Árvores Binárias

Estrutura de Dados Avançada — QXD0015

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Prof. Atílio Gomes Luiz
gomes.atilio@ufc.br

Universidade Federal do Ceará

1º semestre/2024

Introdução

Representando uma hierarquia

- Vetores, listas, filas e pilhas são estruturas lineares.
  - A importância dessas estruturas é inegável, mas elas não são adequadas para representar dados dispostos de maneira hierárquica.

Representando uma hierarquia

- Vetores, listas, filas e pilhas são estruturas lineares.
  - A importância dessas estruturas é inegável, mas elas não são adequadas para representar dados dispostos de maneira hierárquica.

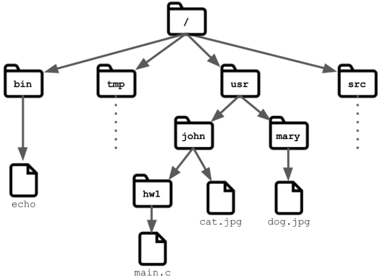


Figura: Hierarquia simplificada do sistema de arquivos de um PC

Representando uma hierarquia

- Vetores, listas, filas e pilhas são estruturas lineares.
  - A importância dessas estruturas é inegável, mas elas não são adequadas para representar dados dispostos de maneira hierárquica.

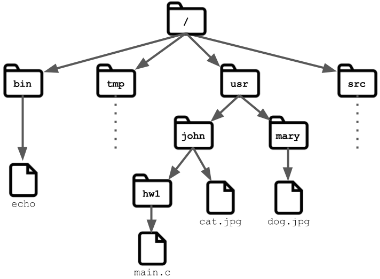


Figura: Hierarquia simplificada do sistema de arquivos de um PC

- As árvores são estruturas de dados mais adequadas para representar hierarquias.

Árvore — Definição Recursiva

Uma árvore T é um conjunto finito de elementos denominados nós, tais que:

Árvore — Definição Recursiva

Uma árvore T é um conjunto finito de elementos denominados nós, tais que:

(a) $T = \emptyset$, e a árvore é dita vazia; ou

Árvore — Definição Recursiva

Uma árvore T é um conjunto finito de elementos denominados nós, tais que:

(a) $T = \emptyset$, e a árvore é dita vazia; ou

(b) $T \neq \emptyset$ e ela possui um nó especial r, chamado raiz de T; os nós restantes constituem um único conjunto vazio ou são divididos em $m \geq 1$ conjuntos disjuntos não vazios, as subárvores de r, cada qual por sua vez um árvore.

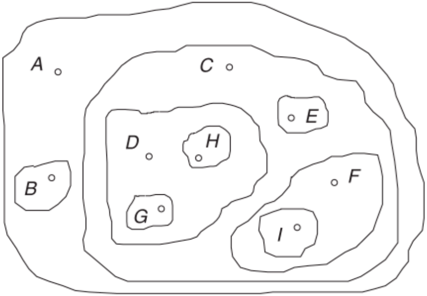
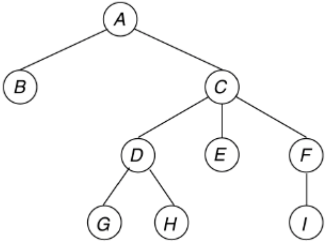


Diagrama de inclusão

Árvore — Outras Representações



Representação hierárquica

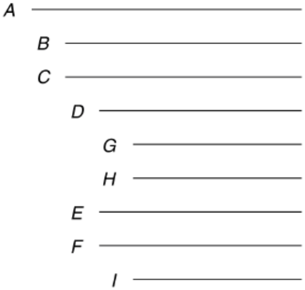
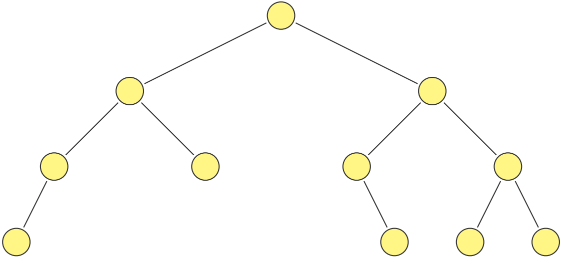


Diagrama de barras

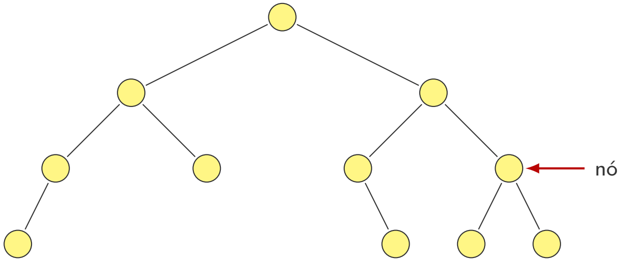
(A (B) (C (D (G) (H)) (E) (F (I))))

