Recorrências

Universidade Federal do Amazonas
Departamento de Eletrônica e Computação

Recorrências

- A expressão:

$$T(n) = \begin{cases} c & n = 1 \\ 2T\left(\frac{n}{2}\right) + cn & n > 1 \end{cases}$$

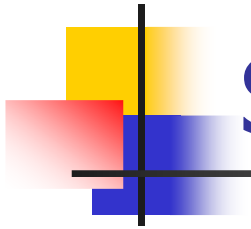
é uma *recorrência*.

  - Recorrência: uma equação que descreve uma função em termos de seus valores para instâncias menores
  - Com frequência omitimos pisos, tetos e condições limite

Exemplos de Recorrências

$$s(n) = \begin{cases} 0 & n = 0 \\ c + s(n-1) & n > 0 \end{cases} \qquad s(n) = \begin{cases} 0 & n = 0 \\ n + s(n-1) & n > 0 \end{cases}$$

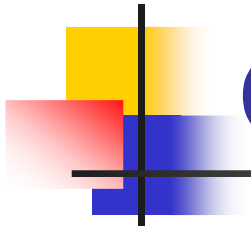
$$T(n) = \begin{cases} c & n = 1 \\ 2T\left(\frac{n}{2}\right) + c & n > 1 \end{cases} \qquad T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$



Solução de Recorrências

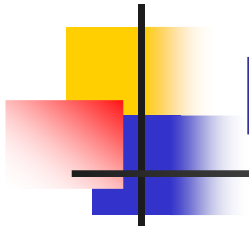
- Métodos
  - Subsitituição
  - Iteração
  - Mestre
  - Aniquilador

Método da Substituição



Conceitos básicos

- Dois passos
  - Pressupor a forma da solução
  - Usar indução matemática para determinar se a solução se aplica
- Substituição da resposta pressuposta para a função quando a **hipótese indutiva** é aplicada
- Útil quando é fácil estimar a **forma da solução**
- Pode ser utilizado para estabelecer limites superiores ou inferiores das recorrências



Exemplo – Substituição (1)

- Resolver a recorrência (que é semelhante à recorrência da ordenação por intercalação)

$$T(n) = 2T(\lfloor n/2 \rfloor) + n$$

T(n) = O(n lg n) ?

Exemplo – Substituição (2)

- Determinar um limite superior para

$$T(n) = 2T(\lfloor n/2 \rfloor) + n$$

- Se $T(n) = O(n \lg n)$
- Provar que $T(n) \leq cn \lg n$ para algum $c>0$ usando indução e para todo $n \geq n_0$

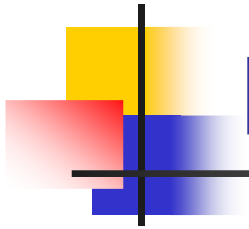
Exemplo – Substituição (3)

- A indução exige que mostremos que a solução se mantém válida para as condições limite
- **Base da indução**: mostrar que a inequação é válida para algum *n* suficientemente pequeno
  - Se n= 1 → T(1) ≤ c * 1 * log 1 = 0 !!!
  - No entanto, T(n) = 2T(⌊n/2⌋) + n ∴ T(1) = 1
  - Mas

n=2 → T(2) =2T(1) + 2 = 4 e *cn* lg *n* = c * 2 * lg 2 = 2c

n=3 → T(3)=2T(1) + 3 = 5 e *cn* lg *n* = c * 3 * lg 3

Exemplo – Substituição (4)



- Pode-se partir de T(2)=4 ou T(3)=5 usando qualquer c ≥ 2, pois $T(n) \leq cn \lg n$
- Válido de acordo com a notação assintótica:
  - $T(n) \leq cn \lg n$ *para* $n \geq n_0$
- Truque: estender as condições de contorno para fazer a hipótose indutiva valer para pequenos valores de *n*

Exemplo – Substituição (5)

- **Hipótese indutiva:**
  - Assumir a inequação para $\lfloor n/2 \rfloor$

$$T(\lfloor n/2 \rfloor) \leq c\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)$$

Exemplo – Substituição (5)

