MC448 — Análise de Algoritmos I

Cid Carvalho de Souza   Cândida Nunes da Silva
Orlando Lee

15 de setembro de 2009

Recorrências

Resolução de Recorrências

- Relações de recorrência expressam a complexidade de algoritmos recursivos como, por exemplo, os algoritmos de divisão e conquista.
- É preciso saber resolver as recorrências para que possamos efetivamente determinar a complexidade dos algoritmos recursivos.

Mergesort

Vamos começar revendo um famoso algoritmo de ordenação, o Mergesort.

O Mergesort usa a subrotina *Intercala*(A, p, q, r) que tem

Entrada: um vetor $A[p \dots r]$ e um índice q tais que $A[p \dots q]$ e $A[q+1 \dots r]$ estão em **ordem crescente** e

Saída: o vetor $A[p \dots r]$ em **ordem crescente**.

Intercala

Exemplo:

Entrada:

	p				q				r
A	22	33	55	77	99	11	44	66	88

Saída:

	p				q				r
A	11	22	33	44	55	66	77	88	99

Exercício. Escreva uma rotina INTERCALA com complexidade de tempo **linear**.

Mergesort

O Mergesort é um algoritmo que usa o método de divisão-e-conquista.

A idéia é muito simples:

- divida o vetor em dois subvetores de tamanho quase iguais (divisão),
- ordene cada subvetor recursivamente, e
- use INTERCALA para ordenar o vetor (conquista).

Corretude do Mergesort

```
MERGESORT(A, p, r)
se p < r
    então q ← ⌊(p + r)/2⌋
        MERGESORT(A, p, q)
        MERGESORT(A, q + 1, r)
        INTERCALA(A, p, q, r)
```

O algoritmo está correto?

A corretude do algoritmo Mergesort apoia-se na corretude do algoritmo Intercala e pode ser demonstrada **por indução** em $n := r - p + 1$ (Exercício).

Mergesort

```
MERGESORT(A, p, r)
se p < r
    então q ← ⌊(p + r)/2⌋
        MERGESORT(A, p, q)
        MERGESORT(A, q + 1, r)
        INTERCALA(A, p, q, r)
```

Qual é a complexidade de MERGESORT?

Seja $T(n) :=$ o consumo de tempo máximo (pior caso) em função de $n = r - p + 1$

Complexidade do Mergesort

```
MERGESORT(A, p, r)
1   se p < r
2       então q ← ⌊(p + r)/2⌋
3             MERGESORT(A, p, q)
4             MERGESORT(A, q + 1, r)
5             INTERCALA(A, p, q, r)
```

linha	consumo de tempo
1	?
2	?
3	?
4	?
5	?

$T(n) = ?$

Complexidade do Mergesort

```
MERGESORT(A, p, r)
1   se p < r
2       então q ← ⌊(p + r)/2⌋
3             MERGESORT(A, p, q)
4             MERGESORT(A, q + 1, r)
5             INTERCALA(A, p, q, r)
```

linha	consumo de tempo
1	b_0
2	b_1
3	$T(\lceil n/2 \rceil)$
4	$T(\lfloor n/2 \rfloor)$
5	an

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + an + (b_0 + b_1)$$

Resolução de recorrências

- Queremos resolver a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + an + b \qquad \text{para } n \geq 2. \end{aligned}$$

- Resolver uma recorrência significa encontrar uma fórmula fechada para $T(n)$.
- Não é necessário achar uma solução exata. Basta encontrar uma função $f(n)$ tal que $T(n) \in \Theta(f(n))$.

Resolução de recorrências

Alguns métodos para resolução de recorrências:

- substituição
- iteração
- árvore de recorrência

Veremos também um resultado bem geral que permite resolver várias recorrências: Teorema Master.

Método da substituição

- Idéia básica: "adivinhe" qual é a solução e prove por indução que ela funciona!
- Método poderoso mas nem sempre aplicável (obviamente).
- Com prática e experiência fica mais fácil de usar!

