1.时间复杂度的渐近分析

以下每个小题中两个函数分别为$f(n)$和$g(n)$。在渐近分析中, 可以写成$f(n) = \ldots(g(n))$的形式。请计算并选择相应的$\mathcal{O}$、Θ、Ω符号。

(1) $f(n) = n^{\frac{1}{2}}$ $g(n) = (\log 2n)^5$

$$\begin{aligned}
\lim_{n\to+\infty} \frac{(\log 2n)^5}{n^{\frac{1}{2}}} &\triangleq \lim_{n\to+\infty} \frac{5\times 2\cdot(\log 2n)^4\cdot\frac{1}{2n\cdot\ln 2}}{\frac{1}{2}n^{-\frac{1}{2}}} \\
&= \lim_{n\to+\infty} \frac{5\cdot(\log 2n)^4\cdot\frac{1}{\ln 2}}{\frac{1}{2}n^{\frac{1}{2}}} \\
&\triangleq \lim_{n\to+\infty} \frac{5\times 4\times 2\cdot(\log 2n)^3\cdot\frac{1}{2n}\cdot\frac{1}{\ln 2}\cdot\frac{1}{\ln 2}}{\frac{1}{2}\times\frac{1}{2}n^{-\frac{1}{2}}} \\
&= \lim_{n\to+\infty} \frac{5\times 4\cdot(\log 2n)^3\cdot(\frac{1}{\ln 2})^2}{\frac{1}{2}\times\frac{1}{2}n^{\frac{1}{2}}} \\
&\triangleq \ldots \\
&\triangleq \lim_{n\to+\infty} \frac{5\times 4\times 3\times 2\cdot(\frac{1}{\ln 2})^5}{(\frac{1}{2})^5\cdot n^{\frac{1}{2}}} \\
&= 0
\end{aligned} \tag{1}$$

上式中，等号上的△表示此等号右方使用了一次洛必达法则(L'Hospital Rule)。

$$\lim_{n\to+\infty} \frac{(\log 2n)^5}{n^{\frac{1}{2}}} = 0 \tag{2}$$

即:

$$n^{\frac{1}{2}} = \Omega((\log 2n)^5) \tag{3}$$

(2) $f(n) = (\log n)^{2\log n}$ $g(n) = n^3$

首先利用换元法，令$\log n = t$ ，则$n = 2^t$ ，$f(n) = t^{2t}$ ，$g(n) = 2^{3t}$

则:

$$\begin{aligned}
\lim_{n\to+\infty} \frac{n^3}{(\log n)^{2\log n}} &= \lim_{t\to+\infty} \frac{2^{3t}}{t^{2t}} \\
&= \lim_{t\to+\infty} (\frac{8}{t^2})^t \\
&= 0
\end{aligned} \tag{4}$$

$$\lim_{n\to+\infty} \frac{n^3}{(\log n)^{2\log n}} = 0 \tag{5}$$

即:

$$(\log n)^{2\log n} = \Omega(n^3) \tag{6}$$

(3) $f(n) = n^2$ $g(n) = 3^{2\log n}$

此题类似于(2)，利用换元法，令$\log n = t$ ，则$n = 2^t$ ，$f(n) = 2^{2t}$ ，$g(n) = 3^{2t}$

则：

$$\begin{aligned}\lim_{n\to+\infty}\frac{3^{2\log n}}{n^2}&=\lim_{t\to+\infty}\frac{3^{2t}}{2^{2t}}\\&=\lim_{t\to+\infty}1.5^{2t}\\&=+\infty\end{aligned}\tag{7}$$

$$\lim_{n\to+\infty}\frac{3^{2\log n}}{n^2}=+\infty\tag{8}$$

即：

$$n^2=\mathcal{O}(3^{2\log n})\tag{9}$$

(4) $f(n)=n!!\ g(n)=2^n$

- n为偶数时

$$\begin{aligned}\lim_{n\to+\infty}\frac{2^n}{n!!}&=\lim_{t\to+\infty}\frac{4^{\frac{n}{2}}}{n(n-2)(n-4)\dots4\times2}\\&=\lim_{n\to+\infty}\frac{4}{n}\cdot\frac{4}{n-2}\cdot\frac{4}{n-4}\dots\frac{4}{4}\times\frac{4}{2}\end{aligned}\tag{10}$$

观察上式，由于$n\to+\infty$ ，$\frac{4}{n}\to+\infty$且除最后两项外其他所有项均小于1

由此可得：

$$\lim_{n\to+\infty}\frac{2^n}{n!!}=0\tag{11}$$

- n为奇数时

用相同方法

$$\begin{aligned}\lim_{n\to+\infty}\frac{2^n}{n!!}&=\lim_{t\to+\infty}\frac{4^{\frac{n}{2}}}{n(n-1)(n-3)\dots3\times1}\\&=\lim_{n\to+\infty}\frac{4}{n}\cdot\frac{4}{n-1}\cdot\frac{4}{n-3}\dots\frac{4}{3}\times\frac{4}{1}\end{aligned}\tag{12}$$

