算法设计与分析第二章学习指南

视频
https://www.icourse163.org/course/HIT-356006
算法设计与分析(基础篇) 第二讲 2.1-2.3

阅读

算法导论(第三版) 第三章，4.3-4.5 节，4.6 节(选读)

练习题

(1) 证明或否证：$f(n)+o(f(n))=\Theta(f(n))$
(2) 证明：$\Theta(f(x)) + O(g(x)) = O(\max(f(x), g(x)))$.
(3) 证明或给出反例：$\Theta(f(n))\cap o(f(n))=\emptyset$
(4) 证明：设 k 是任意常数正整数，则 $\log^k n = o(n)$
(5) 证明：$\log n! = \Theta(n\log n)$
(6) 用迭代法解方程 $T(n)=T(9n/10)+n$
(7) 解方程 $T(n)=6T(n/3)+\log n$
(8) 解方程 $T(n) =3T(n/3 +5) + n/2$
(9) 解方程 $T(n)= T(\lceil n/2\rceil)+1$
(10)解方程 $T(n)=9T(n/3)+n$
(11)解方程 $T(n)=T(\lfloor n/2\rfloor)+n^3$
(12)证明或否证：如果 $f(n)=O(g(n))$，那么 $2^{f(n)}=O(2^{g(n)})$
(13)令 f, g 和 h 为定义在正整数上的正实数函数。假设 $f(n)=O(g(n))$，$g(n)=O(h(n))$，证明 $f(n)=O(h(n))$.
(14) 将下列函数按它们在 $n\to\infty$ 时的无穷大阶数，从小到大排序。
$n, n^{lg7}, n^{2.5}, n^3 lgn, n^2 lgn, nlgn$
(15)一个时间复杂性为 $O(n^2)$的算法的运行时间一定大于一个时间复杂性为 O(n)的算法么？为什么？

思考题

维基百科中的 Master 定理描述和算法导论不同，思考其差别以及哪个正确？(维基百科中的 Master 定理描述如下)

Generic form [edit]

The master theorem often yields asymptotically tight bounds to some recurrences from divide and conquer algorithms that partition an input into smaller subproblems of equal sizes, solve the subproblems recursively, and then combine the subproblem solutions to give a solution to the original problem. The time for such an algorithm can be expressed by adding the work that they perform at the top level of their recursion (to divide the problems into subproblems and then combine the subproblem solutions) together with the time made in the recursive calls of the algorithm. If $T(n)$ denotes the total time for the algorithm on an input of size n, and $f(n)$ denotes the amount of time taken at the top level of the recurrence then the time can be expressed by a recurrence relation that takes the form:

$$T(n) = a\,T\left(\frac{n}{b}\right) + f(n)$$

Here n is the size of an input problem, a is the number of subproblems in the recursion, and b is the factor by which the subproblem size is reduced in each recursive call. The theorem below also assumes that, as a base case for the recurrence, $T(n) = \Theta(1)$ when n is less than some bound $\kappa > 0$, the smallest input size that will lead to a recursive call.

Recurrences of this form often satisfy one of the three following regimes, based on how the work to split/recombine the problem $f(n)$ relates to the *critical exponent* $c_{\text{crit}} = \log_b a$. (The table below uses standard big O notation.)

$$c_{\text{crit}} = \log_b a = \log(\#\text{subproblems}) / \log(\text{relative subproblem size})$$

Case	Description	Condition on $f(n)$ in relation to c_{crit}, i.e. $\log_b a$	Master Theorem bound	Notational examples
1	Work to split/recombine a problem is dwarfed by subproblems. i.e. the recursion tree is leaf-heavy	When $f(n) = O(n^c)$ where $c < c_{\text{crit}}$ (upper-bounded by a lesser exponent polynomial)	... then $T(n) = \Theta\left(n^{c_{\text{crit}}}\right)$ (The splitting term does not appear; the recursive tree structure dominates.)	If $b = a^2$ and $f(n) = O(n^{1/2-\epsilon})$, then $T(n) = \Theta(n^{1/2})$.
2	Work to split/recombine a problem is comparable to subproblems.	When $f(n) = \Theta(n^{c_{\text{crit}}} \log^k n)$ for any $k \geq 0$ (rangebound by the critical-exponent polynomial, times zero or more optional $\log$s)	... then $T(n) = \Theta\left(n^{c_{\text{crit}}} \log^{k+1} n\right)$ (The bound is the splitting term, where the log is augmented by a single power.)	If $b = a^2$ and $f(n) = \Theta(n^{1/2})$, then $T(n) = \Theta(n^{1/2} \log n)$. If $b = a^2$ and $f(n) = \Theta(n^{1/2} \log n)$, then $T(n) = \Theta(n^{1/2} \log^2 n)$.
3	Work to split/recombine a problem dominates subproblems. i.e. the recursion tree is root-heavy.	When $f(n) = \Omega(n^c)$ where $c > c_{\text{crit}}$ (lower-bounded by a greater-exponent polynomial)	... this doesn't necessarily yield anything. If it is furthermore known that $af\left(\frac{n}{b}\right) \leq kf(n)$ for some constant $k < 1$ and sufficiently large n (often called the *regularity condition*) then the total is dominated by the splitting term $f(n)$: $T(n) = \Theta(f(n))$	If $b = a^2$ and $f(n) = \Omega(n^{1/2+\epsilon})$ and the regularity condition holds, then $T(n) = \Theta(f(n))$.

A useful extension of Case 2 handles all values of k:[3]

Case	Condition on $f(n)$ in relation to c_{crit}, i.e. $\log_b a$	Master Theorem bound	Notational examples
2a	When $f(n) = \Theta(n^{c_{\text{crit}}} \log^k n)$ for any $k > -1$	... then $T(n) = \Theta\left(n^{c_{\text{crit}}} \log^{k+1} n\right)$ (The bound is the splitting term, where the log is augmented by a single power.)	If $b = a^2$ and $f(n) = \Theta(n^{1/2} / \log^{1/2} n)$, then $T(n) = \Theta(n^{1/2} \log^{1/2} n)$.
2b	When $f(n) = \Theta(n^{c_{\text{crit}}} \log^k n)$ for $k = -1$	... then $T(n) = \Theta\left(n^{c_{\text{crit}}} \log\log n\right)$ (The bound is the splitting term, where the log reciprocal is replaced by an iterated log.)	If $b = a^2$ and $f(n) = \Theta(n^{1/2} / \log n)$, then $T(n) = \Theta(n^{1/2} \log\log n)$.
2c	When $f(n) = \Theta(n^{c_{\text{crit}}} \log^k n)$ for any $k < -1$	... then $T(n) = \Theta\left(n^{c_{\text{crit}}}\right)$ (The bound is the splitting term, where the log disappears.)	If $b = a^2$ and $f(n) = \Theta(n^{1/2} / \log^2 n)$, then $T(n) = \Theta(n^{1/2})$.