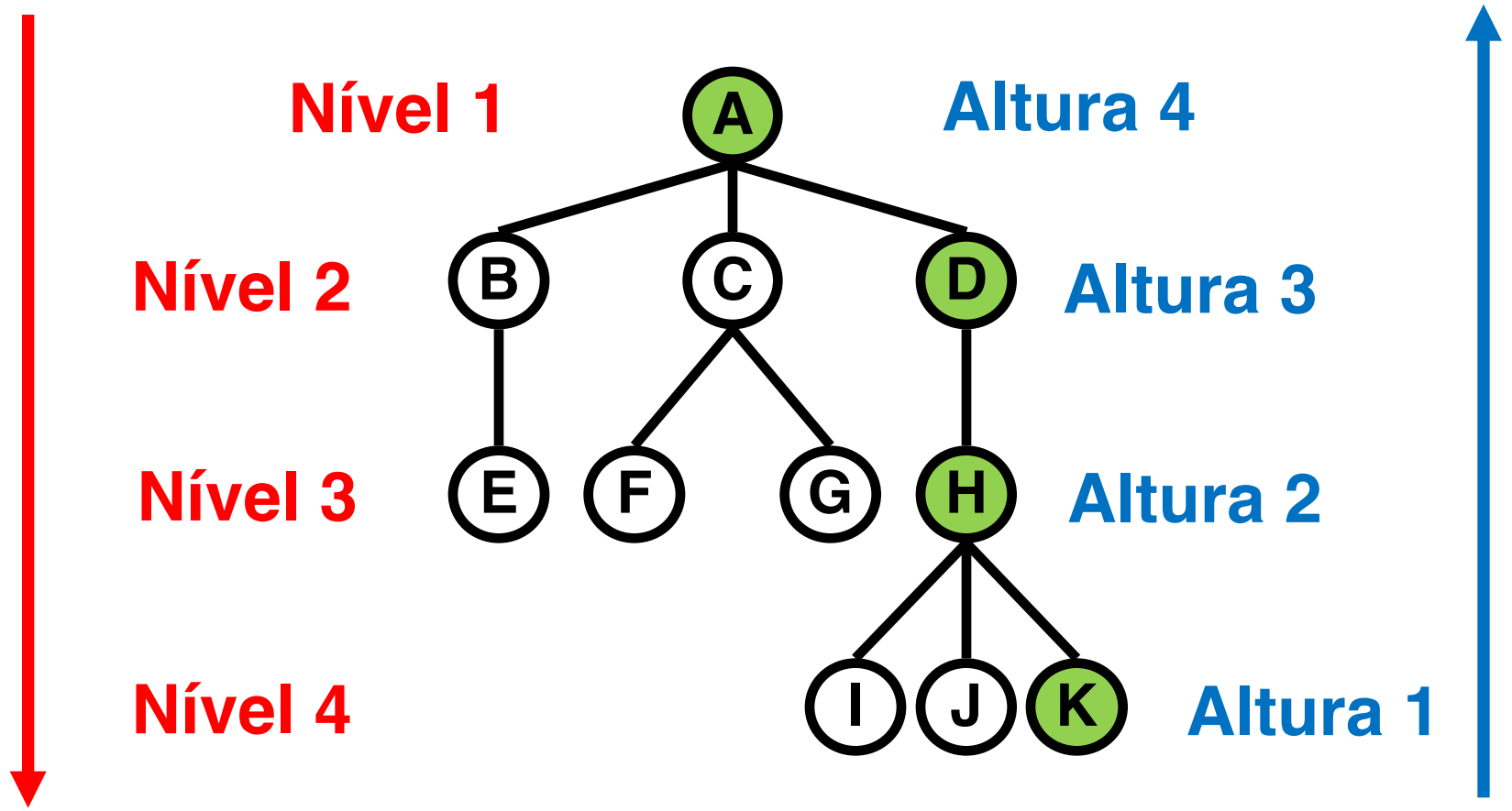
Altura (profundidade) da árvore

- maior nível dentre seus nós
- (equivalente à altura do nó raiz)



Altura da Árvore = 4

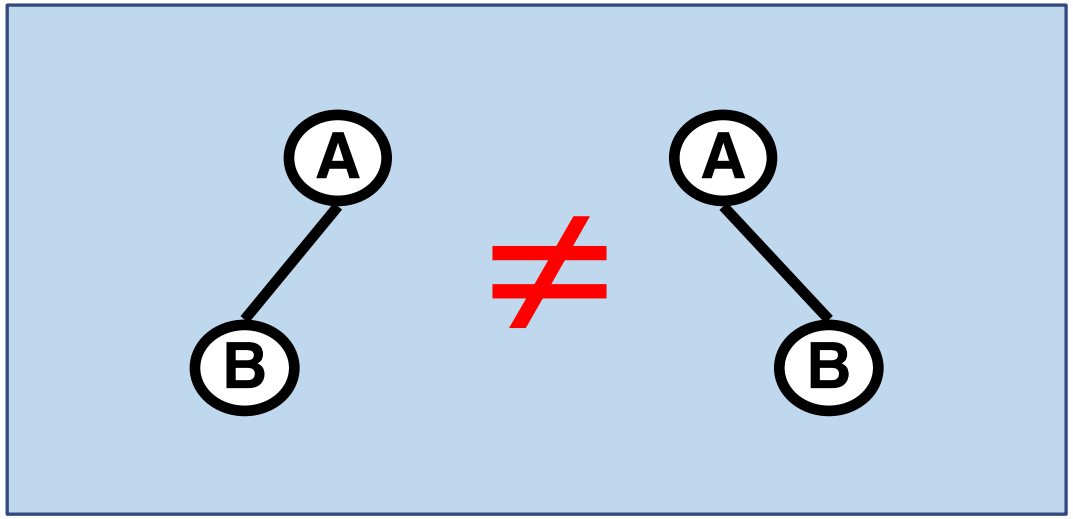
ALTURA X NÍVEL

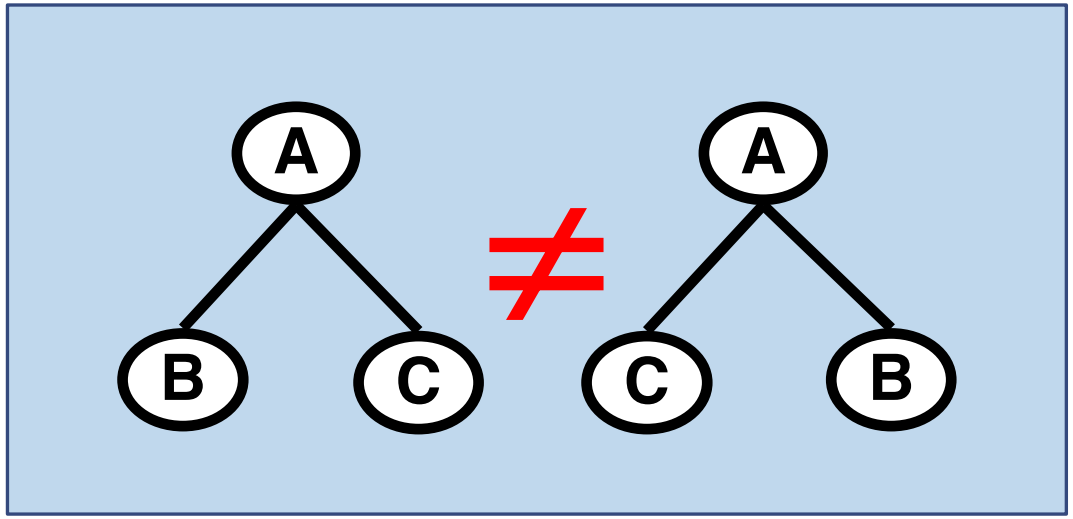


TERMINOLOGIA: ÁRVORE ORDENADA

Árvore ordenada

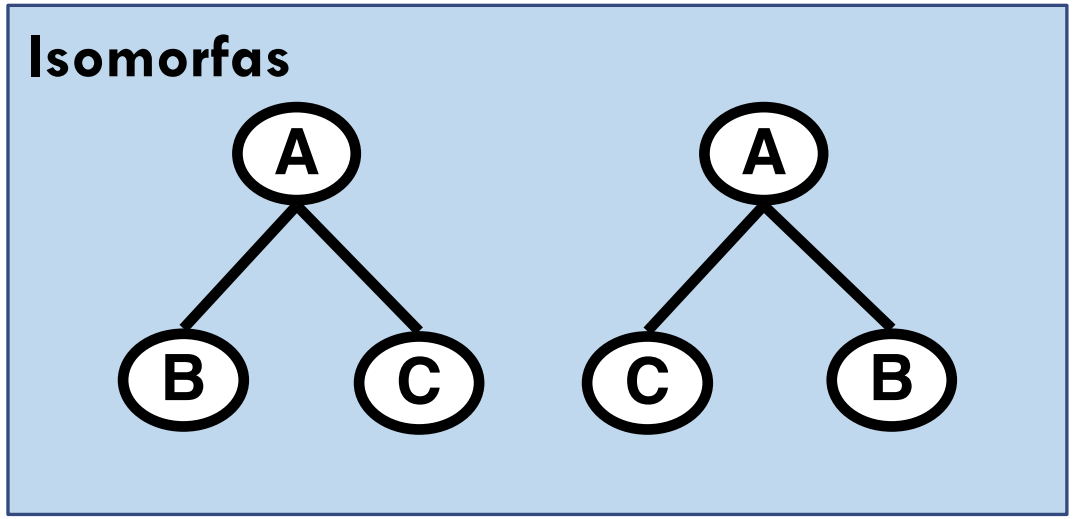
- Ordem das sub-árvores é relevante
- Uma árvore ordenada é aquela na qual os filhos estão ordenados
- Assume-se que essa ordenação se desenvolva da esquerda para a direita





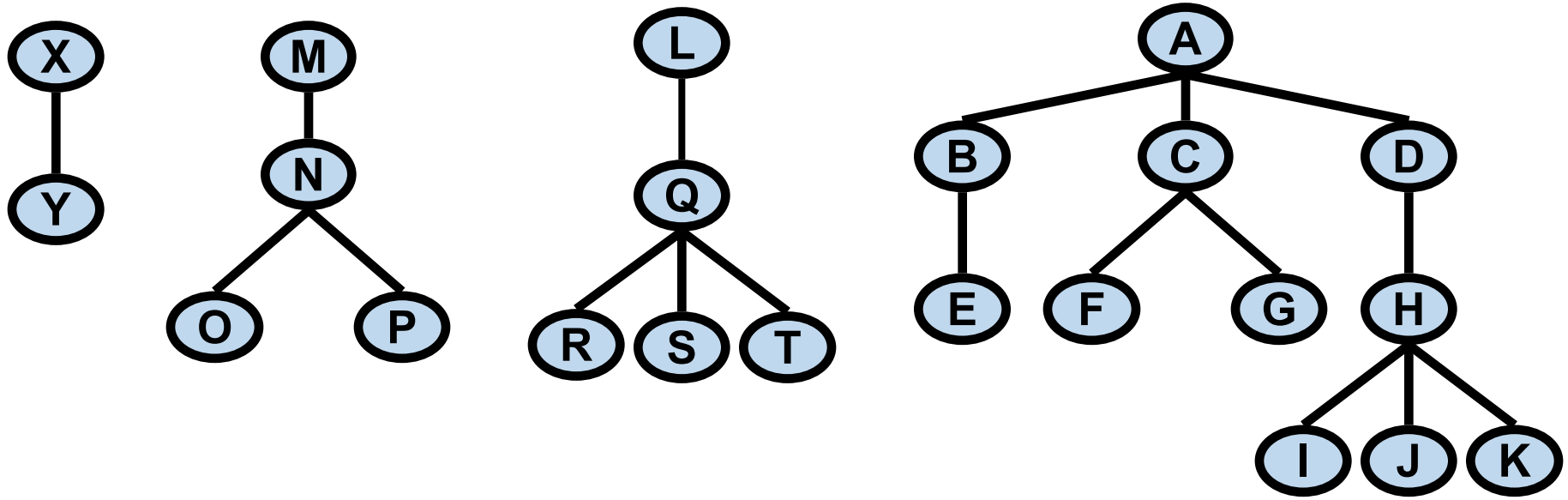
TERMINOLOGIA: ÁRVORE ISOMORFA

Duas árvores são isomorfas quando puderem se tornar coincidentes pela permutação da ordem das subárvores



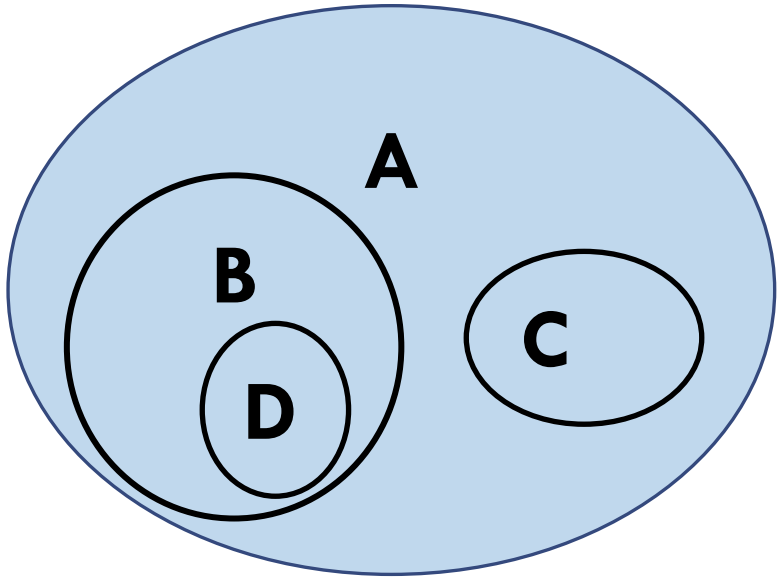
TERMINOLOGIA: FLORESTA

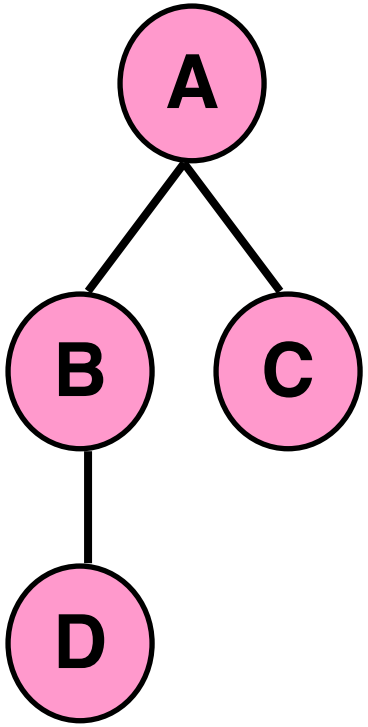
Floresta = Conjunto de árvores



FORMAS DE REPRESENTAÇÃO

Diagrama de inclusão





FORMAS DE REPRESENTAÇÃO

Diagrama de barras

A.....

B.....

D.....

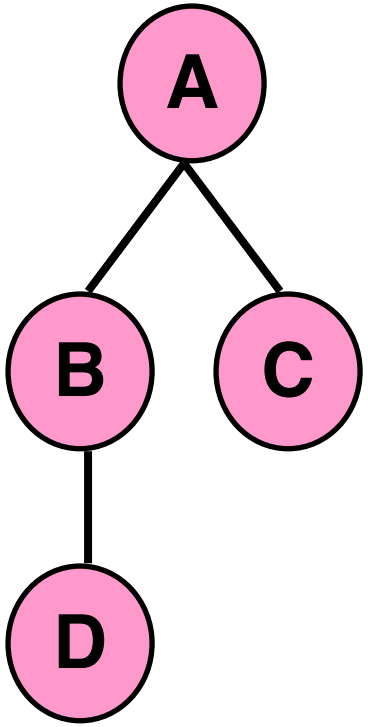
C.....

