Algoritmos e Estrutura de Dados

Aula 4 –Notações, Funções Comuns e Recorrências

Prof. Tiago A. E. Ferreira

Notação O(·)

- As equações de custo são dependentes das constantes $c_1, c_2, \ldots, c_k$
  - Fica impraticável tentarmos estabelecer os valores exatos destas constantes
- Desta forma, é importante ter uma estimativa de como o custo cresce com o tamanho ***n*** da entrada
  - Considera-se o termo de maior importância da expressão de custo:
    - $\boldsymbol{an^2+bn+c}$ → ***O(n²) :*** *da ordem de* $\boldsymbol{n^2}$
    - ***an+b*** → ***O(n) :*** *da ordem de* ***n***

Notação O

- Quando se considera o número de passo efetuados por um algoritmo,
  - Pode-se desprezar constantes aditivas e multiplicativas
    - Ex.: $3n$ será aproximado por n
  - São considerados apenas valores assintóticos, termos de menor grau podem ser desprezados
    - n^2+n+5 será aproximado por n^2

Notação O

- Formalizando:
  - Sejam ***f(n)*** e ***g(n)*** funções positivas e uma variável inteira ***n***. Diz-se que ***f*** é ***O(g(n))***, escrevendo-se ***f=O(g(n))***, quando existir uma constante ***c>0*** e um valor inteiro n_0, tal que

$$n > n_0 \Rightarrow 0 \leq f(n) \leq c \cdot g(n)$$

  - Ou seja, a função ***g(n)*** é um **limite superior** para a função ***f(n)***
- Propriedades, onde ***g(n)*** e ***h(n)*** são funções e ***k*** é uma constante:
  - ***O(g(n)+h(n)) = O(g(n)) + O(h(n))***
  - ***O(k(n) · g(n)) = k · O(g(n)) = O(g(n))***

Notação Θ

- A notação Θ é útil para exprimir limites superiores
  - Sejam ***f(n)*** e ***g(n)*** funções reais e positivas da variável inteira ***n***. Diz que ***f*** **(n)** é ***Θ(g(n))***, escrevendo-se ***f(n) =Θ(g(n))***, quando
    - $0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$
  - A notação Θ exprime o fato que duas funções possuem a mesma ordem de grandeza assintótica.
  - Exs.:
    - $f=n^2-1$; $g=n^2$; $h=n^3$, assim, $f= \Theta(g)$ mas f não é $\Theta(h)$

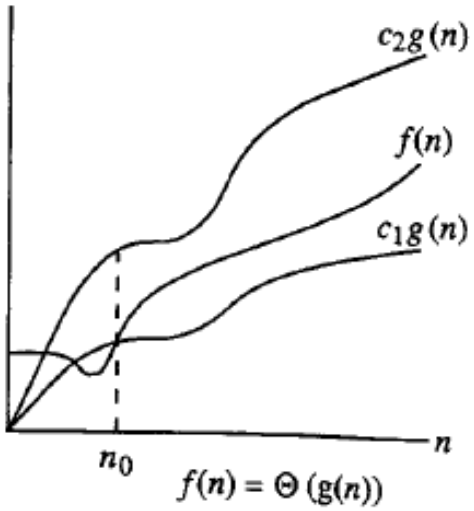
Notação Ω

- A notação Ω é utilizada para limites assintóticos inferiores
  - Sejam ***f(n)*** *e* ***g(n)*** funções reais positivas da variável ***n***. Diz-se que ***f(n)*** *é* ***Ω(g(n))***, escrevendo-se ***f(n)=Ω(g(n))***, quando existir uma constante $c > 0$ e um valor inteiro n_0, tal que

$$\boldsymbol{n > n_0 \Rightarrow f(n) \geq c \cdot g(n) \geq 0}$$

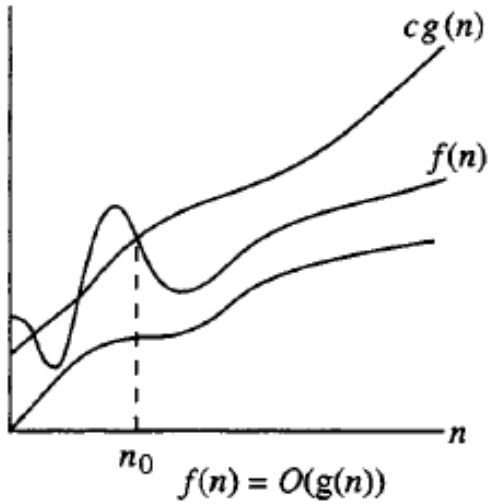
  - *Exemplo, se* $f=n^2-1$, *então é válido afirmar:*
    - $f = \Omega(n^2)$
    - $f = \Omega(n)$
    - $f = \Omega(1)$
    - *Mas não é válido afirmar que* $f = \Omega(n^3)$

