Análise e Projeto de Algoritmos - Recorrências -

Prof^a Jerusa Marchi

Departamento de Informática e Estatística

INE/CTC/UFSC

Recorrências

- Muitos algoritmos são recursivos em essência
  - Para resolver um problema, tais algoritmos chamam a si próprios uma ou mais vezes, passando como parâmetro, parte do problema a ser resolvido
- Quando um algoritmo apresenta uma chamada para si próprio, seu tempo de execução pode ser descrito como uma **relação de recorrência**
  - Expressão recursiva que descreve a função em termos do seu próprio valor em entradas menores

Recorrências

- O termo $T(n)$ é especificado como uma função dos termos anteriores $T(1)$, $T(2)$, ..., $T(n-1)$

$$T(n) = \begin{cases} 1 & \text{se} \quad n = 1 \\ T(n/3) + n & \text{caso contrário} \end{cases}$$

- O problema reduz o tamanho da entrada em $2/3$ a cada chamada e toma tempo linear para efetuar os demais passos
- usa-se de ferramentas matemáticas para resolver a recorrência (encontrar a forma fechada) e prover limites de desempenho do algoritmo
  - Para a recorrência acima, a fórmula fechada é $\frac{3n-1}{2}$ e o limite assintótico é $\Theta(n)$

Antes de começar...

- Na prática, certos detalhes técnicos são omitidos quando da construção e solução de recorrências
  - considerar n sempre como um inteiro
  - considerar n como uma potência do divisor da equação de recorrência (se houver)
  - na definição da recorrência são, em geral, omitidos tetos, pisos e condições de término
    - assume-se que $T(n)$ é constante para um n suficientemente pequeno, ou seja $T(1) = \Theta(1)$
    - apesar do valor de $T(1)$ poder variar, e modificar a solução da recorrência, a solução não mudará por mais do que um fator constante e portanto a ordem de crescimento assintótico não mudará.

Recorrências

- Há três métodos principais para resolução de recorrências:
  - Método Iterativo (Expansão Telescópica)
    - Converta a recorrência em uma soma de termos e encontre limites para a soma
  - Método da árvore de recursão
    - Expanda a recorrência utilizando a notação em árvore e verifique o custo por nível
  - Teorema Mestre
    - aplicação do teorema que traz soluções para recorrências da forma $T(n) = aT(n/b) + f(n)$

Método Iterativo - Exemplo 1

- Considere a seguinte recorrência: $T(n) = T(n/2) + 1$
- assumindo n como uma potência de 2, tal que $n = 2^k$ e considerando $T(n/2^k) = T(1) = 1$, a expansão da relação de recorrência é:

$$
\begin{aligned}
T(n) &= T(n/2) + 1 \\
&= (T(n/4) + 1) + 1 \\
&= ((T(n/8) + 1) + 1) + 1 \\
&= \cdots \\
&= \underbrace{(...(T(n/2^k)) + \cdots + 1) + 1}_{k\ parcelas} \\
&= \sum_{i=1}^{k} 1 \qquad (k = lg\ n) \\
&= O(lg\ n)
\end{aligned}
$$

Método Iterativo - Exemplos 2 e 3

- e para $T(n) = T(n/2) + c$?
- e para $T(n) = T(n/2) + n$?

Método Iterativo - Exemplo 4

- Considere a seguinte recorrência: $T(n) = T(n/3) + n$
- assumindo n como uma potência de 3, tal que $n = 3^k$, tem-se $T(n/3^k) = T(1) = 1$, a expansão da relação de recorrência é:

$$
\begin{aligned}
T(n) &= T(n/3) + n \\
&= (T(n/9) + n/3) + n \\
&= ((T(n/27) + n/9) + n/3) + n \\
&= \cdots \\
&= (((\underbrace{T(n/3^k)}_{T(1)=1} + n/3^{k-1}) + n/3^{k-2}) + \cdots + n/3^1) + n \\
&= \textstyle\sum_{i=0}^{k} 3^i \qquad (k = log_3 n) \\
&= \frac{3^{log_3 n+1}-1}{3-1} = \frac{3^1 . \overbrace{3^{log_3 n}}^{3^{log_3 n}=n^{log_3 3}} -1}{3-1} \\
&= \frac{3n-1}{2} = \Theta(n)
\end{aligned}
$$

Método Iterativo - Exemplo 5

- Considere a seguinte recorrência: $T(n) = 2T(\frac{n}{2}) + n$
- assumindo n como uma potência de 2, tal que $n = 2^k$

