算法复习——Basic Skill

证明算法正确性：`Loop-invariants`

分治法

分治法每层递归都有三个步骤：

分解：将原问题分解成若干个规模较小，相互独立，与原问题形式相同的字问题；

解决：若干个子问题规模较小而容易被解决的则直接解决，否则递归地解决子问题；

合并：将各个子问题的解合并为原问题的

纠结的时间复杂度问题

FIRST ONE：BIG-O :表示上界，你怎么追都追不上我~~

$O(g(n)) = f(n)$ means: 当$n > n_0$ 时，存在$c > 0$

$$0 \leqslant f(n) \leqslant cg(n)$$

注意：当我们说，一个东东的运行时间是$O(n^2)$时，说的是这个程序无论遇到神马情况（当然宕机免谈），最长运行时间就是$O(n^2)$，也就是n^2级别咯

写法:$f(n) \in O(n^2)$或者 $f(n) = O(g(n))$

SECOND ONE：BIG-Ω :表示下界，你怎么瘦都瘦不下来，卡住了~~

$\Omega(g(n)) = f(n)$ means: 当$n > n_0$ 时，存在$c > 0$

$$0 \leqslant cg(n) \leqslant f(n)$$

注意：当我们说，一个东东的运行时间是$\Omega(n^2)$时，说的是这个程序无论遇到神马情况（就算幸运女神爱上你了也不可能），运行时间觉得不会少于cg(n)

THIRD ONE：BIG-Θ ：表示确界，上不去下不来

$\Theta(g(n)) = f(n)$ means: 当$n > n_0$ 时，存在$c_1 > 0, c_2 > 0$

$$0 \leqslant c_1 g(n) \leqslant f(n) \leqslant c_2 g(n)$$

三者表示关系

递归式

三种方法

`Substitution method（代换法）`

1. Guess 猜测解的形式
2. Verify 用数学归纳法验证
3. Solve 得到常数解

`Recursion Tree method（递归树）`

求解方式：

1. 根据给出的T(n)画出递归树
2. 求出每层的代价相加求和，即可得到时间复杂度

`Master Method（主方法）`

形式：

$$T(n) = aT(n/b) + f(n),\ a \geqslant 1,\ b \geqslant 1$$

将$f(n)$和$n^{(\log_b a)}$比较

1. $f(n) = O(n^{\log_b a-\varepsilon})$则$T(n) = \Theta(n^{\log_b a})$
2. $f(n) = \Theta(n^{\log_b a})$则$T(n) = \Theta(n^{\log_b a}\lg n)$
3. $f(n) = \Omega(n^{\log_b a+\varepsilon})$

   要求：若对于`常数c<1`，和所有`足够大的n`，$af(n/b) \leqslant cf(n)$，则

   $T(n) = \Theta(f(n))$