Two Methods for Solving Recurrences

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong

We have seen how to analyze the running time of recursive algorithms by recurrence. It is important to sharpen our skills in solving recurrences. Today, we will learn two techniques for this purpose: the master theorem and the substitution method.

The Master Theorem

The Master Theorem

Let $f(n)$ be a function that returns a positive value for every integer $n > 0$. We know:

$$\begin{aligned} f(1) &= O(1) \\ f(n) &\leq \alpha \cdot f(\lceil n/\beta \rceil) + O(n^{\gamma} \cdot \log^{\lambda} n) \qquad (\text{for } n \geq 2) \end{aligned}$$

where $\alpha \geq 1$, $\beta > 1$, $\gamma \geq 0$, and $\lambda \geq 0$ are constants. Then:

- If $\log_{\beta} \alpha < \gamma$, then $f(n) = O(n^{\gamma} \log^{\lambda} n)$.
- If $\log_{\beta} \alpha = \gamma$, then $f(n) = O(n^{\gamma} \log^{\lambda+1} n)$.
- If $\log_{\beta} \alpha > \gamma$, then $f(n) = O(n^{\log_{\beta} \alpha})$.

The theorem can be proved by carefully applying the "expansion method" we saw earlier. The details are tedious and omitted.

Example 1

Consider the recurrence of binary search:

$$\begin{aligned} f(1) &\leq c_1 \\ f(n) &\leq f(\lceil n/2 \rceil) + c_2 \qquad (\text{for } n \geq 2) \end{aligned}$$

Hence, $\alpha = 1$, $\beta = 2$, $\gamma = 0$, and $\lambda = 0$. Since $\log_\beta \alpha = \gamma$, we know that $f(n) = O(n^0 \cdot \log^{0+1} n) = O(\log n)$.

Example 2

Consider the recurrence of merge sort:

$$\begin{aligned} f(1) &\leq c_1 \\ f(n) &\leq 2 \cdot f(\lceil n/2 \rceil) + c_2 n \qquad (\text{for } n \geq 2) \end{aligned}$$

Hence, $\alpha = 2$, $\beta = 2$, $\gamma = 1$, and $\lambda = 0$. Since $\log_\beta \alpha = \gamma$, we know that $f(n) = O(n^\gamma \cdot \log^{0+1} n) = O(n \log n)$.

Example 3

Consider the recurrence:

$$\begin{aligned} f(1) &\leq c_1 \\ f(n) &\leq 2 \cdot f(\lceil n/4 \rceil) + c_2 \sqrt{n} \cdot \log_2 n \qquad (\text{for } n \geq 2) \end{aligned}$$

Hence, $\alpha = 2$, $\beta = 4$, $\gamma = 1/2$, and $\lambda = 1$. Since $\log_\beta \alpha = \gamma$, we know that $f(n) = O(n^\gamma \cdot \log^{\lambda+1} n) = O(\sqrt{n} \log^2 n)$.

Example 4

Consider the recurrence:

$$\begin{aligned} f(1) &\leq c_1 \\ f(n) &\leq 2 \cdot f(\lceil n/2 \rceil) + c_2\sqrt{n} \qquad (\text{for } n \geq 2) \end{aligned}$$

Hence, $\alpha = 2$, $\beta = 2$, $\gamma = 1/2$, and $\lambda = 0$. Since $\log_\beta \alpha > \gamma$, we know that $f(n) = O(n^{\log_\beta \alpha}) = O(n)$.

Example 5

Consider the recurrence:

$$\begin{aligned} f(1) &\leq c_1 \\ f(n) &\leq 13 \cdot f(\lceil n/7 \rceil) + c_2 n^2 \qquad (\text{for } n \geq 2) \end{aligned}$$

Hence, $\alpha = 13$, $\beta = 7$, $\gamma = 2$, and $\lambda = 0$. Since $\log_\beta \alpha < \gamma$, we know that $f(n) = O(n^\gamma \log^\lambda n) = O(n^2)$.

The Substitution Method

Recall that, to prove $f(n) = O(g(n))$, it suffices to find an **arbitrary** pair of constants c_1 and c_2 such that $\forall n \geq c_1$, $f(n) \leq c_2 \cdot g(n)$.