Exemplo

Considere a recorrência:

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n \qquad \text{para } n \geq 2. \end{aligned}$$

Chuto que $T(n) \in O(n \lg n)$.

Mais precisamente, chuto que $T(n) \leq 3n \lg n$.

(Lembre que $\lg n = \log_2 n$.)

Exemplo

$$\begin{aligned} T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n \\ &\leq 3\left\lceil \frac{n}{2} \right\rceil \lg \left\lceil \frac{n}{2} \right\rceil + 3\left\lfloor \frac{n}{2} \right\rfloor \lg \left\lfloor \frac{n}{2} \right\rfloor + n \\ &\leq 3\left\lceil \frac{n}{2} \right\rceil \lg n + 3\left\lfloor \frac{n}{2} \right\rfloor (\lg n - 1) + n \\ &= 3\left(\left\lceil \frac{n}{2} \right\rceil + \left\lfloor \frac{n}{2} \right\rfloor\right) \lg n - 3\left\lfloor \frac{n}{2} \right\rfloor + n \\ &= 3n \lg n - 3\left\lfloor \frac{n}{2} \right\rfloor + n \\ &\leq 3n \lg n. \end{aligned}$$

(Yeeeeeeesssss!)

Exemplo

- Mas espere um pouco!
- $T(1) = 1$ e $3.1.\lg 1 = 0$ e a base da indução não funciona!
- Certo, mas lembre-se da definição da classe O.

  Só preciso provar que $T(n) \leq 3n \lg n$ para $n \geq n_0$ onde n_0 é alguma constante.

  Vamos tentar com $n_0 = 2$. Nesse caso, é preciso verificar que vale a "base" para $n = 2, 3$ (**por quê?**):

$$\begin{aligned} T(2) = T(1) + T(1) + 2 = 4 &\leq 3.2.\lg 2 = 6, \\ T(3) = T(2) + T(1) + 3 = 8 &\leq 3.3 \lg 3. \end{aligned}$$

Exemplo

- Certo, funcionou para $T(1) = 1$.
- Mas e se por exemplo $T(1) = 8$?

  Então $T(2) = 8 + 8 + 2 = 18$ e $3.2.\lg 2 = 6$.
  Não deu certo...
- Certo, mas aí basta escolher uma constante maior.
  Mostra-se do mesmo jeito que $T(n) \leq 10n \lg n$.
  Para esta escolha temos $T(2) = 18 \leq 10.2.\lg 2 = 20$ e $T(3) = 18 + 8 + 3 = 29 \leq 10.3.\lg 3$.
- De modo geral, se o passo de indução funciona ($T(n) \leq cn \lg n$), é sempre possível escolher c e n_0 de modo conveniente!

Como achar as constantes?

- Tudo bem. Dá até para chutar que $T(n)$ pertence a classe $O(n \lg n)$.
- Mas como descobrir que $T(n) \leq 3n \lg n$? Como achar a constante 3?
- Eis um método simples: suponha como hipótese de indução que $T(n) \leq cn \lg n$ para $n \geq n_0$ onde c e n_0 são constantes que vou tentar determinar.

Primeira tentativa

$$\begin{aligned}
T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n \\
&\leq c \left\lceil \frac{n}{2} \right\rceil \lg \left\lceil \frac{n}{2} \right\rceil + c \left\lfloor \frac{n}{2} \right\rfloor \lg \left\lfloor \frac{n}{2} \right\rfloor + n \\
&\leq c \left\lceil \frac{n}{2} \right\rceil \lg n + c \left\lfloor \frac{n}{2} \right\rfloor \lg n + n \\
&= c \left(\left\lceil \frac{n}{2} \right\rceil + \left\lfloor \frac{n}{2} \right\rfloor \right) \lg n + n \\
&= cn \lg n + n
\end{aligned}$$

(Hummm, não deu certo...)

