Árvores Binárias

Estrutura de Dados Avançada — QXD0115

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

Universidade Federal do Ceará

1º semestre/2023

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.

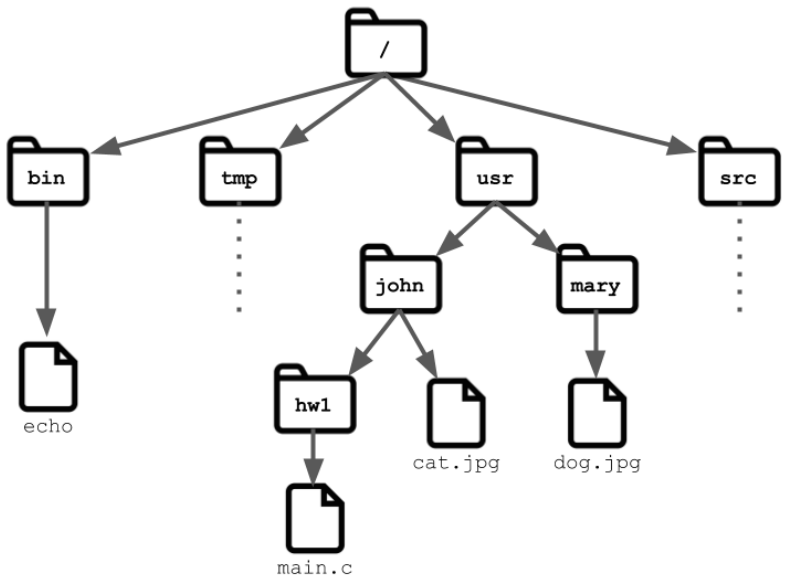


Figura: Hierarquia do sistema de arquivos de um PC Linux

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.

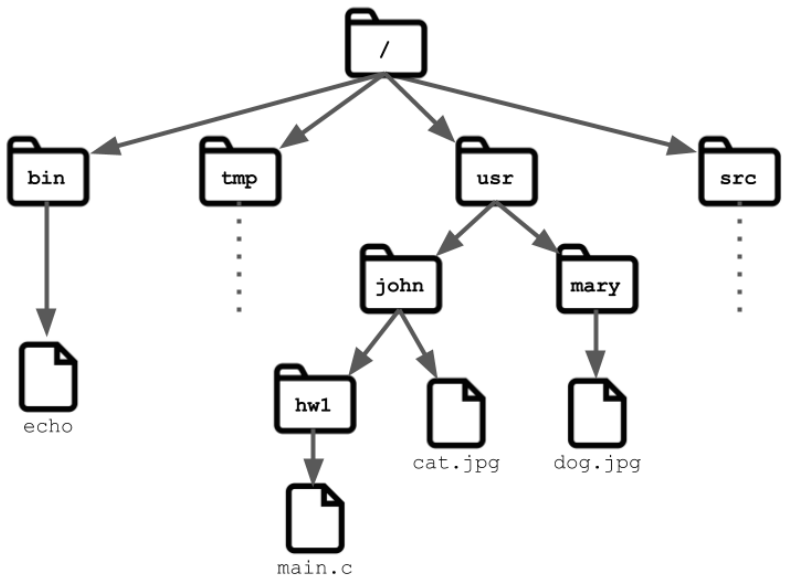


Figura: Hierarquia do sistema de arquivos de um PC Linux

- 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.

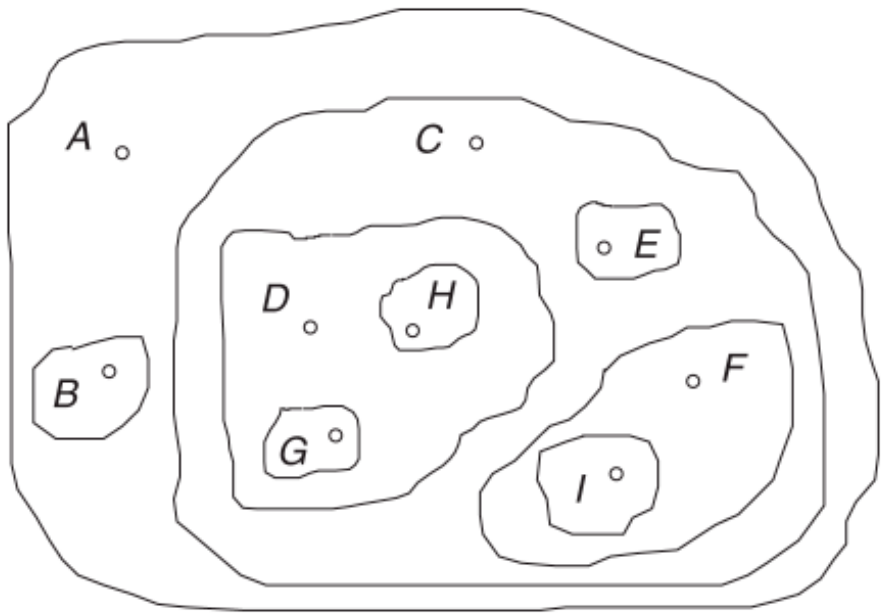
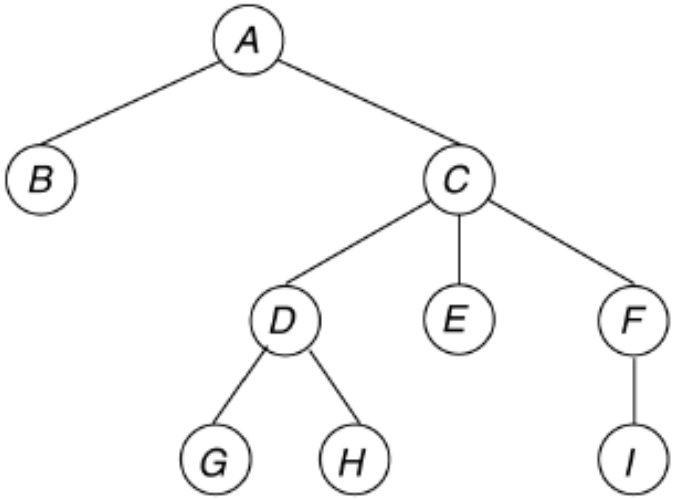


Diagrama de inclusão

Árvore — Outras Representações



Representação hierárquica

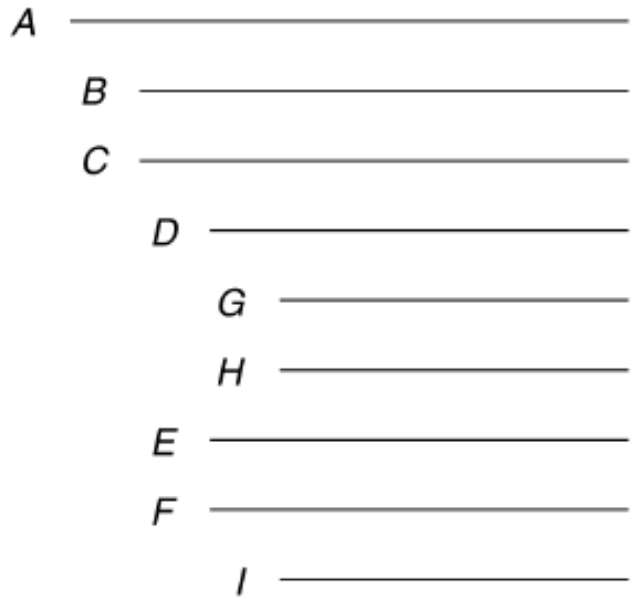
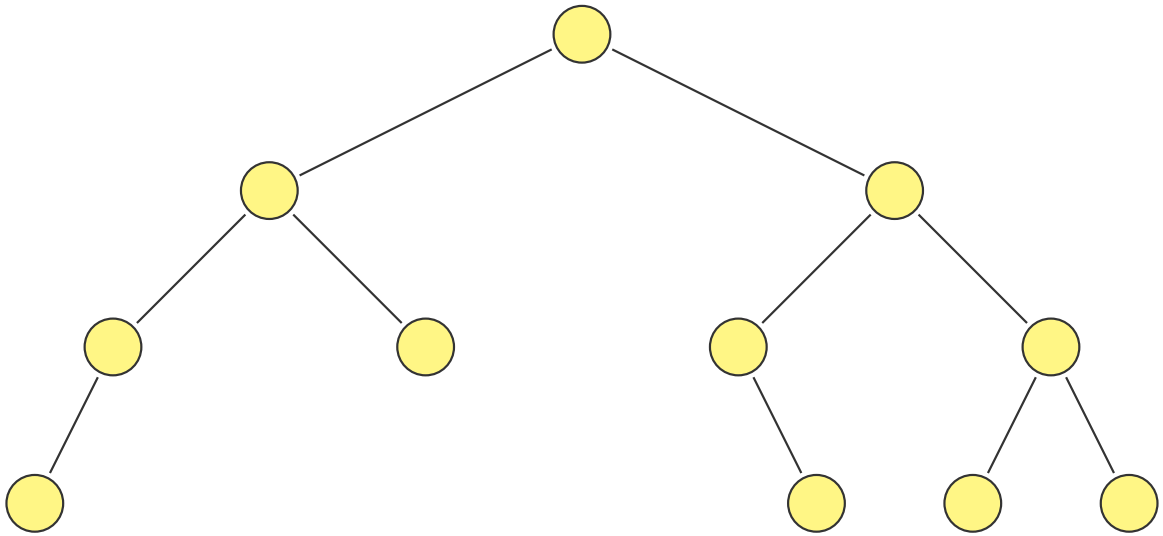


Diagrama de barras

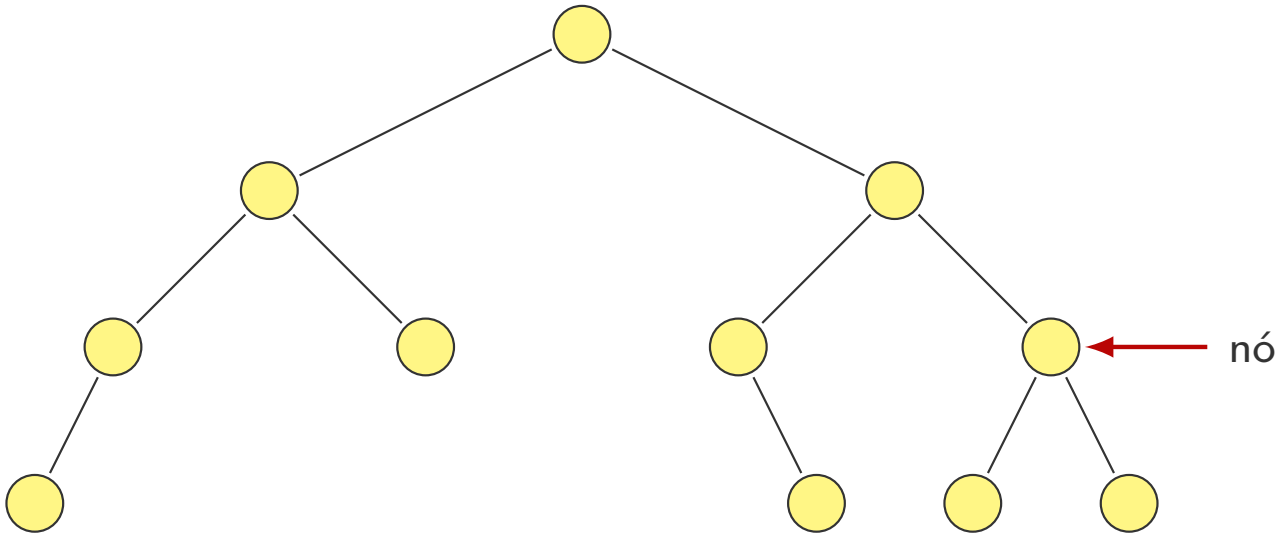
(A (B) (C (D (G) (H)) (E) (F (I))))

