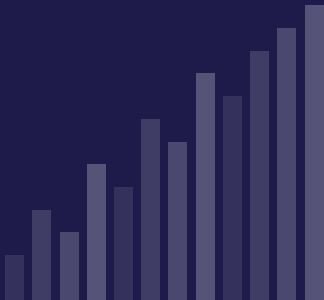

AULA 23 · BLOCO 3: TABELAS HASH E ÁRVORES

Inserção, remoção e altura das árvores de busca

Seções 12.3 e 12.4

Estruturas de dados e seus algoritmos · José Dihego · IFBA



Roteiro da aula

Ao final, você será capaz de

- ▶ Inserir mantendo a propriedade de ABB
- ▶ Remover nos três casos com TRANSPLANT
- ▶ Entender por que a altura esperada de uma ABB aleatória é $O(\lg n)$
- ▶ Medir a altura de árvores aleatórias e o desastre da entrada ordenada



Teoria

Tree-INSERT, Tree-DELETE e a altura esperada
(teorema 12.4)



Prática em C

Inserção e remoção em C



Experimento

Altura média de árvores aleatórias; chaves em ordem



Conexões

Bancos de dados, editores, jogos e redes

Inserção

Desce como numa busca e pendura z como **folha**.
Custo $O(h)$.

Tree-INSERT(T, z)

```

1  $y \leftarrow \text{NIL}; \quad x \leftarrow T.\text{raiz}$ 
2 enquanto  $x \neq \text{NIL}$  faça
3      $y \leftarrow x$                                      //  $y$  segue  $x$  um passo atrás
4     se  $z.\text{chave} < x.\text{chave}$  então
5          $x \leftarrow x.\text{esq}$ 
6     senão
7          $x \leftarrow x.\text{dir}$ 
8  $z.\text{pai} \leftarrow y$ 
9 se  $y = \text{NIL}$  então
10     $T.\text{raiz} \leftarrow z$                                // árvore vazia
11 senão se  $z.\text{chave} < y.\text{chave}$  então
12     $y.\text{esq} \leftarrow z$ 
13 senão
14     $y.\text{dir} \leftarrow z$ 

```

Inserção (exemplo)

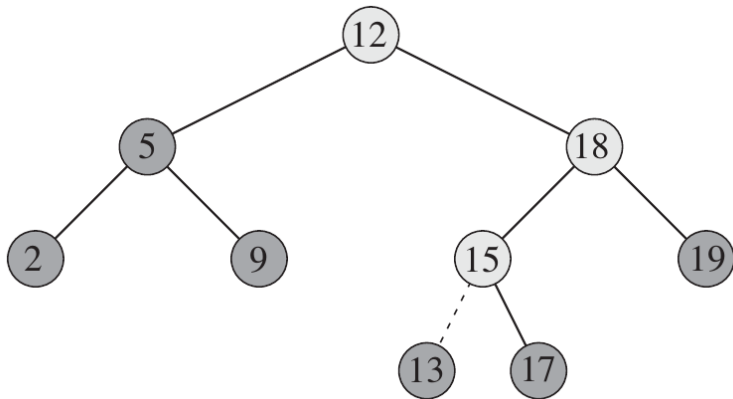


Figura 12.3 de Cormen et al. (2009) — inserção de 13: desce até uma posição vazia e vira folha

Remoção: os três casos

- ▶ **Sem filhos ou um filho:** o filho (ou NIL) toma o lugar de z .
- ▶ **Dois filhos:** o **sucessor** y (mínimo da subárvore direita, que não tem filho esquerdo) toma o lugar de z .
- ▶ $\text{TRANSPLANT}(T, u, v)$ põe a subárvore v no lugar de u .
- ▶ Custo $O(h)$.

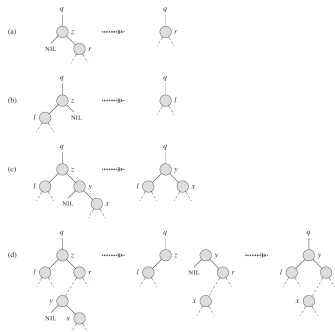


Figura 12.4 de Cormen et al. (2009) — (a) sem filho esquerdo; (b) sem filho direito; (c) sucessor é o filho direito; (d) sucessor mais abaixo.

Qual a altura de uma árvore aleatória?

