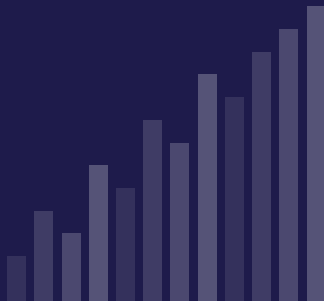

AULA 10 · BLOCO 2: DICIONÁRIOS E ESTRUTURAS LINEARES

Dicionários e conjuntos dinâmicos

Introdução da parte III e seção 10.1

Estruturas de dados e seus algoritmos · José Dihego · IFBA



Roteiro da aula

Ao final, você será capaz de

- ▶ Definir conjunto dinâmico e as operações de dicionário
- ▶ Separar consultas (busca, mínimo, máximo, sucessor) de modificações (inserção, remoção)
- ▶ Implementar um dicionário em vetor ordenado com busca binária
- ▶ Medir o custo de inserir e de buscar em vetor ordenado e desordenado



Teoria

O tipo abstrato de dados, as operações e uma tabela de custos



Prática em C

Dicionário em vetor ordenado com consultas de ordem



Experimento

Vetor ordenado contra desordenado: $n^2/4$, $n \lg n$ e $n^2/2$



Conexões

Bancos de dados, linguagens, compiladores e redes

Conjuntos dinâmicos

- ▶ Um **conjunto dinâmico** cresce, encolhe e muda ao longo da execução do algoritmo.
- ▶ Cada elemento é um objeto com uma **chave** (que o identifica) e **dados satélite**.
- ▶ Um **dicionário** é um conjunto dinâmico com INSERT, DELETE e SEARCH.
- ▶ Separar o **tipo abstrato** (o que as operações fazem) da **implementação** (como): o mesmo programa pode trocar a estrutura por baixo.

Os blocos 2 e 3 são sobre isso: o mesmo dicionário em vetores, listas, tabelas hash e árvores, cada um com seus custos.

As operações (Cormen, parte III)

Consultas

- ▶ $\text{SEARCH}(S, k)$: o elemento de chave k , OU NIL.
- ▶ $\text{MINIMUM}(S)$, $\text{MAXIMUM}(S)$.
- ▶ $\text{SUCCESSOR}(S, x)$, $\text{PREDECESSOR}(S, x)$: o próximo e o anterior na ordem das chaves.

Modificações

- ▶ $\text{INSERT}(S, x)$: acrescenta x .
- ▶ $\text{DELETE}(S, x)$: remove x (dado o elemento, não só a chave).

Analisa-se o custo de cada operação em função de $n = |S|$.

Primeiras implementações

	Search	Insert	Delete	Min/Max	Success
vetor desordenado	$\Theta(n)$	$O(1)$	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$
vetor ordenado	$\Theta(\lg n)$	$\Theta(n)$	$\Theta(n)$	$O(1)$	$O(\lg n)$
lista ligada (aulas 14–16)	$\Theta(n)$	$O(1)$	$O(1)$ dado o nó	$\Theta(n)$	$\Theta(n)$
endereçamento direto (aula 19)	$O(1)$	$O(1)$	$O(1)$	$\Theta(m)$	$\Theta(n)$
tabela hash (aulas 19–21)	$O(1)$ médio	$O(1)$	$O(1)$	$\Theta(n)$	$\Theta(n)$
árvore balanceada (aulas 24–25)	$O(\lg n)$	$O(\lg n)$	$O(\lg n)$	$O(\lg n)$	$O(\lg n)$

Nenhuma estrutura vence em tudo: a escolha depende de quais operações o programa mais usa.

Busca binária num vetor ordenado

- ▶ Cada comparação descarta metade do intervalo:

$$T(n) = T(n/2) + \Theta(1) = \Theta(\lg n)$$

(exercício 2.3-5).

- ▶ Inserir mantendo a ordem exige deslocar, em média, metade dos elementos: $\Theta(n)$.
- ▶ Em C, $lo + (hi - lo) / 2$ evita estouro de inteiro.