- Indução : A inequação é válida para *n*

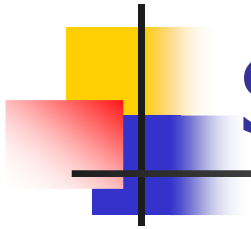
$$
\begin{aligned}
T(n) &= 2T(\lfloor n/2 \rfloor) + n \\
&\le 2(c\lfloor n/2 \rfloor \lg \lfloor n/2 \rfloor)) + n \\
&\le cn \lg(n/2) + n \\
&= cn(\lg n - \lg 2) + n \\
&= cn \lg n - cn + n \\
&\le cn \lg n
\end{aligned}
$$

$T(\lfloor n/2 \rfloor) \le c\lfloor n/2 \rfloor \lg \lfloor n/2 \rfloor)$

(é valida para c ≥1, analisando o limite superior)

Sobre Estimativas (1)

- Requer experiência e criatividade
- Pode-se utilizar **árvores de recursão**
- Se a recursão é similar a uma outra com a qual se está familiarizado, é razoável tentar uma solução similar $T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$
  - T(n) = O(*n* lg *n*) ➔ **Por quê?**
  - O termo adicional (17) não afetará substancialmente a solução da recorrência para um *n* "grande"



Sobre Estimativas (2)

- As vezes a estimativa está correta mas as contas "não fecham" na hora da indução
  - A hipótese indutiva não é suficientemente forte para provar o limite!!!
  - Revisar a estimativa subtraindo um termo de menor ordem assintótica para que a matemática funcione

Sobre Estimativas (3)

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$$ é O(n) ???

Mostrar que $T(n) \leq cn$

$$T(n) \leq c\lfloor n/2 \rfloor + c\lceil n/2 \rceil + 1 = cn + 1$$ não implica $T(n) \leq cn$

Nova tentativa $T(n) \leq cn-b$

$$T(n) \leq (c\lfloor n/2 \rfloor - b) + (c\lceil n/2 \rceil - b) + 1 = cn - 2b + 1$$

$$T(n) \leq cn - b$$ (para todo $b \geq 1$)

A constante c deve ser escolhida com um valor grande o suficiente para tratar as condições limites

Troca de Variáveis

$T(n) = 2T(\sqrt{n}) + \lg n$ $\quad$ $m = \lg n \therefore n = 2^m$

$T(2^m) = 2T(2^{m/2}) + m$ $\quad$ $S(m)=T(2^m)$

$S(m) = 2S(m/2) + m$ semelhante a $T(n) = 2T(n/2) + n$

$T(n) = T(2^m) = S(m)$ $\quad$ S(m)=O(m lg m)

$= O(m \lg m) = O(\lg n \lg \lg n)$

Note que log 2^m = m e log $2^{m/2}$ = m/2

Exercício 1

- Qual seria a estimativa para T(n)=T(n-1)+1?
  - Dica: mostre que a solução T(n)=T(n-1)+1 é O(*n*)

→ Caso base: *n=1, T(1)=T(0)+1=1* e *cn=c*1* (válido para c≥1)

→ Provar que $T(n) \leq cn$ para algum $c>0$ usando indução e para todo $n \geq n_0$

$$T(n) = T(n-1)+1$$

$$T(n) \leq c(n-1)+1 = cn-c+1 \leq cn$$

$$T(n) \leq cn \qquad \text{(é valido para c } \geq 1\text{)}$$

Exercício 2

- Qual seria a estimativa para $T(n)=T(\lceil n/2 \rceil)+1$?
  - Mostre que a solução $T(n)=T(\lceil n/2 \rceil)+1$ é $O(lg\ n)$

→ Caso base: *n=2, T(2)=T(1)+1=2* e *c*lgn=c*1; n=4, T(3)=T(2)+1=3 e c*lgn=c*2* (válido para $c \geq 2$)

→ Provar que $T(n) \leq c\ lgn$ para algum $c>0$ usando indução e para todo $n \geq n_0$

