Recurrence Equations and the Master Method

Hsing-Kuo Pao

Department of Computer Science & Information Engineering
National Taiwan University of Science & Technology
pao@mail.ntust.edu.tw

1 Introduction

Recall that when we analysize the time complexity of merge sort, we have its complexity $T(n)$ written in a recurrence form such as

$$T(n) = 2T(n/2) + n\,,\ T(1) = 1\,. \tag{1}$$

It indicates that we will partition the input of size n into two groups with (almost) equal[*] size of $n/2$, and we need n steps (or $\Theta(n)$ time) for the merge operation. As mentioned before, the result for the equation is by

$$T(n) = n\lg n + n = \Theta(n\lg n)\,.$$

In general, for a recursive *divide-and-conquer* algorithm, the typical recurrence equation we want to solve is

$$T(n) = aT(n/b) + f(n) = aT(n/b) + D(n) + C(n)\,, \tag{2}$$

for constants a, b, and a nonnegative function $f(n)$ which represents the time spending on the dividing ($= D(n)$) and the combining operations ($= C(n)$). Usually we also assume $a \geq 1$, $b > 1$, $n \in \mathbf{N} \cup \{0\}$. We assume the integer arguments just for simplcity and the result from this assumption can be generalized easily to the case when a number other than an integer is involved. We proceed to discuss some examples.

Example 1

$$T(n) = 4T(n/2) + n\,,\ T(1) = 1\,.$$

[*]Given an array A, with elements from index p to r for sorting, and a partition index q in the middle, the complexity for merge sort can be given by $T(n) = T(r-p+1) = T(q-p+1) + T(r-q) +$ time to merge. Either half of the input after partition, of size $q-p+1$ or $r-q$ is very close to $n/2$. So we have the simplified recurrence equation in Eq. 1.

Sol. *Let us try to expand the formula.*

$$\begin{aligned}
T(n) &= 4T(n/2) + n \\
&= 4\left(4T(n/2^2) + n/2\right) + n \\
&= 4^2T(n/2^2) + 2\cdot n + n \\
&= \cdots \\
&= 4^kT(n/2^k) + n\sum_{i=0}^{k-1} 2^i \\
&= 4^kT(n/2^k) + n(2^k - 1)\,.
\end{aligned}$$

Assume $2^k = n$ *or* $k = \lg n$ *and* $4^k = n^2$*, we have*

$$T(n) = n^2T(1) + n(n-1) = 2n^2 - n = \Theta(n^2)\,.$$

□

We can check its accuracy.

$$\begin{aligned}
T(1) &= 2\cdot 1^2 - 1 = 1\,, \\
T(2) &= 4T(1) + 2 = 6 = 2\cdot 2^2 - 2\,, \\
&\text{or} \\
T(4) &= 4T(2) + 4 = 28 = 2\cdot 4^2 - 4\,.
\end{aligned}$$

Remember that we care only those $n = b^k$, for the recurrence Eq. 2. Let us make $f(n)$ to be 0.

Example 2

$$T(n) = 4T(n/2)\,,\ T(1) = 1\,, \tag{3}$$

Sol. *By the similar computation as in Ex. 1, we have*

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

□

Example 3

$$T(n) = 4T(n/2) + 1\,,\ T(1) = 1\,,$$

Sol. *Similarly, by expanding the equation, we have*

$$T(n) = \frac{1}{3}(4n^2 - 1) = \Theta(n^2)\,.$$

□

The first three examples falling in the case when we have a small $f(n)$. In those cases, we have the result with similar *asymptotic* behavior as $\Theta(n^2) = \Theta(n^{\log_2 4})$. Check another example

$$T(n) = 2T(n/4)\,,\ T(1) = 1\,,$$

which will have the result

$$T(n) = \sqrt{n}\,.$$

Later we will understand that the result of the form $\Theta(n^{\log_b a})$ comes quite often in this kind of recurrence equations. Now let us make $f(n)$ larger as follows.

Example 4

$$T(n) = 4T(n/2) + n^2\,,\ T(1) = 1\,.$$

Sol. *We have*

$$\begin{aligned} T(n) &= 4(4T(n/2^2) + (n/2)^2) + n^2 \\ &= 4^2T(n/2^2) + 2n^2 \\ &= 4^kT(n/2^k) + kn^2\,. \end{aligned}$$

Again, we assume $2^k = n$, $k = \lg n$ *and we have*

$$T(n) = n^2 \lg n + n^2 = \Theta(n^2 \lg n)\,.$$

□

The result is asymptotically larger than the solutions of Ex. 1, 2 or 3, because we have an extra term $n^2 \lg n$.

