Projeto e Análise de Algoritmos

Hamilton José Brumatto

Bacharelado em Ciência da Computação - UESC

29 de abril de 2016

Recorrências

Relações de Recorrência

- Relações de recorrência expressam fórmulas para tempo em algoritmos recursivos, basicamente algoritmos resultantes do método de Divisão e Conquista, que veremos em breve.
- Para conseguir encontrar uma fórmula que expresse o tempo da função será necessário entender como se resolve as fórmulas de relações de recorrência.

Exemplo de Algoritmo Recursivo

Entrada: Sequência ordenada de números e um valor
Saída: Posição do número na sequência ou 0 se não existir

Algoritmo BUSCA-BINARIA(A,x)
    $ini \leftarrow 1$; $fim \leftarrow comprimento[A]$
    $meio \leftarrow (ini + fim)/2$
    se $A[meio] = x$ **então**
        Retorna $meio$
    senão se $ini = fim$ **então**
        Retorna 0
    senão se $x > A[meio]$ **então**
        BUSCA-BINARIA$(A[meio + 1, fim], x)$
    senão
        BUSCA-BINARIA$(A[ini, meio], x)$

Neste algoritmo, A representa uma seqüência ordenada crescente de números, e x um valor que se deseja encontrar na seqüência.

Medindo a complexidade no **pior caso**

Vamos considerar o tempo das comparações neste algoritmo: $T(n)$

Linha	Código	Tempo
4	$A[meio] = x$	?
6	$ini = fim$	?
8	$x > A[meio]$	?
9 ou 11	**BUSCA-BINARIA**$(A[meio+1, fim], x)$ **BUSCA-BINARIA**$(A[ini, meio], x)$	?

Medindo a complexidade no **pior caso**

Vamos considerar o tempo das comparações neste algoritmo: $T(n)$

Linha	Código	Tempo
4	$A[meio] = x$	1
6	$ini = fim$	?
8	$x > A[meio]$	?
9 ou 11	**BUSCA-BINARIA**$(A[meio+1, fim], x)$	?
	BUSCA-BINARIA$(A[ini, meio], x)$	

Medindo a complexidade no **pior caso**

Vamos considerar o tempo das comparações neste algoritmo: $T(n)$

Linha	Código	Tempo
4	$A[meio] = x$	1
6	$ini = fim$	1
8	$x > A[meio]$	?
9 ou 11	**BUSCA-BINARIA**$(A[meio+1, fim], x)$	?
	BUSCA-BINARIA$(A[ini, meio], x)$	

Medindo a complexidade no **pior caso**

Vamos considerar o tempo das comparações neste algoritmo: $T(n)$

Linha	Código	Tempo
4	$A[meio] = x$	1
6	$ini = fim$	1
8	$x > A[meio]$	1
9 ou 11	**BUSCA-BINARIA**$(A[meio + 1, fim], x)$ **BUSCA-BINARIA**$(A[ini, meio], x)$	?

Medindo a complexidade no **pior caso**

Vamos considerar o tempo das comparações neste algoritmo: $T(n)$

Linha	Código	Tempo
4	$A[meio] = x$	1
6	$ini = fim$	1
8	$x > A[meio]$	1
9 ou 11	**BUSCA-BINARIA**$(A[meio + 1, fim], x)$ **BUSCA-BINARIA**$(A[ini, meio], x)$	$T(n/2)$

Medindo a complexidade no **pior caso**

Vamos considerar o tempo das comparações neste algoritmo: $T(n)$

Linha	Código	Tempo
4	$A[meio] = x$	1
6	$ini = fim$	1
8	$x > A[meio]$	1
9 ou 11	**BUSCA-BINARIA**$(A[meio+1, fim], x)$ **BUSCA-BINARIA**$(A[ini, meio], x)$	$T(n/2)$

Portanto: $T(n) = T(n/2) + 3$
Observe que: $T(1) = 2$

Relação de recorrência encontrada?

