MIT-lecture01-http-link(click here)

1 Peak Finding

1.0.1 新的分治法快了多少?

- compare n and $log_2(n)$.
- Transform

$$log_2(n) = b \Rightarrow n = 2^b$$

we map $log_2(n) \mapsto b, n \mapsto 2^b$,compare 2^b and b

由此可见这是一个巨大的算法提升。（参见mathematica图形）

2 渐进符号

渐进符号一共有5个$(O, \Omega, \Theta, o, \omega)$，其中3个$(O, \Omega, \Theta)$最为重要也最为常用。

- $O(g(n))$

$$O(g(n)) = \{f(n) : f(n) \leqslant cg(n)\}, c > 0 \text{ is a constant}$$

- $\Theta(g(n))$

$$\Theta(g(n)) = \{f(n) : c_1 g(n) \leqslant f(n) \leqslant c_2 g(n)\}, c_1 > 0, c_2 > 0, \text{are constants}$$

在分析算法复杂度的时候我们使用$f(n) = \Theta(g(n))$，其实是说$f(n) \in \Theta(g(n))$

- $\Omega(g(n))$

$$\Omega(g(n)) = \{f(n) : cg(n) \leqslant f(n)\}, c > 0 \text{ is a constant}$$

- $o(g(n))$

$$O(g(n)) = \{f(n) : f(n) < cg(n)\}, c > 0 \text{ is a constant}, \lim_{n\to\infty} \frac{g(n)}{f(n)} = \infty$$

- $\omega(g(n))$

$$\omega(g(n)) = \{f(n) : cg(n) < f(n)\}, c > 0 \text{ is a constant } \lim_{n\to\infty} \frac{g(n))}{f(n)} = 0$$

2.1 类比记忆

$f(n) = O(g(n))$	$f(n) \leqslant g(n)$
$f(n) = \Theta(g(n))$	$f(n) = g(n)$
$f(n) = \Omega(g(n))$	$f(n) \geqslant g(n)$
$f(n) = o(g(n))$	$f(n) < g(n)$
$f(n) = \omega(g(n))$	$f(n) > g(n)$