算法设计与分析-work2

yu wang

March 2024

1 （5 分）证明 $2n + \Theta(n^2) = \Theta(n^2)$

我们给定一个函数 n^2,用 $\Theta(n^2)$ 来表示一下函数集合 $\Theta(n^2) = \{f(n)|\exists c_1, c_2, n_0 > 0, \forall n \geq n_0, 0 \leq c_1 n^2 \leq f(n) \leq c_2 n^2\}$, 则函数 f(n) 属于 $\Theta(n^2)$. 那么对 2n+$\Theta(n^2)$, 我们可以假设 f(n) 属于 $\Theta(n^2)$，我们可以找到 $n_1 > 0$,，使得对于所有的 n>n_1, 有 2n+$c_1 n^2$ ≤f(n)+2n, 那么 $c_1 n^2$ ≤2n+$\Theta(n^2)$, 下界得证。对于上界，我们已知 $\Theta(n^2) \leq c_2 n^2$. 于是，我们可以令 n_1 >2, 则对于 n>n_1，有 $2n \leq n^2$，所以 2n+$\Theta(n^2) \leq$（c_2+1）n^2. 所以存在 n_1>2>0,$c_1, c_2 > 0$, 对于任意的 $n \geq n_1$, 有 $c_1 n^2 \leq 2n + \Theta(n^2) \leq c_2 n^2 + n^2$, 则 $2n + \Theta(n^2)$ 属于 $\Theta(n^2)$，命题得证

2 （5 分）解递归式 $T(n) = 2T(n/2) + 1$

对于该递归式,我们有 f(n)=1,a=2,b=2,$n^{log_b a} = n$, , 于是又 f(n)=1=O($n^{1-\epsilon}$), 我们可以取 ϵ=0.3, 于是该题符合主定理情况一,于是有 T(n)=$\Theta(n^{log_b a})$=$\Theta(n)$

3 （5 分）解递归式 $T(n) = 4T(n/2) + n^2$

对于该递归式,我们有 f(n)=n^2,a=4,b=2,$n^{log_b a} = n^2$, 于是有 f(n)=n^2=$n^{log_b a}$, 该题符合主定理情况二，于是有 T(n)=$\Theta(n^{log_b a} lgn)$=$\Theta(n^2 lgn)$

4 （5 分）解递归式 $T(n) = 2T(n/2) + n^2$

对于该递归式,我们有 f(n)=n^2,a=2,b=2,$n^{log_b a} = n$, 于是有 f(n)=n^2=$\Omega(n^{1+\epsilon})$, 我们可以取 Ω 为 0.9. 接下来我们要判断是不是符合正则条件,如果符合则可以适用主定理的第 3 个情况,由于对于足够大的 n, 有 $af(n/b) = 2f(n/2) = 2\frac{n}{2}\frac{n}{2} = \frac{n^2}{2} \leq \frac{3}{4}n^2$, 于是有 c=$\frac{3}{4}$, 因此符合正则条件，可以适用主定理的第三个情况，于是有 T(n)=$\Theta(n^2)$

5 （5 分）解递归式 $T(n) = 2T(n/2) + nlgn$

对于该递归式,我们有 f(n)=nlgn,a=2,b=2,$n^{log_b a} = n$,, 于是 f(n)=nlgn=$\Omega(n)$, 但我们不能适用主定理的第三个情况，原因是不存在 $\epsilon > 0$ 使得 $nlgn = \Omega(n^{1+\epsilon})$ ，因为假设存在 $\epsilon > 0, c > 0$, 使得 $nlgn \geq cn^{1+\epsilon}$, 我们有:

$$nlgn \geq cn^{1+\epsilon} \Leftrightarrow lgn \geq cn^{\epsilon} \Leftrightarrow c \leq \frac{lgn}{n^{\epsilon}}$$

然而由洛必达法则:

$$lim_{n\to\infty}\frac{lgn}{n^{\epsilon}} = lim_{n\to\infty}\frac{1/n}{\epsilon n^{\epsilon-1}} = lim_{n\to\infty}\frac{1}{\epsilon n^{\epsilon}} = 0$$

即 $c \leq 0$ ，这就出现了矛盾，所以不存在 $c > 0, n > n_0$ ，使得 $nlgn \geq cn^{1+\epsilon}$ 。因此对于这个递归式，我们不能适用主定理，应该使用递归树法来求解。

如果我们将该递归树完全展开，并对递归树求和，我们可以发现递归数的每一层的和都是 $\Omega(n)$,，递归树的高度是 $log_2 n$, 因此递归数的综合为 $\Theta(nlgn)$,，即递归算法的时间复杂度是 $\Theta(nlgn)$

6 （5 分）解递归式 $T(n) = 2T(\sqrt{n}) + n$

我们令 e^x=n, 则 $\sqrt{n}$=$e^{\frac{x}{2}}$. 则我们可以令 W(x)=T(e^x),W(x)=2W($\frac{x}{2}$)+x. 对于该递归式，我们有 f(x)=x,a=2,b=2,$x^{log_b a} = x$, 于是有 f(x)=$x^{log_b a} = x$, 该题符合主定理情况二，于是有 T(n)=$\Theta(n^{log_b a} lgn)$=$\Theta(nlgn)$

7 （10 分）解递归式 $nT(n) = (n-2)T(n-1) + 2$

. 对于该递归式，我们将其化简得 $T(n) = \frac{n-2}{n}T(n-1) + \frac{2}{n}$, 我们可以知道该递归式不满足主定理式子。我们尝试发现其中的一些规律，T（1）未

知，T（2）=1，T（3）=$\frac{1}{3}T(2)+\frac{2}{3}$=1,T(4）=$\frac{1}{6}T(3)+\frac{5}{6}$=1，..... 如此下去，我们可以猜想 T（n）的为常数 1。下面我们运用强归纳公理来证明猜想。

首先我们确定谓词 P（n)：T（n）的时间复杂度为常数 1.

2，证明基本情况 P（1)，从所给的递归式中，我们可以假设 T（1）的时间复杂度为 1

3，证明一般情况 $\forall n \in N(P(0) \wedge P(1) \wedge \cdots \wedge P(n) \Rightarrow P(n+1)),$，我们假设 T（1)，T（2)，T（3）.....T（n）的时间复杂度都是 1，则对于 n+1 的情况下，我们由递归式可以得 (n+1)T（n+1）=(n-1)T(n)+2=n-1+2=n+1, 所以我们可以求得 T（n+1）=1, 于是命题得证，我们可以解出递归式的时间复杂度为常数 1