Representação por parênteses aninhados

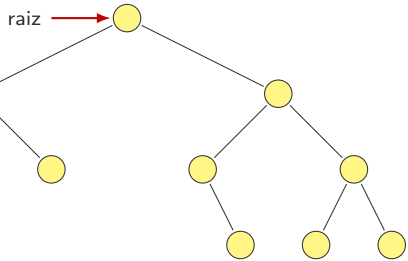
Definições Adicionais



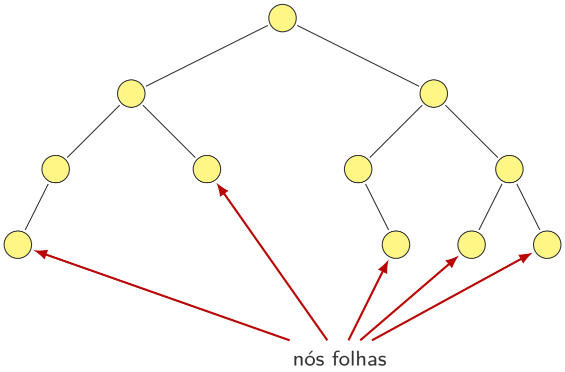
Definições Adicionais



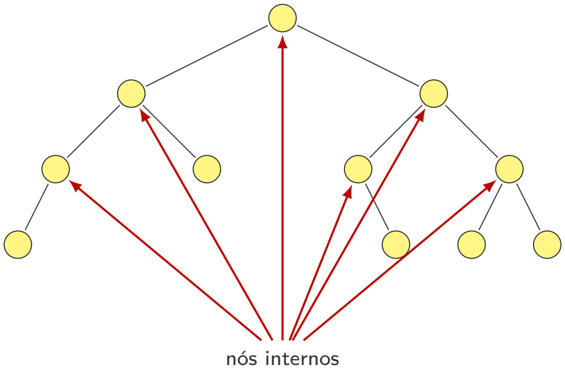
Definições Adicionais



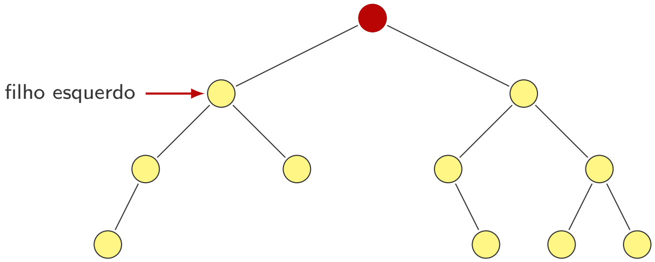
Definições Adicionais



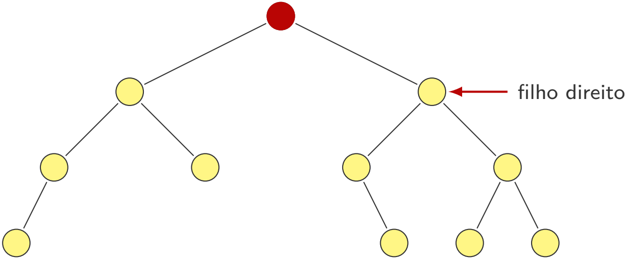
Definições Adicionais



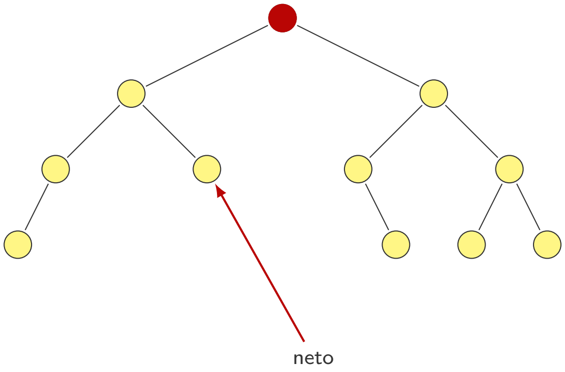
Definições Adicionais



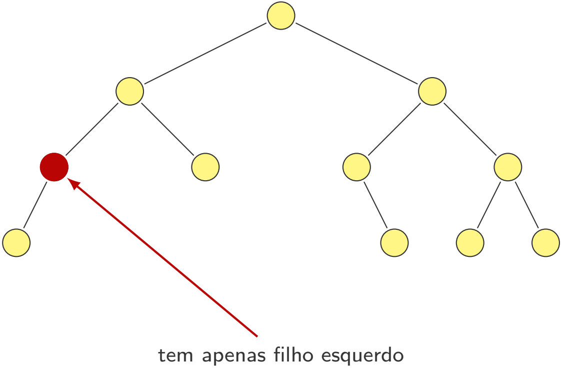
Definições Adicionais



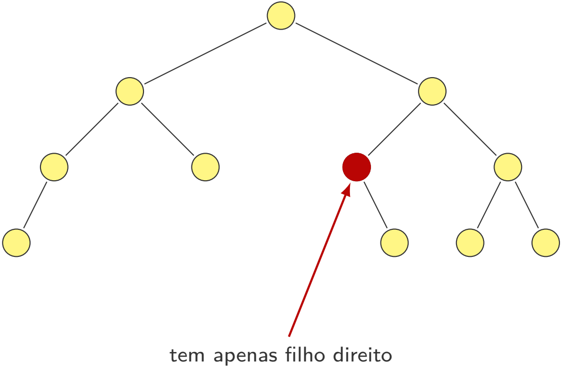
Definições Adicionais



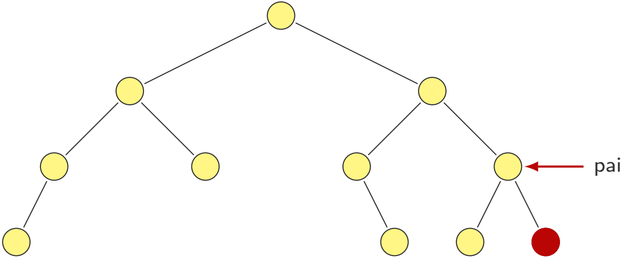
Definições Adicionais



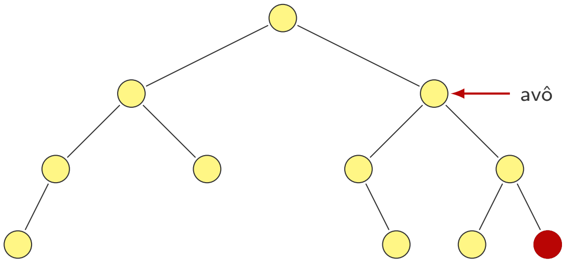
Definições Adicionais



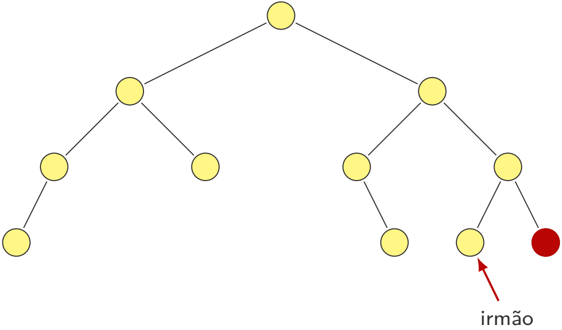
Definições Adicionais



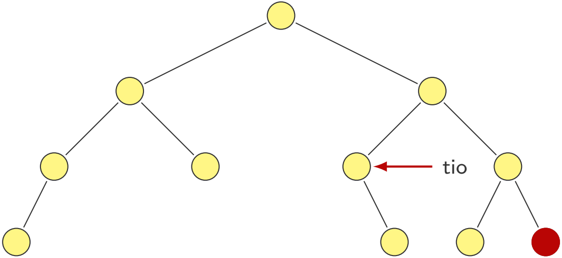
Definições Adicionais



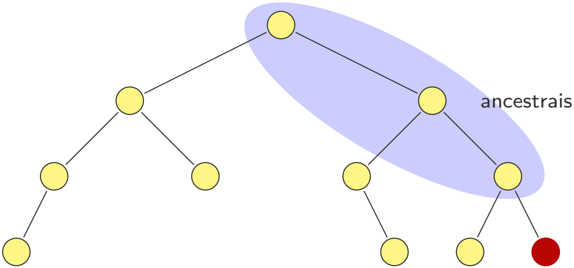
Definições Adicionais



Definições Adicionais

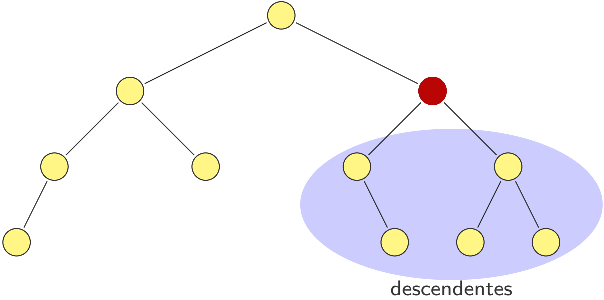


Definições Adicionais

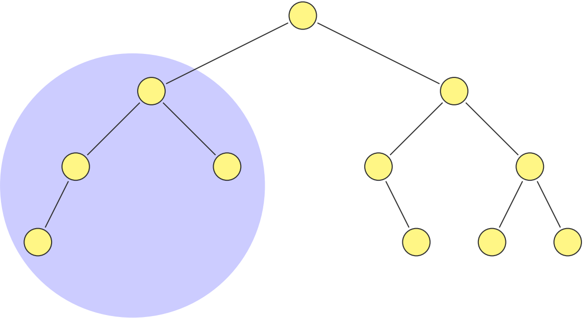


Uma sequência de nós distintos $v_1, v_2, \ldots, v_k$, tal que existe sempre entre nós consecutivos a relação "é filho de" ou "é pai de", é denominada um caminho na árvore.

Definições Adicionais

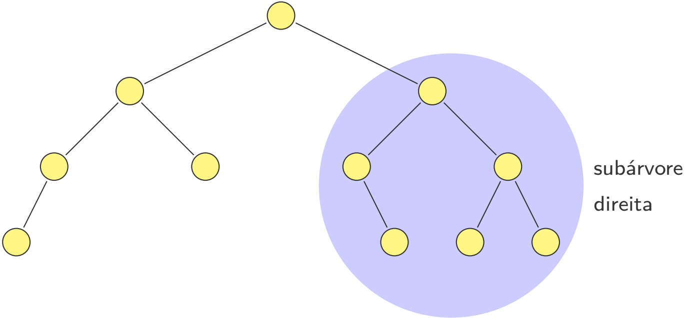


Definições Adicionais

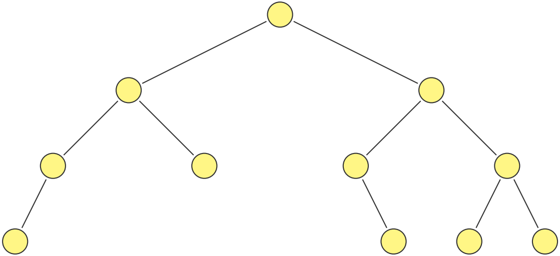


subárvore
esquerda

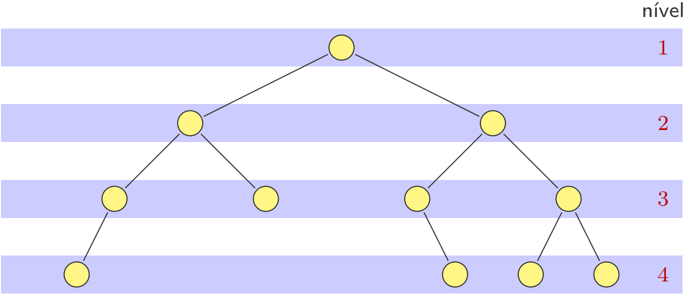
Definições Adicionais



Definições — Profundidade, Nível e Altura

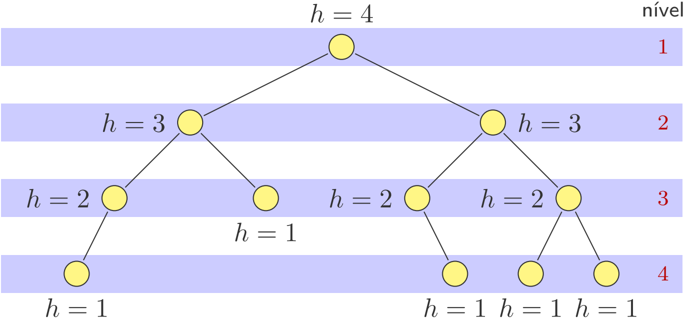


Definições — Profundidade, Nível e Altura



Profundidade de um nó v: Número de nós no caminho de v até a raiz.
Dizemos que todos os nós com profundidade i estão no nível i.

Definições — Profundidade, Nível e Altura

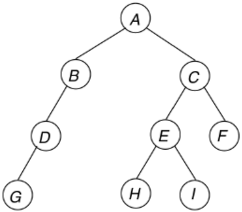


Profundidade de um nó v: Número de nós no caminho de v até a raiz.
Dizemos que todos os nós com profundidade i estão no nível i.

Altura h de um nó v: Número de nós no maior caminho de v até uma folha descendente.

Árvore Binária — Definição Recursiva

- Uma árvore binária T é um conjunto finito de elementos denominados nós, tal que:
  - $T = \emptyset$ e a árvore é dita vazia; ou
  - $T \neq \emptyset$ e existe um nó especial r, chamado raiz de T, e os restantes podem ser divididos em dois subconjuntos disjuntos, T_r^E e T_r^D, a subárvore esquerda e a subárvore direita de r, respectivamente, as quais são também árvores binárias.



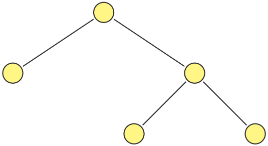
Tipos específicos de árvores binárias

- Árvore estritamente binária: todo nó possui 0 ou 2 filhos.

Tipos específicos de árvores binárias

- Árvore estritamente binária: todo nó possui 0 ou 2 filhos.

- Árvore binária completa: possui a propriedade de que, se v é um nó tal que alguma subárvore de v é vazia, então v se localiza ou no penúltimo ou no último nível da árvore.

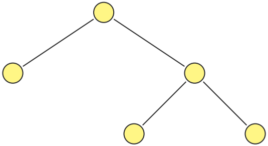


binária completa

Tipos específicos de árvores binárias

- Árvore estritamente binária: todo nó possui 0 ou 2 filhos.

- Árvore binária completa: possui a propriedade de que, se v é um nó tal que alguma subárvore de v é vazia, então v se localiza ou no penúltimo ou no último nível da árvore.

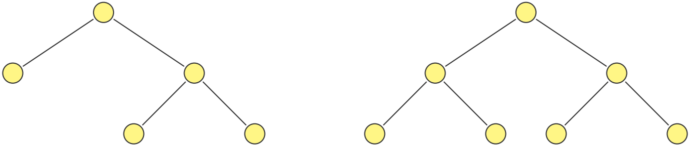


binária completa

Tipos específicos de árvores binárias

- Árvore estritamente binária: todo nó possui 0 ou 2 filhos.

- Árvore binária completa: possui a propriedade de que, se v é um nó tal que alguma subárvore de v é vazia, então v se localiza ou no penúltimo ou no último nível da árvore.



binária completa   binária cheia

- Árvore binária cheia: todos os seus nós internos têm dois filhos e todas as folhas estão no último nível da árvore.

Sequência de Teoremas

Altura da árvore binária completa

Teorema 1 (Jayme e Lilian)

Seja T uma árvore binária completa com $n > 0$ nós.
Então T possui altura mínima.

Altura da árvore binária completa

Teorema 1 (Jayme e Lilian)

Seja T uma árvore binária completa com $n > 0$ nós.
Então T possui altura mínima.

Prova: Seja T é uma árvore binária completa com n nós, e seja T' uma árvore binária de altura mínima com n nós.

Altura da árvore binária completa

Teorema 1 (Jayme e Lilian)

Seja T uma árvore binária completa com $n > 0$ nós.
Então T possui altura mínima.

Prova: Seja T é uma árvore binária completa com n nós, e seja T' uma árvore binária de altura mínima com n nós.

