Operações em Árvores Binárias

Usando Estruturas Dinâmicas

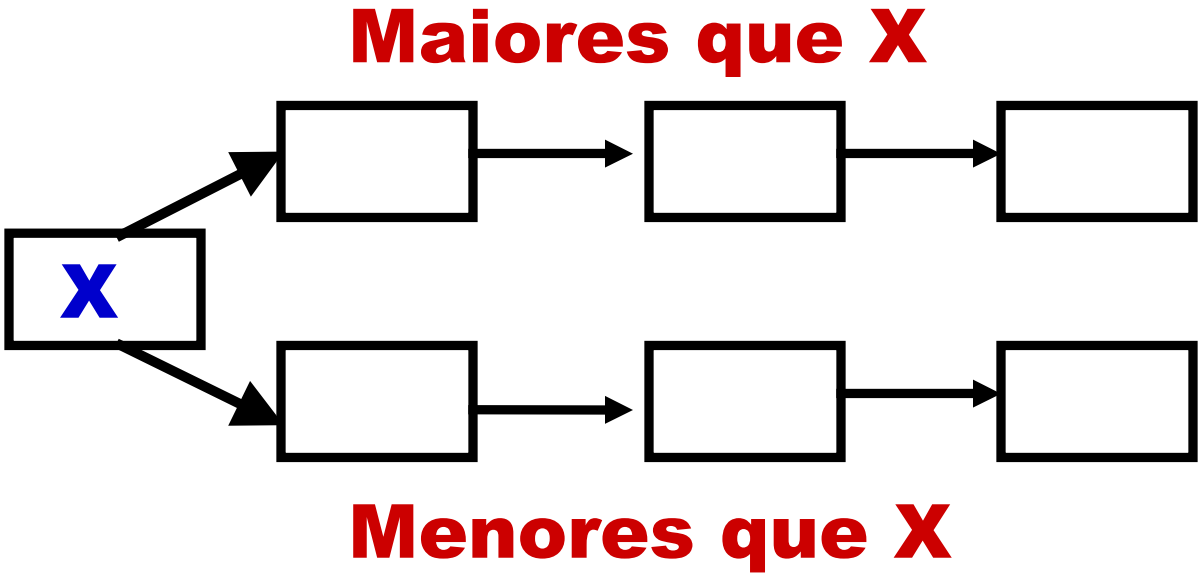
A capacidade de alocação dinâmica é muito desejável em muitos casos.

Mas as estruturas lineares, como listas ligadas, têm um problema por exigir acesso sequencial

Estruturas não Lineares

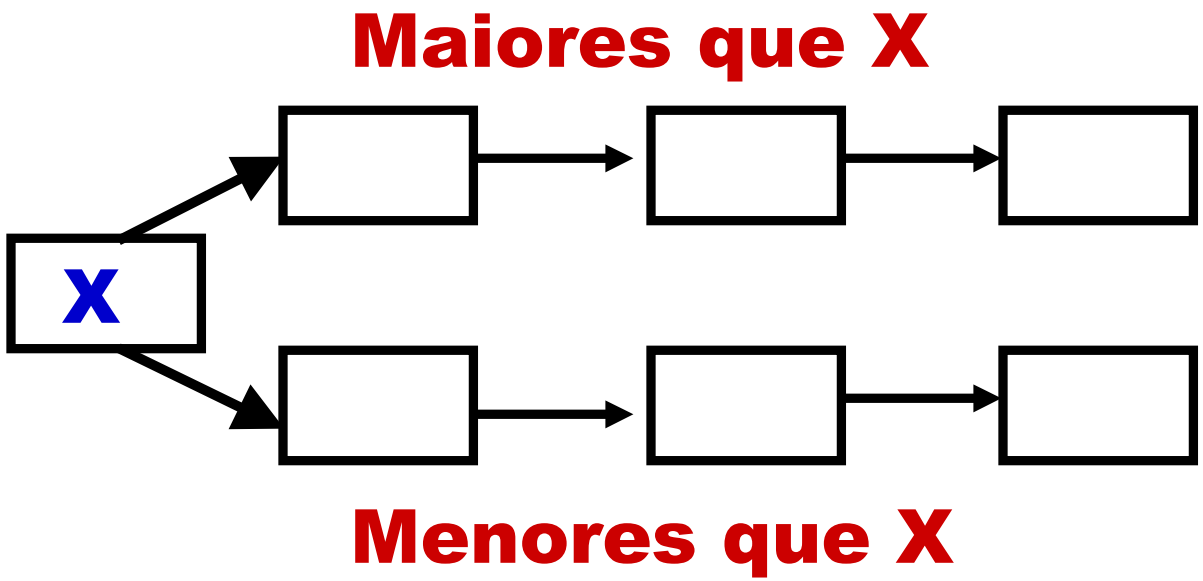
Uma solução para este problema pode ser através do uso de estruturas não lineares.

Ex: Dividir a lista em duas.



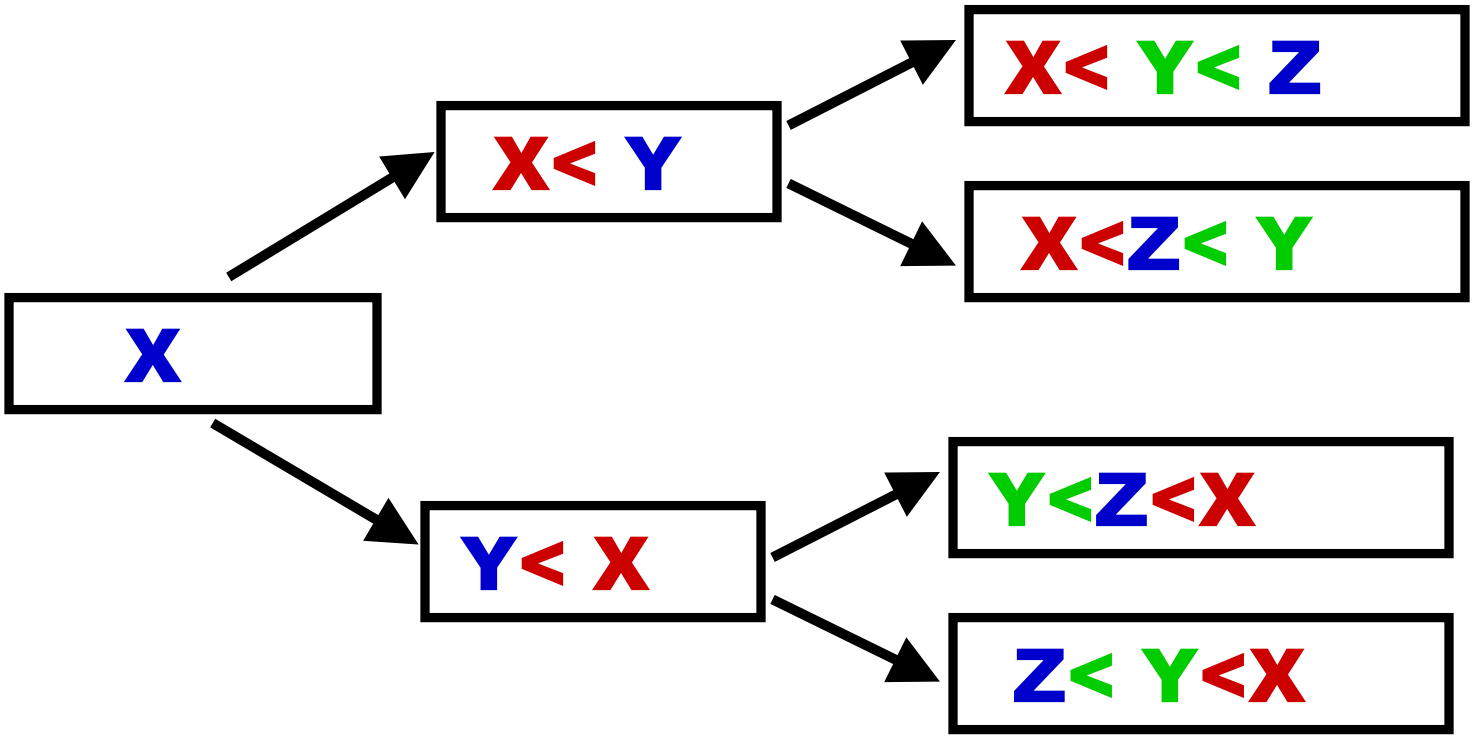
Estruturas não Lineares

O problema de dividir a lista em duas é que isto apenas "divide o problema em dois".