BINARY-SEARCH(*A*, *k*)

```
1 lo ← 1; hi ← n
2 enquanto lo ≤ hi faça
3   m ← ⌊(lo + hi)/2⌋
4   se A[m] = k então
5     retorne m
6   senão se A[m] < k então
7     lo ← m + 1
8   senão
9     hi ← m - 1
10  retorne NIL
```



Prática em C

- ▶ Busca binária que devolve a posição de inserção
- ▶ Inserção e remoção com deslocamento; mínimo e máximo em $O(1)$
- ▶ Sucessor em $O(\lg n)$; comandos i, r, b, m, M, s, p

Dicionário em vetor ordenado (1/5)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  /* Um conjunto dinâmico com as operações do Cormen (introdução da parte III):
5     SEARCH, INSERT, DELETE, MINIMUM, MAXIMUM, SUCCESSOR. Comandos da entrada:
6     i k, r k, b k (busca), m (mínimo), M (máximo), s k (sucessor), p (imprime). */
7  typedef struct { int *v; int n, cap; } Dicionario;
8
9  // busca binária: primeiro índice com v >= k
10 int posicao(const Dicionario *D, int k) {
11     int lo = 0, hi = D->n;
12     while (lo < hi) {
13         int meio = lo + (hi - lo) / 2;
```

Dicionário em vetor ordenado (2/5)

```
14         if (D->v[meio] < k) lo = meio + 1; else hi = meio;
15     }
16     return lo;                                // O(lg n)
17 }
18
19 int busca(const Dicionario *D, int k) {
20     int i = posicao(D, k);
21     return i < D->n && D->v[i] == k;
22 }
```

Dicionário em vetor ordenado (3/5)

```
24 void insere(Dicionario *D, int k) {           // abre espaço deslocando: O(n)
25     int i = posicao(D, k);
26     if (i < D->n && D->v[i] == k) return;      // conjunto: sem repetições
27     if (D->n == D->cap) {
28         D->cap = D->cap ? 2 * D->cap : 8;
29         D->v = realloc(D->v, D->cap * sizeof(int));
30     }
31     for (int j = D->n; j > i; j--) D->v[j] = D->v[j - 1];
32     D->v[i] = k;
33     D->n++;
34 }
```

Dicionário em vetor ordenado (4/5)

```
36 void remove_chave(Dicionario *D, int k) {           // fecha o buraco: O(n)
37     int i = posicao(D, k);
38     if (i == D->n || D->v[i] != k) return;
39     for (int j = i; j < D->n - 1; j++) D->v[j] = D->v[j + 1];
40     D->n--;
41 }
42
43 int main(void) {
44     Dicionario D = {NULL, 0, 0};
45     char c;
46     int k;
47     while (scanf(" %c", &c) == 1) {
48         if (c == 'i' && scanf("%d", &k) == 1) insere(&D, k);
49         else if (c == 'r' && scanf("%d", &k) == 1) remove_chave(&D, k);
50         else if (c == 'b' && scanf("%d", &k) == 1)
51             printf(busca(&D, k) ? "%d encontrado\n" : "%d ausente\n", k);
```

Dicionário em vetor ordenado (5/5)