Caso 1: Se T' é também completa, então T e T' possuem a mesma altura, o que implica que T possui altura mínima.

Altura da árvore binária completa

Teorema 1 (Jayme e Lilian)

Seja T uma árvore binária completa com $n > 0$ nós.
Então T possui altura mínima.

Prova: Seja T é uma árvore binária completa com n nós, e seja T' uma árvore binária de altura mínima com n nós.

Caso 1: Se T' é também completa, então T e T' possuem a mesma altura, o que implica que T possui altura mínima.

Caso 2: Se T' não é completa, efetua-se a seguinte operação: retirar uma folha w de seu último nível e tornar w o filho de algum nó v que possui alguma de suas subárvores vazias, localizado em algum nível acima do penúltimo.

Altura da árvore binária completa

Teorema 1 (Jayme e Lilian)

Seja T uma árvore binária completa com $n > 0$ nós.
Então T possui altura mínima.

Prova: Seja T é uma árvore binária completa com n nós, e seja T' uma árvore binária de altura mínima com n nós.

Caso 1: Se T' é também completa, então T e T' possuem a mesma altura, o que implica que T possui altura mínima.

Caso 2: Se T' não é completa, efetua-se a seguinte operação: retirar uma folha w de seu último nível e tornar w o filho de algum nó v que possui alguma de suas subárvores vazias, localizado em algum nível acima do penúltimo.

Repete-se a operação até que não seja mais possível realizá-la, isto é, até que a árvore T'', resultante da transformação, seja completa.

Altura da árvore binária completa

Continuação da prova do Teorema 1:

T'' não pode ter altura inferior a T', pois T' é mínima.

Altura da árvore binária completa

Continuação da prova do Teorema 1:

T'' não pode ter altura inferior a T', pois T' é mínima.

T'' não pode ter altura superior a T', pois nenhum nó foi movido para baixo.

Altura da árvore binária completa

Continuação da prova do Teorema 1:

T'' não pode ter altura inferior a T', pois T' é mínima.

T'' não pode ter altura superior a T', pois nenhum nó foi movido para baixo.

Então as alturas de T' e T'' são iguais. Ou seja, T'' tem altura mínima.

Altura da árvore binária completa

Continuação da prova do Teorema 1:

T'' não pode ter altura inferior a T', pois T' é mínima.

T'' não pode ter altura superior a T', pois nenhum nó foi movido para baixo.

Então as alturas de T' e T'' são iguais. Ou seja, T'' tem altura mínima.

Como T'' é completa, conclui-se que as alturas de T e T'' também coincidem. Isto é, T possui altura mínima. $\square$

Número mínimo e máximo de nós

Teorema 2

Dada uma árvore binária completa com altura h, seu número mínimo de nós é 2^{h-1} e seu número máximo de nós é $2^h - 1$.

Exercício: Provar este teorema. Dica: note que, numa árvore binária completa com altura h, os nós nos primeiros $h - 1$ níveis formam uma árvore cheia e no último nível há k nós adicionais, tal que $1 \leq k \leq 2^{h-1}$.

Altura da árvore binária completa

Teorema 3

Se T é uma árvore binária completa com $n > 0$ nós, então T possui altura $h = \lfloor \lg n \rfloor + 1$.

Altura da árvore binária completa

Teorema 3

Se T é uma árvore binária completa com $n > 0$ nós, então T possui altura $h = \lfloor \lg n \rfloor + 1$.

Prova: Seja T uma árvore binária completa com $n > 0$ nós.
Pelo Teorema 2, temos que:

$$\begin{aligned} 2^{h-1} &\leq n \leq 2^h - 1 \\ 2^{h-1} &\leq n < 2^h \\ \lg 2^{h-1} &\leq \lg n < \lg 2^h \\ h - 1 &\leq \lg n < h \end{aligned}$$

Isso implica que $\lfloor \lg n \rfloor = h - 1$, o que nos dá $h = \lfloor \lg n \rfloor + 1$. ■

Relação entre altura e número de nós

Seja T uma árvore com altura h qualquer fixa.

- Pergunta: Quantos nós T tem no mínimo (em função de h)?

Relação entre altura e número de nós

Seja T uma árvore com altura h qualquer fixa.

- Pergunta: Quantos nós T tem no mínimo (em função de h)?
  - **Resposta:** T tem no mínimo h nós.
    Neste caso, ela T é uma árvore caminho.

Relação entre altura e número de nós

Seja T uma árvore com altura h qualquer fixa.

- Pergunta: Quantos nós T tem no mínimo (em função de h)?
  - **Resposta:** T tem no mínimo h nós.
    Neste caso, ela T é uma árvore caminho.
- Pergunta: Quantos nós T tem no máximo (em função de h)?

Relação entre altura e número de nós

Seja T uma árvore com altura h qualquer fixa.

- Pergunta: Quantos nós T tem no mínimo (em função de h)?
  - **Resposta:** T tem no mínimo h nós.
    Neste caso, ela T é uma árvore caminho.
- Pergunta: Quantos nós T tem no máximo (em função de h)?
  - **Resposta:** T tem no máximo $2^h - 1$ nós.
    Neste caso, T é uma árvore cheia.

Relação entre número de nós e altura (cont.)

Seja T uma árvore com número n de nós fixo.

- Pergunta: Qual a altura máxima de T (em função de n)?

Relação entre número de nós e altura (cont.)

Seja T uma árvore com número n de nós fixo.

- Pergunta: Qual a altura máxima de T (em função de n)?
  - **Resposta:** T tem no máximo n.
    Neste caso, ela T é uma árvore caminho.

Relação entre número de nós e altura (cont.)

Seja T uma árvore com número n de nós fixo.

- Pergunta: Qual a altura máxima de T (em função de n)?
  - **Resposta:** T tem no máximo n.
    Neste caso, ela T é uma árvore caminho.
- Pergunta: Qual a altura mínima de T (em função de n)?

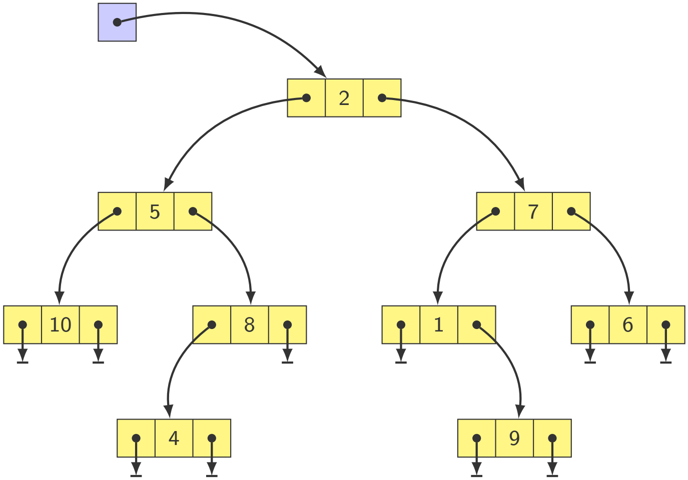
Relação entre número de nós e altura (cont.)

Seja T uma árvore com número n de nós fixo.

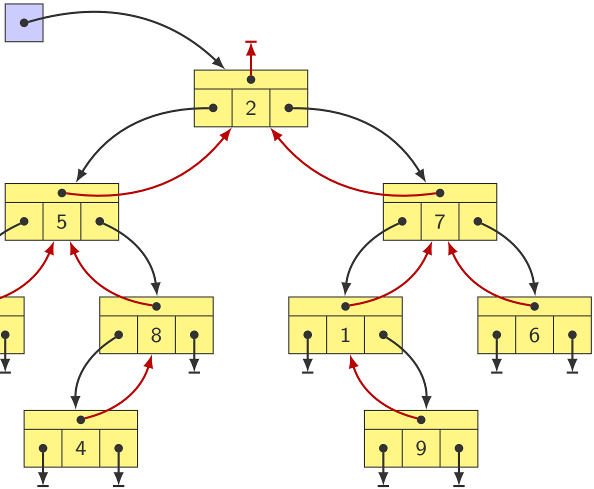
- Pergunta: Qual a altura máxima de T (em função de n)?
  - **Resposta:** T tem no máximo n.
    Neste caso, ela T é uma árvore caminho.
- Pergunta: Qual a altura mínima de T (em função de n)?
  - **Resposta:** T altura no mínimo $\lfloor \lg n \rfloor + 1$.
    Neste caso, T é uma árvore completa ou pode ser transformada numa.

Representação no Computador

Representação no Computador



Representação com ponteiro para pai



Representação — Decisões de projeto

- Em programação de computadores, os nós de uma árvore binária são definidos como um tipo de dado composto contendo pelo menos três ou quatro atributos:
    - um valor (chave a ser guardada)
    - um ponteiro para o filho esquerdo do nó
    - um ponteiro para o filho direito do nó
    - um ponteiro para o pai (obrigatório em algumas implementações)
- Para acessarmos qualquer nó da árvore, basta termos o endereço do nó raiz. Pois podemos usar recursão para fazer todo o trabalho. Portanto, a única informação inicial necessária é um ponteiro para a raiz da árvore.

Implementação do Nó da Árvore em C++

```cpp
// Arquivo Node.h
#ifndef NODE_H
#define NODE_H

struct Node
{
    int key;
    Node* left;
    Node* right;
};

#endif /* NODE_H */
```

Percursos em Árvores Binárias

Percursos em Árvores Binárias

- Muitas operações em árvores binárias envolvem o percurso de todas as subárvores, com execução de alguma ação de tratamento em cada nó.

Percursos em Árvores Binárias

- Muitas operações em árvores binárias envolvem o percurso de todas as subárvores, com execução de alguma ação de tratamento em cada nó.

- É comum percorrer uma árvore em uma das seguintes ordens:
  - pré-ordem:
    - **visita raiz**, percorre `r->left`, percorre `r->right`

Percursos em Árvores Binárias

- Muitas operações em árvores binárias envolvem o percurso de todas as subárvores, com execução de alguma ação de tratamento em cada nó.
- É comum percorrer uma árvore em uma das seguintes ordens:
  - pré-ordem:
    - **visita raiz**, percorre `r->left`, percorre `r->right`
  - ordem simétrica:
    - percorre `r->left`, **visita raiz**, percorre `r->right`

Percursos em Árvores Binárias

- Muitas operações em árvores binárias envolvem o percurso de todas as subárvores, com execução de alguma ação de tratamento em cada nó.

