

Dati e Algoritmi I (Pietracaprina)

Recurrence Relations

1 Introduction

Recurrence relations are a valuable tool in the complexity analysis of algorithms which employ a *divide-and-conquer* strategy. Consider, for example, the well known `MergeSort` algorithm and let $t_{\text{MS}}(n)$ denote its (worst-case) time complexity. Let also $T_c(n)$ be the function defined through the following recurrence

$$T_c(n) = \begin{cases} \beta & \text{for } n = 1 \\ T_c(\lfloor n/2 \rfloor) + T_c(\lceil n/2 \rceil) + cn & \text{for } n > 1. \end{cases} \quad (1)$$

It is easily shown that there exist two constants $0 < c_1 < c_2$ such that

$$T_{c_1}(n) \leq t_{\text{MS}}(n) \leq T_{c_2}(n),$$

for every $n \geq 1$. The methods that will be presented in these notes will show that $T_c(n) \in \Theta(n \log n)$ for every constant c , hence we conclude that $t_{\text{MS}}(n) \in \Theta(n \log n)$.

In general, let $t_A(n)$ denote the time complexity of an algorithm A and $T(n)$ a function defined through a recurrence relation such that $T(n) \in \Theta(f(n))$. If $t_A(n) \leq T(n)$ for every n , then we conclude that $t_A(n) \in O(f(n))$; if instead $t_A(n) \geq T(n)$ for every n , then we conclude that $t_A(n) \in \Omega(f(n))$.

The general form of a recurrence relation is the following:

$$T(n) = \begin{cases} t_n & \text{for } n \leq n_0 \\ \text{Expr}(T, n) + f(n) & \text{for } n > n_0, \end{cases}$$

where $\text{Expr}(T, n)$ denotes an expression combining values $T(n')$, with $n' < n$. By *solving* such a recurrence relation we mean finding an asymptotic estimate or, possibly, a closed formula for $T(n)$. Several methods are known for solving recurrence relations. Here we will describe two of them: the master theorem and the iterative method. Other methods as well as more details on recurrence relations can be found in [CLRS01].

2 Master Theorem

The following theorem, whose proof can be found in [CLRS01], provides a tight asymptotic solution for a large class of recurrence relations often arising in the analysis of algorithms.

Theorem 1 Let $a \geq 1$, $b > 1$ and n_0 be constants, and let $T(n)$ be defined as

$$T(n) = \begin{cases} t_n \in \Theta(1) & \text{for } n \leq n_0 \\ aT(n/b) + f(n) & \text{for } n > n_0, \end{cases}$$

where n/b can be interpreted both as $\lfloor n/b \rfloor$ and $\lceil n/b \rceil$. We have the following three distinct cases

1. $f(n) \in O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$. In this case, $T(n) \in \Theta(n^{\log_b a})$;
2. $f(n) \in \Theta(n^{\log_b a} \log^k n)$ for some $k \geq 0$. In this case, $T(n) \in \Theta(n^{\log_b a} \log^{k+1} n)$;
3. $f(n) \in \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and $af(n/b) \leq cf(n)$ for some $c < 1$ and n large enough. In this case, $T(n) \in \Theta(f(n))$.

Observe that the theorem does not encompass all possible recurrences but only a subset of those that express $T(n)$ as a function of only one value $T(n/b)$ (ignoring discrepancies due to floors and ceilings).

As a first example, function $T_c(n)$ defined by Equation 1, is solved by applying the second case of the theorem with $a = b = 2$ and $k = 0$. In the following, we will see other applications of the master theorem.

Example 1 Let

$$T(n) = \begin{cases} 1 & \text{for } n = 1 \\ 3T(n/4) + \sqrt{n} & \text{for } n > 1. \end{cases}$$

Since $1/2 = \log_4 3 - \epsilon$, for some $\epsilon > 0$, we can apply the first case of the master theorem, hence $T(n) \in \Theta(n^{\log_4 3})$.

Example 2 Let

$$T(n) = \begin{cases} \beta \in \Theta(1) & \text{for } n = 1 \\ 2T(n/4) + \sqrt{n} & \text{for } n > 1. \end{cases}$$

Since $1/2 = \log_4 2$, we can apply the second case of the master theorem with $k = 0$, hence $T(n) \in \Theta(\sqrt{n} \log n)$.

Example 3 (*Binary Search*) Let

$$T(n) = \begin{cases} \beta \in \Theta(1) & \text{for } n = 1 \\ T(n/2) + c & \text{for } n > 1, \end{cases}$$

with c a constant. Since $\log_2 1 = 0$ and $c \in \Theta(n^0)$, we can apply the second case of the master theorem with $k = 0$, hence $T(n) \in \Theta(\log n)$.

Example 4 Let

$$T(n) = \begin{cases} 3 & \text{for } n = 1 \\ 3T(n/2) + 3n^2 & \text{for } n > 1. \end{cases}$$

Since $2 = \log_2 3 + \epsilon$, for some $\epsilon > 0$, and $3(3(n/2)^2) = (3/4)3n^2$ for every $n \geq 0$, the hypotheses of the third case of the master theorem are satisfied (with $c = 3/4$), hence $T(n) \in \Theta(n^2)$.