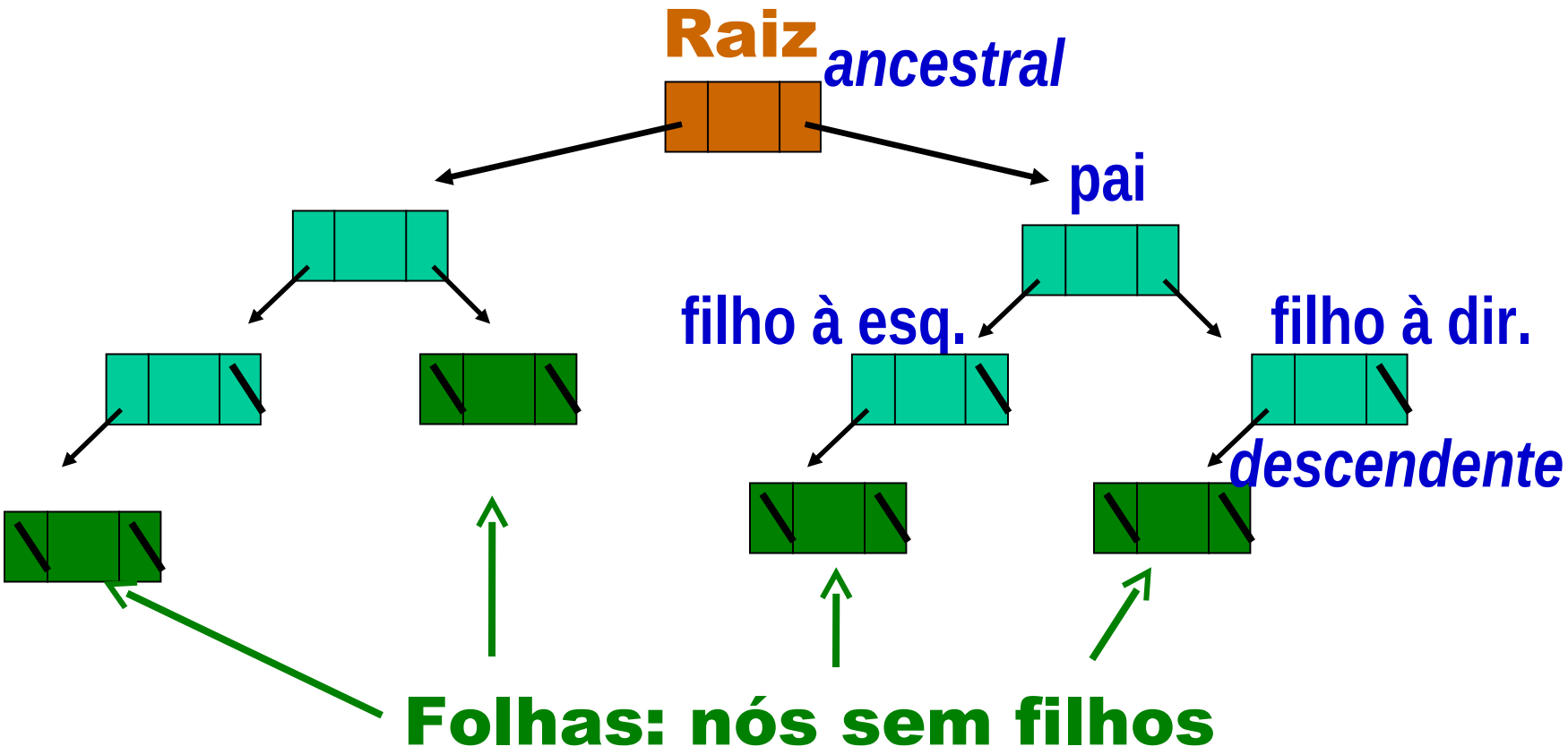


Estruturas não Lineares

Uma solução mais robusta consiste em dividir a lista a cada elemento.



Árvores: Estruturas dinâmicas com:



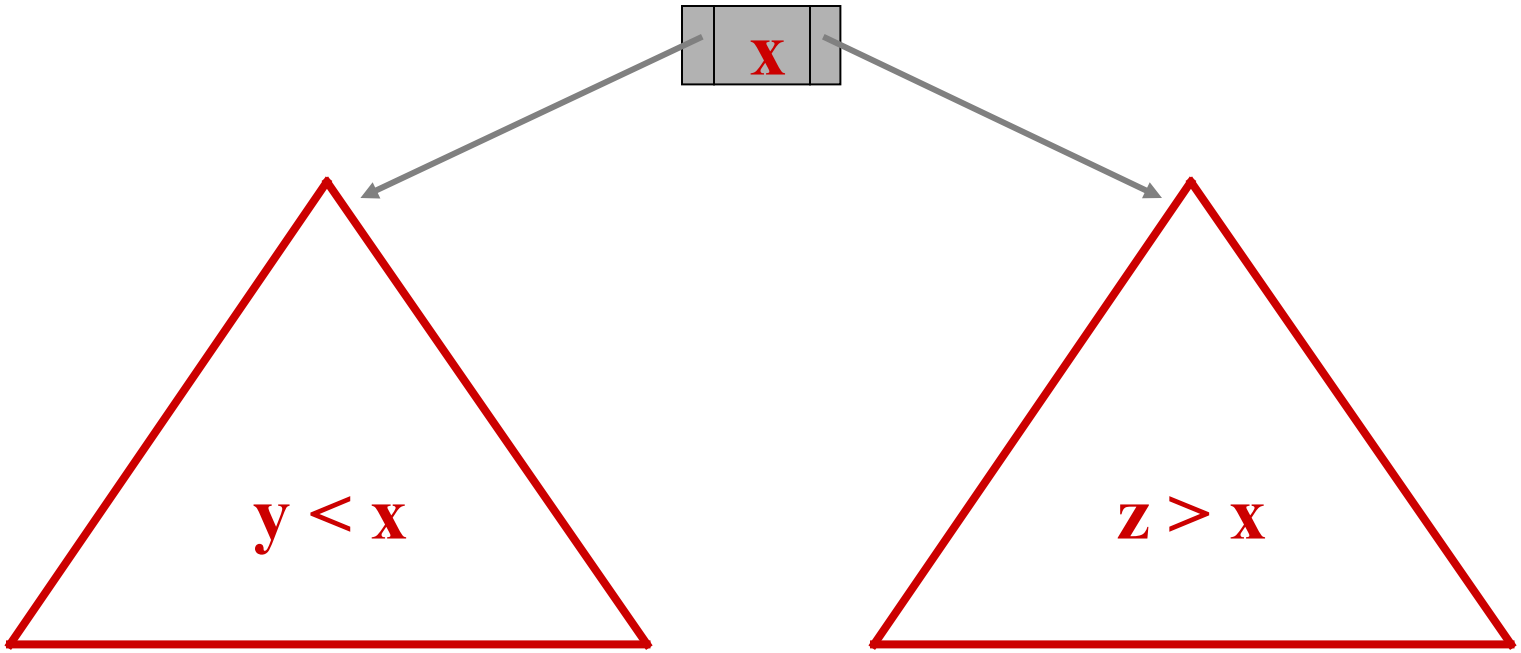
Árvore de Busca Binária

Árvores binárias onde os elementos são organizados de forma que:

- Todos os elementos na sub-árvore esquerda de cada nó *k* têm valor menor ou igual ao valor no nó *k*.
- Todos os elementos na sub-árvore direita de cada nó *k* têm valor maior do que o valor no nó *k*.

