Árvores Binárias de Busca (ABB)

Árvores Binárias de Busca

- ***Motivação:***
  - Estruturas lineares (listas, vetores) têm limitações em buscas.
  - Árvores permitem buscas, inserções e remoções mais rápidas.
  - Aplicações: bancos de dados, compiladores, sistemas de arquivos.

Árvores Binárias de Busca

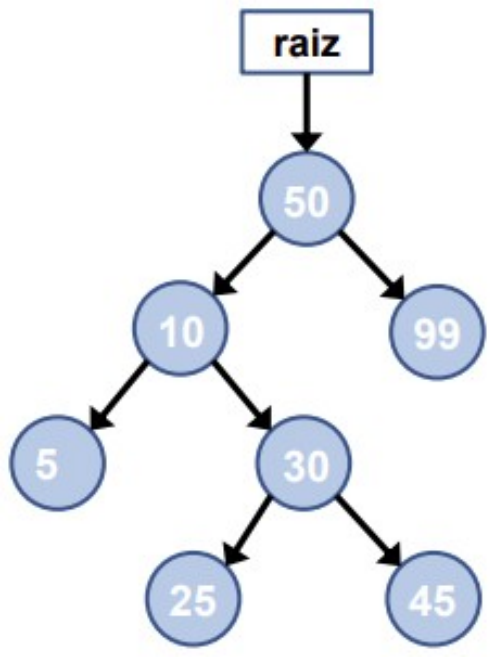
- ***Revisão de árvores:***
  - Estrutura hierárquica de nós.
  - Cada nó: valor + filhos.
  - Definições:
    - Raiz: nó inicial.
    - Folha: nó sem filhos.
    - Altura: maior caminho até uma folha.

Árvores Binárias de Busca

- ***O que é uma ABB?***
  - Árvore binária especial.
  - Propriedade:
    - Valores menores → subárvore esquerda.
    - Valores maiores → subárvore direita.
  - Cada nó da árvore possui um valor (chave) associado a ele. Não existem valores repetidos.
  - Busca eficiente: divide o problema a cada passo.

Árvores Binárias de Busca

- ***O que é uma ABB?***



Árvores Binárias de Busca

- ***Estrutura de Nó em C:***

```
typedef struct No {
    int valor;
    struct No *esq, *dir;
} No;
```

Árvores Binárias de Busca

- ***Criação de Nó:***

```
No* novoNo(int valor) {
    No* n = (No*) malloc(sizeof(No));
    n->valor = valor;
    n->esq = n->dir = NULL;
    return n;
}
```

Árvores Binárias de Busca

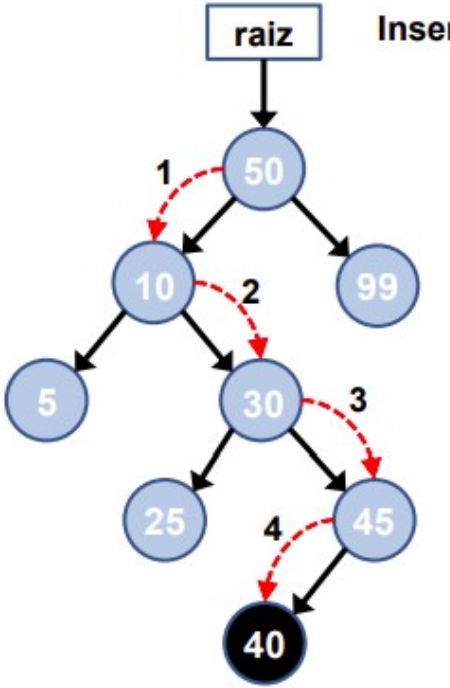
- ***Inserção na ABB:***
  - Se árvore vazia → novo nó é a raiz.
  - Se valor < raiz → inserir na esquerda.
  - Se valor > raiz → inserir na direita.

