算法复杂度

Lecture 1

1. prove:

$$\max\{f(n), g(n)\} = \Theta(f(n)+g(n))$$

$\exists n_0$ st $f(n), g(n) \geq 0 \quad \forall n \geq n_0$

$\therefore \exists c_1 = 1, c_2 = 2, n_0' = n_0$, st

$$0 \leq c_1 \max\{f(n), g(n)\} \leq f(n)+g(n) \leq c_2 \max\{f(n), g(n)\}. \quad \forall n \geq n_0$$

$\therefore$ 成立. $\max\{f(n), g(n)\} = \Theta(f(n)+g(n))$

2. $O(n^2)$ 指的是一个算法时间复杂度的上界. 也就是说 $\exists c, n_0$, st
该算法 耗时 $f(n) \leq cn^2 \quad \forall n \geq n_0$
而 "at least" 是至少. 是下界概念.

Lecture 2.

3. (1) $T(n) = T(\frac{n}{2}) + n^3$ Case III. $T(n) = \Theta(n^3)$.

树:

$C_2 n^3$ $\qquad$ $C_2 n^3$

$C_2 (\frac{n}{2})^3$ $\qquad$ $\frac{1}{8} C_2 n^3$

$\vdots$

C_1

为 $C_1 + \frac{1}{1-\frac{1}{8}} C_2 n^3 = \frac{8}{7} C_2 n^3 + C_1 \sim O(n^3)$

(2) $T(n) = 4T(\frac{n}{3}) + n$ Case I. $T(n) = \Theta(n^{\log_3 4})$

树:

$C_2 n.$ $\qquad$ $C_2 n$

$C_2(\frac{n}{3})$ $C_2(\frac{n}{3})$ $C_2(\frac{n}{3})$ $C_2(\frac{n}{3})$ $\qquad$ $\frac{4}{3} C_2 n$

$\vdots$ $\qquad$ $\vdots$

$C_1 \cdots C_1 \cdots C_1 \cdots C_1.$

$$4^{\log_3 n} C_1 + C_2 n\left[1 + \frac{4}{3} + \cdots + \left(\frac{4}{3}\right)^{\log_3 n}\right]$$

$$= n^{1.26} C_1 + C_2 n \cdot \frac{1-(\frac{4}{3})^{\log_3 n}}{1-\frac{4}{3}} \approx C_1 n^{1.26} + C_2 n \cdot 3 \cdot n^{\log_3 \frac{4}{3}}$$

$$\Rightarrow O(n^{\log_3 4})$$

Date

4.5

(1). $T(n) = 2T(\frac{n}{4}) + 1 \Rightarrow O(n^{\log_4 2}) = O(\sqrt{n})$

(2) $T(n) = 2T(\frac{n}{4}) + \sqrt{n} \Rightarrow O(n^{\log_4 2}(\log n)^{0+1}) = O(\sqrt{n}\log n)$

(3) $T(n) = 2T(\frac{n}{4}) + n^2 \Rightarrow O(n^2)$