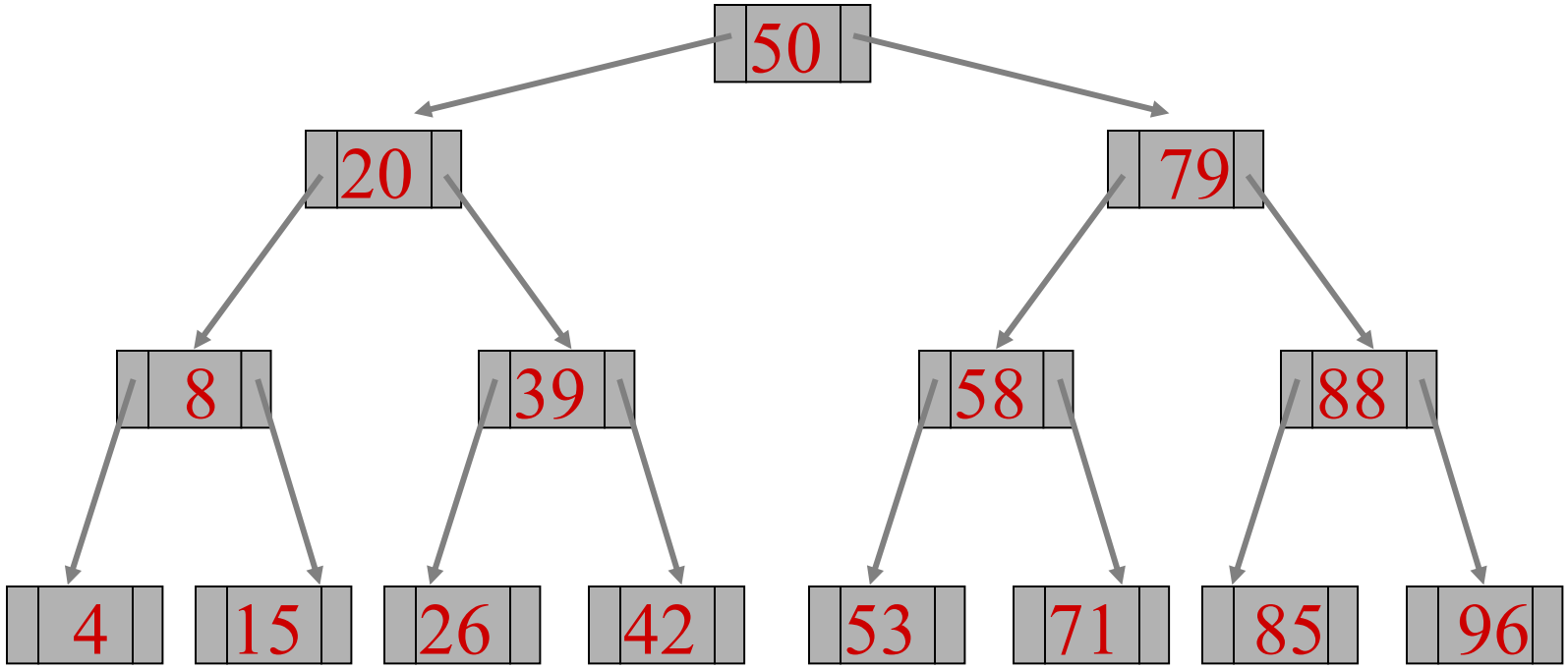
Árvore de Busca Binária

Árvore binária onde os elementos são organizados de forma que:



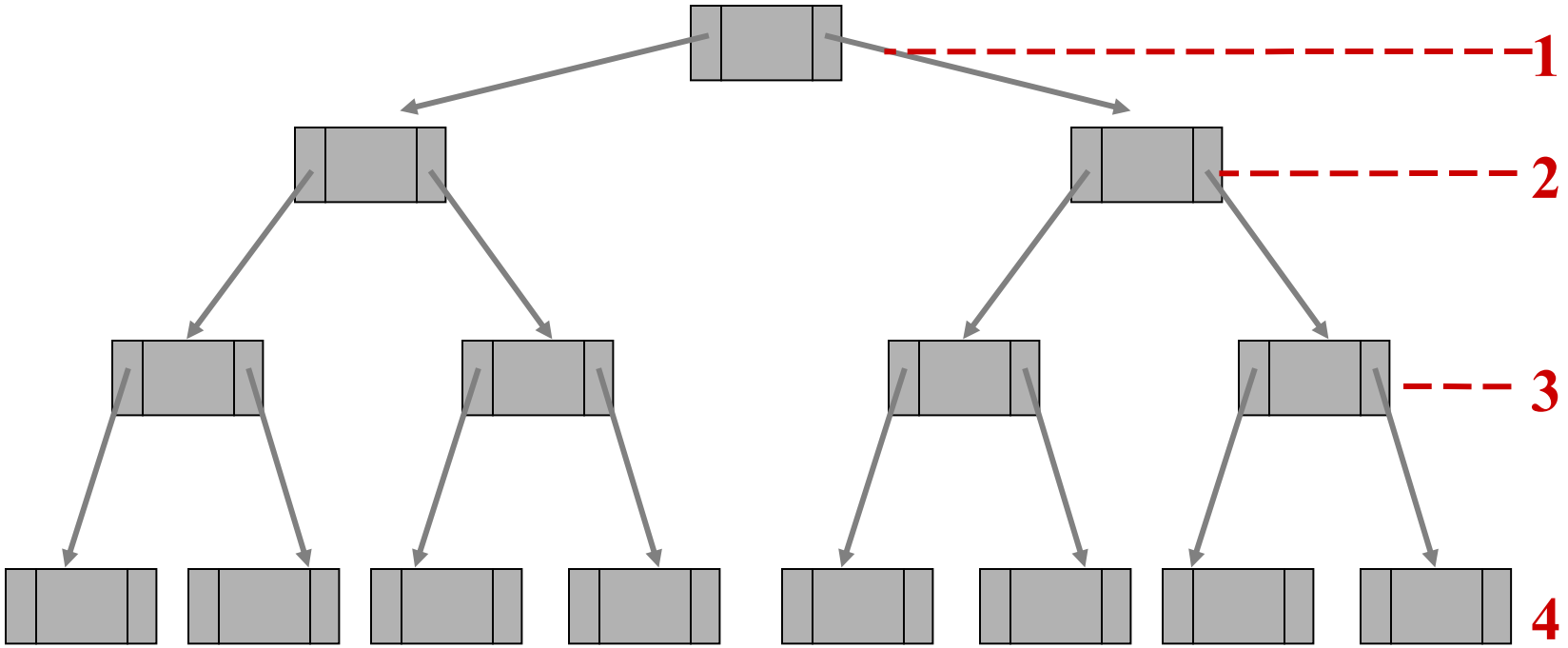
Árvore de Busca Binária

Ex: 50, 20, 39, 8, 79, 26, 58,15, 88, 4, 85, 96, 71, 42, 53.



Árvore de Busca Binária

Estrutura de dados dinâmica, com recuperação em tempo logarítmico.



