PANC: Projeto e Análise de Algoritmos

Aula 08: Resolução de Recorrências - Teorema Mestre

Breno Lisi Romano

http://sites.google.com/site/blromano

Instituto Federal de São Paulo – IFSP São João da Boa Vista
Bacharelado em Ciência da Computação – 3º Semestre

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus São João da Boa Vista

Sumário

- Revisão de Conteúdo
- Resolução de Recorrências:
  - Teorema Mestre
- Exercícios Práticos

Recapitulando... (1)

- **Recorrências:**
  - **Uma recorrência é uma equação ou desigualdade que descreve uma função em termos de seu próprio valor em entradas menores**
  - **Aplicação e Resolução:**
    - A **complexidade de algoritmos recursivos** pode ser frequentemente descrita através de **recorrências**
    - Geralmente, recorremos ao **Teorema Mestre** para **resolver** estas **recorrências**
    - Em **casos** em que o **Teorema Mestre não se aplica**, a **recorrência** deve ser **resolvida** de **outras maneiras**
    - **Resolver** uma **recorrência significa eliminar** as **referências** que **ela faz a si mesma**
    - **Os métodos mais comuns** para resolução de recorrências são o método de **substituição**, método da **interação** o método de **árvore de recursão** e o **teorema mestre**

Recapitulando... (2)

- Existem alguns **métodos para resolver recorrências**, isto é, para **obter limites assintóticos “Θ” ou “O” para a solução**. Os principais são:
  - **Método de substituição:** arrisca-se um palpite para um limite e então utiliza-se da indução matemática para provar que nosso palpite estava correto
  - **Método da Iteração:** expandir (iterar) a recorrência e escrevê-la como uma somatória de termos que dependem apenas de n e das condições iniciais
  - **Método da árvore de recursão:** converte a recorrência em uma árvore cujos nós representam os custos envolvidos em vários níveis da recursão
    - Usa-se técnicas para limitar somatórios, resolvendo-se a recorrência
  - **Método do Teorema mestre**: dá limites para recorrências da forma **$T(n) = aT(n/b) + f(n)$**, onde $a \geq 1$, $b > 1$ e $f(n)$ é uma função dada. Tais recorrências ocorrem frequentemente.
    - Uma recorrência da forma da equação apresentada caracteriza um algoritmo de divisão e conquista que cria **a subproblemas**, cada um **com 1/b do tamanho do problema original** e no qual as etapas de divisão e conquista, juntas, levam o **tempo f(n)**

Recapitulando... (3)

- O **método de substituição** para resolver recorrências envolve duas etapas:
  1. **Arriscar um palpite** para a forma da solução
  2. **Usar indução** para **determinar as constantes** e mostrar que a solução funciona

- **Substitui-se a função pela solução suposta na primeira etapa quando aplica-se a hipótese indutiva a valores menores**; daí o nome “método de substituição”

- **Método é poderoso**, mas tem que adivinhar a forma da resposta para aplicá-lo

- **Aplica-se** este método **em casos que é fácil pressupor** a forma de resposta

Recapitulando... (4)

- **Método da Interação:**
  - Não é necessário adivinhar a resposta para a recorrência
  - Precisa-se fazer mais conta → Mão na massa
  - **Ideia**: expandir (iterar) a recorrência e escrevê-la como uma somatória de termos que dependem apenas de n e das condições iniciais
    - Existem **alguns truques** que podem ser aplicados:
      - **Procurar por algum padrão ao expandir uma recorrência**, como alguma recorrência básica
      - **Realizar manipulações algébricas**, como troca de variáveis ou divisão da recorrência, que favoreçam a resolução
  - Para tanto, é necessário ter **conhecimento algébrico**, de recorrências básicas e uma **dose de “maldade”**

Recapitulando... (5)