Representação por parênteses aninhados

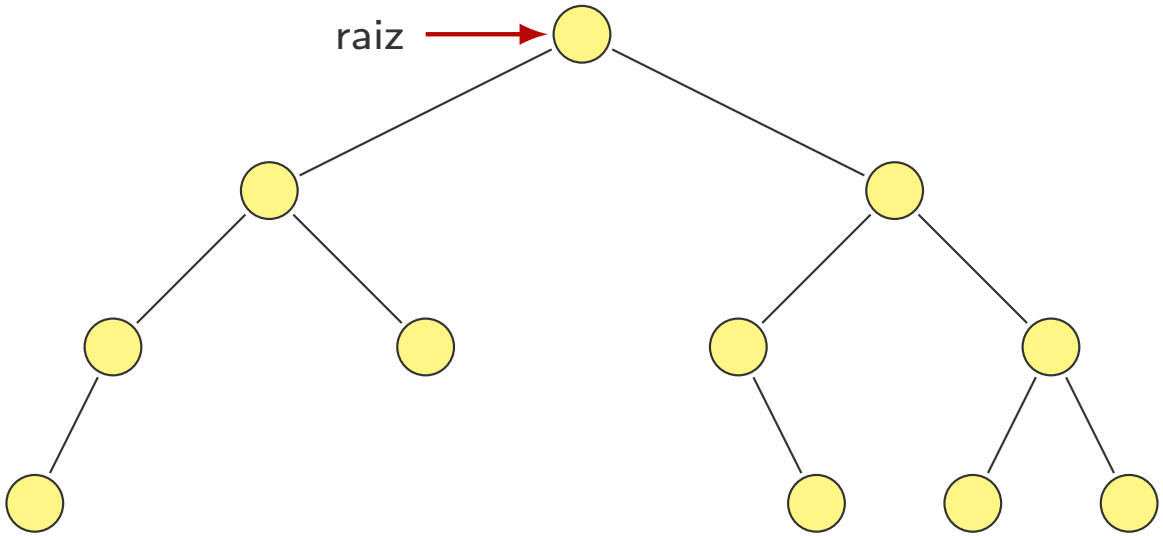
Definições Adicionais



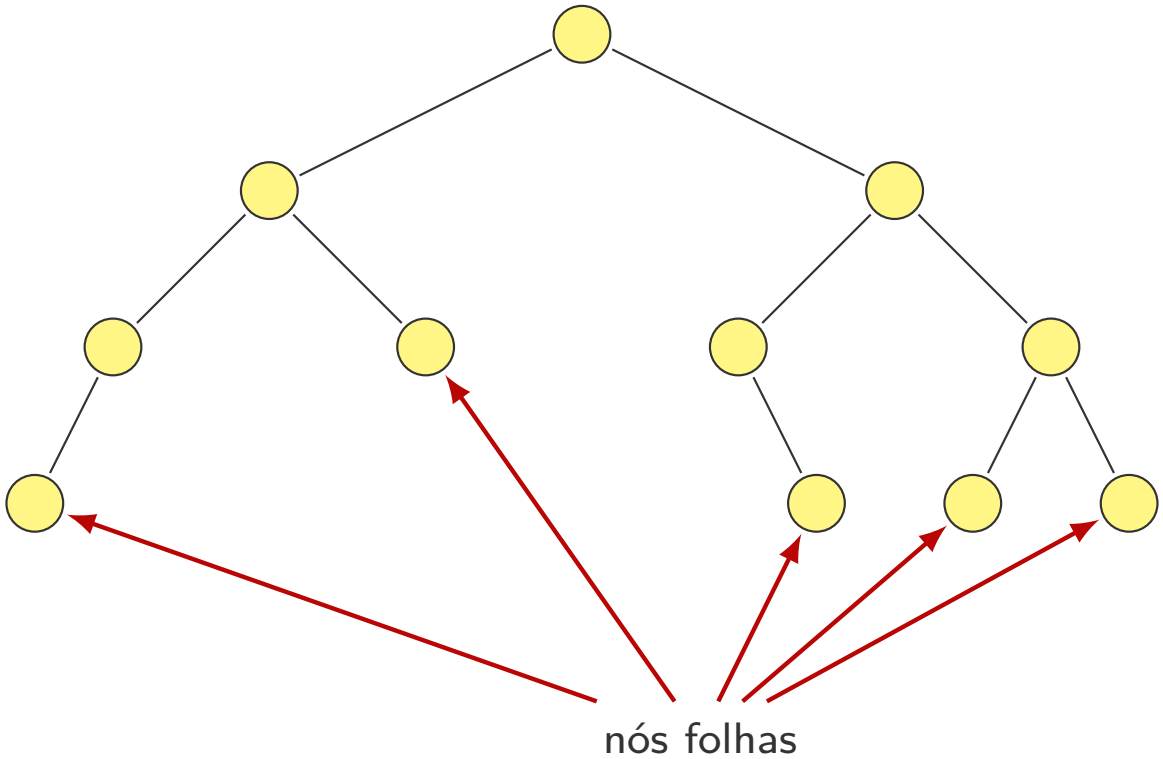
Definições Adicionais



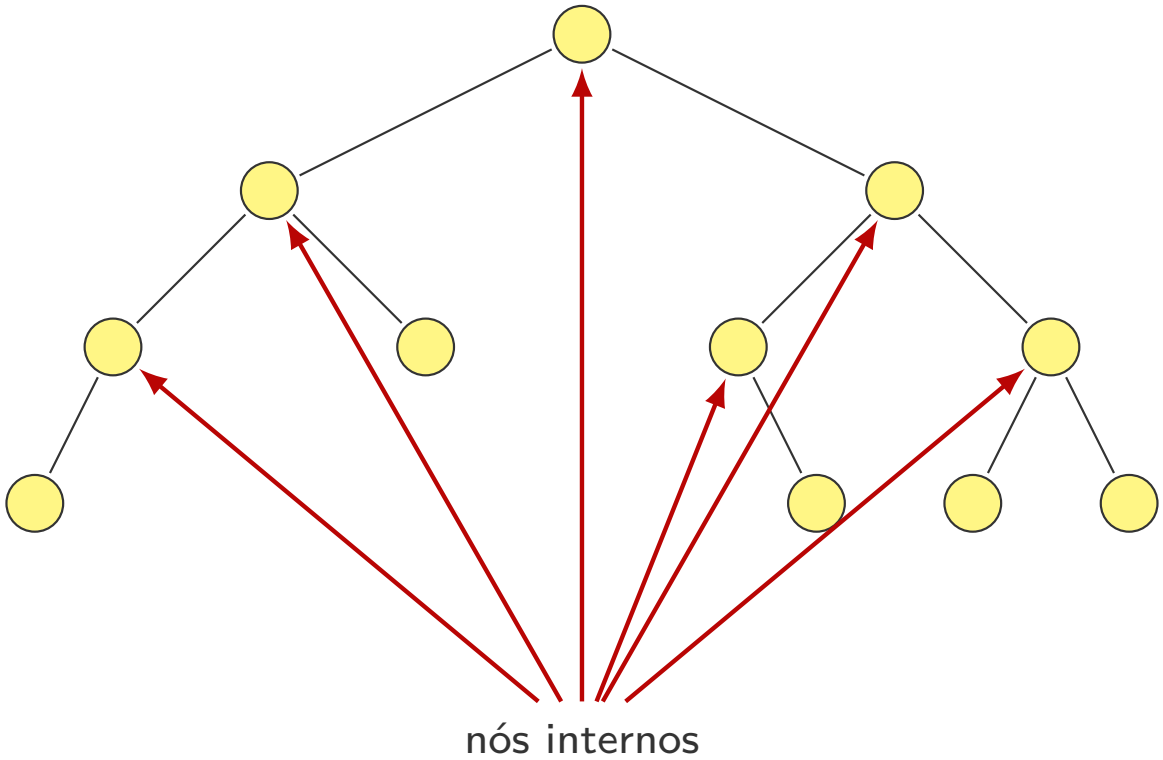
Definições Adicionais



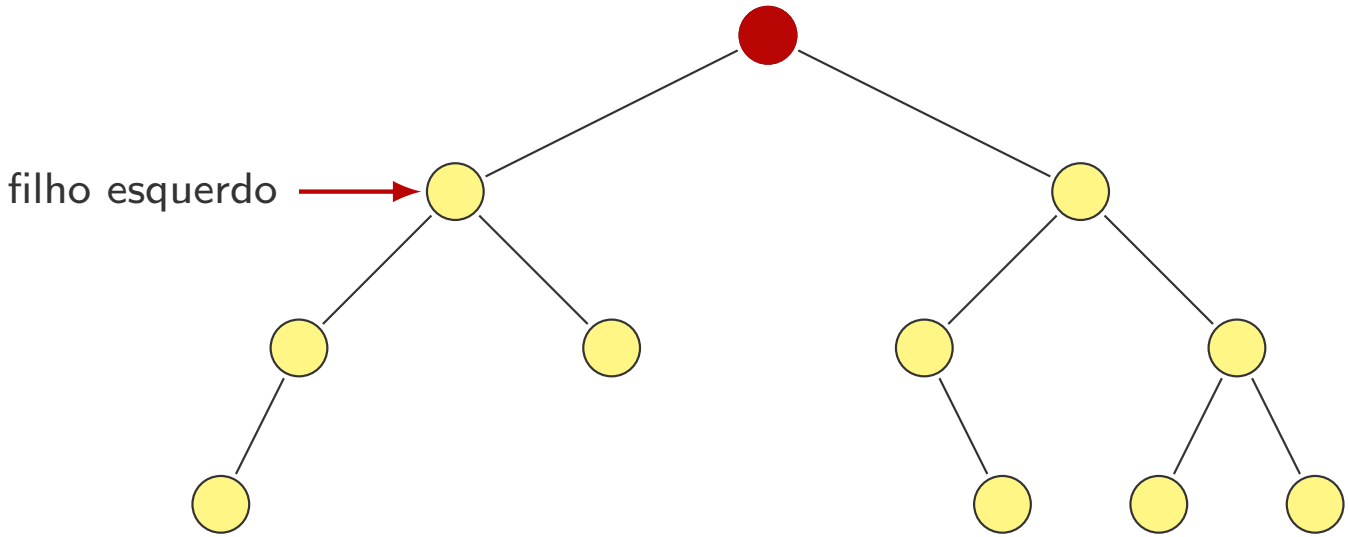
Definições Adicionais



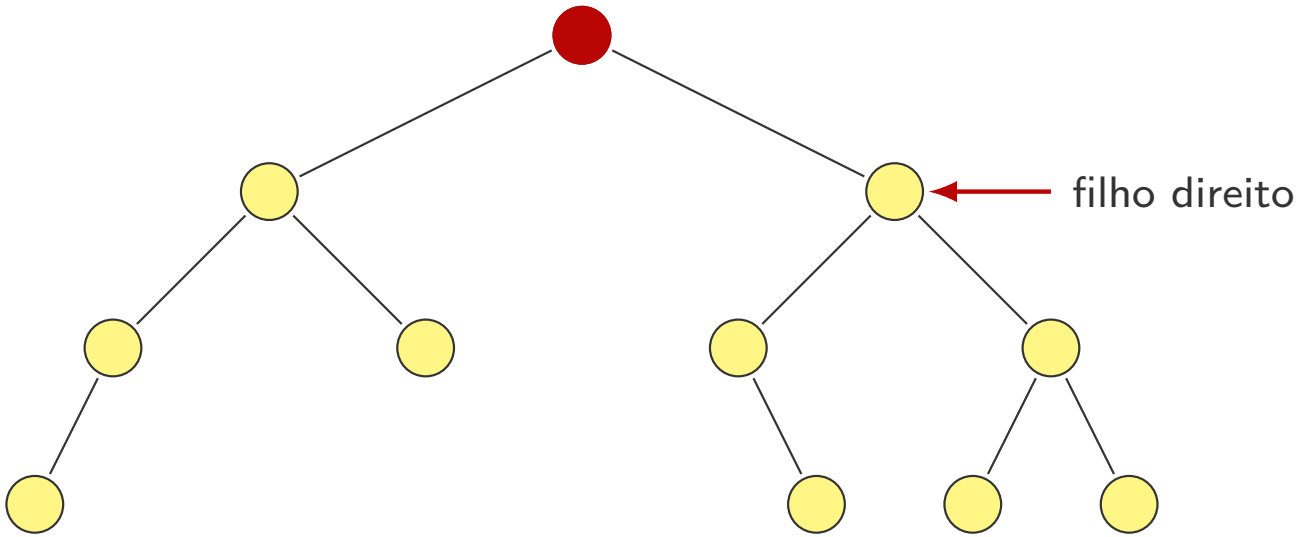
Definições Adicionais



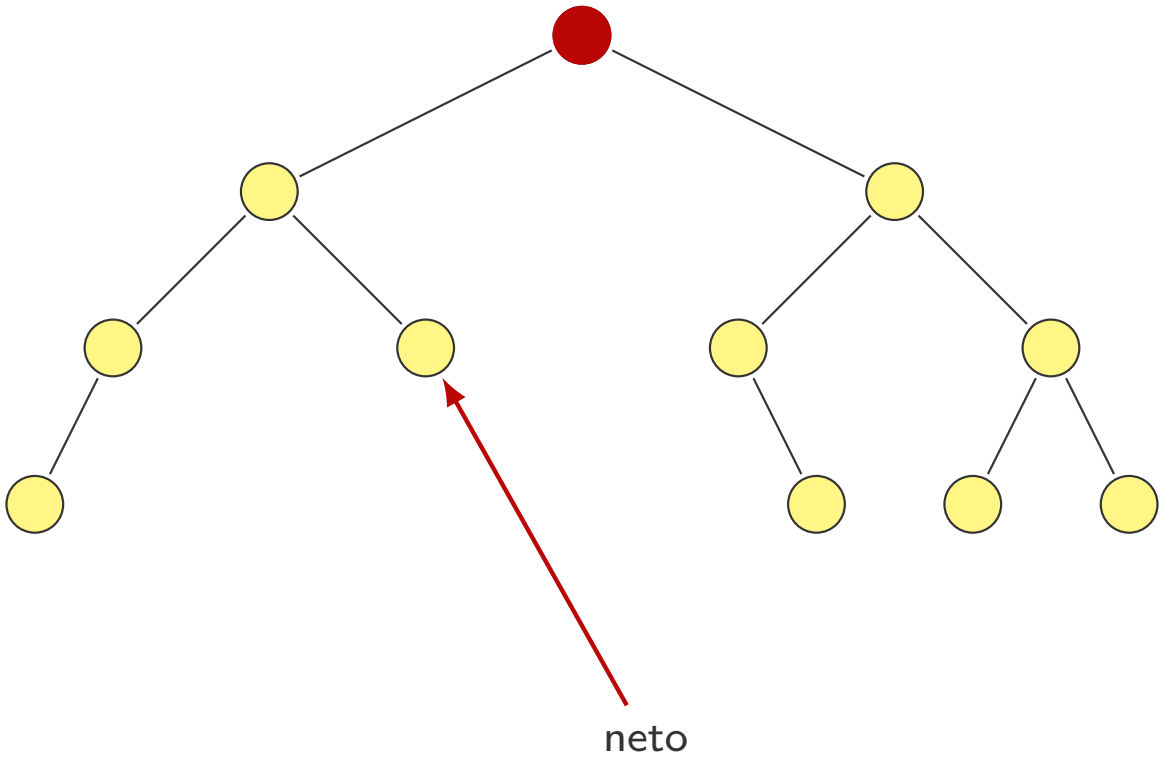
Definições Adicionais



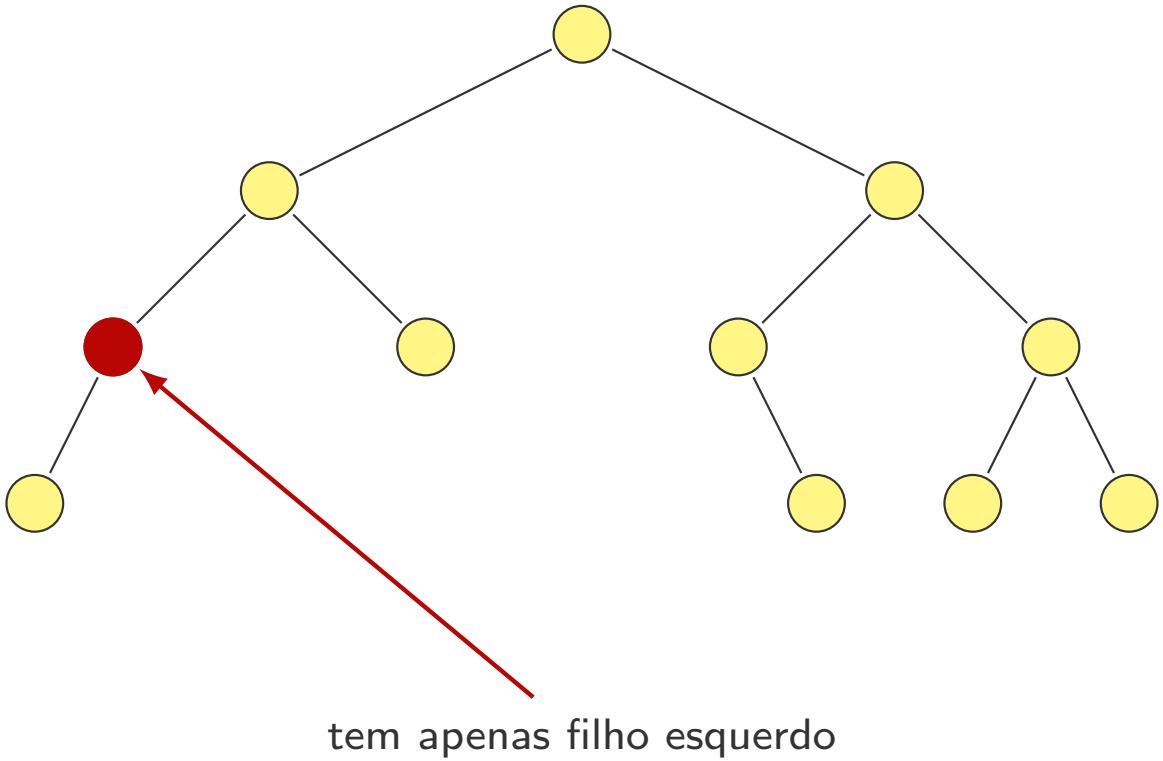
Definições Adicionais



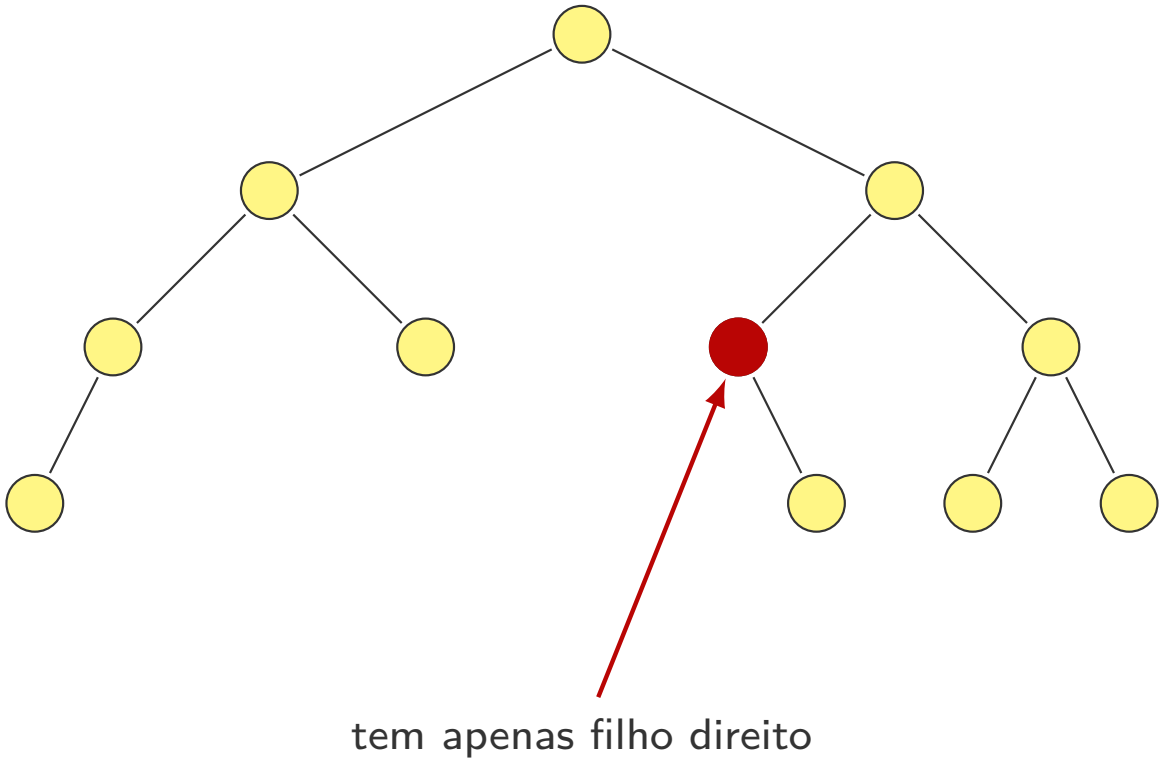
Definições Adicionais



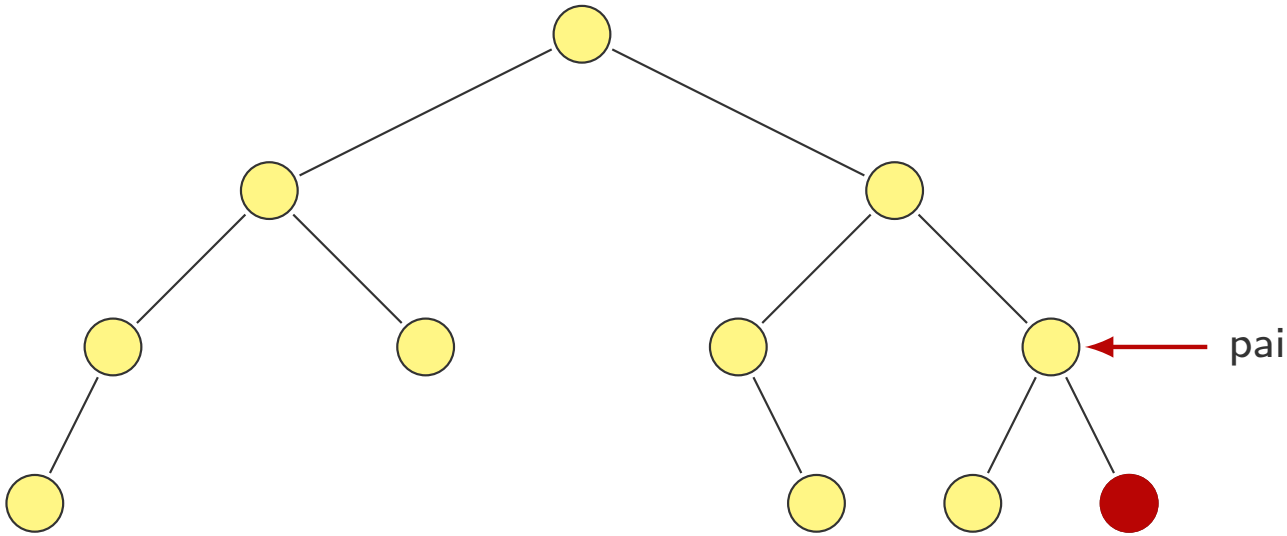
Definições Adicionais



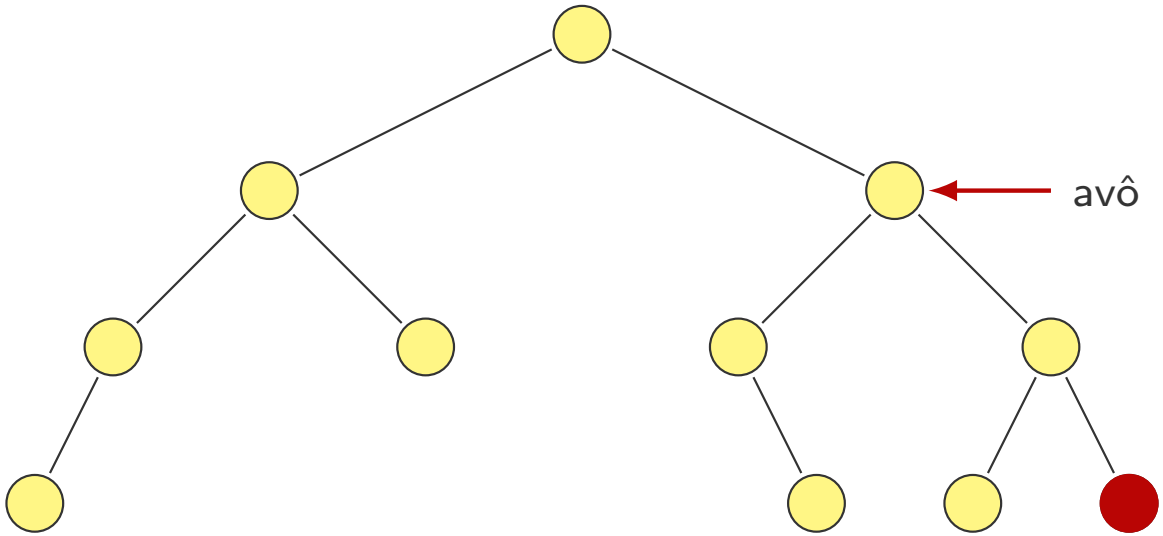
Definições Adicionais



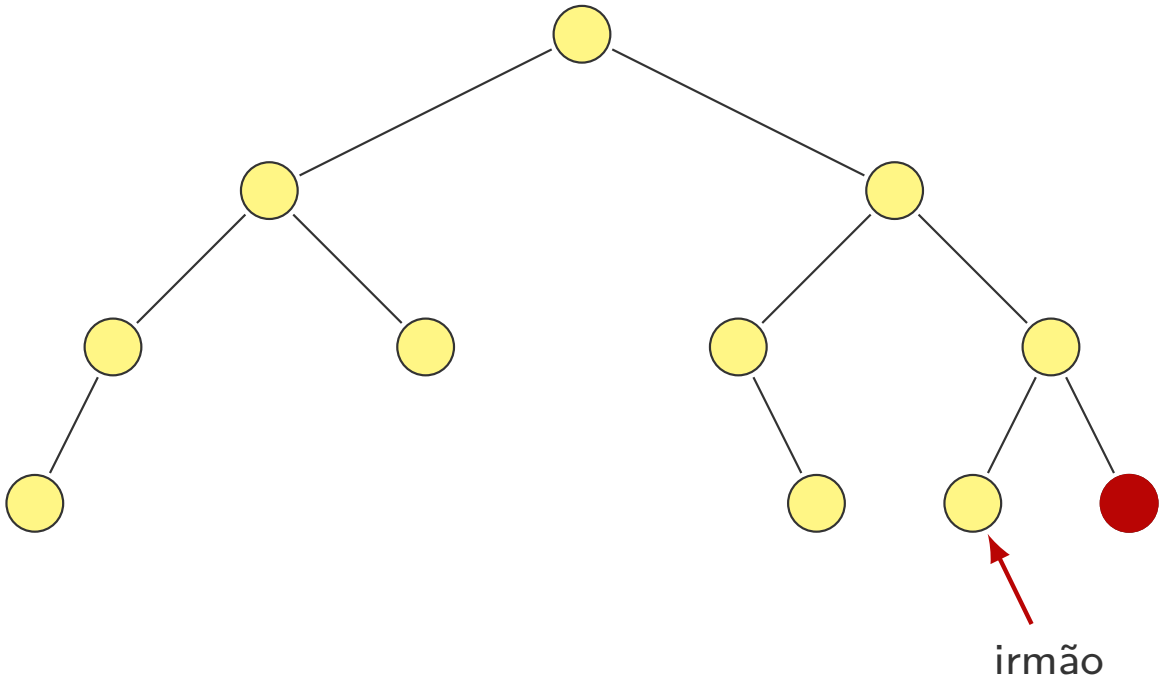
Definições Adicionais



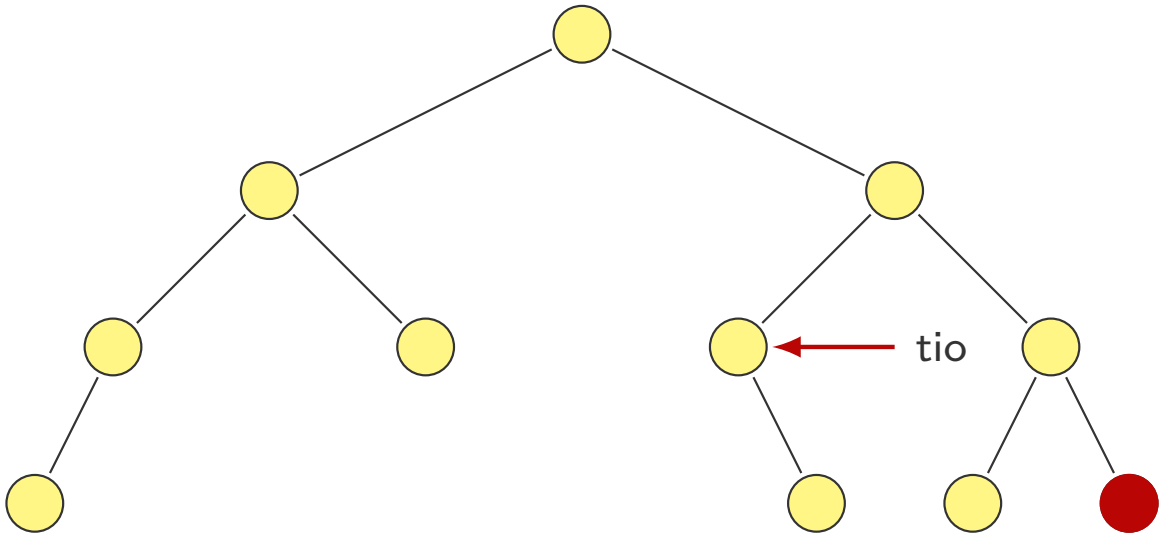
Definições Adicionais



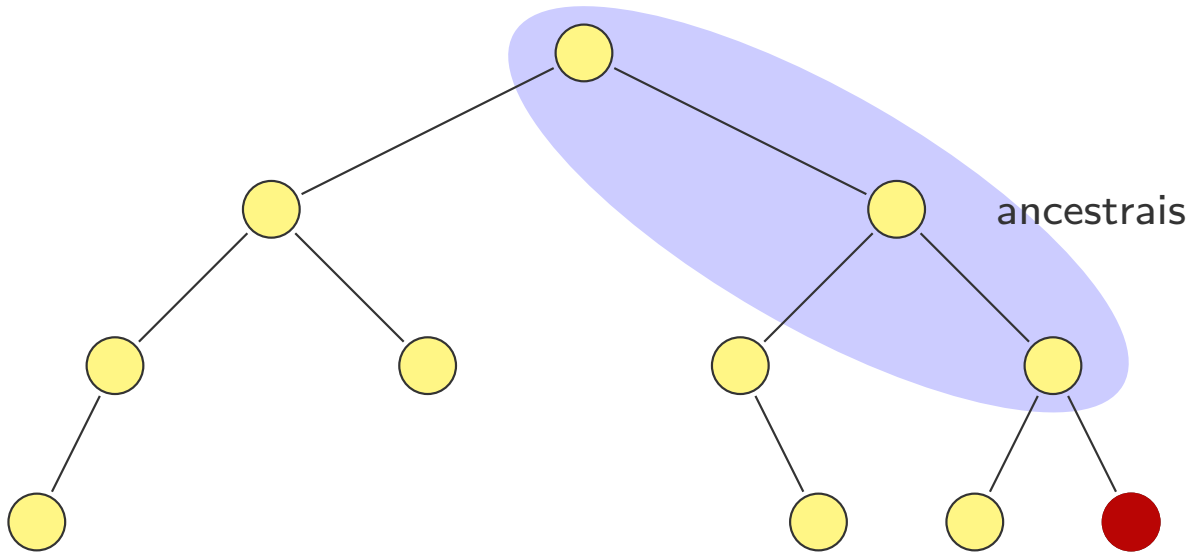
Definições Adicionais



Definições Adicionais

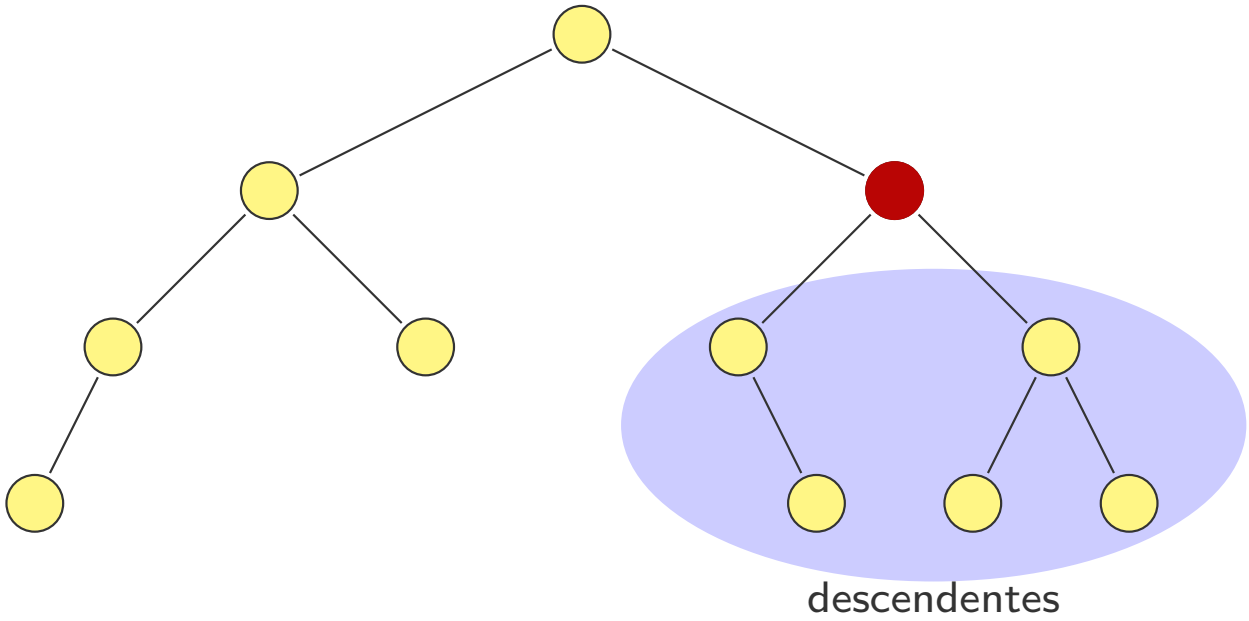


Definições Adicionais

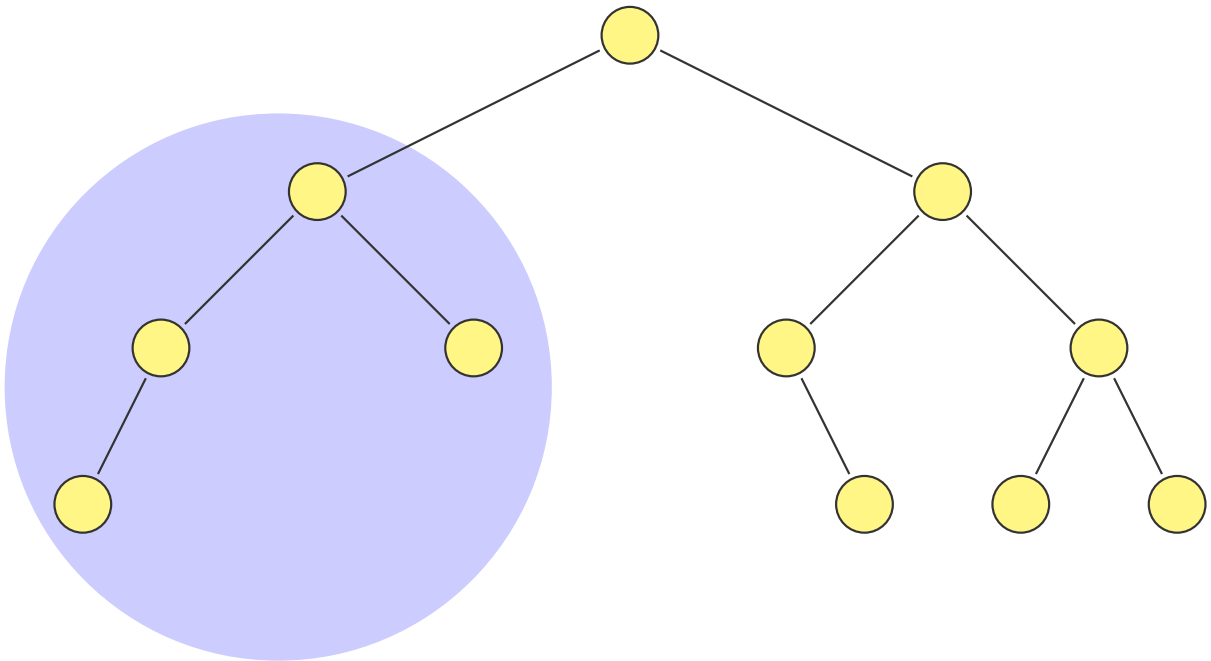


Uma sequência de nós distintos $v_1, v_2, \ldots, v_k$, tal que existe sempre entre nós consecutivos a relação "é filho de " ou "é pai de", é denominada um caminho na árvore.

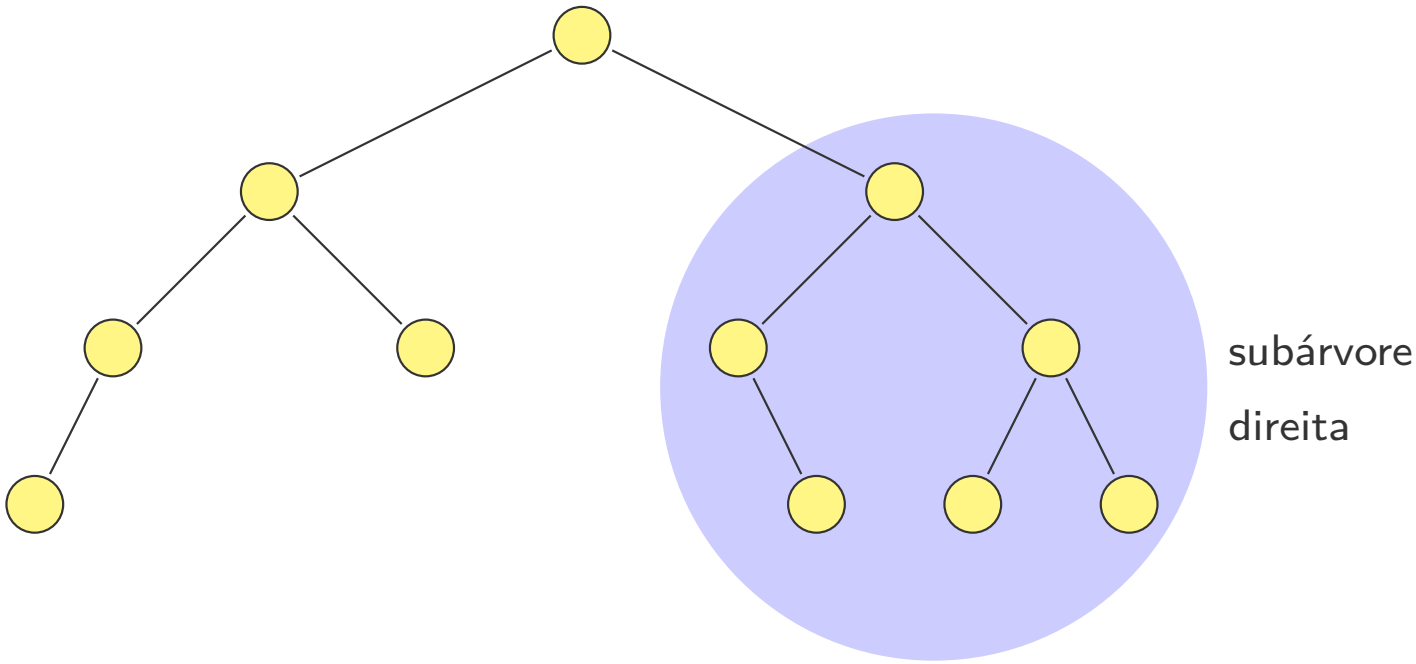
Definições Adicionais



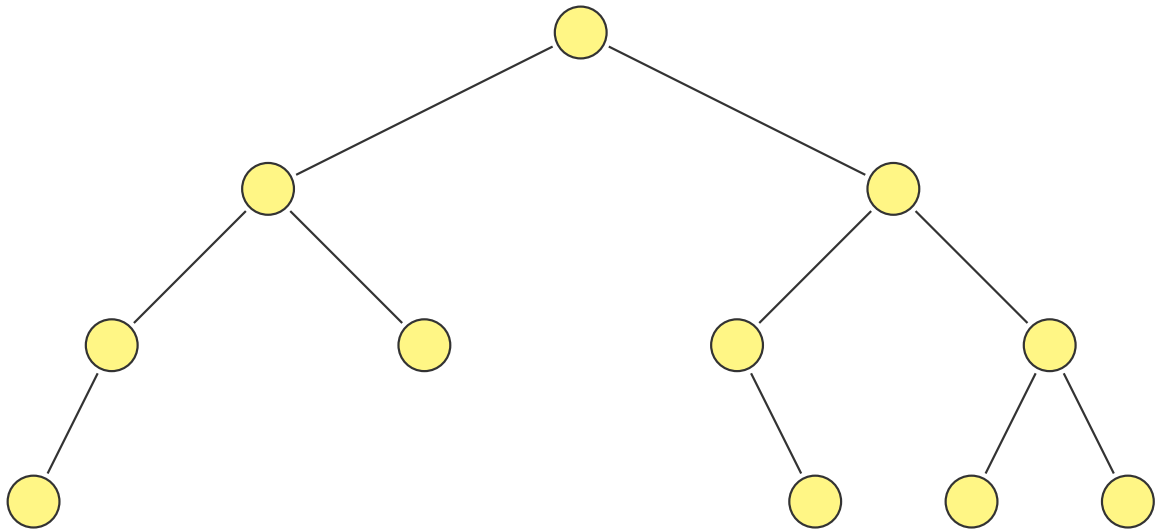
Definições Adicionais



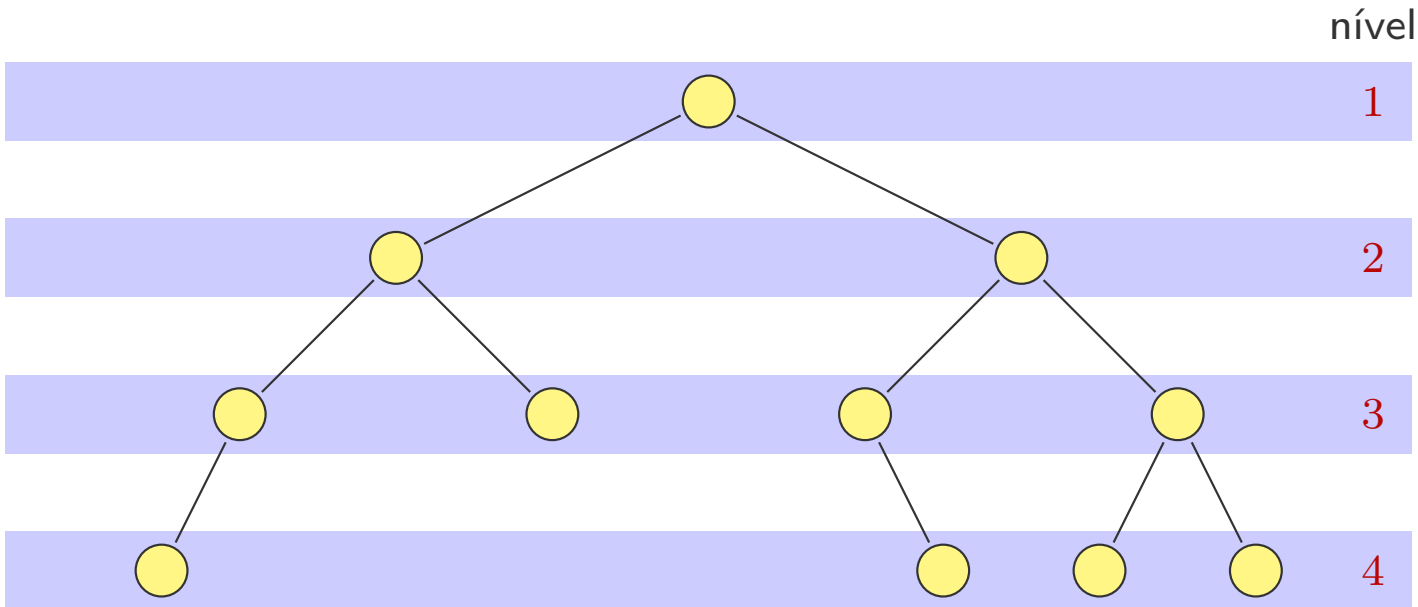
Definições Adicionais



Definições Adicionais — Nível e Altura

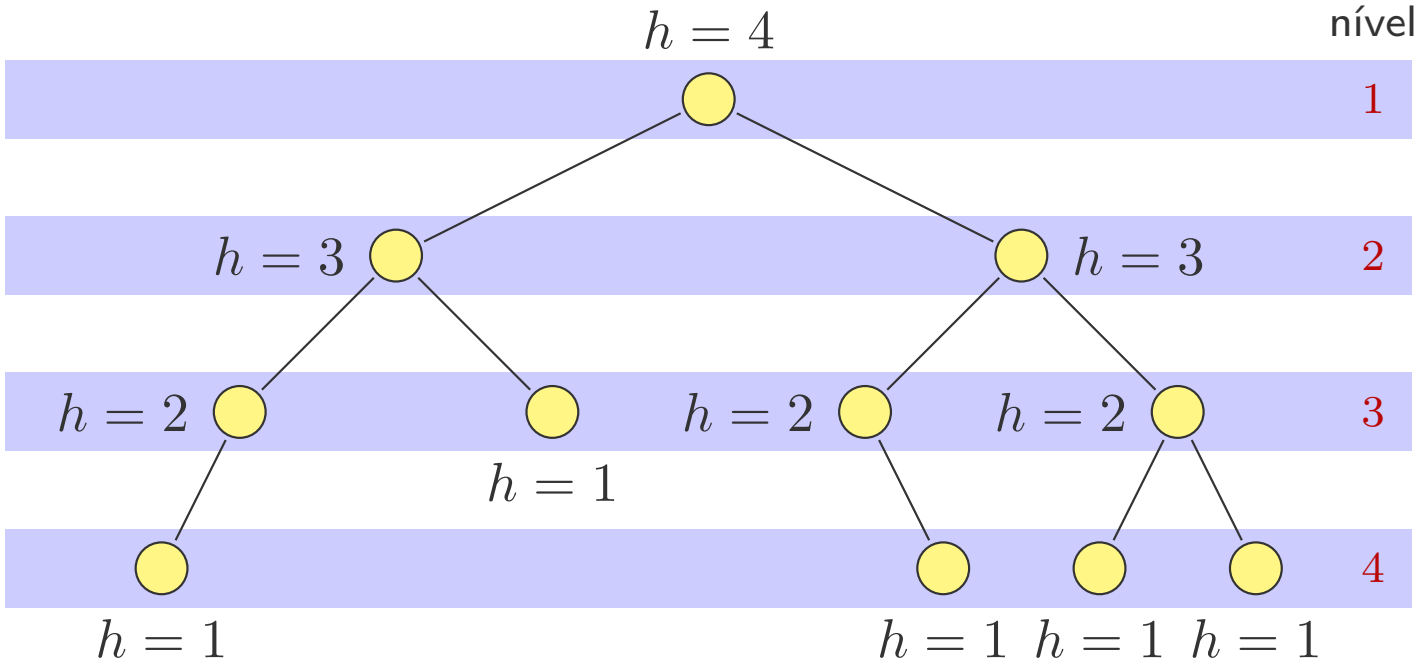


Definições Adicionais — Nível e Altura



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.

