Recursividade e Equações de Recorrência

Prof. Jussara Almeida

Adaptado do material de Prof. Luiz Chaimowicz

Algoritmos Recursivos

- Um procedimento recursivo é aquele que chama a si mesmo direta ou indiretamente
- Normalmente possui duas partes
  - Uma ou mais chamadas recursivas
  - Condição de Parada
- Vantagens:
  - implementação mais simples, normalmente direta a partir da definição do problema
- Desvantagens:
  - Custo das chamadas recursivas / pilha de ativação

Exemplo Clássico 1

- Fatorial:

$$\begin{cases} n! = n*(n-1)! & p/\ n>0 \\ 0! = 1 & p/\ n=0 \end{cases}$$

- Em C

```
int Fat (int n) {
    if (n<=0)
        return 1;
    else
        return n * Fat(n-1);
}
```

Custo:

Tempo: Θ(n)

Espaço: Θ(n)

☹

Implementação iterativa gasta Θ(1) de espaço...

Exemplo Clássico 2

De quando **não** se deve usar recursividade

- **Série de Fibonnaci:**

$$\begin{cases} F_n = F_{n-1} + F_{n-2} & n > 2, \\ F_0 = F_1 = 1 \end{cases}$$

  - 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89...

```
Fib(int n) {
  if (n<2)
    return 1;
  else
    return Fib(n-1)+Fib(n-2);
}
```

Custo: $\Theta(2^n)$!?!

Implementação iterativa gasta $\Theta(n)$ de tempo e $\Theta(1)$ de espaço...

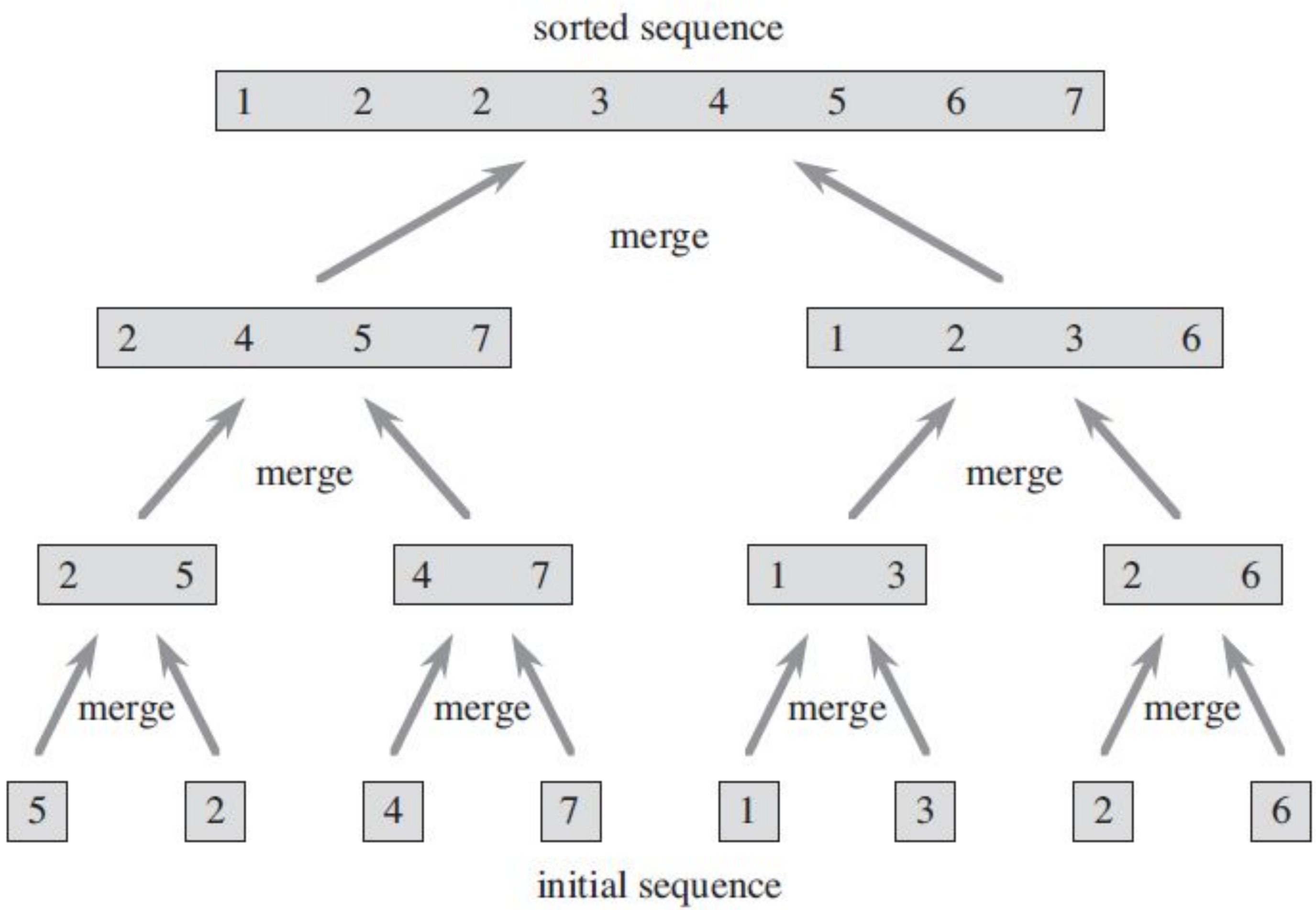
n	20	30	50	100
Recursiva	1 seg	2 min	21 dias	10^9 anos
Iterativa	1/3 mseg	1/2 mseg	3/4 mseg	1,5 mseg