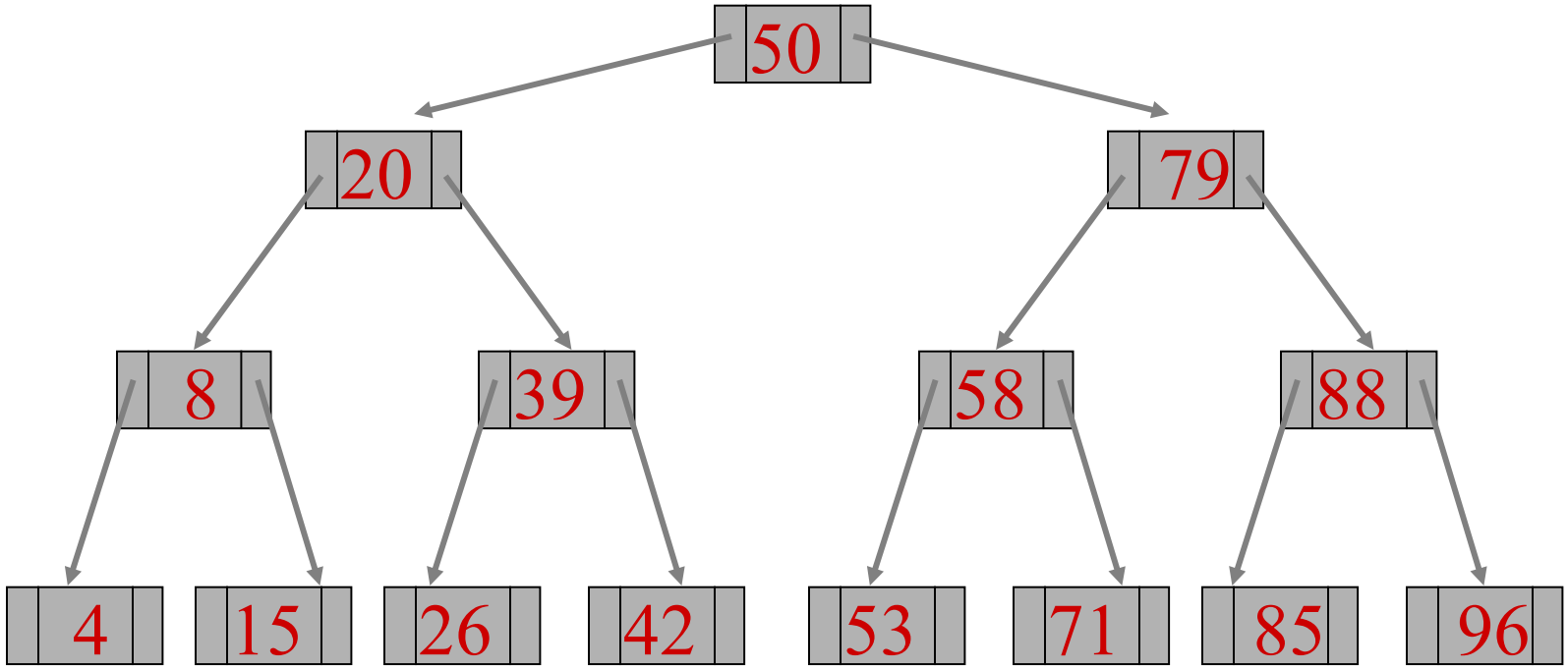
Inserção em Árvores de Busca

Ocorre sempre em uma folha.

```
Procedimento Inclui (raiz,  x):
   Se raiz então
         se elemento na raiz > x
               então Inclui (esquerda, x)
               senão Inclui (direita, x)
   senão { aloque espaço para um nó;
            coloque x neste nó;
            retorne apontador novo nó}
```

Árvore de Busca Binária

Ex: 50, 20, 39, 8, 79, 26, 58,15, 88, 4, 85, 96, 71, 42, 53.



Remoção em Árvore de Busca

Nem sempre ocorre em uma folha.

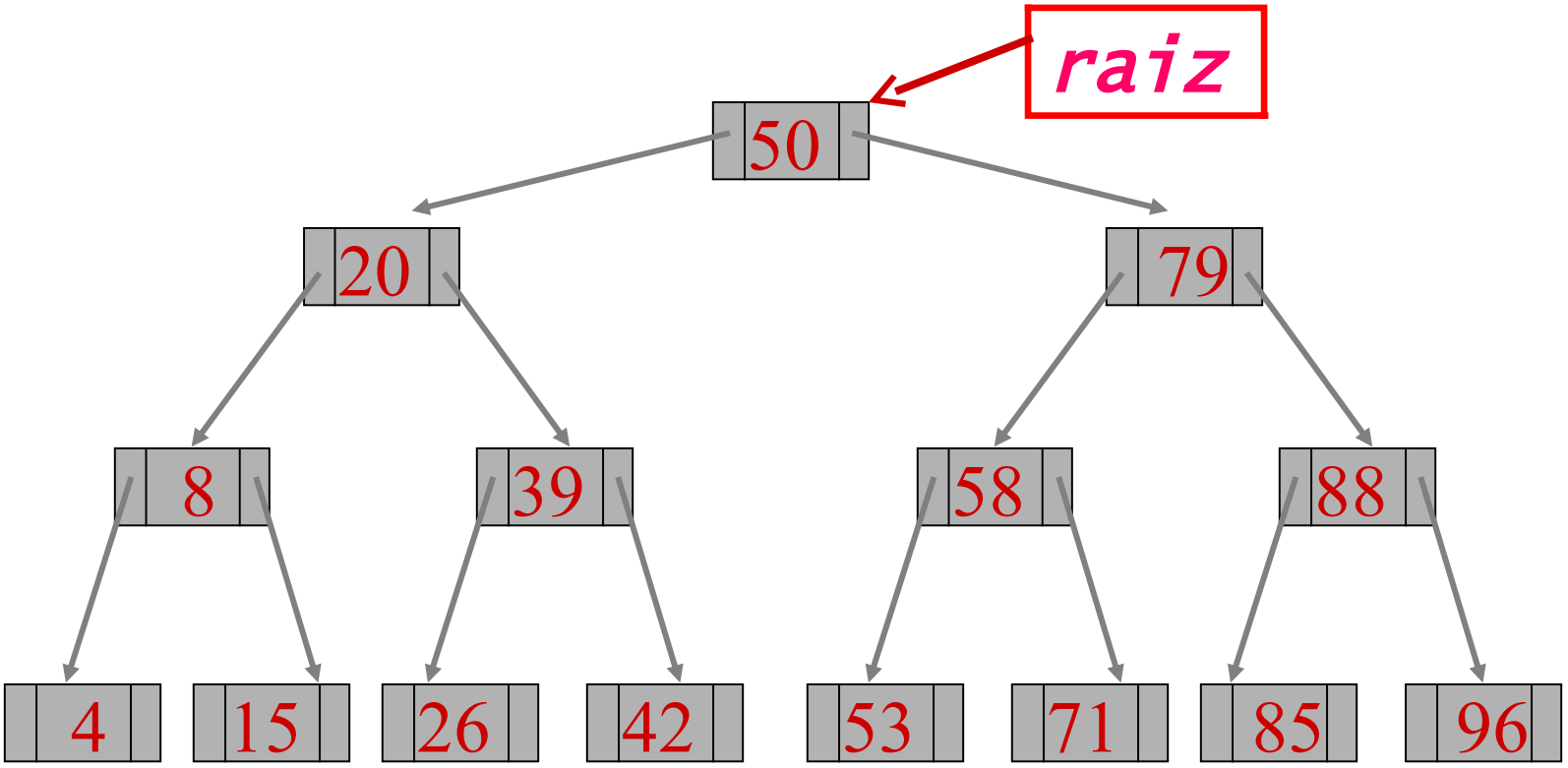
Procedimento Remove (raiz, *x*)

EndNó <- Busca (*x*);

Se ¬EndNó então relate insucesso

Remoção em Árvore de Busca

Exemplo: Remove (*raiz*, 60)



Remoção em Árvores de Busca

Nem sempre ocorre em uma folha.

Procedimento Remove (raiz, *x*)

EndNó <- Busca (*x*);