- Método da **árvore de recursão:** apresenta uma **forma bem intuitiva** para a **análise de complexidade** de algoritmos **recursivos**
  - Numa árvore de recursão, **cada nó representa o custo de um único subproblema** da respectiva chamada recursiva
  - **Somam-se os custos de todos os nós** de um mesmo nível, para obter o custo daquele nível
  - **Somam-se os custos de todos os níveis** para obter o custo da árvore
- Permite visualizar melhor o que acontece quando a recorrência é iterada
- É mais fácil organizar as contas
- **Útil** para recorrências de **algoritmos de divisão e conquista**

Teorema Mestre (1)

- O Teorema Mestre fornece "receitas" para resolver recorrências da forma

  T (n) = ***a*** T(n/**b**) + f(n)

  - em que a ≥ 1 e b > 1 são constantes positivas e f(n) é uma função assintoticamente positiva

- A recorrência acima descreve a complexidade de tempo de um algoritmo que **divide o problema original** de tamanho n em **a subproblemas** de **tamanho n/b**
- Os **a subproblemas** são **resolvidos** em **tempo T (n/b)** cada um
- A função **f(n) engloba o custo de dividir o problema original** e, eventualmente, combinar os resultados dos subproblemas
- Note que n/b pode não ser inteiro, entretanto, o termo T (n/b) pode ser substituído por $T(\lceil n/b \rceil)$ ou $T(\lfloor n/b \rfloor)$ sem afetar o comportamento assintótico

Teorema Mestre (2)

- **Usando o Teorema Mestre:**
  - O Teorema Mestre **enumera três casos** em que se torna fácil resolver recorrências
  - Note que **não se está interessado em obter a forma fechada para a recorrência**, mas sim em seu comportamento assintótico, de maneira direta
  - Isto é o **contrário de quando empregamos o método de substituição**, no qual **encontramos a forma fechada** e depois **analisamos seu comportamento assintótico**

Teorema Mestre (3)

- **Teorema:**

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

$$T(n) = aT(n/b) + f(n), \tag{1}$$

em que interpretamos $T(n/b)$ como $T(\lceil n/b \rceil)$ ou $T(\lfloor n/b \rfloor)$.

Então $T(n)$ possui os seguintes limites assintóticos:

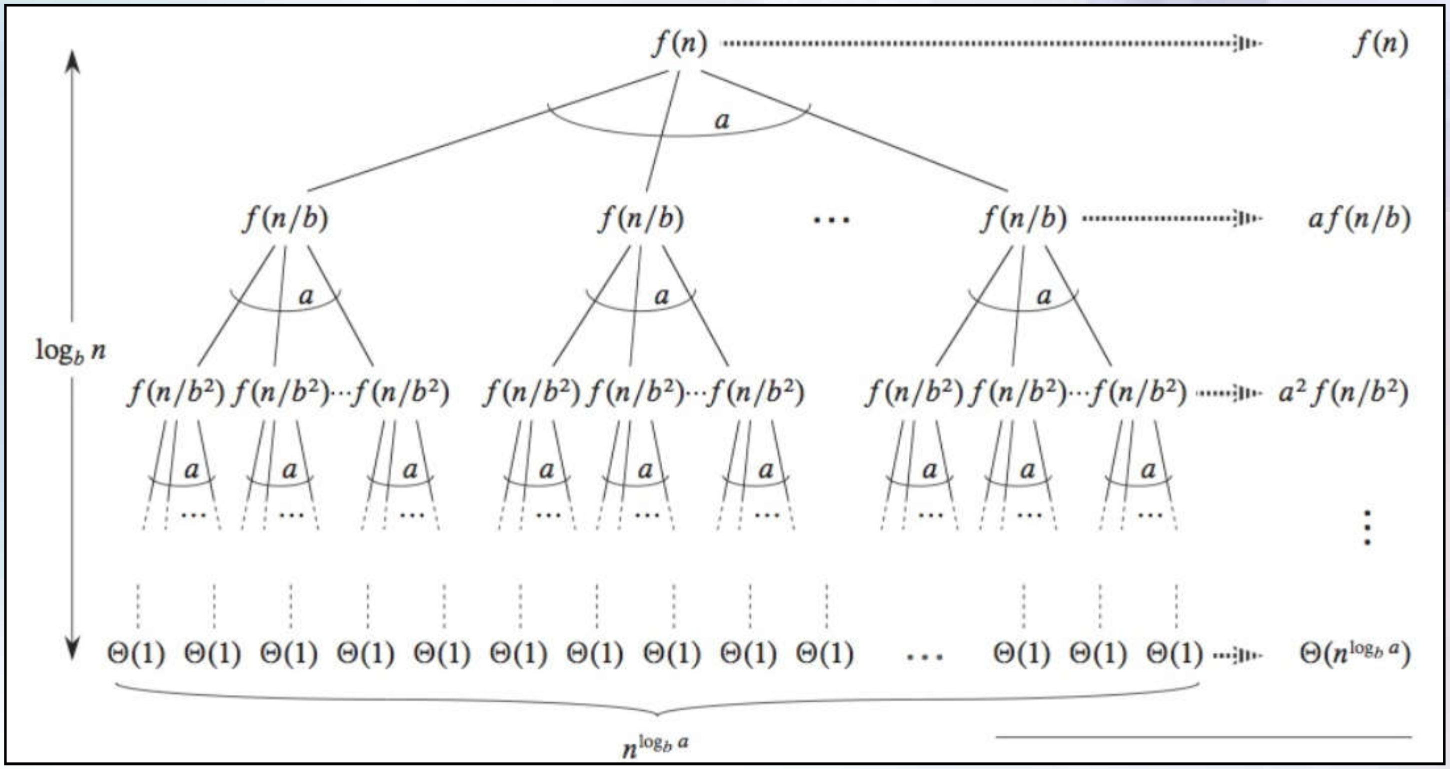
1. Se $f(n) = O(n^{log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) = \Theta(n^{log_b a})$.
2. Se $f(n) = \Theta(n^{log_b a})$, então $T(n) = \Theta(n^{log_b a} log\ n)$.
3. Se $f(n) = \Omega(n^{log_b a + \epsilon})$ para alguma constante $\epsilon > 0$ e se $af(n/b) \leq cf(n)$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