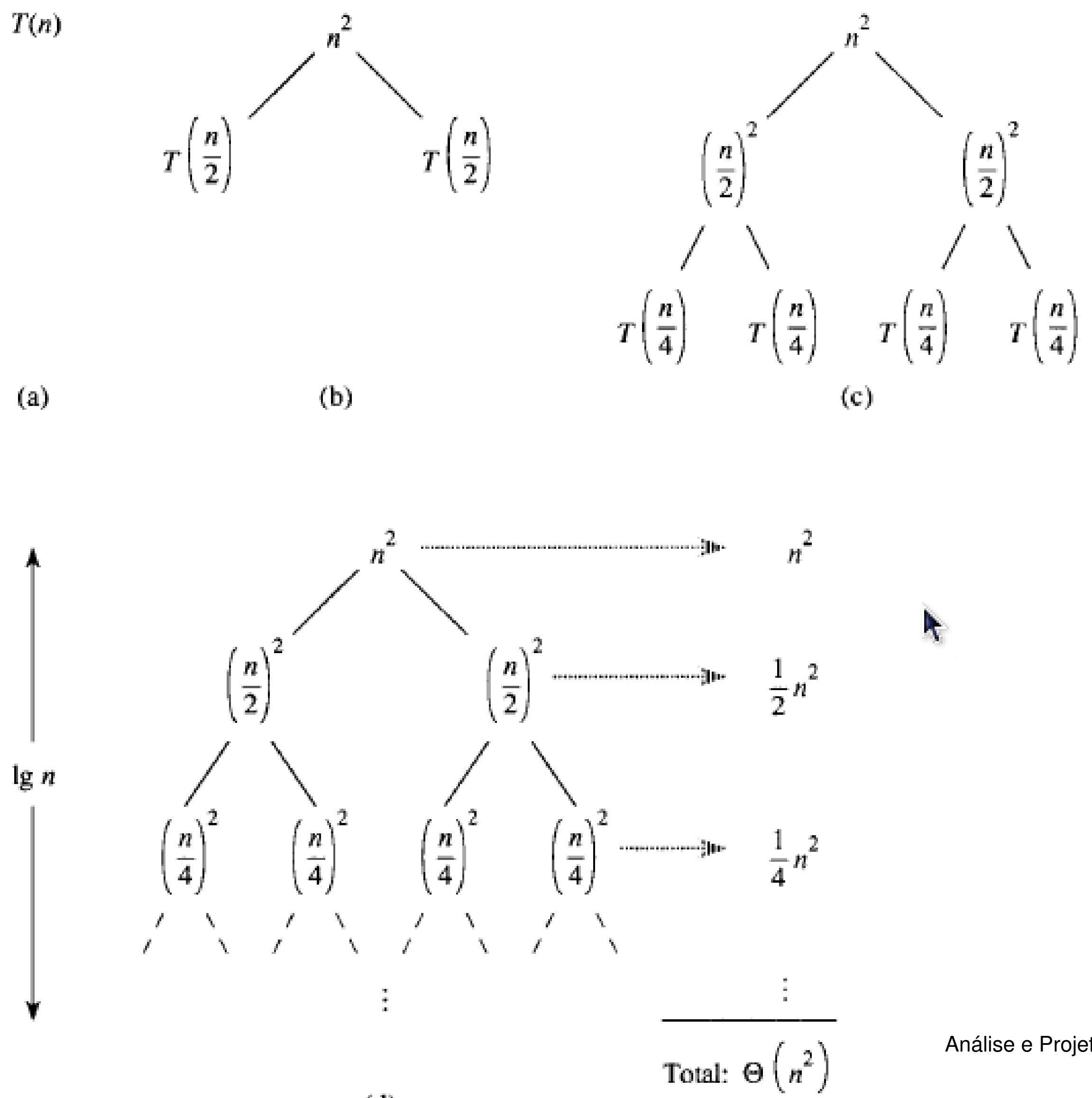
$$
\begin{aligned}
T(n) &= n + 2T(n/2) \\
&= n + 2(n/2 + 2T(n/4)) \\
&= n + 2(n/2 + 2(n/4 + 2T(n/8))) \\
&= \cdots \\
&= \underbrace{n + 2n/2 + 4n/4 + 8n/8 + \cdots + 2^k\ n/2^k}_{k\ parcelas} \\
&= n\ lg\ n \\
&= O(n\ lg\ n)
\end{aligned}
$$

Método da Árvore de Recursão

- Uma excelente forma de visualizar o que acontece quando a recorrência é iterada é utilizando **árvores de recorrência**
  - cada nó da árvore representa o custo de um único subproblema na função recursiva
  - a avaliação é feita somando os custos em cada nível da árvore, obtendo o custo dos níveis, e então, somando todos os custos dos níveis, obtem-se o custo total.