A

.....B

..... D

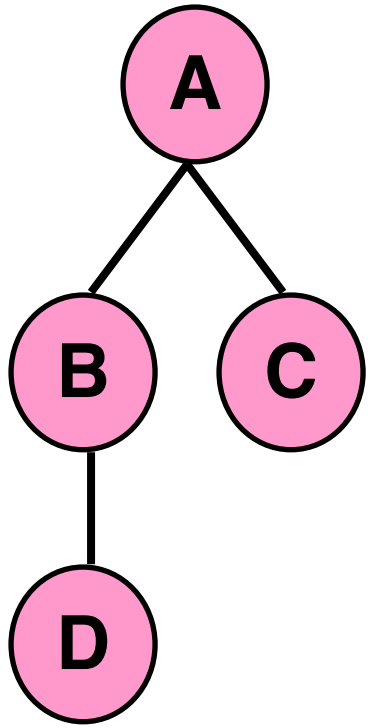
.....C



FORMAS DE REPRESENTAÇÃO

Níveis

1A; 1.1B; 1.1.1D; 1.2C



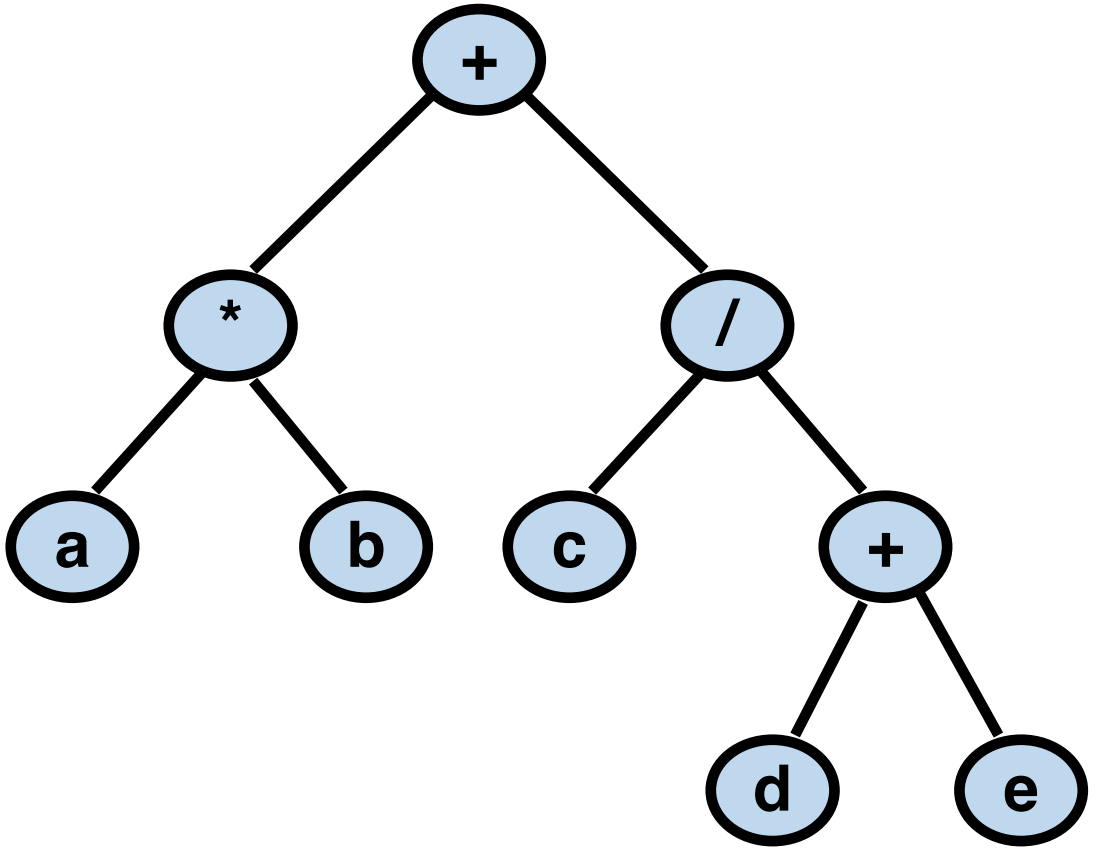
Aninhamento

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

ÁRVORES BINÁRIAS

ÁRVORES BINÁRIAS

Árvores binárias são uma das árvores mais usadas em computação



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

DEFINIÇÃO

Conjunto finito T de zero ou mais nós (nodos), tal que:

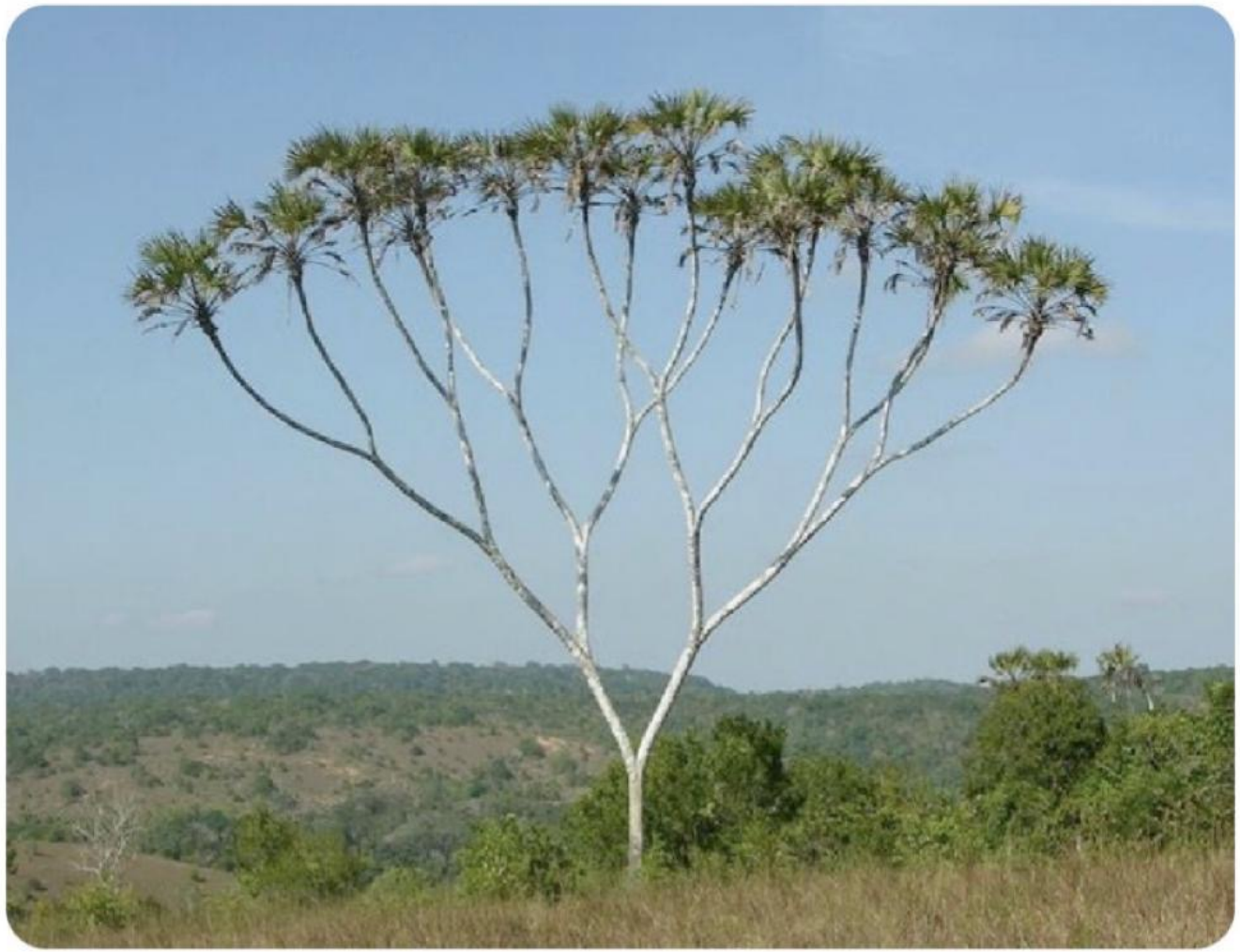
Se número de nós é maior do que zero

- existe um nó denominado raiz da árvore
- os demais nós formam 2 conjuntos disjuntos S_1, S_2 (subárvore da esquerda e subárvore da direita) onde cada um destes é uma árvore binária

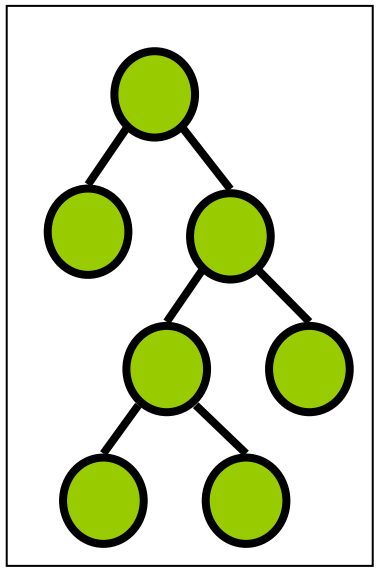
Se número de nós é igual a zero

- árvore vazia

The binary tree actually exists!!

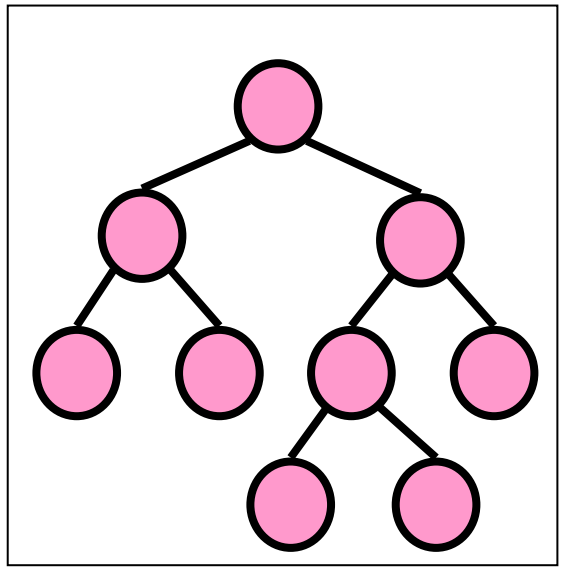


TIPOS DE ÁRVORES BINÁRIAS



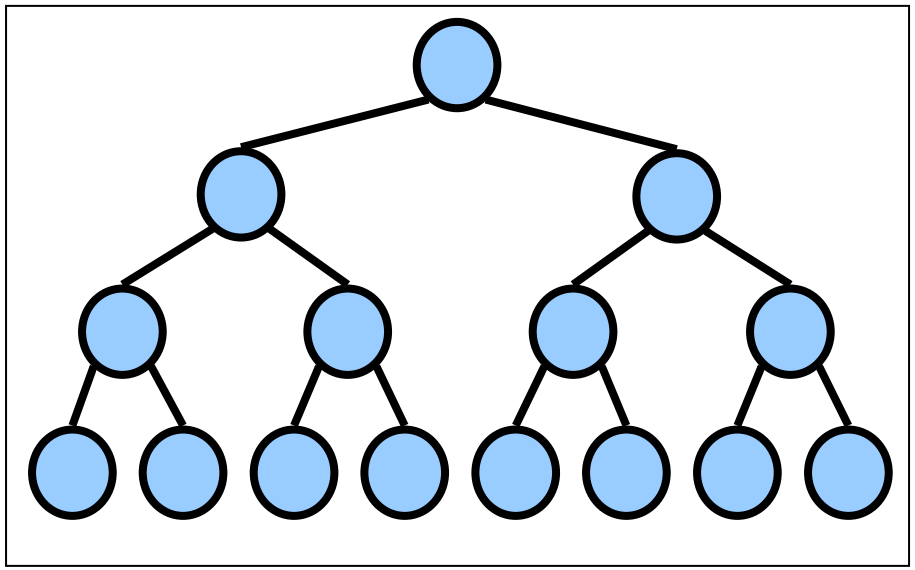
Estritamente Binária

0 ou 2 filhos



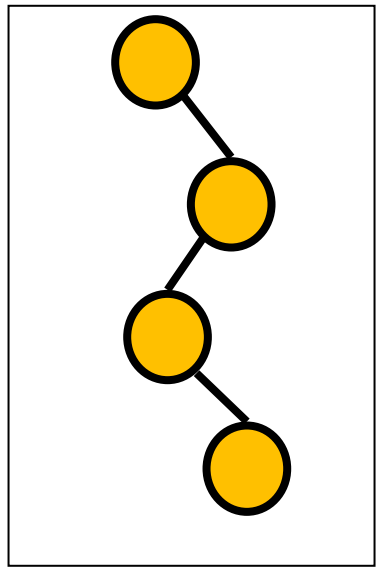
Binária Completa

Sub-árvores vazias apenas no último ou penúltimo nível



Binária Cheia

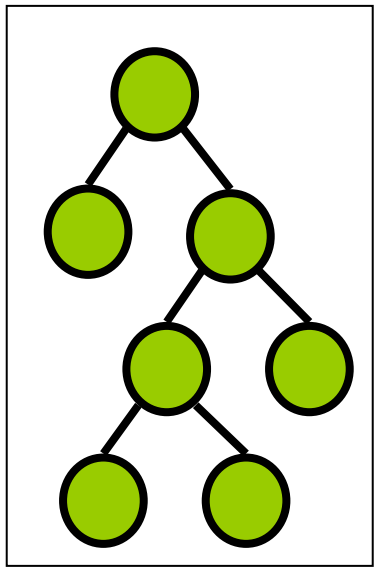
Sub-árvores vazias somente no último nível



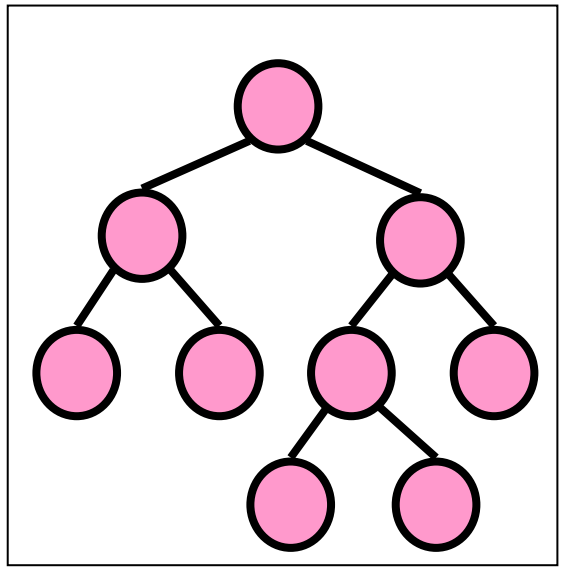
Zigue Zague

Nós internos com 1 subárvore vazia

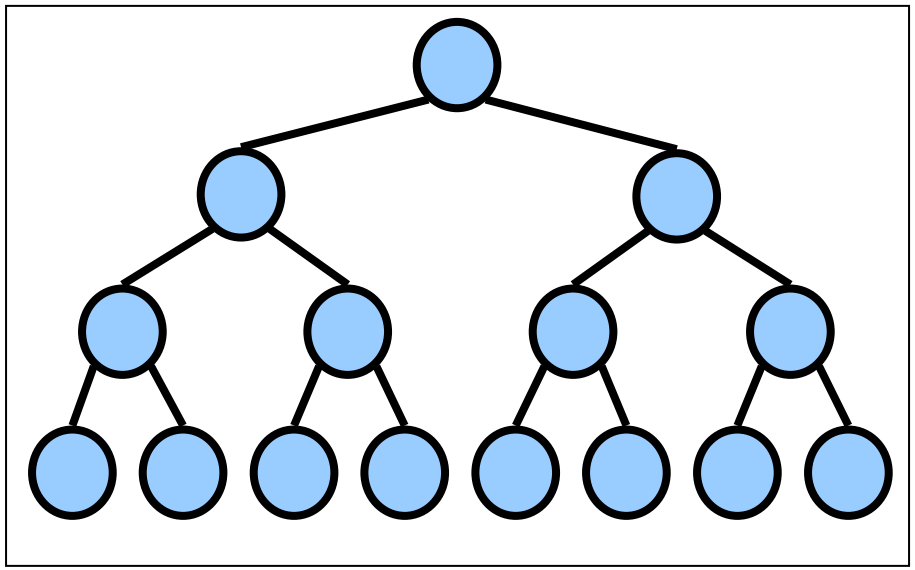
QUAL DELAS POSSUI ALTURA MÁXIMA?
(CONSIDERANDO O MESMO NÚMERO N DE NÓS NA ÁRVORE)



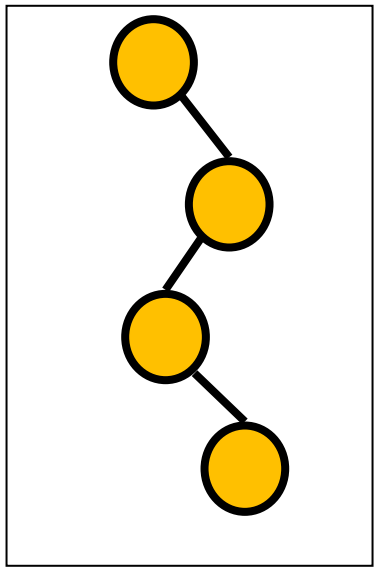
Estritamente Binária
0 ou 2 filhos



Binária Completa
Sub-árvores vazias apenas no último ou penúltimo nível



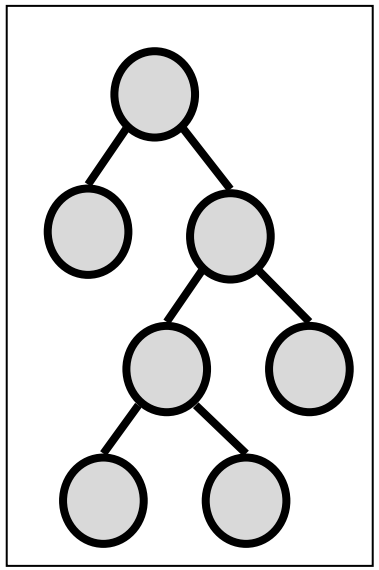
Binária Cheia
Sub-árvores vazias somente no último nível



Zigue Zague
Nós internos com 1 subárvore vazia

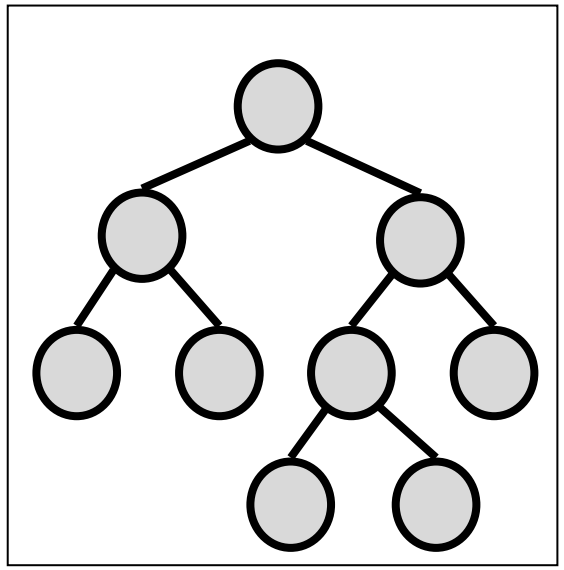
QUAL DELAS POSSUI ALTURA MÁXIMA?