Graficamente



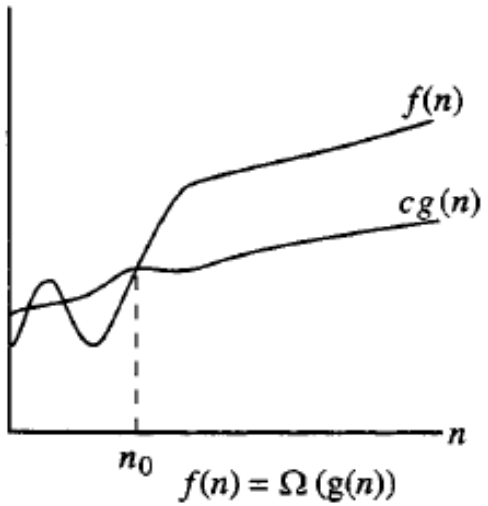
$f(n) = \Theta(g(n))$

(a)



$f(n) = O(g(n))$

(b)



$f(n) = \Omega(g(n))$

(c)

Notação o (minúsculo)

- O limite superior dado pela notação O pode ser
  - Assintoticamente restrito
    - $2n^2 = O(n^2)$
  - Não Assintoticamente restrito
    - $2n = O(n^2)$
- Para o caso não assintoticamente restrito é utilizada a notação o ("*o minúsculo*")

Notação o (minúsculo)

- Formalizando:
  - Sejam ***f(n)*** e ***g(n)*** funções positivas e uma variável inteira ***n***. Diz-se que ***f*** é ***O(g(n))***, escrevendo-se ***f=o(g(n))***, quando existir uma constante ***c>0*** e um valor inteiro n_0, tal que

$$n > n_0 \Rightarrow 0 \leq f(n) < c \cdot g(n)$$

  - ***Esta definição implica que:***

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

Notação ω

- Por analogia,
  - A notação ω está para a notação Ω assim como a notação **o** está para a notação **O**.
  - Utiliza-se a notação ω para denotar um limite inferior que não é assintoticamente restrito.
- Formalizando
  - Sejam ***f(n)*** *e* ***g(n)*** funções reais positivas da variável ***n***. Diz-se que ***f(n)*** *é* ***ω(g(n))***, escrevendo-se ***f(n)= ω(g(n))***, quando existir uma constante ***c > 0*** e um valor inteiro n_0, tal que

$$n > n_0 \Rightarrow f(n) > c \cdot g(n) \geq 0$$

Notação ω

- Por exemplo:
  - $n^2/2 = \omega(n)$
  - $n^2/2 \neq \omega(n^2)$
- Assim, se ***f(n) = ω(g(n))***, então:

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty$$

Propriedades

Transitividade:

$f(n) = \Theta(g(n))$ e $g(n) = \Theta(h(n))$ implicam $f(n) = \Theta(h(n))$,
$f(n) = O(g(n))$ e $g(n) = O(h(n))$ implicam $f(n) = O(h(n))$,
$f(n) = \Omega(g(n))$ e $g(n) = \Omega(h(n))$ implicam $f(n) = \Omega(h(n))$,
$f(n) = o(g(n))$ e $g(n) = o(h(n))$ implicam $f(n) = o(h(n))$,
$f(n) = \omega(g(n))$ e $g(n) = \omega(h(n))$ implicam $f(n) = \omega(h(n))$.

Reflexividade:

$f(n) = \Theta(f(n))$
$f(n) = O(f(n))$
$f(n) = \Omega(f(n))$

Simetria:

$f(n) = \Theta(g(n))$ se e somente se $g(n) = \Theta(f(n))$

Simetria de transposição:

$f(n) = O(g(n))$ se e somente se $g(n) = \Omega(f(n))$
$f(n) = o(g(n))$ se e somente se $g(n) = \omega(f(n))$

Tricotomia

- Para dois números reais quaisquer ***a*** e ***b***, exatamente uma das relações deve ocorrer
  - ***a < b;*** ou
  - ***a > b;*** ou
  - ***a = b.***
- Embora dois números quaisquer possam ser comparados, nem toda a função é assintoticamente comparável
  - Pode existir situação onde:
    - **f(n)** $\neq$ **O(g(n))**; e
    - **f(n)** $\neq$ **Ω(g(n))**