观察上式，由于$n\to+\infty$ ，$\frac{4}{n}\to+\infty$且除最后两项外其他所有项均小于1

由此可得：

$$\lim_{n\to+\infty}\frac{2^n}{n!!}=0\tag{13}$$

综上：

$$\lim_{n\to+\infty}\frac{2^n}{n!!}=0\tag{14}$$

即：

$$n!!=\Omega(2^n)\tag{15}$$

2.递归算法的时间复杂度

参考第1.3小节里的推导过程和结果，完成以下小题：

(1) 如果$f(n)=n^{\log_b a+\epsilon}$ ，且$0<\epsilon\ll1$。请推导$T(n)$。

根据第1.3小节推导内容，把规模大小为的问题平均分成b(⩾2)份，然后在这些子问题上迭代地使用分治算法。在构造问题的解时，我们选择$a\in\{1,\cdots,b\}$个子问题的解进行合并。此时子问题解

的合并过程耗时为$\mathcal{O}(f(n))$。则递归算法的时间复杂度函数$T(n)$可以写成

$$T(n) = aT\left(\frac{n}{b}\right) + \mathcal{O}(f(n)) \tag{16}$$

递推公式以得到更小的n，直到到达递归基，也就是对应着n=1的情形。

$$\begin{aligned}
T(n) &= aT\left(\frac{n}{b}\right) + \mathcal{O}(f(n)) \\
&= a\left[aT\left(\frac{n}{b^2}\right) + \mathcal{O}\left(f\left(\frac{n}{b}\right)\right)\right] + \mathcal{O}(f(n)) \\
&= a^2T\left(\frac{n}{b^2}\right) + a\mathcal{O}\left(f\left(\frac{n}{b}\right)\right) + \mathcal{O}(f(n)) \\
&= a^2\left[aT\left(\frac{n}{b^3}\right) + \mathcal{O}\left(f\left(\frac{n}{b^2}\right)\right)\right] + a\mathcal{O}\left(f\left(\frac{n}{b}\right)\right) + \mathcal{O}(f(n)) \\
&= a^3T\left(\frac{n}{b^3}\right) + a^2\mathcal{O}\left(f\left(\frac{n}{b^2}\right)\right) + a\mathcal{O}\left(f\left(\frac{n}{b}\right)\right) + \mathcal{O}(f(n)) \\
&= \cdots \\
&= a^tT\left(\frac{n}{b^t}\right) + \sum_{i=0}^{t-1} a^i\mathcal{O}\left(f\left(\frac{n}{b^i}\right)\right)
\end{aligned} \tag{17}$$

上式中，$t \in \{0, 1, 2, \cdots\}$为递推步骤的序号。

对于足够大的t,我们有n/b=1,即$t = \log_b n$。此时对应着算法操作到了单个元素的层次。同时我们假定递推基$T(1) = \Theta(1)$，即为常数操作时间。则：

$$T(n) = a^{\log_b n}T(1) + \sum_{i=0}^{\log_b n-1} a^i\mathcal{O}\left(f\left(\frac{n}{b^i}\right)\right) \tag{18}$$

利用等式

$$a^{\log_b n} = n^{\log_n a \cdot \log_b n} = n^{\frac{\ln a}{\ln n} \cdot \frac{\ln n}{\ln b}} = n^{\frac{\ln a}{\ln b}} = n^{\log_b a} \tag{19}$$

得到

$$T(n) = n^{\log_b a}T(1) + \sum_{i=0}^{\log_b n-1} a^i\mathcal{O}\left(f\left(\frac{n}{b^i}\right)\right) \tag{20}$$

代入$f(n)=n^{\log_b a+\epsilon}$得

$$
\begin{aligned}
T(n) &= n^{\log_b a}T(1) + \sum_{i=0}^{\log_b n-1} a^i \mathcal{O}\left[\left(\frac{n}{b^i}\right)^{\log_b a+\epsilon}\right] \\
&= n^{\log_b a}T(1) + \sum_{i=0}^{\log_b n-1} a^i \mathcal{O}\left[\frac{n^{\log_b a+\epsilon}}{b^{i(\log_b a+\epsilon)}}\right] \\
&= n^{\log_b a}T(1) + \sum_{i=0}^{\log_b n-1} \cancel{a}^i \mathcal{O}\left(\frac{n^{\log_b a+\epsilon}}{\cancel{a}^i \cdot b^{i\epsilon}}\right) \\
&= n^{\log_b a}T(1) + \sum_{i=0}^{\log_b n-1} \mathcal{O}\left(\frac{n^{\log_b a+\epsilon}}{b^{i\epsilon}}\right) \\
&= n^{\log_b a}T(1) + \mathcal{O}\left(n^{\log_b a+\epsilon} \cdot \sum_{i=0}^{\log_b n-1} b^{-i\epsilon}\right)
\end{aligned}
\tag{21}
$$

上式中的级数求和为

$$
\sum_{i=0}^{\log_b n-1} b^{-i\epsilon} = \frac{1-b^{-\epsilon\log_b n}}{1-b^{-\epsilon}} = \frac{1-n^{-\epsilon}}{1-b^{-\epsilon}} \tag{22}
$$