(CONSIDERANDO O MESMO NÚMERO N DE NÓS NA ÁRVORE)



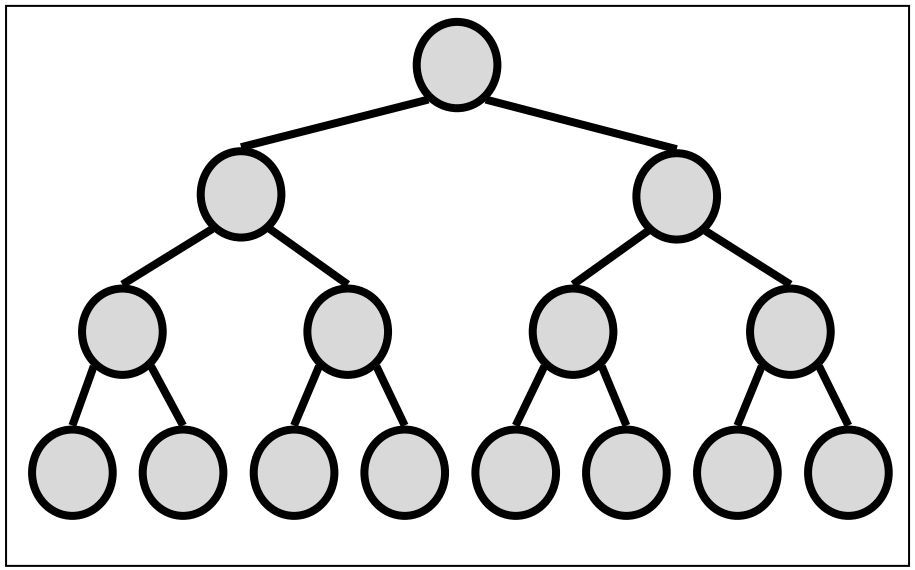
Estritamente Binária

0 ou 2 filhos



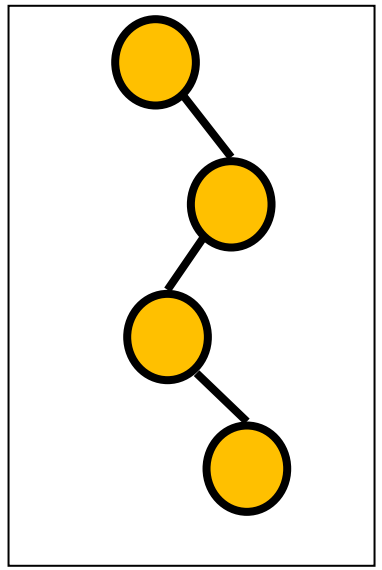
Binária Completa

Sub-árvores vazias apenas no último ou penúltimo nível



Binária Cheia

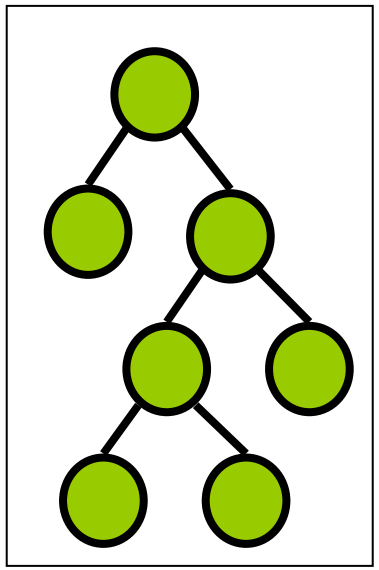
Sub-árvores vazias somente no último nível



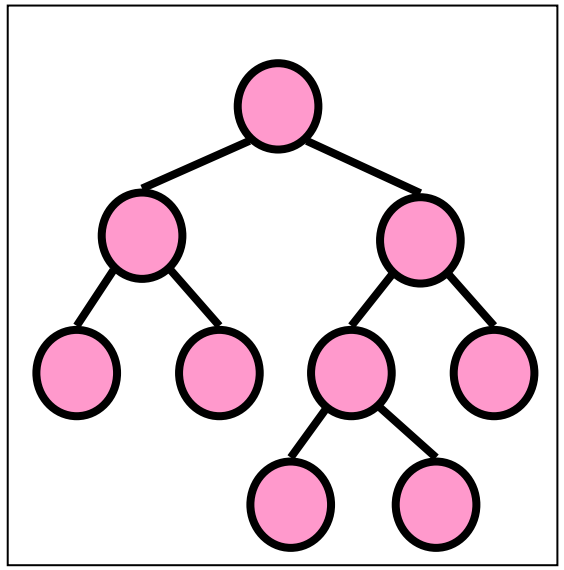
Zigue Zague

Nós internos com 1 subárvore vazia

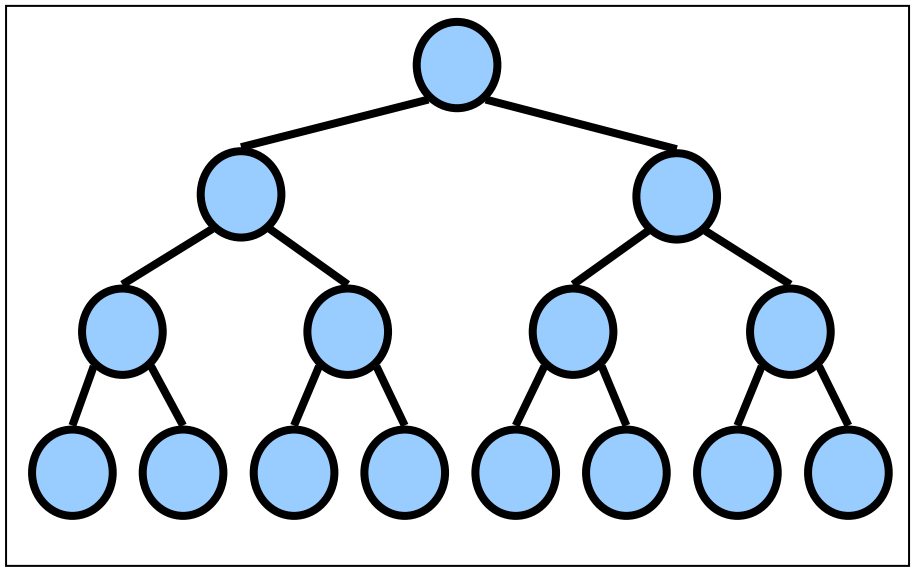
QUAL DELAS POSSUI ALTURA MÍNIMA?
(CONSIDERANDO O MESMO NÚMERO N DE NÓS NA ÁRVORE)



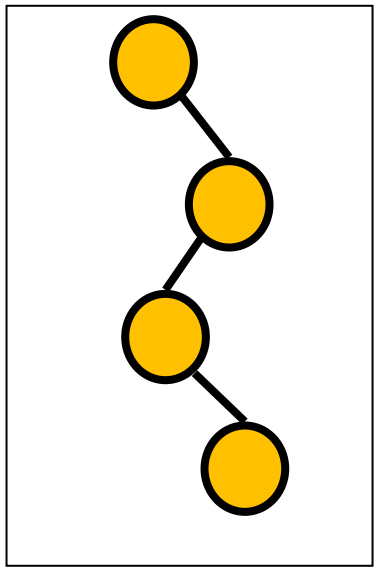
Estritamente Binária
0 ou 2 filhos



Binária Completa
Sub-árvores vazias apenas no último ou penúltimo nível



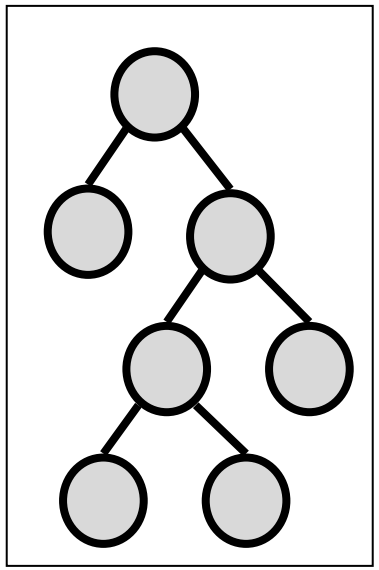
Binária Cheia
Sub-árvores vazias somente no último nível



Zigue Zague
Nós internos com 1 subárvore vazia

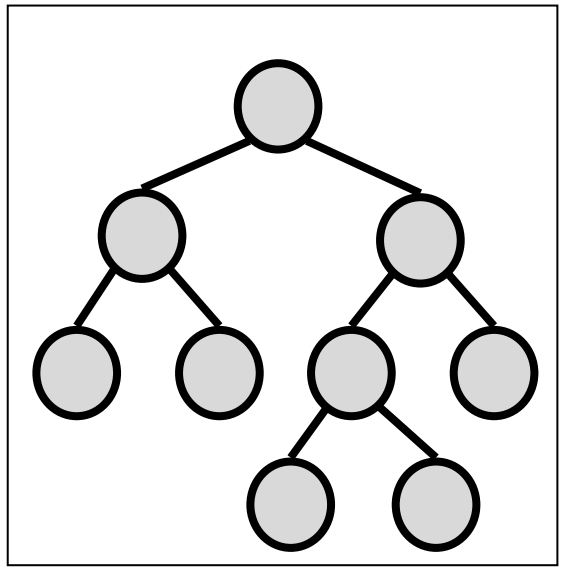
QUAL DELAS POSSUI ALTURA MÍNIMA?

(CONSIDERANDO O MESMO NÚMERO N DE NÓS NA ÁRVORE)



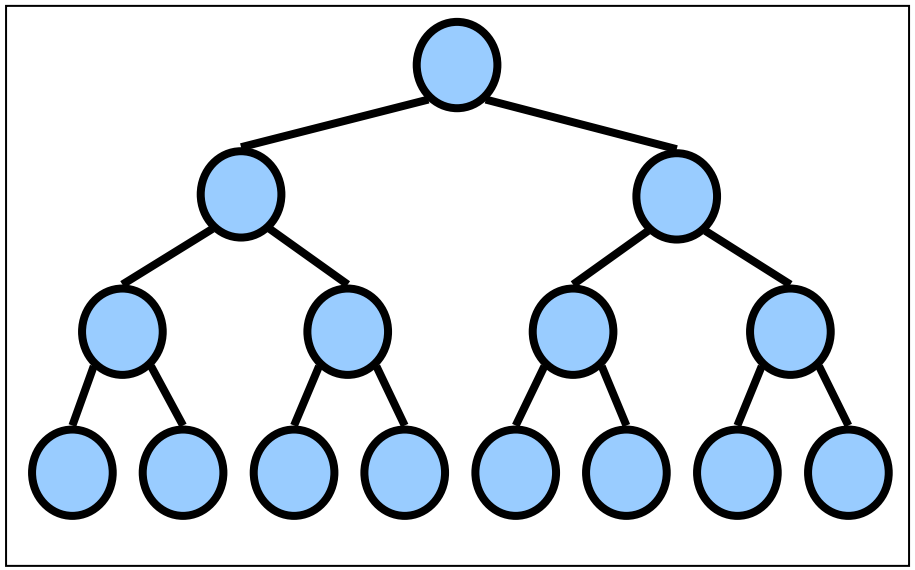
Estritamente Binária

0 ou 2 filhos



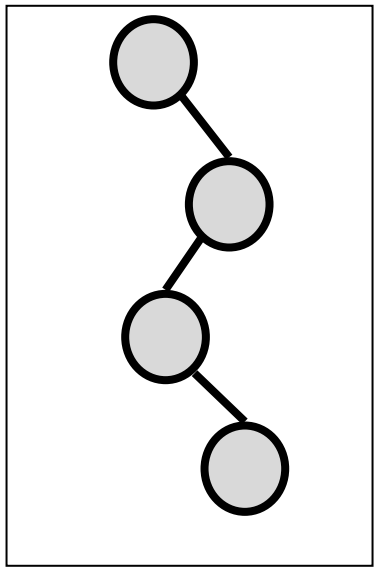
Binária Completa

Sub-árvores vazias apenas no último ou penúltimo nível



Binária Cheia

Sub-árvores vazias somente no último nível



Zigue Zague

Nós internos com 1 subárvore vazia

IMPLEMENTAÇÃO DE ÁRVORES BINÁRIAS

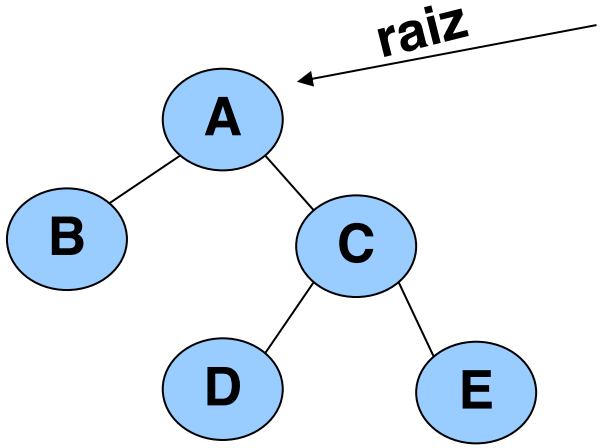
REPRESENTAÇÃO DE ÁRVORE BINÁRIA EM C

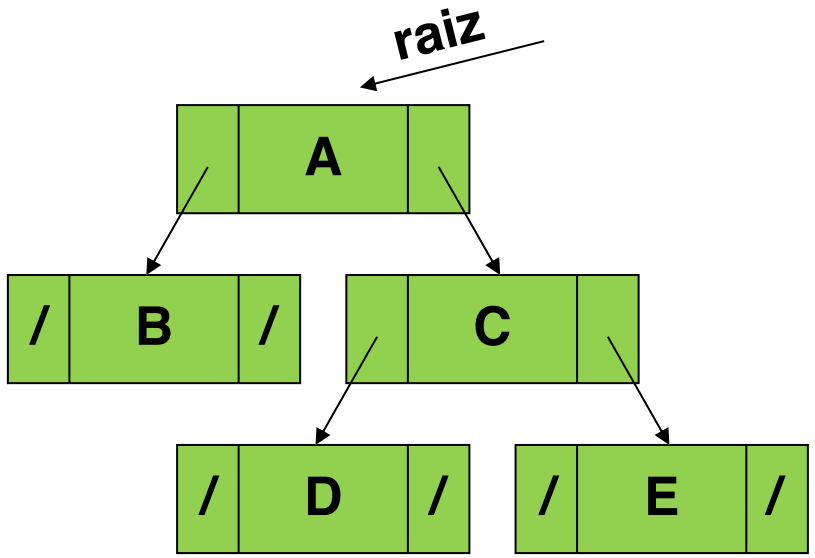
```
/* representação dos nós de a */
typedef struct noA {
   char info;
   struct noA* esq;
   struct noA* dir;
} TNoA;
```

Representação do nó:

REPRESENTAÇÃO DE ÁRVORE BINÁRIA EM C

```
/* representação dos nós de a */
typedef struct noA {
    char info;
    struct noA* esq;
    struct noA* dir;
} TNoA;
```

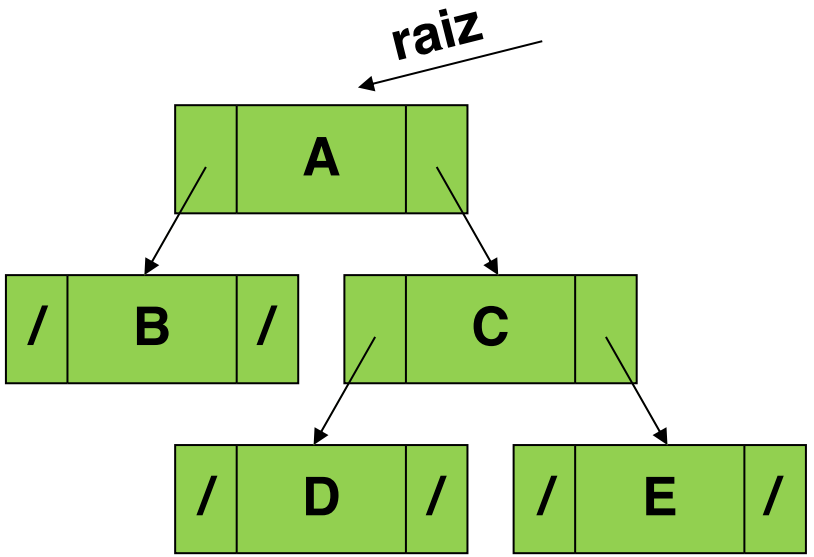




Representação do nó:

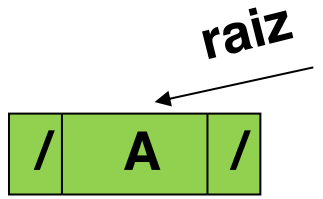
CRIAR NÓ

```
typedef struct noA{
    char info;
    struct noA *esq;
    struct noA *dir;
} TNoA;

TNoA *criaNo(char ch);
```