Algoritmos *Dividir para Conquistar*

- Algoritmos que dividem o problema em vários subproblemas menores que são resolvidos recursivamente (detalhes com o Prof. Vinícius)
- Normalmente possuem um custo equivalente aos algoritmos iterativos, que requerem o uso de estruturas auxiliares (pilhas, etc)
- Exemplo: **Mergesort:**

```
MERGE-SORT(A, p, r)
if p < r
    q = ⌊(p + r)/2⌋
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q + 1, r)
    MERGE(A, p, q, r)
```

MergeSort


sorted sequence
1 2 2 3 4 5 6 7
merge
2 4 5 7
1 2 3 6
merge
merge
2 5
4 7
1 3
2 6
merge
merge
merge
merge
5
2
4
7
1
3
2
6
initial sequence

Análise de Algoritmos Recursivos

- A análise de algoritmos recursivos consiste na **representação** do seu custo através de uma **equação de recorrência** e na **resolução** desta equação
- Equações de Recorrência
  - Descrevem o custo de resolver um problema de tamanho *n* em termos da solução de problemas menores
  - Exigem um ferramental matemático em sua resolução

Análise de Algoritmos Recursivos

- Representação através da recorrência
  - Requer um entendimento dos custos envolvidos na chamada recursiva e demais operações
- Resolução da equação de recorrência
  - Requer o uso de diferentes métodos:
    - "Expansão de termos" (ou "método iterativo")
    - Teorema Mestre
    - Método da Substituição
    - Método da Árvore de Recursão

Análise do Fatorial

```
int Fat (int n) {
  if (n<=0)
    return 1;
  else
    return n * Fat(n-1);
}
```

$$\begin{cases} T(n) = T(n-1) + c & \text{p/ } n>0 \\ T(n) = d & \text{p/ } n\leq 0 \end{cases}$$

Expansão de termos:

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

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

…

$T(2) = T(1) + c$

$T(1) = T(0) + c$

$T(0) = d$

$T(n) = c+c+\ldots+c+d$

$T(n) = nc + d$

$\mathbf{T(n) = \Theta(n)}$

Análise do Mergesort

```
MERGE-SORT(A, p, r)
if p < r
    q = ⌊(p + r)/2⌋
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q + 1, r)
    MERGE(A, p, q, r)
```

$$
\begin{cases} T(n) = 2*T(n/2) + n & \text{p/ } n>1 \\ T(n) = 1 & \text{p/ } n=1 \end{cases}
$$

Alguns Detalhes:

- n potência de 2
- Algumas vezes caso base é omitido

Resolvendo por Expansão

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

$$T(1) = 1$$

Expandindo a equação :