- Achamos uma relação de recorrência para o tempo do algoritmo:

$$\begin{aligned} T(1) &= 2 \\ T(n) &= T(n/2) + 3; \text{ para } n \geqslant 2 \end{aligned}$$

- Queremos resolver a recorrência, e isto significa econtrar uma fórmula fechada para $T(n)$.
- Queremos encontrar um valor assintótico, portanto não precisamos achar uma solução exata, apenas uma solução $f(n)$ tal que $T(n) \in O(f(n))$.
- Por que não achamos $T(n) \in \Theta(f(n))$?

Relação de recorrência encontrada?

- Achamos uma relação de recorrência para o tempo do algoritmo:

$$
\begin{aligned}
T(1) &= 2 \\
T(n) &= T(n/2) + 3;\ \text{para } n \geqslant 2
\end{aligned}
$$

- Queremos resolver a recorrência, e isto significa econtrar uma fórmula fechada para $T(n)$.
- Queremos encontrar um valor assintótico, portanto não precisamos achar uma solução exata, apenas uma solução $f(n)$ tal que $T(n) \in O(f(n))$.
- Por que não achamos $T(n) \in \Theta(f(n))$?

Métodos para resolver recorrências

Alguns métodos que podem ser utilizados:

- Substituição
- Iteração
- Árvore de recorrência

Também é possível encontrar um resultado bem mais geral que permite resolver várias recorrências: o Teorema do Mestre

Resolvendo pelo método da substituição

A proposta básica é:

- Adivinhar qual é a solução;
- Provar por indução que a solução é válida.

O problema deste método é que nem sempre é fácil de identificar qual é a solução, é preciso prática para conseguir visualizar uma possível solução.

Vamos resolver a recorrência

A recorrência é:

$$\begin{aligned} T(1) &= 2 \\ T(n) &= T(n/2) + 3; \text{ para } n \geqslant 2 \end{aligned}$$

O meu chute é que: $T(n) \in O(\log n)$.
Ou, para ser mais exato, $T(n) \leqslant 5 \log n$

Vamos provar: Passo da Indução

$$
\begin{aligned}
T(n) &= T(n/2) + 3 \\
&\leqslant 5\log\frac{n}{2} + 3 \text{ (Hipótese da Indução)} \\
&\leqslant 5\log n - 5log2 + 3 \\
&\leqslant 5\log n - 2 \\
&\leqslant 5\log n
\end{aligned}
$$

Beleza! o passo está provado!

Vamos provar: Passo da Indução

$$\begin{aligned}
T(n) &= T(n/2) + 3 \\
&\leqslant 5 \log \frac{n}{2} + 3 \text{ (Hipótese da Indução)} \\
&\leqslant 5 \log n - 5 log 2 + 3 \\
&\leqslant 5 \log n - 2 \\
&\leqslant 5 \log n
\end{aligned}$$

Beleza! o passo está provado!

Precisamos provar a base da indução

- $T(1) = 2$, mas $5 \log 1 = 0$, a base da indução não funciona?

Devemos lembrar que pela definição da classe $O(\ )$, é preciso provar que $T(n) \leqslant 5 \log n$ para $n \geqslant n_0$, para algum valor de n_0.

Que tal tomarmos como base $n_0 = 2$?.

- $T(2) = T(1) + 3 = 5 \leqslant 5 \log 2 = 5$

Provado!

O que mais no método da indução?

- Vamos supor que no caso anterior, $T(1) = 10$. Como resolver?
- Teríamos: $T(2) = T(1) + 3 = 13 \nleqslant 5 \log 2$
- Neste caso poderíamos ter suposto $T(n) \leqslant 13 \log n$ e daria certo. Lembramos que c e n_0 podem ser quaisquer na classe $O(\ )$.
- De uma forma geral, trabalhando com um termo genérico c ($T(n) \leqslant c \log n$), tanto no *passo da indução*, quanto na *base da indução*, é possível escolher c e n_0 de maneira conveniente.
- c é aquele que atende às duas condições, n_0 geralmente é obtido na base da indução.