Exemplo

- Dadas as funções:
  - $f(n) = n$
  - $g(n) = n^{1+sen(n)}$
- Pergunta-se:
  - ***f(n)*** e ***g(n)*** **são assintoticamente comparáveis?**

Funções Comuns

- Funções monótonas
  - Função monotonicamente crescente (ou monotonamente crescente)
    - Se $m \leq n \Rightarrow f(m) \leq f(n)$
  - Função monotonicamente decrescente (ou monotonamente decrescente)
    - Se $m \leq n \Rightarrow f(m) \leq f(n)$
- Funções estritas
  - Função estritamente crescente
    - Se $m \leq n \Rightarrow f(m) < f(n)$
  - Função estritamente decrescente
    - Se $m \leq n \Rightarrow f(m) \leq f(n)$

Funções Comuns

- Piso
  - Seja x um número real, seu piso é o maior inteiro menor ou igual a *x*
  - Notação: $\lfloor X \rfloor$
- Teto
  - Seja x um número real, seu teto é o menor inteiro maior que ou igual a *x*
  - Notação $\lceil x \rceil$
- Para todo real x:
  - $x-1 > \lfloor x \rfloor \leq x \leq \lceil x \rceil < x+1$
- Para qualquer inteiro n
  - $\lfloor n/2 \rfloor + \lceil n/2 \rceil = n$
- Para qualquer real $n \geq 0$ e inteiros $a,b > 0$
  - $\lceil \lceil n/a \rceil / b \rceil = \lceil n/ab \rceil$
  - $\lfloor \lfloor n/a \rfloor / b \rfloor = \lfloor n/ab \rfloor$
  - $\lceil a/b \rceil \leq (a+(b-1))/b$
  - $\lfloor a/b \rfloor \leq (a-(b-1))/b$

Funções Comuns

- Aritmética Modular
  - Para qualquer inteiro ***a*** e qualquer inteiro positivo ***n***,

    $$a \bmod n = a - \lfloor a/n \rfloor n$$

  - *Se* ***(a mod n) = (b mod n)****,* então ***a*** ***≡ b (mod n)*** *(****a*** *é equivalente a* ***b****, modulo* ***n****)*
- Polinômios
  - Dado um inteiro não negativo ***d***, um *polinômio em* ***n*** *de grau* ***d*** é uma função ***p(n)*** da forma,

    $$p(n) = \sum_{i=0}^{d} a_i n^i$$

  - É dito que uma função *f(n)* é **polinomialmente limitada** se ***f(n)=O(n^k)*** para alguma constante ***k***.

Funções Comuns

- Função Exponencial
  - Para todo ***n*** *e* ***a*** ***≥ 1***, a função a^n é **monotonicamente crescente** em ***n***.
  - Para ***a>1***,

$$\lim_{n\to\infty} \frac{n^b}{a^n} = 0 \quad \Rightarrow \quad n^b = O\left(a^n\right)$$

    - ***Portanto, qualquer função exponencial com base escritamente maior que 1 cresce mais rapidamente que qualquer função polinomial***
- Função Logarítmica

$$\lim_{n\to\infty} \frac{\log_b n}{n^a} = 0 \quad \Rightarrow \quad \log_b n = O\left(n^a\right)$$

  - Para qualquer constante *a>0.* Qualquer função polinomial positiva cresce mais rapidamente que qualquer função polilogarítmica.

Funções Comuns

- Função Fatorial

$$n! = \begin{cases} 1 & se\ n = 0 \\ n \cdot (n-1)! & se\ n > 0 \end{cases}$$

  - Note que $n! \leq n^n$, logo $n! = O(n^n)$

- Iteração Funcional

$$f^{(i)}(n) = \begin{cases} n & se\ i = 0 \\ f\left(f^{(i-1)}(n)\right) & se\ i > 0 \end{cases}$$

  - Se $f(n) = 2n$, então $f^{(i)}(n) = 2^i n$

Números de Fibonacci

- Os números de Fibonacci são definidos como:
  - $F_0=0$
  - $F_1=1$
  - $F_i=F_{i-1}+F_{i-2}$, para $i \geq 2$
- Seqüência: 0,1,1,2,3,5,8,13,21,34,55,89,...
- Uma forma alternativa e definir os números de Fibonacci é a partir da **razão áurea** ϕ e seu conjugado $\hat{\phi}$

$$\left.\begin{aligned} \phi &= \frac{1+\sqrt{5}}{2} = 1{,}61803\ldots \\ \hat{\phi} &= \frac{1-\sqrt{5}}{2} = 0{,}61803\ldots \end{aligned}\right\} F_i = \frac{\phi^i - \hat{\phi}^i}{\sqrt{5}}$$