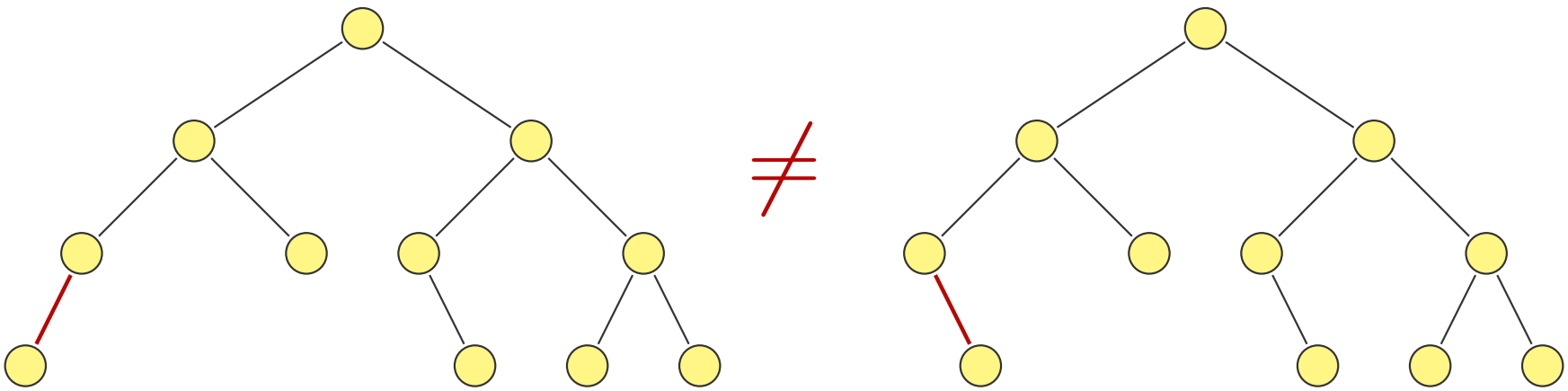
Definições Adicionais — Nível e Altura



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.

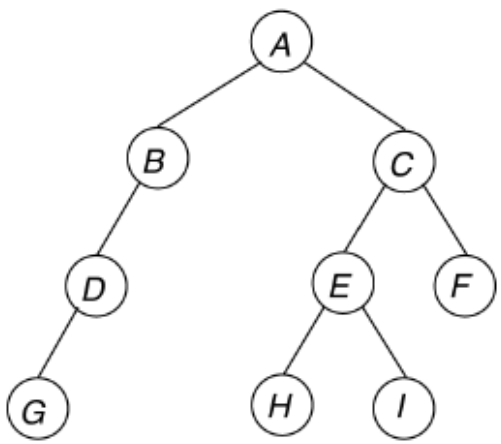
Comparando com atenção



Ordem dos filhos é relevante!

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.

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.

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

Quantidade de subárvores vazias

Lema 1: O número de subárvores esquerda e direita vazias em uma árvore binária com $n > 0$ nós é $n + 1$.

Exercício: Prove este lema. Dica: use indução no número de nós da árvore.

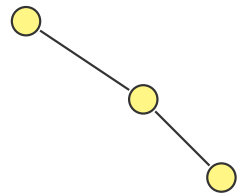
Árvore Binária - relação entre altura e número de nós

Se a altura é h, então a árvore binária:

Árvore Binária - relação entre altura e número de nós

Se a altura é h, então a árvore binária:

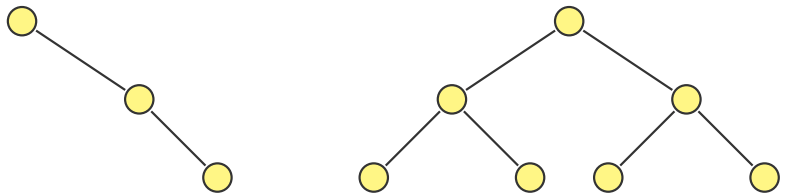
- tem no mínimo h nós
  (árvore caminho)



Árvore Binária - relação entre altura e número de nós

Se a altura é h, então a árvore binária:

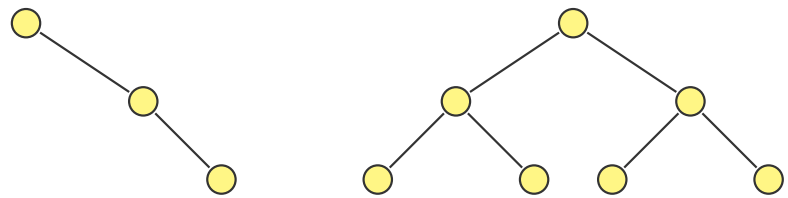
- tem no mínimo h nós
  (árvore caminho)
- tem no máximo $2^h - 1$ nós
  (árvore cheia)



Árvore Binária - relação entre altura e número de nós

Se a altura é h, então a árvore binária:

- tem no mínimo h nós
  (árvore caminho)
- tem no máximo $2^h - 1$ nós
  (árvore cheia)

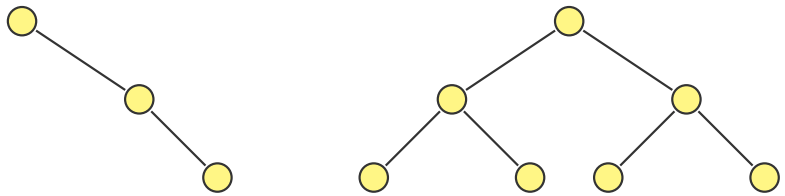


Se a árvore binária tem $n \geq 1$ nós, então:

Árvore Binária - relação entre altura e número de nós

Se a altura é h, então a árvore binária:

- tem no mínimo h nós
  (árvore caminho)
- tem no máximo $2^h - 1$ nós
  (árvore cheia)



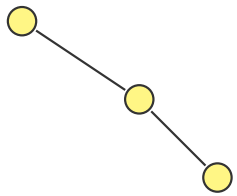
Se a árvore binária tem $n \geq 1$ nós, então:

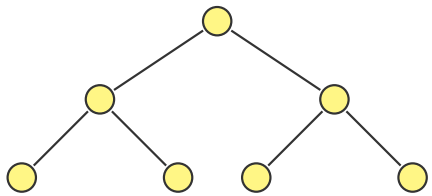
- a altura é no máximo n

Árvore Binária - relação entre altura e número de nós

Se a altura é h, então a árvore binária:

- tem no mínimo h nós
  (árvore caminho)
- tem no máximo $2^h - 1$ nós
  (árvore cheia)





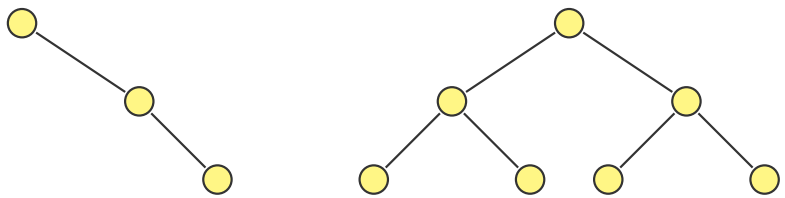
Se a árvore binária tem $n \geq 1$ nós, então:

- a altura é no máximo n
  - quando cada nó interno tem apenas um filho
    (a árvore é um caminho)

Árvore Binária - relação entre altura e número de nós

Se a altura é h, então a árvore binária:

- tem no mínimo h nós
  (árvore caminho)
- tem no máximo $2^h - 1$ nós
  (árvore cheia)



Se a árvore binária tem $n \geq 1$ nós, então:

- a altura é no máximo n
  - quando cada nó interno tem apenas um filho
    (a árvore é um caminho)
- Quem é a árvore de altura mínima e que altura ela tem?

Árvore Binária - relação entre altura e número de nós

Lema 2: Se T é uma árvore binária completa com $n > 0$ nós, então T possui altura h mínima.

Árvore Binária - relação entre altura e número de nós

Lema 2: Se T é uma árvore binária completa com $n > 0$ nós, então T possui altura h mínima.

Exercício: Prove este lema. Dica: Pegue uma árvore binária T^* de altura h mínima com n nós. A seguir, considere dois casos: T^* sendo completa e T^* sendo não completa. Sabendo que toda árvore completa com n nós possui a mesma altura, tente chegar, em cada caso, em alguma conclusão sobre a altura da árvore T do enunciado.

Árvore Binária - relação entre altura e número de nós

Lema 2: Se T é uma árvore binária completa com $n > 0$ nós, então T possui altura h mínima.

Exercício: Prove este lema. Dica: Pegue uma árvore binária T^* de altura h mínima com n nós. A seguir, considere dois casos: T^* sendo completa e T^* sendo não completa. Sabendo que toda árvore completa com n nós possui a mesma altura, tente chegar, em cada caso, em alguma conclusão sobre a altura da árvore T do enunciado.

Lema 3: Se T é uma árvore binária completa com $n > 0$ nós, então T possui altura $h = 1 + \lfloor \lg n \rfloor$.

Árvore Binária - relação entre altura e número de nós

Lema 2: Se T é uma árvore binária completa com $n > 0$ nós, então T possui altura h mínima.

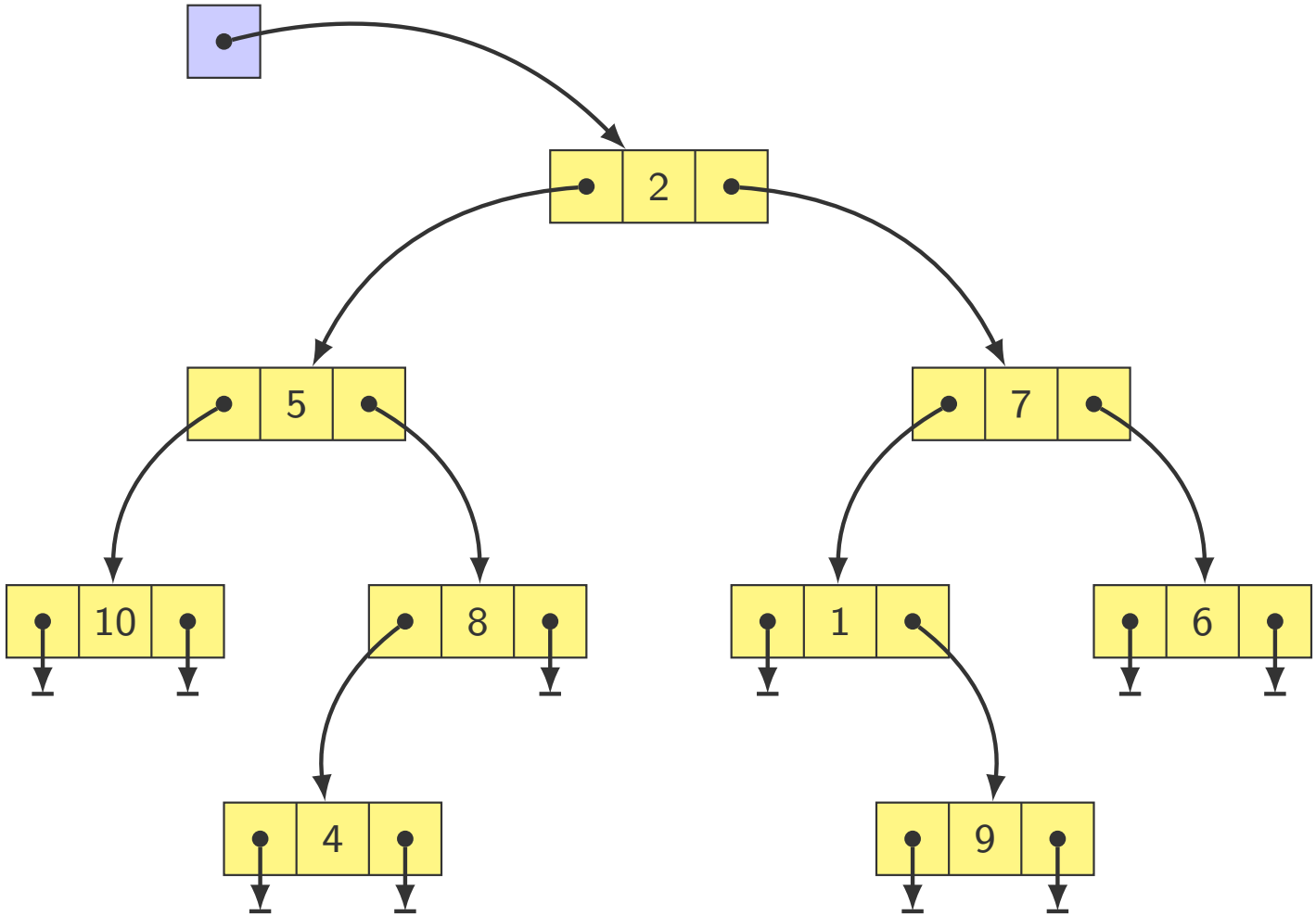
Exercício: Prove este lema. Dica: Pegue uma árvore binária T^* de altura h mínima com n nós. A seguir, considere dois casos: T^* sendo completa e T^* sendo não completa. Sabendo que toda árvore completa com n nós possui a mesma altura, tente chegar, em cada caso, em alguma conclusão sobre a altura da árvore T do enunciado.

Lema 3: Se T é uma árvore binária completa com $n > 0$ nós, então T possui altura $h = 1 + \lfloor \lg n \rfloor$.

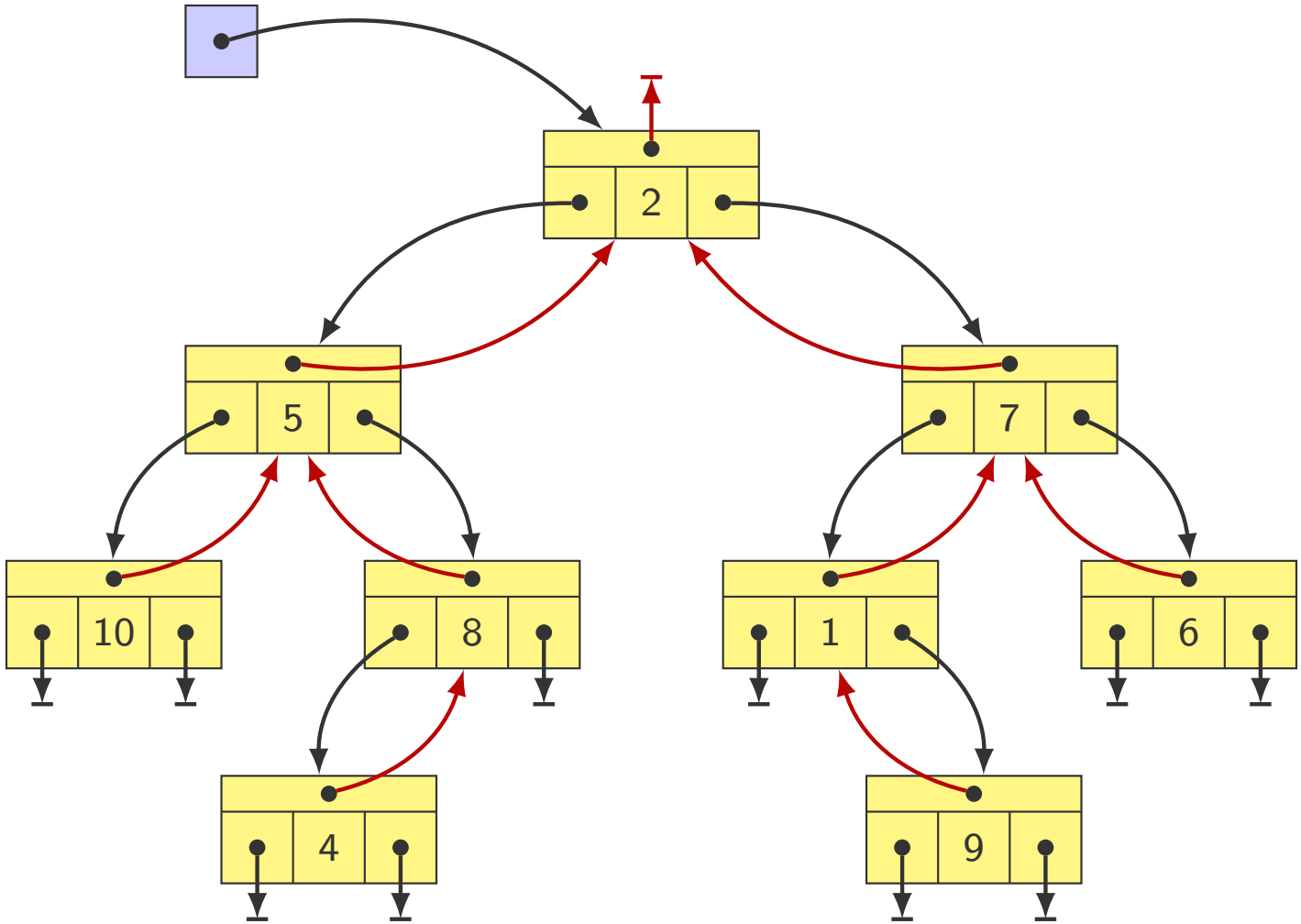
Exercício: Prove este lema. Dica: use indução forte no número de nós da árvore. (Resposta no final do slide)

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 atributos:
  - um valor (chave a ser guardada)
  - um ponteiro para o filho esquerdo do nó
  - um ponteiro para o filho direito do nó