Árvores Binárias de Busca

- ***Inserção na ABB (cont.):***

Árvores Binárias de Busca

- ***Inserção na ABB (cont.):***



Insere o valor '40' como um nó folha

1	valor é menor do que 50: visita filho da esquerda
2	valor é maior do que 10: visita filho da direita
3	valor é maior do que 30: visita filho da direita
4	valor é menor do que 45: visita filho da esquerda
Fim	Não existe filho da esquerda. Valor passa a ser o filho da esquerda de 45

Árvores Binárias de Busca

- ***Inserção na ABB (cont.):***

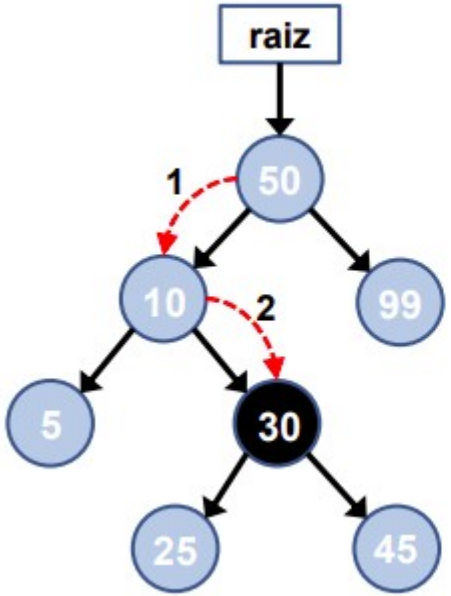
```
No* inserir(No* raiz, int valor) {
    if (raiz == NULL) return novoNo(valor);
    if (valor < raiz->valor)
        raiz->esq = inserir(raiz->esq, valor);
    else if (valor > raiz->valor)
        raiz->dir = inserir(raiz->dir, valor);
    return raiz;
}
```

Árvores Binárias de Busca

- ***Busca em ABB:***
  - Começa na raiz.
  - Se valor == raiz → achou.
  - Se valor < raiz → vai para esquerda.
  - Se valor > raiz → vai para direita.

Árvores Binárias de Busca

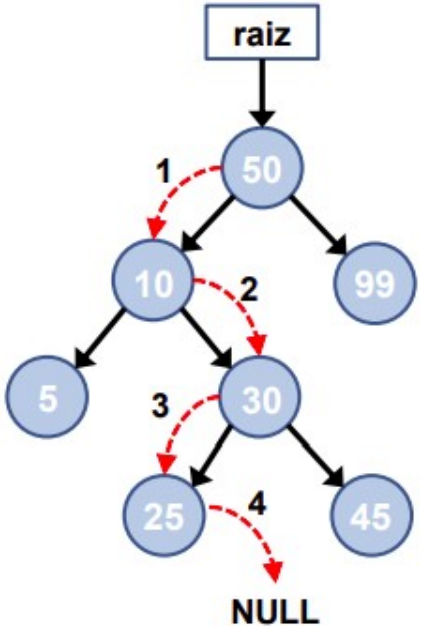
- ***Busca em ABB (cont.):***



Valor procurado: 30	
1	valor procurado é menor do que 50: visita filho da esquerda
2	valor procurado é maior do que 10: visita filho da direita
Fim	valor procurado é igual ao do nó: retornar dados do nó

Árvores Binárias de Busca

- ***Busca em ABB (cont.):***



Valor procurado: 28

1	valor procurado é menor do que 50: visita filho da esquerda
2	valor procurado é maior do que 10: visita filho da direita
3	valor procurado é menor do que 30: visita filho da esquerda
4	valor procurado é maior do que 25: visita filho da direita
Fim	Filho da direita de 25 não existe: a busca falhou

Árvores Binárias de Busca

- ***Busca em ABB (cont.):***

```
No* buscar(No* raiz, int valor) {
    if (raiz == NULL || raiz->valor == valor) return raiz;
    if (valor < raiz->valor)
        return buscar(raiz->esq, valor);
    else
        return buscar(raiz->dir, valor);
}
```

Árvores Binárias de Busca

- ***Travessias em ABB:***
  - Pré-ordem: raiz → esquerda → direita.
  - Em-ordem: esquerda → raiz → direita.
  - Pós-ordem: esquerda → direita → raiz.

