渐进记号

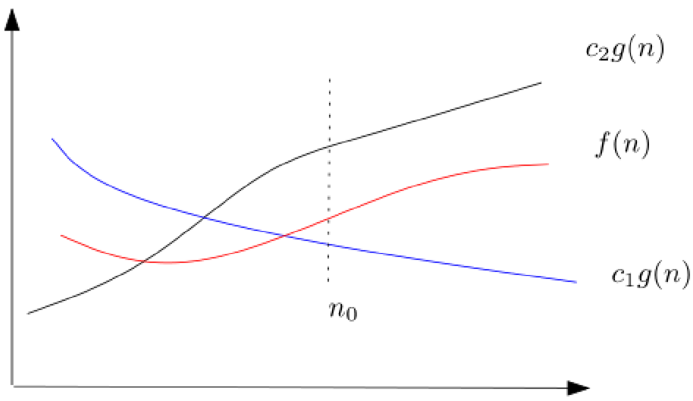
用来表示算法的渐进运行时间的记号是用定义域为自然数集$N=\{0,1,2,...\}$的函数来定义的，这些记号便于用来表示最坏情况运行时间$T(n)$，因为$T(n)$一般仅定义于整数的输入规模上。

Θ记号（紧渐进界）

对于Θ记号有如下的定义：

Θ-notation

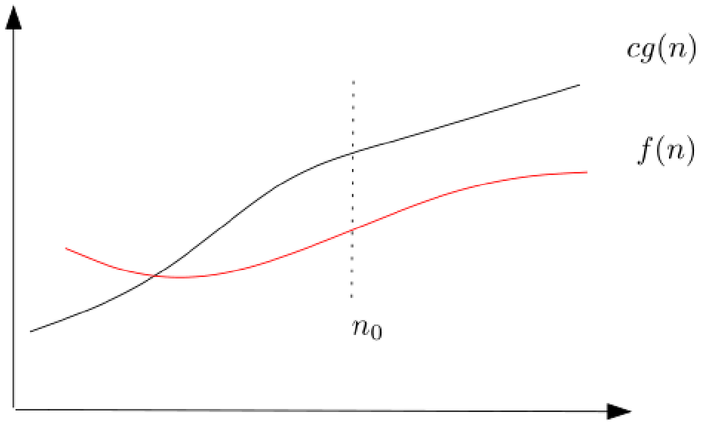
- $f(n) = \Theta(g(n))$
- $\exists c_1, c_2\ \exists n_0$ such that, for all $n \geq n_0$, we have $c_1 g(n) \leq f(n) \leq c_2 g(n)$, where c_1, c_2 are non-negative constants。



Θ记号限制一个函数在常数因子内，如图所示，n_0是最小的可能值。如果存在正常数n_0, c_1, c_2使得在n_0右边$f(n)$的值永远在$c_1 g(n)$与$c_2 g(n)$之间，那么可以写成$f(n)=\Theta(g(n))$。

O记号（渐进上界）

对于O记号有如下的定义：

- 

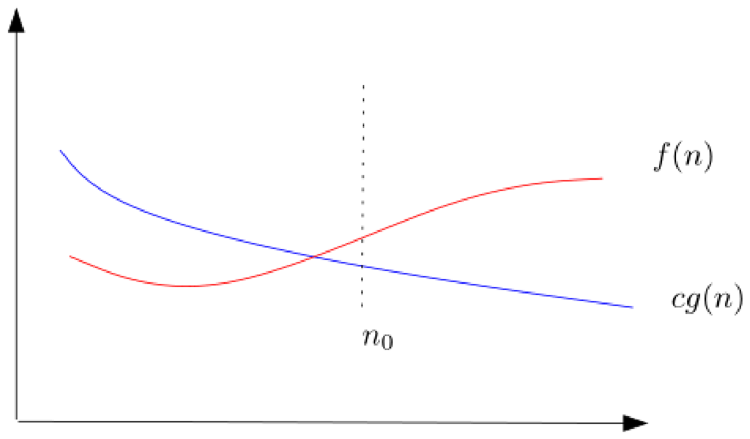
- $$\boxed{O(g(n)) = \{f(n) : \exists \;\; constant \;\; c, n_0, s.t. \;\; 0 \le f(n) \le c(g(n)) \;\; \forall n \ge n_0\}.}$$

O记号给出一个函数在常数因子内的上限。如图所示，n_0是最小的可能值。如果存在正常数n_0, c使得在n_0右边$f(n)$的值永远等于或小于$cg(n)$，那么可以写成$f(n) = O(g(n))$。

Ω记号（渐进下界）

对于Ω记号有如下的定义：

- $\Omega(g(n)) = \{f(n) : \exists \ \ constant \ \ c, n_0 \ \ s.t. \ \ 0 \le cg(n) \le f(n), \ \ \forall n \ge n_0\}.$



-

Ω记号给出一个函数在常数因子内的下限。如图所示，n_0是最小的可能值。如果存在正常数n_0, c使得在n_0右边$f(n)$的值永远等于或大于$cg(n)$，那么可以写成$f(n) = \Omega(g(n))$。

o记号（渐进非紧上界）

O记号所提供的渐进上界可能是紧的，但也有可能不是。例如，$2n^2 = O(n^2)$是一个紧的上界，但$2n = O(n^2)$却不是一个紧的上界。于是，我们使用o记号来表示一个紧的上界。

对于o记号有如下的定义：

- $o(g(n)) = \{f(n) : \ \ for \ \ any \ \ positive \ \ constant \ \ c > 0, \exists n_0 > 0, \ \ s.t. \ \ 0 \le f(n) < cg(n), \forall \ \ n \ge n_0\}.$