Teorema Mestre (4)

- **Porquê três casos:**
  - Nos 03 casos, comparamos uma função **f(n)** com a função $\mathbf{n^{\log_b a}}$
  - A **complexidade da recorrência é determinada pela maior das duas**:
    1. No **primeiro caso**, a função $\mathbf{n^{\log_b a}}$ **é maior**, portanto, $\mathbf{T(n) = \Theta(n^{\log_b a})}$
    2. No **terceiro caso**, a **função f(n) é maior**, portanto, $\mathbf{T(n) = \Theta(f(n))}$
    3. No **segundo caso**, as **duas funções têm o mesmo "tamanho"**. A **solução é multiplicada por um fator logarítmico**, portanto**,** $\mathbf{T(n) = \Theta(n^{\log_b a}. \log n) = \Theta(f(n). \log n)}$

Teorema Mestre (5)

- **Entendendo:**



- $n^{\log_b a}$ é o número de folhas da árvore de recursão a-ária gerada por T(n) = a T(n/b) + f(n)
- No caso 2, multiplicamos f(n) (equivalente a $n^{\log_b a}$) por log n para indicar que o custo f(n) se dá a cada nível da árvore mostrada anteriormente

Teorema Mestre (6)

- **Função Polinomialmente Maior**
  - Uma função f(n) é polinomialmente maior que outra função g(n) se pudermos achar algum $\epsilon > 0$ tal que $\frac{f(n)}{g(n)} = n^{\epsilon}$
  - *Exemplo: n^2 é polinomialmente maior que n. No entanto, n logn e 2n não são polinomialmente maiores que n*
- **Função Polinomialmente Menor**
  - Uma função f(n) é polinomialmente menor que outra função g(n) se pudermos achar algum $\epsilon > 0$ tal que $\frac{g(n)}{f(n)} = n^{\epsilon}$
- **Função Assistoticamente Igual**
  - Uma função f(n) é assintoticamente igual a outra função g(n) se $\lim_{n\to\infty}(\frac{g(n)}{f(n)}) = 1$,