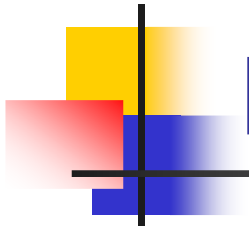
$$T(n) = T(\lceil n/2 \rceil) + 1$$

$$T(n) \leq c \lg n/2 + 1 = c \lg n - c \lg 2 + 1$$

$$T(n) \leq c \lg n - c + 1 \leq c \lg n$$

$$T(n) \leq c \lg n$$ (é valido para c ≥1)

Método da Iteração



Método da Iteração

- Consiste em:
  - Expandir a recorrência
  - Usar proprieadades algébricas para encontrar um somatório
  - Resolver o somatório

Exemplo 1: Iteração (1)

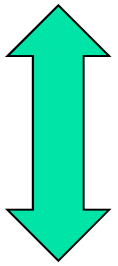
- Resolver

$$s(n) = \begin{cases} 0 & n = 0 \\ c + s(n-1) & n > 0 \end{cases}$$

Exemplo 1: Iteração (2)

- s(n) = c + s(n-1)
  = c + c + s(n-2)
  = c + c + c + s(n-3)
  = 3c + s(n-3)
  …
  = kc + s(n-k) = ck + s(n-k)

s(n) = c + s(n-1)
s(n-1) = c + s(n-2)
s(n-2) = c + s(n-3)



$$s(n) = \begin{cases} 0 & n = 0 \\ c + s(n-1) & n > 0 \end{cases}$$

Exemplo 1: Iteração (3)

- Até agora para $n > k$ temos
  - $s(n) = ck + s(n-k)$
- E se k=n ?
  - $s(n) = cn + s(0) = cn$
- Portanto
  - $S(n) = cn$

$$s(n) = \begin{cases} 0 & n = 0 \\ c + s(n-1) & n > 0 \end{cases}$$

Exemplo 2: Iteração (1)

- Resolver

$$s(n) = \begin{cases} 0 & n = 0 \\ n + s(n-1) & n > 0 \end{cases}$$

Exemplo 2: Iteração (2)

- s(n)

$$s(n) = \begin{cases} 0 & n = 0 \\ n + s(n-1) & n > 0 \end{cases}$$

k=1 = n + s(n-1)

k=2 = n + n-1 + s(n-2)

k=3 = n + n-1 + n-2 + s(n-3)

k=4 = n + n-1 + n-2 + n-3 + s(n-4)

= ...

= n + n-1 + n-2 + n-3 + ... + n-(k-1) + s(n-k)

$$= \sum_{i=n-k+1}^{n} i \quad + \quad s(n-k)$$

Exemplo 2: Iteração (3)

- Até agora para $n > k$ temos

$$\sum_{i=n-k+1}^{n} i \quad + \quad s(n-k)$$

- E se k=n ?

$$\sum_{i=1}^{n} i \quad + \quad s(0) \quad = \quad \sum_{i=1}^{n} i \quad + \quad 0 \quad = \quad n\frac{n+1}{2}$$

- Portanto:

$$s(n) \quad = \quad n\frac{n+1}{2}$$

Exemplo 3: Iteração (1)

- Resolver

$$T(n) = \begin{cases} c & n = 1 \\ 2T\left(\frac{n}{2}\right) + c & n > 1 \end{cases}$$

Exemplo 3: Iteração (2)

- $T(n) = 2T(n/2) + c$
  $= 2(2T(n/2/2) + c) + c$
  $= 2^2T(n/2^2) + 2c + c$
  $= 2^2(2T(n/2^2/2) + c) + 3c$
  $= 2^3T(n/2^3) + 4c + 3c$
  $= 2^3T(n/2^3) + 7c$
  $= 2^3(2T(n/2^3/2) + c) + 7c$
  $= 2^4T(n/2^4) + 15c$
  $= \ldots$
  $= 2^kT(n/2^k) + (2^k - 1)c$

$$T(n) = \begin{cases} c & n = 1 \\ 2T\left(\frac{n}{2}\right) + c & n > 1 \end{cases}$$

Exemplo 3: Iteração (3)