- 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++

```
1  #ifndef NODE_H
2  #define NODE_H
3  #include <iostream>
4
5  struct Node
6  {
7      // atributos
8      int key;
9      Node *left;
10     Node *right;
11
12     // Construtor
13     Node(int key, Node *left, Node *right)
14     {
15         this->key = key;
16         this->left = left;
17         this->right = right;
18     }
19 };
20
21 #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

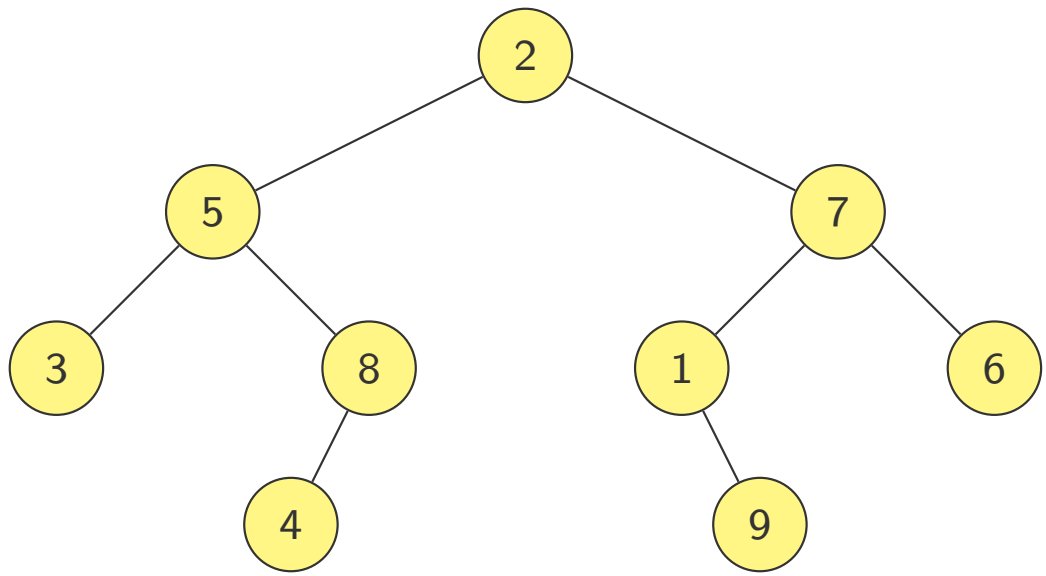
A pré-ordem

- 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

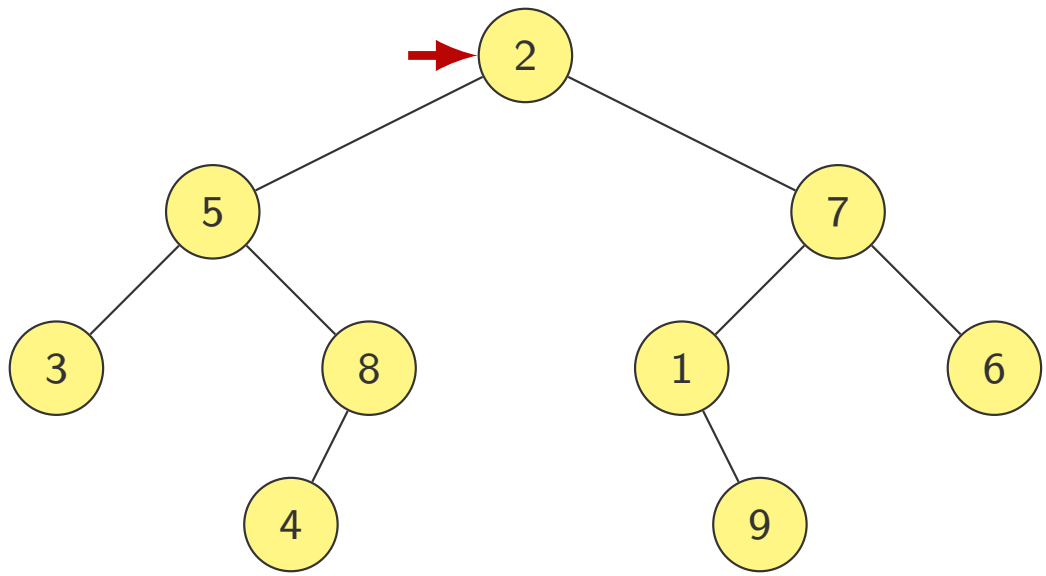


Ex:

Percorrendo os nós — Pré-ordem

A pré-ordem

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

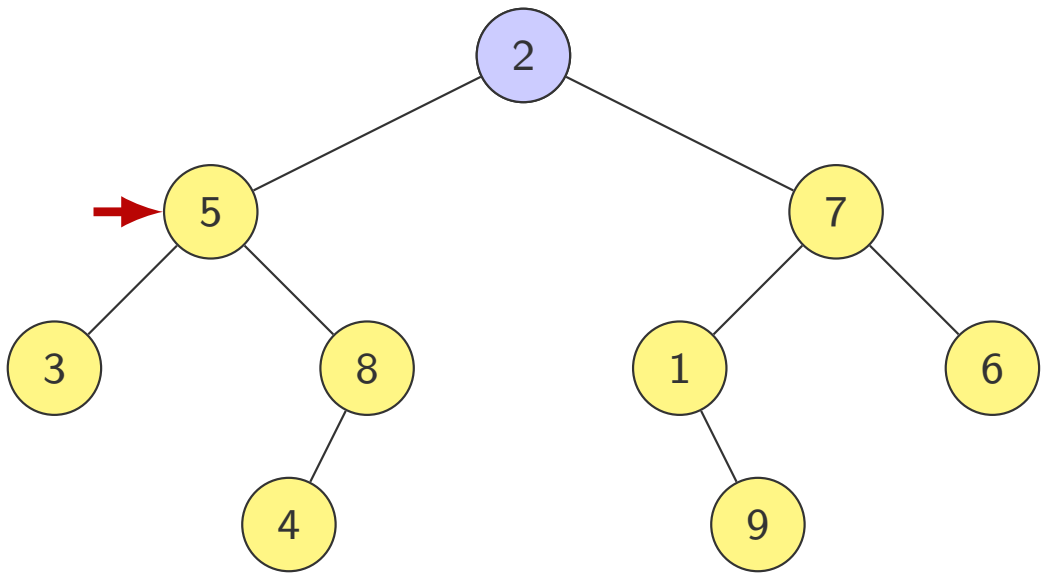


Ex:

Percorrendo os nós — Pré-ordem

A pré-ordem

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

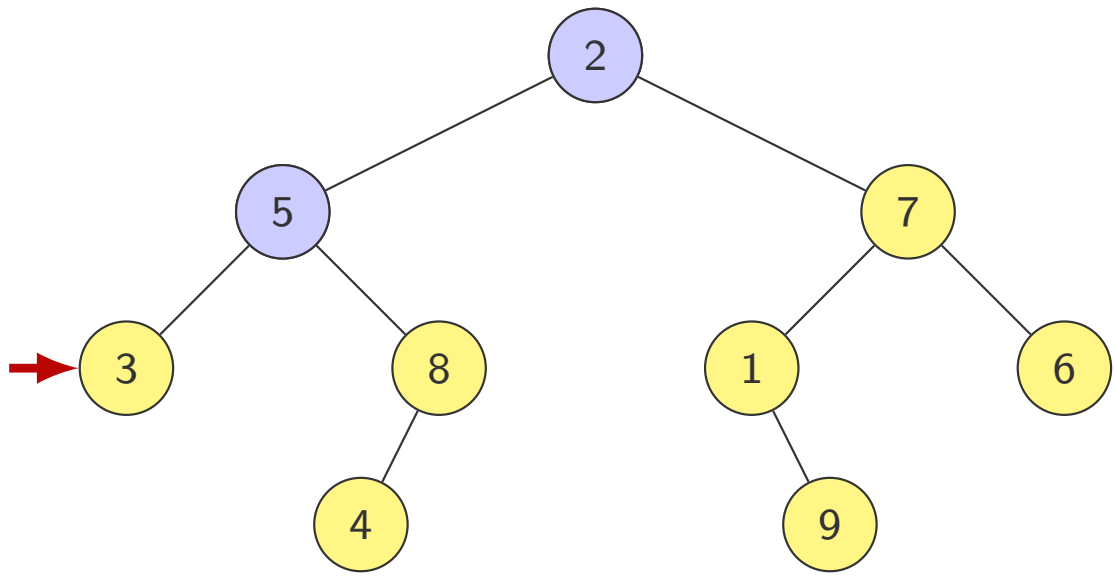


Ex: 2,

Percorrendo os nós — Pré-ordem

A pré-ordem

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

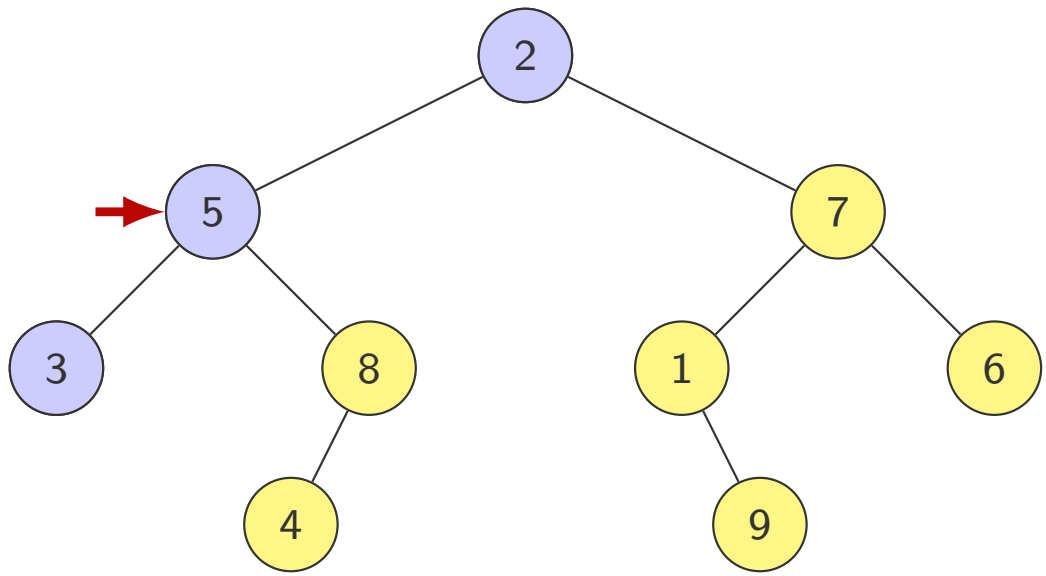


Ex: 2, 5,

Percorrendo os nós — Pré-ordem

A pré-ordem

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

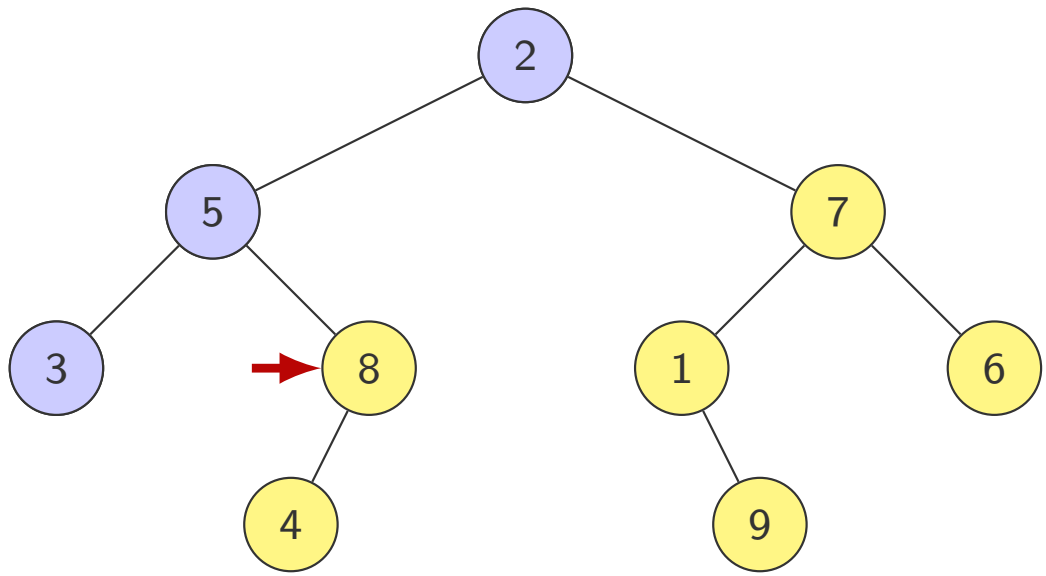


Ex: 2, 5, 3,

Percorrendo os nós — Pré-ordem

A pré-ordem

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

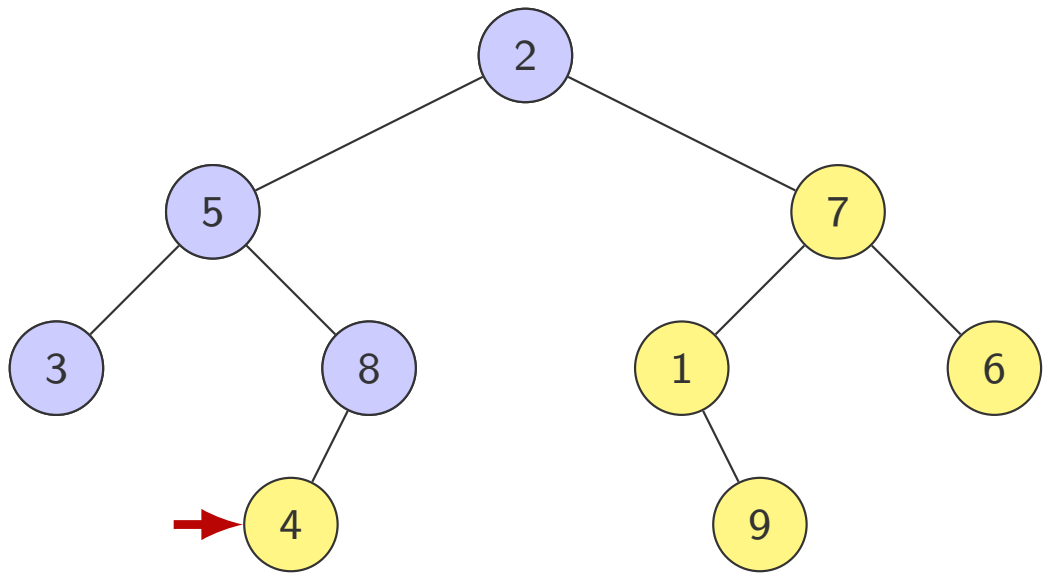


Ex: 2, 5, 3,

Percorrendo os nós — Pré-ordem

A pré-ordem

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

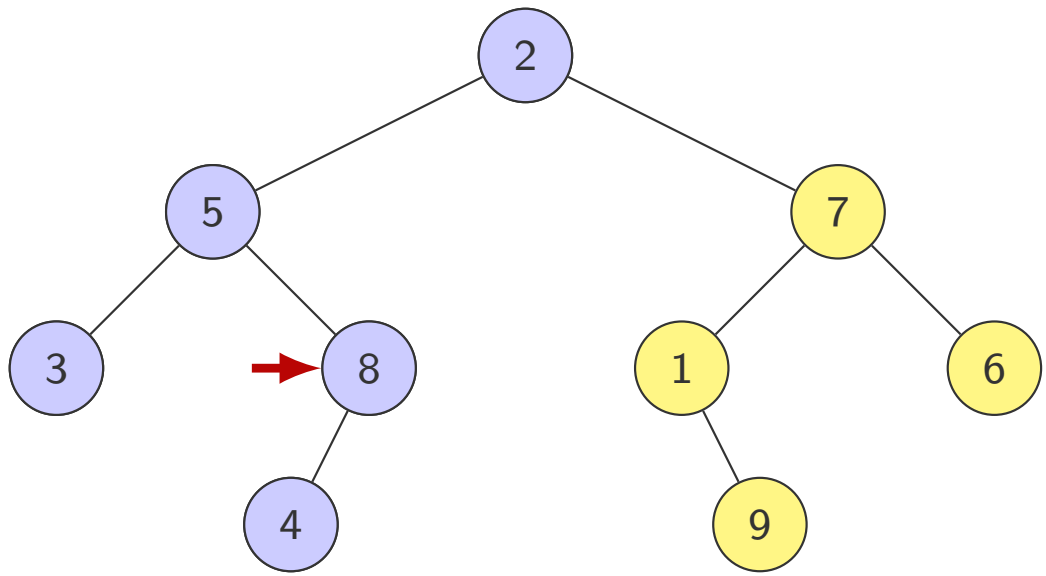


Ex: 2, 5, 3, 8,

Percorrendo os nós — Pré-ordem

A pré-ordem

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

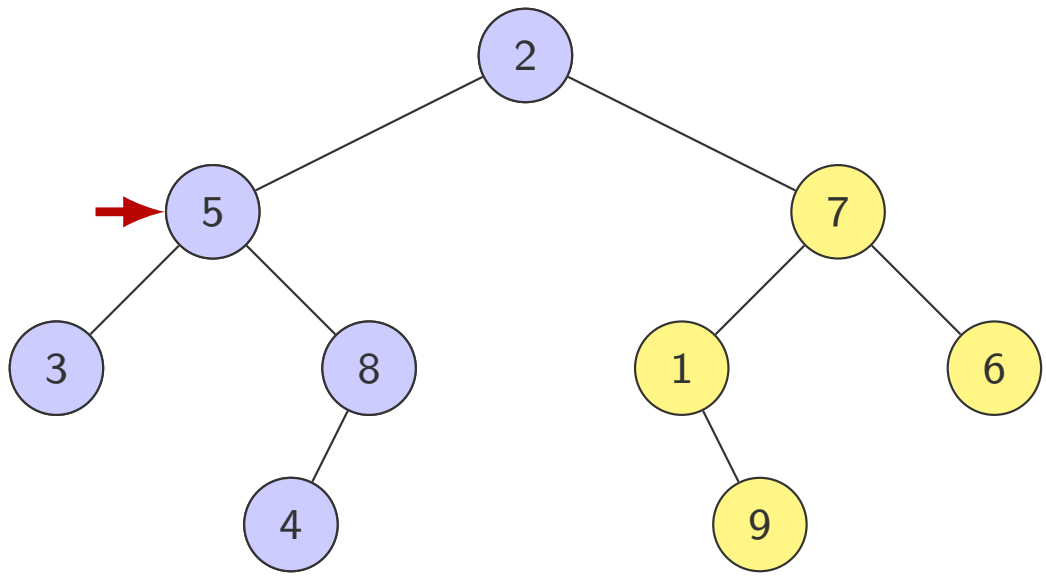


Ex: 2, 5, 3, 8, 4,

Percorrendo os nós — Pré-ordem

A pré-ordem

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

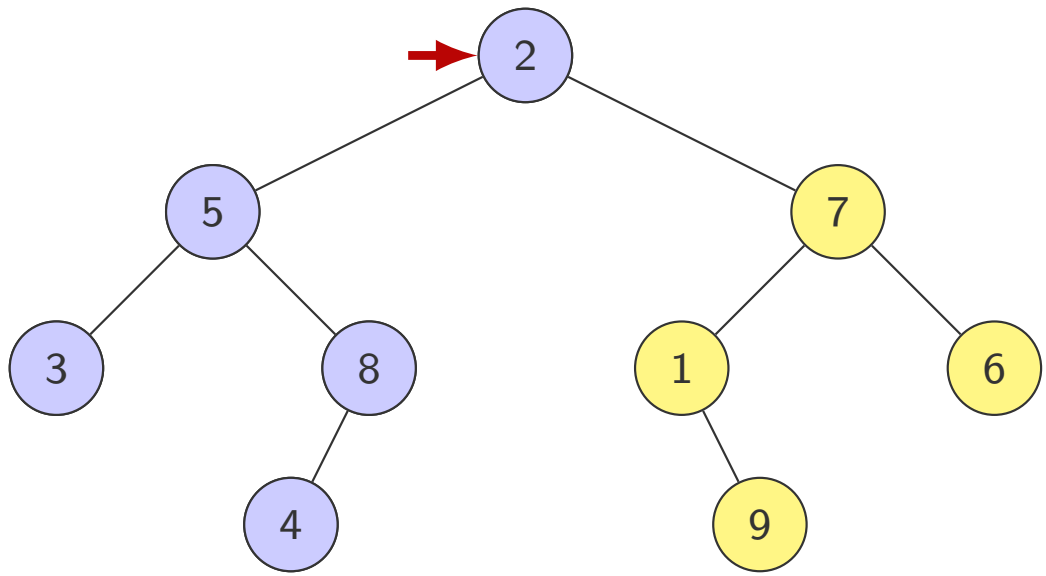


Ex: 2, 5, 3, 8, 4,

Percorrendo os nós — Pré-ordem

A pré-ordem

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

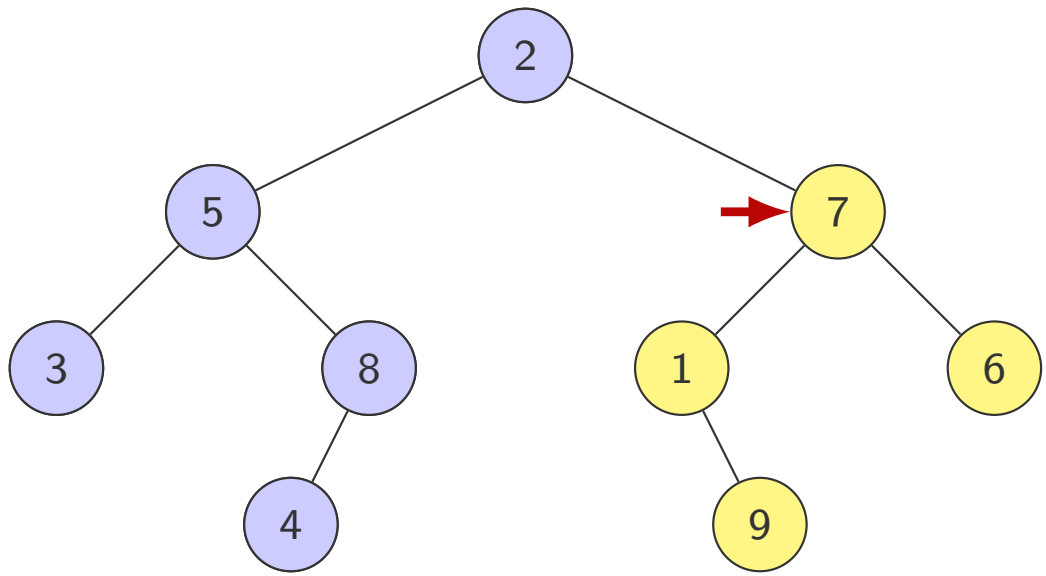


Ex: 2, 5, 3, 8, 4,

Percorrendo os nós — Pré-ordem

A pré-ordem

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

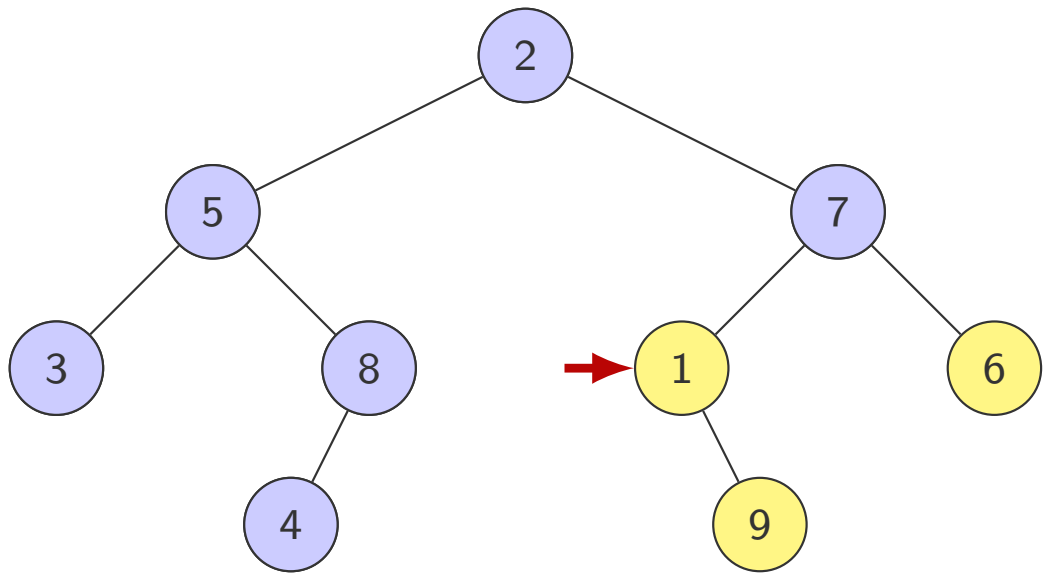


Ex: 2, 5, 3, 8, 4,

Percorrendo os nós — Pré-ordem

A pré-ordem

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

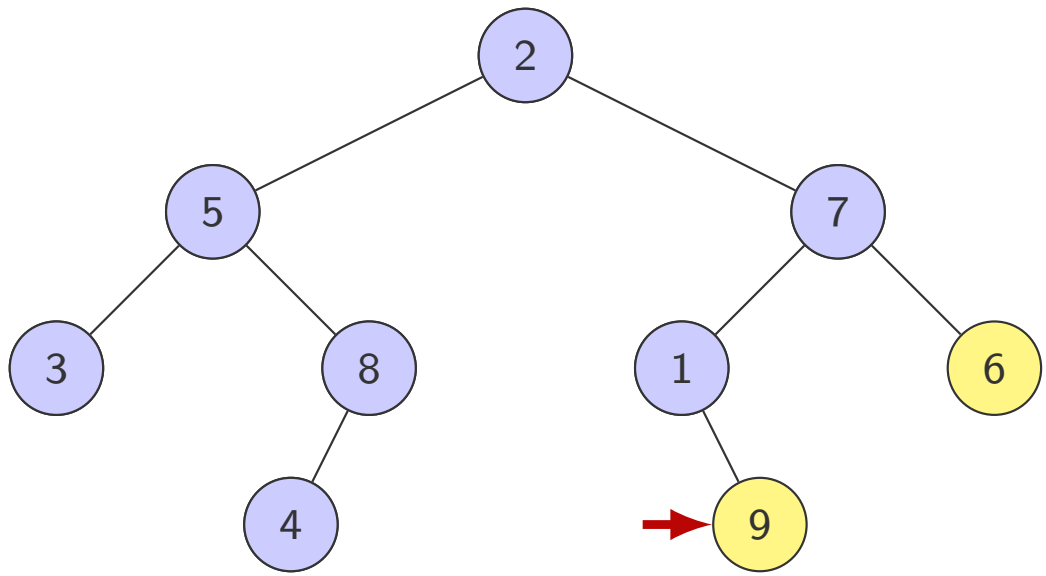


Ex: 2, 5, 3, 8, 4, 7,

Percorrendo os nós — Pré-ordem

A pré-ordem

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

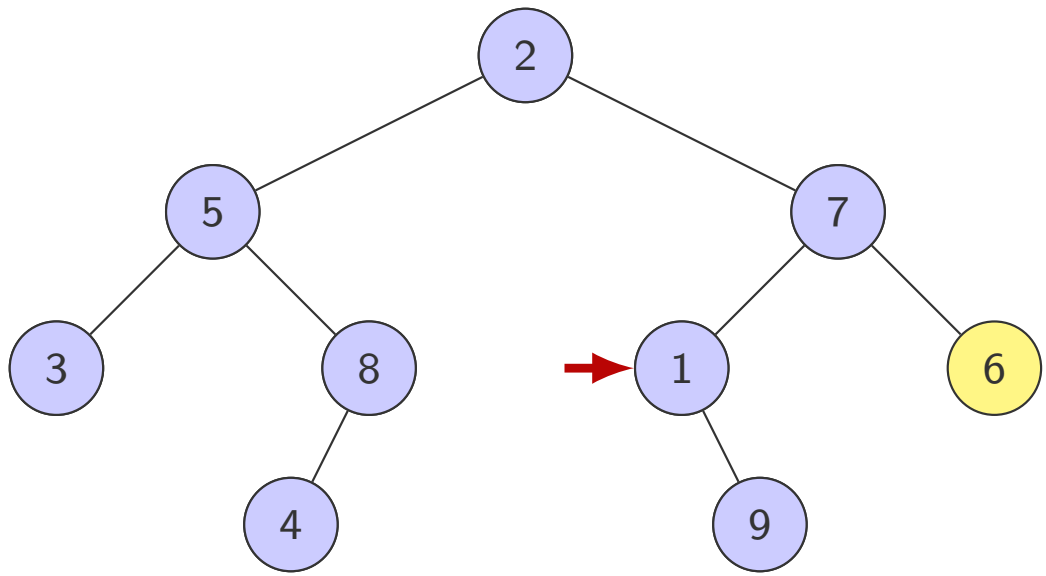


Ex: 2, 5, 3, 8, 4, 7, 1,

Percorrendo os nós — Pré-ordem

A pré-ordem

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

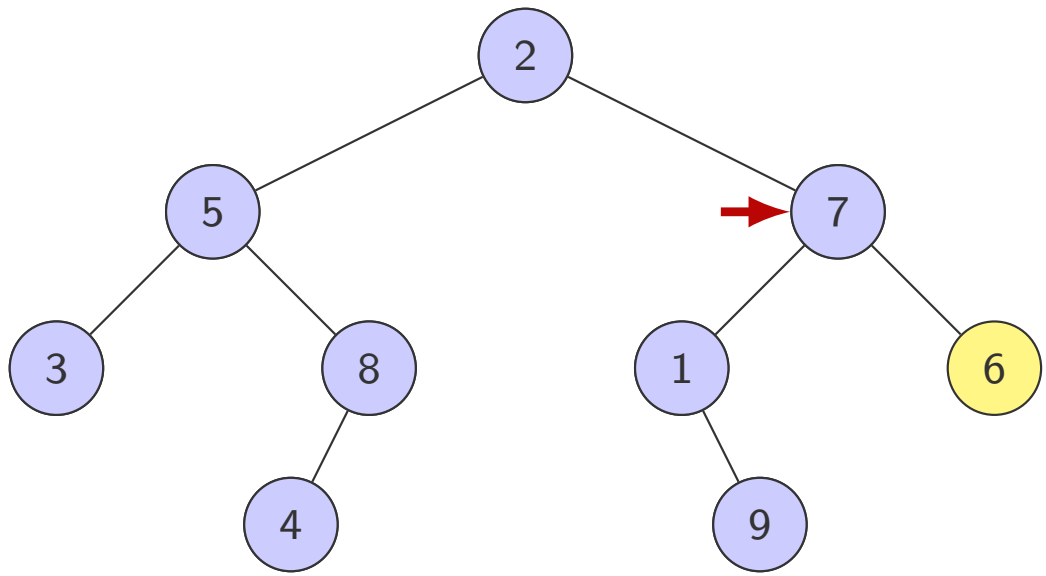


Ex: 2, 5, 3, 8, 4, 7, 1, 9,

Percorrendo os nós — Pré-ordem

A pré-ordem

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

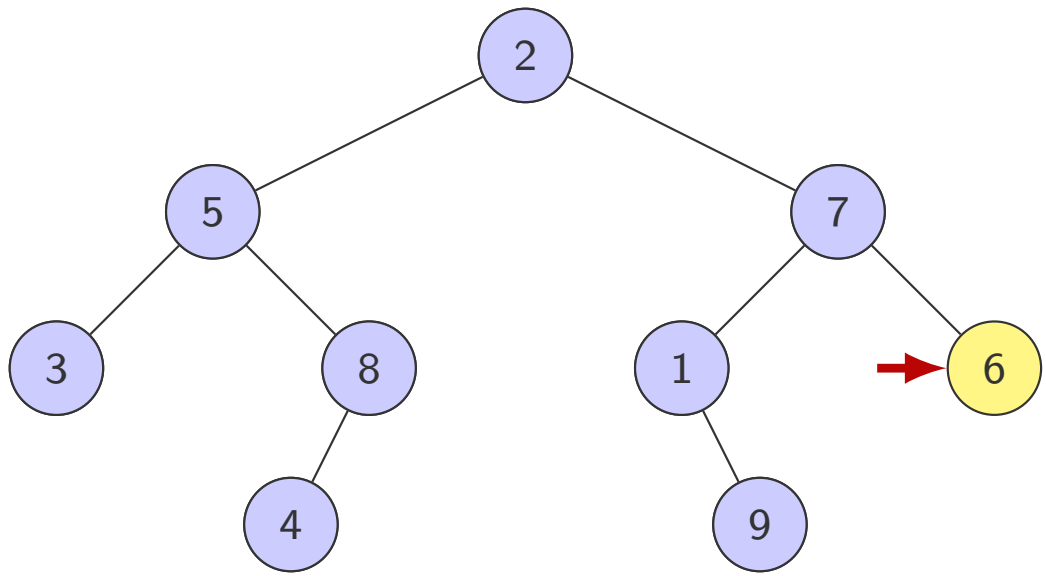


Ex: 2, 5, 3, 8, 4, 7, 1, 9,

Percorrendo os nós — Pré-ordem

A pré-ordem

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

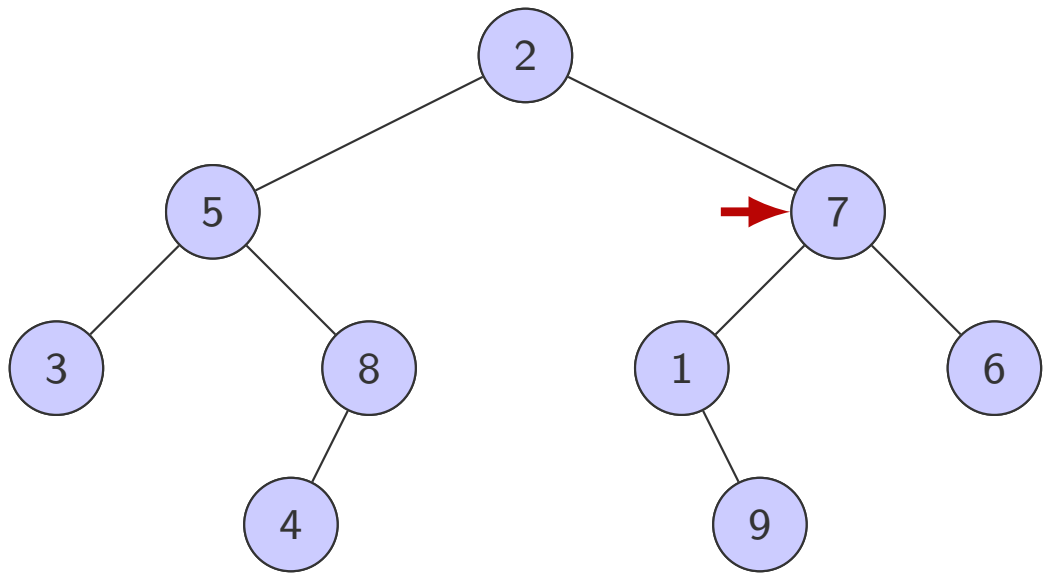


Ex: 2, 5, 3, 8, 4, 7, 1, 9,

Percorrendo os nós — Pré-ordem

A pré-ordem

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

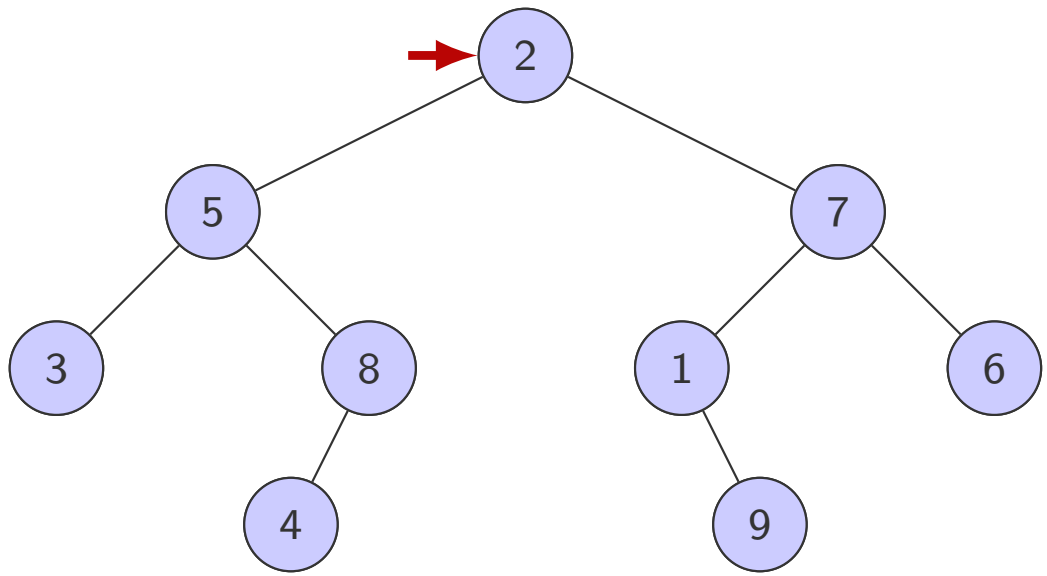


Ex: 2, 5, 3, 8, 4, 7, 1, 9, 6

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)

```
1: if ptr ≠ NULL then
2:     visit(ptr)
3:     preorder(ptr→left)
4:     preorder(ptr→right)
5: 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

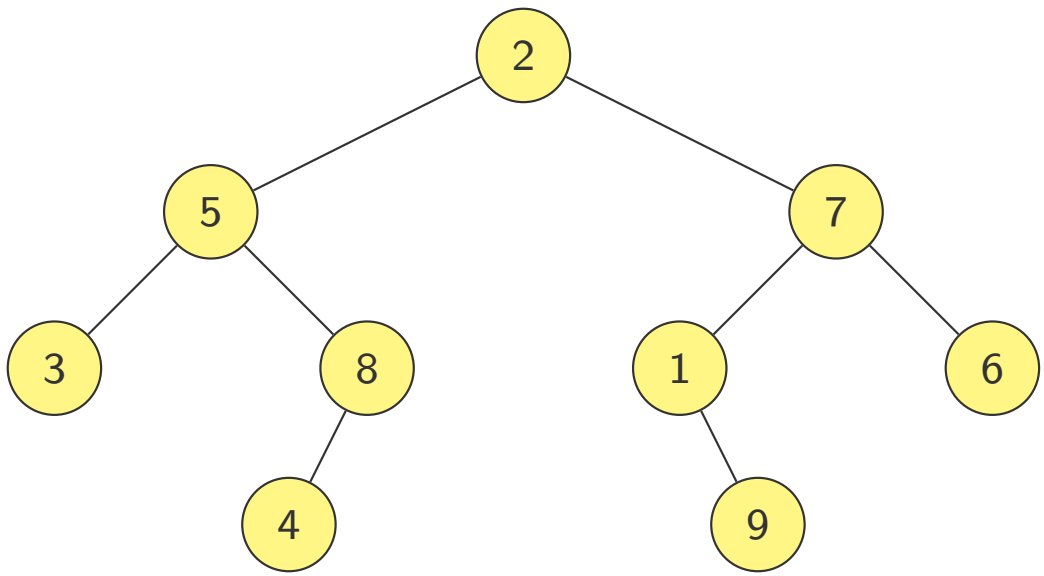
A pós-ordem

- 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

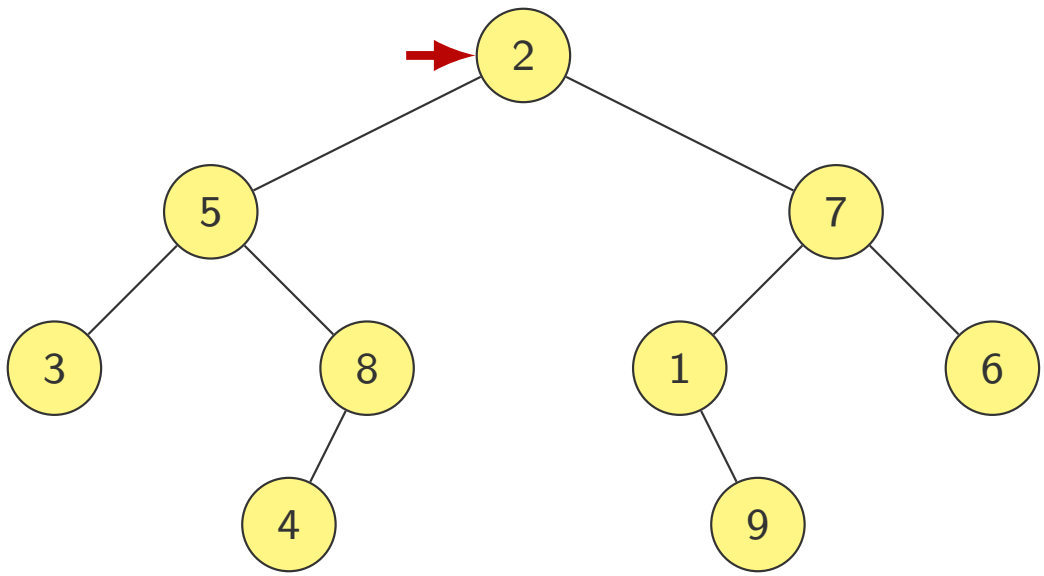


Ex:

Percorrendo os nós — Pós-ordem

A pós-ordem

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

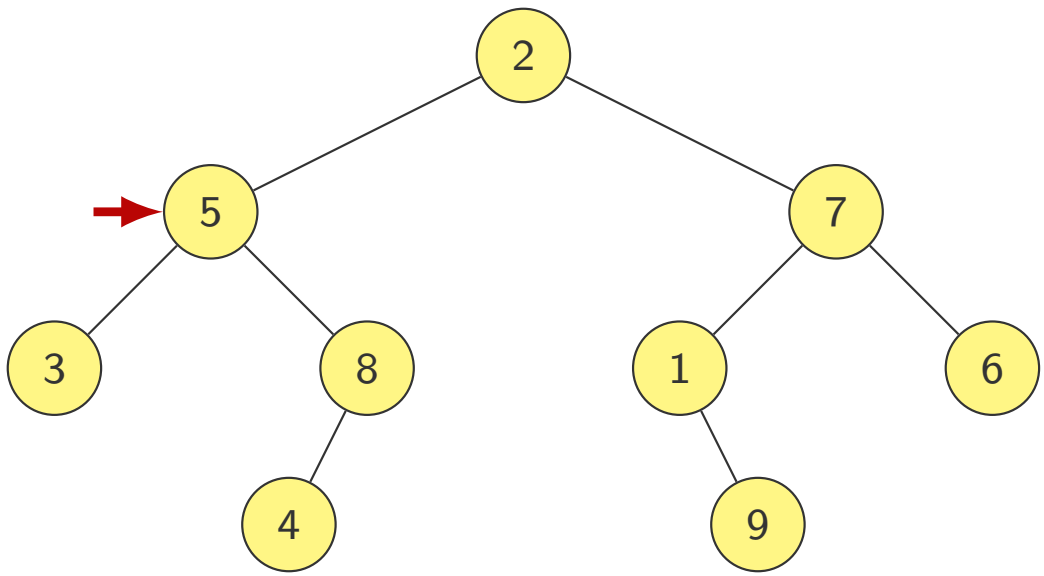


Ex:

Percorrendo os nós — Pós-ordem

A pós-ordem

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

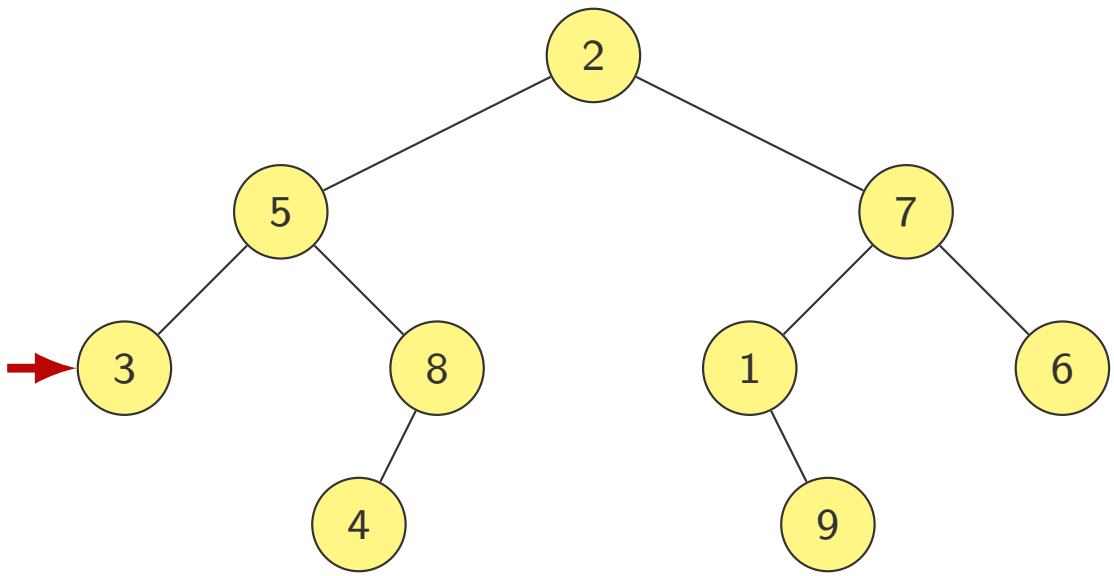


Ex:

Percorrendo os nós — Pós-ordem

A pós-ordem

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

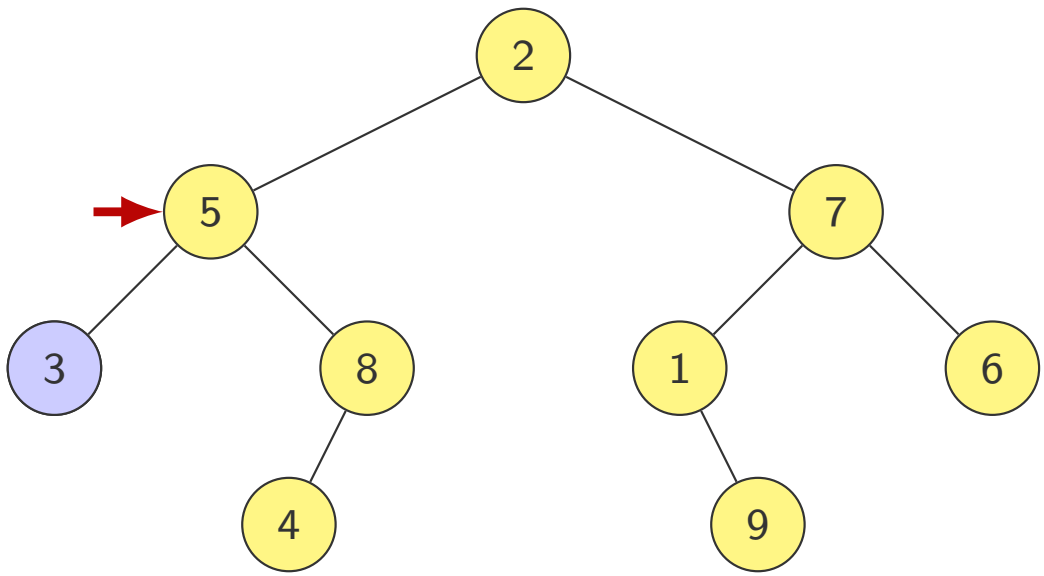


Ex:

Percorrendo os nós — Pós-ordem

A pós-ordem

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

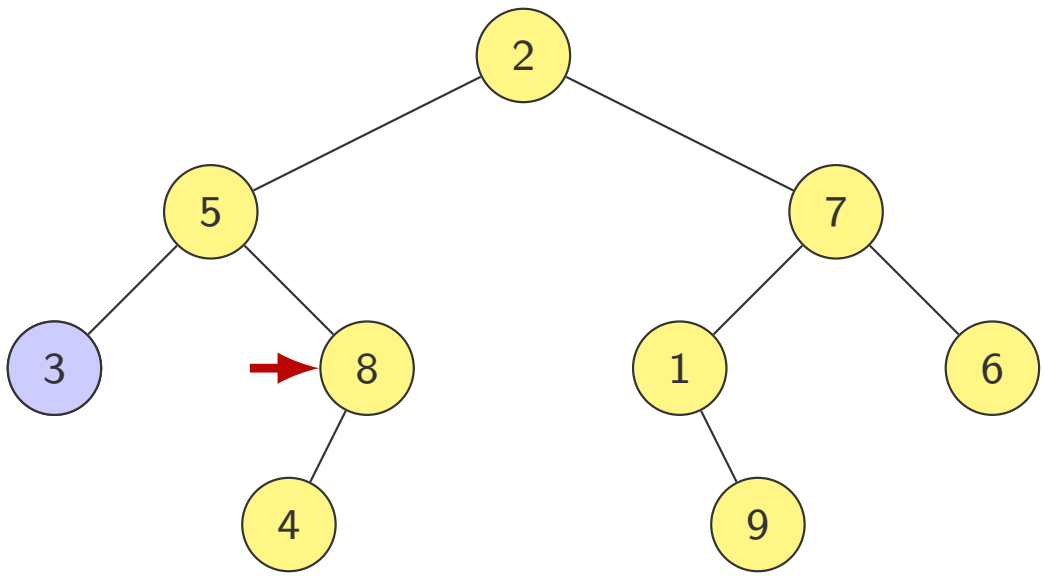


Ex: 3,

Percorrendo os nós — Pós-ordem

A pós-ordem

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

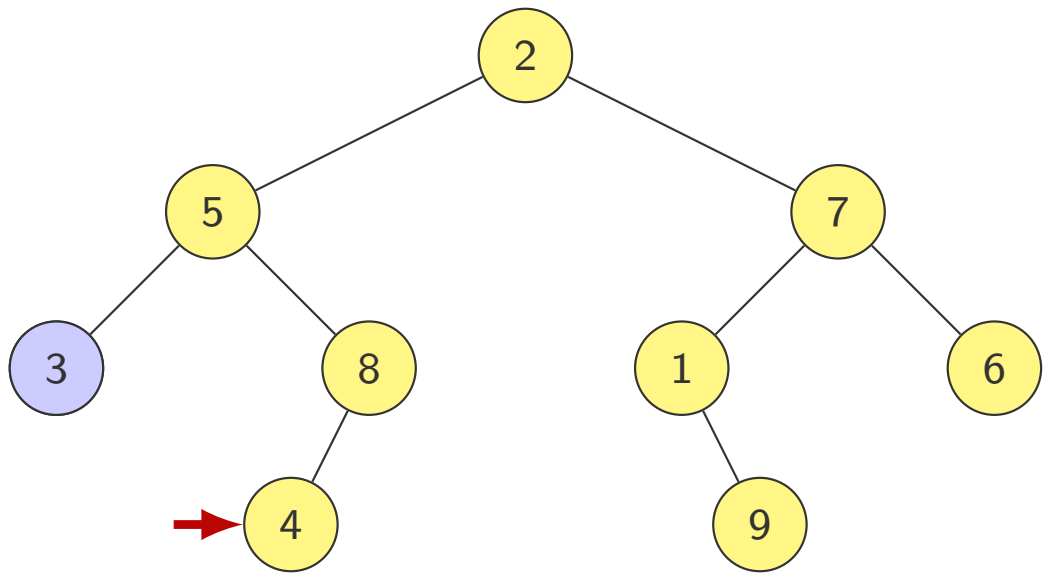


Ex: 3,

Percorrendo os nós — Pós-ordem

A pós-ordem

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

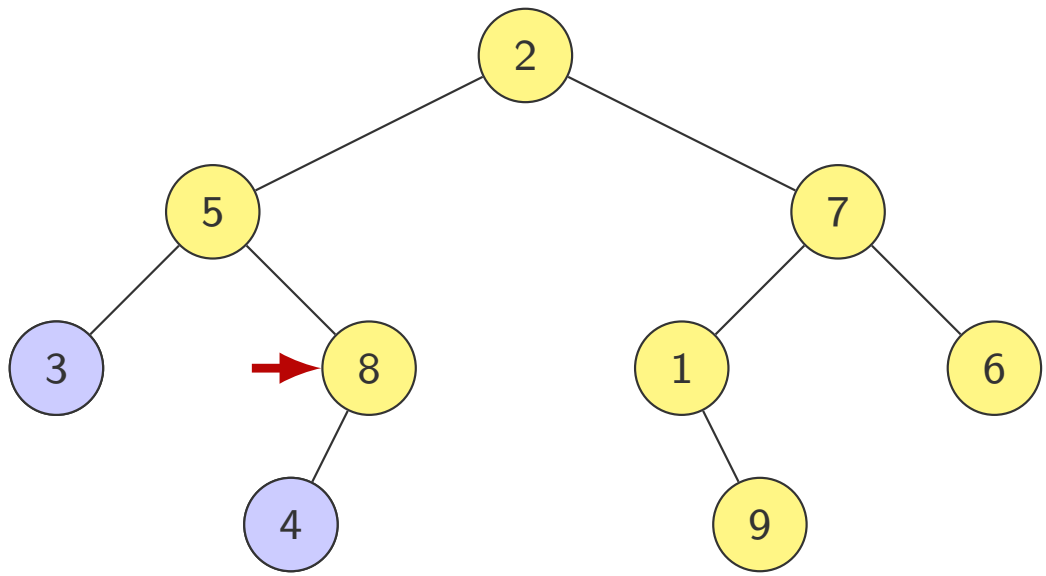


Ex: 3,

Percorrendo os nós — Pós-ordem

A pós-ordem

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

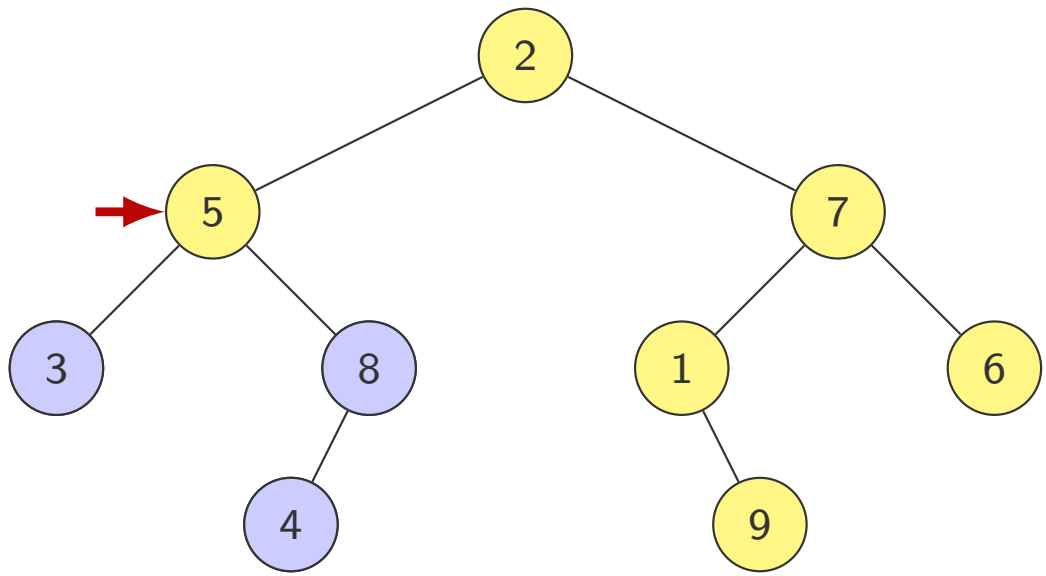


Ex: 3, 4,

Percorrendo os nós — Pós-ordem

A pós-ordem

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

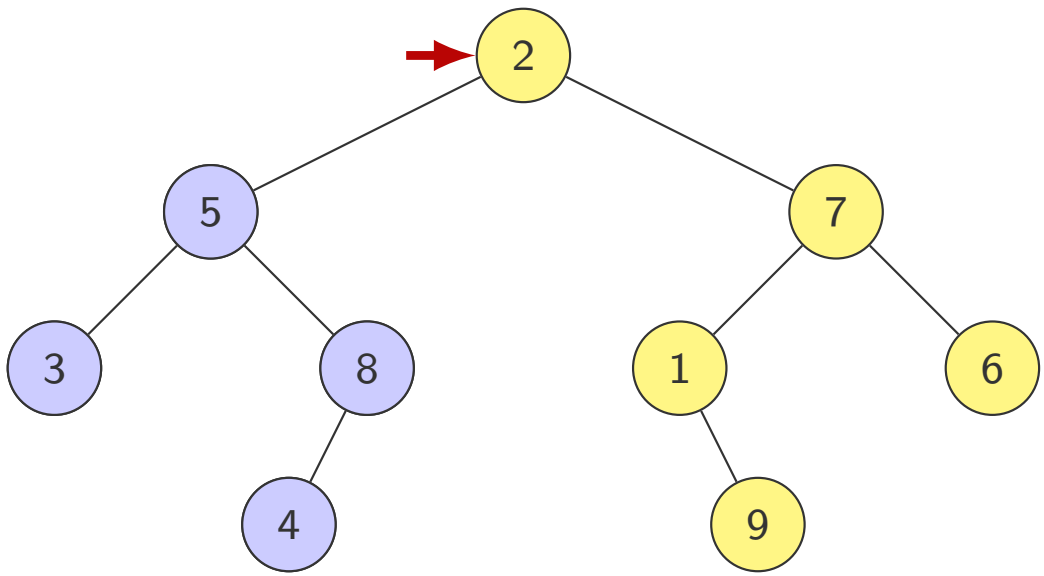


Ex: 3, 4, 8,

Percorrendo os nós — Pós-ordem

A pós-ordem

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

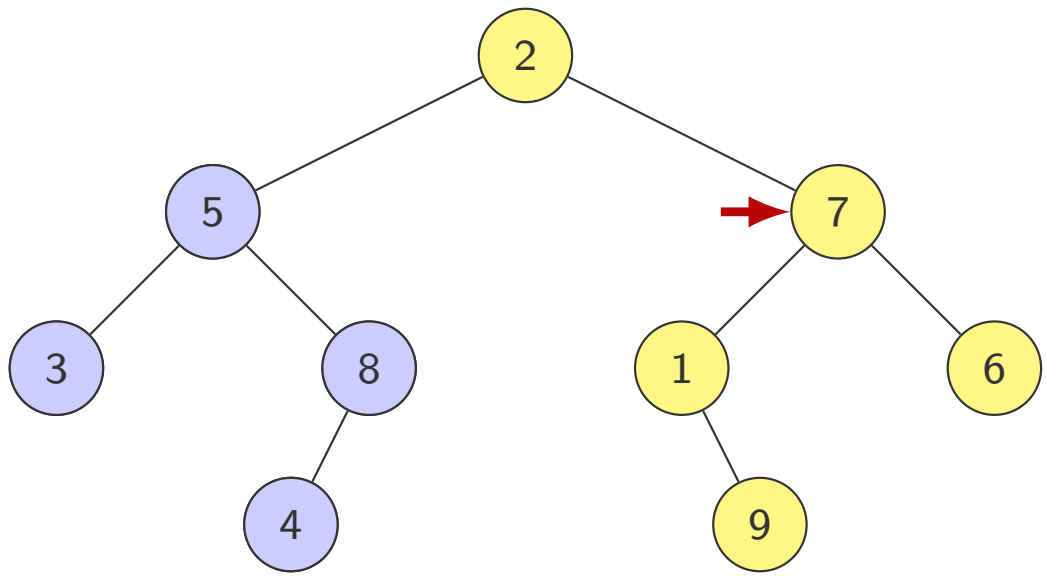


Ex: 3, 4, 8, 5,

Percorrendo os nós — Pós-ordem

A pós-ordem

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

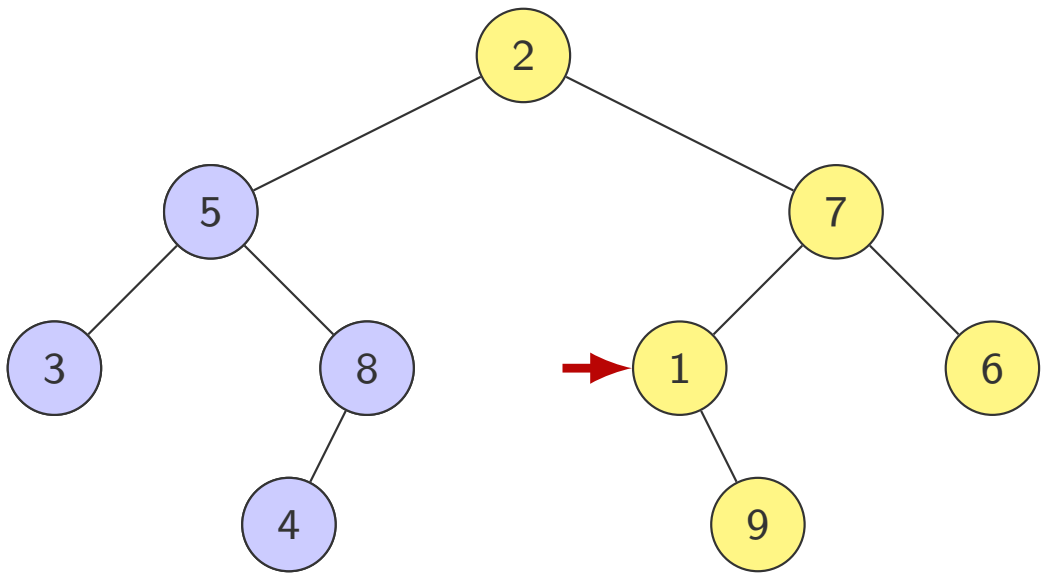


Ex: 3, 4, 8, 5,

Percorrendo os nós — Pós-ordem

A pós-ordem

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

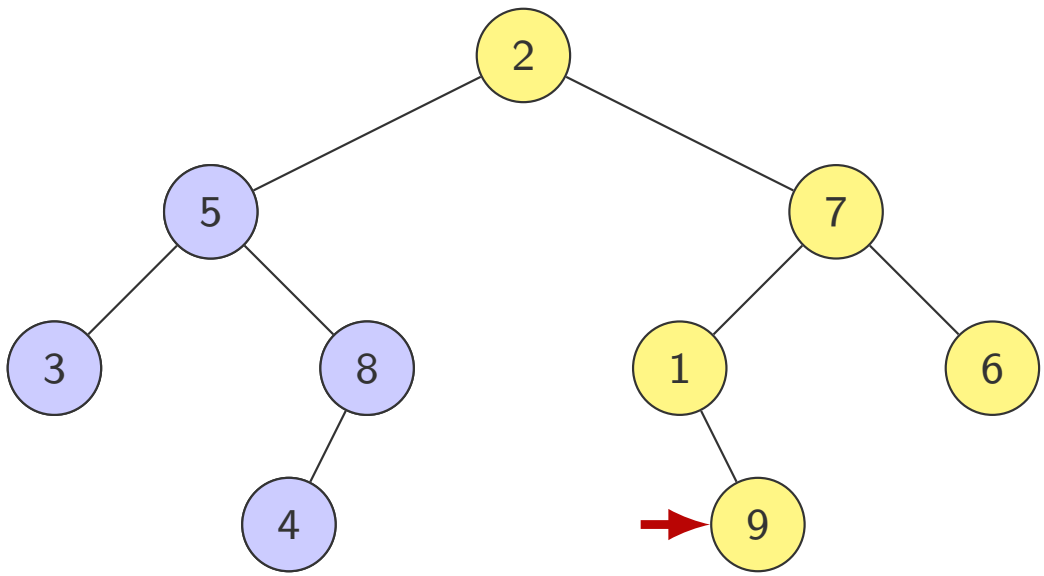


Ex: 3, 4, 8, 5,

Percorrendo os nós — Pós-ordem

A pós-ordem

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

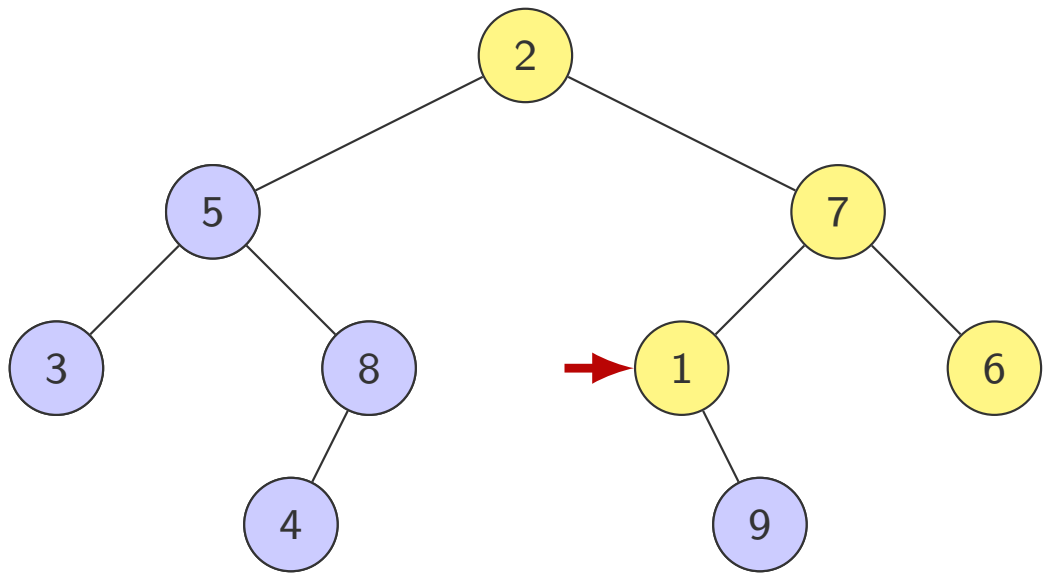


Ex: 3, 4, 8, 5,

Percorrendo os nós — Pós-ordem

A pós-ordem

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

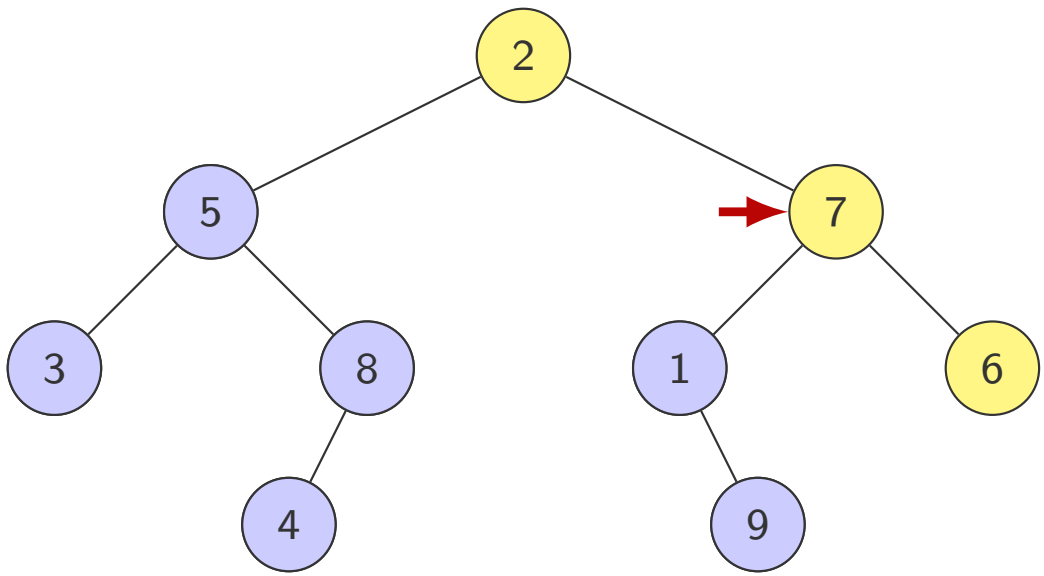


Ex: 3, 4, 8, 5, 9,

Percorrendo os nós — Pós-ordem

A pós-ordem

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

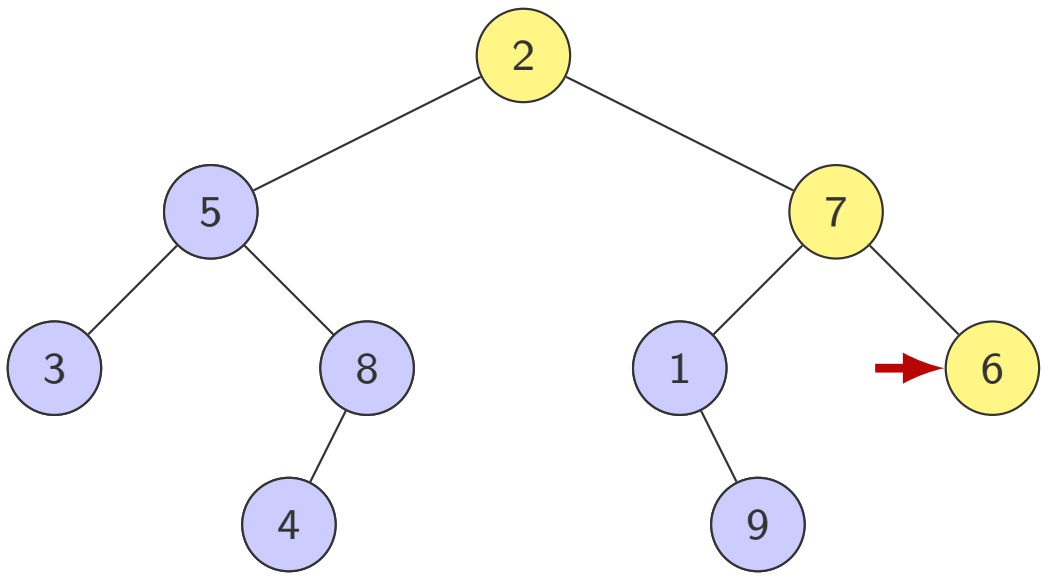


Ex: 3, 4, 8, 5, 9, 1,

Percorrendo os nós — Pós-ordem

A pós-ordem

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

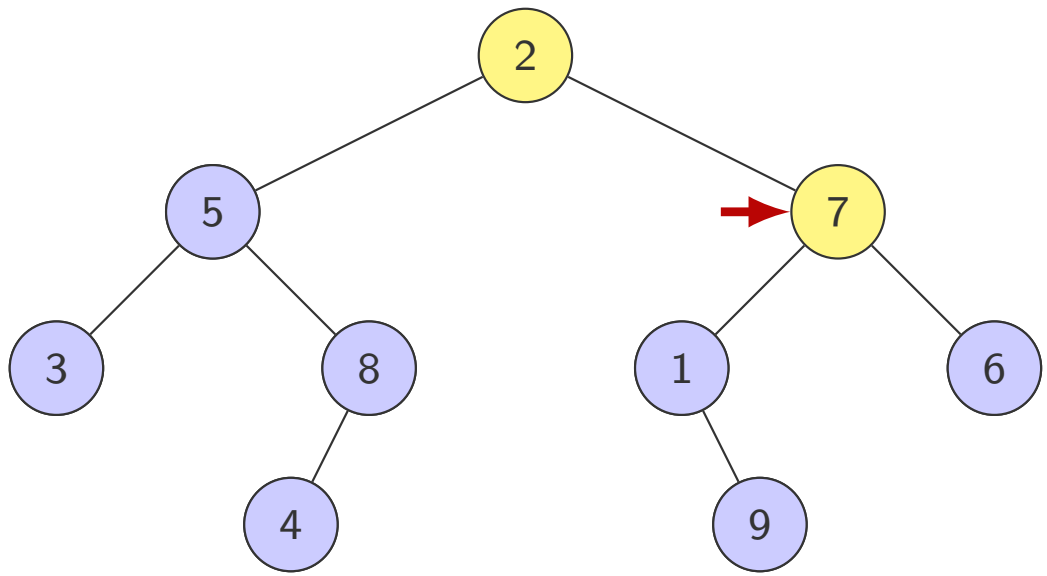


Ex: 3, 4, 8, 5, 9, 1,

Percorrendo os nós — Pós-ordem

A pós-ordem

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

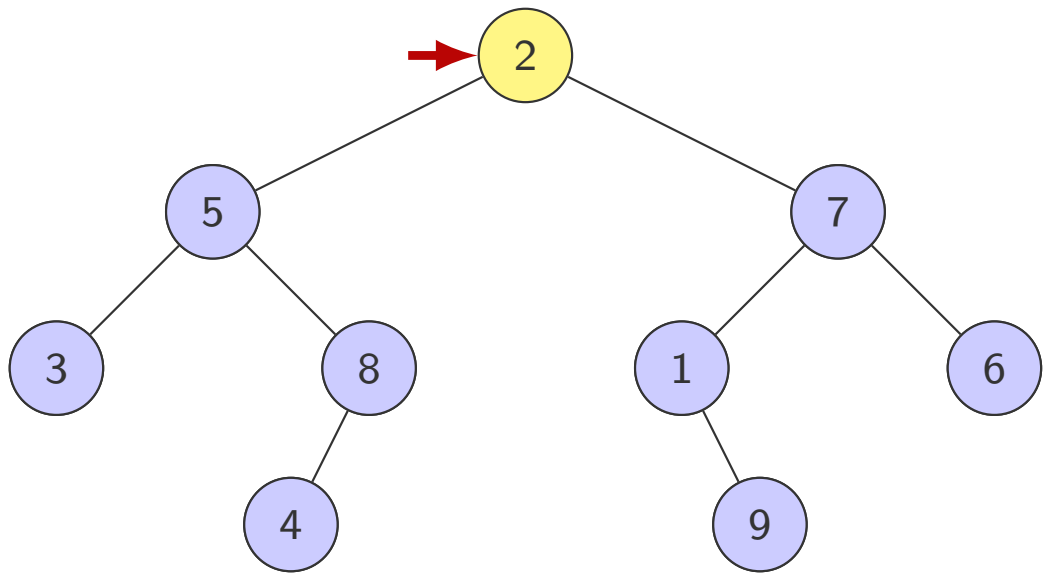


Ex: 3, 4, 8, 5, 9, 1, 6,

Percorrendo os nós — Pós-ordem

A pós-ordem

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

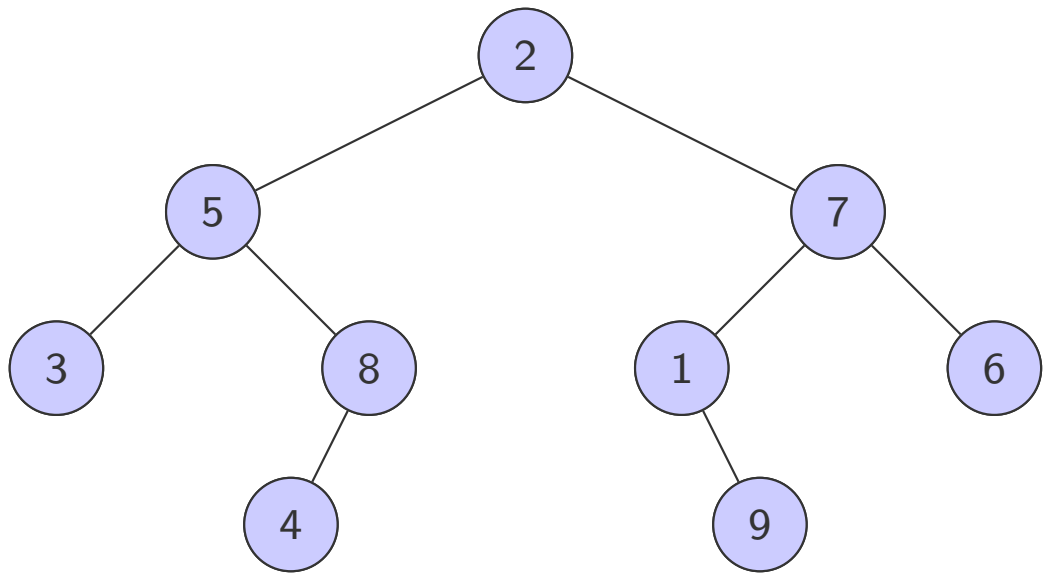


Ex: 3, 4, 8, 5, 9, 1, 6, 7,

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)

```
1: if ptr ≠ NULL then
2:     posorder(ptr→left)
3:     posorder(ptr→right)
4:     visit(ptr)
5: 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

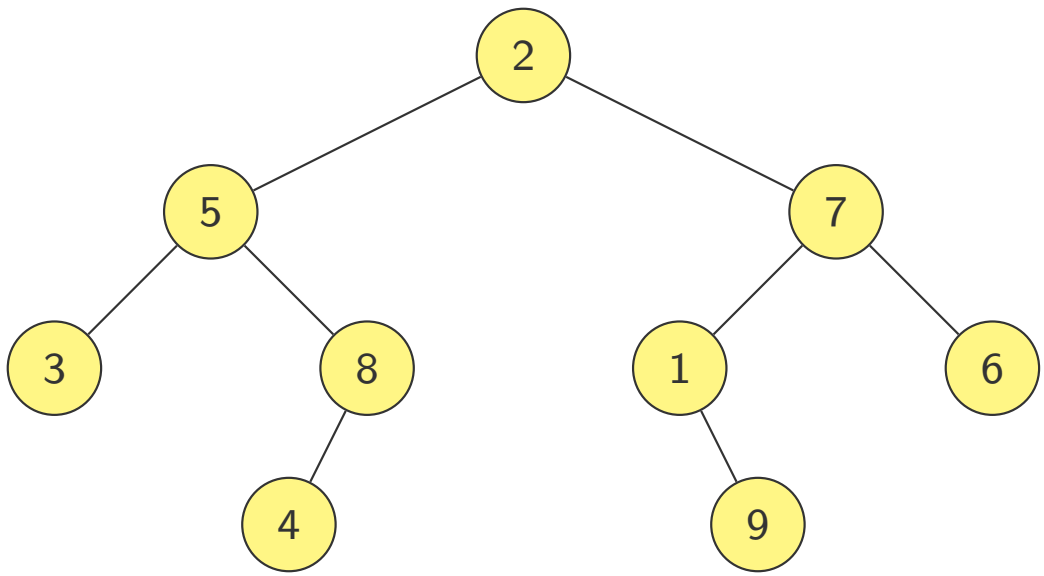
A ordem simétrica

- 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

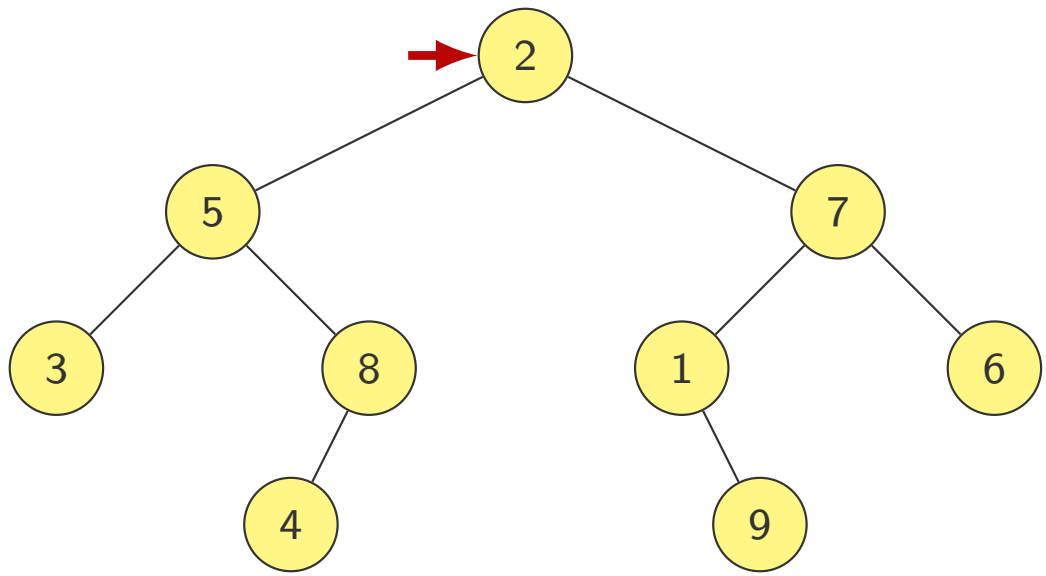


Ex:

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

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

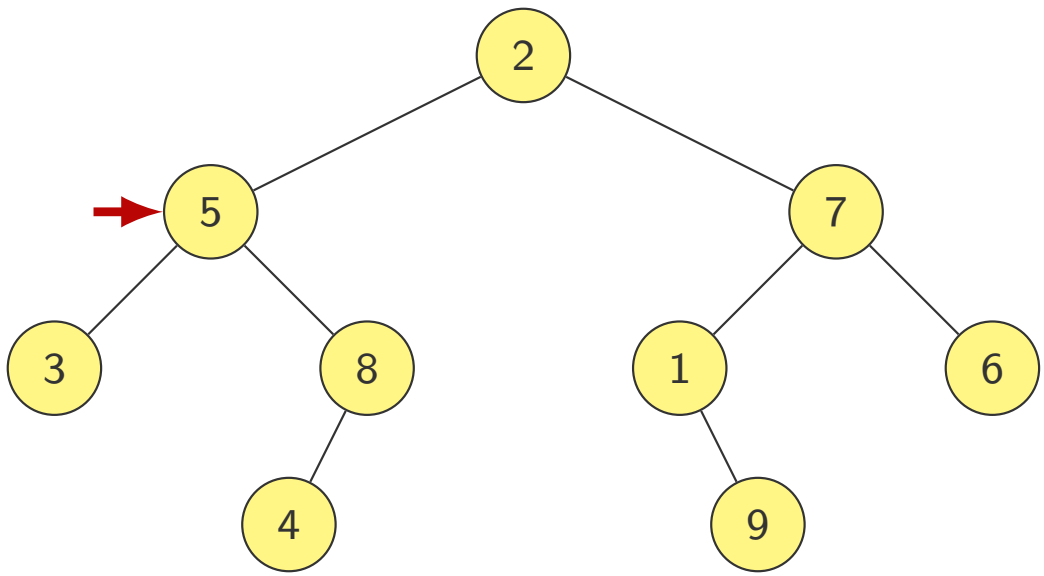


Ex:

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

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

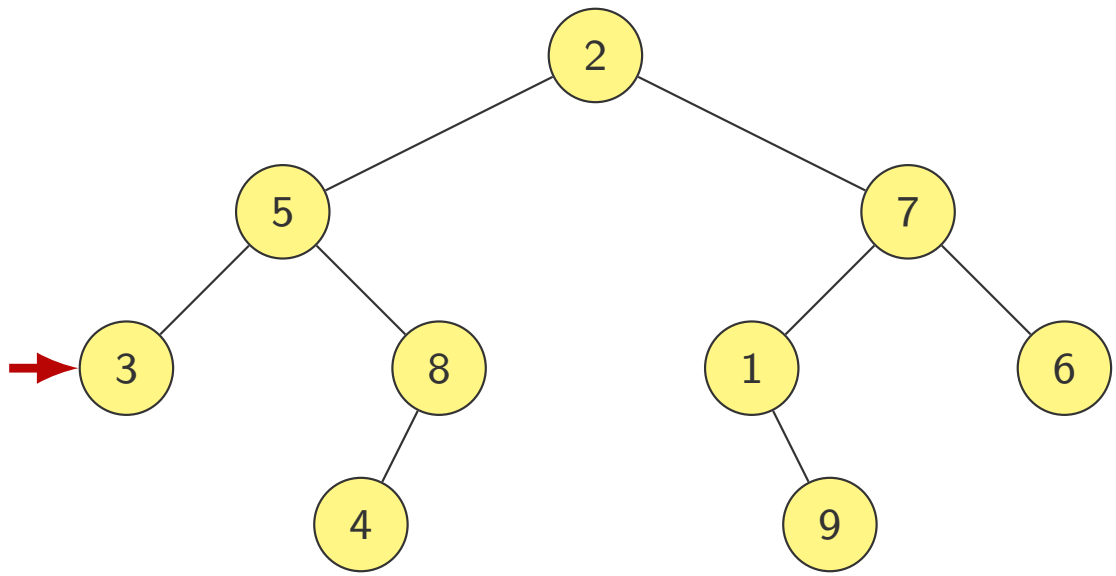


Ex:

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

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

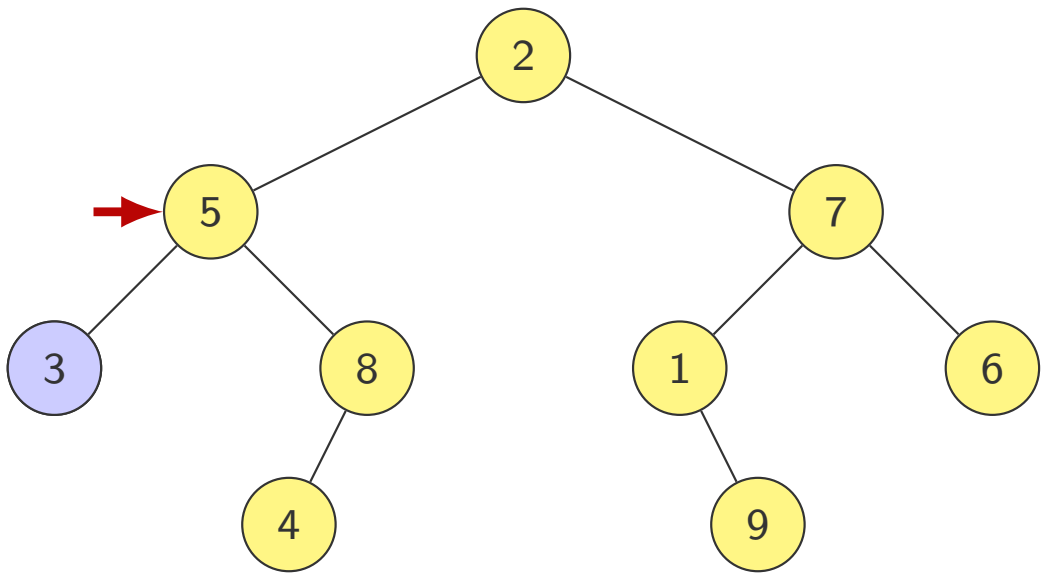


Ex:

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

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

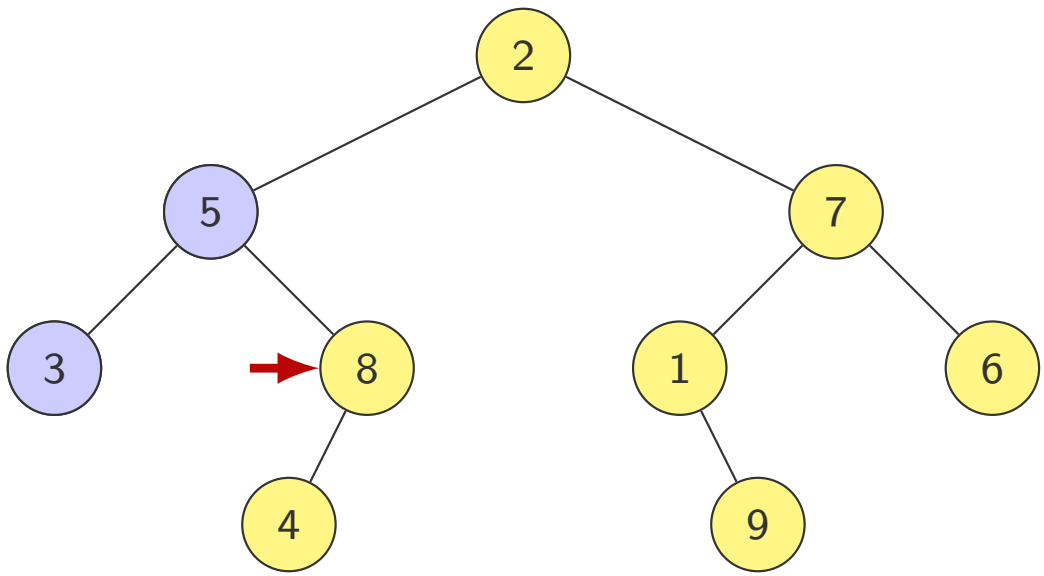


Ex: 3,

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

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

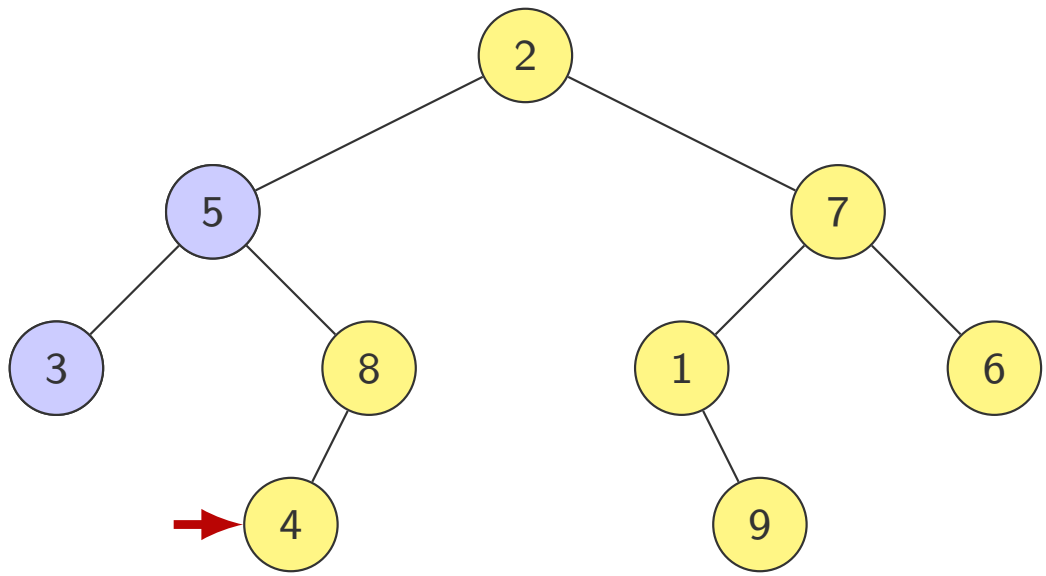


Ex: 3, 5,

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

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

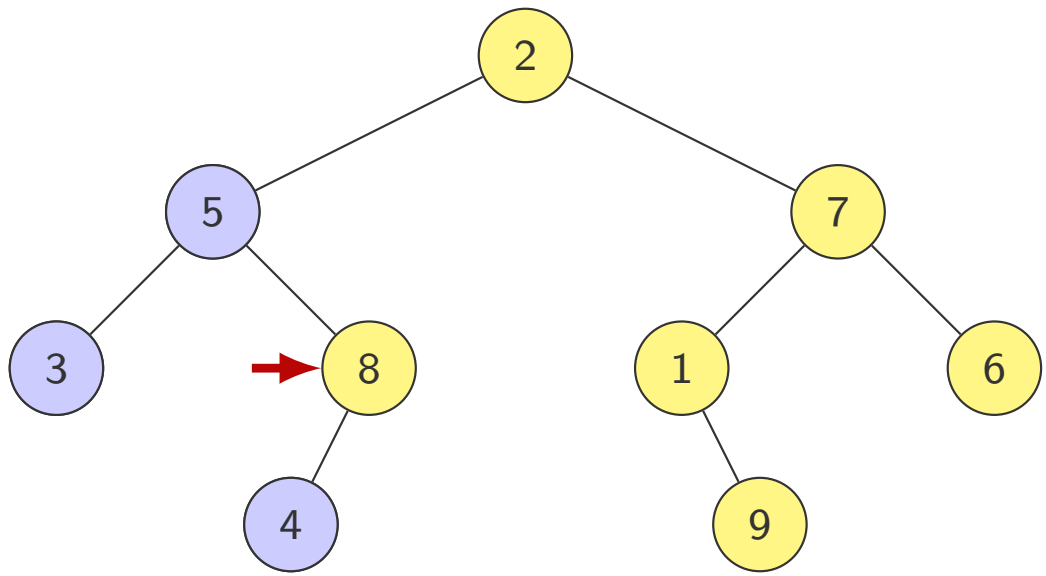


Ex: 3, 5,

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

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

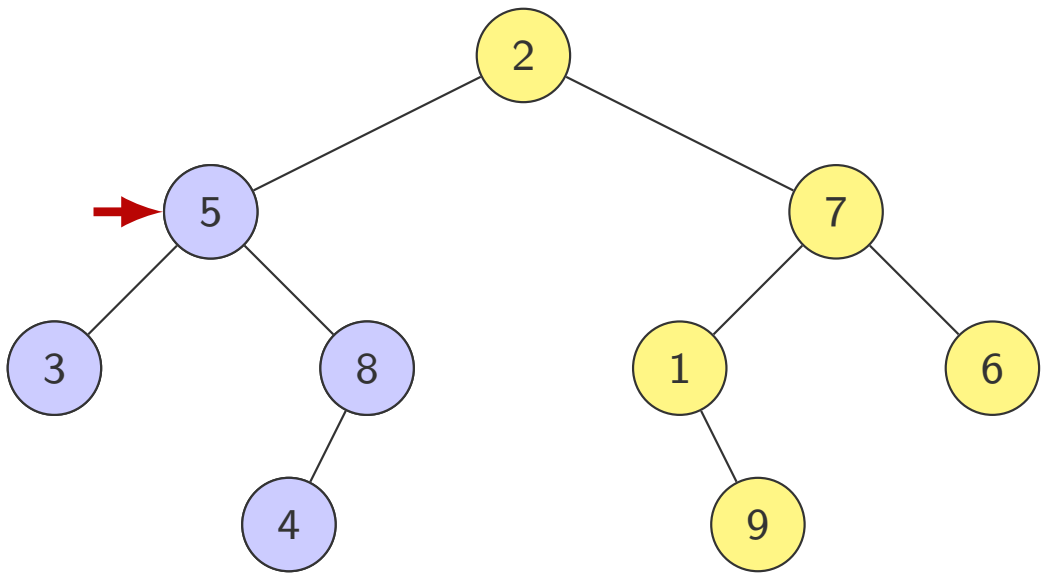


Ex: 3, 5, 4,

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

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

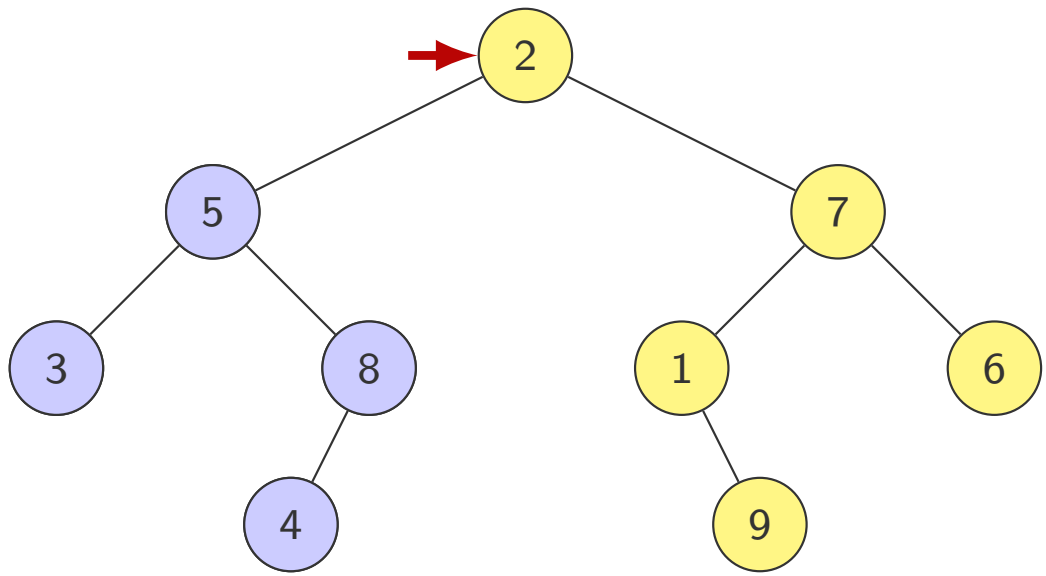


Ex: 3, 5, 4, 8,

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

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

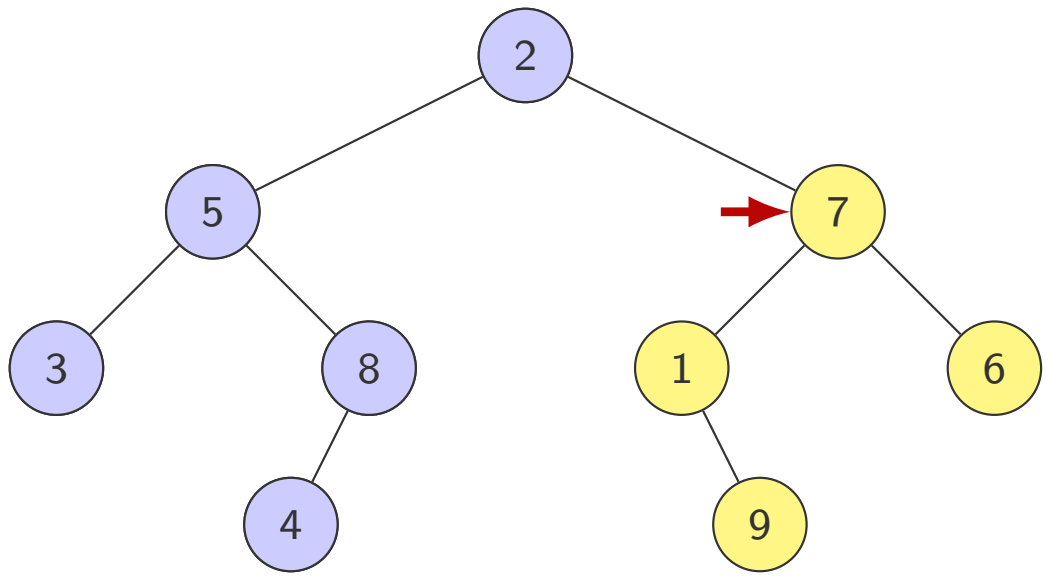


Ex: 3, 5, 4, 8,

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

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

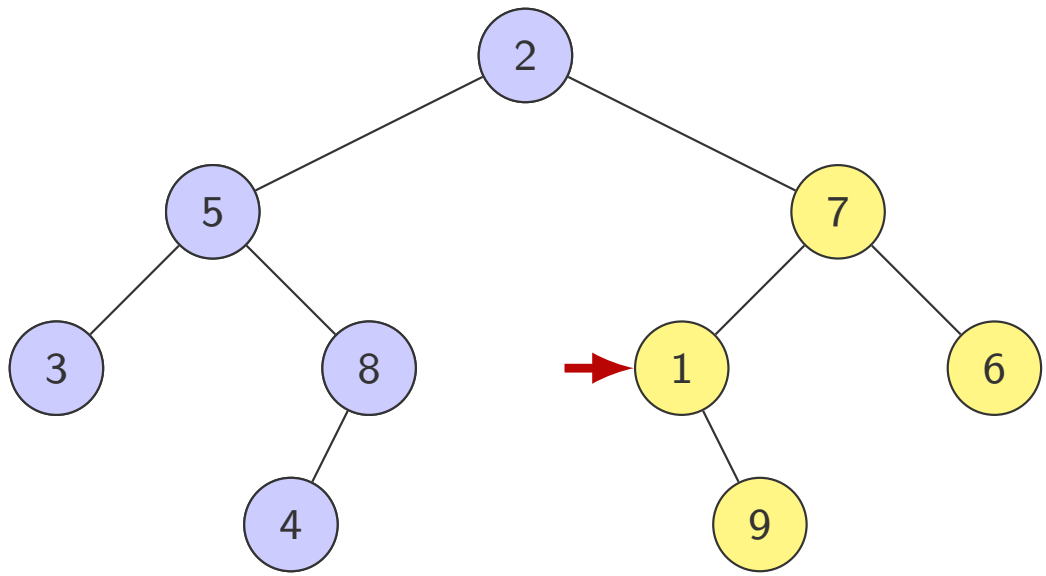


Ex: 3, 5, 4, 8, 2,

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

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

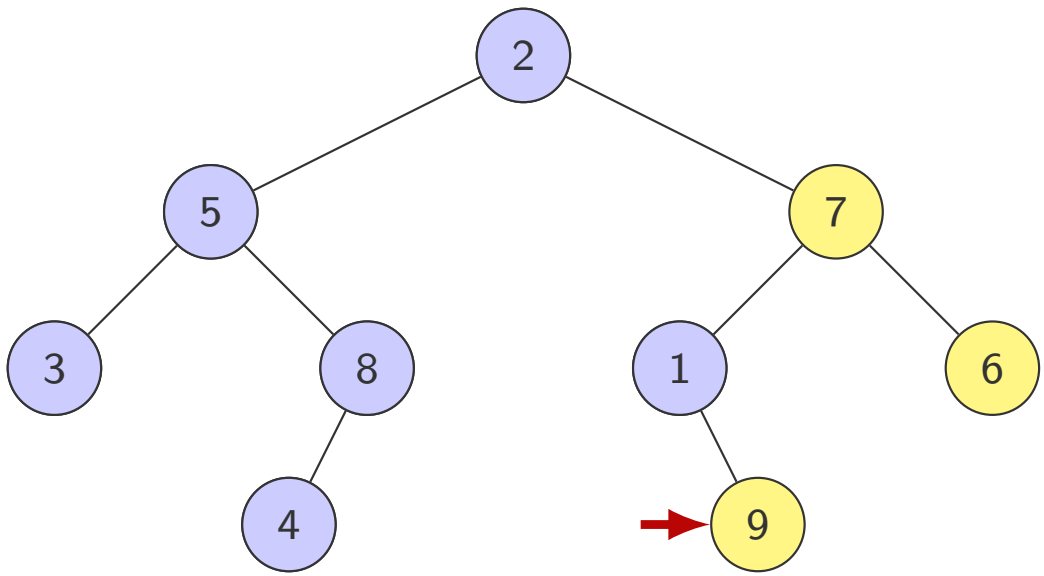


Ex: 3, 5, 4, 8, 2,

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

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

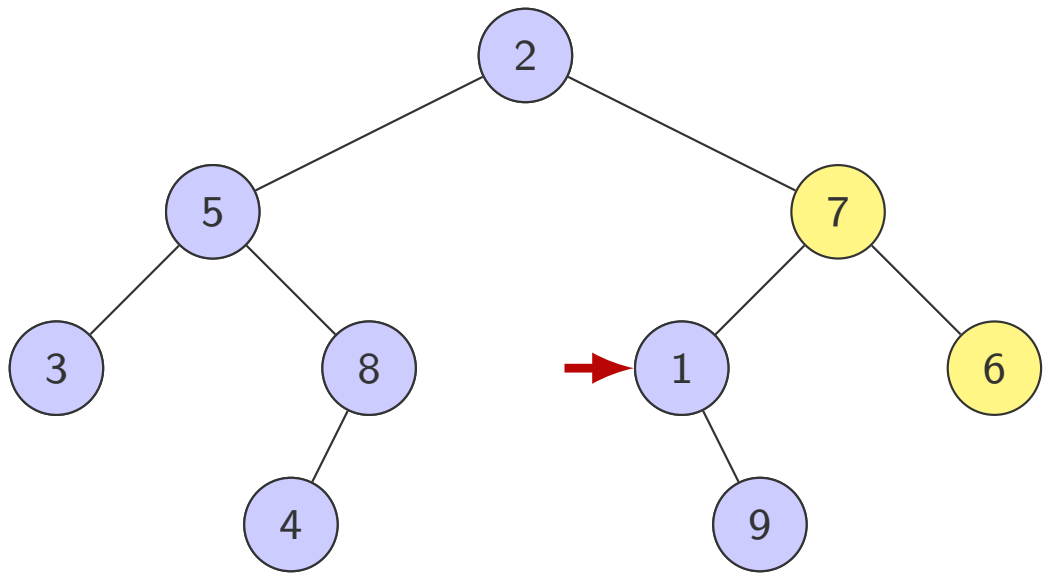


Ex: 3, 5, 4, 8, 2, 1,

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

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

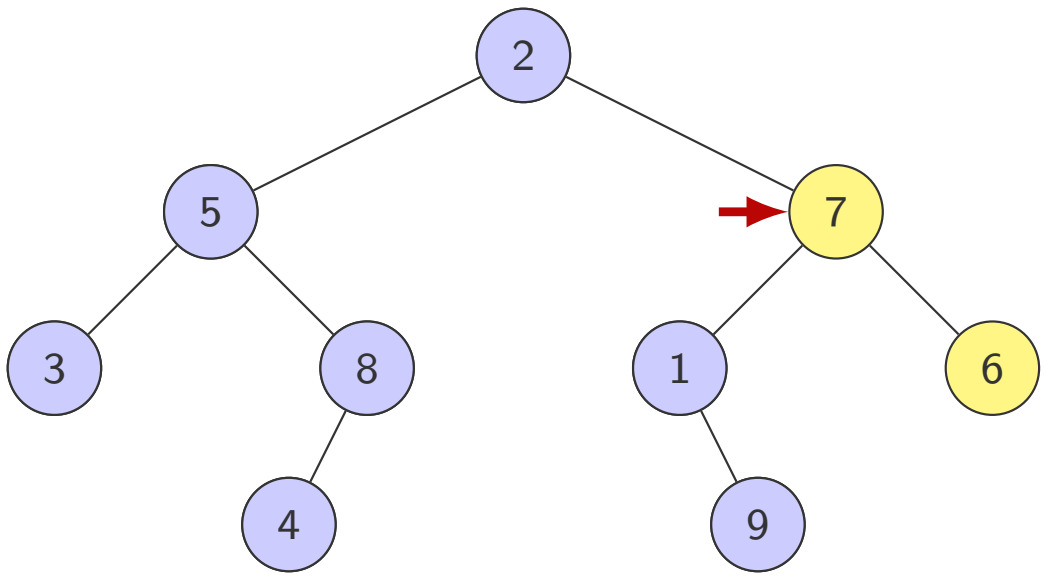


Ex: 3, 5, 4, 8, 2, 1, 9,

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

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

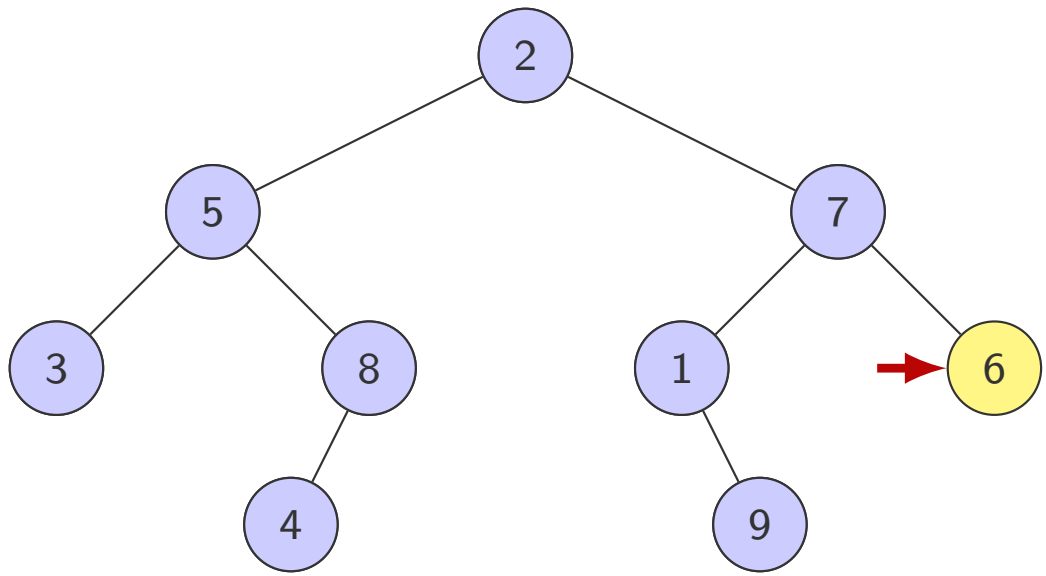


Ex: 3, 5, 4, 8, 2, 1, 9,

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

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

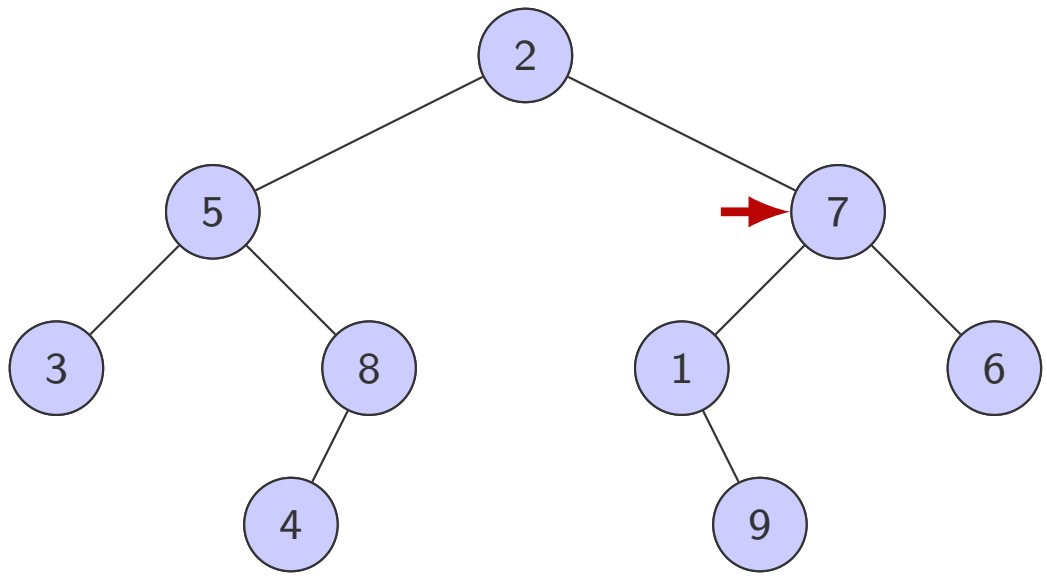


Ex: 3, 5, 4, 8, 2, 1, 9, 7,

Percorrendo os nós — Ordem Simétrica

A ordem simétrica

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

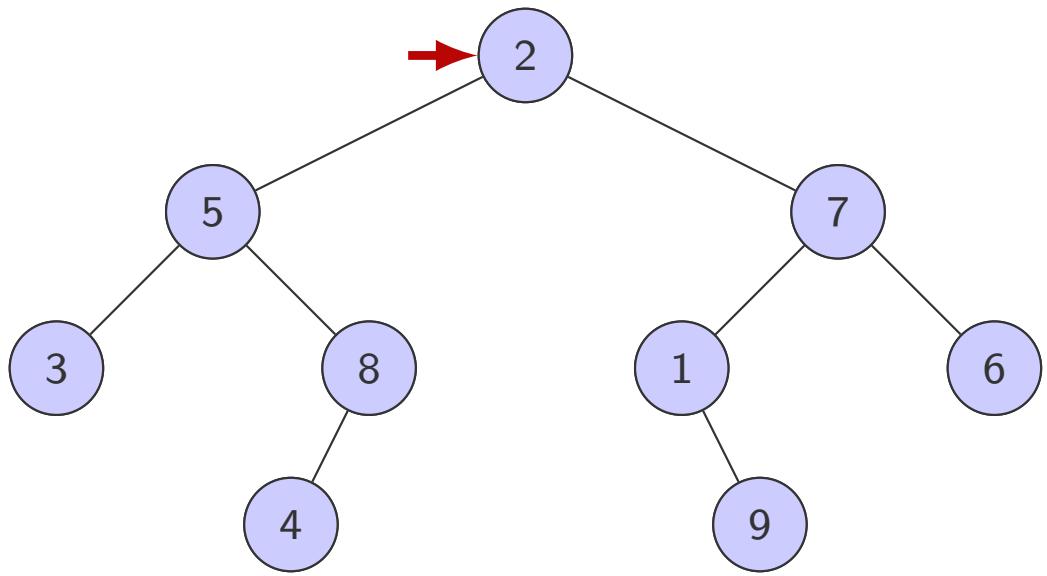


Ex: 3, 5, 4, 8, 2, 1, 9, 7, 6

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)