Teorema Mestre (7)

- **Observação sobre o Caso 1**
  - A função f(n) não deve ser apenas menor do que $n^{\log_b a}$, deve ser **polinomialmente** menor do que $n^{\log_b a}$
  - Em outras palavras, $\frac{n^{\log_b a}}{f(n)} = n^{\epsilon}$
    - f(n) deve ser assintoticamente menor que $n^{\log_b a}$ por um favor n^{ϵ}, para alguma constante $\epsilon > 0$

Teorema Mestre (8)

- **Observação sobre o Caso 3**
  - A função f(n) não deve ser apenas maior do que $n^{\log_b a}$, deve ser **polinomialmente** maior do que $n^{\log_b a}$
  - Em outras palavras, $\frac{f(n)}{n^{\log_b a}} = n^{\epsilon}$
    - f(n) deve ser assintoticamente maior que $n^{\log_b a}$ por um favor n^{ϵ}, para alguma constante $\epsilon > 0$
  - Ainda, f(n) deve satisfazer a condição de “regularidade” $a.f(n/b) \leq c.f(n)$ para alguma constante $c < 1$ e n suficientemente grande
  - A condição de regularidade é satisfeita pela maioria das funções polinomiais que encontramos

Teorema Mestre (9)

- **Condição de Regularidade:**
  - A condição de regularidade estabelece que $a.f(n/b) \leq c.f(n)$ para alguma constante $c < 1$ e para todo n suficientemente grande
  - Isto é, para n tendendo ao infinito, há uma constante c ($0 < c < 1$) tal que $a.f(n/b) \leq c.f(n)$
    - A condição de regularidade sempre é satisfeita quando T(n) é uma função monotonicamente não decrescente (por exemplo 2^n, n^2, log n, n!, etc.)
    - Algumas funções de n (tais como sen(n) e cos(n)) não monotonicamente não decrescentes, e não satisfazem a condição de regularidade
  - Em problemas reais e de interesse prático, o tempo de execução nunca decresce quando n cresce

Teorema Mestre (10)

- **Observações Gerais:**
  - Existem **casos não cobertos para f(n) pelo Teorema Mestre**:
    - Entre os casos 1 e 2, existem funções f(n) que são menores que $n^{\log_b a}$, mas não são polinomialmente menores
    - Entre os casos 2 e 3, existem funções f(n) que são maiores que $n^{\log_b a}$, mas não são polinomialmente maiores
  - **Se f(n) cai em um destes casos, ou não atende à condição de regularidade do caso 3, então não é possível aplicar o Teorema Mestre**

Teorema Mestre – Exemplo 1

- **Recorrência:**

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

Cola:

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

$$T(n) = aT(n/b) + f(n), \quad (1)$$

em que interpretamos $T(n/b)$ como $T(\lceil n/b \rceil)$ ou $T(\lfloor n/b \rfloor)$.

Então $T(n)$ possui os seguintes limites assintóticos:

- Se $f(n) = O(n^{log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) = \Theta(n^{log_b a})$.
- Se $f(n) = \Theta(n^{log_b a})$, então $T(n) = \Theta(n^{log_b a} log\ n)$.
- 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 n suficientemente grande, então $T(n) = \Theta(f(n))$.

- **Resolução da Recorrência:**

  - $a = 9$;
  - $b = 3$;
  - $f(n) = n$.

Logo, $n^{log_b a} = n^{log_3 9} = \Theta(n^2)$.

Como $f(n) = O(n^{log_3 9 - \epsilon})$, onde $\epsilon = 1$, podemos aplicar o caso 1 do Teorema Mestre e concluir que a solução da recorrência é

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

Teorema Mestre – Exemplo 2

Cola:

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

$$T(n) = aT(n/b) + f(n), \qquad (1)$$

em que interpretamos $T(n/b)$ como $T(\lceil n/b \rceil)$ ou $T(\lfloor n/b \rfloor)$.