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

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

$$4T(n/4) = 8T(n/8) + n$$

$$\vdots$$

$$2^{i-1}T(n/2^{i-1}) =$$

$$= 2^{i}T(n/2^{i}) + n$$

Substituindo os termos :

$$T(n) = 2^{i}T(n/2^{i}) + i.n$$

Caso base :

$$T(n/2^{i}) \rightarrow T(1)$$

$$n/2^{i} = 1 \rightarrow i = \log_2 n$$

Logo :

$$T(n) = 2^{i}T(n/2^{i}) + i.n$$

$$= 2^{\log_2 n}.1 + (\log_2 n).n$$

$$= n + n.\log_2 n = \Theta(n.\log_2 n)$$

Procedimento Recursivo

```
    Pesquisa(n);
(1) if (n ≤ 1)
(2) 'inspecione elemento' e termine
    else{
(3)       para cada um dos n elementos 'inspecione elemento';
(4)       Pesquisa(n/3);
     }
```

- Para cada procedimento recursivo é associada uma função de complexidade $f(n)$ desconhecida, onde n mede o tamanho dos argumentos para o procedimento.
- Obtemos uma equação de recorrência para $f(n)$.
- **Equação de recorrência**: maneira de definir uma função por uma expressão envolvendo a mesma função.

Análise do Procedimento Recursivo

- Seja $T(n)$ uma função de complexidade que represente o número de inspeções nos n elementos do conjunto.
- O custo de execução das linhas (1) e (2) é $O(1)$ e o da linha (3) é exatamente n.
- Usa-se uma **equação de recorrência** para determinar o nº de chamadas recursivas.
- O termo $T(n)$ é especificado em função dos termos anteriores $T(1)$, $T(2)$, ..., $T(n-1)$.
- $T(n) = n + T(n/3), \quad T(1) = 1$ (para $n = 1$ fazemos uma inspeção)
- Por exemplo, $T(3) = T(3/3) + 3 = 4$, $T(9) = T(9/3) + 9 = 13$, e assim por diante.
- Para calcular o valor da função seguindo a definição são necessários $k - 1$ passos para computar o valor de $T(3^k)$.

Exemplo de Resolução de Equação de Recorrência

- Sustitui-se os termos $T(k)$, $k < n$, até que todos os termos $T(k)$, $k > 1$, tenham sido substituídos por fórmulas contendo apenas $T(1)$.

$$\begin{aligned} T(n) &= n + T(n/3) \\ T(n/3) &= n/3 + T(n/3/3) \\ T(n/3/3) &= n/3/3 + T(n/3/3/3) \\ \vdots \quad &\quad\; \vdots \\ T(n/3/3\cdots/3) &= n/3/3\cdots/3 + T(n/3\cdots/3) \end{aligned}$$

- Adicionando lado a lado, temos $T(n) = n + n\cdot(1/3) + n\cdot(1/3^2) + n\cdot(1/3^3) + \cdots + (n/3/3\cdots/3)$ que representa a soma de uma série geométrica de razão $1/3$, multiplicada por n, e adicionada de $T(n/3/3\cdots/3)$, que é menor ou igual a 1.

Exemplo de Resolução de Equação de Recorrência

$$T(n) = n + n \cdot (1/3) + n \cdot (1/3^2) + n \cdot (1/3^3) + \cdots +$$
$$+ (n/3/3 \cdots /3)$$

- Se desprezarmos o termo $T(n/3/3 \cdots /3)$, quando n tende para infinito, então $T(n) = n \sum_{i=0}^{\infty} (1/3)^i = n \left(\frac{1}{1-\frac{1}{3}} \right) = \frac{3n}{2}$.
- Se considerarmos o termo $T(n/3/3/3 \cdots /3)$ e denominarmos x o número de subdivisões por 3 do tamanho do problema, então $n/3^x = 1$, e $n = 3^x$. Logo $x = \log_3 n$.
- Lembrando que $T(1) = 1$ temos $T(n) = \sum_{i=0}^{x-1} \frac{n}{3^i} + T(\frac{n}{3^x}) = n \sum_{i=0}^{x-1} (1/3)^i + 1 = \frac{n(1-(\frac{1}{3})^x)}{(1-\frac{1}{3})} + 1 = \frac{3n}{2} - \frac{1}{2}$.
- Logo, o programa do exemplo é $O(n)$.