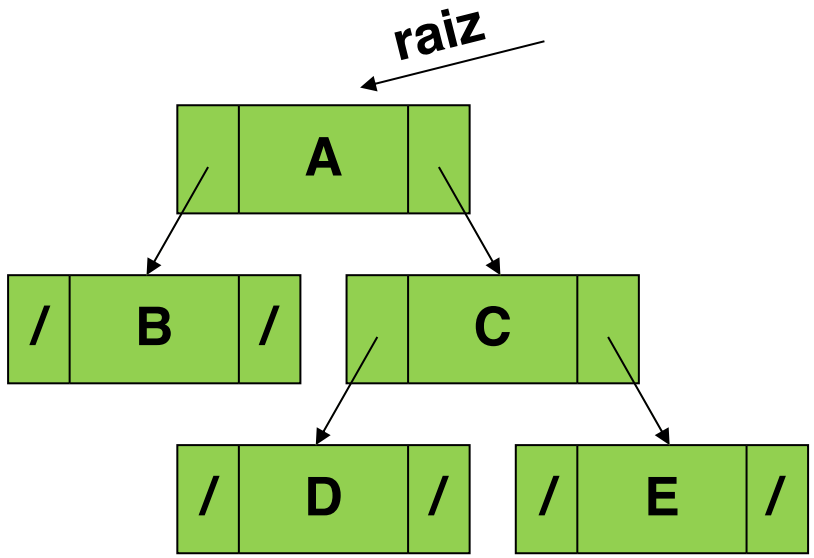
CRIAR RAIZ

```
TNoA *criaNo(char ch) {
    TNoA *novo;
    novo = (TNoA *) malloc(sizeof(TNoA));
    novo->info = ch;
    novo->esq = NULL;
    novo->dir = NULL;
    return novo;
}

int main(void) {
    TNoA *raiz;
    raiz = criaNo('A');

}
```



CRIAR FILHOS

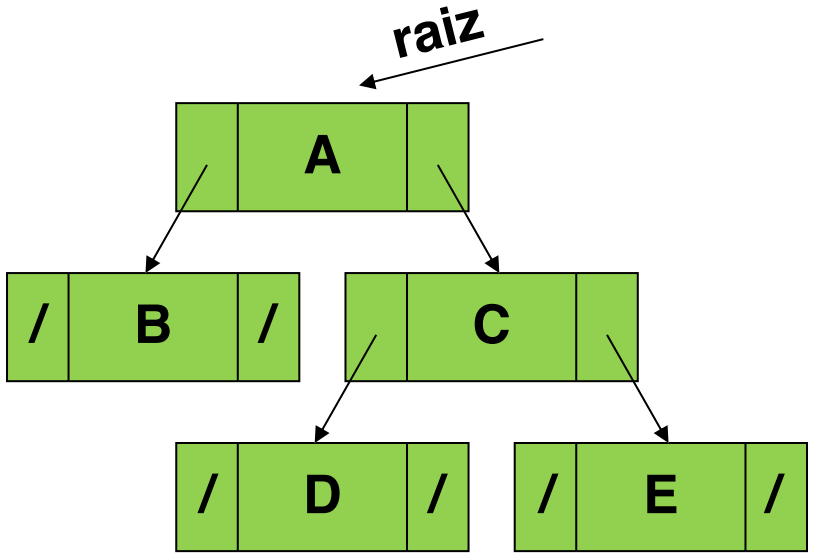
```
TNoA *criaNo(char ch) {
    TNoA *novo;
    novo = (TNoA *) malloc(sizeof(TNoA));
    novo->info = ch;
    novo->esq = NULL;
    novo->dir = NULL;
    return novo;
}

int main(void) {
    TNoA *raiz;
    raiz = criaNo('A');
    raiz->esq = criaNo('B');
    raiz->dir = criaNo('C');
    raiz->dir->esq = criaNo('D');
    raiz->dir->dir = criaNo('E');
    imprime(raiz, 0);
};
```



CÓDIGO COMPLETO

Ver código completo em C no site da disciplina (inclui função para imprimir a árvore usando diagrama de barras)

```
A
--B
----vazio
----vazio
--C
----D
------vazio
------vazio
----E
------vazio
------vazio
```



CAMINHAMENTOS EM ÁRVORES BINÁRIAS

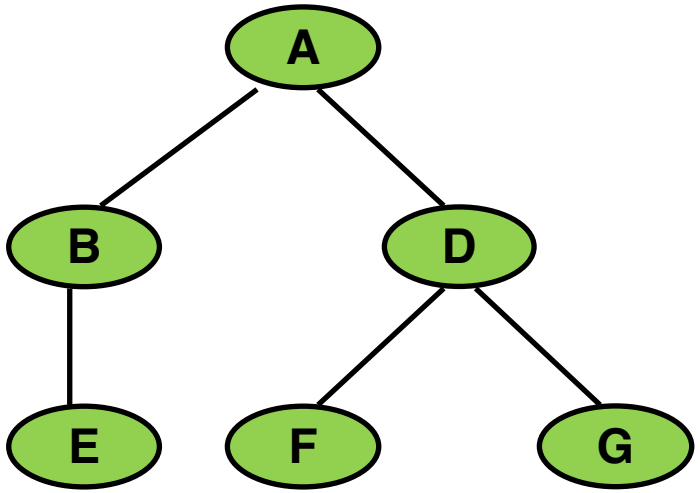
CONSULTA A UM NÓ

Acesso sempre através da raiz

Cada nó deve ser "visitado" uma vez, e apenas uma vez

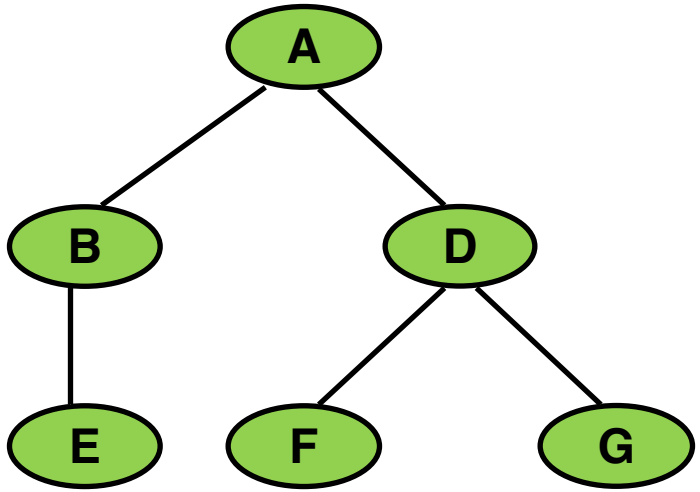
Visita a um nó:

- Acesso a um nó para realizar alguma **operação**



CAMINHAMENTOS

Método de percurso sistemático de todos os nós de uma árvore, de modo que cada nó seja visitado exatamente uma vez

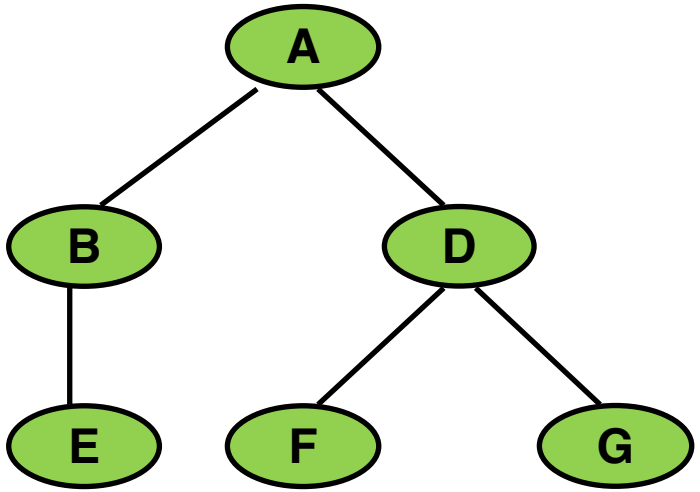


CAMINHAMENTOS

Um caminhamento (ou percurso) define uma sequência de nós

Cada nó passa a ter um nó seguinte, ou um nó anterior, ou ambos (exceto árvore com 1 só nó)

Sequência de nós depende do caminhamento



Exemplo:

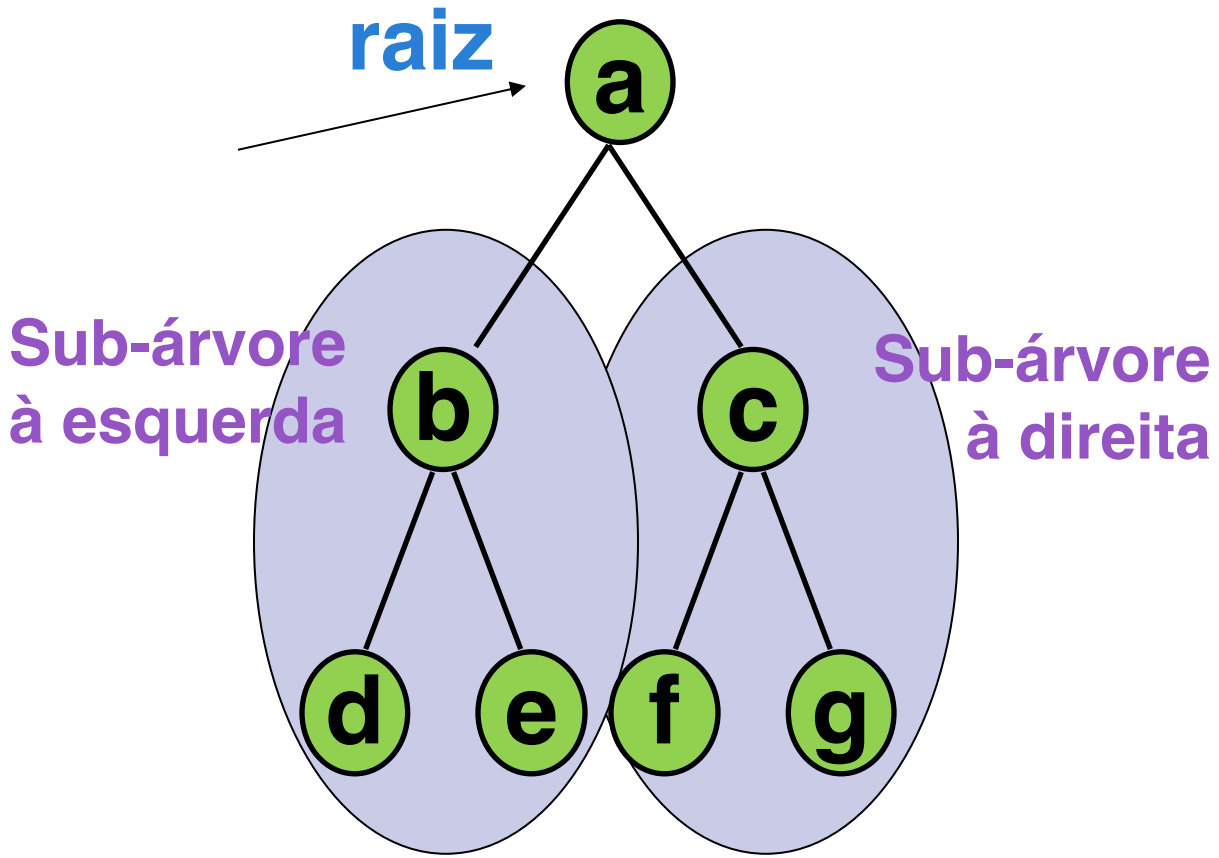
Caminhamento 1:

- A – B – D – E – F - G

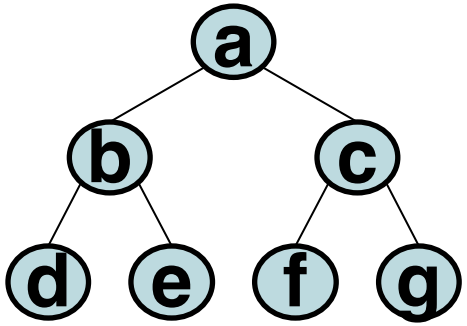
Caminhamento 2:

- A – B – E – D- F- G

PRINCIPAIS CAMINHAMENTOS



CAMINHAMENTOS



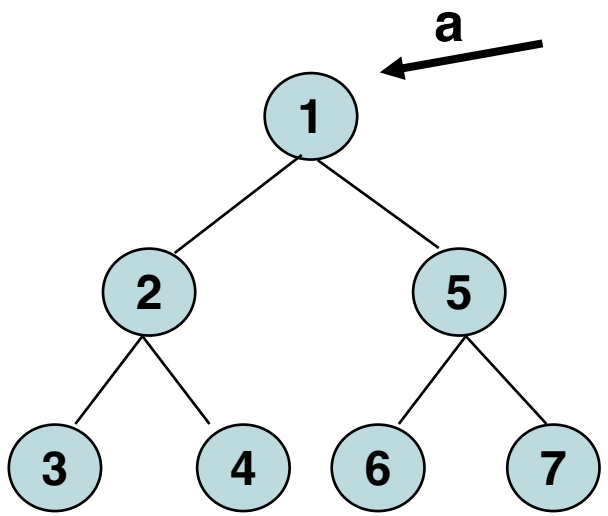
Pré-Ordem (Profundidade)

.Visita a raiz
.Percorre a sub-árvore esquerda
.Percorre a sub-árvore direita

a - b - d - e - c - f - g

CAMINHAMENTO EM PROFUNDIDADE: IMPLEMENTAÇÃO RECURSIVA

```
void profundidade(TNoA* a)
{
      if (a!= NULL)
      {
            printf("%c\n",a->info);
            profundidade(a->esq);
            profundidade(a->dir);
       }
}
```



Profundidade: raiz, esquerda, direita

CAMINHAMENTOS

Pré-Ordem (Profundidade)

.Visita a raiz
.Percorre a sub-árvore esquerda
.Percorre a sub-árvore direita

a - b - d - e - c - f - g

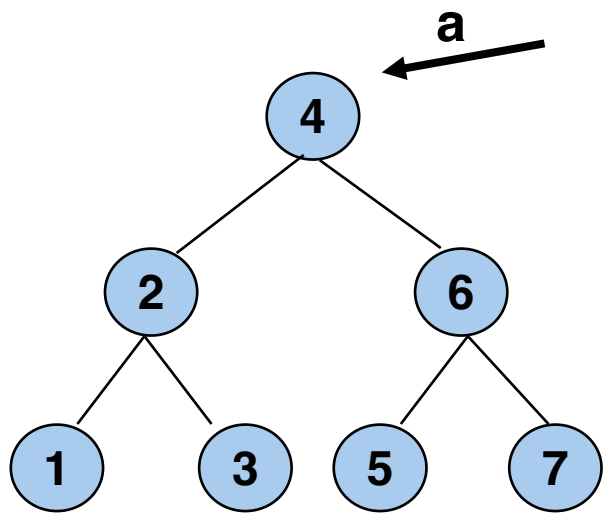
Ordem Simétrica

.Percorre a sub-árvore esquerda
.Visita a raiz
.Percorre a sub-árvore direita

d - b - e - a - f - c - g

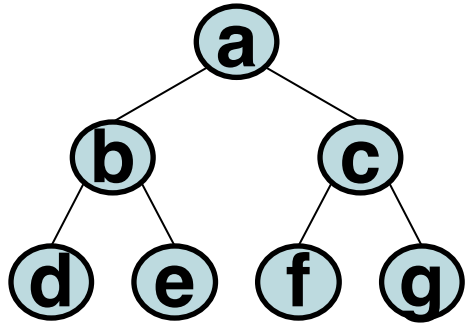
CAMINHAMENTO EM ORDEM SIMÉTRICA: IMPLEMENTAÇÃO RECURSIVA

```
void simetrica(TNoA* a)
{
      if (a!= NULL)
      {
            simetrica(a->esq);
            printf("%c\n",a->info);
            simetrica(a->dir);
       }
}
```



Simétrica: esquerda, raiz, direita

CAMINHAMENTOS



Pré-Ordem (Profundidade)

.Visita a raiz
.Percorre a sub-árvore esquerda
.Percorre a sub-árvore direita

a - b - d - e - c - f - g

Pós-Ordem

.Percorre a sub-árvore esquerda
.Percorre a sub-árvore direita
.Visita a raiz

d - e - b - f - g - c - a

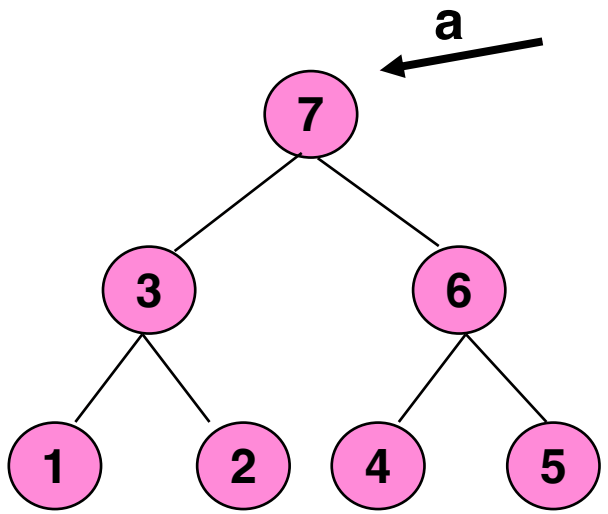
Ordem Simétrica

.Percorre a sub-árvore esquerda
.Visita a raiz
.Percorre a sub-árvore direita

d - b - e - a - f - c - g

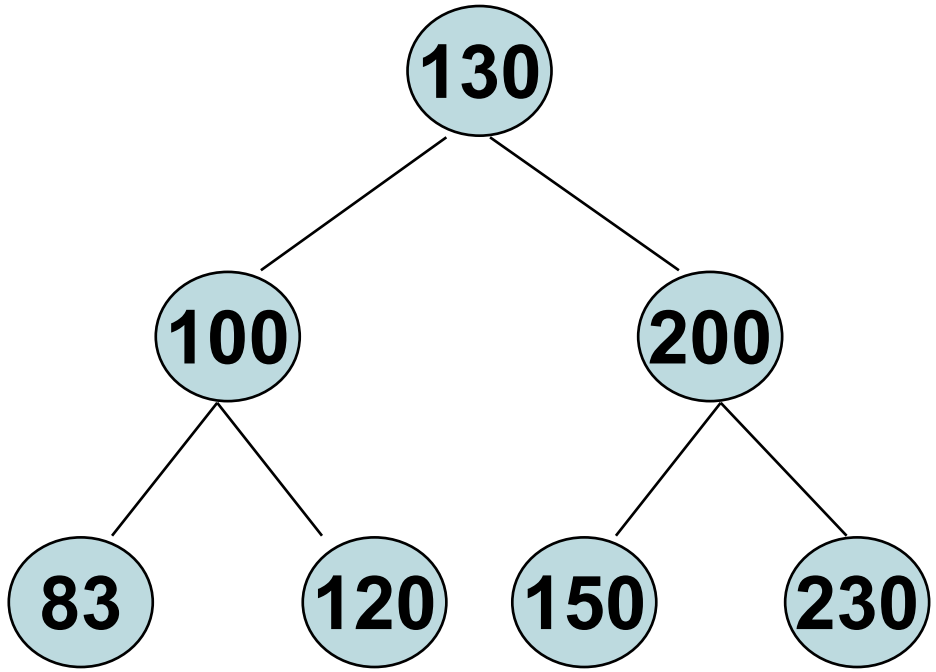
CAMINHAMENTO EM PÓS-ORDEM: IMPLEMENTAÇÃO RECURSIVA

```
void posOrdem(TNoA* a)
{
      if (a!= NULL)
      {
            posOrdem(a->esq);
            posOrdem(a->dir);
            printf("%c\n",a->info);
       }
}
```