Segunda tentativa

$$\begin{aligned}
T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n \\
&\leq c \left\lceil \frac{n}{2} \right\rceil \lg \left\lceil \frac{n}{2} \right\rceil + c \left\lfloor \frac{n}{2} \right\rfloor \lg \left\lfloor \frac{n}{2} \right\rfloor + n \\
&\leq c \left\lceil \frac{n}{2} \right\rceil \lg n + c \left\lfloor \frac{n}{2} \right\rfloor (\lg n - 1) + n \\
&= c \left(\left\lceil \frac{n}{2} \right\rceil + \left\lfloor \frac{n}{2} \right\rfloor \right) \lg n - c \left\lfloor \frac{n}{2} \right\rfloor + n \\
&= cn \lg n - c \left\lfloor \frac{n}{2} \right\rfloor + n \\
&\leq cn \lg n.
\end{aligned}$$

Para garantir a última desigualdade basta que $-c\lfloor n/2 \rfloor + n \leq 0$. Ela vale para $c \geq 3$ e $n \geq 2$.
(Yeeeeeeessssss!)

Segunda tentativa

Note que a escolha da constante c deve levar em conta os valores da "base". O que é a base depende do valor n_0 escolhido.

Como foi dito antes, em geral é suficiente escolher c grande o suficiente para garantir isto.

Completando o exemplo

Mostramos que a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n \quad \text{para } n \geq 2. \end{aligned}$$

satisfaz $T(n) \in O(n \lg n)$.

Mas quem garante que $T(n)$ não é "menor"?

O melhor é mostrar que $T(n) \in \Theta(n \lg n)$.

Resta então mostrar que $T(n) \in \Omega(n \lg n)$. A prova é similar. (Exercício!)

Como chutar?

Não há nenhuma receita genérica para adivinhar soluções de recorrências. A experiência é o fator mais importante.

Felizmente, há várias idéias que podem ajudar.

Considere a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 2T(\lfloor n/2 \rfloor) + n \quad \text{para } n \geq 2. \end{aligned}$$

Ela é quase idêntica à anterior e podemos chutar que $T(n) \in \Theta(n \lg n)$. Isto de fato é verdade. (Exercício ou consulte o CLRS)

Como chutar?

Considere agora a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 2T(\lfloor n/2 \rfloor + 17) + n \quad \text{para } n \geq 2. \end{aligned}$$

Ela parece bem mais difícil por causa do "17" no lado direito.

Intuitivamente, porém, isto não deveria afetar a solução. Para n **grande** a diferença entre $T(\lfloor n/2 \rfloor)$ e $T(\lfloor n/2 \rfloor + 17)$ não é tanta.

Chuto então que $T(n) \in \Theta(n \lg n)$. (Exercício!)

Truques e sutilezas

Algumas vezes adivinhamos corretamente a solução de uma recorrência, mas as contas aparentemente não funcionam! Em geral, o que é necessário é fortalecer a hipótese de indução.

Considere a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + 1 \quad \text{para } n \geq 2. \end{aligned}$$

Chutamos que $T(n) \in O(n)$ e tentamos mostrar que $T(n) \leq cn$ para alguma constante c.

$$\begin{aligned} T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + 1 \\ &\leq c\lceil n/2 \rceil + c\lfloor n/2 \rfloor + 1 \\ &= cn + 1. \end{aligned}$$

(Humm, falhou. . .)

E agora? Será que erramos o chute? Será que $T(n) \in \Theta(n^2)$?

Truques e sutilezas

Na verdade, adivinhamos corretamente. Para provar isso, é preciso usar uma hipótese de indução mais forte.

Vamos mostrar que $T(n) \leq cn - b$ onde $b > 0$ é uma constante.

$$\begin{aligned} T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + 1 \\ &\leq c\lceil n/2 \rceil - b + c\lfloor n/2 \rfloor - b + 1 \\ &= cn - 2b + 1 \\ &\leq cn - b \end{aligned}$$

onde a última desigualdade vale se $b \geq 1$. (Yeeeessss!)
Para que a recorrência valha, basta escolher c suficientemente grande para satisfazer os valores na “base”.