- É comum percorrer uma árvore em uma das seguintes ordens:
  - pré-ordem:
    - **visita raiz**, percorre `r->left`, percorre `r->right`
  - ordem simétrica:
    - percorre `r->left`, **visita raiz**, percorre `r->right`
  - pós-ordem:
    - percorre `r->left`, percorre `r->right`, **visita raiz**

Percorrendo os nós — Pré-ordem

A pré-ordem

Percorrendo os nós — Pré-ordem

A pré-ordem

- primeiro visita (processa) a raiz

Percorrendo os nós — Pré-ordem

A pré-ordem

- primeiro visita (processa) a raiz
- depois a subárvore esquerda

Percorrendo os nós — Pré-ordem

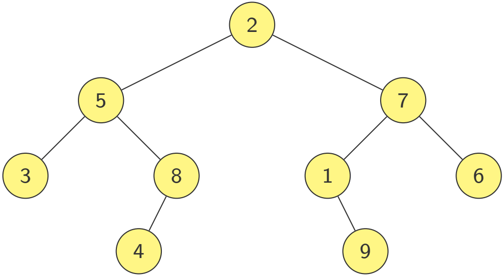
A pré-ordem

- primeiro visita (processa) a raiz
- depois a subárvore esquerda
- depois a subárvore direita

Percorrendo os nós — Pré-ordem

A pré-ordem

- primeiro visita (processa) a raiz
- depois a subárvore esquerda
- depois a subárvore direita

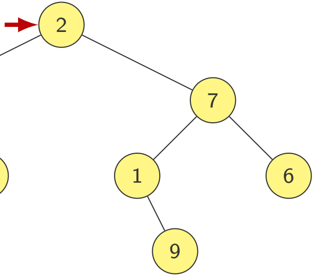


Ex:

Percorrendo os nós — Pré-ordem

A pré-ordem

- primeiro visita (processa) a raiz
- depois a subárvore esquerda
- depois a subárvore direita

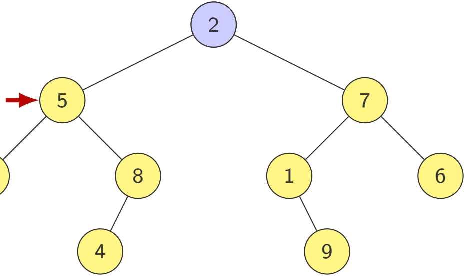


Ex:

Percorrendo os nós — Pré-ordem

A pré-ordem

- primeiro visita (processa) a raiz
- depois a subárvore esquerda
- depois a subárvore direita

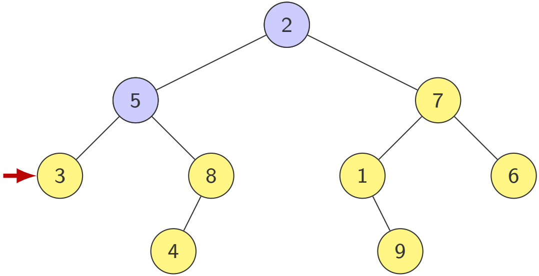


Ex: 2,

Percorrendo os nós — Pré-ordem

A pré-ordem

- primeiro visita (processa) a raiz
- depois a subárvore esquerda
- depois a subárvore direita

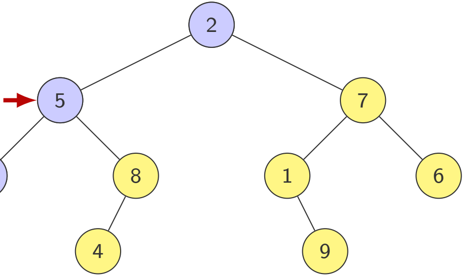


Ex: 2, 5,

Percorrendo os nós — Pré-ordem

A pré-ordem

- primeiro visita (processa) a raiz
- depois a subárvore esquerda
- depois a subárvore direita

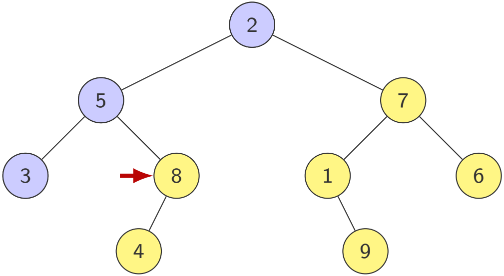


Ex: 2, 5, 3,

Percorrendo os nós — Pré-ordem

A pré-ordem

- primeiro visita (processa) a raiz
- depois a subárvore esquerda
- depois a subárvore direita

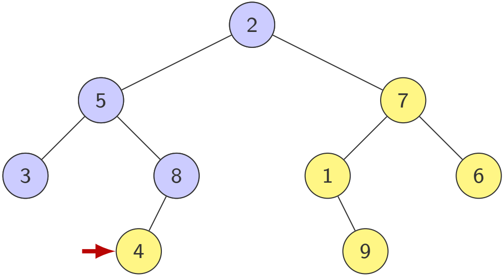


Ex: 2, 5, 3,

Percorrendo os nós — Pré-ordem

A pré-ordem

- primeiro visita (processa) a raiz
- depois a subárvore esquerda
- depois a subárvore direita

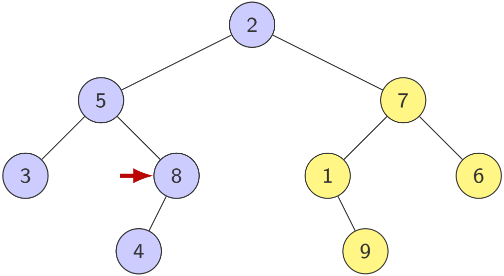


Ex: 2, 5, 3, 8,

Percorrendo os nós — Pré-ordem

A pré-ordem

- primeiro visita (processa) a raiz
- depois a subárvore esquerda
- depois a subárvore direita

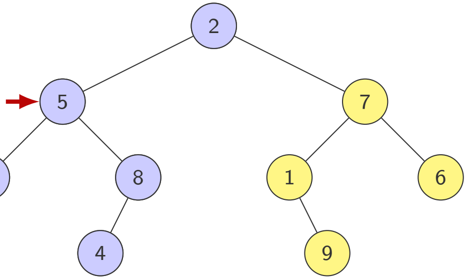


Ex: 2, 5, 3, 8, 4,

Percorrendo os nós — Pré-ordem

A pré-ordem

- primeiro visita (processa) a raiz
- depois a subárvore esquerda
- depois a subárvore direita

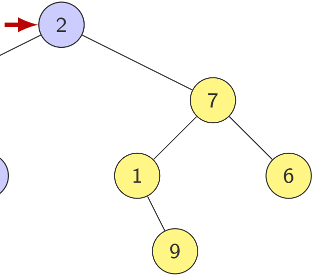


Ex: 2, 5, 3, 8, 4,

Percorrendo os nós — Pré-ordem

A pré-ordem

- primeiro visita (processa) a raiz
- depois a subárvore esquerda
- depois a subárvore direita

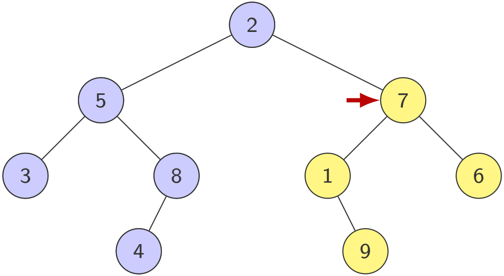


Ex: 2, 5, 3, 8, 4,

Percorrendo os nós — Pré-ordem

A pré-ordem

- primeiro visita (processa) a raiz
- depois a subárvore esquerda
- depois a subárvore direita

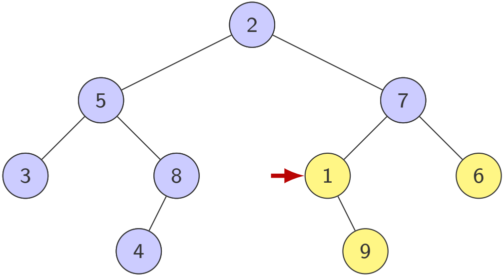


Ex: 2, 5, 3, 8, 4,

Percorrendo os nós — Pré-ordem

A pré-ordem

- primeiro visita (processa) a raiz
- depois a subárvore esquerda
- depois a subárvore direita

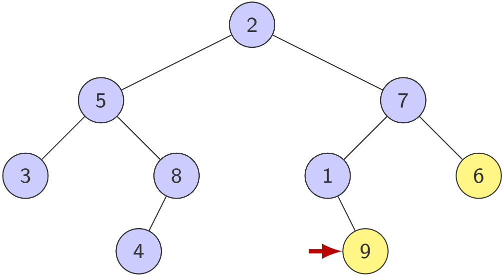


Ex: 2, 5, 3, 8, 4, 7,

Percorrendo os nós — Pré-ordem

A pré-ordem

- primeiro visita (processa) a raiz
- depois a subárvore esquerda
- depois a subárvore direita

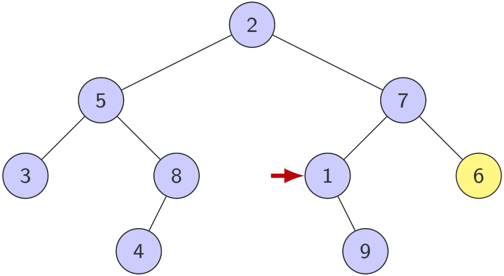


Ex: 2, 5, 3, 8, 4, 7, 1,

Percorrendo os nós — Pré-ordem

A pré-ordem

- primeiro visita (processa) a raiz
- depois a subárvore esquerda
- depois a subárvore direita

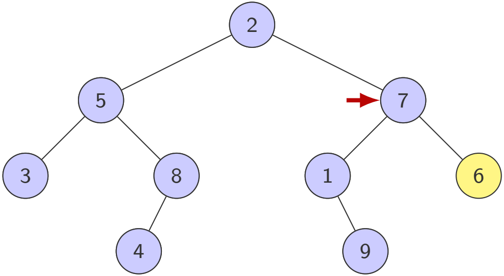


Ex: 2, 5, 3, 8, 4, 7, 1, 9,

Percorrendo os nós — Pré-ordem

A pré-ordem

- primeiro visita (processa) a raiz
- depois a subárvore esquerda
- depois a subárvore direita

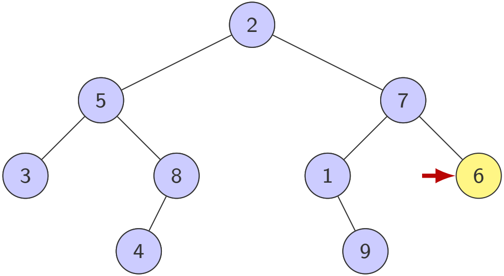


Ex: 2, 5, 3, 8, 4, 7, 1, 9,

Percorrendo os nós — Pré-ordem

A pré-ordem

- primeiro visita (processa) a raiz
- depois a subárvore esquerda
- depois a subárvore direita