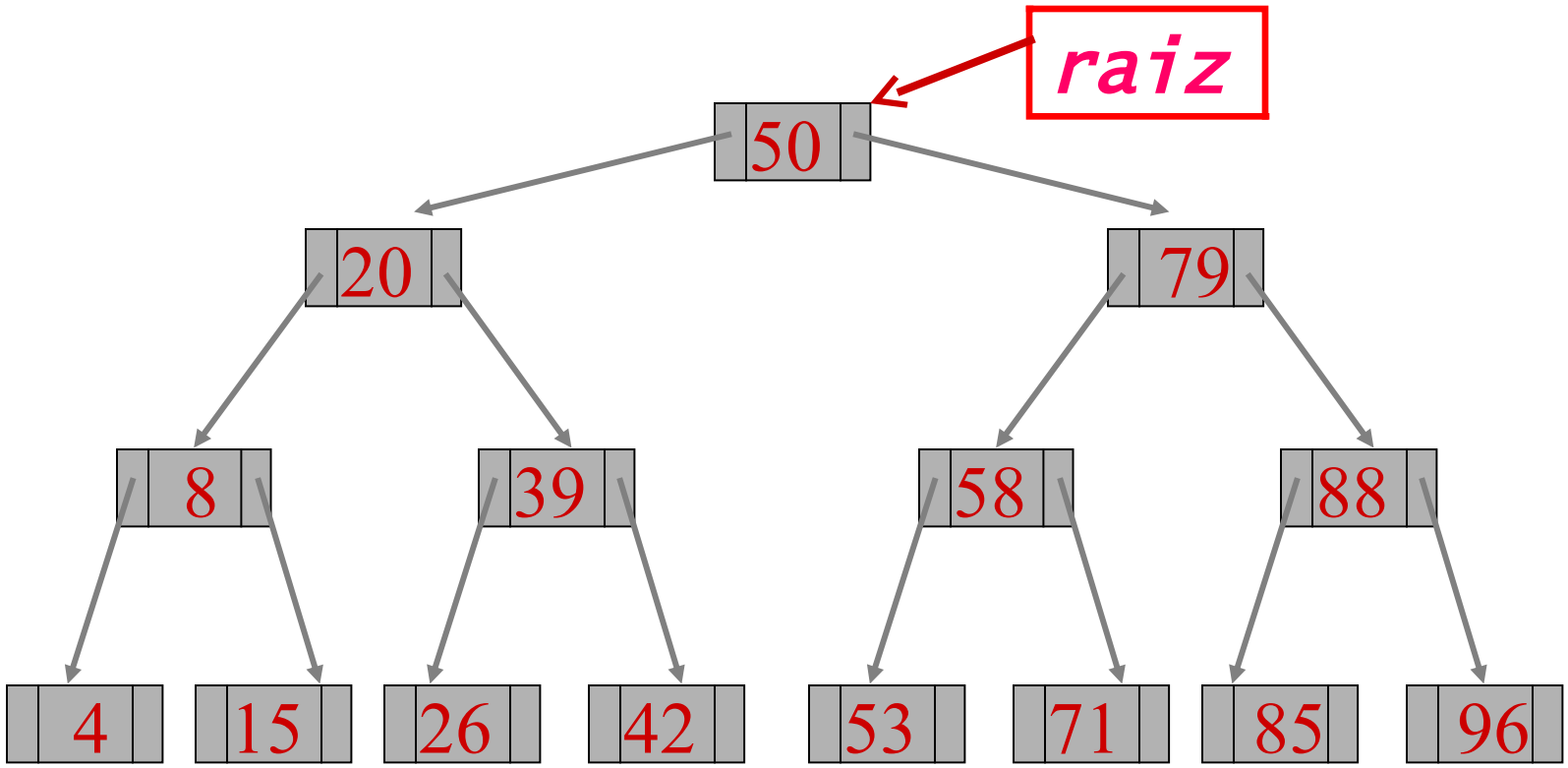
Se ¬EndNó então relate insucesso

senão se EndNó é folha (¬esq AND ¬dir)

então apague apontador pai;

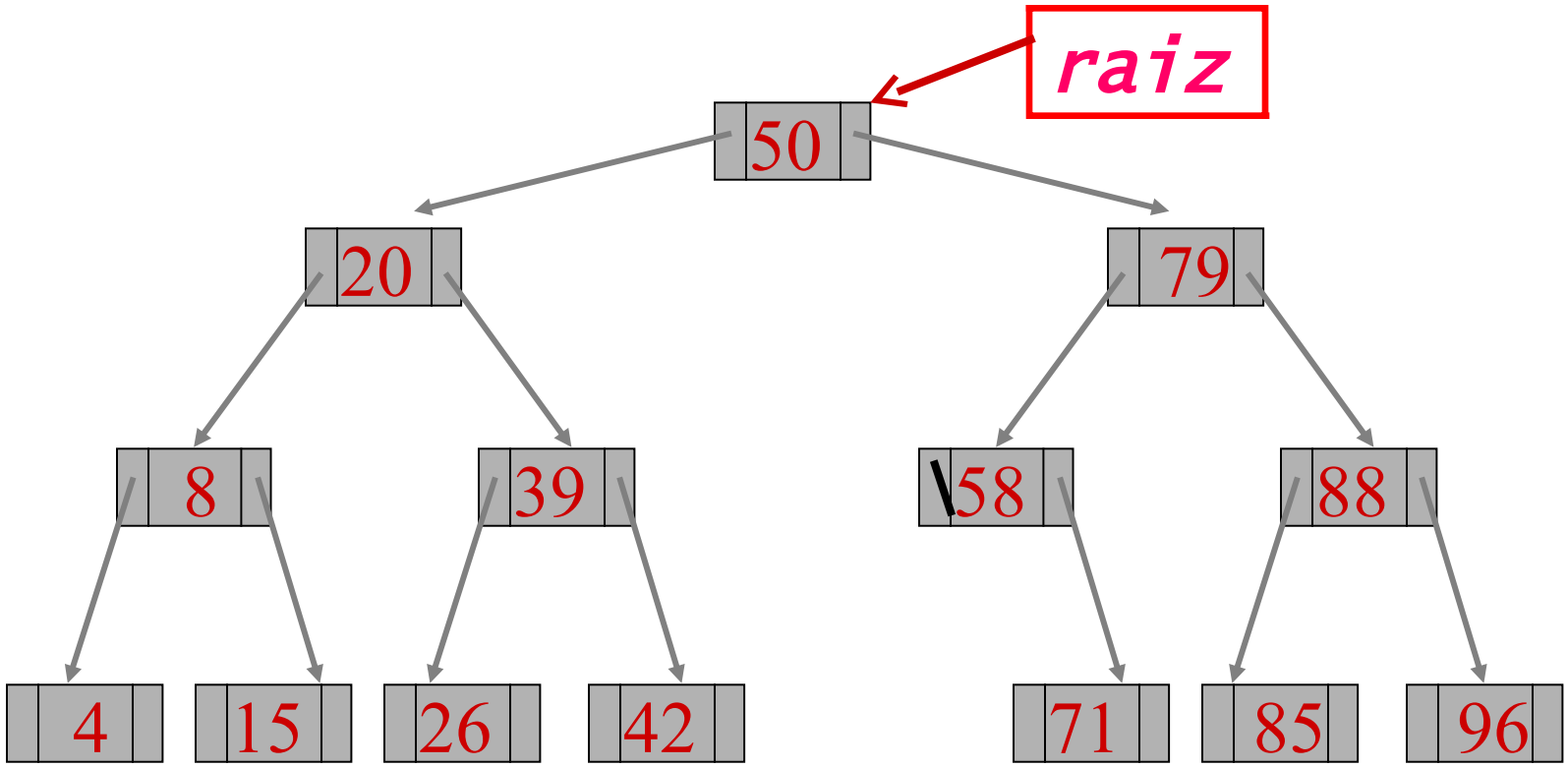
Remoção em Árvore de Busca

Exemplo: Remove (raiz, 53)



Remoção em Árvore de Busca

Exemplo: Remove (raiz, 53)



Remoção em Árvore de Busca

Procedimento Remove (raiz, *x*)

EndNó <- Busca (*x*);

Se ¬EndNó então relate insucesso

senão se EndNó é folha (¬esq AND ¬dir)

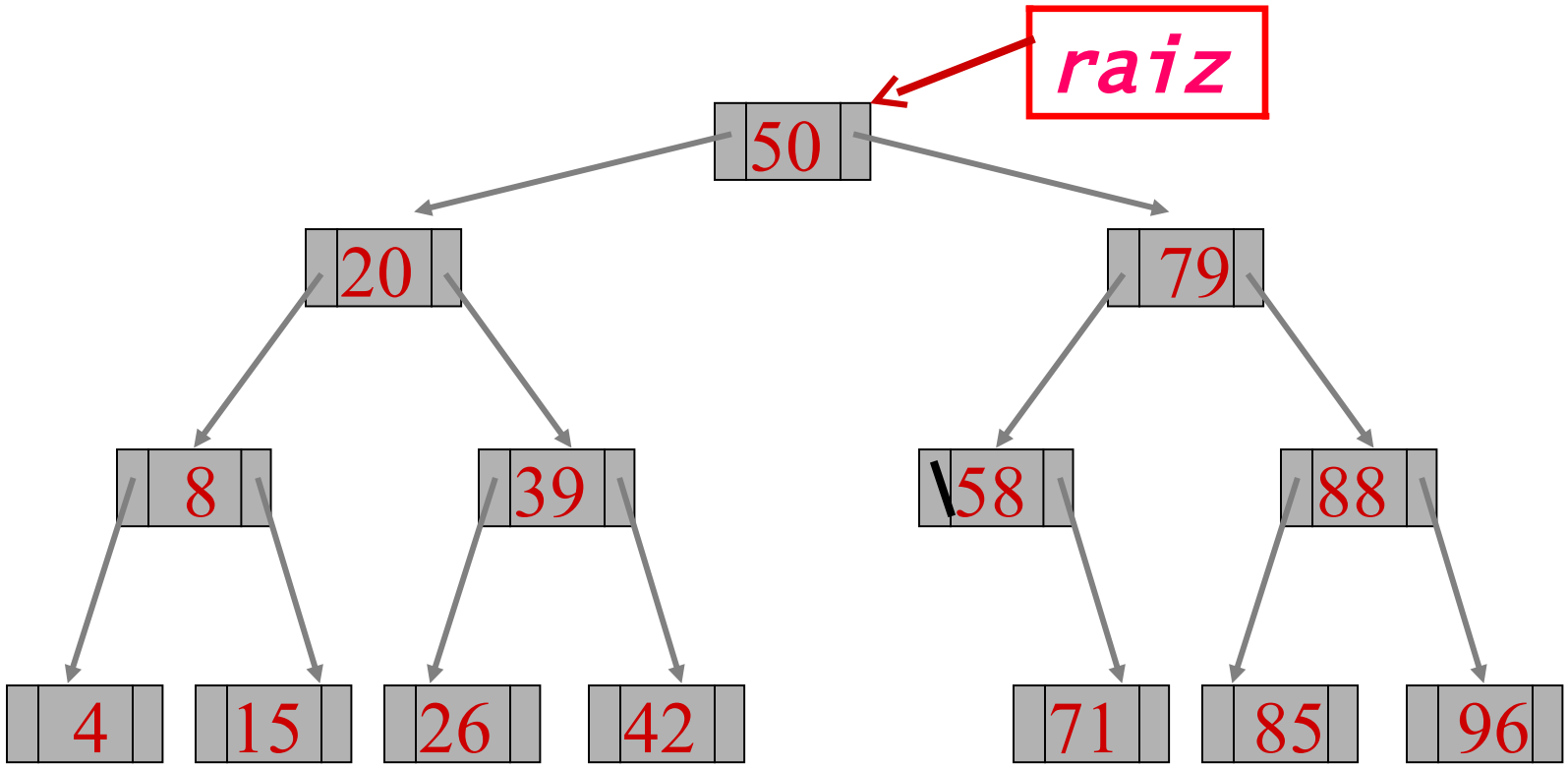
então apague apontador pai;