Método da iteração

- Não é necessário adivinhar a resposta!
- Precisa fazer mais contas. . .
- Idéia: expandir (iterar) a recorrência e escrevê-la como uma somatória de termos que dependem apenas de n e das condições iniciais.
- Precisa conhecer limitantes para várias somatórias.

Algumas somatórias

- Soma dos termos de uma P.A. de razão r com n termos:

$$a + (a+r) + (a+2r) + \cdots + (a+(n-1)r) = an + \frac{r(n-1)n}{2}.$$

- Soma dos termos de uma P.G. de razão q com n termos:

$$a + aq + aq^2 + \cdots + aq^{n-1} = \frac{a(1-q^n)}{1-q}.$$

- Soma dos termos de uma P.G. infinita de razão q ($0 < q < 1$):

$$a + aq + aq^2 + \cdots + aq^{n-1} + aq^n + \cdots = \frac{a}{1-q}.$$

Método da iteração

Considere a recorrência

$$\begin{aligned} T(n) &= b && \text{para } n \le 3, \\ T(n) &= 3T(\lfloor n/4 \rfloor) + n && \text{para } n \ge 4. \end{aligned}$$

Iterando a recorrência obtemos

$$\begin{aligned} T(n) &= n + 3T(\lfloor n/4 \rfloor) \\ &= n + 3(\lfloor n/4 \rfloor + 3T(\lfloor n/16 \rfloor)) \\ &= n + 3(\lfloor n/4 \rfloor + 3(\lfloor n/16 \rfloor + 3T(\lfloor n/64 \rfloor))) \\ &= n + 3\lfloor n/4 \rfloor + 9\lfloor n/16 \rfloor + 27T(\lfloor n/64 \rfloor). \end{aligned}$$

Certo, mas quando devo parar?
O i-ésimo termo da série é $3^i \lfloor n/4^i \rfloor$. Ela termina no j-ésimo termo em que $\lfloor n/4^j \rfloor \le 3$. Note que $j \le \log_4 n$.

Método da iteração

$$\begin{aligned} T(n) &\le n + 3\lfloor n/4 \rfloor + 9\lfloor n/16 \rfloor + 27T\lfloor n/64 \rfloor + \cdots + 3^j b \\ T(n) &\le n + 3n/4 + 9n/16 + 27n/64 + \cdots + 3^{\log_4 n} b \\ &\le n.(1 + 3/4 + 9/16 + 27/64 + \cdots) + bn^{\log_4 3} \\ &\le n \sum_{i=0}^{\infty} \left(\frac{3}{4}\right)^i + bn^{\log_4 3} \\ &= 4n + bn^{\log_4 3} \end{aligned}$$

pois $j \le \log_4 n$ e $\sum_{i=0}^{\infty} q^i = \frac{1}{1-q}$ para $0 < q < 1$.

Como $\log_4 3 < 1$ segue que $n^{\log_4 3} \in o(n)$ e logo, $T(n) \in O(n)$.

Método de iteração

- As contas ficam mais simples se supormos que a recorrência está definida apenas para potências de um número, por exemplo, $n = 4^i$.
- Note, entretanto, que a recorrência deve ser provada para todo natural suficientemente grande.
- Muitas vezes, é possível depois de iterar a recorrência, adivinhar a solução e usar o método da substituição!

Método de iteração

$$\begin{aligned} T(n) &= b && \text{para } n \le 3, \\ T(n) &= 3T(\lfloor n/4 \rfloor) + n && \text{para } n \ge 4. \end{aligned}$$

Chuto que $T(n) \le cn$.

$$\begin{aligned} T(n) &= 3T(\lfloor n/4 \rfloor) + n \\ &\le 3c\lfloor n/4 \rfloor + n \\ &\le 3c(n/4) + n \\ &\le cn \end{aligned}$$

onde a última desigualdade vale se $c \ge 4$.
(Yeeessss!)