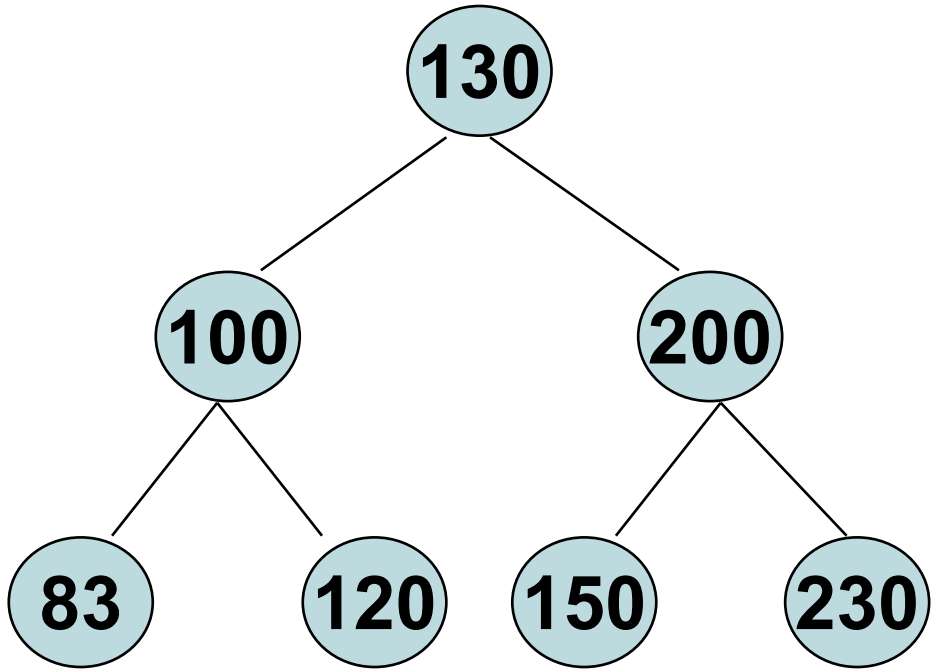
Pós-ordem: esquerda, direita, raiz

EXERCÍCIO



Profundidade? (raiz, esquerda, direita)
Pós-Ordem? (esquerda, direita, raiz)

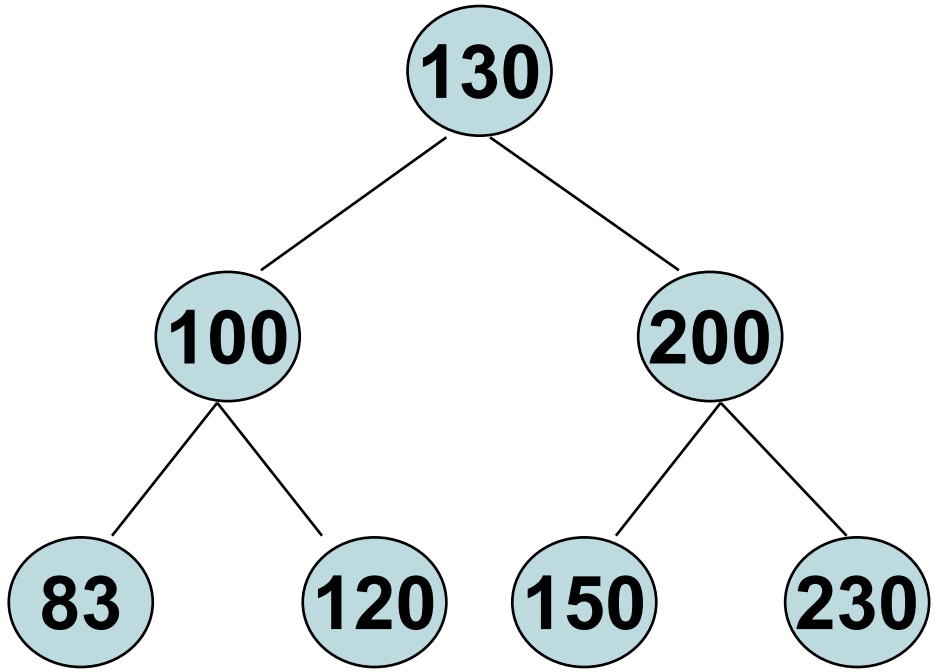
EXERCÍCIO



Profundidade? (raiz, esquerda, direita)

130– 100 – 83 – 120 – 200 – 150 – 230

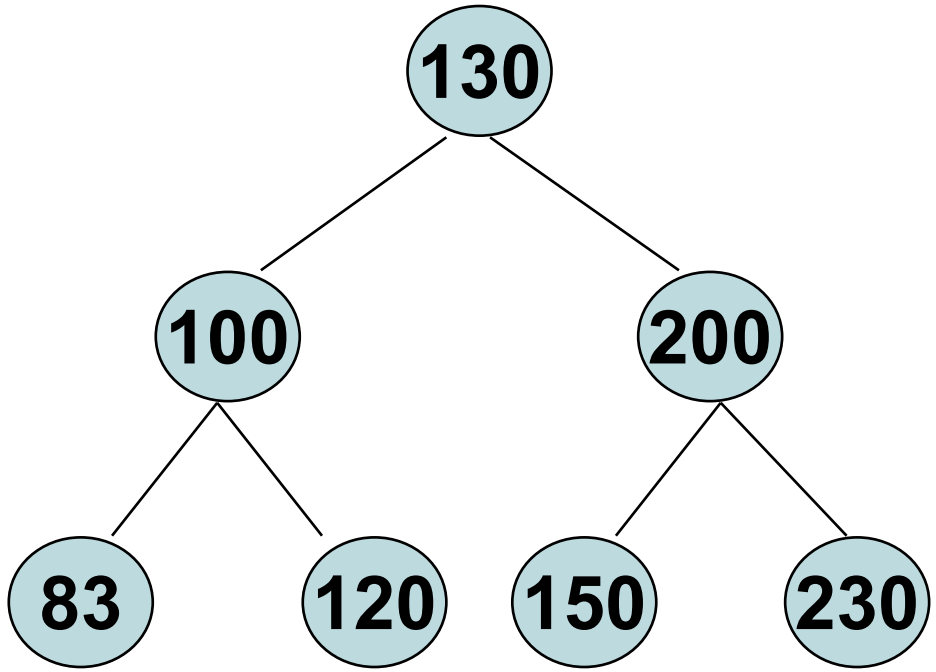
EXERCÍCIO



Profundidade? (raiz, esquerda, direita)

130– 100 – 83 – 120 – 200 – 150 – 230

EXERCÍCIO



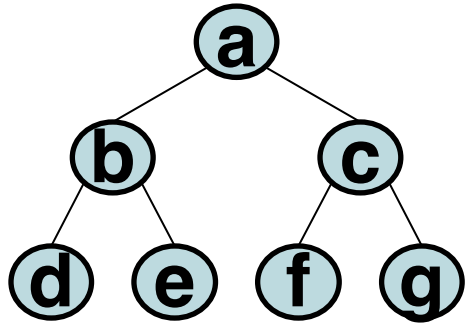
Profundidade? (raiz, esquerda, direita)

130– 100 – 83 – 120 – 200 – 150 – 230

Pós-Ordem? (esquerda, direita, raiz)

83 – 120 – 100 – 150 – 230 – 200 – 130

CAMINHAMENTOS



Pré-Ordem (Profundidade)

.Visita a raiz
.Percorre a sub-árvore esquerda
.Percorre a sub-árvore direita

a - b - d - e - c - f - g

Pós-Ordem

.Percorre a sub-árvore esquerda
.Percorre a sub-árvore direita
.Visita a raiz

d - e - b - f - g - c - a

Ordem Simétrica

.Percorre a sub-árvore esquerda
.Visita a raiz
.Percorre a sub-árvore direita

d - b - e - a - f - c - g

Largura

.Visita é feita por nível, da esquerda para a direita

a - b - c - d - e - f - g

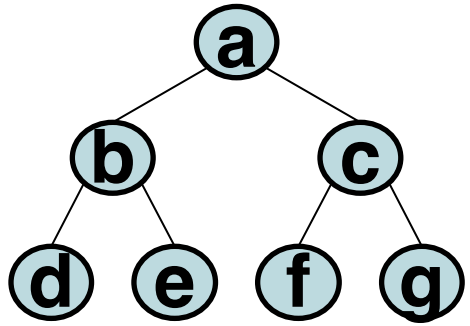
IMPLEMENTAÇÃO: PERCURSO EM LARGURA

Estrutura auxiliar necessária: fila

1. Adicionar a raiz na fila
2. Repetir até que a fila fique vazia
   1. Retirar primeiro da fila (visita)
   2. Adicionar nó da esquerda na fila (se diferente de NULL)
   3. Adicionar nó da direita na fila (se diferente de NULL)

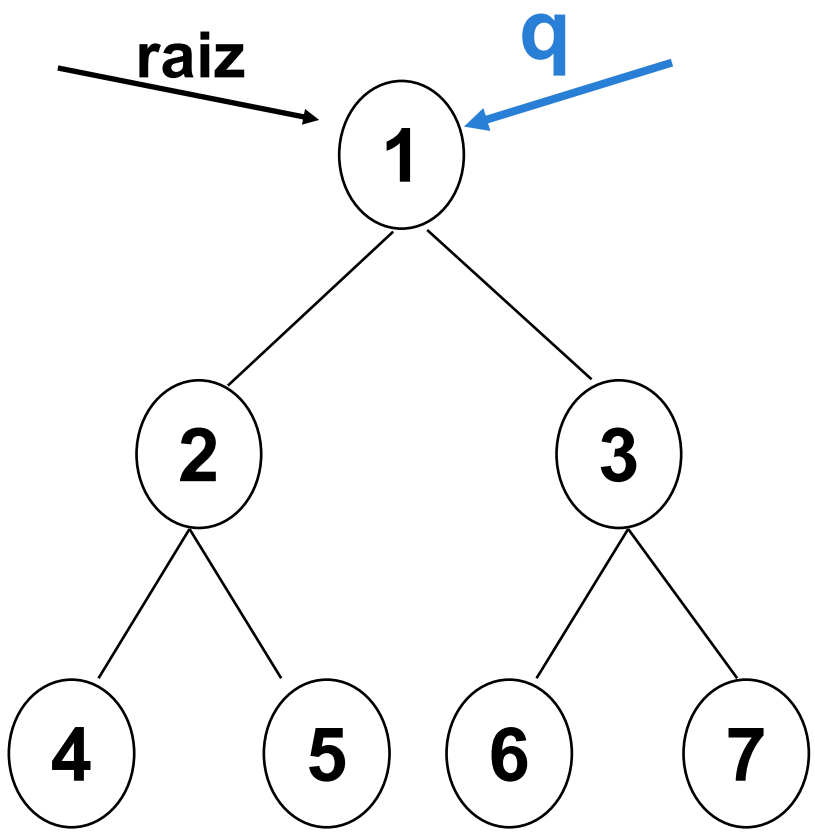
Largura

.Visita é feita por nível, da esquerda para a direita



a - b - c - d - e - f - g

PERCORRER EM LARGURA COM USO DE FILA

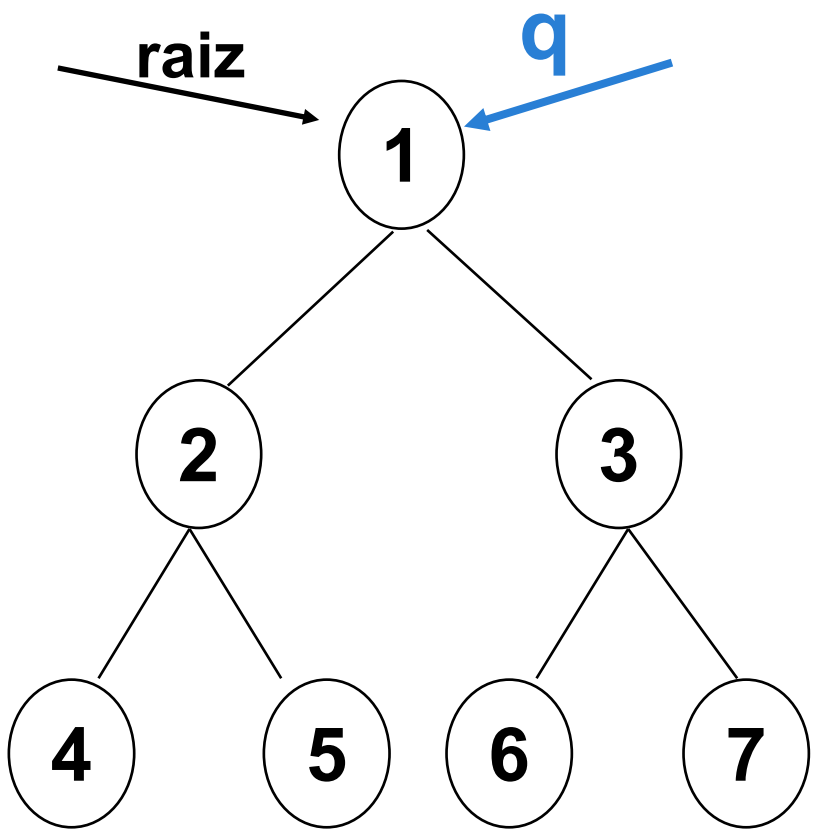


Fila

1

Caminhamento:

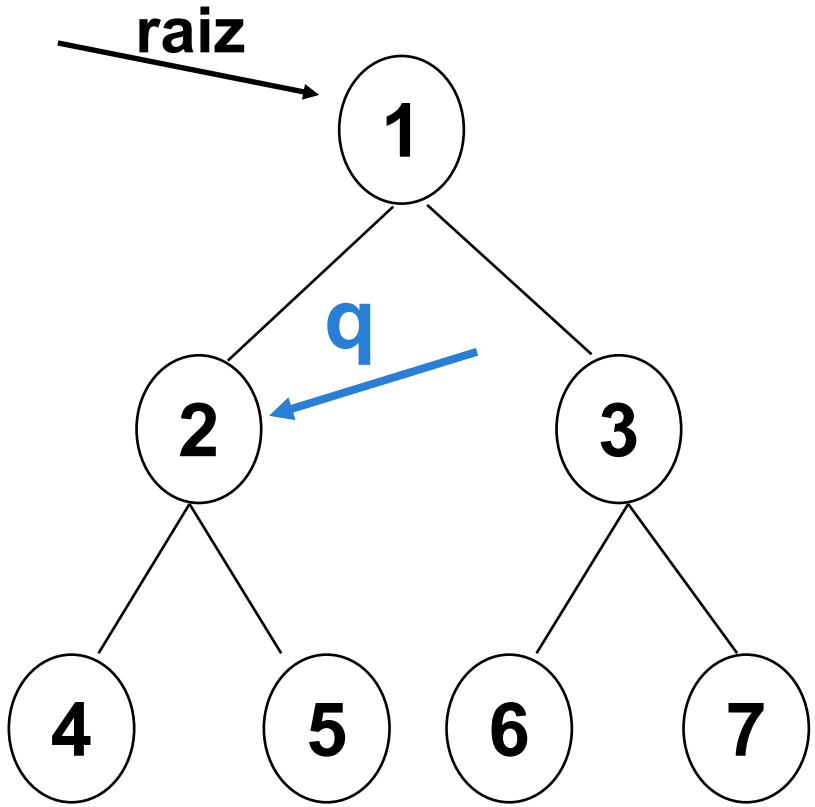
PERCORRER EM LARGURA COM USO DE FILA



Fila

Caminhamento: 1

PERCORRER EM LARGURA COM USO DE FILA

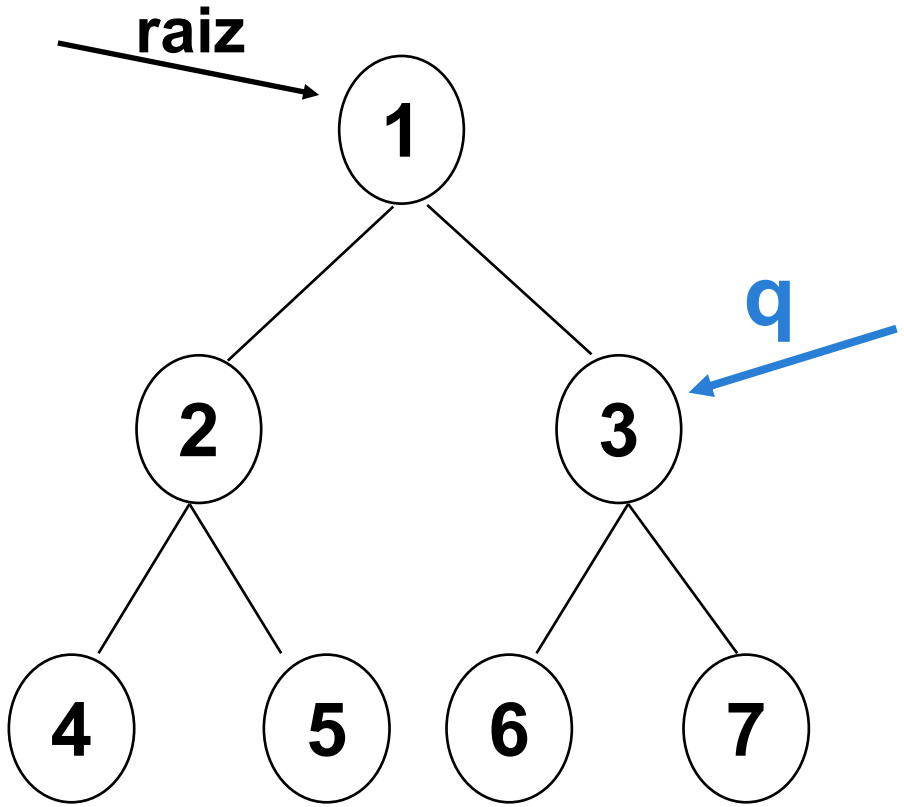


Fila

~~1~~	~~2~~	3	4	5

Caminhamento: 1 – 2

PERCORRER EM LARGURA COM USO DE FILA

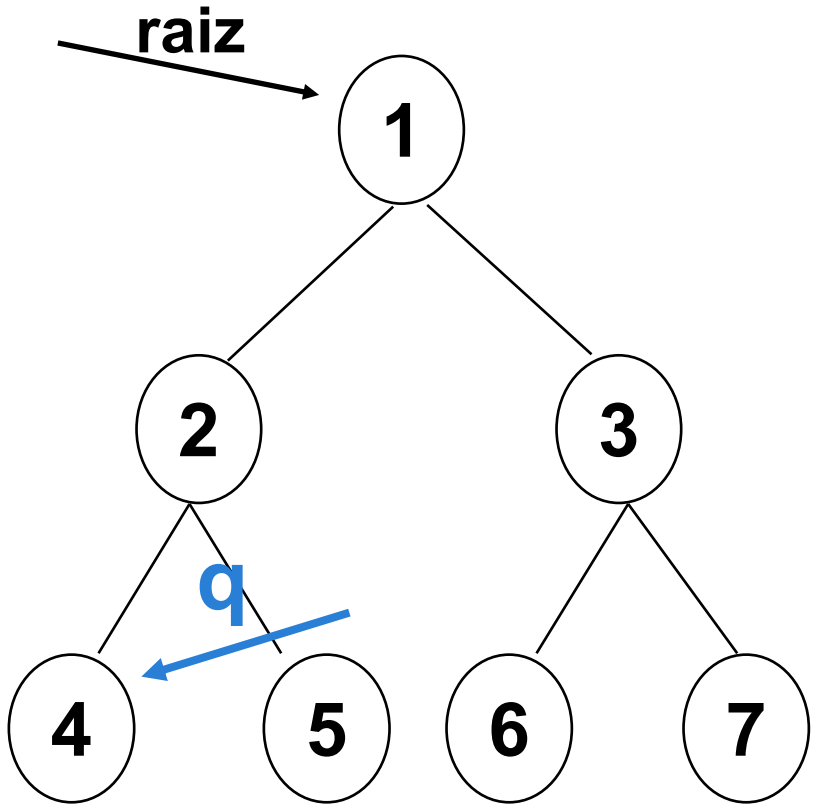


Fila

~~1~~	~~2~~	~~3~~	4	5	6	7

Caminhamento: 1 – 2 – 3

PERCORRER EM LARGURA COM USO DE FILA



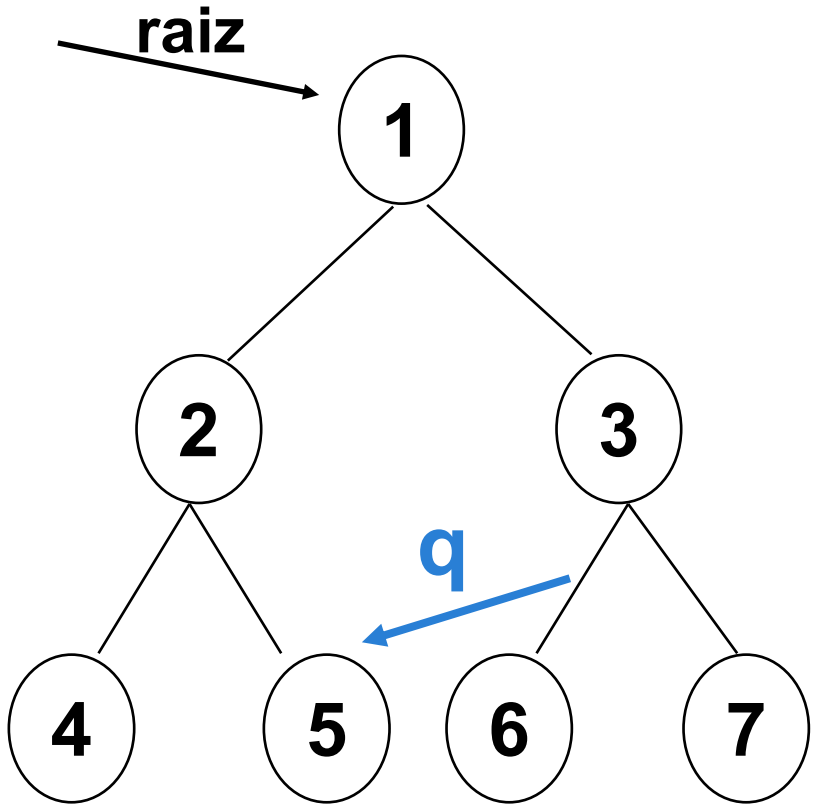
Fila

~~1~~	~~2~~	~~3~~	~~4~~	5	6	7

Filhos NULL não entram na fila

Caminhamento: 1 – 2 – 3 – 4

PERCORRER EM LARGURA COM USO DE FILA



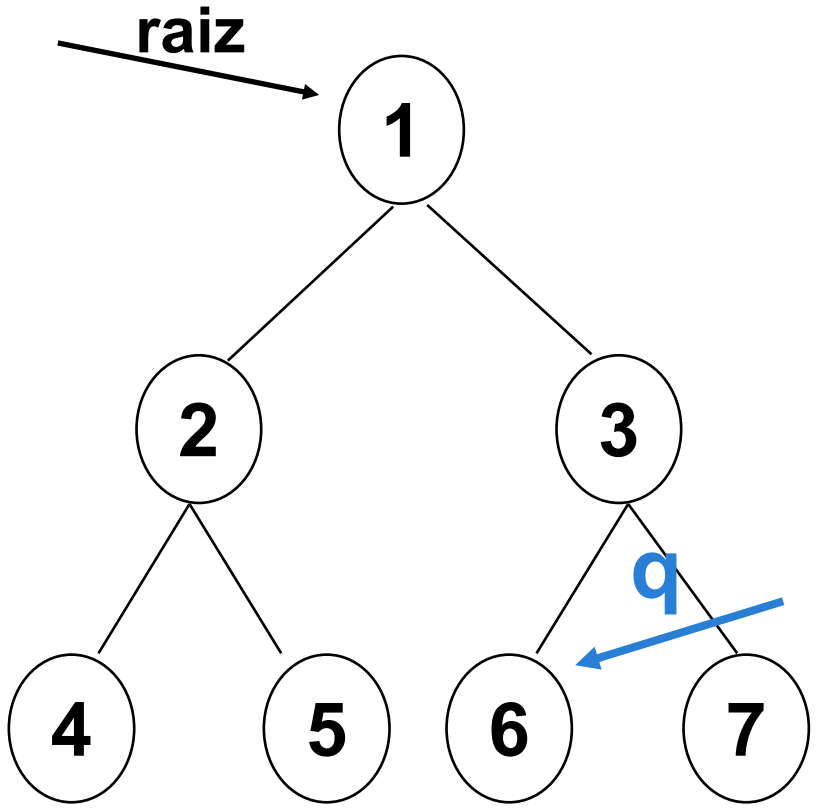
Fila

~~1~~	~~2~~	~~3~~	~~4~~	~~5~~	6	7

Filhos NULL não entram na fila

Caminhamento: 1 – 2 – 3 – 4 – 5

PERCORRER EM LARGURA COM USO DE FILA



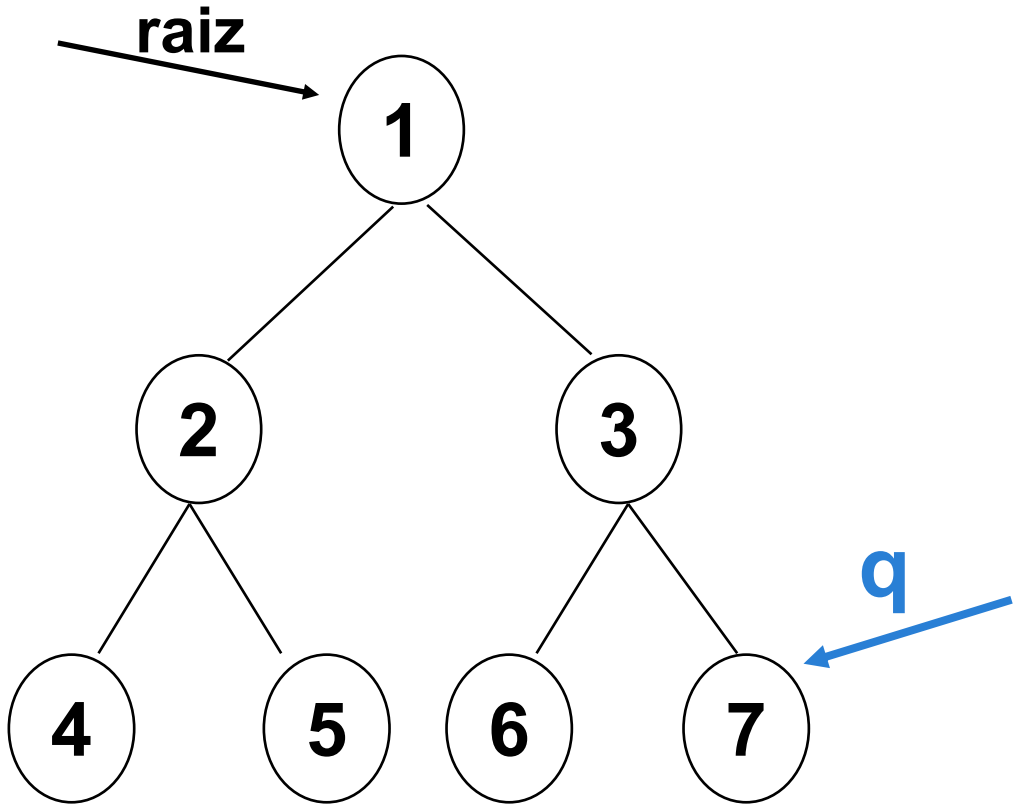
Fila

~~1~~	~~2~~	~~3~~	~~4~~	~~5~~	~~6~~	7

Filhos NULL não entram na fila

Caminhamento: 1 – 2 – 3 – 4 – 5 – 6

PERCORRER EM LARGURA COM USO DE FILA



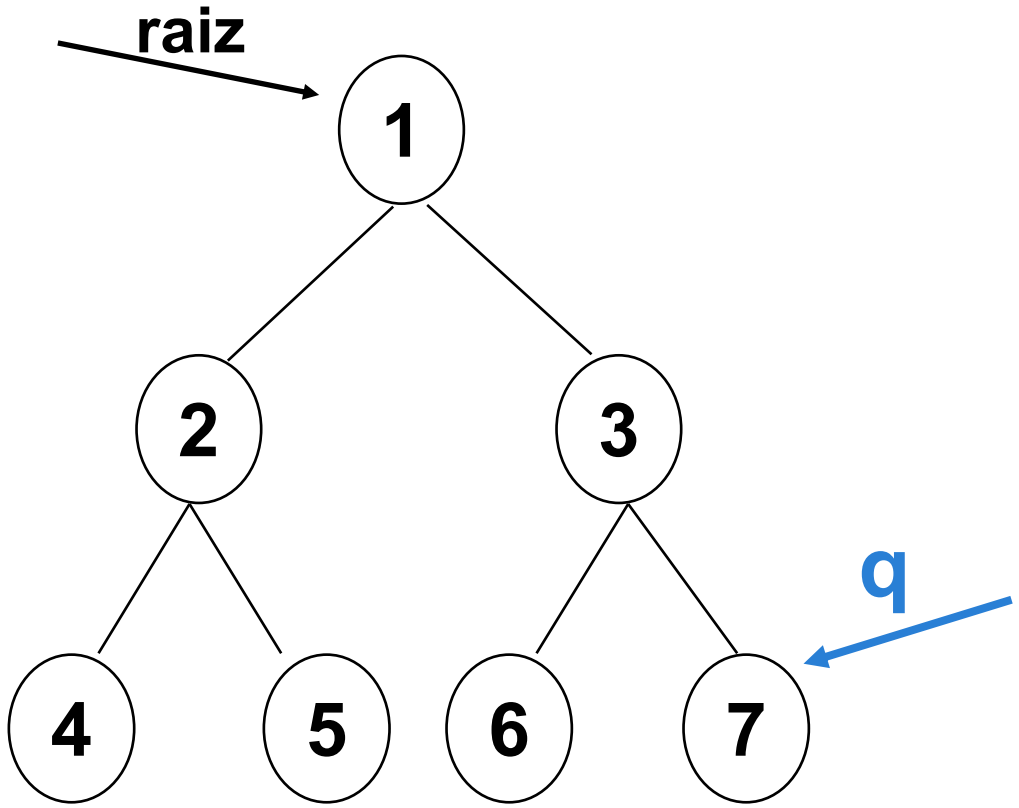
Fila

~~1~~	~~2~~	~~3~~	~~4~~	~~5~~	~~6~~	~~7~~

Filhos NULL não entram na fila

Caminhamento: 1 – 2 – 3 – 4 – 5 – 6 – 7

PERCORRER EM LARGURA COM USO DE FILA



Fila

~~1~~	~~2~~	~~3~~	~~4~~	~~5~~	~~6~~	~~7~~

Fila vazia: fim da execução

Caminhamento: 1 – 2 – 3 – 4 – 5 – 6 – 7

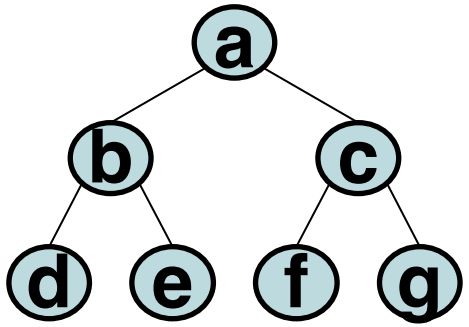
IMPLEMENTAÇÃO: PERCURSO EM PROFUNDIDADE

Estrutura auxiliar necessária: pilha

1. Empilhar a raiz
2. Repetir até que a pilha fique vazia
   1. Desempilha topo da pilha (visita)
   2. Empilha nó da direita (se diferente de NULL)
   3. Empilha nó da esquerda (se diferente de NULL)

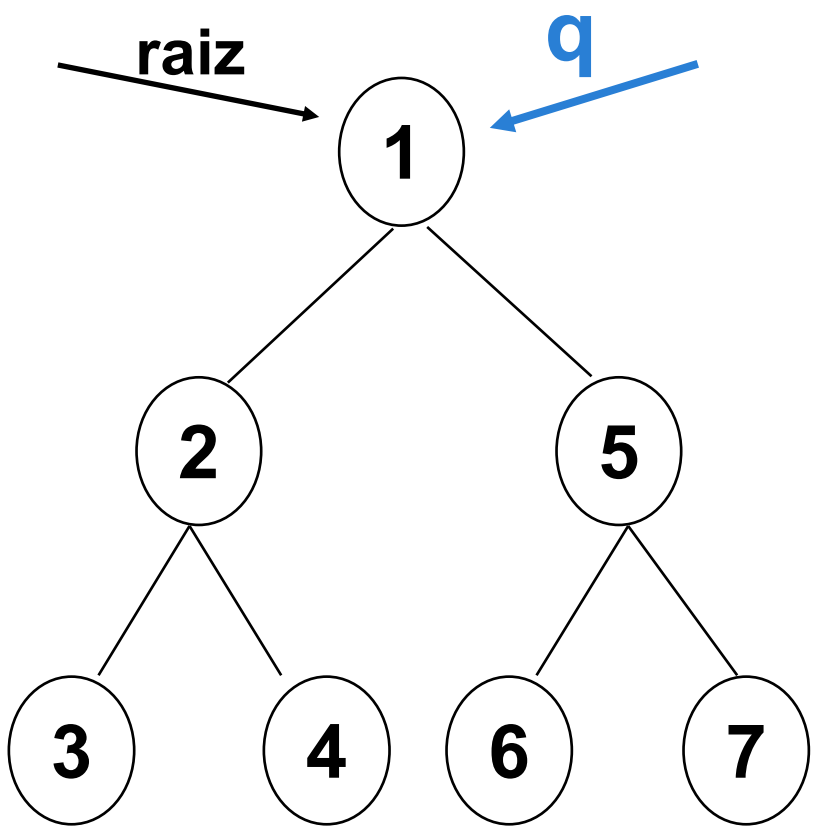
Pré-Ordem (Profundidade)