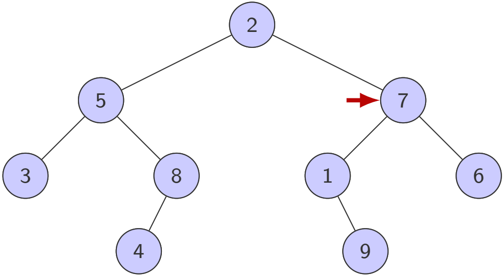


Ex: 2, 5, 3, 8, 4, 7, 1, 9,

Percorrendo os nós — Pré-ordem

A pré-ordem

- primeiro visita (processa) a raiz
- depois a subárvore esquerda
- depois a subárvore direita

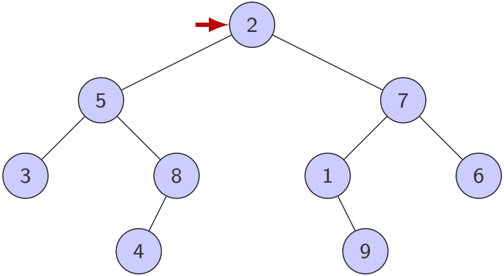


Ex: 2, 5, 3, 8, 4, 7, 1, 9, 6

Percorrendo os nós — Pré-ordem

A pré-ordem

- primeiro visita (processa) a raiz
- depois a subárvore esquerda
- depois a subárvore direita



Ex: 2, 5, 3, 8, 4, 7, 1, 9, 6

Pré-ordem

Algorithm preorder(ptr)

Require: ptr (pointer to node)

```
if ptr ≠ NULL then
    visit(ptr)
    preorder(ptr→left)
    preorder(ptr→right)
end if
```

Percorrendo os nós — Pós-ordem

A pós-ordem

Percorrendo os nós — Pós-ordem

A pós-ordem

- primeiro visita a subárvore esquerda

Percorrendo os nós — Pós-ordem

A pós-ordem

- primeiro visita a subárvore esquerda
- depois a subárvore direita

Percorrendo os nós — Pós-ordem

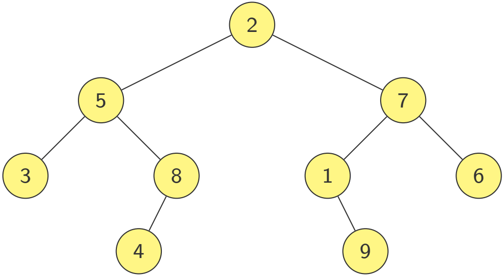
A pós-ordem

- primeiro visita a subárvore esquerda
- depois a subárvore direita
- e por último visita a raiz

Percorrendo os nós — Pós-ordem

A pós-ordem

- primeiro visita a subárvore esquerda
- depois a subárvore direita
- e por último visita a raiz

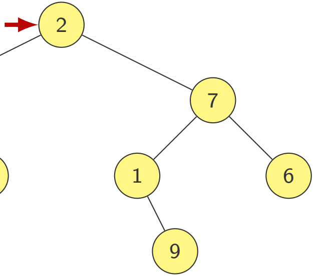


Ex:

Percorrendo os nós — Pós-ordem

A pós-ordem

- primeiro visita a subárvore esquerda
- depois a subárvore direita
- e por último visita a raiz

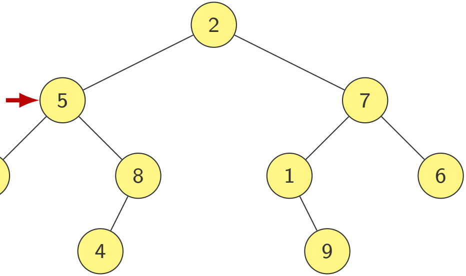


Ex:

Percorrendo os nós — Pós-ordem

A pós-ordem

- primeiro visita a subárvore esquerda
- depois a subárvore direita
- e por último visita a raiz

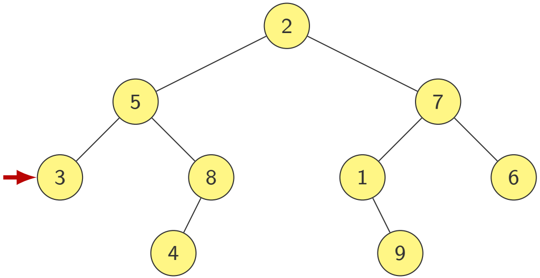


Ex:

Percorrendo os nós — Pós-ordem

A pós-ordem

- primeiro visita a subárvore esquerda
- depois a subárvore direita
- e por último visita a raiz

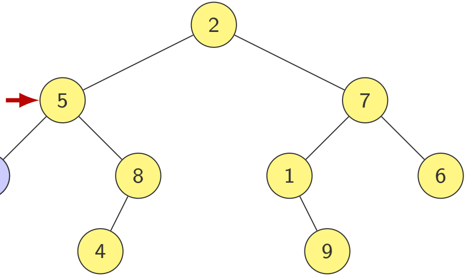


Ex:

Percorrendo os nós — Pós-ordem

A pós-ordem

- primeiro visita a subárvore esquerda
- depois a subárvore direita
- e por último visita a raiz

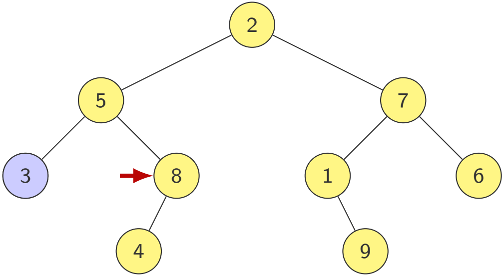


Ex: 3,

Percorrendo os nós — Pós-ordem

A pós-ordem

- primeiro visita a subárvore esquerda
- depois a subárvore direita
- e por último visita a raiz

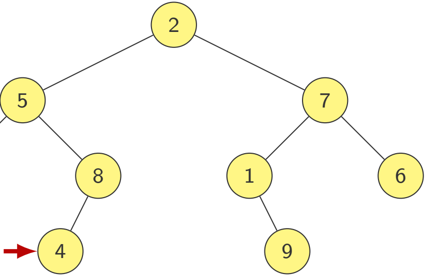


Ex: 3,

Percorrendo os nós — Pós-ordem

A pós-ordem

- primeiro visita a subárvore esquerda
- depois a subárvore direita
- e por último visita a raiz

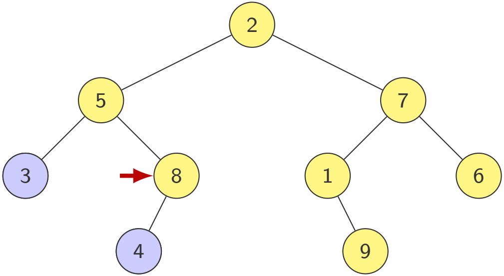


Ex: 3,

Percorrendo os nós — Pós-ordem

A pós-ordem

- primeiro visita a subárvore esquerda
- depois a subárvore direita
- e por último visita a raiz

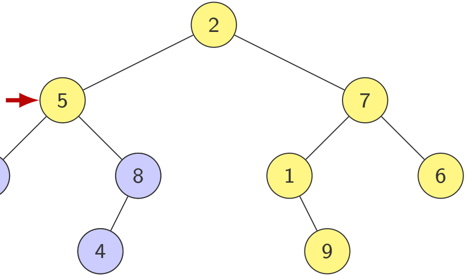


Ex: 3, 4,

Percorrendo os nós — Pós-ordem

A pós-ordem

- primeiro visita a subárvore esquerda
- depois a subárvore direita
- e por último visita a raiz

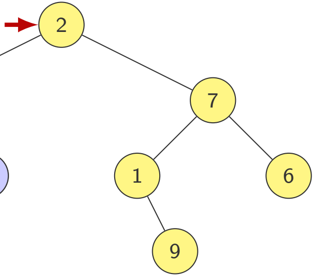


Ex: 3, 4, 8,

Percorrendo os nós — Pós-ordem

A pós-ordem

- primeiro visita a subárvore esquerda
- depois a subárvore direita
- e por último visita a raiz

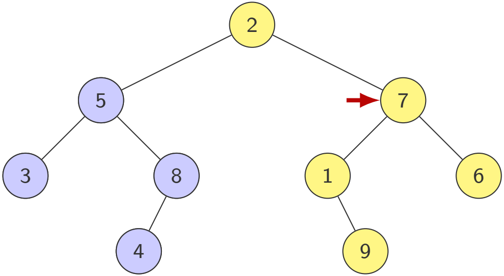


Ex: 3, 4, 8, 5,

Percorrendo os nós — Pós-ordem

A pós-ordem

- primeiro visita a subárvore esquerda
- depois a subárvore direita
- e por último visita a raiz

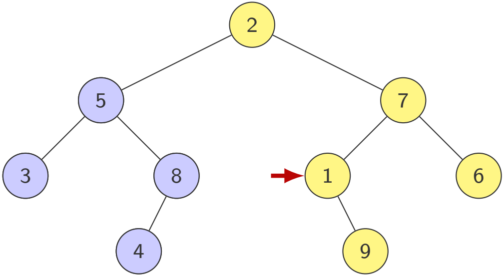


Ex: 3, 4, 8, 5,

Percorrendo os nós — Pós-ordem

A pós-ordem

- primeiro visita a subárvore esquerda
- depois a subárvore direita
- e por último visita a raiz

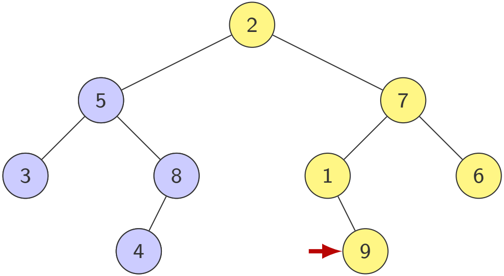


Ex: 3, 4, 8, 5,

Percorrendo os nós — Pós-ordem

A pós-ordem

- primeiro visita a subárvore esquerda
- depois a subárvore direita
- e por último visita a raiz