Inserindo n chaves distintas em ordem **aleatória** (todas as $n!$ ordens igualmente prováveis):

- ▶ **Teorema 12.4:** a altura esperada é $O(\lg n)$.
- ▶ Ideia da prova: com $Y_n = 2^h$ e a raiz escolhida ao acaso, $E[Y_n] \leq \frac{4}{n} \sum_i E[Y_i] = O(n^3)$; pela desigualdade de Jensen, $E[h] \leq 3 \lg n + O(1)$.
- ▶ Resultado exato (Reed, 2003): $E[h] \approx 4,311 \ln n - 1,953 \ln \ln n \approx 2,99 \lg n$.
- ▶ Mas a entrada real raramente é aleatória: chaves **em ordem** geram uma corrente de altura $n - 1$.

Solução: árvores que se reequilibram sozinhas, garantindo $h = O(\lg n)$ para **qualquer** ordem de inserção (aulas 24 e 25).



Prática em C

- ▶ transplanta: o coração da remoção
- ▶ Os quatro casos de Tree-Delete, como no livro
- ▶ Entrada: chaves e comandos r k

Inserção e remoção em ABB (1/5)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  typedef struct No { int chave; struct No *esq, *dir, *pai; } No;
5  typedef struct { No *raiz; } Arvore;
6
7  void insere(Arvore *T, int k) {                                // TREE-INSERT
8      No *z = calloc(1, sizeof(No)), *y = NULL, *x = T->raiz;
9      z->chave = k;
10     while (x) { y = x; x = k < x->chave ? x->esq : x->dir; }
11     z->pai = y;
12     if (!y) T->raiz = z;
13     else if (k < y->chave) y->esq = z;
14     else y->dir = z;
15 }
```

Inserção e remoção em ABB (2/5)

```
16
17  /* TRANSPLANT: põe a subárvore v no lugar da subárvore u. */
18  void transplanta(Arvore *T, No *u, No *v) {
19      if (!u->pai) T->raiz = v;
20      else if (u == u->pai->esq) u->pai->esq = v;
21      else u->pai->dir = v;
22      if (v) v->pai = u->pai;
23  }
24
25  No *minimo(No *x) { while (x->esq) x = x->esq; return x; }
26
27  void remove_no(Arvore *T, No *z) { // TREE-DELETE: três casos
28      if (!z->esq) transplanta(T, z, z->dir); // (a) sem filho esquerdo
29      else if (!z->dir) transplanta(T, z, z->esq); // (b) sem filho direito
30      // (c)/(d) dois filhos: entra o sucessor y
```

Inserção e remoção em ABB (3/5)

```
31     else {
32         No *y = minimo(z->dir);
33         if (y->pai != z) {
34             transplanta(T, y, y->dir);
35             y->dir = z->dir;
36             y->dir->pai = y;
37         }
38         transplanta(T, z, y);
39         y->esq = z->esq;
40         y->esq->pai = y;
41     }
42     free(z);
43 }
```

Inserção e remoção em ABB (4/5)

```
45 No *busca(No *x, int k) {
46     while (x && k != x->chave) x = k < x->chave ? x->esq : x->dir;
47     return x;
48 }
49
50 void em_ordem(No *x) {
51     if (!x) return;
52     em_ordem(x->esq);
53     printf(" %d", x->chave);
54     em_ordem(x->dir);
55 }
56 void libera(No *x) { if (x) { libera(x->esq); libera(x->dir); free(x); } }
57
58 int main(void) {
59     Arvore T = {NULL};
60     char s[16];
```

Inserção e remoção em ABB (5/5)