```
1: if ptr ≠ NULL then
2:     inorder(ptr→left)
3:     visit(ptr)
4:     inorder(ptr→right)
5: end if
```

Percursos Iterativos em Árvore

Percurso em pré-ordem — Recursivo

Vimos que o percurso em pré-ordem recursivo é implementado pela seguinte função:

Algorithm preorder(ptr)

Require: ptr (pointer to node)

```
1: if ptr ≠ NULL then
2:     visit(ptr)
3:     preorder(ptr→left)
4:     preorder(ptr→right)
5: end if
```

Como implementar a pré-ordem sem usar recursão?

Percurso em pré-ordem — Iterativo

Percurso em pré-ordem — Iterativo

Algorithm preorderIterativo()

Require: root (ponteiro para a raiz)

```
1:  Cria uma pilha vazia P de ponteiros para nós
2:  P.push(root)
3:  while P ≠ ∅ do
4:      node = P.top()
5:      P.pop()
6:      if node ≠ NULL then
7:          visit(node)
8:          P.push(node→right)
9:          P.push(node→left)
10:     end if
11: end while
```

Percurso em pré-ordem — Iterativo

Algorithm preorderIterativo()

Require: root (ponteiro para a raiz)

```
1:  Cria uma pilha vazia P de ponteiros para nós
2:  P.push(root)
3:  while P ≠ ∅ do
4:      node = P.top()
5:      P.pop()
6:      if node ≠ NULL then
7:          visit(node)
8:          P.push(node→right)
9:          P.push(node→left)
10:     end if
11: end while
```

Por que empilhamos `node->right` primeiro? E se fosse o contrário?

Percurso em ordem simétrica — Recursivo

O percurso em ordem simétrica (inordem) recursivo é implementado pela seguinte função:

Algorithm inorder(ptr)

Require: ptr (pointer to node)

```
1: if ptr ≠ NULL then
2:     inorder(ptr→left)
3:     visit(ptr)
4:     inorder(ptr→right)
5: end if
```

Como percorrer em ordem simétrica sem usar recursão?

Percurso em ordem simétrica — Iterativo

Percurso em ordem simétrica — Iterativo

Algorithm inorderIterativo()

Require: root (ponteiro para a raiz)