Então $T(n)$ possui os seguintes limites assintóticos:

- Se $f(n) = O(n^{log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) = \Theta(n^{log_b a})$.
- Se $f(n) = \Theta(n^{log_b a})$, então $T(n) = \Theta(n^{log_b a} log\ n)$.
- 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 n suficientemente grande, então $T(n) = \Theta(f(n))$.

- **Recorrência:**

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

- **Resolução da Recorrência:**

- $a = 1$;
- $b = 3/2$;
- $f(n) = 1$.

Logo, $n^{log_b a} = n^{log_{3/2} 1} = n^0 = 1.$

Obs.: $(3/2)^x = 1$

Como $f(n) = \Theta(n^{log_b a}) = \Theta(1)$, podemos aplicar o caso 2 do Teorema Mestre e concluir que a solução da recorrência é

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

Teorema Mestre – Exemplo 3

- **Recorrência:**

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

Cola:

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

$$T(n) = aT(n/b) + f(n), \quad (1)$$

em que interpretamos $T(n/b)$ como $T(\lceil n/b \rceil)$ ou $T(\lfloor n/b \rfloor)$.

Então $T(n)$ possui os seguintes limites assintóticos:

- Se $f(n) = O(n^{log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) = \Theta(n^{log_b a})$.
- Se $f(n) = \Theta(n^{log_b a})$, então $T(n) = \Theta(n^{log_b a} log\ n)$.
- 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 n suficientemente grande, então $T(n) = \Theta(f(n))$.

- **Resolução da Recorrência:**
  - $a = 3$;
  - $b = 4$;
  - $f(n) = n\ log\ n$.

Logo, $n^{log_b a} = n^{log_4 3} = n^{0,793}$.

Como $f(n) = \Omega(n^{log_4 3 + \epsilon})$, em que $\epsilon \approx 0,2$, podemos aplicar o caso 3 do Teorema Mestre se provarmos que a condição de regularidade é verdadeira para $f(n)$:

$$af(n/b) = 3(n/4)log(n/4) \leq (3/4)n\ log\ n = cf(n)$$

para $c = 3/4$. Desta forma, podemos aplicar o caso 3 do Teorema Mestre e concluir que a solução da recorrência é

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

Teorema Mestre – Exemplo 4

- **Recorrência:**

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

Cola:

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

$$T(n) = aT(n/b) + f(n), \qquad (1)$$

em que interpretamos $T(n/b)$ como $T(\lceil n/b \rceil)$ ou $T(\lfloor n/b \rfloor)$.

Então $T(n)$ possui os seguintes limites assintóticos:

- Se $f(n) = O(n^{log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) = \Theta(n^{log_b a})$.
- Se $f(n) = \Theta(n^{log_b a})$, então $T(n) = \Theta(n^{log_b a} log\ n)$.
- 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 n suficientemente grande, então $T(n) = \Theta(f(n))$.

- **Resolução da Recorrência:**

  - $a = 2$;
  - $b = 2$;
  - $f(n) = n\ log\ n$.

Logo, $n^{log_b a} = n^{log_2 2} = n$.

Embora $f(n)$ seja *assintoticamente* maior do que $n^{log_b a}$, 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 positiva ϵ.

Desta forma, a recorrência cai entre os casos 2 e 3 e não pode ser resolvida pelo Teorema Mestre.

Conclusões

- **Recorrência é uma equação ou desigualdade** que **descreve uma função em termos de si mesma**, mas com entradas menores
- Como a **complexidade de um algoritmo recursivo** é expressa através de uma **recorrência**, é preciso determiná-la efetivamente
- **“Resolvemos” uma recorrência quando conseguimos eliminar as referências a si mesma**
- **Melhores técnicas**:
  - Substituição de variáveis
  - Iterações
  - Árvore de recorrência
  - Teorema Mestre

PANC: Projeto e Análise de Algoritmos

Aula 08: Resolução de Recorrências - Teorema Mestre

Breno Lisi Romano

Dúvidas???

http://sites.google.com/site/blromano

Instituto Federal de São Paulo – IFSP São João da Boa Vista
Bacharelado em Ciência da Computação – 3º Semestre

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus São João da Boa Vista