Recorrência

- Quando um algoritmo contem uma chamada recursiva a ele próprio, seu tempo de execução pode frequentemente ser descrito por **recorrência.**
- Veremos três métodos para resolver recorrência:
  - Método da substituição
    - É suposto um limite hipotético e depois usa-se a indução matemática para se provar a suposição
  - Método de Árvore de Recursão
    - Converte-se a recorrência em uma árvore, onde esta é utilizada para o cálculo dos custos
  - Método Mestre
    - Fornece limites para recorrências da forma $T(n)=aT(n/b)+f(n)$, onde $a, b>0$ e $f(n)$ é uma função dada.

Método da Substituição

- Engloba duas etapas:
  - Pressupor a forma da solução
  - Usar a indução matemática para encontrar as constantes e mostrar que a solução funciona
- Este método pode ser utilizado para estabelecer limites superiores ou inferiores sobre uma recorrência

Exemplo

- Determinar um limite superior para a recorrência:
  - $T(n)=2T(\lfloor n/2 \rfloor)+n$
    - Supõem-se $T(n)=O(n \lg n)$
    - Quer-se provar que $T(n) \leq cn \lg n$, com $c> 0$
    - Começa-se supondo que este limite permanece válido para $\lfloor n/2 \rfloor$, ou seja, $T(\lfloor n/2 \rfloor) \leq c \lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)$
    - Na recorrência,

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

onde o último passo é válido desde que $c \geq 1$.

Exemplo (cont.)

- Ainda falta mostrar a validade em valores limites!
  - Vamos supor que T(1) = 1, então para n=1
    - O limite tem que ser válido: $T(n) \leq cn \lg n$, mas para n=1, $T(1) \leq c\ 1 \lg 1 = 0$, assim o caso básico deixa de ser válido!
    - Porém, devido a notação assintótica, $T(n) \leq cn \lg n$ deve ser válido para $n \geq n_0$
    - Fazendo $n_0 = 2$, os casos básicos são verdadeiros para um $c \geq 2$

Cuidado! Armadilhas!

- É fácil errar utilizando a notação assintótica!
- Ex.:
  - Dada a recorrência $T(n)=2T(\lfloor n/2 \rfloor)+n$, considere $T(n) \leq cn$, então:

$$
\begin{aligned}
T(n) &\leq 2(c\lfloor n/2 \rfloor) + n \\
&\leq cn + n \\
&= O(n) , \qquad \Leftarrow \textit{errado!!}
\end{aligned}
$$

  - O erro é que não se prova a forma exata da hipótese indutiva, $T(n) \leq cn$.

Método da Árvore de Recursão

- Uma grande dificuldade do método de substituição é encontrar uma boa suposição
- Traçar uma árvore de recursão pode ser uma forma de se encontrar uma boa suposição
  - Em uma árvore de recursão, tem-se:
    - Nó – Representa o custo de um único subproblema
    - Soma-se os custos por nível da árvore
    - Soma-se os custos de todos os níveis para se obter uma estimativa do custo total
  - Estas árvores de recursão são muito úteis para algoritmos do tipo dividir e conquistar