Árvore de recorrência

- Permite visualizar melhor o que acontece quando a recorrência é iterada.
- É mais fácil organizar as contas.
- Útil para recorrências de algoritmos de divisão-e-conquista.

Árvore de recorrência

Considere a recorrência

$$\begin{array}{lcll} T(n) & = & \Theta(1) & \text{para } n = 1, 2, 3, \\ T(n) & = & 3T(\lfloor n/4 \rfloor) + cn^2 & \text{para } n \geq 4, \end{array}$$

onde $c > 0$ é uma constante.

Costuma-se usar a notação $T(n) = \Theta(1)$ para indicar que $T(n)$ é uma constante. (CLRS)

Árvore de recorrência

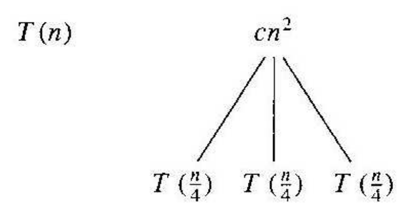
Simplificação

Vamos supor que a recorrência está definida apenas para potências de 4

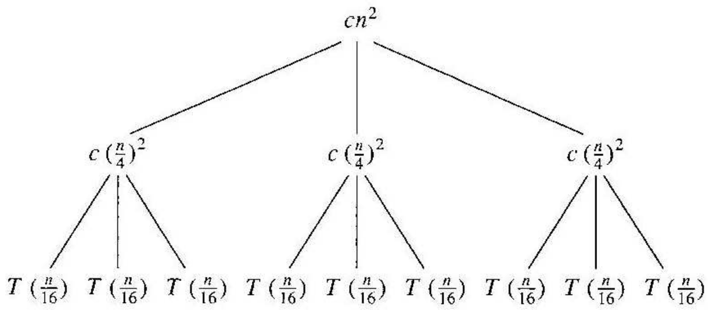
$$\begin{array}{lcll} T(n) & = & \Theta(1) & \text{para } n = 1, \\ T(n) & = & 3T(n/4) + cn^2 & \text{para } n = 4, 16, \ldots, 4^i, \ldots. \end{array}$$

Isto permite descobrir mais facilmente a solução. Depois usamos o método da substituição para formalizar.

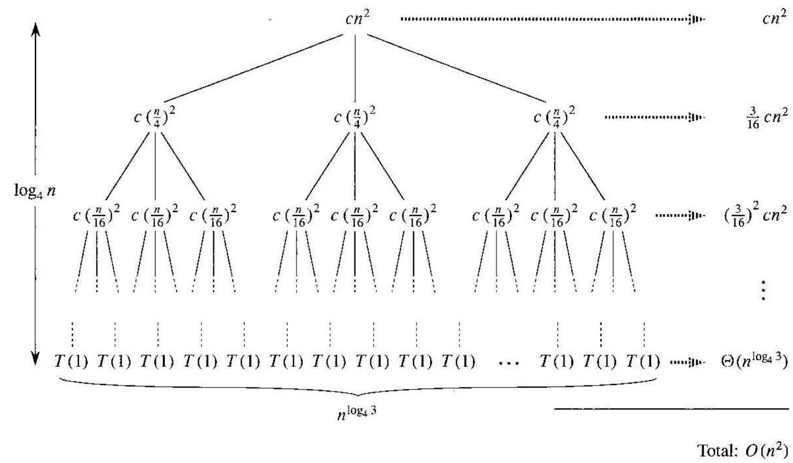
Árvore de recorrência



Árvore de recorrência



Árvore de recorrência



Árvore de recorrência

- O número de níveis é $\log_4 n + 1$.
- No nível i o tempo gasto (sem contar as chamadas recursivas) é $(3/16)^i cn^2$.
- No último nível há $3^{\log_4 n} = n^{\log_4 3}$ folhas. Como $T(1) = \Theta(1)$ o tempo gasto é $\Theta(n^{\log_4 3})$.

