算法分析与设计: 第一次作业

PB20061372　朱云沁　Mar. 18, 2023

2.3-4 我们可以把插入排序表示为如下的一个递归过程. 为了排序 $A[1..n]$, 我们递归地排序 $A[1..n-1]$, 然后把 $A[n]$ 插入已排序的数组 $A[1..n-1]$. 为插入排序的这个递归版本的最坏情况运行时间写一个递归式.

解: 递归式为

$$T(n) = \begin{cases} \Theta(1) & n = 1, \\ T(n-1) + \Theta(n) & n > 1. \end{cases}$$

2.3-7 描述一个运行时间为 $\Theta(n \lg n)$ 的算法, 给定 n 个整数的集合 S 和另一个整数 x, 该算法能确定 S 中是否存在两个其和刚好为 x 的元素.

解: 伪代码如下:

Algorithm 1: 2.3-7

Input: a set S of n integers and an integer x
Output: a pair of elements in S whose sum is x

$A[1..n] \leftarrow \textsc{Sort}(S)$　　// 对 S 归并排序, 时间为 $\Theta(n \lg n)$
for $i = 1$ **to** n **do**
// 在 A 中二分搜索下标不为 i 的、值为 $x - A[i]$ 的元素, 时间为 $\Theta(\lg n)$
　if $\textsc{Search}(x - A[i], A, i)$ is successful
　　return true
return false

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

3.1-1 假设 $f(n)$ 与 $g(n)$ 都是渐近非负函数. 使用 Θ 记号的基本定义来证明 $\max(f(n), g(n)) = \Theta(f(n) + g(n))$.

解: 首先, 由于 $f(n)$ 与 $g(n)$ 渐近非负, $\max(f(n), g(n))$ 也渐近非负. 其次, 当 n 足够大时,

$$\frac{1}{2}(f(n) + g(n)) \leq \max(f(n), g(n)) \leq f(n) + g(n).$$

其中, 第一个不等式由 max 的定义得出, 第二个不等式由 $f(n), g(n) \geq 0$ 得出. 由 Θ 记号的定义知, $\max(f(n), g(n)) = \Theta(f(n) + g(n))$.

3.1-3 解释为什么 "算法 A 的运行时间至少是 $O(n^2)$" 这一表述是无意义的.

解: 从字面上看, 该命题说明算法 A 运行时间的一个下界的渐进上界为 n^2. 实际上, 对任意算法 A, 其运行时间至少为 $0 = O(n^2)$, 这意味着该命题总是成立, 因而这一表述是无意义的.