Revendo a solução de uma forma mais geral

$$\begin{aligned} T(n) &= T(n/2) + 3 \\ &\leqslant c \log \frac{n}{2} + 3 \\ &\leqslant c \log n - clog 2 + 3 \\ &\leqslant c \log n - (c - 3) \\ &\leqslant c \log n \end{aligned}$$

Para tanto é necessário que $c - 3 \geqslant 0$, ou seja, $c \geqslant 3$.

Agora trabalhando também com a base

- $T(2) = T(1) + 3 = 5 \leqslant c \log 2 = c$

Logo, $c \geqslant 5$, portanto, $c = 5$ é suficiente.

Como discutimos anteriormente, $n_0 = 2$, e provamos que $T(n) \in O(\log n)$.

Completando o método

- Observe que não provamos que $T(n) \in \Theta(\log n)$ (por quê?).
- Para isto é necessário provar que a recorrência acima também é válida para $T(n) \in \Omega(n)$.
- O método é o mesmo.

Fica como desafio provar que a recorrência é $\Omega(\log n)$.

Outros exemplos de prova pelo método da substituição

- Vamos considerar a recorrência:

$$
\begin{aligned}
T(1) &= 1 \\
T(n) &= T(n/2) + T(n/2) + n
\end{aligned}
$$

Precisamos provar que: $T(n) \in \Theta(n \log n)$.

Provando a classe $O(\ )$: Passo da indução

$$\begin{aligned}
T(n) &= T(n/2) + T(n/2) + n \\
&\leqslant c\left[\frac{n}{2}\right]\log\left[\frac{n}{2}\right] + c\left[\frac{n}{2}\right]\log\left[\frac{n}{2}\right] + n \\
&\leqslant cn\log\left[\frac{n}{2}\right] + n \\
&\leqslant cn\log n - (c-1)n \\
&\leqslant cn\log n \text{ para } c > 1
\end{aligned}$$

Precisamos provar a base da indução

Provando a classe $O(\ )$: Base da indução

Vamos considerar novamente $n_0 = 2$

$$\begin{aligned} T(2) &= T(1) + T(1) + 2 = 4 \\ &\leqslant c 2 \log 2 = 2c \end{aligned}$$

Precisamos simplesmente $c \geqslant 2$, ou seja $c = 2$ é suficiente, para $n_0 = 2$ provamos que $T(n) \in O(n \log n)$

Provando a classe $\Omega(\ )$: Passo da indução

$$
\begin{aligned}
T(n) &= T(n/2) + T(n/2) + n \\
&\geqslant c\left[\frac{n}{2}\right]\log\left[\frac{n}{2}\right] + c\left[\frac{n}{2}\right]\log\left[\frac{n}{2}\right] + n \\
&\geqslant cn\log\left[\frac{n}{2}\right] + n \\
&\geqslant cn\log n + (1-c)n \\
&\geqslant cn\log n \text{ para } 0 < c < 1
\end{aligned}
$$

Precisamos provar a base da indução

Provando a classe $\Omega(\ )$: Base da indução

Vamos considerar novamente $n_0 = 2$

$$\begin{aligned} T(2) &= T(1) + T(1) + 2 = 4 \\ &\geqslant c2\log 2 = 2c \end{aligned}$$

Para a base precisamos simplesmente $c \leqslant 2$, ou seja $c = 1/2$ é suficiente para ambas condições, e com $n_0 = 2$ provamos que $T(n) \in \Omega(n \log n)$.
Com isto, provamos também que $T(n) \in \Theta(n \log n)$

Prova incorreta usando o método

Vamos provar que o exemplo anterior é $O(n)$,
Precisamos provar que $T(n) \leqslant cn$:

$$\begin{aligned}
T(n) &= T(n/2) + T(n/2) + n \\
&\leqslant c\left[\frac{n}{2}\right] + c\left[\frac{n}{2}\right] + n \\
&\leqslant cn + n, \text{ logo:} \\
T(n) &= O(n)
\end{aligned}$$

Onde está o erro?

Às vezes é necessário um ajuste além do valor do c

Veja o exemplo para a recorrência:

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 2T(n/2) + 1 \end{aligned}$$

Queremos provar que $T(n) \in O(n)$

Intuitivamente, faremos $T(n) \leqslant cn$, como chute inicial.