Árvores Binárias de Busca

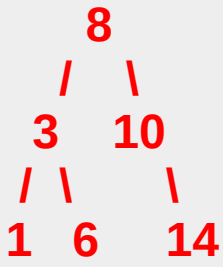
- ***Travessias em ABB (cont.):***

```
void emOrdem(No* raiz) {
    if (raiz != NULL) {
        emOrdem(raiz->esq);
        printf("%d ", raiz->valor);
        emOrdem(raiz->dir);
    }
}
```

Árvores Binárias de Busca

- ***Travessias em ABB (cont.):***

  Ex.: Inserir os valores em uma ABB: *8, 3, 10, 1, 6, 14.*



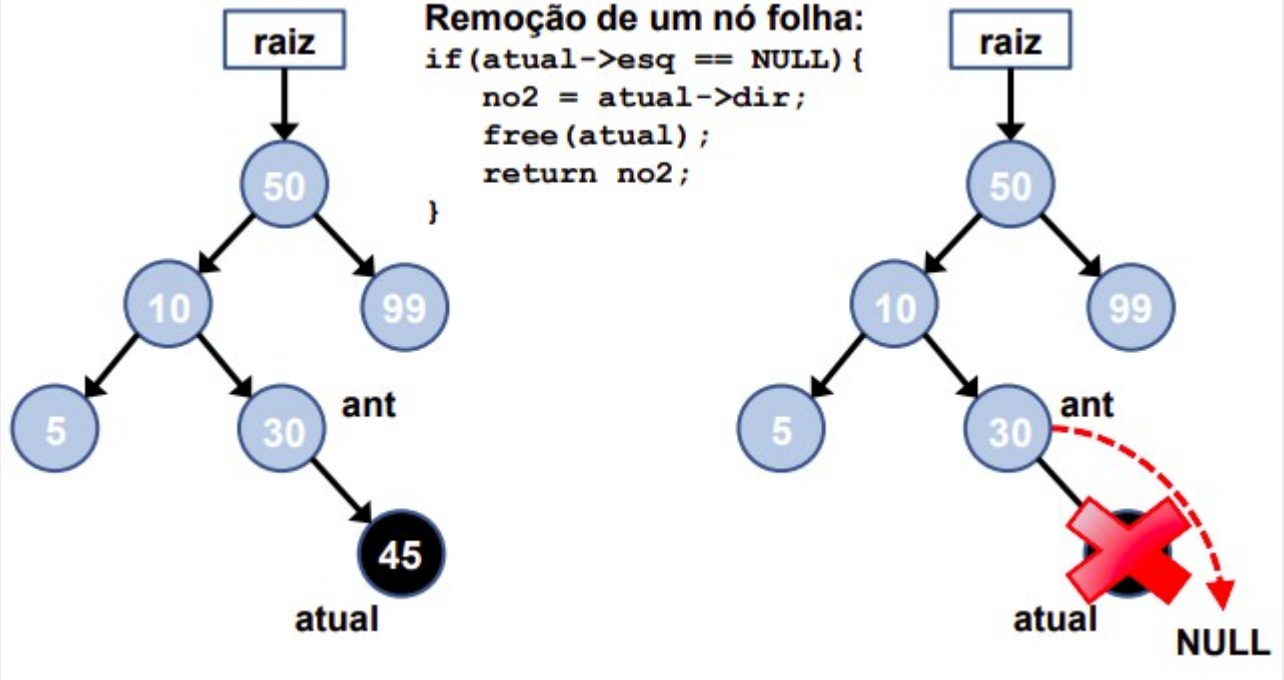
Resultado: 1 3 6 8 10 14.

Árvores Binárias de Busca

- ***Remoção em ABB:***
  - Três casos:
    - Nó sem filhos → remover diretamente.
    - Nó com 1 filho → substitui pelo filho.
    - Nó com 2 filhos → substitui pelo sucessor em-ordem.