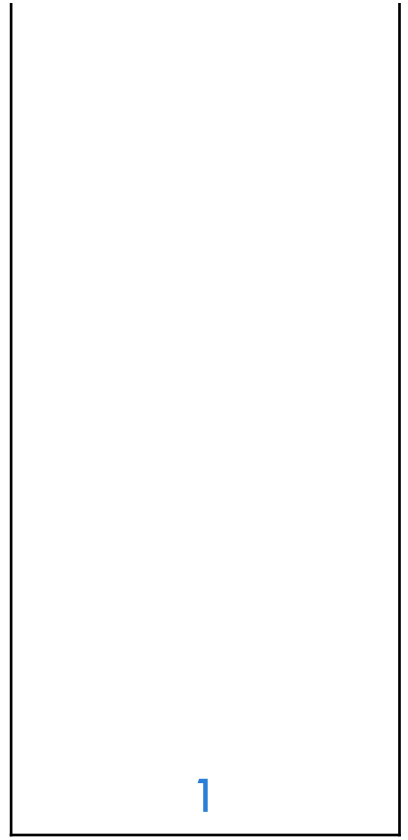
.Visita a raiz
.Percorre a sub-árvore esquerda
.Percorre a sub-árvore direita



a - b - d - e - c - f - g

PERCORRER EM PROFUNDIDADE COM USO DE PILHA

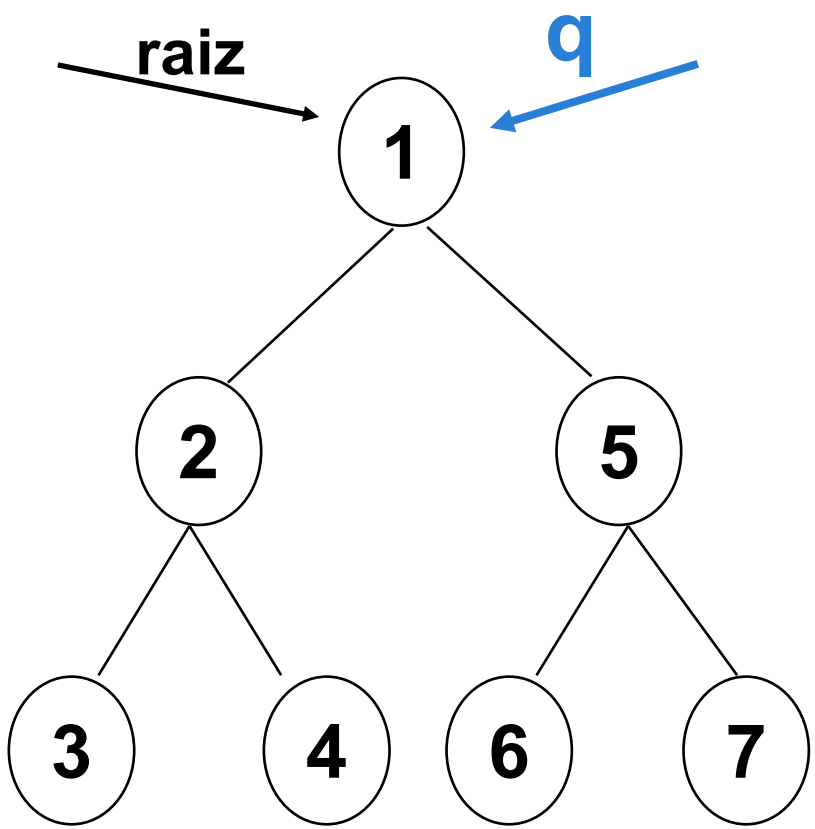


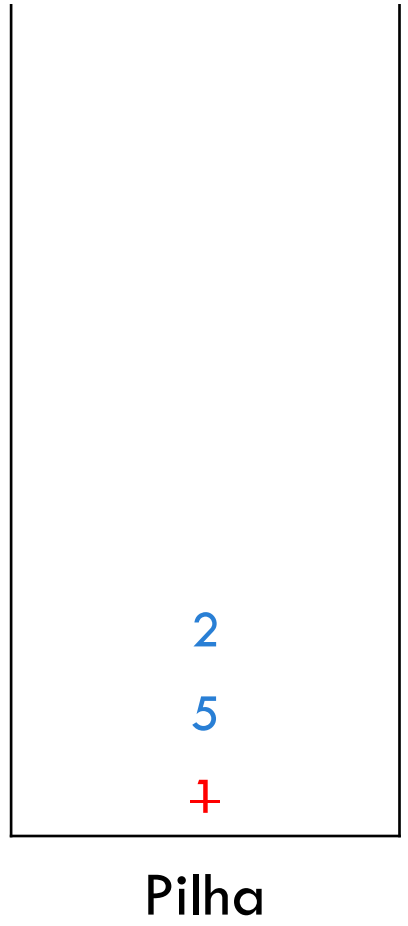


Pilha

Caminhamento:

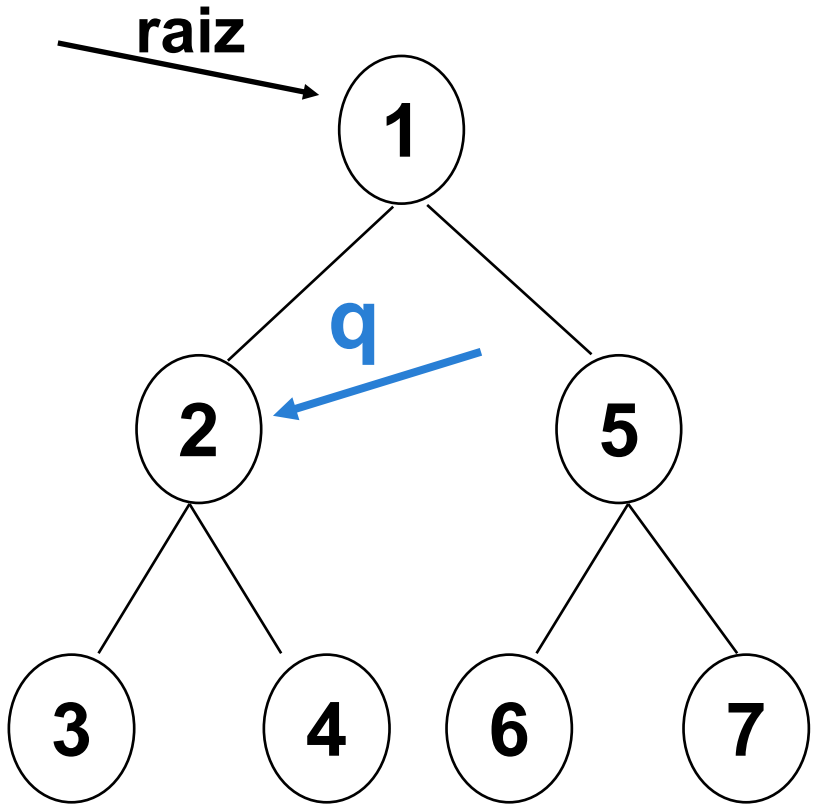
PERCORRER EM PROFUNDIDADE COM USO DE PILHA

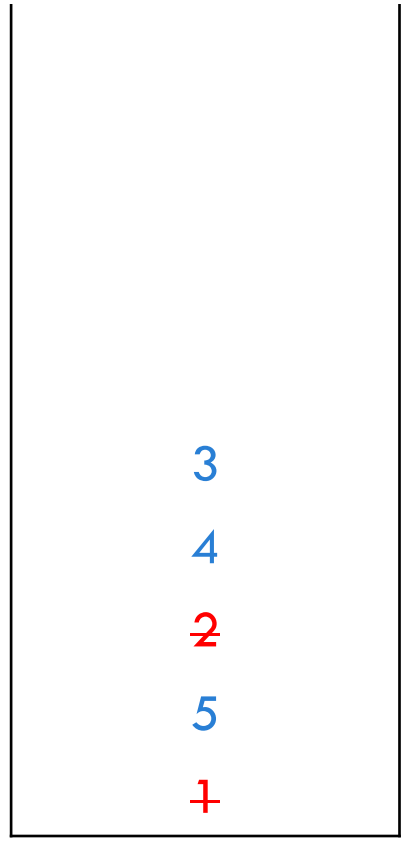




Caminhamento: 1

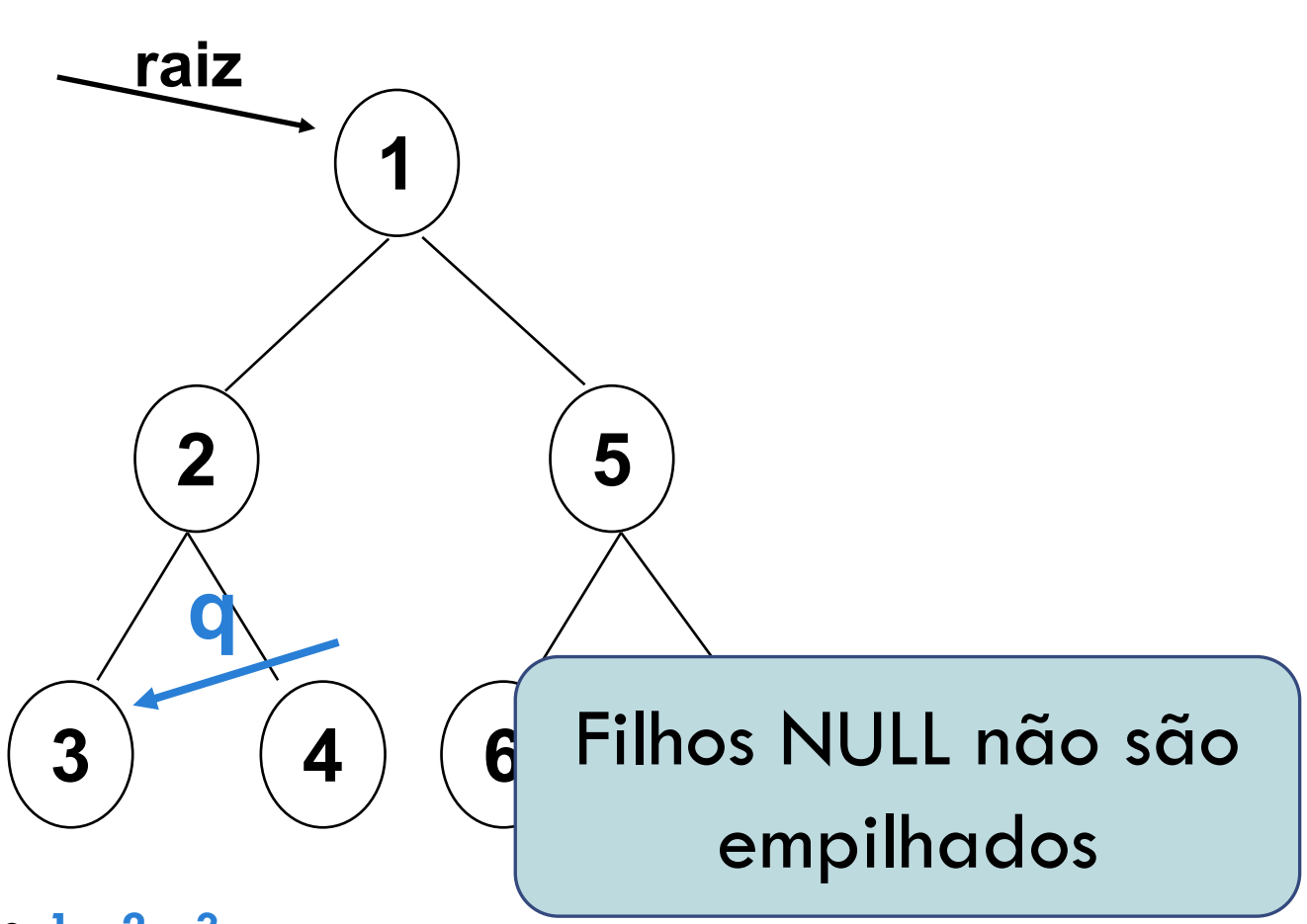
PERCORRER EM PROFUNDIDADE COM USO DE PILHA



