Recorrências: Teorema Mestre

Profa. Juliana Kaizer Vizzotto

Projeto e Análise de Algoritmos. Material do livro **Introduction to Algorithms**. Thomas H. Cormen. Charles E. Leiserson. Ronald L. Rivest. Clifford Stein. Third Edition. The MIT Press.

Teorema Mestre

- Teorema Mestre para resolver Recorrências da forma:

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

  onde $a \geq 1$ e $b > 1$ são constantes e $f(n)$ é uma função assintóticamente positiva.
- Para usar o teorema mestre basta memorizar três casos! Então poderemos resolver muitas recorrências de uma maneira direta!
- A recorrência acima descreve o tempo de execução de um algoritmo que divide um problema de tamanho n em a subproblemas, cada um com tamanho n/b.
- Os as subproblemas são resolvidos recursivamente, cada um com tempo $T(n/b)$.

Teorema Mestre

- A função $f(n)$ inclui o custo de dividir o problema e combinar os resultados dos subproblemas.

Teorema Mestre

- **Teorema Mestre:** sejam $a \geq 1$ e $b > 1$ constantes, seja $f(n)$ uma função, e seja $T(n)$ definida para inteiros não negativos pela recorrência:

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

Então, $T(n)$ tem os seguintes limites assintóticos:

1. Se $f(n) = O(n^{\log_b a - \epsilon})$ para alguma constrante $\epsilon > 0$, então $T(n) = \Theta(n^{\log_b^a})$.
2. Se $f(n) = \Theta(n^{\log_b a})$, então $T(n) = \Theta(n^{\log_b^a} lgn)$.
3. Se $f(n) = \Omega(n^{\log_b a + \epsilon})$ para alguma constrante $\epsilon > 0$, 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

- Devemos comparar a função $f(n)$ com a função $n^{\log_b^a}$.
- Intuitivamente, a maior das duas funções determina a solução da recorrência.
- **Caso 1:** $f(n) = O(n^{\log_b^{a-\epsilon}})$ para alguma constante $\epsilon > 0$. Ou seja, $f(n)$ é polinomialmente menor que $n^{\log_b^a}$. A solução é $T(n) = \Theta(n^{\log_b^a})$. Intuitivamente, o custo é dominado pelas folhas!
- **Caso 2:** $f(n) = \Theta(n^{\log_b^a} lg^k n)$. Ou seja, $f(n)$ é um fator polylog de $n^{\log_b^a}$, mas não menor. A solução é $T(n) = \Theta(n^{\log_b^a} lg^{k+1} n)$. Intuitivamente, o custo é $n^{\log_b^a} lg^k n$ em cada nível, e existem $\Theta(lgn)$ níveis. No caso mais simples, $k = 0 \Rightarrow f(n) = \Theta(n^{\log_b^a} lgn)$.

Teorema Mestre

- **Caso 3:** $f(n) = \Omega(n^{\log_b^{a+\epsilon}})$ para alguma constante $\epsilon > 0$ e $f(n)$ satisfaz a condição de regularidade $af(n/b) \leq cf(n)$ para alguma constante $c < 1$ e n suficientemente grande. Ou seja, $f(n)$ é polinomialmente maior que $n^{\log_b^a}$. A solução é $T(n) = \Theta(f(n))$. Intuitivamente, o custo é dominado pela raiz!

Teorema Mestre

Exemplos:

- $T(n) = 5T(n/2) + \Theta(n^2)$: Temos $a = 5$, $b = 2$ e $f(n) = \Theta(n^2)$. Comparamos $n^{\log_b^a} = n^{\log_2^5}$ versus n^2. Como $\log_2^5 - \epsilon = 2$ para algum $\epsilon > 0$. Caso 1 se aplica, assim $T(n) = \Theta(n^{lg\ 5})$.

Teorema Mestre

Exemplos:

- $T(n) = 5T(n/2) + \Theta(n^3)$: Temos $a = 5$, $b = 2$ e $f(n) = \Theta(n^3)$. Comparamos $n^{\log_b^a} = n^{\log_2^5}$ versus n^3. Agora $\log_2^5 + \epsilon = 3$ para algum $\epsilon > 0$. Verificando a condição de regularidade $af(n/b) = 5(n/2)^3 = 5(n^3/8) \leq cn^3$ para $c = 5/8 < 1$.
  Caso 3 se aplica, assim $T(n) = \Theta(n^3)$.

Teorema Mestre

Solução Exercícios

- $T(n) = 8T(n/2) + \Theta(n^2)$: descreve o tempo de execução de um algoritmo de divisão e conquista para multiplicação de matrizes. Temos $a = 8$, $b = 2$ e $f(n) = \Theta(n^2)$. Assim, $n^{\log_b^a} = n^{\log_2^8 = 3}$. Como n^3 é polinomialmente maior que $f(n)$ (i.e, $f(n) = O(n^{3-\epsilon})$ para $\epsilon = 1$). Caso 1 se aplica, assim $T(n) = \Theta(n^3)$.

Teorema Mestre

Solução Exercícios

- $T(n) = 7T(n/2) + \Theta(n^2)$: descreve o tempo de execução do algoritmo de *Strassen* para multiplicação de matrizes. Temos $a = 7$, $b = 2$ e $f(n) = \Theta(n^2)$. Assim, $n^{\log_b^a} = n^{\log_2^7 = 3}$. Reescrevendo $\log_2^7$ como $lg7$ e $2.8 < lg7 < 2.81$. Temos que $f(n) = O(n^{lg7-\epsilon})$ para $\epsilon = 0.8$). Caso 1 se aplica, assim $T(n) = \Theta(n^{lg7})$.

Teorema Mestre

Solução Exercícios

- $T(n) = T(9n/10) + \Theta(n)$: Temos $a = 1$, $b = 10/9$ e $f(n) = \Theta(n)$. Assim, $n^{\log_b^a} = n^{\log_{10/9}^1 = n^0 = 1}$. Como $f(n) = \Omega(n^{\log_{10/9}^{1+1}})$ e $a/b^k = 1/(10/9)^1 = 9/10 < 1$. Caso 3 do teorema mestre se aplica e temos que $T(n) = \Theta(n)$.

Teorema Mestre

Solução Exercícios

- 4.5-3 binary search
- $T(n) = T(n/2) + \Theta(1)$: Temos $a = 1$, $b = 2$ e $f(n) = \Theta(1)$. Assim, $n^{\log_b^a} = n^{\log_2^1 = n^0 = 1}$. Como $f(n) = \Theta(n^{\log_2^1} lg^0 n)$. Caso 2 se aplica. Solução é $T(n) = \Theta(n^{\log_2^1} lg^1 n) = \Theta(lgn)$.

Teorema Mestre

Solução Exercícios

- $T(n) = 2T(n/2) + \Theta(n)$: Temos $a = 2$, $b = 2$ e $f(n) = \Theta(n)$. Assim, $n^{\log_b^a} = n^{\log_2^2 = n^1 = n}$. Como $f(n) = \Theta(n^{\log_2^2} lg^0 n)$. Caso 2 se aplica. Solução é $T(n) = \Theta(n^{\log_2^2} lg^1 n) = \Theta(nlgn)$.