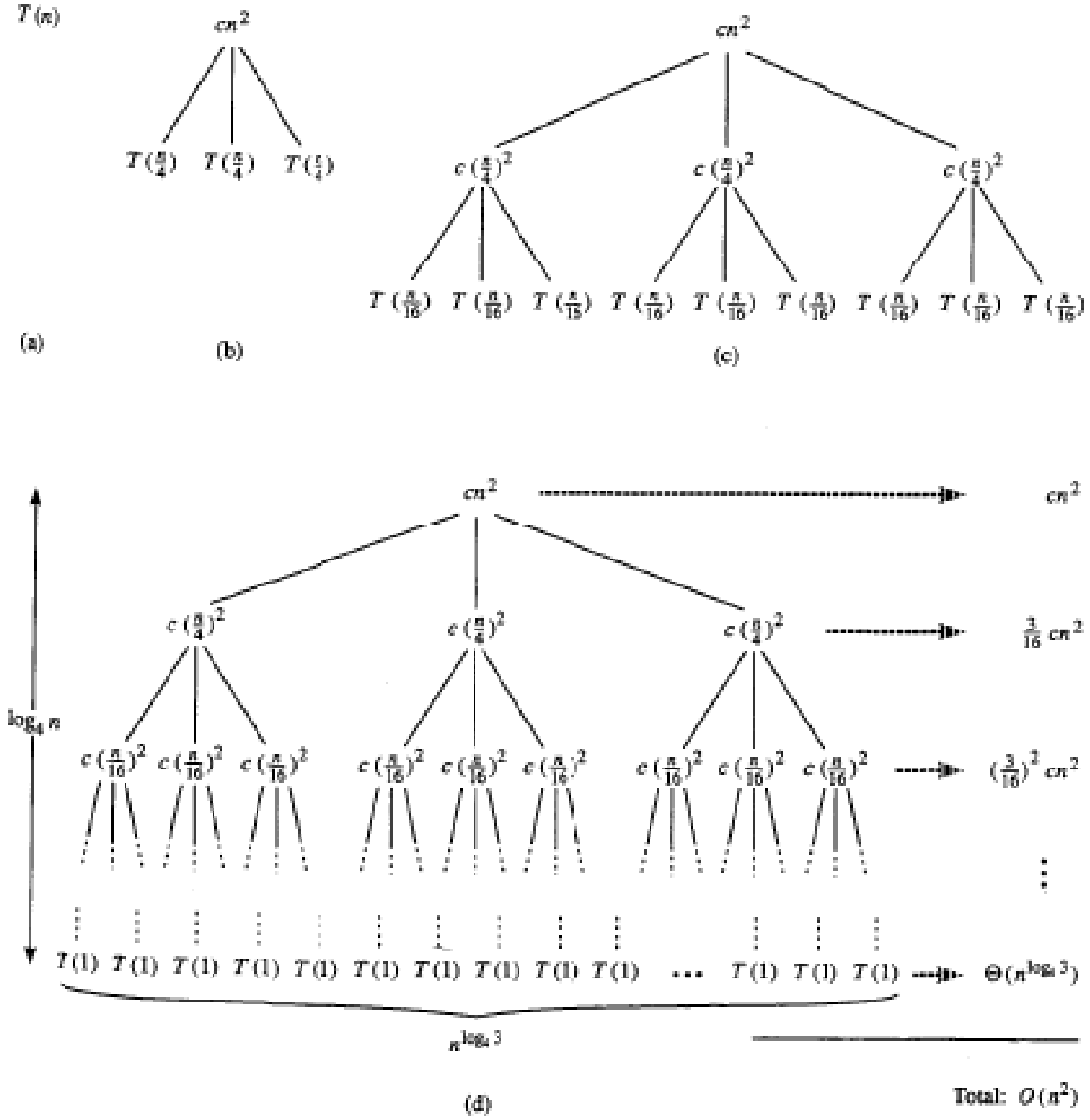
Método da Árvore de Recursão

- Dependendo da forma com que se admite ou não uma certa“sujeira” nos cálculos dos custos, uma árvore de recursão tanto pode ser utilizada para:
  - A geração de uma estimativa de custo, para a criação de uma boa suposição
  - Ou, para a determinação assintótica do custo do algoritmo

Gerando uma boa suposição...

- Dada a recursão: $T(n)=3T(\lfloor n/4 \rfloor)+\Theta(n^2)$
  - Deseja-se encontrar um limite superior!
  - "Sujeira" – tolera-se o fato de que pisos e tetos são normalmente insatisfatórios para resolver recorrências
  - Cria-se uma árvore de recursão para $T(n)=3T(n/4)+cn^2$, onde $c>0$
    - Por conveniência, é suposto que *n* é uma potência de 4 (outra "sujeira")

Gerando uma boa suposição...



Analisando a Árvore

- O tamanho de um subproblema na profundidade *i* é $n/4^i$
  - *Onde* $i=log_4(n)$
  - Logo, a árvore tem $log_4(n)+1$ níveis $(0,1,2,3,...,\log_4 n)$
- O custo de cada nível:
  - Cada nível tem 3 vezes mais nós que o nível anterior
    - Nível *i*, número de número de nós é 3^i
  - Como os tamanhos dos subproblemas são reduzidos por um fator 4 a cada nível, cada nó no nível *i* tem custo $c(n/4^i)^2$
  - Assim, o custo total para um nível *i* será $3^i \cdot c(n/4^i)^2=(3/16)^i cn^2$