3 Iterative method

If we need to determine the exact solution of a recurrence or to solve a recurrence not encompassed by the master theorem, we can apply the following iterative method. Let $T(n)$ be a function defined through a recurrence relation.

1. Starting from a value n sufficiently large, apply the definition of $T(n)$ a number of times until it is possible to *guess* a closed formula $f(n, i)$ for the right-hand-side of the equation obtained after i iterations of the definition of $T(n)$.
2. Determine the first index i_0 such that all occurrences of $T(\cdot)$ in $f(n, i_0)$ refer to base cases.
3. Obtain the final formula for $T(n)$ by substituting the appropriate base cases within $f(n, i_0)$.

As an example, let us use the above method to obtain an exact solution for $T_c(n)$ defined in Equation 1. For convenience, suppose that only values of n powers of two are considered. Therefore the recurrence can be rewritten as follows:

$$T_c(n) = \begin{cases} \beta & \text{for } n = 1 \\ 2T_c(n/2) + cn & \text{for } n = 2^d \text{ with } d > 0. \end{cases}$$

By repeatedly applying the definition we get:

$$\begin{aligned} T_c(n) &= 2T_c(n/2) + cn \\ &= 4T_c(n/4) + 2cn \\ &= 8T_c(n/8) + 3cn \\ &= \dots \\ &= 2^i T_c(n/2^i) + icn = f(n, i). \end{aligned}$$

We reach a base case with $i_0 = \log_2 n$, which yields:

$$T_c(n) = f(n, i_0) = nT(1) + cn \log_2 n = \beta n + cn \log_2 n.$$

Note that the formula $f(n, i)$ has been only “guessed”. In order to be entirely rigorous one should formally prove (for example by induction) that the final formula obtained for $T_c(n)$ is correct. Let us see other examples.

Example 5 The following recurrence is the same as the one in Example 1 limited to values of n that are powers of four.

$$T(n) = \begin{cases} 1 & \text{for } n = 1 \\ 3T(n/4) + \sqrt{n} & \text{for } n = 4^d \text{ with } d > 0. \end{cases}$$

By repeatedly applying the definition we get:

$$\begin{aligned} T(n) &= 3T(n/4) + \sqrt{n} \\ &= 9T(n/16) + (3/2)\sqrt{n} + \sqrt{n} \\ &= 27T(n/64) + (9/4)\sqrt{n} + (3/2)\sqrt{n} + \sqrt{n} \\ &= \dots \\ &= 3^i T(n/4^i) + \sqrt{n} \sum_{j=0}^{i-1} (3/2)^j = f(n, i). \end{aligned}$$

We reach a base case with $i_0 = \log_4 n$, which yields:

$$T(n) = f(n, i_0) = n^{\log_4 3} + \sqrt{n} \frac{(3/2)^{\log_4 n} - 1}{(3/2) - 1} = 3n^{\log_4 3} - 2\sqrt{n}.$$

Note that $3n^{\log_4 3} - 2\sqrt{n} \in \Theta(n^{\log_4 3})$, which is in accordance with the master theorem.

Example 6 We want to find an exact solution for the following recurrence which does not belong to the class of recurrences encompassed by the master theorem.

$$T(n) = \begin{cases} 1 & \text{for } 1 \leq n \leq 3 \\ T(n-3) + cn & \text{for } n > 3, \end{cases}$$

where $c > 0$ is a constant. By repeatedly applying the definition we get:

$$\begin{aligned} T(n) &= T(n-3) + cn \\ &= T(n-6) + c(n-3) + cn \end{aligned}$$

$$\begin{aligned}
&= T(n-9) + c(n-6) + c(n-3) + cn \\
&= \dots \\
&= T(n-3i) + c \sum_{j=0}^{i-1} (n-3j) = f(n, i).
\end{aligned}$$

Note that n can be written as $3m + a$, with $a \in [1, 3]$. We reach a base case with $i_0 = m$, which yields:

$$T(n) = f(n, i_0) = T(a) + c \sum_{j=0}^{m-1} (n-3j) = 1 + cmn - 3c \frac{(m-1)m}{2} \in \Theta(n^2)$$

4 Exercises

Ex.1 Let

$$T(n) = \begin{cases} 1 & \text{for } n = 1 \\ 2T(n/4) + 3n & \text{for } n > 1. \end{cases}$$

Find an asymptotic solution for $T(n)$ using the master theorem. For values of n powers of four, find an exact solution for $T(n)$ using the iterative method, and show the correctness of the solution by induction.

Ex.2 Let

$$T(n) = \begin{cases} 1 & \text{for } n = 1 \\ T(n/2) + \log_2 n & \text{for } n > 1. \end{cases}$$

Find an asymptotic solution for $T(n)$ using the master theorem. For values of n powers of two, find an exact solution for $T(n)$ using the iterative method, and show the correctness of the solution by induction.

References

[CLRS01] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms. Second Edition*. McGraw-Hill/MIT Press, Cambridge MA, 2001.