- Até agora para $n > 2^k$ temos
  - $T(n) = 2^k T(n/2^k) + (2^k - 1)c$
- E se $k=\lg n$?
  - $T(n) = 2^{\lg n} T(n/2^{\lg n}) + (2^{\lg n} - 1)c$

$$= n\,T(n/n) + (n - 1)c$$

$$= n\,T(1) + (n-1)c$$

$$= nc + (n-1)c = (2n - 1)c$$

$$a^{\lg_b c} = c^{\lg_b a}$$

$$T(n) = \begin{cases} c & n = 1 \\ 2T\left(\frac{n}{2}\right) + c & n > 1 \end{cases}$$

Exemplo 4: Iteração (1)

- Resolver

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

Exemplo 4: Iteração (2)

- T(n) =

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

aT(n/b) + cn

a(aT(n/b/b) + cn/b) + cn

$a^2T(n/b^2)$ + cna/b + cn

$a^2T(n/b^2)$ + cn(a/b + 1)

$a^2(aT(n/b^2/b) + cn/b^2)$ + cn(a/b + 1)

$a^3T(n/b^3)$ + cn(a^2/b^2) + cn(a/b + 1)

$a^3T(n/b^3)$ + cn(a^2/b^2 + a/b + 1)

...

$a^kT(n/b^k)$ + cn(a^{k-1}/b^{k-1} + a^{k-2}/b^{k-2} + ... + a^2/b^2 + a/b + 1)

Exemplo 4: Iteração (3)

- Assim, temos
  - $T(n) = a^k T(n/b^k) + cn(a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$
- Para $k = \log_b n$
  - $n = b^k$
  - $T(n) = a^k T(1) + cn(a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$

    $= a^k c + cn(a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$

    $= ca^k + cn(a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$

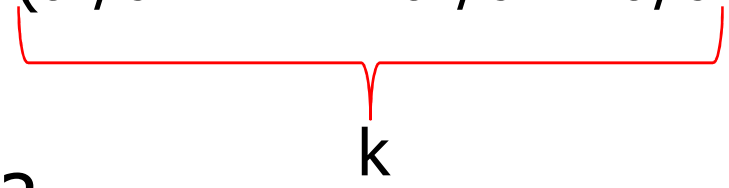
    $= cn(a^k/n + a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$

    $= cn(a^k/b^k + a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$

$$\lg_b a = \frac{\lg_c a}{\lg_c b}$$

Exemplo 4: Iteração (4)

- Então, para $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$



- E se a = b?
  - $T(n) = cn(k + 1)$

    $= cn(\log_b n + 1)$

    $= \Theta(n \log n)$

Exemplo 4: Iteração (5)

- E se a < b?

$T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$

  - Lembre que:

$$\sum_{n=0}^{k} x^n = \left(x^k + x^{k-1} + \ldots + x^2 + x + 1\right) = \frac{x^{k+1} - 1}{x - 1}$$

  - Temos:

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 = \frac{(a/b)^{k+1} - 1}{(a/b) - 1} = \frac{1 - (a/b)^{k+1}}{1 - (a/b)} < \frac{1}{1 - a/b}$$

  - Então:

$$T(n) = cn \cdot \Theta(1) = \Theta(n)$$

Exemplo 4: Iteração (6)

- E se a > b?

$T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 = \frac{(a/b)^{k+1} - 1}{(a/b) - 1} = \Theta\left((a/b)^k\right)$$

- $T(n) = cn \cdot \Theta(a^k / b^k)$

  $= cn \cdot \Theta(a^{\log_b n} / b^{\log_b n}) = cn \cdot \Theta(a^{\log_b n} / n)$

  como, $a^{\log_b n} = n^{\log_b a}$

  $= cn \cdot \Theta(n^{\log_b a} / n) = \Theta(cn \cdot n^{\log_b a} / n)$

  $= \Theta(n^{\log_b a})$

Exemplo 4: Iteração (7)

- Portanto, finalmente …

$$T(n) = \begin{cases} \Theta(n) & a < b \\ \Theta(n \log_b n) & a = b \\ \Theta\left(n^{\log_b a}\right) & a > b \end{cases}$$

Exercício 1: Iteração (1)

- Considere um sistema discreto modelado pela seguinte eq. de diferença onde 0<a<1:

$$y(n) = -ay(n-1) + x(n) \qquad x(n) = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases}$$

- Pode ser representado pela seguinte recursão:

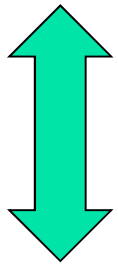
$$s(n) = \begin{cases} 0, & \mathrm{n} < 0 \\ 1, & \mathrm{n} = 0 \\ -as(n-1) + c, & n > 0 \end{cases}$$

- Qual seria o tempo de execução deste sistema?