  Em outras palavras, o tempo gasto é uma função que pertence a $\Theta(n^{\log_4 3})$.

Árvore de recorrência

Logo,

$$\begin{aligned}
T(n) &= cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \left(\frac{3}{16}\right)^3 cn^2 + \cdots + \\
&\quad + \left(\frac{3}{16}\right)^{\log_4 n - 1} cn^2 + \Theta(n^{\log_4 3}) \\
&= cn^2 \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i + \Theta(n^{\log_4 3}) \\
&\le cn^2 \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i + \Theta(n^{\log_4 3}) = \frac{16}{13}cn^2 + \Theta(n^{\log_4 3}),
\end{aligned}$$

e $T(n) \in O(n^2)$.

Árvore de recorrência

Mas $T(n) \in O(n^2)$ é realmente a solução da recorrência original?

Com base na árvore de recorrência, chutamos que $T(n) \leq dn^2$ para alguma constante $d > 0$.

$$\begin{aligned} T(n) &= 3T(\lfloor n/4 \rfloor) + cn^2 \\ &\leq 3d\lfloor n/4 \rfloor^2 + cn^2 \\ &\leq 3d(n/4)^2 + cn^2 \\ &= \frac{3}{16}dn^2 + cn^2 \\ &\leq dn^2 \end{aligned}$$

onde a última desigualdade vale se $d \geq (16/13)c$. (Yeeesssss!)

Árvore de recorrência

Resumo

- O número de nós em cada nível da árvore é o número de chamadas recursivas.
- Em cada nó indicamos o "tempo" ou "trabalho" gasto naquele nó que **não** corresponde a chamadas recursivas.
- Na coluna mais à direita indicamos o tempo total naquele nível que **não** corresponde a chamadas recursivas.
- Somando ao longo da coluna determina-se a solução da recorrência.

Vamos tentar juntos?

Eis um exemplo um pouco mais complicado.

Vamos resolver a recorrência

$$\begin{array}{lcll} T(n) & = & 1 & \text{para } n = 1, 2, \\ T(n) & = & T(\lceil n/3 \rceil) + T(\lfloor 2n/3 \rfloor) + n & \text{para } n \geq 3. \end{array}$$

Qual é a solução da recorrência?

Resposta: $T(n) \in O(n \lg n)$. (Exercício)

Recorrências com O à direita (CLRS)

Uma "recorrência"

$$\begin{array}{ll} T(n) = \Theta(1) & \text{para } n = 1, 2, \\ T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2) & \text{para } n \geq 3 \end{array}$$

representa todas as recorrências da forma

$$\begin{array}{ll} T(n) = a & \text{para } n = 1, 2, \\ T(n) = 3T(\lfloor n/4 \rfloor) + bn^2 & \text{para } n \geq 3 \end{array}$$

onde a e $b > 0$ são constantes.

As soluções exatas dependem dos valores de a e b, mas estão todas na mesma classe Θ.

A "solução" é $T(n) = \Theta(n^2)$, ou seja, $T(n) \in \Theta(n^2)$.

As mesmas observações valem para as classes O, Ω, o, ω.

Algumas convenções

- É comum usar a notação assintótica para as classes $O, \Omega, \Theta, o, \omega$ com outro significado.
- Dizemos que um algoritmo tem complexidade de tempo $O(f(n))$ se a função que expressa a complexidade de tempo do algoritmo pertence à classe $O(f(n))$.

  Expressões similares valem para $\Omega, \Theta, o, \omega$.
- Dizemos que um certo objeto (conjunto, seqüência etc) tem/contém $O(f(n))$ elementos se a função que expressa esta quantidade pertence à classe $O(f(n))$.