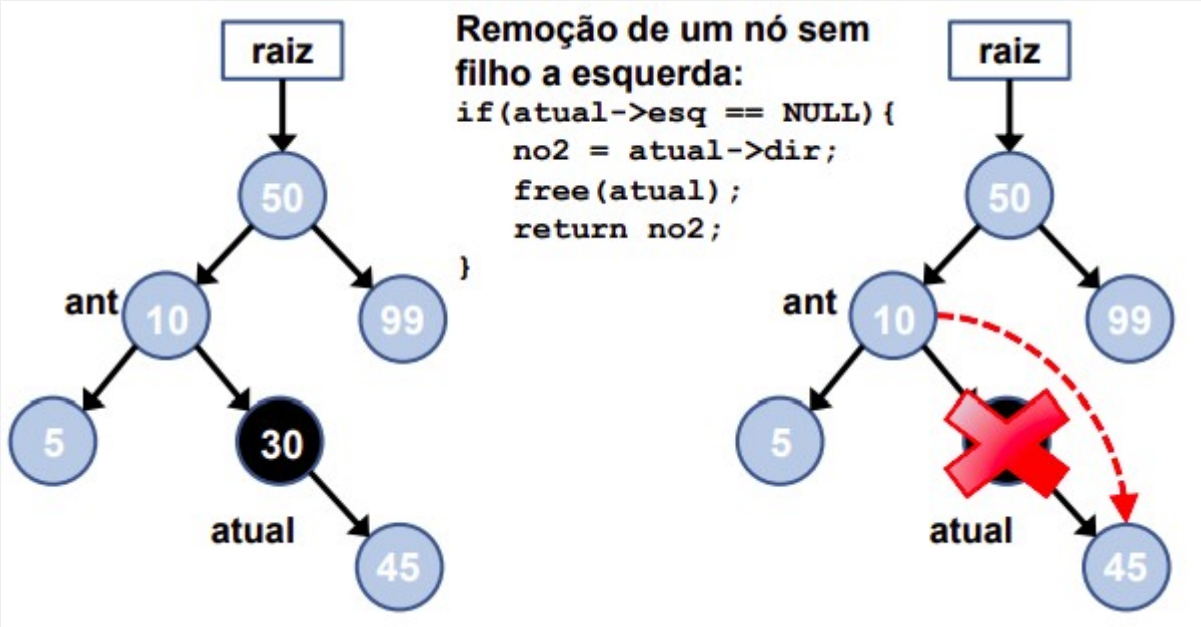
Árvores Binárias de Busca

- ***Remoção em ABB (cont.):***



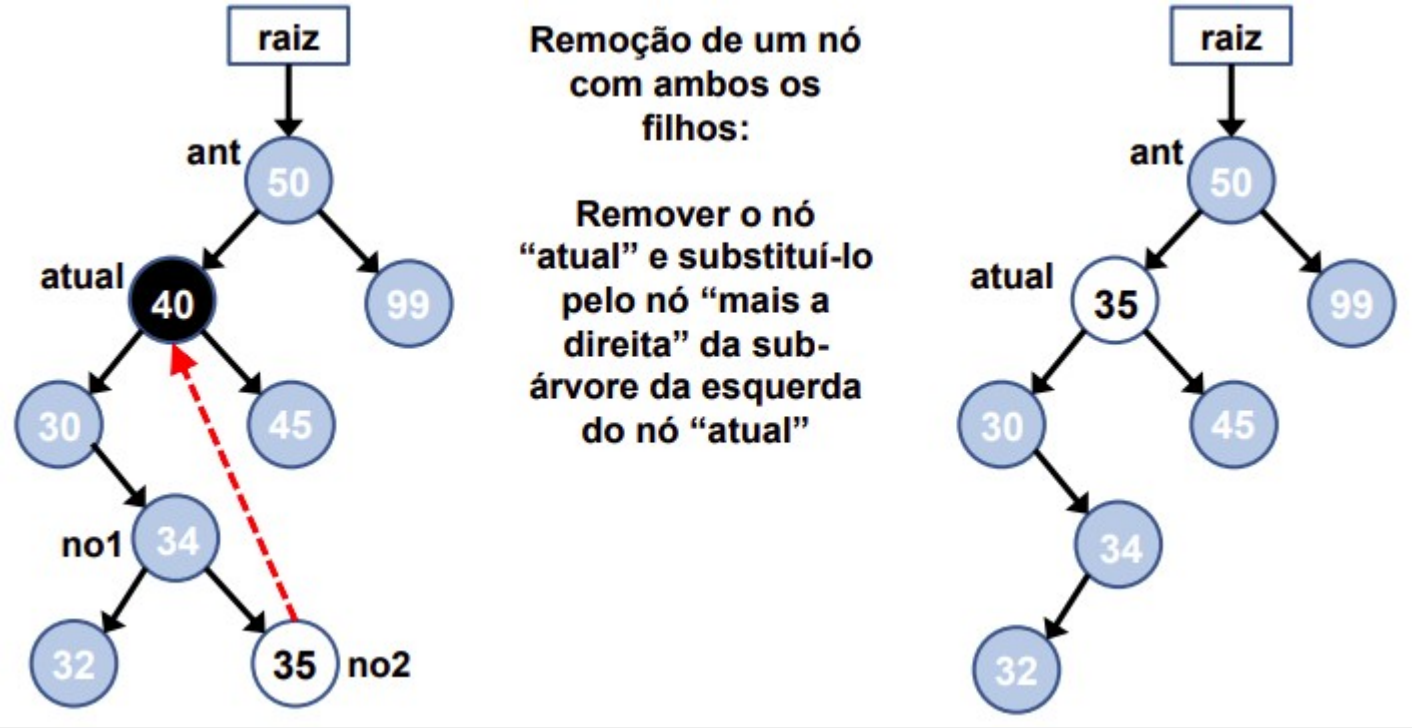
Árvores Binárias de Busca

- ***Remoção em ABB (cont.):***



Árvores Binárias de Busca

- ***Remoção em ABB (cont.):***



Árvores Binárias de Busca

- ***Remoção em ABB (cont.):***

```
No* remover(No* raiz, int v) {
    if (raiz == NULL) return NULL;
    if (v < raiz->valor)    raiz->esq = remover(raiz->esq, v);
    else if (v > raiz->valor)    raiz->dir = remover(raiz->dir, v);
    else {
        if (raiz->esq == NULL && raiz->dir == NULL) { free(raiz); return NULL; }
        else if (raiz->esq == NULL) { No* tmp = raiz->dir; free(raiz); return tmp; }
        else if (raiz->dir == NULL) { No* tmp = raiz->esq; free(raiz); return tmp; }
        else {
            No* pred = maiorNo(raiz->esq);
            raiz->valor = pred->valor;
            raiz->esq = remover(raiz->esq, pred->valor);
       }    }
    return raiz;}
```

Árvores Binárias de Busca

- ***Encontrar Menor Valor em ABB:***

```
No* maiorNo(No* raiz) {
    No* atual = raiz;
    while(atual && atual->dir)
        atual = atual->dir;