Exercício 1: Iteração (2)

- $s(n) = -as(n-1) + c$
  $= -a(-as(n-2)+c)+c$
  $= a^2s(n-2)-ac+c$
  $= a^2(-as(n-3)+c)-ac+c$
  $= -a^3s(n-3)+a^2c-ac+c$
  $= -a^3(-as(n-4)+c)+a^2c-ac+c$
  $= a^4s(n-4)-a^3c+a^2c-ac+c$
  ...
  $= a^ks(n-k)-a^{k-1}c+a^{k-2}c-...-ac+c$
  $= a^ks(n-k)+c(-a^{k-1}+ a^{k-2}-...-a+1)$

$s(n) = -as(n-1) + c$
$s(n-1) = -as(n-2)+c$
$s(n-2) = -as(n-3)+c$
$s(n-3) = -as(n-4)+c$



$$s(n) = \begin{cases} 0, & n < 0 \\ 1, & n = 0 \\ -as(n-1)+c, & n > 0 \end{cases}$$

Exercício 1: Iteração (3)

- Até agora para *n* > *k* temos
  - $a^k s(n-k)+c(-a^{k-1}+ a^{k-2}-\ldots-a+1)$
- E se k=n ?
  - $s(n) = a^n + c(-a^{n-1}+ a^{n-2}-\ldots-a+1)$
- E se k > n?
  - $s(n) = c(-a^{n-1}+ a^{n-2}-\ldots-a+1)$

$$s(n)=\begin{cases}0, & n<0\\ 1, & n=0\\ -as(n-1)+c, & n>0\end{cases}$$

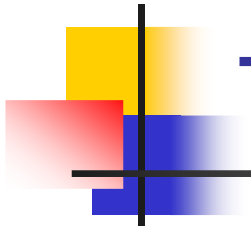
Exercício 2: Iteração

- Agora considere um sistema discreto modelado pela seguinte eq. de diferença de segunda ordem:

$$y(n) = -ay(n-1) - by(n-2) + x(n) \qquad x(n) = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases}$$

Qual seria o tempo de execução deste sistema?

Teorema Mestre



Teorema Mestre

- Considere um algoritmo que use **divisão e conquista**
  - Divide um problema de tamanho *n* em *a* sub-problemas, cada um de tamanho *n*/*b* (interpretamos *n/b* como $\lfloor n/b \rfloor$ ou $\lceil n/b \rceil$, pois *n/b* deve ser um inteiro)
- Seja $f(n)=D(n)+C(n)$ custo de cada estágio, ou seja, o custo de dividir (D) os problemas e combinar (C) as soluções
- O método mestre consiste em uma receita de bolo para encontrar a função de complexidade