例如，$2n = o(n^2)$，但$2n^2 \neq o(n^2)$。

O记号与o记号的定义是类似的，主要区别在于对于$f(n) = O(g(n))$，界$0 \le f(n) \le cg(n)$对某个常数$c > 0$成立即可，而对于$f(n) = o(g(n))$，界$0 \le f(n) \le cg(n)$对所有常数$c > 0$成立。

ω记号（渐进非紧下界）

ω记号与Ω记号的关系就与前面小o和大o之间的关系是类似的，我们用ω记号表示一个紧的下界。

对于ω记号有如下的定义：

- $\omega(g(n)) = \{f(n): \ for\ any\ positive\ constant\ c > 0, \exists n_0 > 0, \ s.t.\ 0 \le cg(n) < f(n), \forall\ n \ge n_0\}.$

例如，$n^2/2 = \omega(n)$，但$n^2/2 \neq \omega(n^2)$。

函数间的比较

实数的许多关系属性可以用于渐进比较，以上的记号之间具有传递性和对称性，下面假设$f(n)$和$g(n)$是渐进正值函数。

- $f(n) = \Theta(g(n)), g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n))$
- $f(n) = O(g(n)), g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$
- $f(n) = o(g(n)), g(n) = o(h(n)) \Rightarrow f(n) = o(h(n))$
- $f(n) = \Omega(g(n)), g(n) = \Omega(h(n)) \Rightarrow f(n) = \Omega(h(n))$
- $f(n) = \omega(g(n)), g(n) = \omega(h(n)) \Rightarrow f(n) = \omega(h(n))$
- $f(n) = \Theta(g(n))$ iff $\Theta(f(n)) = g(n)$

解递归式的三种方法

求解递归式，即找出解的渐进“Θ”或“O”界的方法主要有三种：

- **代换法**：先猜某个界存在，然后用数学归纳法证明该猜测的正确性。

- **递归树方法**：将递归式转换成树形结构，树中的节点代表在不同递归层次付出的代价。
- **主方法**：给出递归形式$T(n) = aT(n/b) + f(n)$的界，其中$a \geq 1, b > 1$，$f(n)$是给定的函数。这种方法要记忆三种情况，就可以确定很多简单递归式的界了。

代换法

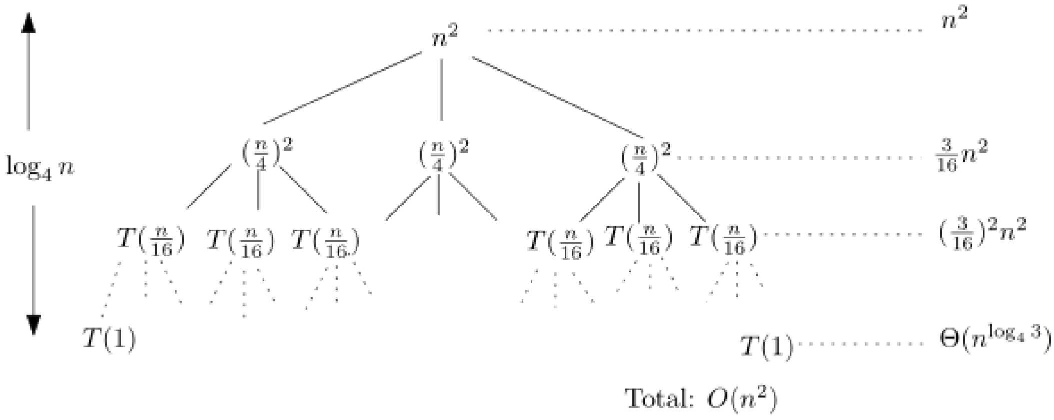
用代换法解递归式需要两个步骤：

1. 猜测解的形式。
2. 用数学归纳法找出使解真正有效地常数。

递归树方法

虽然代换法给递归式的解的正确性提供了一种简单的证明方法，但是有的时候很难得到一个好的猜测。此时，画出一个递归树是一种得到好猜测的直接方法。

设$T(n) = 3T(n/4) + n^2$，则使用递归树求解该递归式的过程如下图所示：



$$
\begin{aligned}
T(n) &= n^2 + \frac{3}{16}n^2 + (\frac{3}{16})^2 n^2 + \cdots + (\frac{3}{16})^{\log_4 n - 1} n^2 + \Theta(n^{\log_4 3}) \\
&= \sum_{i=0}^{\log_4 n - 1} (\frac{3}{16})^i n^2 + \Theta(n^{\log_4 3}) \\
&= ((3/16)^{\log_4 n} - 1)/(3/16 - 1) n^2 + \Theta(n^{\log_4 3})
\end{aligned}
$$

主方法

设$a \geq 1, b > 1$，$f(n)$为一函数，$T(n)$由递归式

$$T(n) = aT(n/b) + f(n)$$

对非负整数定义，那么$T(n)$有如下的渐进界：

- If $af(n/b) = Kf(n)$, for some constant $K > 1$, then $T(n) = \Theta(n^{\log_b a})$.
- If $af(n/b) = f(n)$, then $T(n) = \Theta(f(n) \log_b n)$.
- If $af(n/b) = \kappa f(n)$, for some constant $\kappa < 1$, then $T(n) = \Theta(f(n))$.

求解和式时有一个比较常用的公式，假设$f(k)$是单调递增的函数，那么有如下的性质：

$$\int_{m}^{n+1} f(x)dx \leq \sum_{k=m}^{n} f(k) \leq \int_{m-1}^{n} f(x)dx$$