```
52     else if (c == 'm' && D.n) printf("mínimo %d\n", D.v[0]);           // 0(1)
53     else if (c == 'M' && D.n) printf("máximo %d\n", D.v[D.n - 1]);     // 0(1)
54     else if (c == 's' && scanf("%d", &k) == 1) {                       // 0(lg n)
55         int i = posicao(&D, k + 1);
56         if (i < D.n) printf("sucessor de %d: %d\n", k, D.v[i]);
57     } else if (c == 'p') {
58         printf("conjunto:");
59         for (int j = 0; j < D.n; j++) printf(" %d", D.v[j]);
60         printf("\n");
61     }
62 }
63 free(D.v);
64 return 0;
65 }
```



Experimento

- ▶ Ordenado: inserir desloca $\approx n^2/4$; buscar custa $\approx n \lg n$
- ▶ Desordenado: inserir não desloca; buscar custa $n(n+1)/2$
- ▶ n de 1.000 a 16.000 chaves aleatórias

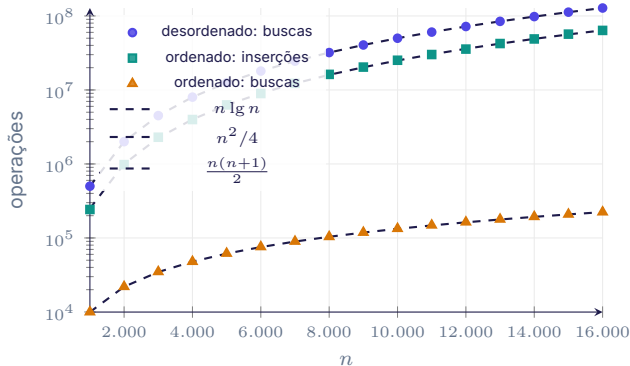
Ordenado ou desordenado?

Hipótese. No vetor ordenado, a i -ésima inserção desloca em média $i/2$ elementos: total $\approx n^2/4$. As n buscas binárias custam $\approx n \lg n$ comparações. No desordenado, inserir é só colocar no fim; mas a busca com sucesso pelo i -ésimo elemento examina i posições: total $n(n+1)/2$.

Método

1. Mesmas n chaves aleatórias nas duas estruturas
2. Contadores de deslocamentos e de comparações
3. n de 1.000 a 16.000
4. Buscas por todas as chaves inseridas

Inserções e buscas



n	medido	razão
1.000	9.978	1,001
4.000	47.906	1,001
7.000	89.810	1,004
10.000	133.618	1,006
13.000	178.618	1,005
16.000	223.618	1,001

razão = medido $\div$ $n \lg n$

✓ desvio máx. 0,6%

Vetor ordenado: buscar é barato ($n \lg n$), inserir é caro ($n^2/4$).
Desordenado: o contrário. Hash e árvores (bloco 3) acabam com esse dilema.

EXPERIMENTO

O que o experimento mostrou

As três previsões se confirmam: $n^2/4$ deslocamentos e $n \lg n$ comparações no vetor ordenado, $n(n+1)/2$ comparações no desordenado. Vetores obrigam a escolher entre inserir rápido e buscar rápido. O resto do curso mostra estruturas que fazem as duas coisas em $O(1)$ ou $O(\lg n)$.

➤ Reproduza: `gcc -O2 aula10_experimento.c -o exp -lm && ./exp`



Conexões

- ▶ Bancos de dados
- ▶ Linguagens
- ▶ Compiladores
- ▶ Redes

Onde isso aparece na computação

Bancos de dados

Uma tabela é um conjunto dinâmico de registros; índices são dicionários sobre uma coluna, e o otimizador escolhe entre varrer e buscar conforme o custo.

Linguagens

`dict` e `set` do Python, `Map` do JavaScript, `std::map` e `std::unordered_map` do C++: o mesmo tipo abstrato com implementações diferentes.

Compiladores

A tabela de símbolos é um dicionário de identificadores; escopos aninhados viram uma pilha de dicionários (aula 11).

Redes

Tabelas ARP, de rotas e de conexões são dicionários consultados milhões de vezes por segundo: a implementação define a latência.

Para casa

Exercícios

1. Escreva $\text{Predecessor}(S, x)$ para o vetor ordenado.
2. (2.3-5) Escreva a busca binária recursiva e prove que é $\Theta(\lg n)$.
3. (2.3-7) Dado um conjunto S de n inteiros e x , decida em $\Theta(n \lg n)$ se há dois elementos com soma x .
4. Implemente o mesmo dicionário em vetor desordenado e compare os dois com 50% de buscas e 50% de inserções.
5. Em que proporção de buscas e inserções o vetor ordenado passa a valer a pena? Meça.

Leitura

Cormen et al., introdução da parte III (conjuntos dinâmicos) e exercício 2.3-5.

Código

josedihego.net/estruturas