Teorema Mestre

- Pode ser aplicado para resolver recorrências da forma:

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

onde $a \geq 1$ e $b > 1$ e $f(n)$ é uma função assintoticamente positiva

Intuição:

- algoritmo divide problema em a subproblemas, cada um com tamanho n/b, resolve os a subproblemas e combina as soluções na solução global
- f(n) = custo de dividir os problemas e combinar os resultados dos subproblemas

Teorema Mestre

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

- Pode-se derivar o limite assintótico pra T(n) para três casos:

1. Se $f(n) = O\left(n^{log_b a - \epsilon}\right)$ para alguma constante $\epsilon > 0$: T(n) = $\Theta(n^{log_b a})$

2. Se $f(n) = \Theta\left(n^{log_b a} \, log^k n\right)$ com k $\geq$ 0: T(n) = $\Theta(n^{log_b a} log^{k+1} n)$

3. Se f(n) = $\Omega(n^{log_b a + \epsilon})$ para $\epsilon > 0$, e se af(n/b) $\leq$ cf(n) para alguma constante c < 1 e para todo n suficientemente grande: T(n) = $\Theta(f(n))$

Teorema Mestre

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

- Pode-se derivar o limite assintótico pra T(n) para três casos:

1. Se $f(n) = O\left(n^{log_b a - \epsilon}\right)$ para alguma constante ϵ > 0: T(n) = $\Theta\left(n^{log_b a}\right)$

Significa que f(n) é polinomialmente (por um fator n^{ϵ}) menor que O$\left(n^{log_b a}\right)$

Se for menor, mas não polinomialmente menor: Teorema Mestre não se aplica

Teorema Mestre

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

- Pode-se derivar o limite assintótico pra T(n) para três casos:

3. Se f(n) = $\Omega(n^{log_b a + \epsilon})$ para ϵ > 0, e se af(n/b) $\leq$ cf(n) para alguma constante c < 1 e para todo n suficientemente grande: T(n) = $\Theta(f(n))$

Significa que f(n) é polinomialmente (por um fator n^{ϵ}) maior que O$\left(n^{log_b a}\right)$
Condição de regularidade tem que ser atendida.

Se for maior, mas não polinomialmente menor: Teorema Mestre não se aplica

Teorema Mestre

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

- Pode-se derivar o limite assintótico pra T(n) para três casos:

2. Se $f(n) = \Theta\left(n^{log_b a}\, log^k n\right)$ com k ≥ 0: T(n) = $\Theta(n^{log_b a} log^{k+1} n)$

Para k = 0 (caso mais comum):

As duas funções, f(n) e $n^{log_b a}$, são polinomialmente equivalentes;

Solução equivale a f(n) multiplicado por um fator logaritmo (log n)

Para k > 0

f(n) domina $n^{log_b a}$ por um fator $log^k n$

Solução equivale a f(n) multiplicado por um fator (poli)logaritmo (log n)

Aplicação do Teorema Mestre

1. T(n) = 9T(n/3) + n

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

1. Se $f(n) = O\left(n^{\log_b a - \epsilon}\right)$:
   T(n) = $\Theta(n^{\log_b a})$

2. Se $f(n) = \Theta\left(n^{\log_b a} \log^k n\right)$:
   T(n) = $\Theta(n^{\log_b a} \log^{k+1} n)$

3. Se f(n) = $\Omega(n^{\log_b a + \epsilon})$ e af(n/b) $\leq$ cf(n):
   T(n) = $\Theta(f(n))$

Aplicação do Teorema Mestre

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

1. Se $f(n) = O\left(n^{\log_b a\, -\, \epsilon}\right)$:
   $T(n) = \Theta(n^{\log_b a})$

2. Se $f(n) = \Theta\left(n^{\log_b a}\, \log^k n\right)$:
   $T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$

3. Se $f(n) = \Omega(n^{\log_b a +\, \epsilon})$ e $af(n/b) \leq cf(n)$:
   $T(n) = \Theta(f(n))$

1. $T(n) = 9T(n/3) + n$

- $a = 9$, $b = 3$; $f(n) = n$
- $n^{\log_b a} = n^2$
- $f(n) = O(n^{\log_b a - 1}) = O(n) \rightarrow \epsilon = 1$