Teorema Mestre

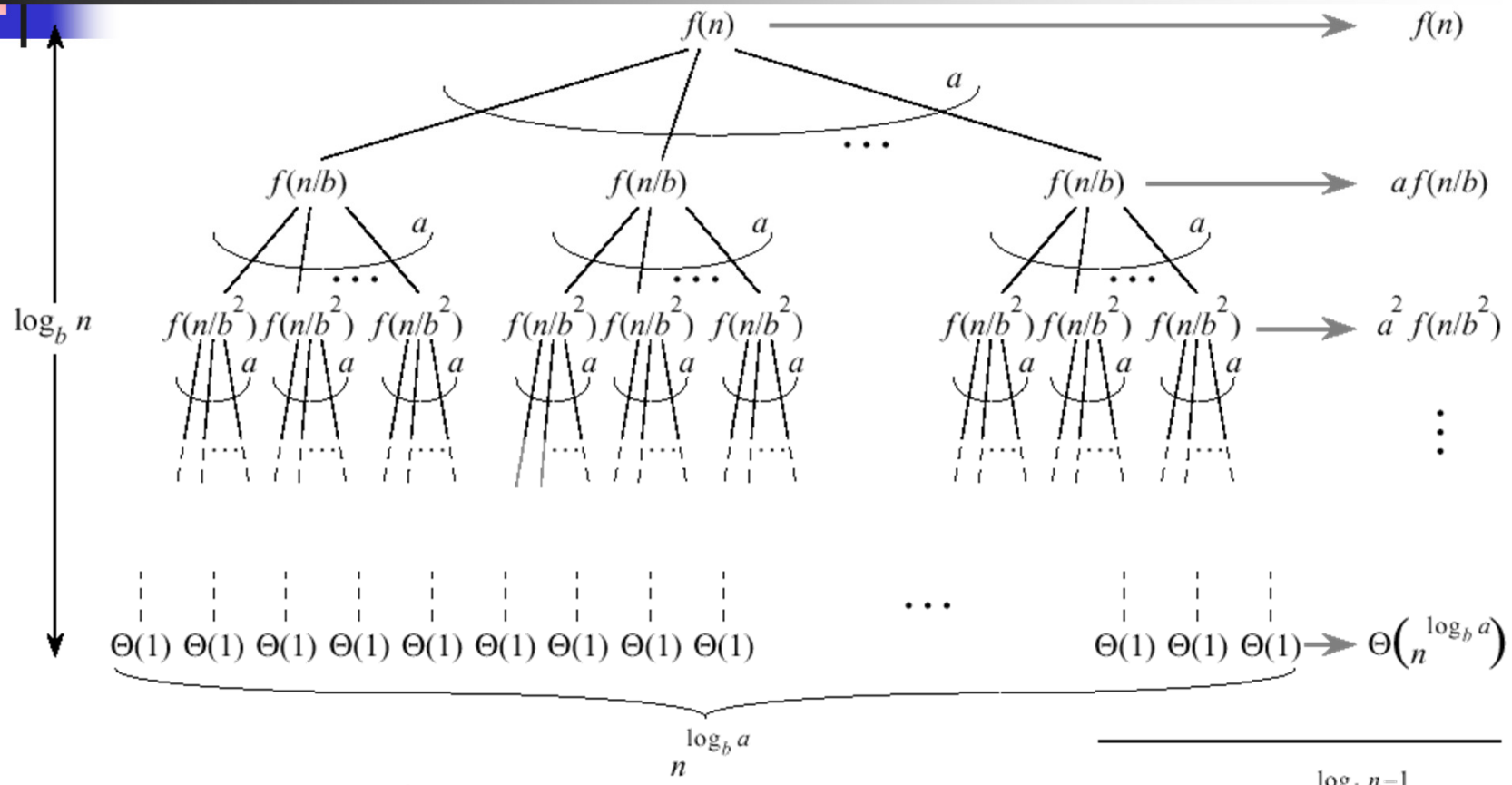
- Usado para a solução de recorrências da forma

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

- $a \geq 1$, $b > 1$, e f assintóticamente positiva;
- Sendo $T(n)$ o tempo de execução do algoritmo, podemos dizer que
  - *Sub-problemas de tamanho n/b* são resolvidos recursivamente cada um em tempo $T(n/b)$
  - $f(n)$ é o custo de dividir o problema e combinar seus resultados. No *MergeSort* temos:

$$T(n) = 2T(n/2) + \Theta(n) \qquad a = 2, b = 2, f(n) = \Theta(n)$$

Teorema Mestre (2)



- Os sub problemas são divididos em *a* partes.
- Existirão $log_b n$ níveis
- Ocorrerão $a^{\log_b n} = n^{\log_b a}$ folhas

Teorema Mestre (3)

- Três casos comuns:
  - Tempo de execução é dominado pelo custo nas folhas
  - Tempo de execução é uniformemente distribuído pela árvore
  - Tempo de execução é dominado pelo custo na raiz
- Assim, para resolver a recorrência, teremos de caracterizar o termo dominante
- Para cada caso comparar