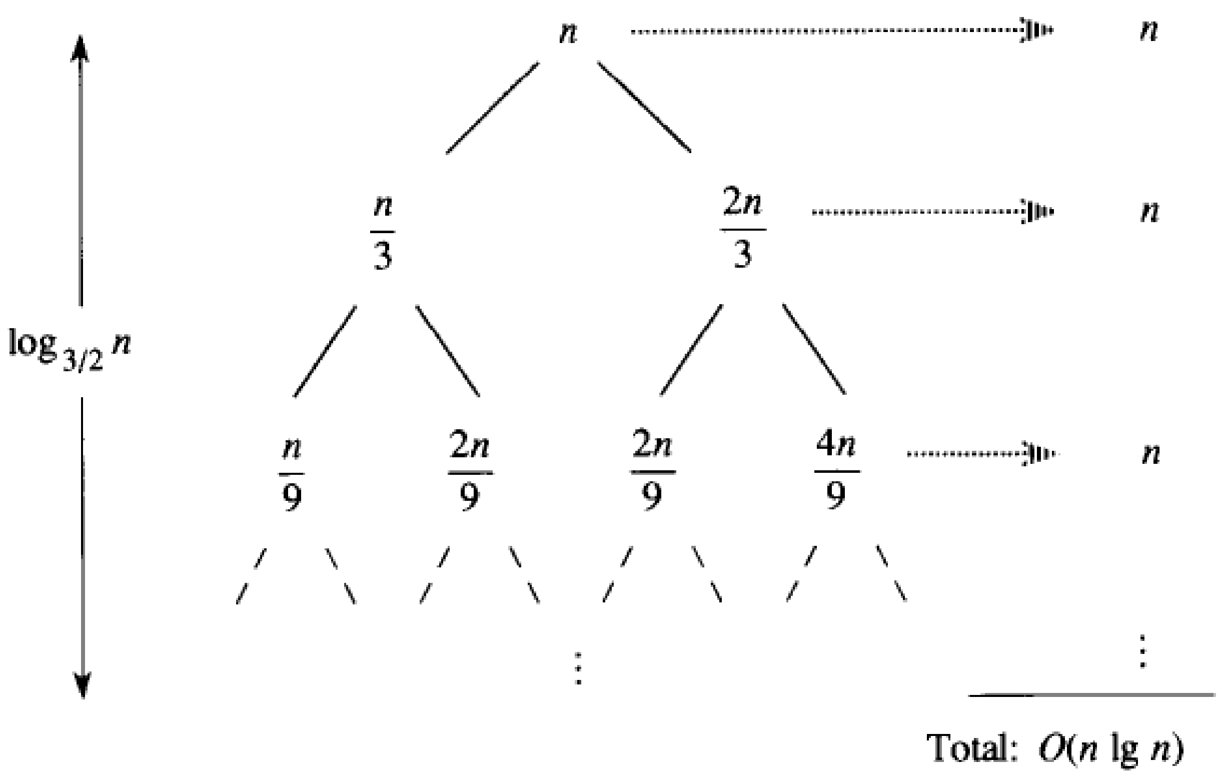
Método da Árvore de Recursão

- Exemplo 1: $T(n) = 2T(n/2) + n^2$



Método da Árvore de Recursão

- Exemplo 2: $T(n) = T(n/3) + T(2n/3) + n$



Teorema Mestre

- O teorema Mestre provê uma receita para resolver recorrências da forma
$$T(n) = aT(\frac{n}{b}) + f(n)$$
onde $a \geq 1$, $b > 1$ e $f(n)$ é uma função assintoticamente positiva
    - O problema de tamanho n é dividido em a subproblemas, cada um com tamanho n/b, onde a e b são constantes
    - Os a subproblemas podem ser resolvidos em tempo $T(n/b)$
    - $f(n)$ agrega os tempos de divisão do problema e combinação dos resultados

