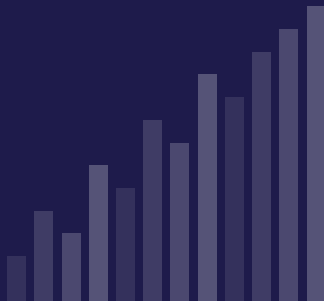

AULA 6 · BLOCO 1: FUNDAMENTOS, ORDENAÇÃO E CUSTO

Recorrências: árvore de recursão e teorema mestre

Seções 4.3 a 4.5

Estruturas de dados e seus algoritmos · José Dihego · IFBA



Roteiro da aula

Ao final, você será capaz de

- ▶ Modelar algoritmos recursivos por recorrências
- ▶ Resolver recorrências pela árvore de recursão e confirmar por substituição
- ▶ Aplicar os três casos do teorema mestre
- ▶ Conferir numericamente as previsões do teorema



Teoria

Substituição, árvore de recursão, teorema mestre



Prática em C

Avaliar $T(n) = aT(n/b) + n^k$ e classificar o caso



Experimento

Quatro recorrências medidas contra suas soluções



Conexões

FFT, multiplicação de matrizes, paralelismo e pilhas de chamada

Três métodos

- ▶ **Substituição:** chute a forma da solução e prove por indução, achando as constantes.
Ex.: $T(n) = 2T(\lfloor n/2 \rfloor) + n$. Chute $T(n) \leq cn \lg n$; então
 $T(n) \leq 2c \frac{n}{2} \lg \frac{n}{2} + n = cn \lg n - cn + n \leq cn \lg n$ para $c \geq 1$.
- ▶ **Árvore de recursão:** desenhe as chamadas, some o custo de cada nível, some os níveis. Ótima para gerar o chute.
- ▶ **Teorema mestre:** resolve de uma vez $T(n) = aT(n/b) + f(n)$, com $a \geq 1$, $b > 1$.

Cuidado na substituição: provar $T(n) \leq cn + n$ **não** prova $T(n) = O(n)$. É preciso a forma exata da hipótese.

Árvore de recursão de $T(n) = 3T(n/4) + cn^2$

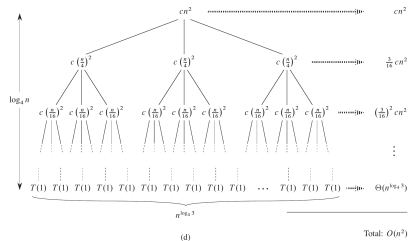
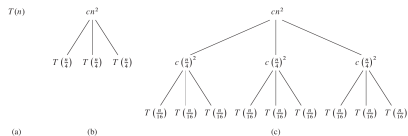


Figura 4.5 de Cormen et al. (2009) — o custo por nível cai em progressão geométrica ($\frac{3}{16}$): a raiz domina e

Árvore desbalanceada: $T(n) = T(n/3) + T(2n/3) + cn$

- ▶ Todo nível completo custa cn .
- ▶ O caminho mais longo, $n \rightarrow \frac{2}{3}n \rightarrow (\frac{2}{3})^2n \rightarrow \dots$, tem $\log_{3/2} n$ níveis.
- ▶ Chute: $O(n \lg n)$, confirmado por substituição.

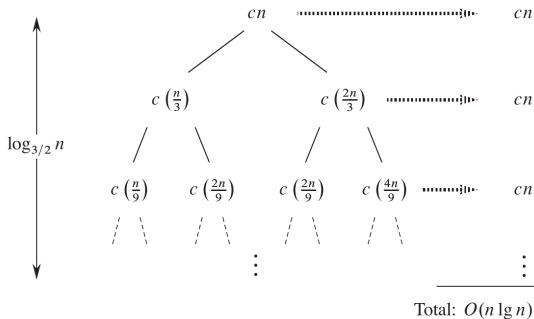


Figura 4.6 de Cormen et al. (2009)

O teorema mestre

Seja $T(n) = aT(n/b) + f(n)$. Compare $f(n)$ com $n^{\log_b a}$ (o custo das folhas):

1. Se $f(n) = O(n^{\log_b a - \varepsilon})$: as **folhas dominam**, $T(n) = \Theta(n^{\log_b a})$.
2. Se $f(n) = \Theta(n^{\log_b a})$: todos os níveis pesam igual, $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. Se $f(n) = \Omega(n^{\log_b a + \varepsilon})$ e $af(n/b) \leq cf(n)$ para algum $c < 1$: a **raiz domina**, $T(n) = \Theta(f(n))$.

Recorrência	$n^{\log_b a}$	Caso	Solução
$2T(n/2) + n$ (merge sort)	n	2	$\Theta(n \lg n)$
$4T(n/2) + n$	n^2	1	$\Theta(n^2)$
$T(n/2) + 1$ (busca binária)	1	2	$\Theta(\lg n)$
$7T(n/2) + n^2$ (Strassen)	$n^{2,81}$	1	$\Theta(n^{\lg 7})$



Prática em C

- ▶ Avalia $T(n) = aT(\lfloor n/b \rfloor) + n^k$ recursivamente
- ▶ Calcula $\log_b a$ e indica o caso do teorema mestre
- ▶ Testes: casos 1, 2 e 3

Calculando recorrências (1/2)

```
1  #include <stdio.h>
2  #include <math.h>
3
4  int a, b;
5  double k;
6
7  /* T(1) = 1; divisão inteira (piso), como em código real. */
8  double T(long n) {
9      if (n <= 1) return 1;
10     return a * T(n / b) + pow(n, k);
11 }
```