    return atual;
}
```

Árvores Binárias de Busca

- ***Funções Utilitárias para ABB:***
  - Contar nós.
  - Calcular altura.
  - Verificar se é ABB válida.

Árvores Binárias de Busca

- ***Vantagens da ABB:***
  - Busca, inserção e remoção em O(log n) (em média).
  - Mantém elementos ordenados.
  - Estrutura flexível (dinâmica).

Árvores Binárias de Busca

- ***Comparação:***
  - Lista encadeada: busca O(n).
  - ABB balanceada: busca O(log n).
  - Hash Table: busca O(1) médio, mas sem ordem natural.

Árvores Binárias de Busca

- ***Aplicações:***
  - Compiladores: tabelas de símbolos.
  - Sistemas de arquivos: indexação.
  - Bancos de dados: índices ordenados.
  - Algoritmos: suporte a dicionários e conjuntos.

Árvores Binárias de Busca

- ***Atividade:***

1. Modularizar o código fornecido.
2. Construa as árvores:

a) 50, 30, 70, 20, 40, 60, 80, 10, 25, 35, 45, 55, 65, 75, 85

Remover: 10 / Remover: 25 / Remover: 50

b) 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120

Remover: 120 / Remover: 100 / Remover: 60

c) 50, 40, 99, 30, 45, 34, 32, 35, 70, 65, 85, 100, 95, 110

Remover: 95 / Remover: 100 / Remover: 99

Fim da aula