Analisando a Árvore

- O custo do último nível:
  - O último nível, profundidade $\log_4 n$, tem uma quantidade de nós igual a:

$$3^{\log_4 n} = n^{\log_4 3}$$

    - Cada qual contribuindo com o custo T(1), gerando um custo total para este nível de:

$$n^{\log_4 3} T(1) \rightarrow O\left(n^{\log_4 3}\right)$$

- Custo Total:

$$T(n) = cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n - 1} cn^2 + \Theta(n^{\log_4 3})$$

$$= \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3})$$

$$= \frac{(3/16)^{\log_4 n} - 1}{(3/16) - 1} cn^2 + \Theta(n^{\log_4 3})$$

Análise a Árvore

- Fazendo uso de algumas aproximações, o custo total pode ser expresso como:

$$T(n) = \sum_{i=0}^{\log_4 n-1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3})$$

$$< \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3})$$

$$= \frac{1}{1-(3/16)} cn^2 + \Theta(n^{\log_4 3})$$

$$= \frac{16}{13} cn^2 + \Theta(n^{\log_4 3})$$

$$= O(n^2) .$$

Analisando suposição

- O próximo passo é verificar, com o método da substituição, se a suposição é boa, isto é, se $T(n)=O(n^2)$ é um limite superior para a recorrência $T(n)=3T(\lfloor n/4 \rfloor)+\Theta(n^2)$
  - Quer-se mostrar que $T(n) \leq dn^2$, $d>0$. Usando a mesma cte $c>0$,

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

onde a última etapa é válida desde que $d \geq (16/13)c$.

Método Mestre

- O método mestre fornece uma "receita" para a resolução de recorrências na forma $T(n)=aT(n/b)+f(n)$,
  - onde $a>=1$ e $b>1$ (constantes) e $f(n)$ é uma função assintoticamente positiva
  - Esta recorrência descreve um problema de tamanho n dividido em a subproblemas de tamanho n/b
  - O custo em tempo para cada um dos subproblemas *é* $T(n/b)$ e o custo de dividir e combinar está na função $f(n)$.

Teorema Mestre

Teorema 4.1 (Teorema mestre)
Sejam $a \geq 1$ e $b > 1$ constantes, seja $f(n)$ uma função e seja $T(n)$ definida sobre os inteiros não negativos pela recorrência

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

onde interpretamos n/b com o significado de $\lfloor n/b \rfloor$ ou $\lceil n/b \rceil$. Então, $T(n)$ pode ser limitado assintoticamente como a seguir.

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

Usando o Teorema Mestre

- **Primeiro:** Verificar a que caso o problema se aplica ao teorema (1,2 ou 3)
- **Segundo:** Anotar a resposta
- Ex.:
  - **T(n)=9T(n/3)+n**
    - a=9, b=3, f(n)=n, e $n^{\log_b a} = n^{\log_3 9} = \Theta\left(n^2\right)$
    - Como, $f(n) = O\left(n^{\log_3 9 - \varepsilon}\right), \varepsilon = 1$
    - Estamos no caso 1 do teorema: $T(n)=\Theta(n^2)$

Exemplo do uso do teorema mestre

- Considere, $T(n) = T(2n/3) + 1,$
  - a=1, b=3/2, f(n)=1 e $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$
  - Assim, aplica-se o caso 2, pois $f(n) = \Theta(n^{\log_b a}) = \Theta(1)$
  - E a solução da recorrência é $T(n) = \Theta(\lg n)$

- Considere agora, $T(n) = 3T(n/4) + n \lg n$
  - a=3, b=4, $f(n) = n \lg n$ e $n^{\log_b a} = n^{\log_4 3} = O(n^{0,793})$
  - Como $f(n) = \Omega(n^{\log_4 3+\epsilon}),\ \epsilon \approx 0{,}2$
  - Aplica-se o caso 3, que para n suficientemente grande

    $$af(n/b) = 3(n/4)\lg(n/4) \leq (3/4)n \lg n = cf(n) \text{ para } c = 3/4.$$

  - E a solução para a recorrência é $T(n) = \Theta(n \lg n)$.

Contra-Exemplo

- Considere a recorrência,

$$T(n) = 2T(n/2) + n \lg n.$$

- Para este caso não é possível se aplicar o teorema mestre!
  - Observe que

$$f(n)/n^{\log_b a} = (n \lg n)/n = \lg n$$

  - Logo, f(n) é assintoticamente menor que n^{ϵ} para qualquer ϵ positiva.