Aplicação do Teorema Mestre

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

1. $T(n) = 9T(n/3) + n$

- $a = 9$, $b = 3$; $f(n) = n$
- $n^{log_b a} = n^2$
- $f(n) = O(n^{log_b a - 1}) = O(n) \rightarrow \epsilon = 1$

1. Se $f(n) = O\left(n^{log_b a - \epsilon}\right)$:
   $T(n) = \Theta(n^{log_b a})$
2. Se $f(n) = \Theta\left(n^{log_b a}\, log^k n\right)$:
   $T(n) = \Theta(n^{log_b a} log^{k+1} n)$
3. Se $f(n) = \Omega(n^{log_b a + \epsilon})$ e $af(n/b) \leq cf(n)$:
   $T(n) = \Theta(f(n))$

CASO 1 $\quad T(n) = \Theta\left(n^{log_3 9}\right) = \Theta(n^2)$

Aplicação do Teorema Mestre

2. T(n) = T(2n/3) + 1

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

1. Se $f(n) = O\left(n^{log_b a\, -\, \epsilon}\right)$:
   T(n) = $\Theta(n^{log_b a})$

2. Se $f(n) = \Theta\left(n^{log_b a}\, log^k n\right)$:
   T(n) = $\Theta(n^{log_b a} log^{k+1} n)$

3. Se f(n) = $\Omega(n^{log_b a +\, \epsilon})$ e af(n/b) $\leq$ cf(n):
   T(n) = $\Theta(f(n))$

Aplicação do Teorema Mestre

2. T(n) = T(2n/3) + 1

- a = 1, b= 3/2; f(n) = 1 = O(1)
- $n^{log_b a} = n^{log_{3/2} 1} = n^0 = 1$
- $f(n) = \Theta\left(n^{log_b a} \, log^k n\right)$ para k = 0

  $= \Theta\left(n^{log_{3/2} 1} \, log^0 n\right)$

  $= \Theta(n^0 \, 1) = \Theta(1)$

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

1. Se $f(n) = O\left(n^{log_b a - \epsilon}\right)$:
   T(n) = $\Theta(n^{log_b a})$
2. Se $f(n) = \Theta\left(n^{log_b a} \, log^k n\right)$:
   T(n) = $\Theta(n^{log_b a} log^{k+1} n)$
3. Se f(n) = $\Omega(n^{log_b a + \epsilon})$ e af(n/b) $\leq$ cf(n):
   T(n) = $\Theta(f(n))$

Aplicação do Teorema Mestre

2. T(n) = T(2n/3) + 1

- a = 1, b= 3/2; f(n) = 1 = O(1)
- $n^{log_b a} = n^{log_{3/2} 1} = n^0 = 1$
- $f(n) = \Theta\left(n^{log_b a}\, log^k n\right)$ para k =0
  $= \Theta\left(n^{log_{3/2} 1}\, log^0 n\right)$
  $= \Theta(n^0\, 1) = \Theta(1)$

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

1. Se $f(n) = O\left(n^{log_b a\, -\, \epsilon}\right)$:
   T(n) = $\Theta(n^{log_b a})$
2. Se $f(n) = \Theta\left(n^{log_b a}\, log^k n\right)$:
   T(n) = $\Theta(n^{log_b a} log^{k+1} n)$
3. Se f(n) = $\Omega(n^{log_b a + \epsilon})$ e af(n/b) $\leq$ cf(n):
   T(n) = $\Theta(f(n))$