```
61     while (scanf("%15s", s) == 1) {
62         if (s[0] == 'r') {                                     // "r k" remove a chave k
63             int k;
64             if (scanf("%d", &k) != 1) break;
65             No *z = busca(T.raiz, k);
66             if (z) remove_no(&T, z);
67         } else insere(&T, atoi(s));
68     }
69     printf("em ordem:");
70     em_ordem(T.raiz);
71     printf("\n");
72     if (T.raiz) printf("raiz %d\n", T.raiz->chave);
73     libera(T.raiz);
74     return 0;
75 }
```



Experimento

- ▶ Aleatória: altura entre $\lg n$ e $3 \lg n$ (teorema 12.4)
- ▶ Ordenada: exatamente $n - 1$
- ▶ n até 2^{21} , média de 10 árvores

Altura: ordem aleatória contra ordem crescente

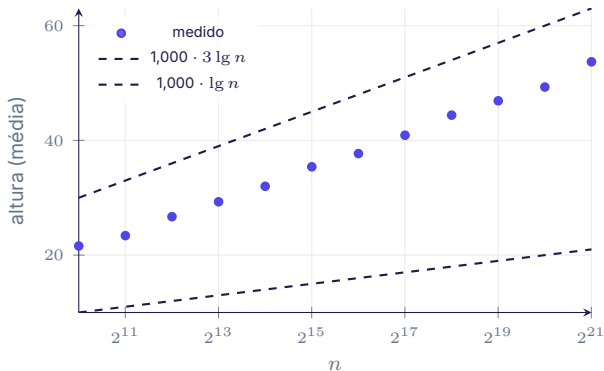
Hipótese. Com chaves aleatórias, a altura deve ficar entre o mínimo possível, $\lg n$, e o limite da prova, $3 \lg n$, aproximando-se de $\approx 2,99 \lg n$ bem devagar. Com chaves em ordem crescente, cada chave vira filho direito da anterior: altura exatamente $n - 1$.

Método

1. Altura registrada durante as inserções
2. Aleatória: $n = 2^{10}, \dots, 2^{21}$, média de 10 árvores
3. Ordenada: n de 500 a 8.000
4. Verificação dos limites em todos os pontos

🧪 EXPERIMENTO

Altura de árvores aleatórias



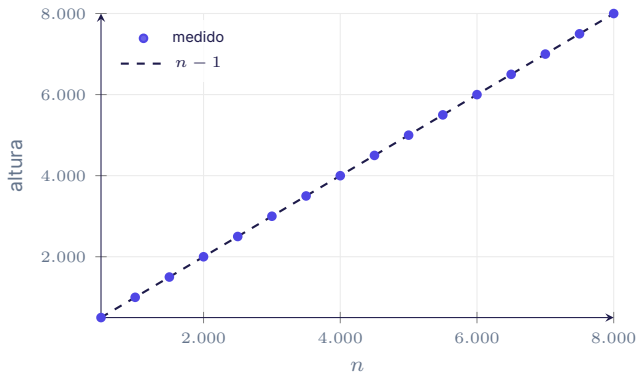
n	medido	razão
1.024	21,6	0,720
4.096	26,7	0,742
16.384	32,0	0,762
65.536	37,7	0,785
262.144	44,4	0,822
2.097.152	53,7	0,852

razão = medido $\div$ $3 \lg n$

✓ sempre $\leq$ o limite

A razão altura / $\lg n$ sobe devagar (2,2 para 2,6) rumo ao valor assintótico 2,99.

Chaves em ordem: uma corrente



n	medido	razão
500,0	499,0	1,000
2.000	1.999	1,000
3.500	3.499	1,000
5.000	4.999	1,000
6.500	6.499	1,000
8.000	7.999	1,000

razão = medido $\div$ $n - 1$

✓ desvio máx. 0,0%

Altura $n - 1$: a árvore é uma lista ligada, e todas as operações viram $\Theta(n)$.

EXPERIMENTO

O que o experimento mostrou

Com chaves aleatórias, a altura fica sempre entre $\lg n$ e $3 \lg n$: para dois milhões de chaves, cerca de 54 níveis. Com chaves em ordem, exatamente $n - 1$: dois milhões de níveis. **A mesma estrutura, com a mesma quantidade de dados, fica 40.000 vezes mais funda** só pela ordem de chegada.

➤ Reproduza: `gcc -O2 aula23_experimento.c -o exp -lm && ./exp`



Conexões

- ▶ Bancos de dados
- ▶ Editores e cordas
- ▶ Jogos
- ▶ Redes

Onde isso aparece na computação

 Bancos de dados

Chaves autoincrementadas chegam *em ordem*: é exatamente o pior caso de uma ABB comum. Por isso bancos usam árvores balanceadas (B-trees).

 Editores e cordas

Editores de texto grandes guardam o documento numa árvore de trechos (*rope*); inserir e remover no meio custa $O(h)$.

 Jogos

Placares (*leaderboards*) mantêm jogadores ordenados por pontuação com inserção e remoção constantes; a altura define a latência.

 Redes

Tabelas de roteamento e de temporizadores de conexões TCP mudam o tempo todo: remoções e inserções eficientes são essenciais.

Para casa

Exercícios

1. (12.3-1) Escreva Tree-INSERT recursivo.
2. (12.3-3) Ordene n números inserindo-os numa ABB e percorrendo em ordem. Pior e melhor caso?
3. (12.3-4) A remoção é "comutativa? Remover x e depois y dá a mesma árvore que y e depois x ?
4. Mostre que existe ABB aleatória com altura média maior que a esperada, e explique por que isso não contradiz o teorema.
5. Altere o experimento: insira as chaves quase em ordem (1% trocadas). Qual a altura?

Leitura

Cormen et al., seções 12.3 e 12.4.

Código

josedihego.net/estruturas