senão se ¬esq XOR ¬dir então

faça pai de *x* apontar para *filho* x

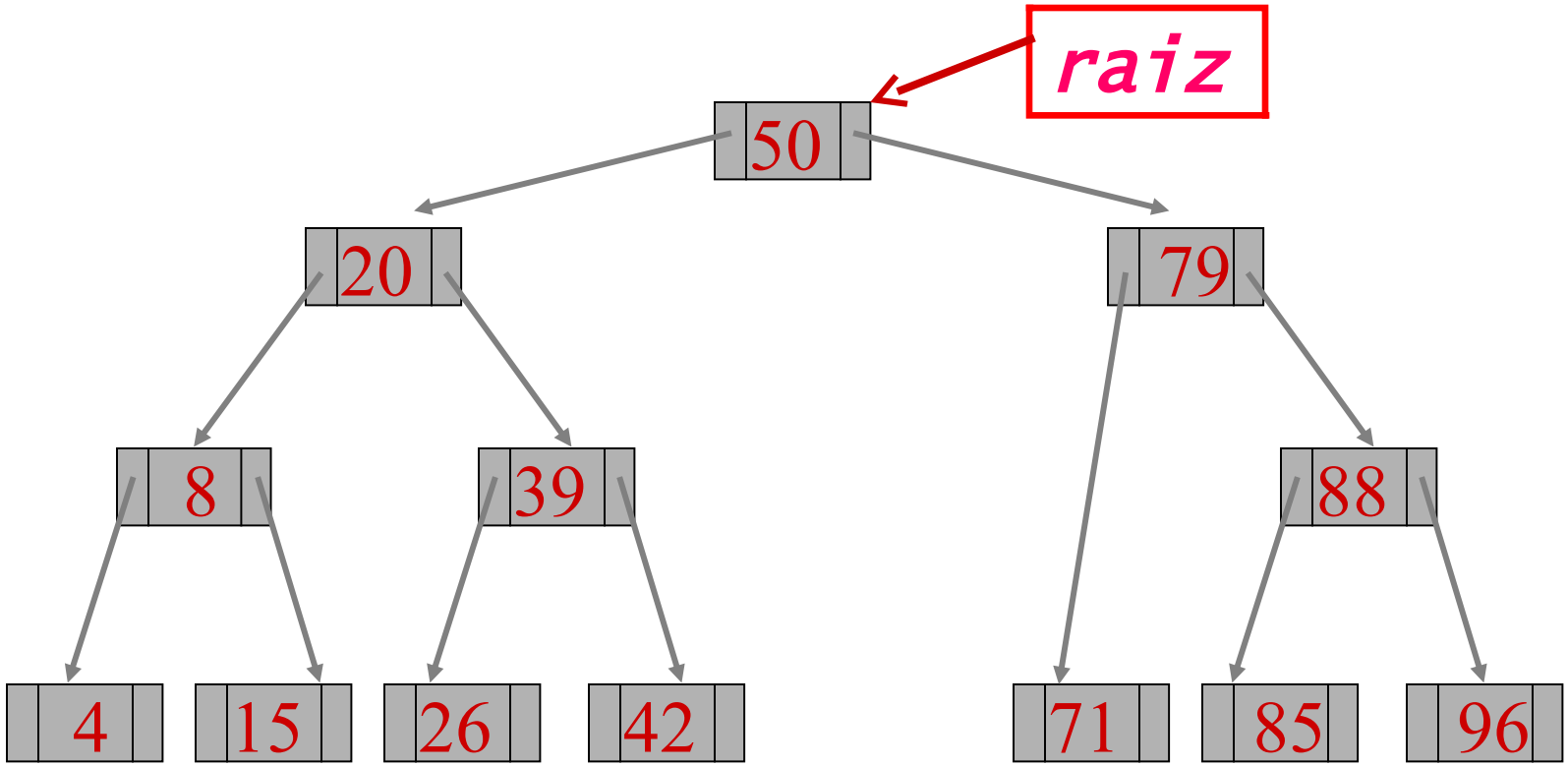
Remoção em Árvore de Busca

Exemplo: Remove (*raiz*, 58)



Remoção em Árvore de Busca

Exemplo: Remove (raiz, 58)



Remoção em Árvore de Busca

Procedimento Remove (raiz, *x*)

EndNó <- Busca (*x*);

Se ¬EndNó então relate insucesso

senão se EndNó é folha (¬esq AND ¬dir)

então apague apontador pai;