CASO 2

T(n) = $\Theta(n^{log_{3/2} 1} log^{0+1} n)$

T(n) = $\Theta(n^0 \log n)$

T(n) = $\Theta(\log n)$

Aplicação do Teorema Mestre

3. T(n) = 3T(n/4) + n log n

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

1. Se $f(n) = O\left(n^{log_b a\,-\,\epsilon}\right)$:
   T(n) = $\Theta(n^{log_b a})$

2. Se $f(n) = \Theta\left(n^{log_b a}\, log^k n\right)$:
   T(n) = $\Theta(n^{log_b a} log^{k+1} n)$

3. Se f(n) = $\Omega(n^{log_b a + \,\epsilon})$ e af(n/b) $\leq$ cf(n):
   T(n) = $\Theta(f(n))$

Aplicação do Teorema Mestre

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

1. Se $f(n) = O(n^{log_b a\,-\,\epsilon})$:
   $T(n) = \Theta(n^{log_b a})$
2. Se $f(n) = \Theta(n^{log_b a}\, log^k n)$:
   $T(n) = \Theta(n^{log_b a} log^{k+1} n)$
3. Se $f(n) = \Omega(n^{log_b a+\,\epsilon})$ e $af(n/b) \leq cf(n)$:
   $T(n) = \Theta(f(n))$

3. T(n) = 3T(n/4) + n log n

- a = 3, b= 4; f(n) = n log n
- $n^{log_b a} = n^{log_4 3} = O\ (n^{0.793})$
- f(n) = $\Omega(n^{log_b a+\,\epsilon})$ para ϵ ~ 0.2
  $\Omega(n^{0.793+0.2})$
- Condição de regularidade válida para f(n)? af(n/b) $\leq$ cf(n) para algum c < 1?

af(n/b) = 3f(n/4) = 3 $\frac{n}{4}$ $\log(\frac{n}{4}) = \frac{3}{4} n \log(\frac{n}{4}) \leq \frac{3}{4} n$ log n = $\frac{3}{4}$ f(n) : SIM (c = ¾)

Aplicação do Teorema Mestre

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

1. Se $f(n) = O\left(n^{log_b a - \epsilon}\right)$:
   $T(n) = \Theta(n^{log_b a})$
2. Se $f(n) = \Theta\left(n^{log_b a}\, log^k n\right)$:
   $T(n) = \Theta(n^{log_b a} log^{k+1} n)$
3. Se $f(n) = \Omega(n^{log_b a + \epsilon})$ e $af(n/b) \leq cf(n)$:
   $T(n) = \Theta(f(n))$

3. $T(n) = 3T(n/4) + n \log n$

- $a = 3$, $b = 4$; $f(n) = n \log n$
- $n^{log_b a} = n^{log_4 3} = O\ (n^{0.793})$
- $f(n) = \Omega(n^{log_b a + \epsilon})$ para ϵ ~ 0.2
  $\Omega(n^{0.793+0.2})$
- Condição de regularidade válida para f(n)? $af(n/b) \leq cf(n)$ para algum $c < 1$?

SIM ($c = ¾$)

CASO 3 $T(n) = \Theta(n \lg n)$

Aplicação do Teorema Mestre

4. T(n) = 2T(n/2) + n log n

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

1. Se $f(n) = O\left(n^{\log_b a - \epsilon}\right)$:
   T(n) = $\Theta(n^{\log_b a})$
2. Se $f(n) = \Theta\left(n^{\log_b a} \log^k n\right)$:
   T(n) = $\Theta(n^{\log_b a} \log^{k+1} n)$
3. Se f(n) = $\Omega(n^{\log_b a + \epsilon})$ e af(n/b) $\leq$ cf(n):
   T(n) = $\Theta(f(n))$

Aplicação do Teorema Mestre

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

3. T(n) = 2T(n/2) + n log n

- a = 2, b= 2; f(n) = n log n
- $n^{log_b a} = n^{log_2 2}$= O(n)

1. Se $f(n) = O\left(n^{log_b a\,-\,\epsilon}\right)$:
   T(n) = $\Theta(n^{log_b a})$
2. Se $f(n) = \Theta\left(n^{log_b a}\,log^k n\right)$:
   T(n) = $\Theta(n^{log_b a} log^{k+1} n)$
3. Se f(n) = $\Omega(n^{log_b a+\,\epsilon})$ e af(n/b) ≤ cf(n):
   T(n) = $\Theta(f(n))$

- Caso 3? Note que não existe ϵ > 0 tal que f(n) = n log n $= \Omega(n^{log_b a+\,\epsilon})$= $\Omega(n^{\,1+\epsilon})$

f(n)/$n^{log_b a}$ = n log n / n = log n é assintoticamente menor que $n^{\,\epsilon}$

CASO 3 não se aplica

Caso 2? SIM !!! T(n) = O(n log^2 n)

Teorema Mestre - Exemplos

$$T(n) = 2T(n/2) + n - 1$$

onde $a = 2$, $b = 2$, $f(n) = n - 1$ e $n^{\log_b a} = n^{\log_2 2} = \Theta(n)$.