Provando o passo da indução:

$$\begin{aligned} T(n) &= 2T(n/2)+1 \\ &\leqslant 2c\left[\frac{n}{2}\right]+1 \\ &\leqslant cn+1 \end{aligned}$$

E agora, não conseguimos $T(n) \leqslant cn$. Apareceu um termo positivo, precisávamos de algo negativo para sumir com o termo.

Corrigindo a hipótese da indução

Vamos mostrar que $T(n) \leqslant cn - b$, para $b > 0$, que é uma hipótese de indução mais forte.

$$\begin{aligned} T(n) &= 2T(n/2) + 1 \\ &\leqslant 2c\left[\frac{n}{2}\right] - 2b + 1 \\ &\leqslant cn - b - (b-1) \\ &\leqslant cn - b, \text{ para } b \geqslant 1 \end{aligned}$$

Desta vez deu certo.

O Método da Iteração

- Devemos desdobrar a recorrência na recursão, pegando os termos a cada iteração até que atingimos a base.
- No final teremos uma somatória de termos que dependem apenas de n e do valor definido para base.
- Precisa fazer mais contas, principalmente somatórias.

Aplicando o método da iteração:

Considere a recorrência inicial:

$$\begin{aligned} T(1) &= 2 \\ T(n) &= T(n/2) + 3 \end{aligned}$$

Iterando, teremos:

$$\begin{aligned} T(n) &= 3 + T(n/2) \\ &= 3 + (3 + T(n/4)) = 6 + T(n/4) \\ &= 6 + (3 + T(n/8)) = 9 + T(n/8) \\ \ldots \\ &= 3i + T(n/2^i) \\ \ldots \end{aligned}$$

Quando parar?

Calculando a recorrência

O *i-ésimo* termo da série termina quando chegarmos à base ($T(1)$), isto é, $n/2^i = 1$, também, $n = 2^i$ o que nos dá como último termo da série $i = \log n$

$$
\begin{aligned}
T(n) &= 3\log n + T\left(\frac{n}{2^{\log n}}\right) \\
T(n) &= 3\log n + T(n/n) \\
T(n) &= 3\log n + 2
\end{aligned}
$$

Precisamos provar a recorrência, novamente, a prova é por indução.

Provando a recorrência

Passo:

$$
\begin{aligned}
T(n) &= T(n/2) + 3 \\
&= (3\log(n/2) + 2) + 3 \\
&= 3\log n - 3\log 2 + 5 \\
&= 3\log n + 2
\end{aligned}
$$

Para base: $T(1) = 3\log 1 + 2$, ou seja, $T(1) = 2$.

Outro exemplo para o método da Iteração

Entrada: Sequência qualquer de números
Saída: Sequência com os mesmos números de forma ordenada

Algoritmo SELECT_SORT(A)
  se tamanho[A] $= 1$ **então**
    Retorna A
  senão
    menor $\leftarrow 1$
    para $i \leftarrow 2$ **até** tamanho[A] **faça**
      se $A[menor] > A[i]$ **então** ▷ Comparando Elementos
        $menor \leftarrow i$
    $A[1] \Leftrightarrow A[menor]$ ▷ Troca de Elementos
    Retorna $A[1, \text{SELECT\_SORT}(A[2, \text{tamanho}[A]])]$

Definindo a recorrência

Veja a recorrência para o select sort, vamos considerar o número de comparações:
Nos algoritmos de ordenação normalmente são contados os números de comparações entre elementos da sequência

$$\begin{aligned} T(1) &= 0 \\ T(n) &= T(n-1) + n - 1 \end{aligned}$$

Aplicando a iteração:

$$\begin{aligned} T(n) &= T(n-1) + n - 1 \\ T(n) &= T(n-2) + [(n-1) - 1] + n - 1 = T(n-2) + 2n - 1 - 2 \\ T(n) &= T(n-3) + [(n-2) - 1] + 2n - 1 - 2 = T(n-3) + 3n - 1 - 2 - 3 \\ &\dots \\ T(n) &= T(n-i) + in - \sum_{j=1}^{i} j \\ &\dots \end{aligned}$$

