算法与复杂性　作业四

516021910528 - SHEN Jiamin

2020 年 3 月 12 日

1. 证明

$$\sum_{k=1}^{n} \frac{1}{k} = \Theta(\log n)$$

证明　由 $\frac{1}{k}$ 单调递减

$$\int_1^{n+1} \frac{1}{x}\mathrm{d}x \leq \sum_{k=1}^{n} \frac{1}{k} \leq 1 + \int_1^{n} \frac{1}{x}\mathrm{d}x$$

$$\ln(n+1) - \ln 1 \leq \sum_{k=1}^{n} \frac{1}{k} \leq 1 + \ln n - \ln 1$$

$$\ln(n) < \ln(n+1) \leq \sum_{k=1}^{n} \frac{1}{k} \leq 1 + \ln n$$

所以有

$$\Omega(\log n) = \sum_{k=1}^{n} \frac{1}{k} = O(\log n)$$

即

$$\sum_{k=1}^{n} \frac{1}{k} = \Theta(\log n)$$

□

2. 设有如下递推关系

$$T(n) = \begin{cases} T(\frac{n}{2}) + 1 & ,n\text{为偶数} \\ 2T(\frac{n-1}{2}) & ,n\text{为奇数} \end{cases}$$

其中 $T(1) = 1$

(a) 证明当 $n = 2^k$ 时，$T(n) = O(\log n)$

证明

$$\begin{aligned} T(n) = T(2^k) = T(\frac{n}{2}) + 1 = T(2^{k-1}) + 1 \\ = T(2^{k-2}) + 2 = \dots \\ = T(1) + k - 1 = k = \log n \end{aligned}$$

所以

$$T(n) = O(\log n)$$

□

(b) 证明存在无穷集合 X，当 $n \in X$ 时，$T(n) = \Omega(n)$

令 $a_1 = 1, a_n = 2a_{n-1} + 1 \Rightarrow a_n = 2^n - 1$，则无穷集合 $X = \{2^k - 1 \mid k \in \mathbb{N}^*\}$。

证明

$$\begin{aligned} T(n) = T(2^k - 1) = 2 \cdot T(\frac{n-1}{2}) + 1 = 2 \cdot T(2^{k-1} - 1) \\ = 4 \cdot T(2^{k-2} - 1) = \dots \\ = 2^{k-1} \cdot T(2 - 1) = 2^{k-1} \end{aligned}$$

$\exists c = \frac{1}{2}, N = 1$ 使得 $\forall n > N, n \in X$

$$T(n) = T(2^k - 1) = 2^{k-1} > 2^{k-1} - \frac{1}{2} = \frac{2^k - 1}{2} = \frac{n}{2}$$

所以

$$T(n) = \Omega(n)$$

□

(c) 以上两个结论说明了什么?

一个算法在输入具有不同特征时，可能具有不同的时间复杂度。
最佳状况和最差状况可能有不同的时间复杂度。