We can consider a case similar to Ex. 4

$$\begin{aligned} T(n) &= 9T(n/3) + n^2 \\ &= 9\left(9T(n/3^2) + (n/3)^2\right) + n^2 \\ &= 9^kT(n/3^k) + kn^2 \\ &= n^2 + n^2 \lg n \quad \text{(assume } n = 3^k, T(1) = 1) \\ &= \Theta(n^2 \lg n)\,. \end{aligned}$$

Also, as a smaller $f(n)$, the solution to $T(n) = 9T(n/3)$ and $T(1) = 1$ will be $T(n) = n^2$. The order of n is again $\log_b a$ or $\log_3 9 = 2$.

In general, for Eq. 2, the first part of the recurrence will produce the term

$$a^k = a^{\log_b n} = a^{\log_a n/\log_a b} = n^{\log_b a}\,, \quad \text{(assume } b^k = n \text{ or } k = \log_b n)$$

and the second part of the computation will produce a term with lower order compared to $n^{\log_b a}$, if $f(n)$ is small; or produce a term $n^{\log_b a} \lg$ if $f(n)$ is a bit larger (roughly speaking). We continue to make $f(n)$ larger and larger.

Example 5

$$T(n) = 4T(n/2) + n^3\,,\ T(1) = 1\,.$$

Sol. *We solve it by*

$$\begin{aligned} T(n) &= 4(4T(n/2^2) + (n/2)^3) + n^3 \\ &= 4^2T(n/2^2) + n^3(1 + 1/2) \\ &= 4^kT(n/2^k) + n^3 \sum_{i=0}^{k-1} 2^{-i} \\ &= n^2T(1) + 2n^3(1 - 1/n) \quad (\textit{assume } k = \lg n) \\ &= 2n^3 - n^2 \\ &= \Theta(n^3)\,. \end{aligned}$$

□

As we can see, the computation from these examples gives us some hints to solve Eq.2:

1. When $f(n)$ is small enough, we have $T(n) = \Theta(n^{\log_b a})$.
2. When $f(n)$ is not too small nor too large, we have $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. When $f(n)$ is large enough, we will have $T(n) = \Theta(f(n))$.

Our conjecture is not very far from the master theorem. Before we move on to the master theorem, we need to emphasize that to rigorously obtain the results for the previous examples, we need *mathematical induction*, illustrated as follows. That is, taking the Ex. 2, we can only "guess" the answer to Eq. 3 is $T(n) = n^2$. We need to prove the answer by induction:

i Check $n = 1$, $T(1) = 1^2 = 1$,

ii Assume $n = k$, $T(k) = k^2$,
$\Rightarrow T(2k) = 4T(k) = 4k^2 = (2k)^2\,.$

We proved the assertion $T(n) = n^2$ for the equation Eq. 3.

2 The Master Method

Theorem 1 (Master theorem) *Let $a \geq 1$, $b > 1$, $n \in \mathbf{N} \cup \{0\}$, and let $f(n)$ be a nonnegative function. Let $T(n)$ be defined by the recurrence equation $T(n) = aT(n/b) + f(n)$, then $T(n)$ can be bounded asymptotically as follows.*
Case 1: If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$,
Case 2: If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$,
Case 3: If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$.

From now on, we will omit the initial condition such as $T(1) = 1$ and also we will basically try to find out the asymptotic form of the answer instead of solving for the exact one.

Example 6

$$T(n) = 7T(n/3) + n\,.$$

Sol. *Let us compare* $f(n)$ *with* $n^{\log_b a}$

$$n^{\log_b a} = n^{\log_3 7} \textit{ greater than } n\,,$$

and we can guess

$$T(n) = \Theta(n^{\log_3 7})\,.$$

By the master theorem (case 1), more rigorously, we have

$$\begin{aligned}&n^{\log_b a} \approx n^{1.772}\\ &f(n) = n = O(n^{\log_b a - \epsilon}) \quad \textit{choose } \ \epsilon = 0.5\,,\\ &\Rightarrow T(n) = \Theta(n^{\log_3 7})\,.\end{aligned}$$

□

Example 7

$$T(n) = 2T(n/4) + \sqrt{n}\,.$$

Sol.

$$f(n) = \sqrt{n} = n^{\log_4 2}\,.$$

By the master theorem, case 2

$$T(n) = \Theta(\sqrt{n}\lg n)\,.$$

□

Example 8

$$T(n) = 3T(n/2) + n\lg n\,.$$

Sol.