Calculando a recorrência

A recorrência termina para $i = n - 1$

$$\begin{aligned} T(n) &= T(1) + (n-1)n - \sum_{j=1}^{n-1} j \\ &= 0 + n^2 - n - \frac{(n)(n-1)}{2} \\ &= n^2 - \frac{n^2}{2} - n + \frac{n}{2} \\ &= \frac{1}{2}(n^2 - n) \end{aligned}$$

Vamos provar isto?

Provando a recorrência do Select Sort

Passo:

$$
\begin{aligned}
T(n) &= T(n-1) + n - 1 \\
T(n) &= \frac{1}{2}[(n-1)^2 - (n-1)] + n - 1 \\
T(n) &= \frac{1}{2}(n^2 - 2n + 1 - n + 1 + 2n - 2) \\
T(n) &= \frac{1}{2}(n^2 - n)
\end{aligned}
$$

Base: $T(1) = \frac{1}{2}(1^2 - 1) = 0$

Provada a recorrência, encontramos que $T(n) = \Theta(n^2)$

Árvore de Recorrência

- Aplica-se a iteração, mas é possível visualizar o que acontece quando a recorrência é iterada.
- É mais fácil organizar as contas.
- Será útil para recorrências de algoritmos de Divisão e Conquista.

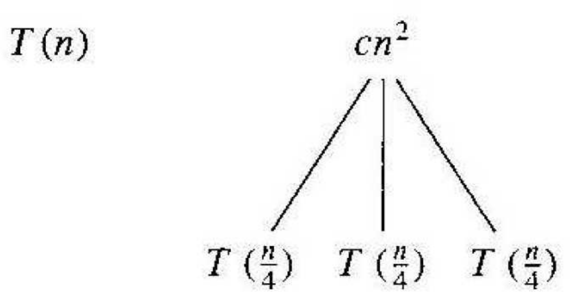
Aplicando o método

Vamos considerar a recorrência:

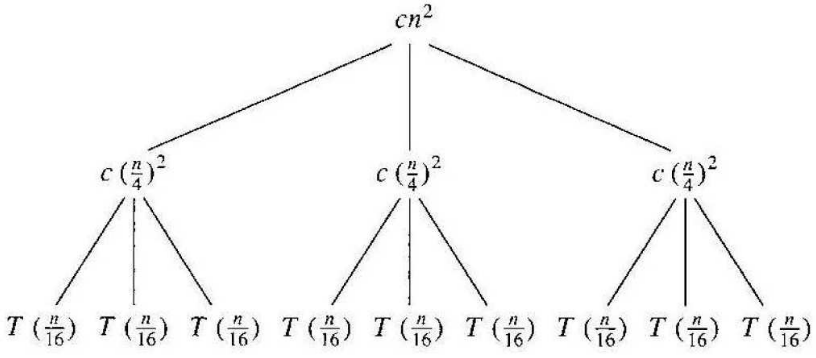
$$\begin{aligned} T(n) &= \Theta(1), \text{ para } n = 1, 2 \text{ e } 3 \\ T(n) &= 3T(n/4) + cn^2, \text{ para } n \geqslant 4. \end{aligned}$$

Na recorrência acima, $c > 0$ é uma constante.
A notação $T(n) = \Theta(1)$ indica que $T(n)$ é uma constante.
Para simplificar, vamos supor que n somente assume valores que são potências de 4, ou seja, $n = 1$, ou $n = 4, 16, ..., 4^i, ...$