Teorema Mestre

Teorema: Sejam $a \geq 1$ e $b > 1$ constantes, seja $f(n)$ uma função, e $T(n)$ um função de recorrência definida em termos de inteiros não negativos

$$T(n) = aT(\frac{n}{b}) + f(n)$$

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

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 ser $af(n/b) \leq cf(n)$, para alguma constante $c \leq 1$ e para n suficientemente grande, então $T(n) = \Theta(f(n))$

Teorema Mestre

- O teorema compara $f(n)$ com $n^{log_b a}$ - a maior destas funções determina a solução da recorrência
  - se $n^{log_b a}$ é **polinomialmente** maior, então (caso 1) $T(n) = \Theta(n^{log_b a})$
  - se $f(n)$ é **polinomialmente** maior, então (caso 3) $T(n) = \Theta(f(n))$
  - se ambas possuem o mesmo tamanho, então multiplica-se por um fator logarítmico, e a solução é (caso 2) $T(n) = \Theta(f(n)\ lg\ n) = \Theta(n^{log_b a}\ lg\ n)$

Teorema Mestre

- Tecnicalidades
  - Caso 1: $f(n)$ precisa ser assintoticamente menor que $n^{log_b a}$ por um fator n^ϵ para alguma constante $\epsilon > 0$
  - Caso 3: $f(n)$ precisa ser assintoticamente maior que $n^{log_b a}$ e satisfazer a condição $af(n/b) \leq cf(n)$.
- Os 3 casos não cobrem todas as possibilidades para $f(n)$
  - gap entre os casos 1 e 2: $f(n)$ é menor que $n^{log_b a}$, mas não polinomialmente menor.
  - gap entre os casos 2 e 3: $f(n)$ é maior que $n^{log_b a}$, mas não polinomialmente maior.
  - $f(n)$ não respeita a condição $af(n/b) \leq cf(n)$

Nestes casos **NÃO** use o Teorema Mestre!

Teorema Mestre

- Exemplos:
  - $T(n) = 9T(n/3) + n$
    - $a = 9$, $b = 3$, $f(n) = n$ ou seja $f(n) = \Theta(n)$
    - $n^{log_b\ a} = n^{log_3\ 9} = \Theta(n^2)$ (>)
    - Assim, $f(n) = O(n^{log_3\ 9 - \epsilon})$, com $\epsilon = 1$ e portanto

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

Teorema Mestre

- Exemplos:
  - $T(n) = T(2n/3) + 1$
    - $a = 1$, $b = 3/2$, $f(n) = 1$ ou seja $f(n) = \Theta(1)$
    - $n^{log_b\ a} = n^{log_{\frac{3}{2}}\ 1} = n^0 = 1 = \Theta(1)$ (=)
    - Assim, $f(n) = \Theta(n^{log_{\frac{3}{2}}\ 1}) = \Theta(1)$, portanto

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

Teorema Mestre

- Exemplos:
  - $T(n) = 2T(n/2) + n\ lg\ n$
    - $a = 2$, $b = 2$, $f(n) = n\ lg\ n$ ou seja $f(n) = \Theta(n\ lg\ n)$
    - $n^{log_b\ a} = n^{lg_2} = n = \Theta(n)$ (<)
    - Cuidado!!! $n\ log\ n$ NÃO difere POLINOMIALMENTE de n
      - A razão $f(n)/n^{log_b\ a} = n\ lg\ n/n = lg\ n$ é assintoticamente menor que n^ϵ para qualquer constante ϵ
    - Portanto a recorrência cai no gap entre as condições 2 e 3.

Teorema Mestre

- Caso Particular (para um sub-conjunto expressivo de recorrências)

Teorema: Dada uma função de recorrência na forma

$$T(n) = aT(\frac{n}{b}) + cn^d)$$

onde $a, b \in \mathrm{N}$, $a > 1, b \geq 2$ e $c, d \in \mathrm{R}^+$

$$T(n) = \begin{cases} \Theta(n^{log_b a}) & \text{se} \quad a > b^d \\ \Theta(n^d \; log \; n) & \text{se} \quad a = b^d \\ \Theta(n^d) & \text{se} \quad a \leq b^d \end{cases}$$

Bibliografia

- Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C., *Introduction to Algorithms*, 2009.
- Ziviani, N.C.; *Projeto e Análise de algoritmos com implementações em Java e C++*, 2007.
- Rezende, P.J., Material sobre Recorrências (http://www.dcc.unicamp.br/∼rezende/ensino/mo417/2010s2/notas/recorrencias−notas.pdf)
- Papadimitriou, C.H/ Dasgupta, S.; Vazirani, U.V., *Algorithms*, McGraw-Hill, 2006.