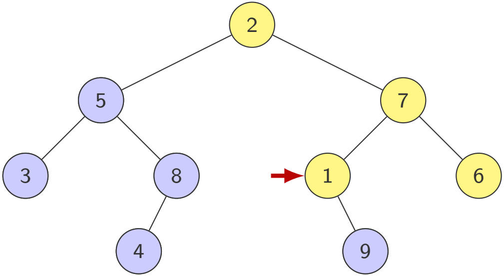


Ex: 3, 4, 8, 5,

Percorrendo os nós — Pós-ordem

A pós-ordem

- primeiro visita a subárvore esquerda
- depois a subárvore direita
- e por último visita a raiz

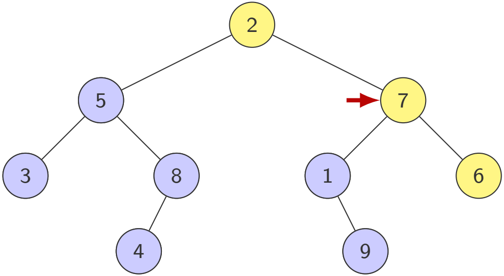


Ex: 3, 4, 8, 5, 9,

Percorrendo os nós — Pós-ordem

A pós-ordem

- primeiro visita a subárvore esquerda
- depois a subárvore direita
- e por último visita a raiz

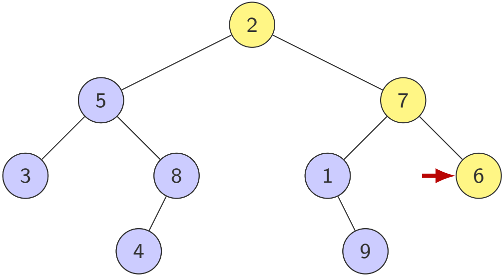


Ex: 3, 4, 8, 5, 9, 1,

Percorrendo os nós — Pós-ordem

A pós-ordem

- primeiro visita a subárvore esquerda
- depois a subárvore direita
- e por último visita a raiz

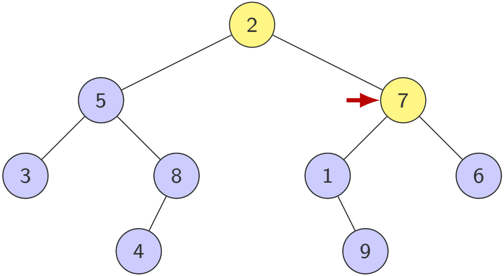


Ex: 3, 4, 8, 5, 9, 1,

Percorrendo os nós — Pós-ordem

A pós-ordem

- primeiro visita a subárvore esquerda
- depois a subárvore direita
- e por último visita a raiz

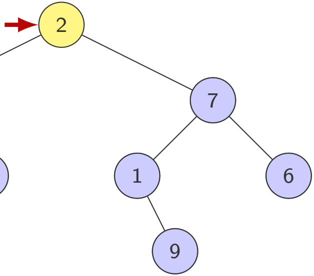


Ex: 3, 4, 8, 5, 9, 1, 6,

Percorrendo os nós — Pós-ordem

A pós-ordem

- primeiro visita a subárvore esquerda
- depois a subárvore direita
- e por último visita a raiz

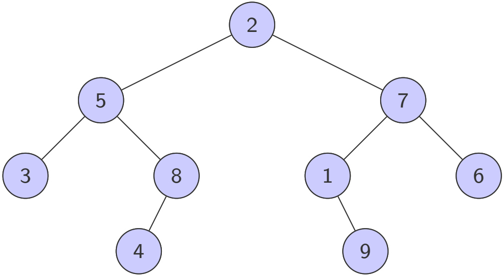


Ex: 3, 4, 8, 5, 9, 1, 6, 7,

Percorrendo os nós — Pós-ordem

A pós-ordem

- primeiro visita a subárvore esquerda
- depois a subárvore direita
- e por último visita a raiz



Ex: 3, 4, 8, 5, 9, 1, 6, 7, 2

Pós-ordem

Algorithm posorder(ptr)

Require: ptr (pointer to node)

```
if ptr ≠ NULL then
    posorder(ptr→left)
    posorder(ptr→right)
    visit(ptr)
end if
```

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

- primeiro visita a subárvore esquerda

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

- primeiro visita a subárvore esquerda
- depois visita a raiz

Percorrendo os nós — Ordem Simétrica

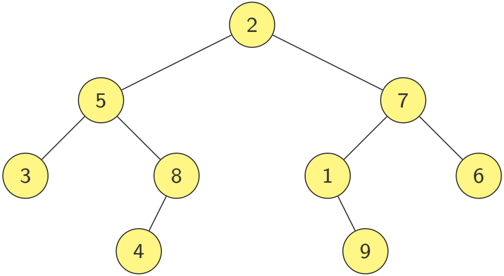
A ordem simétrica

- primeiro visita a subárvore esquerda
- depois visita a raiz
- e por última visita a subárvore direita

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

- primeiro visita a subárvore esquerda
- depois visita a raiz
- e por última visita a subárvore direita

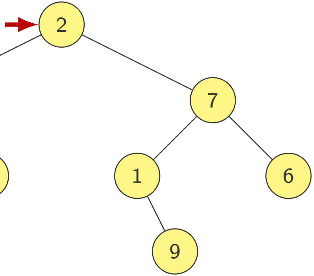


Ex:

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

- primeiro visita a subárvore esquerda
- depois visita a raiz
- e por última visita a subárvore direita

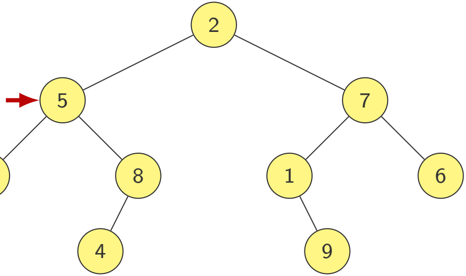


Ex:

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

- primeiro visita a subárvore esquerda
- depois visita a raiz
- e por última visita a subárvore direita

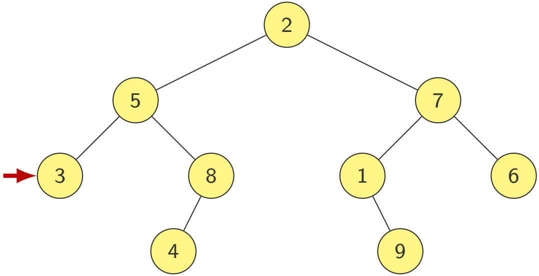


Ex:

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

- primeiro visita a subárvore esquerda
- depois visita a raiz
- e por última visita a subárvore direita

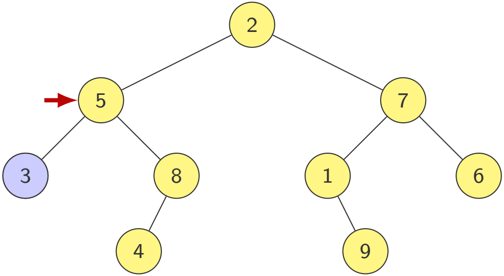


Ex:

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

- primeiro visita a subárvore esquerda
- depois visita a raiz
- e por última visita a subárvore direita

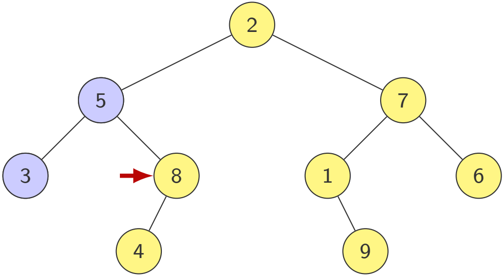


Ex: 3,

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

- primeiro visita a subárvore esquerda
- depois visita a raiz
- e por última visita a subárvore direita

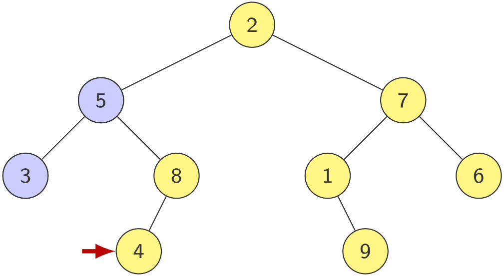


Ex: 3, 5,

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

- primeiro visita a subárvore esquerda
- depois visita a raiz
- e por última visita a subárvore direita

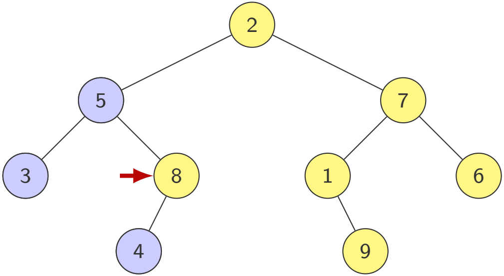


Ex: 3, 5,

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

- primeiro visita a subárvore esquerda
- depois visita a raiz
- e por última visita a subárvore direita

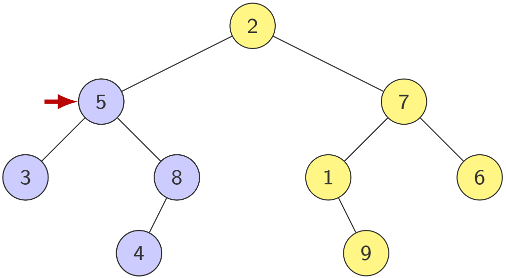


Ex: 3, 5, 4,

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

- primeiro visita a subárvore esquerda
- depois visita a raiz
- e por última visita a subárvore direita

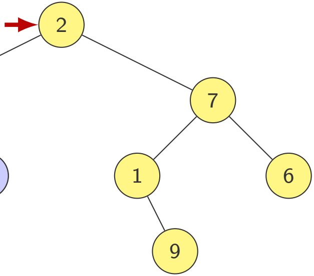


Ex: 3, 5, 4, 8,

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

- primeiro visita a subárvore esquerda
- depois visita a raiz
- e por última visita a subárvore direita

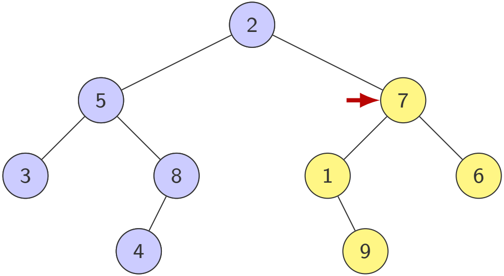


Ex: 3, 5, 4, 8,

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

- primeiro visita a subárvore esquerda
- depois visita a raiz
- e por última visita a subárvore direita