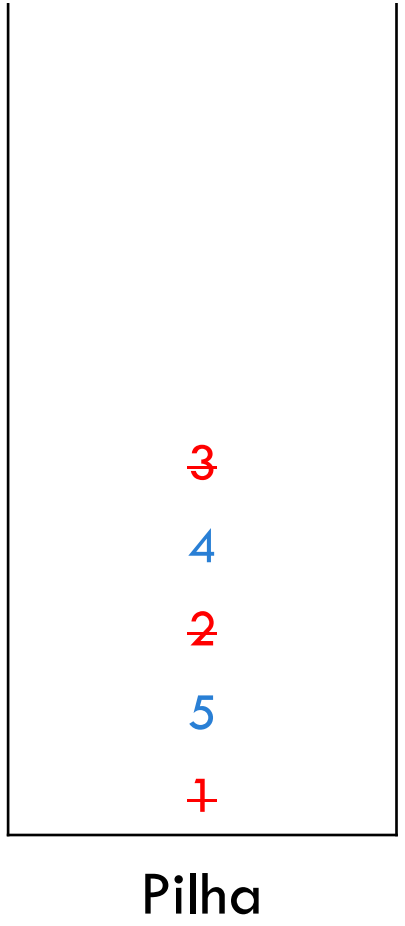


Caminhamento: 1 – 2

PERCORRER EM PROFUNDIDADE COM USO DE PILHA

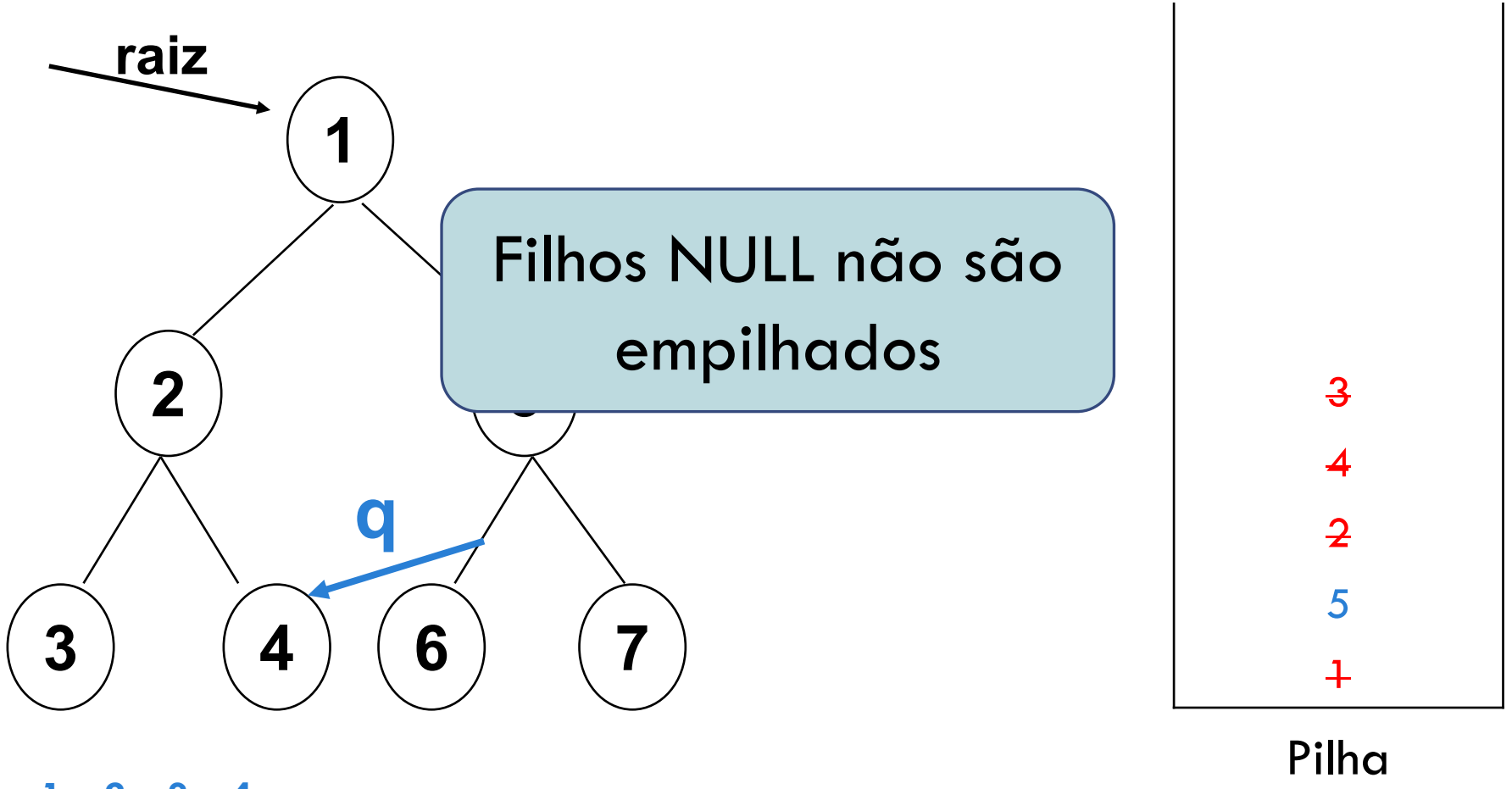


Filhos NULL não são empilhados



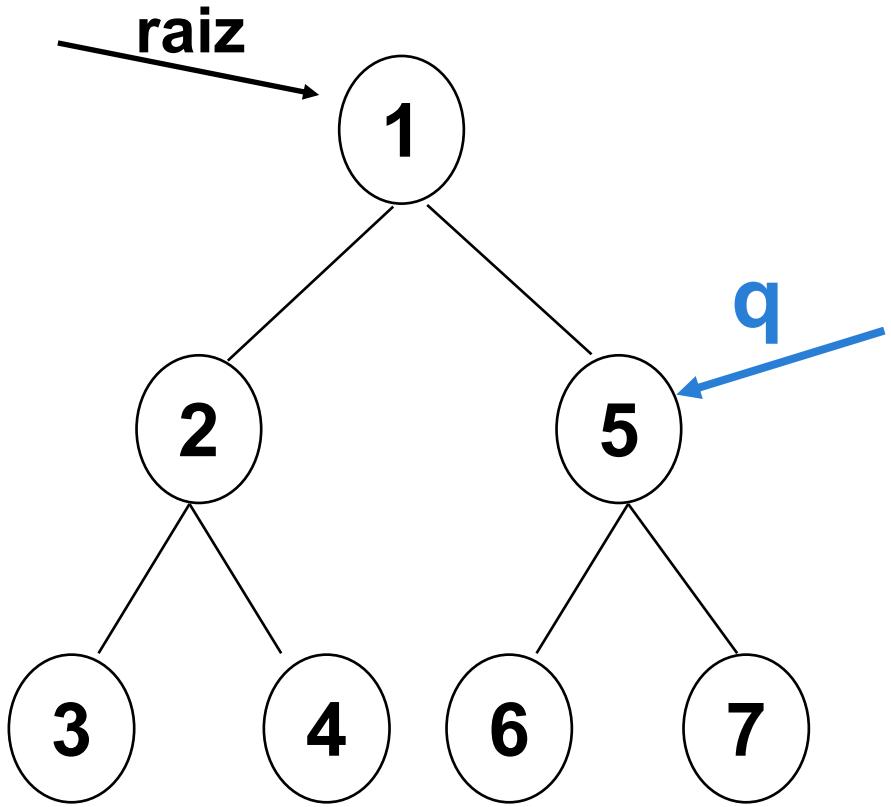
Caminhamento: 1 – 2 – 3

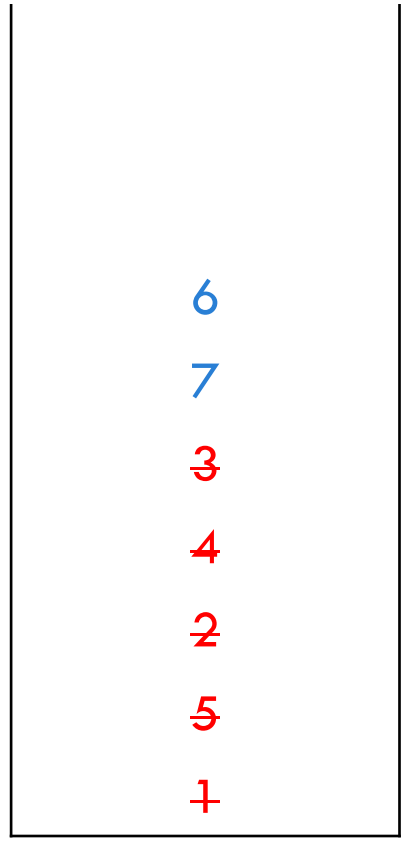
PERCORRER EM PROFUNDIDADE COM USO DE PILHA



Caminhamento: 1 – 2 – 3 – 4

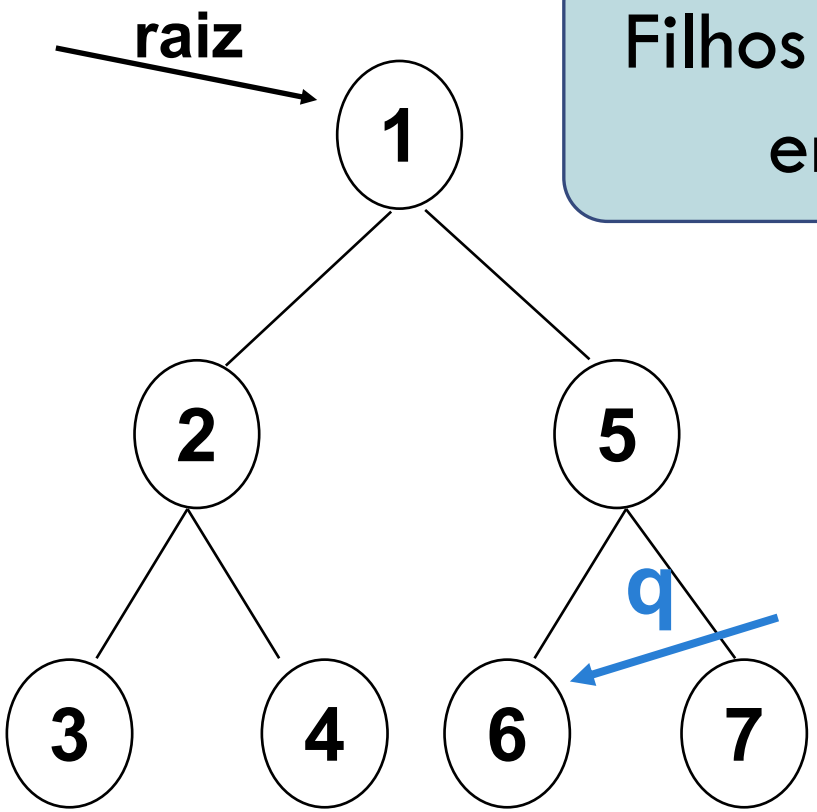
PERCORRER EM PROFUNDIDADE COM USO DE PILHA



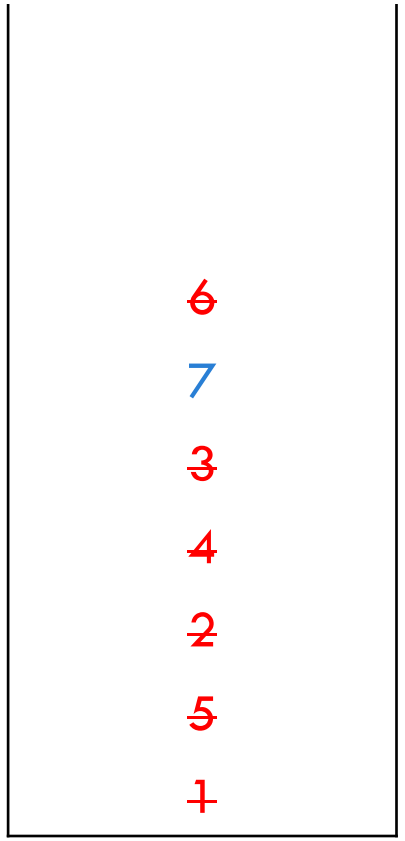


Caminhamento: 1 – 2 – 3 – 4 – 5

PERCORRER EM PROFUNDIDADE COM USO DE PILHA



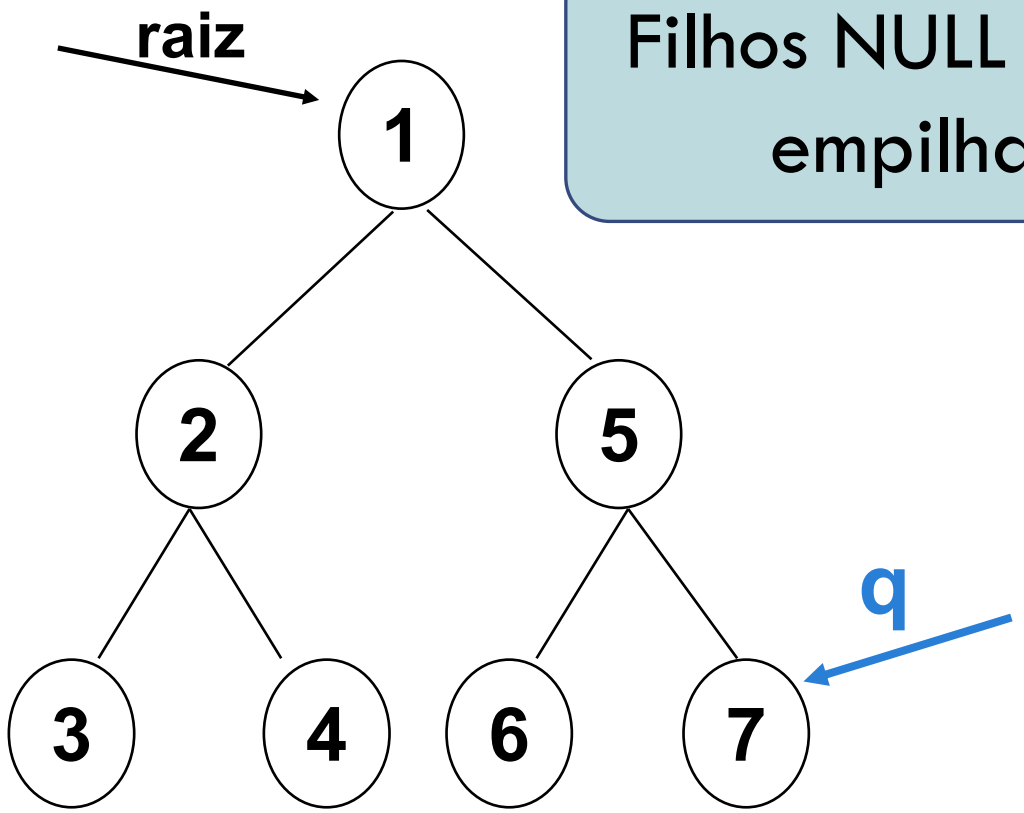
Filhos NULL não são empilhados



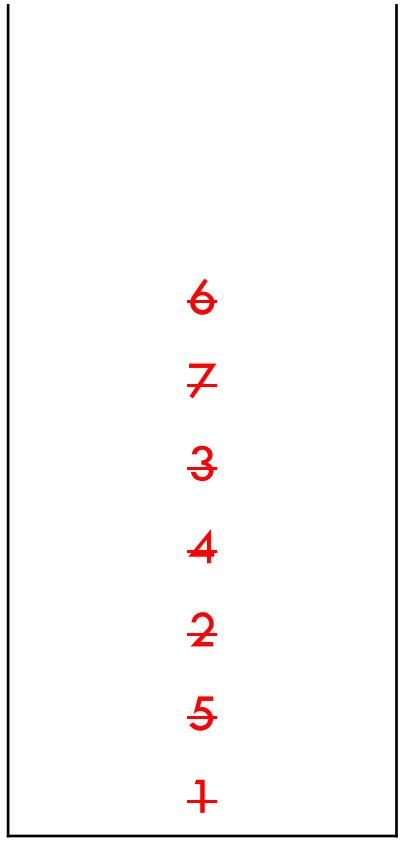
Pilha

Caminhamento: 1 – 2 – 3 – 4 – 5 – 6

PERCORRER EM PROFUNDIDADE COM USO DE PILHA



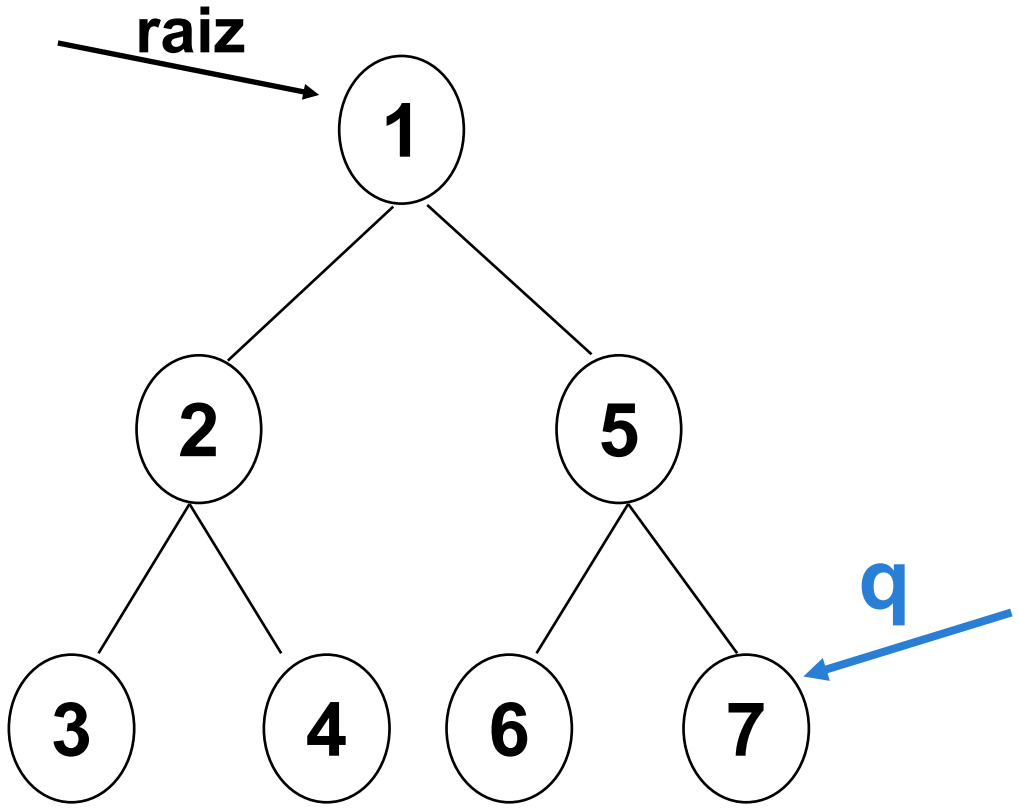
Filhos NULL não são empilhados



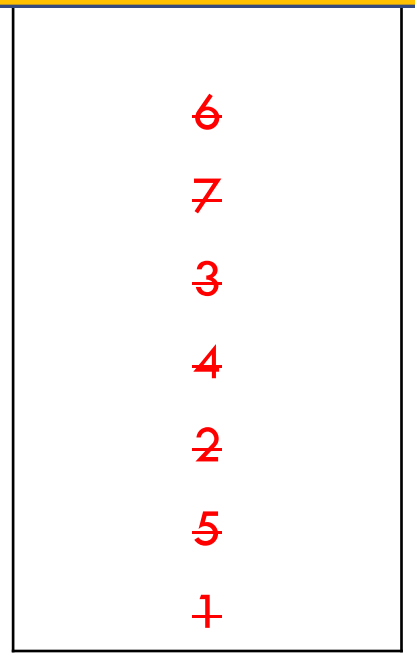
Pilha

Caminhamento: 1 – 2 – 3 – 4 – 5 – 6 – 7

PERCORRER EM PROFUNDIDADE COM USO DE PILHA



Pilha vazia: fim da execução



Pilha

Caminhamento: 1 – 2 – 3 – 4 – 5 – 6 – 7

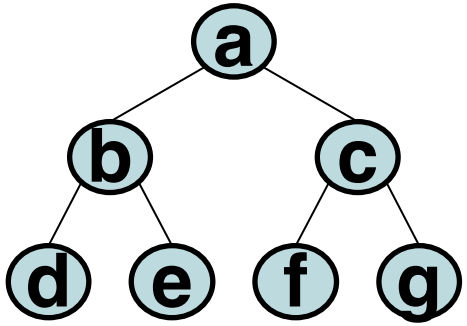
EXERCÍCIOS

1. Escreva uma função que determine se uma árvore binária é cheia ou não.
2. Escreva uma função que cria uma imagem espelho de uma árvore binária, isto é, todos os filhos à esquerda tornam-se filhos à direita, e vice-versa.
3. Ache a raiz de cada uma das seguintes árvores binárias:
   a. Árvore com percurso pós-ordem: FCBDG
   b. Árvore com percurso pré-ordem (profundidade): IBCDFEN
   c. Árvore com percurso em ordem simétrica (assuma que é uma árvore binária cheia): CBIDFGE

EXERCÍCIOS

4. Qual a altura máxima e mínima de uma árvore com 28 nós?
5. Em uma árvore binária, qual é o número máximo de nós que pode ser achado nos níveis 3, 4 e 12?
6. Qual é o menor número de níveis que uma árvore binária com 42 nós pode apresentar?

EXERCÍCIOS



7. Escreva um algoritmo não recursivo para percurso de uma árvore binária em ordem simétrica. Dica: usar uma pilha.
8. Escreva um algoritmo não recursivo para percurso de uma árvore binária em pós-ordem. Dica: usar uma pilha.

Ordem Simétrica

.Percorre a sub-árvore esquerda
.Visita a raiz
.Percorre a sub-árvore direita

d - b - e - a - f - c - g

Pós-Ordem

.Percorre a sub-árvore esquerda
.Percorre a sub-árvore direita
.Visita a raiz

d - e - b - f - g - c - a

AGRADECIMENTOS

Material baseado nos slides de Renata Galante, UFRGS

Exercícios baseados nos slides de Jairo Souza, UFJF