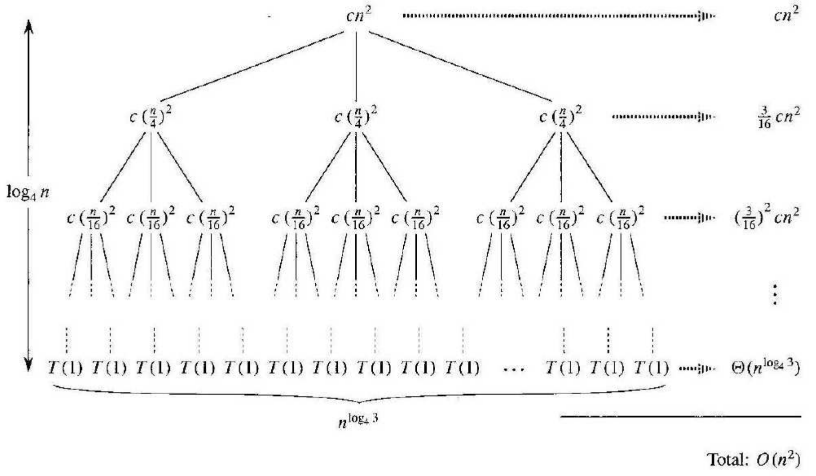
Primeira iteração



Segunda iteração



… última iteração



Contando os "galhos"

- 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 $\Theta(n^{\log_4 3})$

Somando os níveis

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

Isto pois, $\sum_{i=1}^{\infty} q^i = \frac{1}{1-q}$. Concluímos que $T(n) \in O(n^2)$

Precisamos verificar o resultado: $T(n) \in O(n^2)$

Nosso chute será: $T(n) \leqslant dn^2$ (método de substituição)

$$\begin{aligned} T(n) &= 3T(n/4) + cn^2 \\ &\leqslant 3d(n/4)^2 + cn^2 \\ &\leqslant \frac{3}{16}dn^2 + cn^2 \\ &\leqslant dn^2, \text{ para } d \geqslant (16/13)c \end{aligned}$$

Uma prova completa precisa levar em consideração a base!

Revendo a Árvore de Recorrências

- 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 às chamadas recursivas.
- Somando ao longo da coluna determina-se a solução da recorrência.

Teorema do Mestre

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

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

- O caso *base* é omitido na definição e convenciona-se que é uma constante para valores pequenos.
- O Teorema do Mestre **não** fornece a resposta para **todas** as recorrências da forma acima.

Theorem ((CLRS) Teorema Mestre)

Sejam $a \geqslant 1$ *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* $f(n) \in O(n^{\log_b a-\varepsilon})$ *para alguma constante* $\varepsilon > 0$, *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* $f(n) \in \Omega(n^{\log_b a+\varepsilon})$, *para alguma constante* $\varepsilon > 0$ *e se* $af(n/b) \leqslant cf(n)$, *para alguma constante* $c < 1$ *e para* n *suficientemente grande, então* $T(n) \in \Theta(f(n))$

Exemplos onde o Teorema Mestre pode ser aplicado:

- Caso 1

$$\begin{aligned} T(n) &= 9T(n/3) + n \\ T(n) &= 4T(n/2) + n \log n \end{aligned}$$

- Caso 2

$$\begin{aligned} T(n) &= T(2n/3) + 1 \\ T(n) &= 2T(n/2) + (n + \log n) \end{aligned}$$

- Caso 3

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

Exemplos onde o Teorema Mestre **não** pode ser aplicado:

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

Reduzindo o Teorema Mestre para funções polinomiais

Theorem

Sejam $a \geqslant 1$ e $b > 1$ constantes, seja $p(n)$ um polinômio com $p(n) \in \Theta(n^k)$ e seja $T(n)$ definida para os inteiros não negativos pela relação de recorrência:

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

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

1. *Se $k < \log_b a$ então $T(n) \in \Theta(n^{\log_b a})$*
2. *Se $k = \log_b a$ então $T(n) \in \Theta(n^k \log n)$*
3. *Se $k > \log_b a$ então $T(n) \in \Theta(n^k)$*

Atividades baseadas no CLRS

- Leitura: Capítulo 4 (não precisa o 4.4)
- Exercícios: 4.1-1, 4.2-1, 4.2-2, 4.2-3, 4.3-1, 4.3-2, 4.3-3
- Problemas: 4-1, 4-4, .

Resolva a 4ª Lista de Exercícios