O caso 2 se aplica porque $f(n) = \Theta(n^{\log_b a}) = \Theta(n)$, e a solução é $T(n) = \Theta(n \log n)$.

$$T(n) = 3T(n/3) + n \log n$$

onde $a = 3$, $b = 3$, $f(n) = n \log n$ e $n^{\log_b a} = n^{\log_3 3} = n$.

O caso 3 não se aplica porque, embora $f(n) = n \log n$ seja assintoticamente maior do que $n^{\log_b a} = n$, a função $f(n)$ não é polinomialmente maior: a razão $f(n)/n^{\log_b a} = (n \log n)/n = \log n$ é assintoticamente menor do que n^{ϵ} para qualquer constante ϵ positiva.

Método da Substituição

- Dois passos:
  - Estimar ("chutar") uma solução
  - Usar indução matemática para mostrar que o "chute" é válido
- Pode ser usada para estabelecer limites superiores e inferiores para a recorrência
- Possíveis problemas
  - Estimação da solução
  - Cuidados na aplicação da indução

Método da Substituição - Exemplo

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

Chute: $T(n) = O(n \lg n)$

Devemos provar portanto que: $T(n) \leq c.n \lg n$

Vamos começar com o passo indutivo, mostrando que: se a a solução é verdadeira para todo *m* < *n* especialmente para *n / 2* ela é válida para *T(n).* Isso é feito **substituindo-se** *T(n/2)* na equação de recorrência:

$$T(n/2) \leq cn/2 \lg(n/2)$$

$$T(n) \leq 2(cn/2 \lg(n/2)) + n$$

$$\begin{aligned} T(n) &\leq cn \lg(n/2) + n \\ &= cn \lg n - cn \lg 2 + n \\ &= cn \lg n - cn + n \end{aligned}$$

$$T(n) \leq cn \lg n \text{ para } c \geq 1$$

Método da Substituição - Exemplo

Agora é necessário mostrar que a solução é valida para o caso base *T(1) = 1.*

O problema é que **ela não é válida!**

$$T(n) \le c.n \lg n \rightarrow T(1) \le c.1 \lg 1 = 0$$

Para resolver esse problema, basta lembrar que a notação assintótica exige que: $T(n) \le c.n \lg n \ \forall n \ge n_o$

Portanto, basta escolher outro caso base, como *T(2)*. Pela equação de recorrência, se *T(1) = 1* então *T(2) = 4*.