因为$-\epsilon<0$，当n很大时，$\lim_{n\to+\infty}(1-n^{-\epsilon})=1$，则:

$$
\begin{aligned}
T(n) &= n^{\log_b a}T(1) + \mathcal{O}\left(n^{\log_b a+\epsilon} \cdot \frac{1}{1-b^{-\epsilon}}\right) \\
&= n^{\log_b a}T(1) + \mathcal{O}\left(n^{\log_b a} \cdot (b-1)\right) \\
&= \mathcal{O}\left(n^{\log_b a}\right)
\end{aligned}
\tag{23}
$$

(2) 在二分查找算法里，如果$T(n)=T(\frac{n}{2})+c$，其中c为某一常整数。请推导$T(n)$

$$
\begin{aligned}
T(n) &= T\left(\frac{n}{2}\right) + \mathcal{O}(c) \\
&= T\left(\frac{n}{4}\right) + \mathcal{O}(c) + \mathcal{O}(c) \\
&= T\left(\frac{n}{8}\right) + \mathcal{O}(c) + \mathcal{O}(c) + \mathcal{O}(c) \\
&= \cdots \\
&= T\left(\frac{n}{2^t}\right) + \sum_{i=0}^{t-1} 1^i \mathcal{O}(c)
\end{aligned}
\tag{24}
$$

上式中，$t\in\{0,1,2,\cdots\}$为递推步骤的序号。

对于足够大的t,我们有$\frac{n}{2^t}=1$,即$t=\log n$。此时对应着算法操作到了单个元素的层次。同时我们

假定递推基$T(1)=\Theta(1)$，即为常数操作时间。则：

$$
\begin{aligned}
T(n) &= T(1) + \sum_{i=0}^{\log n-1} 1^i \mathcal{O}(c) \\
&= T(1) + \log n \mathcal{O}\left(c\right) \\
&= \mathcal{O}\left(\log n\right)
\end{aligned}
\tag{25}
$$

对于上式中的级数求和

由于1的任意次方都是1，上式中的$\sum_{i=0}^{\log n-1} 1^i$ 可视作0到$\log_2 n-1$ ，一共$\log n$个$\mathcal{O}(1)$相加。

二分查找算法的递归树一共有$\log n$层，每一层时间复杂度为c，时间复杂度即为$\mathcal{O}\left(\log n\right)$

(3) 在归并排序算法里，如果$T(n)=2T(\frac{n}{2})+cn+d$ ，其中c和d均为常整数。请推导$T(n)$

$$
\begin{aligned}
T(n) &= 2T\left(\frac{n}{2}\right) + \mathcal{O}(cn) + \mathcal{O}(d) \\
&= 2\left[2T\left(\frac{n}{2^2}\right) + \mathcal{O}(c\frac{n}{2}) + \mathcal{O}(d)\right] + \mathcal{O}(cn) + \mathcal{O}(d) \\
&= 2^2T\left(\frac{n}{2^2}\right) + \mathcal{O}(cn) + 2\mathcal{O}(d) + \mathcal{O}(cn) + \mathcal{O}(d) \\
&= \cdots \\
&= 2^tT\left(\frac{n}{2^t}\right) + \sum_{i=0}^{t-1} 1^i \mathcal{O}(cn) + \sum_{i=0}^{t-1} 2^i \mathcal{O}(d)
\end{aligned}
\tag{26}
$$

上式中，$t\in\{0,1,2,\cdots\}$为递推步骤的序号。

对于足够大的t,我们有$\frac{n}{2^t}=1$,即$t=\log n$。此时对应着算法操作到了单个元素的层次。同时我们假定递推基$T(1)=\Theta(1)$，即为常数操作时间。则：

$$
T(n) = 2^{\log n}T(1) + \sum_{i=0}^{\log n-1} 1^i \mathcal{O}(cn) + \sum_{i=0}^{\log n-1} 2^i \mathcal{O}(d) \tag{27}
$$

利用等式

$$
a^{\log_b n} = n^{\log_n a \cdot \log_b n} = n^{\frac{\ln a}{\ln n}\cdot\frac{\ln n}{\ln b}} = n^{\frac{\ln a}{\ln b}} = n^{\log_b a} \tag{28}
$$

得到

$$
\begin{aligned}
T(n) &= n^{\log 2}T(1) + \sum_{i=0}^{\log n-1} 1^i \mathcal{O}(cn) + \sum_{i=0}^{\log n-1} 2^i \mathcal{O}(d) \\
&= nT(1) + \log n \mathcal{O}\left(cn\right) + (n-1)\mathcal{O}(d) \\
&= \mathcal{O}(n\log n)
\end{aligned}
\tag{29}
$$

对于上式中的级数求和

由于1的任意次方都是1，上式中的$\sum_{i=0}^{\log n-1} 1^i$ 可视作0到$\log_2 n-1$ ，一共$\log n$个$\mathcal{O}(1)$相加。

$$
\sum_{i=0}^{\log n-1} 2^i = \frac{2^{\log n}-1}{2-1} = n-1 \tag{30}
$$

归并排序算法的递归树一共有$\log n$层，每一层时间复杂度为n，时间复杂度即为$\mathcal{O}(n\log n)$