A Generalisation of The Master Theorem

August 25, 2010

The Master Recurrence frequently occurs in Divide-and-Conquer algorithms:

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

where $a, b > 0$ and $d(n)$ is the driving function. Define the watershed function as n^α, where $\alpha := \log_b a$. Depending upon the relation between $d(n)$ and n^α we get different asymptotics for $T(n)$. The Master Theorem gives us the following:

THEOREM 1. *Let $T(n), d(n), a, b, \alpha$ be as defined above. Then*

$$T(n) = \begin{cases} \Theta(d(n)) & \text{if } d(n) = \Omega(n^{\alpha+\epsilon}), \text{ for some } \epsilon > 0; \\ \Theta(n^\alpha \log n) & \text{if } d(n) = \Theta(n^\alpha); \\ \Theta(n^\alpha) & \text{if } d(n) = \mathcal{O}(n^{\alpha-\epsilon}), \text{ for some } \epsilon > 0. \end{cases}$$

The theorem fails when $d(n)$ is not polynomially related to n^α, e.g., when $d(n) = n \log n$. The following theorem fills this log-gap.

THEOREM 2. *Let $T(n), d(n), a, b, \alpha$ be as defined above. Then*

$$T(n) = \begin{cases} \Theta(d(n)) & \text{if } d(n) \geq Ca \cdot d(n/b), \text{ for } n \text{ large enough and some } C > 1; \\ \Theta(d(n) \log n) & \text{if } d(n) = \Theta(n^\alpha \log^\delta n), \text{ for some } \delta > -1; \\ \Theta(d(n) \log n \log\log n) & \text{if } d(n) = \Theta(n^\alpha / \log n); \\ \Theta(n^\alpha) & \text{if } d(n) = \Theta(n^\alpha \log^\delta n), \text{ for some } \delta < -1. \end{cases}$$

Proof. Using domain and range transformations, we get that

$$T(n) = \sum_{j=1}^{\log_b n} a^j d(n/b^j). \tag{2}$$

Case 1): $T(n) = \Omega(d(n))$ follows from (1). We prove the upper bound. From the regularity conditiion we get, $a^j d(n/b^j) < d(n)/C^j$. Putting this upper bound in (2), we obtain

$$T(n) \leq \sum_{j=1}^{\log_b n} d(n)/C^j < d(n) \sum_{j \geq 0} C^{-j} = d(n)/(1-C) = \mathcal{O}(d(n))$$

Case 2) Using $d(n) = \Theta(n^\alpha \log^\delta n)$ in (2), we get

$$\begin{aligned} T(n) &= \Theta(\textstyle\sum_{j=1}^{\log_b n} a^j \left(\frac{n}{b^j}\right)^\alpha \log^\delta(n/b^j)) \\ &= \Theta(\textstyle\sum_{j=1}^{\log_b n} n^\alpha \log^\delta(n/b^j)) \qquad (\text{Since } b^{\alpha j} = a^j) \\ &= \Theta(n^\alpha \textstyle\sum_{j=1}^{\log_b n} (\log_b n - j)^\delta) \\ &= \Theta(n^\alpha \textstyle\sum_{j=1}^{\log_b n - 1} j^\delta) \\ &= \Theta(n^\alpha \log_b n^{\delta+1}) \\ &= \Theta(d(n) \log n), \end{aligned}$$

where we use the relation $\sum_{j=1}^{k} j^{\delta} = \Theta(k^{1+\delta})$, for positive δ; this can be derived using the integration technique, and since $\delta + 1 > 0$, the constant on the RHS is well defined.

Case 3) Plugging in $d(n) = \Theta(d(n)/\log n)$ in (2) we obtain

$$\begin{aligned} T(n) &= \Theta(\textstyle\sum_{j=1}^{\log_b n-1} n^{\alpha}/(\log_b n - j)) \\ &= \Theta(n^{\alpha} \textstyle\sum_{j=1}^{\log_b n-1} \frac{1}{j}) \\ &= \Theta(n^{\alpha} \log\log n), \end{aligned}$$

where we use the relation $\sum_{j=1}^{k} 1/j = \log k$, which can also be proved using the integration technique, with the integral function being $1/x$.

Case 4) Plugging in $d(n) = \mathcal{O}(n^{\alpha} \log^{\delta} n)$, $\delta < -1$, in (2) we obtain

$$\begin{aligned} T(n) &= \mathcal{O}(\textstyle\sum_{j=1}^{\log_b n-1} n^{\alpha}(\log_b n - j)^{\delta}) \\ &= \mathcal{O}(n^{\alpha} \textstyle\sum_{j=1}^{\log_b n-1} j^{\delta}) \\ &= \mathcal{O}(n^{\alpha} \textstyle\sum_{j\geq 1} j^{\delta}) \\ &= \mathcal{O}(n^{\alpha}), \end{aligned}$$

since $\sum j \geq 1 j^{\delta}$, for $\delta \leq -2$, is bounded from above by a constant; for $\delta = -2$ it converges to $\pi^2/6$. **Q.E.D.**

The theorem can be found in Fundamentals of Algorithms, by Brassard and Bratley.