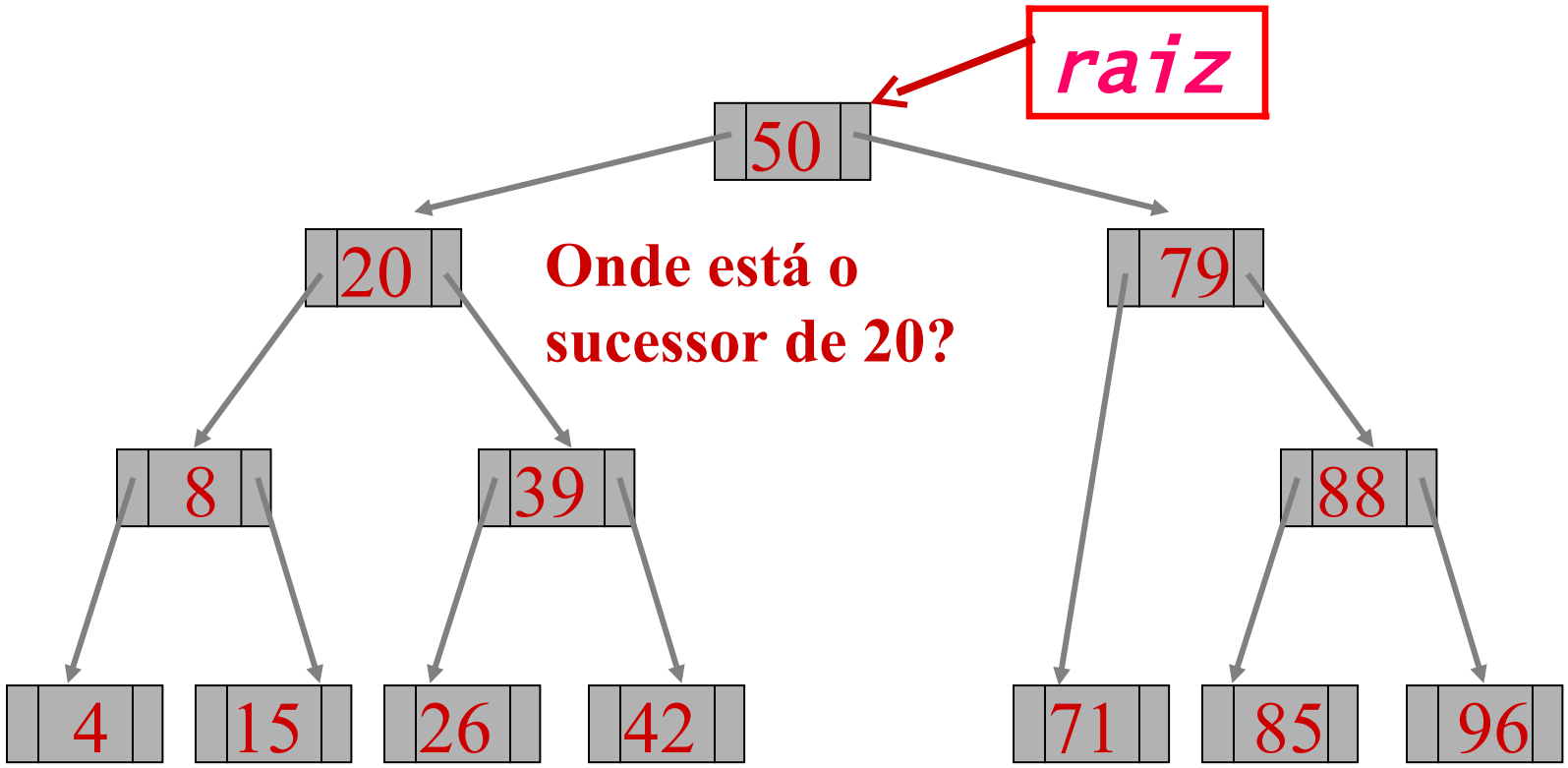
senão se ¬esq OR ¬dir então

faça pai de *x* apontar para filho x

senão substitua x pelo seu sucessor

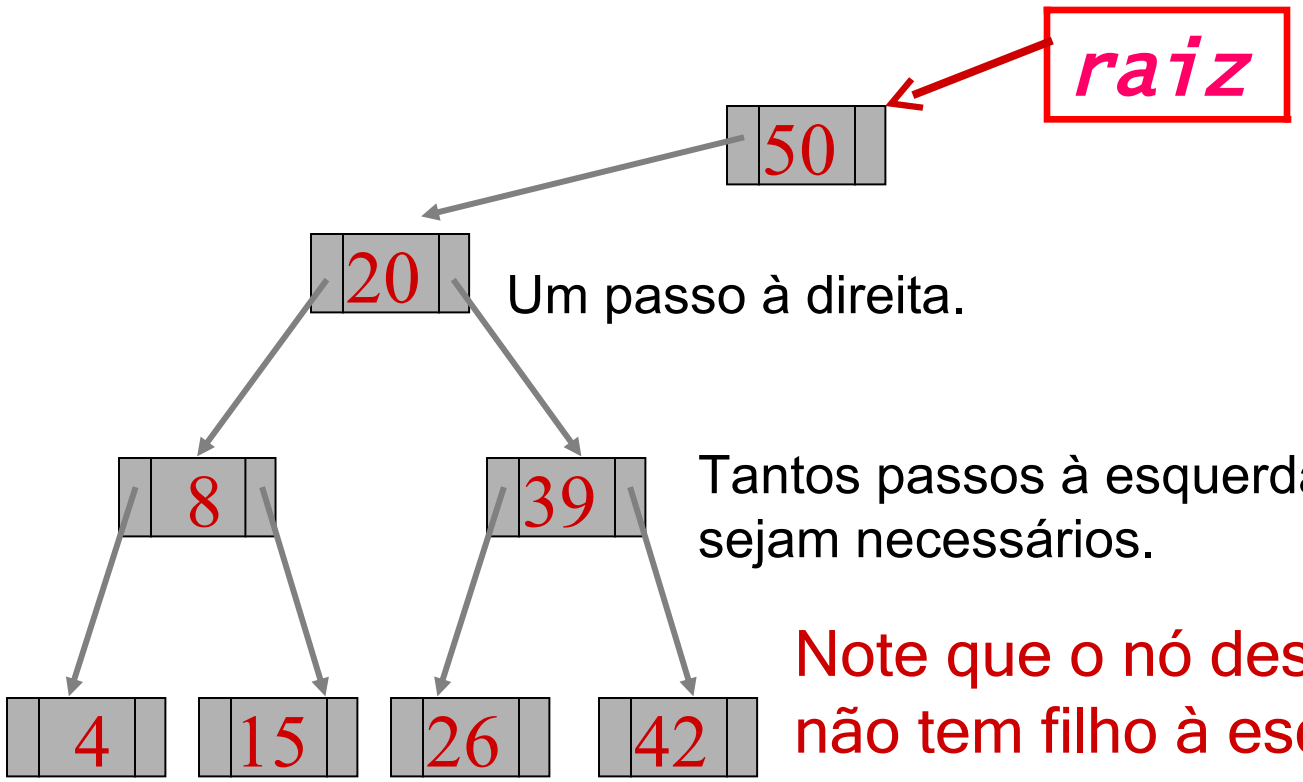
Remoção em Árvore de Busca

Exemplo: Remove (raiz, 20)



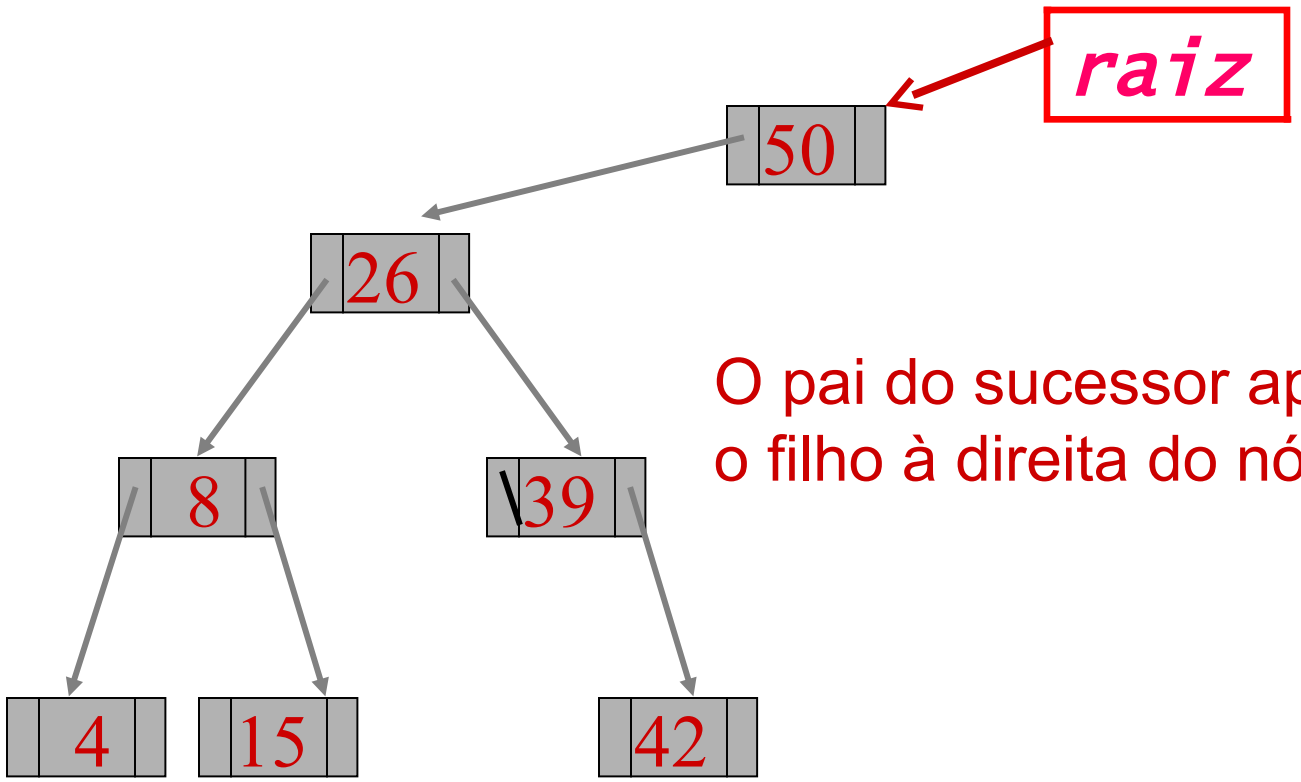
Remoção em Árvore de Busca

Onde está o sucessor de um elemento cujo nó tem filho à direita?