```
1:  Cria uma pilha vazia P de ponteiros para nós
2:  node = root
3:  while P ≠ ∅ or node ≠ NULL do
4:      if node ≠ NULL then
5:          P.empilha(node)
6:          node = node→left
7:      else
8:          node = P.topo()
9:          P.desempilha()
10:         visit(node)
11:         node = node→right
12:     end if
13: end while
```

Percurso em pós-ordem iterativo

- Exercício para casa: Implementar o percurso em pós-ordem iterativo.

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)

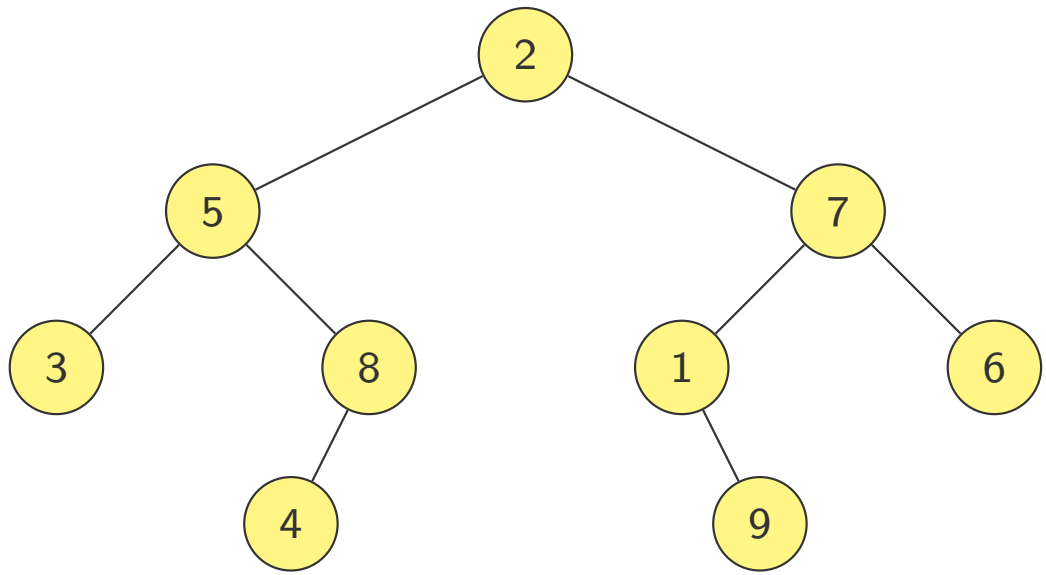
O percurso em largura

- 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

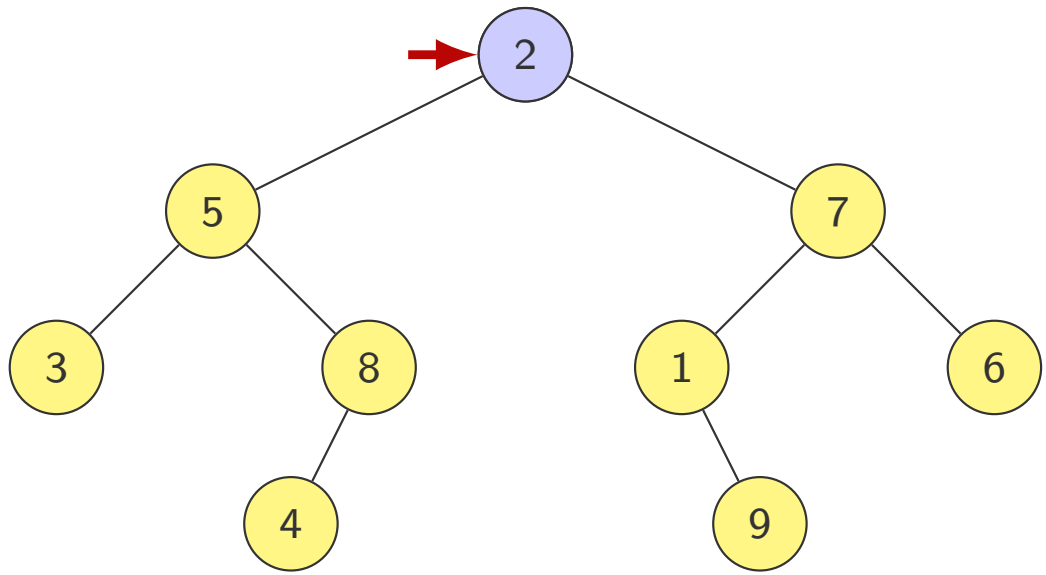


Ex:

Percorrendo os nós (em largura)

O percurso em largura

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

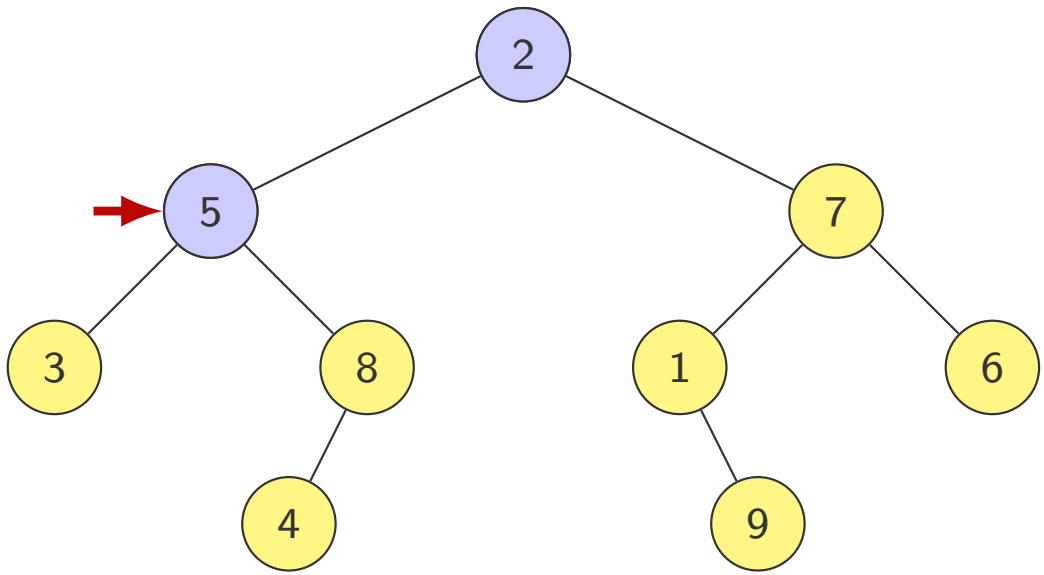


Ex: 2,

Percorrendo os nós (em largura)

O percurso em largura

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

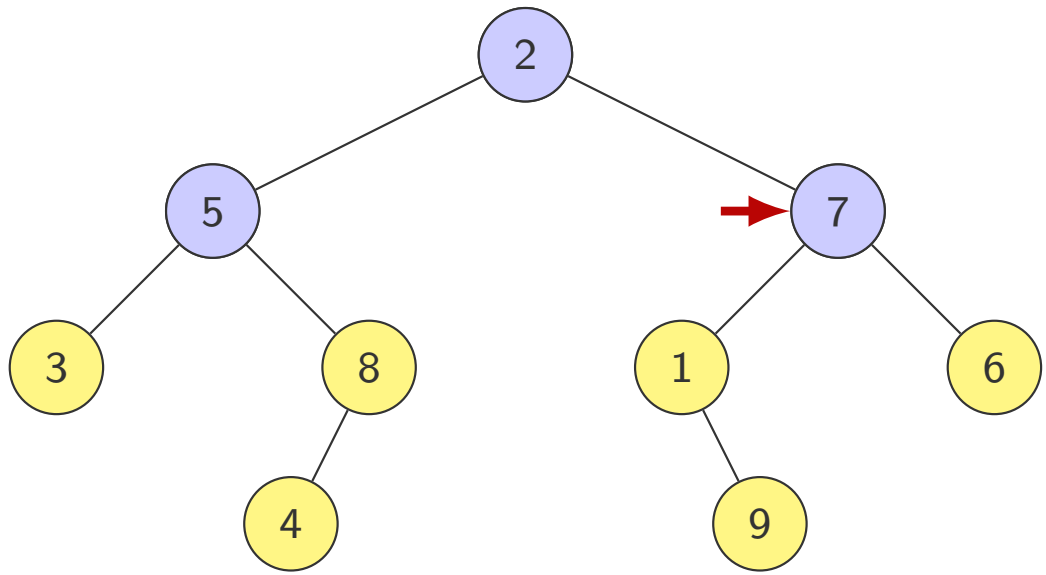


Ex: 2, 5,

Percorrendo os nós (em largura)

O percurso em largura

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

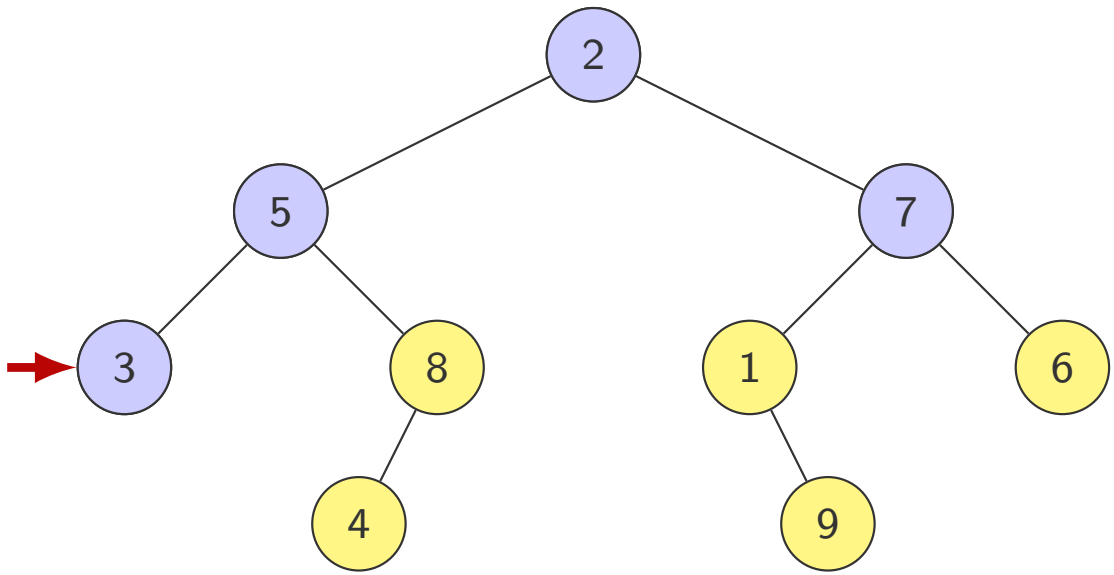


Ex: 2, 5, 7,

Percorrendo os nós (em largura)

O percurso em largura

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

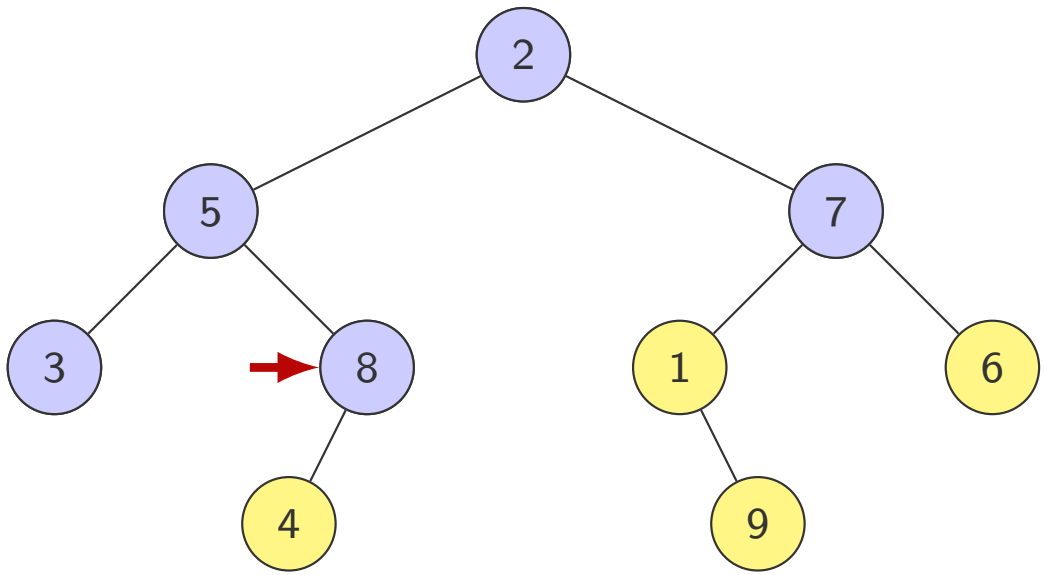


Ex: 2, 5, 7, 3,

Percorrendo os nós (em largura)

O percurso em largura

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

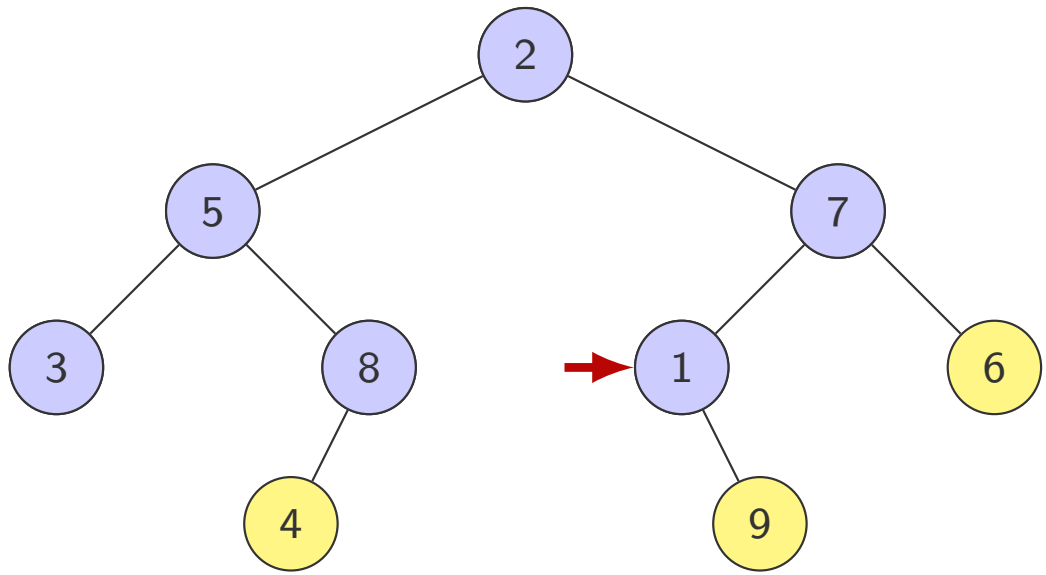


Ex: 2, 5, 7, 3, 8,

Percorrendo os nós (em largura)

O percurso em largura

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

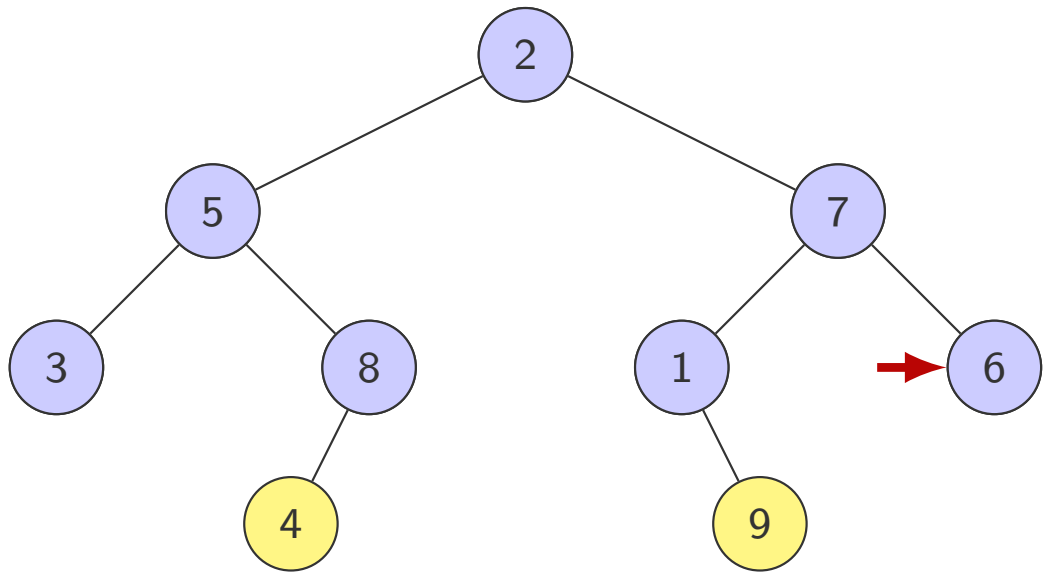


Ex: 2, 5, 7, 3, 8, 1,

Percorrendo os nós (em largura)

O percurso em largura

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

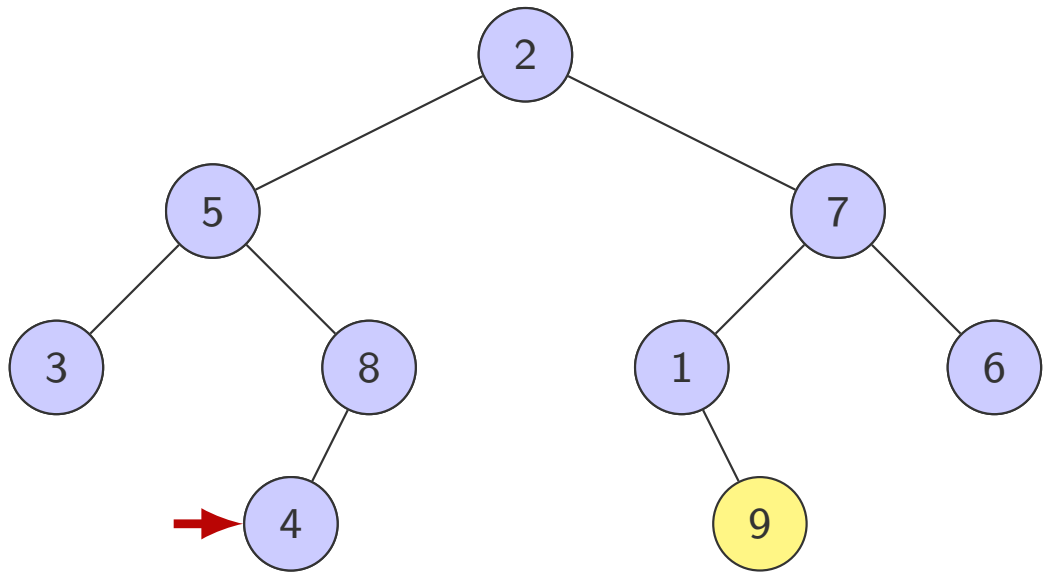


Ex: 2, 5, 7, 3, 8, 1, 6,

Percorrendo os nós (em largura)

O percurso em largura

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

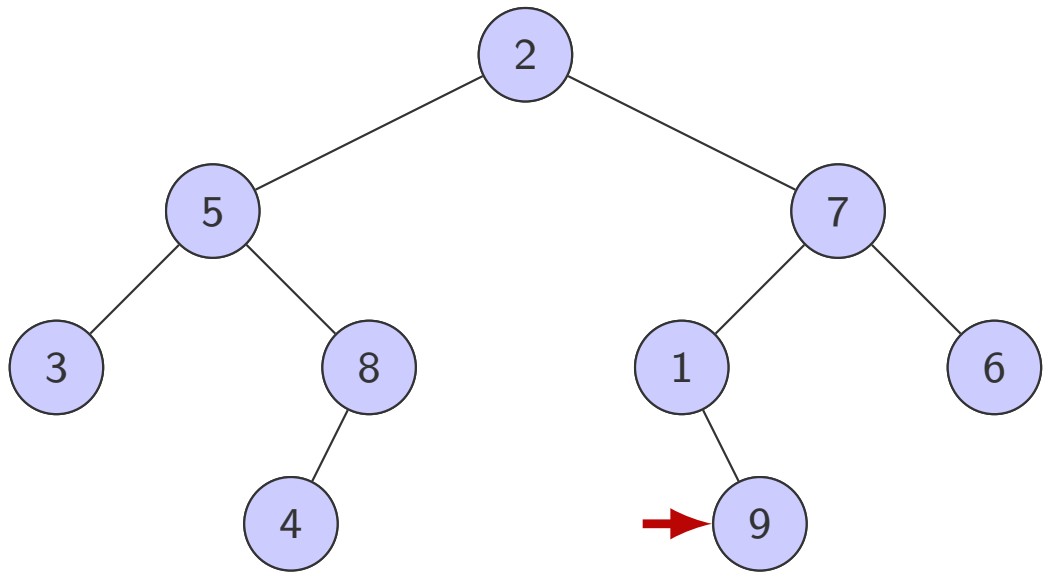


Ex: 2, 5, 7, 3, 8, 1, 6, 4,

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

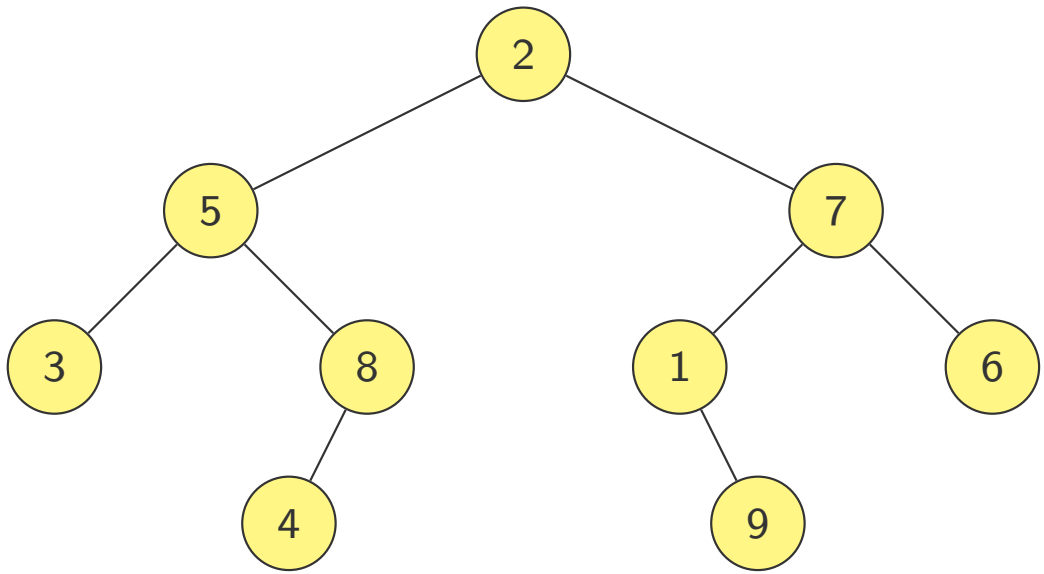
Como implementar a busca em largura?

- 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

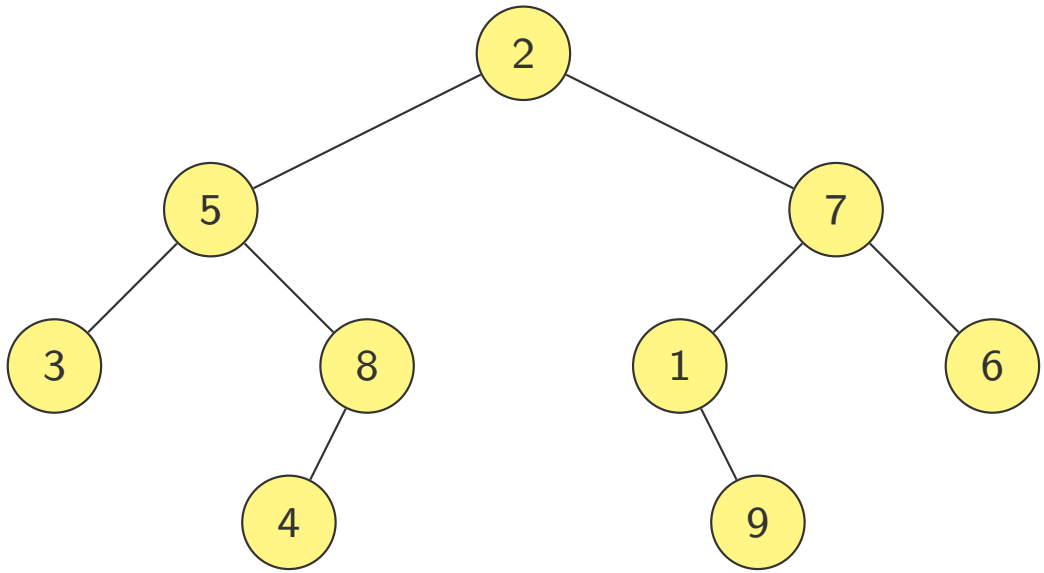


Fila

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

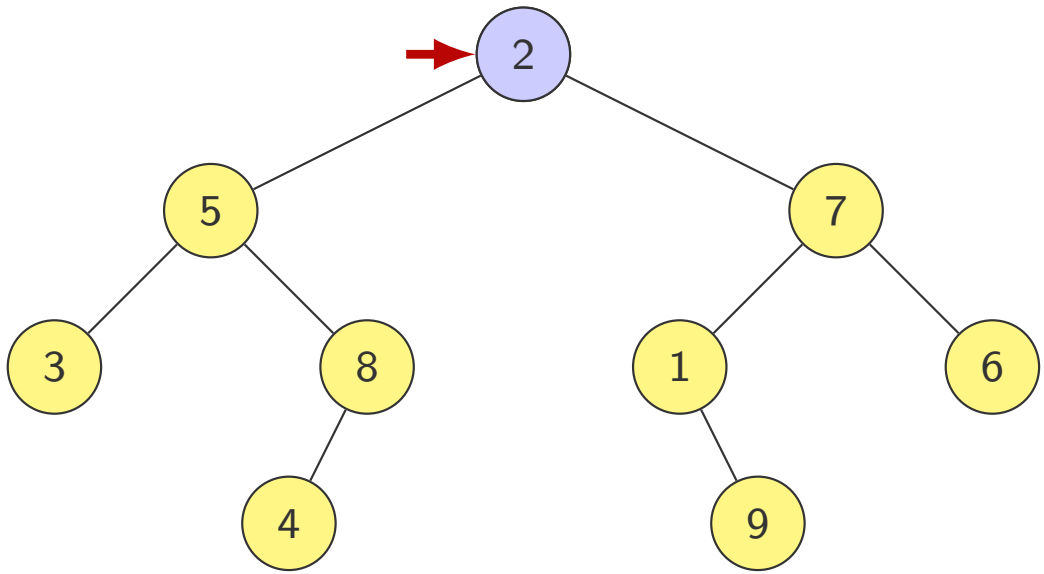


Fila 

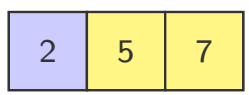
Implementação do percurso em largura

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



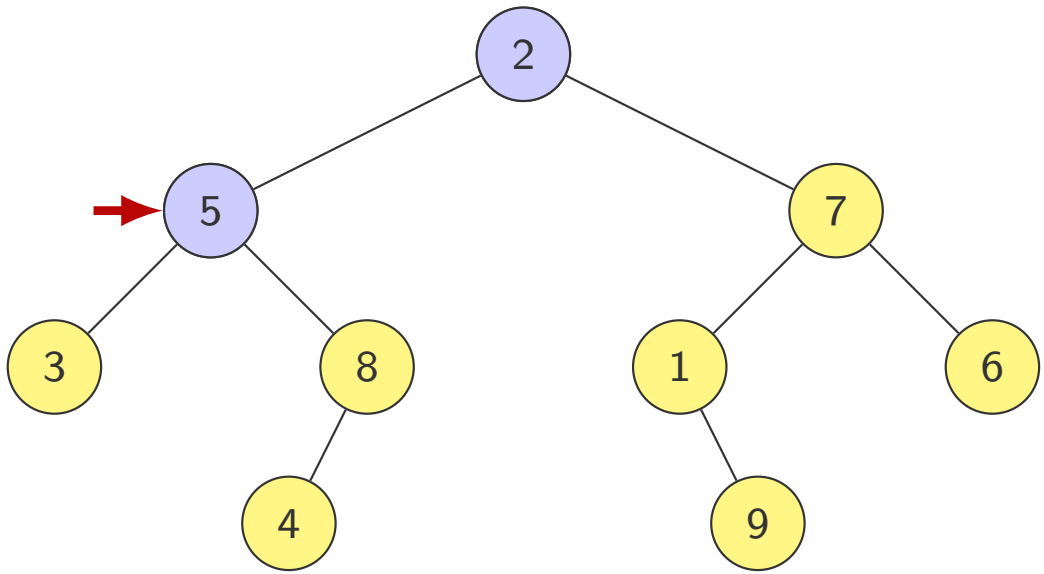
Fila



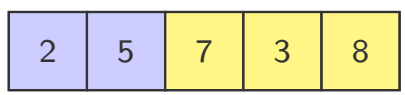
Implementação do percurso em largura

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



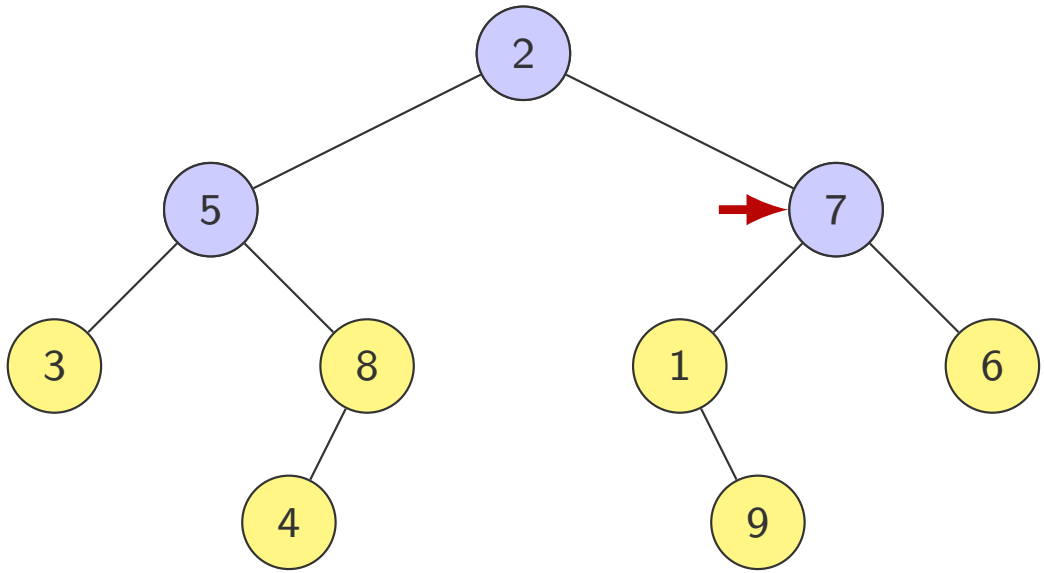
Fila



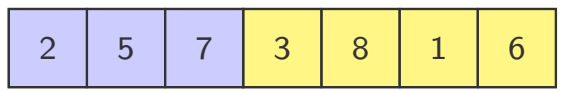
Implementação do percurso em largura

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



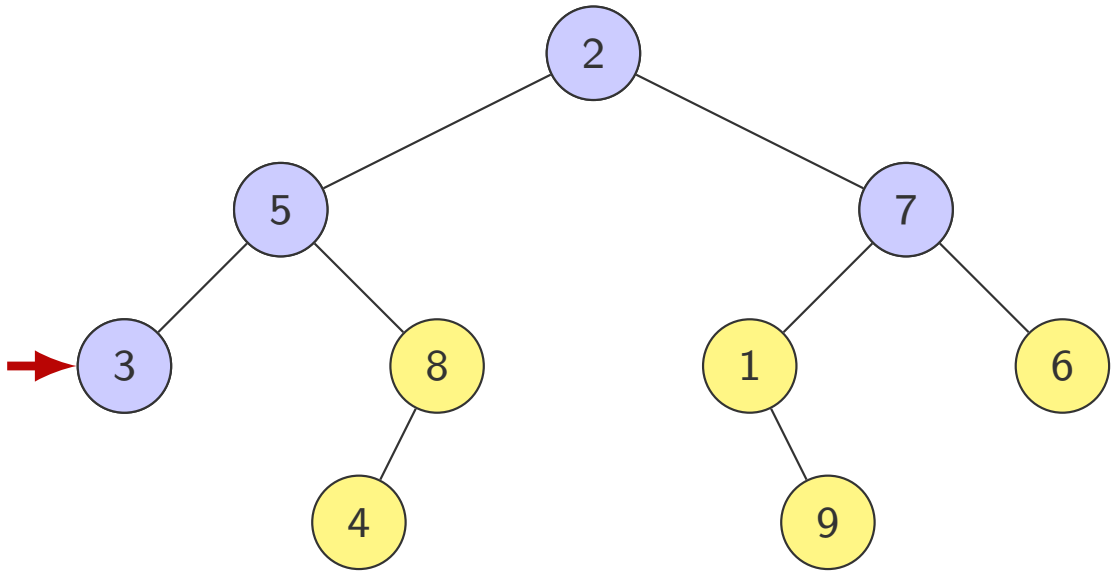
Fila



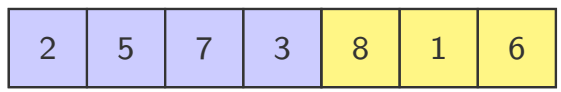
Implementação do percurso em largura

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



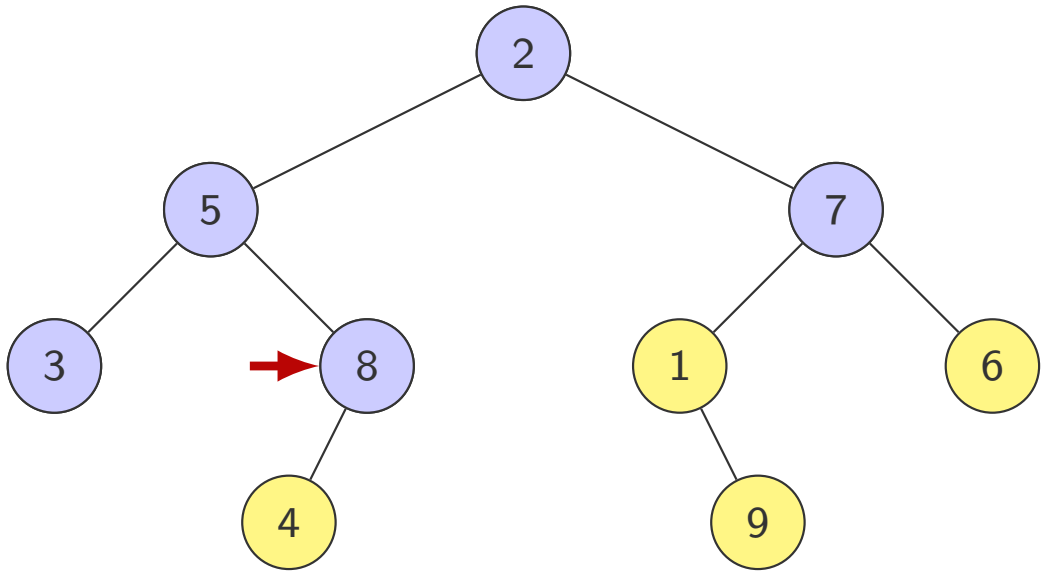
Fila



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

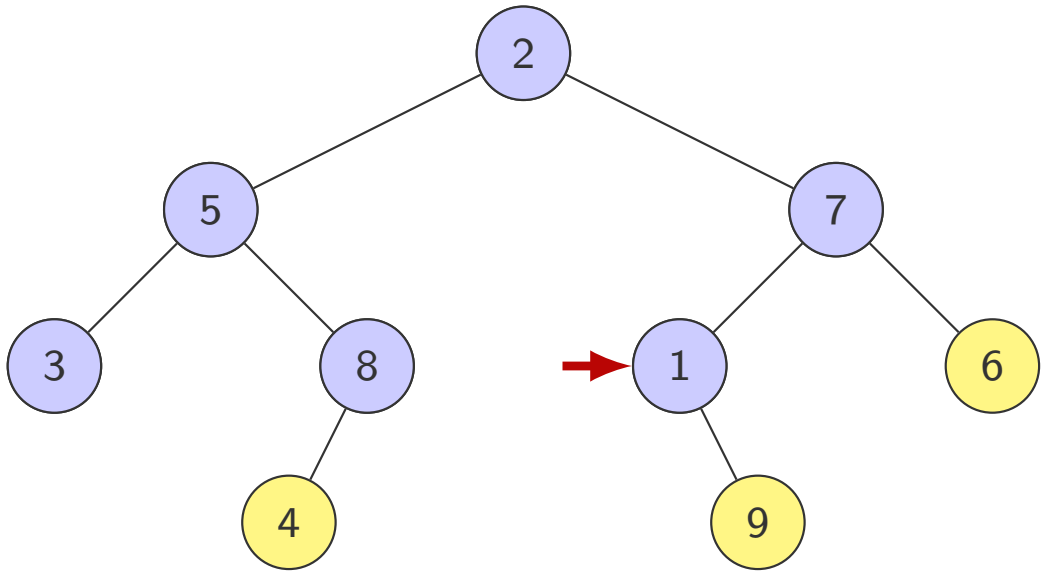


Fila

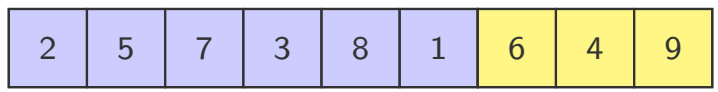
Implementação do percurso em largura

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



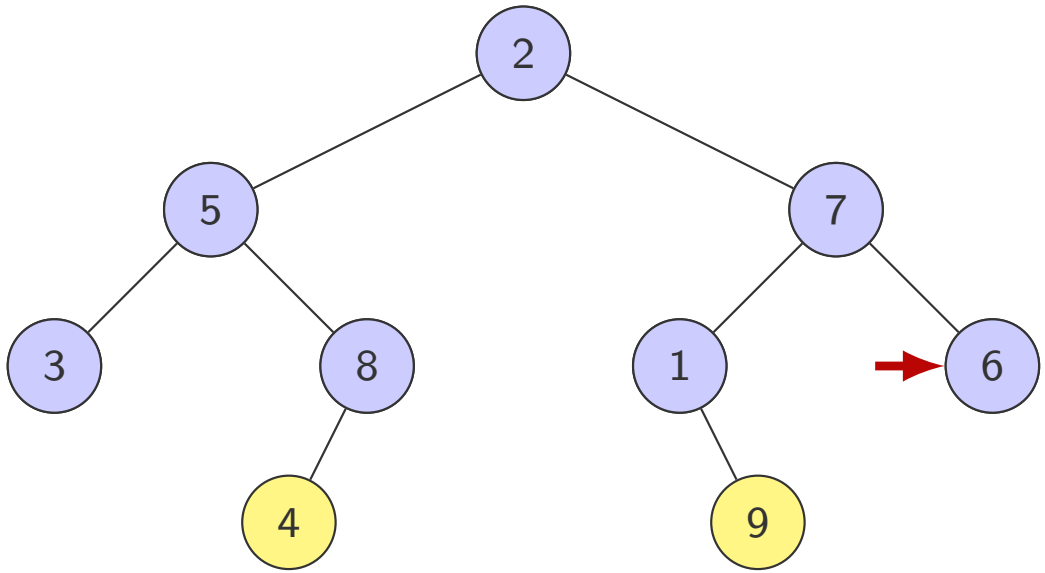
Fila



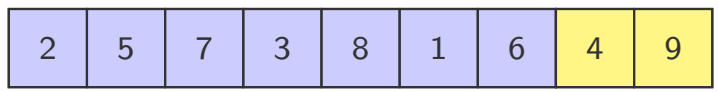
Implementação do percurso em largura

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



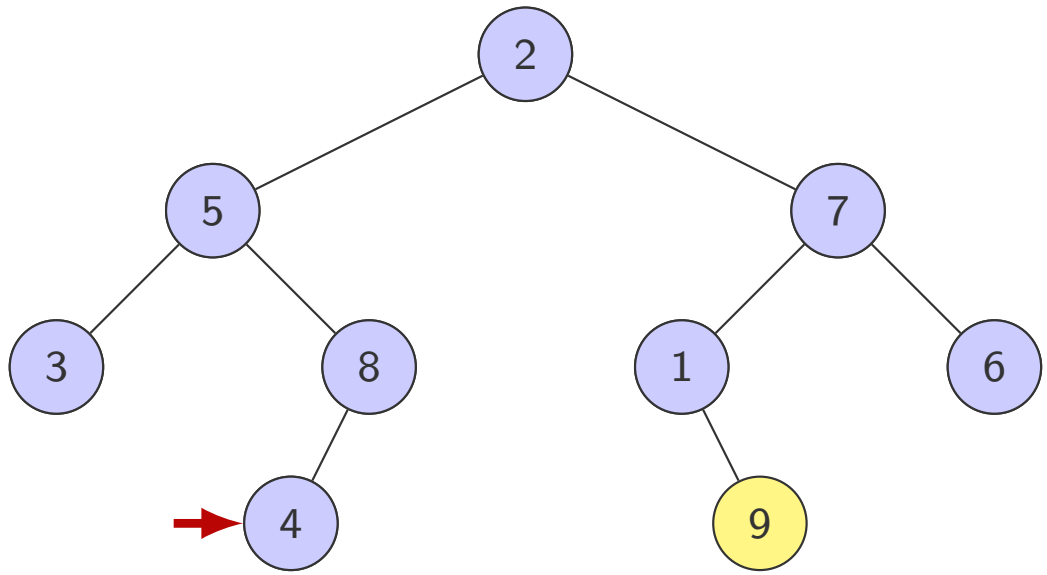
Fila



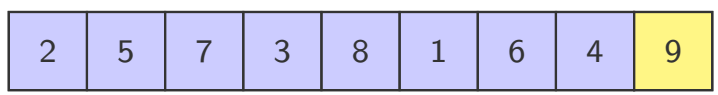
Implementação do percurso em largura

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



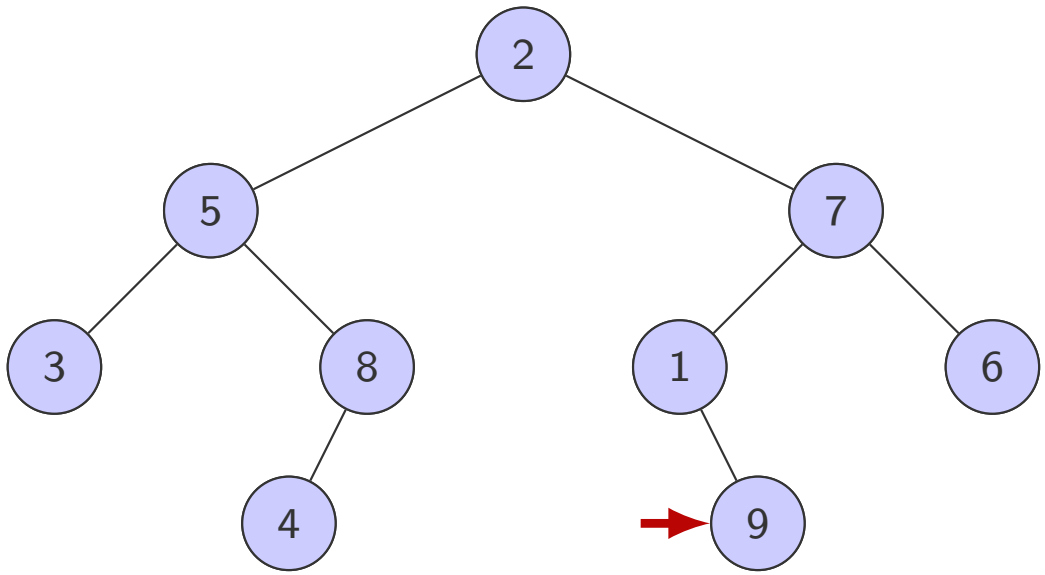
Fila



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



Fila

2	5	7	3	8	1	6	4	9

Percurso em largura

Algorithm levelTraversal()

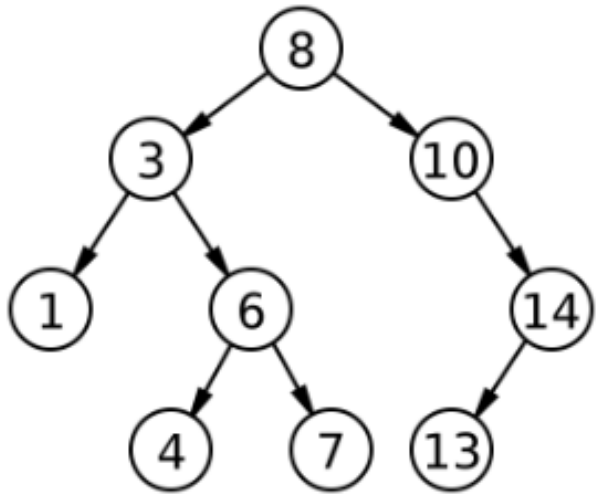
Require: root (ponteiro para a raiz)

```
1:  Cria uma fila vazia Q de ponteiros para nós
2:  Q.push(root)
3:  while Q ≠ ∅ do
4:      node = Q.front()
5:      Q.pop()
6:      if node ≠ NULL then
7:          visit(node)
8:          Q.push(node→left)
9:          Q.push(node→right)
10:     end if
11: end while
```

Serialização de árvores

Serialização de Árvores

- A serialização de uma árvore binária é um processo pelo qual percorremos a árvore em pré-ordem e adicionamos o valor de cada chave encontrada ao final de uma string que inicialmente começa vazia, sendo que, para cada filho nulo encontrado, seu valor é representado pelo caractere '#'.
  Exemplo:



A serialização da árvore acima consiste na string:

`8 3 1 # # 6 4 # # 7 # # 10 # 14 13 # # #`

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.

Anexos

Demonstração do Lema 2

Lema 2: Se T é uma árvore binária completa com $n > 0$ nós, então T possui altura $h = 1 + \lfloor \lg n \rfloor$.

Prova: Indução forte em n. Se $n = 1$, então $h = 1 + \lfloor \lg 1 \rfloor = 1$, correto.

Quando $n > 1$, suponha o resultado verdadeiro para todas as árvores binárias com até $n - 1$ nós. Seja T' a árvore obtida de T pela remoção de todos os k nós que estão do último nível de T. Logo, T' é uma árvore cheia com $n' = n - k$ nós.

Pela hipótese de indução, $h(T') = 1 + \lfloor \lg n' \rfloor$. Como T' é cheia, $n' = 2^m - 1$, para algum inteiro $m > 0$.

Como T' é uma árvore cheia com $2^m - 1$ nós, concluímos que $h(T') = m$.

Demonstração do Lema (Continuação)

Logo, $h(T') = m$. Além disso, $1 \leq k \leq n' + 1$ (Lema 1). Assim,

$$\begin{aligned}
h(T) &= 1 + h(T') \\
&= 1 + m \\
&= 1 + \lg(n' + 1) && \text{pois } n' = 2^m - 1 \\
&= 1 + \lfloor \lg(n' + k) \rfloor && \text{pois } 1 \leq k \leq n' + 1 \\
&= 1 + \lfloor \lg n \rfloor. && \text{pois } n' = n - k
\end{aligned}$$

□

FIM