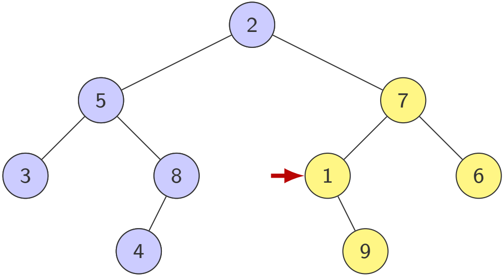


Ex: 3, 5, 4, 8, 2,

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

- primeiro visita a subárvore esquerda
- depois visita a raiz
- e por última visita a subárvore direita

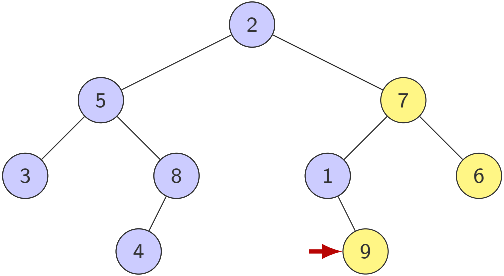


Ex: 3, 5, 4, 8, 2,

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

- primeiro visita a subárvore esquerda
- depois visita a raiz
- e por última visita a subárvore direita

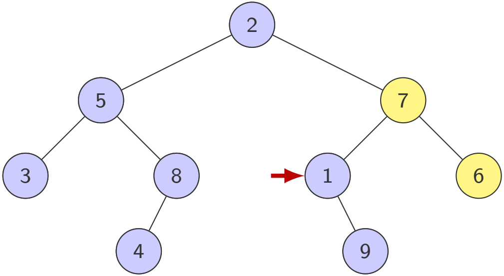


Ex: 3, 5, 4, 8, 2, 1,

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

- primeiro visita a subárvore esquerda
- depois visita a raiz
- e por última visita a subárvore direita

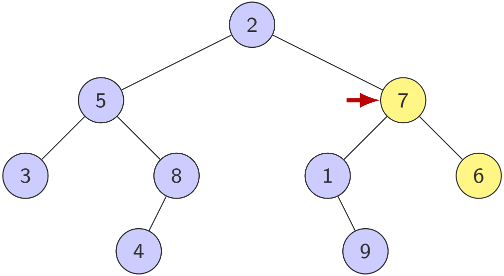


Ex: 3, 5, 4, 8, 2, 1, 9,

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

- primeiro visita a subárvore esquerda
- depois visita a raiz
- e por última visita a subárvore direita

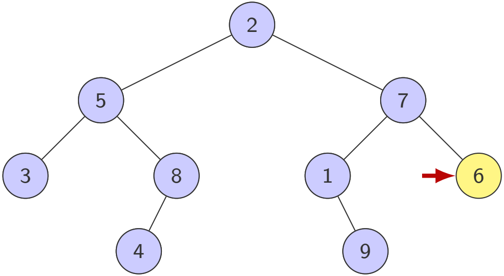


Ex: 3, 5, 4, 8, 2, 1, 9,

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

- primeiro visita a subárvore esquerda
- depois visita a raiz
- e por última visita a subárvore direita

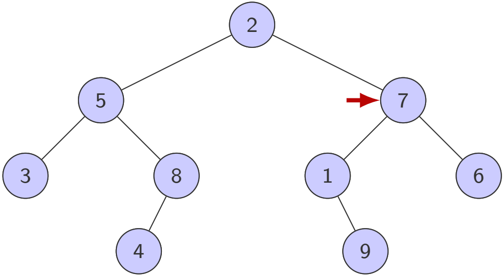


Ex: 3, 5, 4, 8, 2, 1, 9, 7,

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

- primeiro visita a subárvore esquerda
- depois visita a raiz
- e por última visita a subárvore direita

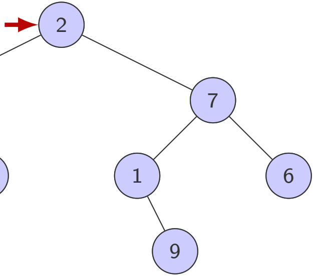


Ex: 3, 5, 4, 8, 2, 1, 9, 7, 6

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

- primeiro visita a subárvore esquerda
- depois visita a raiz
- e por última visita a subárvore direita



Ex: 3, 5, 4, 8, 2, 1, 9, 7, 6

Ordem Simétrica (inorder)

Algorithm inorder(ptr)

Require: ptr (pointer to node)

```
if ptr ≠ NULL then
    inorder(ptr→left)
    visit(ptr)
    inorder(ptr→right)
end if
```

Percurso em largura

Percorrendo os nós (em largura)

O percurso em largura

Percorrendo os nós (em largura)

O percurso em largura

- visita os nós por níveis

Percorrendo os nós (em largura)

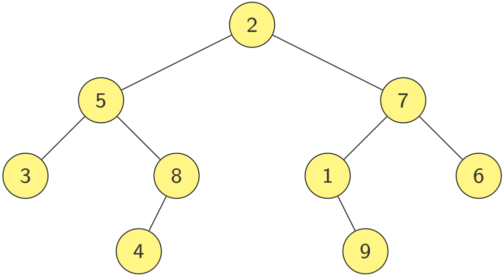
O percurso em largura

- visita os nós por níveis
- da esquerda para a direita

Percorrendo os nós (em largura)

O percurso em largura

- visita os nós por níveis
- da esquerda para a direita

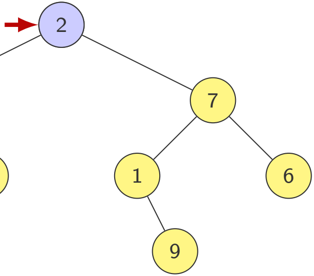


Ex:

Percorrendo os nós (em largura)

O percurso em largura

- visita os nós por níveis
- da esquerda para a direita

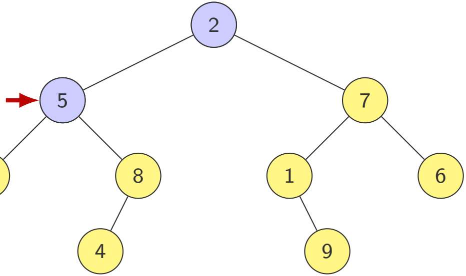


Ex: 2,

Percorrendo os nós (em largura)

O percurso em largura

- visita os nós por níveis
- da esquerda para a direita

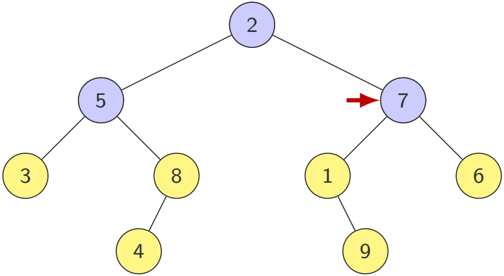


Ex: 2, 5,

Percorrendo os nós (em largura)

O percurso em largura

- visita os nós por níveis
- da esquerda para a direita

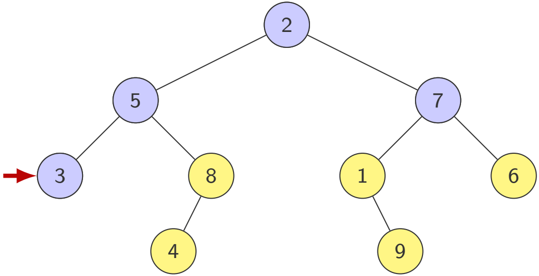


Ex: 2, 5, 7,

Percorrendo os nós (em largura)

O percurso em largura

- visita os nós por níveis
- da esquerda para a direita

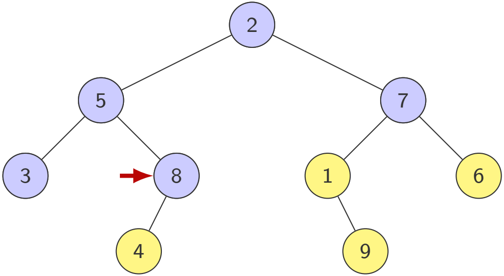


Ex: 2, 5, 7, 3,

Percorrendo os nós (em largura)

O percurso em largura

- visita os nós por níveis
- da esquerda para a direita

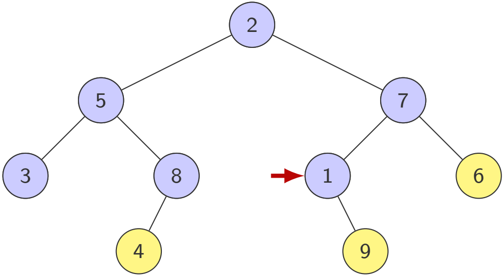


Ex: 2, 5, 7, 3, 8,

Percorrendo os nós (em largura)

O percurso em largura

- visita os nós por níveis
- da esquerda para a direita

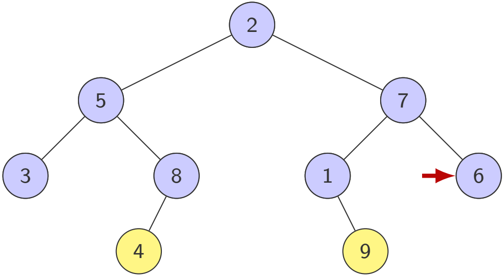


Ex: 2, 5, 7, 3, 8, 1,

Percorrendo os nós (em largura)

O percurso em largura

- visita os nós por níveis
- da esquerda para a direita

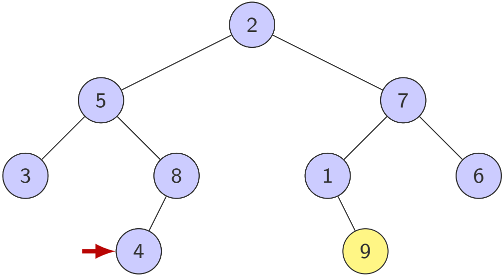


Ex: 2, 5, 7, 3, 8, 1, 6,

Percorrendo os nós (em largura)

O percurso em largura

- visita os nós por níveis
- da esquerda para a direita

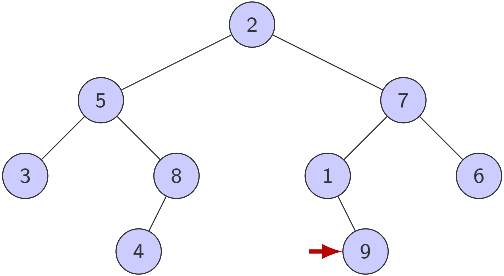


Ex: 2, 5, 7, 3, 8, 1, 6, 4,

Percorrendo os nós (em largura)

O percurso em largura

- visita os nós por níveis
- da esquerda para a direita



Ex: 2, 5, 7, 3, 8, 1, 6, 4, 9

Implementação do percurso em largura

Como implementar a busca em largura?