Remoção em Árvore de Busca

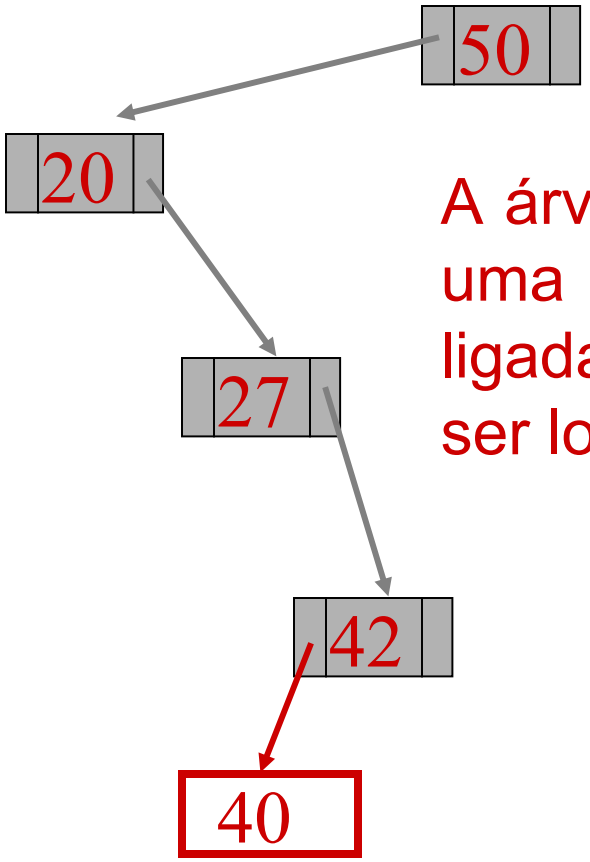
Exemplo: Remove (raiz, 20)



O pai do sucessor apontará para o filho à direita do nó do sucessor.

Possível Problema

Exemplo: 50, 20, 27, 42, 40 ...



A árvore binária pode degenerar para uma estrutura próxima a uma lista ligada, e o tempo de acesso deixa de ser logarítmico.

Solução

Procurar manter todas as folhas mais ou menos na mesma altura.

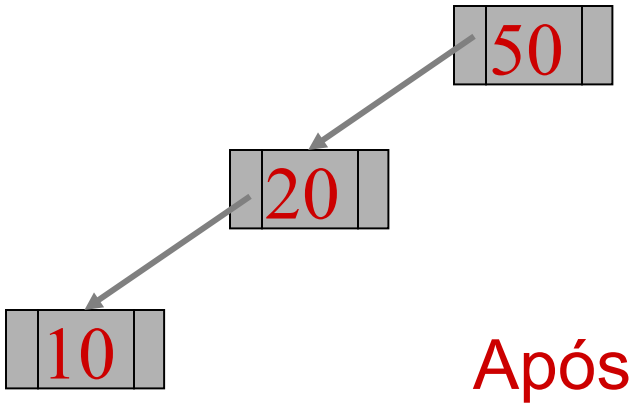
PROPRIEDADE :

Para todo nó

| altura(dir) - altura(esq) | < 2

Possível Problema

Exemplo: 50, 20, 10, ...



Após a inserção do elemento 10, a árvore binária perde a propriedade.

Percursos em Árvores Binárias

A busca nada mais é do que um percurso em uma árvore

O percurso em uma árvore visitando cada nó uma única vez gera uma seqüência linear de nós

Há três maneiras recursivas de se percorrer árvores binárias:

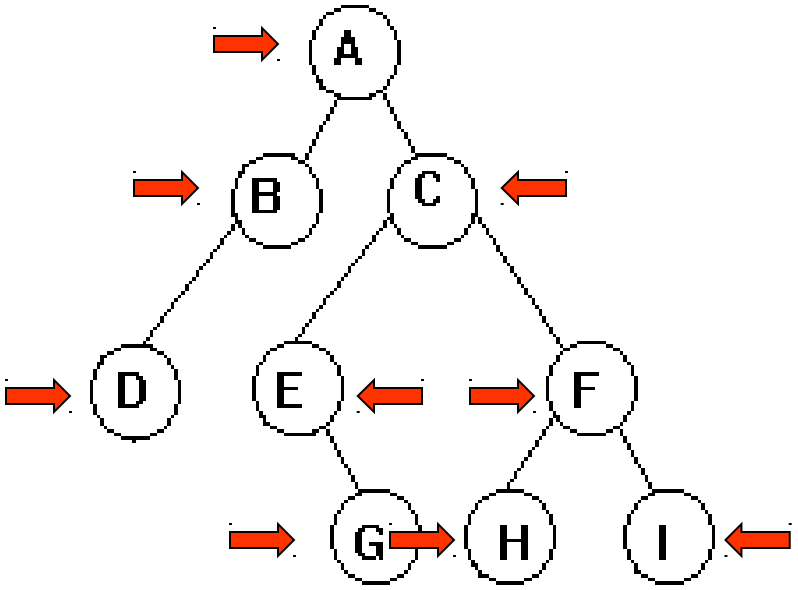
Percurso em pré-ordem (RED)

Percurso em pós-ordem (EDR)

Percurso em ordem (ERD) - central

Percurso em Pré-Ordem (RED)

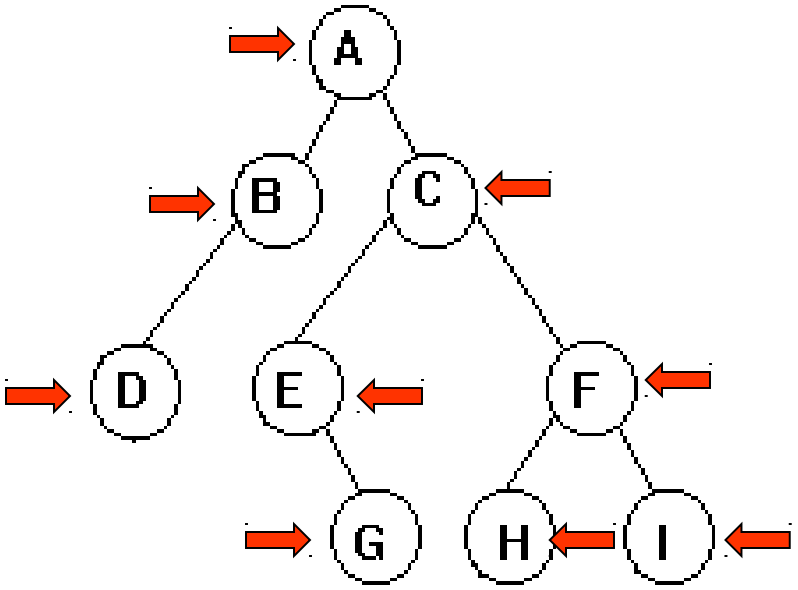
- **Algoritmo básico:**
  1. visitar o nó raiz
  2. sub-árvore esquerda
  3. sub-árvore direita



ABDCEGFHI

Percurso em Em Ordem

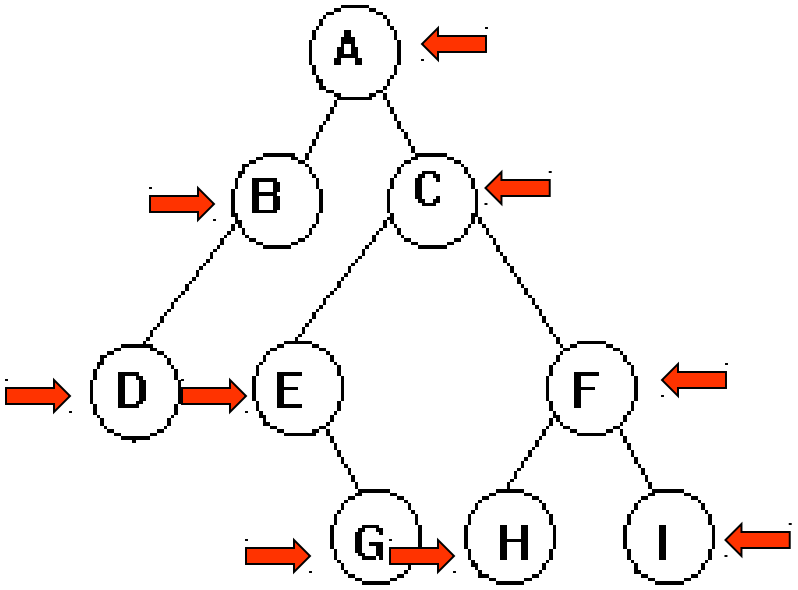
- **Algoritmo básico:**
  1. sub-árvore esquerda
  2. visitar o nó raiz
  3. sub-árvore direita



DBAEGCHFI

Percurso em Pós-Ordem

- **Algoritmo básico:**
  1. sub-árvore esquerda
  2. sub-árvore direita
  3. visitar o nó raiz



DBGEHIFCA