$f(n)$ $O(n^{\log_b a})$

Resumo do Teorema Mestre

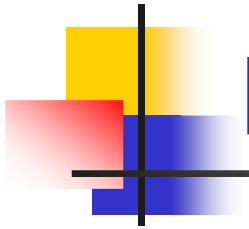
- Seja uma recorrência da forma

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

1. $f(n) = O(n^{\log_b a - \varepsilon}), \varepsilon > 0 \quad \Rightarrow \quad T(n) = \Theta(n^{\log_b a})$
2. $f(n) = \Theta(n^{\log_b a}) \quad \Rightarrow \quad T(n) = \Theta(n^{\log_b a} \lg n)$
3. $f(n) = \Omega(n^{\log_b a + \varepsilon}), \varepsilon > 0 \quad \Rightarrow \quad T(n) = \Theta(f(n))$

   $sendo\ a.f(n/b) \leq c.f(n), c < 1$

- Para cada caso comparar *f(n)* com $O(n^{\lg_b a})$



Estratégia

- **Passo 1:** Identificar *a*, *b* e *f (n)*
- **Passo 2:** Determinar $n^{\log_b a}$
- **Passo 3:** Comparar *f (n)* e $n^{\log_b a}$ assintóticamente
- **Passo 4:** De acordo com o caso, aplicar a regra correspondente

Exemplo 1

- $T(n) = 9T(n/3) + n$
  - Passo1: $a=9,\ b=3,\ f(n) = n$
  - Passo2: $n^{\log_b a} = n^{\log_3 9} = n^2$
  - Passo 3: $f(n) = O(n^{\log_3 9 - \varepsilon})$, $O(n)$ onde $\varepsilon=1$
  - Passo 4: o caso 1 se aplica:

$$T(n) = \Theta\left(n^{\log_b a}\right) \text{ quando } f(n) = O\left(n^{\log_b a - \varepsilon}\right)$$

  - Portanto $T(n) = \Theta(n^2)$

Exemplo 2: Mergesort

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

$$a = 2,\ b = 2;\ n^{\log_b a} = n^{\log_2 2} = n = \Theta(n)$$

Como $f(n) = \Theta(n)$

Temos o caso 2 :

$$f(n) = \Theta(n^{\log_b a}) \quad T(n) = \Theta(n^{\log_b a} \lg n)$$

Aplicando os valores:

$$T(n) = \Theta\left(n^{\log_b a} \lg n\right) = \Theta\left(n \lg n\right)$$

Exemplo 3

- Encontre o tempo de execução para recorrência:

$$T(n) = 3T(n/4) + n \lg n$$

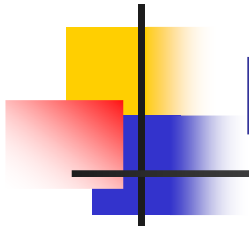
- Passo 1: *a=3*, *b=4* e *f(n)=nlg n*
- Passo 2: $n^{\log_b a} = n^{\log_4 3} = n^{0,793}$
- Passo 3: $f(n) = \Omega(n^{\log_4 3 + \varepsilon})$, $O(n)$ onde $\varepsilon \approx 0,2$
- Passo 4: o caso 3 se aplica,

$T(n) = \Theta(f(n))$ quando $f(n) = O(n^{\log_b a + \varepsilon})$

$sendo$ a.f$(n/b) \leq c.f(n), c < 1$ para n suficientemente grande

$3(n/4)\lg(n/4) \leq cn \lg n = 3/4 f(n)$ para c = 3/4

- Portanto, T(n)= $\Theta(n \lg n)$



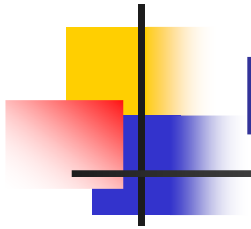
Exercício

- Use o teorema mestre para fornecer limites assinatóticos restritos para as recorrências a seguir

$$T(n) = 4T(n/2) + n$$

$$T(n) = 4T(n/2) + n^2$$

$$T(n) = 4T(n/2) + n^3$$



Método aniquilador

- Sugestão para estudo mais aprofundado