Rationale: In the substitution method, we aim to prescribe a set of **sufficient** conditions for c_1 and c_2, and then "solve" c_1 and c_2 from those conditions.

We will resort to mathematical induction to achieve the purpose.

Example 6

Consider the recurrence:

$$\begin{aligned} f(1) &= 1 \\ f(n) &\leq f(n-1) + 11n \qquad (\text{for } n \geq 2) \end{aligned}$$

We will prove $f(n) = O(n^2)$ by the substitution method.

Aim: Find c_1, c_2 such that $f(n) \leq c_2 n^2$ for all $n \geq c_1$.

Base case: $n = c_1$. We need:

$$f(c_1) \leq c_2 \cdot c_1^2 \quad \Leftrightarrow \quad c_2 \geq f(c_1)/c_1^2 \tag{1}$$

Inductive case: Suppose that $f(n) \leq c_2 n^2$ for all $n \leq k-1$ where $k \geq 1 + c_1$.Then, we have:

$$f(k) \leq f(k-1) + 11k \leq c_2 \cdot (k-1)^2 + 11k$$

To make sure $f(k) \leq c_2 k^2$, it suffices to have

$$\begin{aligned} c_2 \cdot (k-1)^2 + 11k &\leq c_2 k^2 \\ \Leftrightarrow c_2 &\geq 11k/(2k-1). \end{aligned}$$

For $k \geq 1$, we always have $\frac{11k}{2k-1} \leq 11$. Thus, it suffices to ensure

$$c_2 \geq 11. \tag{2}$$

Any pair of c_1 and c_2 satisfying (1) and (2) works. For example, we can set $c_1 = 1$ and $c_2 = \max\{f(1)/1^2, 11\} = 11$.

Example 7

Consider the recurrence:

$$f(1) = f(2) = f(3) = 1$$
$$f(n) \leq f(\lceil n/5 \rceil) + f\left(\left\lceil \frac{7n}{10} \right\rceil\right) + n \qquad (\text{for } n \geq 4)$$

This is really a non-trivial recurrence (the master theorem is inapplicable here). We will prove that $f(n) = O(n)$ using the substitution method.

Aim: Find c_1, c_2 such that $f(n) \leq c_2 n$ for all $n \geq c_1$.

Base case: $n = c_1$. We need: $f(c_1) \leq c_2 \cdot c_1$. To make this happen, it suffices to ensure

$$f(c_1) \leq c_2 \cdot c_1 \Leftrightarrow c_2 \geq f(c_1)/c_1. \tag{3}$$

Inductive case: Suppose that $f(n) \leq c_2 n$ under $n \leq k-1$ where $k \geq 1 + c_1$. We have:

$$\begin{aligned} f(k) &\leq f(\lceil k/5 \rceil) + f(\lceil 7k/10 \rceil) + k \\ &\leq c_2(\lceil k/5 \rceil) + c_2(\lceil 7k/10 \rceil) + k \\ &\leq c_2(k/5+1) + c_2((7k/10)+1) + k \\ &= c_2(9/10)k + 2c_2 + k \end{aligned}$$

To ensure $f(k) \leq c_2 k$, it suffices to have:

$$\begin{aligned} c_2(9/10)k + 2c_2 + k &\leq c_2 k \\ \Leftrightarrow c_2(k/10 - 2) &\geq k \end{aligned} \tag{4}$$

To simplify calculation, we demand $k \geq 21$. To make this happen, it suffices to ensure

$$c_1 \geq 20. \tag{5}$$

When $k \geq 21$, it holds that $k/10 - 2 > 0$. With this, we derive:

$$(4) \Leftrightarrow c_2 \geq \frac{k}{k/10 - 2} = \frac{10k}{k - 20} \tag{6}$$

We will further demand that $k \geq 40$. To make this happen, it suffices to ensure

$$c_1 \geq 39. \tag{7}$$

For $k \geq 40$, $k - 20 \geq k/2$. So to ensure (6), it suffices to have

$$c_2 \geq \frac{10k}{k/2} = 20. \tag{8}$$

We now know that any pair of c_1, c_2 satisfying (3), (5), (7), and (8) works. For example, we can set

$$\begin{aligned} c_1 &= 39 \\ c_2 &= \max\{f(39)/39, 20\}. \end{aligned}$$