Implementação do percurso em largura

Como implementar a busca em largura?

- Usamos uma fila

Implementação do percurso em largura

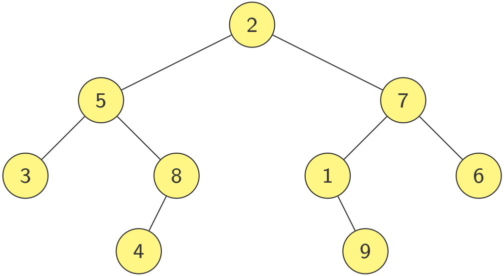
Como implementar a busca em largura?

- Usamos uma fila
- Colocamos a raiz na fila e depois

Implementação do percurso em largura

Como implementar a busca em largura?

- Usamos uma fila
- Colocamos a raiz na fila e depois
- pegamos um elemento da fila e enfileiramos seus filhos

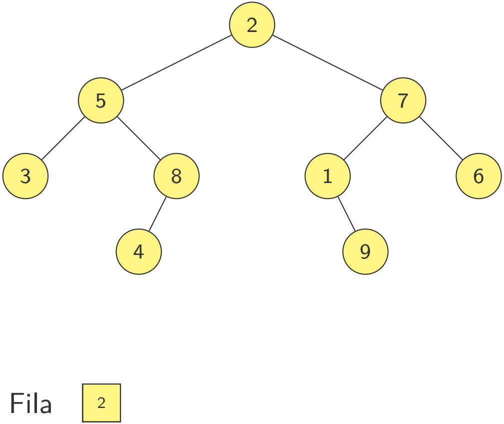


Fila

Implementação do percurso em largura

Como implementar a busca em largura?

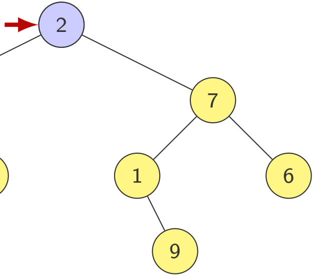
- Usamos uma fila
- Colocamos a raiz na fila e depois
- pegamos um elemento da fila e enfileiramos seus filhos



Implementação do percurso em largura

Como implementar a busca em largura?

- Usamos uma fila
- Colocamos a raiz na fila e depois
- pegamos um elemento da fila e enfileiramos seus filhos



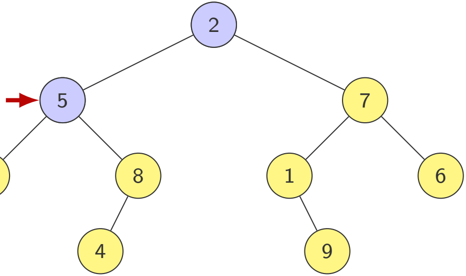
Fila

2	5	7

Implementação do percurso em largura

Como implementar a busca em largura?

- Usamos uma fila
- Colocamos a raiz na fila e depois
- pegamos um elemento da fila e enfileiramos seus filhos



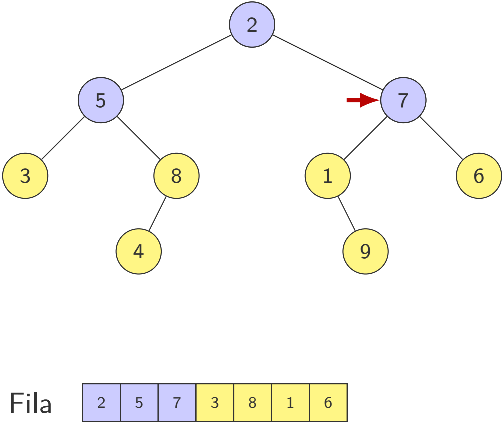
Fila

2	5	7	3	8

Implementação do percurso em largura

Como implementar a busca em largura?

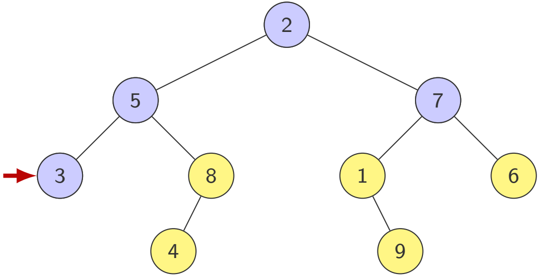
- Usamos uma fila
- Colocamos a raiz na fila e depois
- pegamos um elemento da fila e enfileiramos seus filhos



Implementação do percurso em largura

Como implementar a busca em largura?

- Usamos uma fila
- Colocamos a raiz na fila e depois
- pegamos um elemento da fila e enfileiramos seus filhos



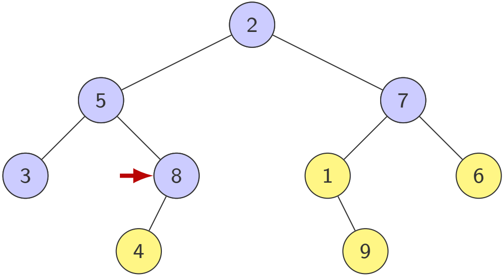
Fila

2	5	7	3	8	1	6

Implementação do percurso em largura

Como implementar a busca em largura?

- Usamos uma fila
- Colocamos a raiz na fila e depois
- pegamos um elemento da fila e enfileiramos seus filhos



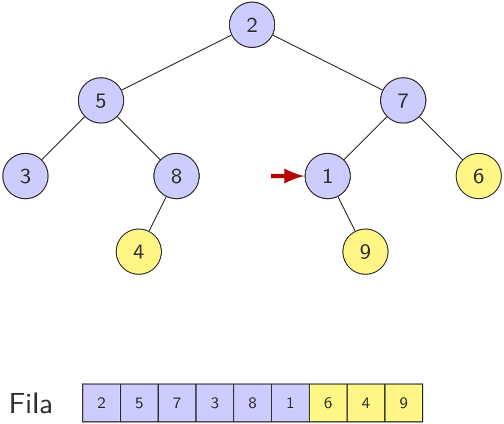
Fila

2	5	7	3	8	1	6	4

Implementação do percurso em largura

Como implementar a busca em largura?

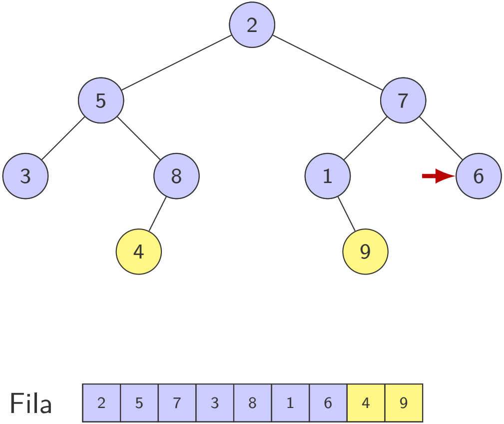
- Usamos uma fila
- Colocamos a raiz na fila e depois
- pegamos um elemento da fila e enfileiramos seus filhos



Implementação do percurso em largura

Como implementar a busca em largura?

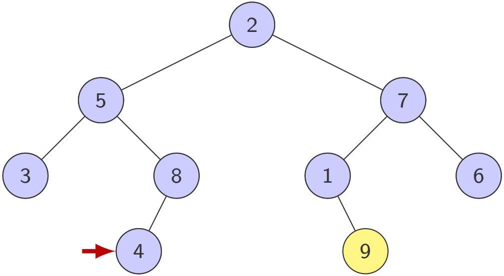
- Usamos uma fila
- Colocamos a raiz na fila e depois
- pegamos um elemento da fila e enfileiramos seus filhos



Implementação do percurso em largura

Como implementar a busca em largura?

- Usamos uma fila
- Colocamos a raiz na fila e depois
- pegamos um elemento da fila e enfileiramos seus filhos



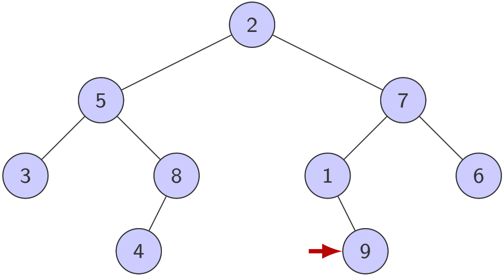
Fila

2	5	7	3	8	1	6	4	9

Implementação do percurso em largura

Como implementar a busca em largura?

- Usamos uma fila
- Colocamos a raiz na fila e depois
- pegamos um elemento da fila e enfileiramos seus filhos



Fila

2	5	7	3	8	1	6	4	9

Percurso em largura

Algorithm levelTraversal(root)

```
Require: root (ponteiro para a raiz)
Cria uma fila vazia Q de ponteiros para nós
Q.push(root)
while Q ≠ ∅ do
    node = Q.front()
    Q.pop()
    if node ≠ NULL then
        visit(node)
        Q.push(node→left)
        Q.push(node→right)
    end if
end while
```

Exercícios

Exercícios

- Escreva uma função que calcula o número de nós de uma árvore. A função deve obedecer o seguinte protótipo:
  `int bt_size(Node* node);`

Exercícios

- Escreva uma função que calcula o número de nós de uma árvore. A função deve obedecer o seguinte protótipo:
  `int bt_size(Node* node);`

- Escreva uma função que calcula a altura de uma árvore. A função deve obedecer o seguinte protótipo:
  `int bt_height(Node* node);`

Exercícios

- Escreva uma função que calcula o número de nós de uma árvore. A função deve obedecer o seguinte protótipo:
  `int bt_size(Node* node);`

- Escreva uma função que calcula a altura de uma árvore. A função deve obedecer o seguinte protótipo:
  `int bt_height(Node* node);`

- Adicione o campo `height` ao struct `Node`. O campo `height` deve ser do tipo `int`. Implemente a função `bt_height(Node* node)` de modo que ela preencha o campo `height` de cada nó com a altura do nó.

Exercícios

- Um caminho que vai da raiz de uma árvore até um nó qualquer pode ser representado por uma sequência de 0s e 1s, do seguinte modo:
  - toda vez que o caminho "desce para a esquerda" temos um 0; toda vez que "desce para a direita" temos um 1.
  - Diremos que essa sequência de 0s e 1s é o código do nó.

- Suponha agora que todo nó de nossa árvore tem um campo adicional `code`, do tipo `std::string`, capaz de armazenar uma cadeia de caracteres de tamanho variável. Escreva uma função que preencha o campo `code` de cada nó com o código do nó.

FIM