Calculando recorrências (2/2)

```
13 int main(void) {
14     long n;
15     if (scanf("%d %d %lf %ld", &a, &b, &k, &n) != 4) return 1;
16     if (a < 1 || b < 2 || n < 1) return 1;
17     double e = log(a) / log(b);           // expoente crítico log_b a
18     printf("T(%ld) = %.0f\n", n, T(n));
19     printf("log_%d %d = %.3f e f(n) = n^%.2f\n", b, a, e, k);
20     if (fabs(k - e) < 1e-9)
21         printf("caso 2: T(n) = O(n^%.3f lg n), previsão %.0f\n", e,
22               pow(n, e) * log2(n));
23     else if (k < e)
24         printf("caso 1: T(n) = O(n^%.3f), previsão %.0f\n", e, pow(n, e));
25     else
26         printf("caso 3: T(n) = O(n^%.2f), previsão %.0f\n", k, pow(n, k));
27     return 0;
28 }
```



Experimento

- ▶ $2T(n/2) + n, 4T(n/2) + n, T(n/2) + 1, 7T(n/2) + n^2$
- ▶ $n = 2^4$ a 2^{24}
- ▶ Soluções exatas e ajuste da constante de Strassen

O teorema mestre confere?

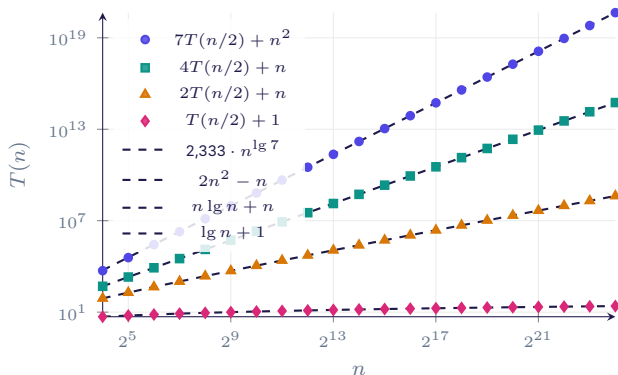
Hipótese. Com $T(1) = 1$ e n potência de 2, as soluções exatas são $2T(n/2) + n = n \lg n + n$, $4T(n/2) + n = 2n^2 - n$ e $T(n/2) + 1 = \lg n + 1$. Para Strassen, o teorema prevê $\Theta(n^{\lg 7})$, com $\lg 7 \approx 2,807$.

Método

1. Recorrência avaliada diretamente (uma chamada por nível)
2. $n = 2^4, \dots, 2^{24}$
3. Três soluções comparadas sem ajuste; Strassen com constante ajustada
4. Gráfico em escala log-log

EXPERIMENTO

Valores exatos e soluções previstas



n	medido
16,0	5.261
256,0	13.363.821
4.096	32.273.967.181
65.536	77.537.778.039.341
1.048.576	186.180.488.678.924.320
16.777.216	447.022.497.921.352.794.112

razão = medido $\div n^{\lg 7}$

✓ desvio máx. 0,0%

Strassen: a razão $T(n)/n^{\lg 7}$ converge para $7/3$. As outras três batem com a solução exata.

EXPERIMENTO

O que o experimento mostrou

As soluções exatas coincidem com os valores calculados, e a de Strassen converge para $\frac{7}{3} n^{\lg 7}$. O teorema mestre acerta a **ordem de crescimento**; para a constante, é preciso resolver a recorrência com mais cuidado (substituição ou árvore completa).

Desafio: inclua $3T(n/4) + n^2$ (caso 3) e $2T(n/2) + n \lg n$, que cai na lacuna entre os casos 2 e 3.

➤ Reproduza: `gcc -O2 aula06_experimento.c -o exp -lm && ./exp`



Conexões

- ▶ Processamento de sinais
- ▶ Álgebra linear e IA
- ▶ Computação paralela
- ▶ Linguagens e sistemas

Onde isso aparece na computação

Processamento de sinais

A FFT satisfaz $T(n) = 2T(n/2) + \Theta(n)$: $n \lg n$ em vez de n^2 viabiliza áudio digital, 4G/5G, ressonância magnética e compressão JPEG/MP3.

Álgebra linear e IA

Strassen ($n^{2,81}$) e sucessores mostram que multiplicar matrizes, o núcleo das redes neurais, pode custar menos que n^3 .

Computação paralela

Algoritmos paralelos são analisados por duas recorrências: trabalho total e *span* (caminho crítico). O paralelismo é a razão entre elas.

Linguagens e sistemas

A profundidade da recursão é a altura da árvore: com $\lg n$ níveis, a pilha de chamadas fica pequena; com n níveis, ocorre estouro de pilha.

Para casa

Exercícios

1. (4.3-1) Mostre por substituição que $T(n) = T(n-1) + n$ é $O(n^2)$.
2. (4.4-1) Use a árvore de recursão para estimar $T(n) = 3T(\lfloor n/2 \rfloor) + n$.
3. (4.5-1) Aplique o teorema mestre a $2T(n/4) + 1$, $2T(n/4) + \sqrt{n}$, $2T(n/4) + n$, $2T(n/4) + n^2$.
4. (4.5-3) Mostre que a busca binária, $T(n) = T(n/2) + \Theta(1)$, é $\Theta(\lg n)$.
5. Use `aula06_recorrencias.c` com b não potência de 2 ($b = 3$) e compare com a previsão.

Leitura

Cormen et al., seções 4.3, 4.4 e 4.5.

Código

josedihego.net/estruturas