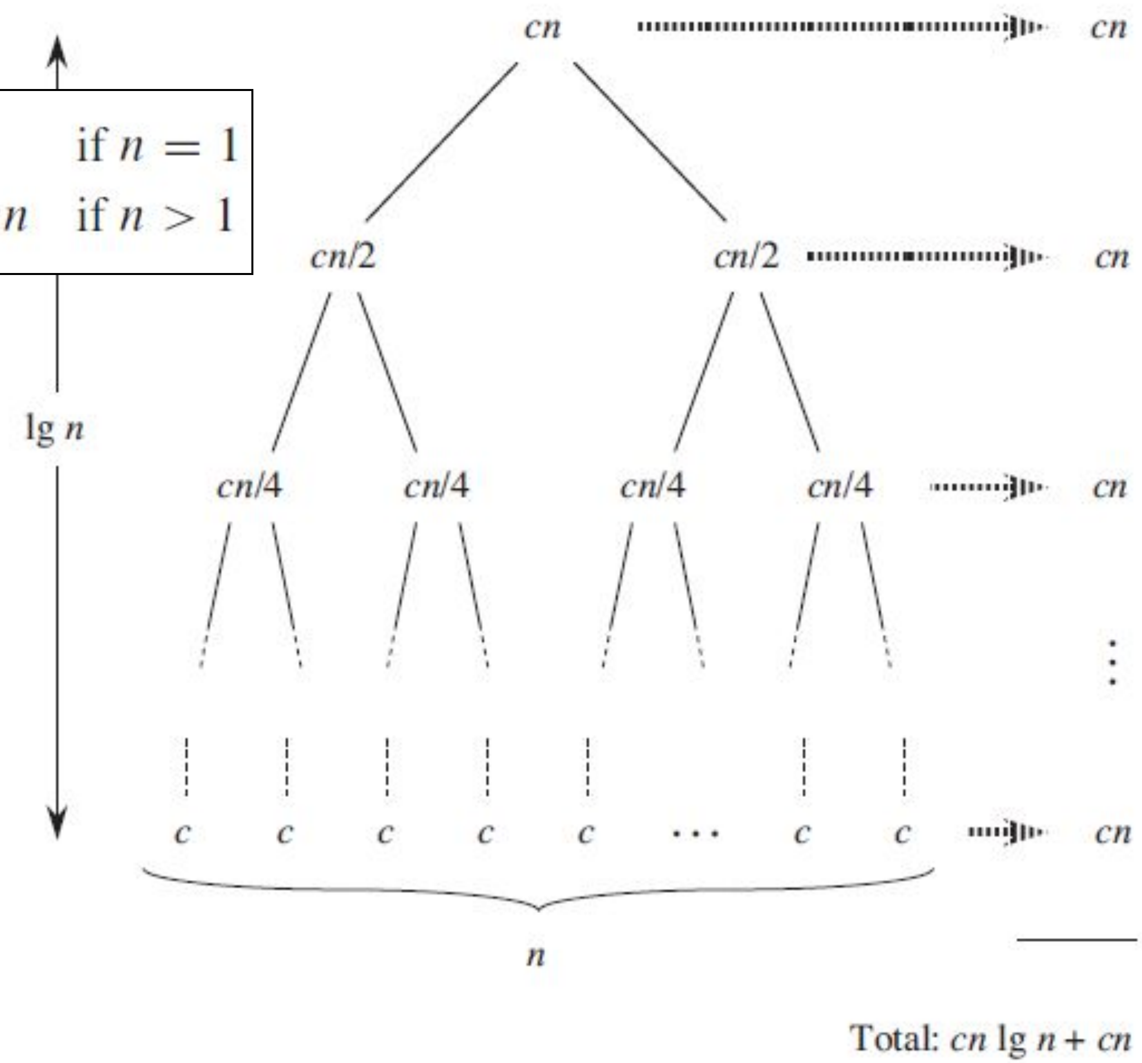
Logo podemos mostrar que $T(2) \le c.2 \lg 2$
é válida para c ≥ 2. Portando, a solução *T(n) = O(n.lg(n*)) funciona

Método da Árvore de Recursão

- Aplicando-se a recursão, monta-se uma árvore onde cada nó representa o custo de um subproblema.
- Expandindo-se a árvore, pode-se obter a soma de todos os custos de cada nível e depois do problema como um todo.
- De certa forma funciona como uma visualização do método da “expansão de termos” apresentado inicialmente.

Árvore de Recursão - Exemplo

$$T(n) = \begin{cases} c & \text{if } n = 1 \\ 2T(n/2) + cn & \text{if } n > 1 \end{cases}$$



$\Theta(n.lg(n))$

Comentários Gerais sobre os Métodos

- Segundo Cormen, os métodos da substituição e do Teorema Mestre são preferíveis.
  - O método iterativo deve ser usado com cuidado para evitar erros na expansão de simplificação de termos
  - O método da árvore de recursão não é um método matemático formal, portanto deve ser usado em conjunto com outros métodos

“Para Casa”

- Ler capítulo 4 do Cormen (ênfase nos métodos de resolução de equações de recorrência).

- Resolver os seguintes exercícios:

  Escolha algumas (todas?) equações de recorrência do livro e as resolva (exercícios: 4.5-1, 4-1, 4-3)