Recorrência do Mergesort

Podemos escrever a recorrência de tempo do Mergesort da seguinte forma

$$
\begin{aligned}
T(1) &= \Theta(1) \\
T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) \qquad \text{para } n \geq 2.
\end{aligned}
$$

A solução da recorrência é $T(n) = \Theta(n \lg n)$.

A prova é essencialmente a mesma do primeiro exemplo. (Exercício!)

Cuidados com a notação assintótica

A notação assintótica é muito versátil e expressiva. Entretanto, deve-se tomar alguns cuidados.

Considere a recorrência

$$
\begin{aligned}
T(1) &= 1 \\
T(n) &= 2T(\lfloor n/2 \rfloor) + n \qquad \text{para } n \geq 2.
\end{aligned}
$$

É similar a recorrência do Mergesort!

Mas eu vou “provar” que $T(n) = O(n)$!

Cuidados com a notação assintótica

Vou mostrar que $T(n) \leq cn$ para alguma constante $c > 0$.

$$
\begin{aligned}
T(n) &= 2T(\lfloor n/2 \rfloor) + n \\
&\leq 2c\lfloor n/2 \rfloor + n \\
&\leq cn + n \\
&= O(n) \qquad \Longleftarrow \textbf{ERRADO!!!}
\end{aligned}
$$

Por quê?

Não foi feito o passo indutivo, ou seja, não foi mostrado que $T(n) \leq cn$.

Teorema Master

- Veremos agora um resultado que descreve soluções para recorrências da forma

$$T(n) = aT(n/b) + f(n),$$

onde $a \geq 1$ e $b > 1$ são constantes.

- O caso base é omitido na definição e convenciona-se que é uma constante para valores pequenos.
- A expressão n/b pode indicar tanto $\lfloor n/b \rfloor$ quanto $\lceil n/b \rceil$.
- O Teorema Master **não** fornece a resposta para todas as recorrências da forma acima.

Teorema Master

Teorema (Teorema Master (CLRS))

Sejam $a \geq 1$ e $b > 1$ constantes, seja $f(n)$ uma função e seja $T(n)$ definida para os inteiros não-negativos pela relação de recorrência

$$T(n) = aT(n/b) + f(n).$$

Então $T(n)$ pode ser limitada assintoticamente da seguinte maneira:

1. *Se $f(n) \in O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) \in \Theta(n^{\log_b a})$*
2. *Se $f(n) \in \Theta(n^{\log_b a})$, então $T(n) \in \Theta(n^{\log_b a} \log n)$*
3. *Se $f(n) \in \Omega(n^{\log_b a + \epsilon})$, para alguma constante $\epsilon > 0$ e se $af(n/b) \leq cf(n)$, para alguma constante $c < 1$ e para n suficientemente grande, então $T(n) \in \Theta(f(n))$*

Exemplos onde o TM pode ser usado

- $T(n) = 9T(n/3) + n$
  Solução: $T(n) = \Theta(n^2)$ (Caso 1 do TM)
- $T(n) = T(2n/3) + 1$
  Solução: $T(n) = \Theta(\log n)$ (Caso 2 do TM)
- $T(n) = T(3n/4) + n \log n$
  Solução: $T(n) = \Theta(n \log n)$ (Caso 3 do TM)
- $T(n) = 4T(n/2) + n \log n$
  Solução: $T(n) = \Theta(n^2)$ (Caso 1 do TM)
- $T(n) = 2T(n/2) + (n + \log n)$
  Solução: $T(n) = n \log n$ (Caso 2 do TM)

Exemplos onde o TM **não pode ser usado**

- $T(n) = T(n-1) + n$
- $T(n) = T(n-a) + T(a) + n$, ($a \geq 1$ inteiro)
- $T(n) = T(\alpha n) + T((1-\alpha)n) + n$, ($0 < \alpha < 1$)
- $T(n) = T(n-1) + \log n$
- $T(n) = 2T(\frac{n}{2}) + n \log n$ (por quê?)