$$f(n) = n\lg n = O(n^{\log_2 3 - \epsilon})\,, \quad \textit{choose } \epsilon = 0.5 \quad (\log_2 3 \approx 1.585)$$

Because we know

$$\lg n = O(n^c)\,, \quad \textit{for any small } c > 0$$

By the master theorem, case 1, we have

$$T(n) = \Theta(n^{\log_2 3})\,.$$

□

Let us discuss another recurrence equation as follow

$$T(n) = 3T(n/2) + n^3 .$$

We can compare between $f(n) = n^3$ and $n^{\log_b a} = n^{\log_2 3}$ as

$$f(n) = n^3 = \Omega(n^{\log_2 3+\epsilon}) , \text{ choose } \epsilon = 1 \quad (\log_2 3 \approx 1.585)$$

Can we apply the master theorem, case 3 to obtain

$$T(n) = \Theta(n^3)?$$

Not yet! We need to check the *regularity condition*

$$af(n/b) = 3(n/2)^3 = \frac{3}{8}n^3 \leq cn^3 . \quad \text{choose } c = \frac{3}{8} < 1$$

So the case 3 of the master theorem is applied and we can conclude

$$T(n) = \Theta(n^3) .$$

Example 9

$$T(n) = 3T(n/2) + n^2 \lg n .$$

Sol.

$$f(n) = n^2 \lg n = \Omega(n^{\log_2 3+\epsilon}) , \textit{ choose } \epsilon = 0.2 \quad (\log_2 3 \approx 1.585)$$

also

$$af(n/b) = 3(n/2)^2 \lg(n/2) = \frac{3}{4}n^2 \lg(n/2) \leq \frac{3}{4}n^2 \lg n = cn^2 \lg n , \quad \textit{for } c = \frac{3}{4} .$$

By the master theorem, case 3, we have

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

□

In many cases, the regularity condition will not create too much trouble. For instance, it holds whenever $f(n) = n^k, k \in \mathbf{N}$ because

$$af(n/b) = a(n/b)^k = \frac{a}{b^k}n^k = cf(n) , \quad \text{choose } c = \frac{a}{b^k}$$

If that is the case when we can apply the master theorem, the case 3, we will have

$$f(n) = \Omega(n^{\log_b a+\epsilon}) ,$$

by the definition, we can find a constant c' and n_0 s.t.

$$n^k \geq c' n^{\log_b a+\epsilon} , \; \forall n \geq n_0 \Rightarrow k > \log_b a \Rightarrow b^k > a .$$

Therefore we can find the $c = a/b^k < 1$ such that the case 3 of the master theorem can be applied.

We must understand that the master theorem does not apply to all cases of recurrence equations with the form of Eq. 2. For instance, we may have the recurrence where the regularity condition fails and we can not apply the case 3 of the master method to find the solution. On the other hand, there are other cases which also can not be solved by the master theorem. There are gaps either between the case 1 and case 2, or between the case 2 and case 3 of the master theorem. Let us discuss the following example.

Example 10

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

Sol. *We can compare the function $f(n) = n\lg n$ with $n^{\log_b a} = n$ and find out*

$$f(n) = n\lg n = \Omega(n^{\log_b a})\,.$$

However, we can not find any $\epsilon > 0$ s.t.

$$f(n) = n\lg n = \Omega(n^{\log_b a+\epsilon})\,.$$

That is, $f(n)$ is larger than $n^{\log_b a}$, but not polynomially larger than $n^{\log_b a}$. The true solution can be found by

$$\begin{aligned}
T(n) &= 2\left(2T(n/2^2) + (n/2)\lg(n/2)\right) + n\lg n \\
&= 2^2T(n/2^2) + n\lg(n/2) + n\lg n \\
&= 2^3T(n/2^3) + n\lg(n/2^2) + n\lg(n/2) + n\lg n \\
&= 2^kT(n/2^k) + n\lg(n/2^{k-1}) + \cdots + n\lg(n/2) + n\lg n \\
&= 2^kT(n/2^k) + kn\lg n - n((k-1) + (k-2) + \cdots + 2 + 1) \\
&= 2^kT(n/2^k) + kn\lg n - nk(k-1)/2 \\
&= nT(1) + n\lg^2 n - n\lg n(\lg n - 1)/2 \quad (\textit{assume}\ \ 2^k = n, T(1) = 1) \\
&= n + \frac{1}{2}n\lg^2 n + \frac{1}{2}n\lg n \\
&= \Theta(n\lg^2 n)
\end{aligned}$$

$\square$

In this case, $f(n)$ is between the case 2 and case 3, where both predict the wrong answer which is $\Theta(n\lg n)$.