主定理 (Master公式)

定义与应用实例

Master公式，又称为Master定理或主定理，是分析递归算法时间复杂度的一种重要工具，尤其适用于具有具有分治的递归算法递归算法。

> 其实部分没有分治的递归也可以用, 极少数非递归代码也可以用

$$\mathrm{T}(n) = a * \mathrm{T}(n/b) + \mathrm{O}(n^d)$$

Master公式本身就是递归的形式，是递归方法时间复杂度的一种表示法。

解释

$\mathrm{T}(n)$ 代表递归方法处理规模为 n 的数据量的时间复杂度。

$\mathrm{T}(n/b)$ 代表调用子递归方法的总体时间复杂度，

$\mathrm{O}(n^d)$ 代表调用子递归方法这行代码外其他代码（下面简称递归外代码）的时间复杂度。

a: 每次递归调用子问题的数量。即在一个递归方法，需要调用几次子递归方法 b: 子问题的规模缩小的比例。例如二分法递归搜索时，每次需要查找的数据都缩小了一半，那么 $b = 2$ d: 每次递归调用之外的代码时间复杂度的参数。例如二分法递归搜索时，每次递归时除了调用递归的方法，没有什么 `for` 循环代码，时间复杂度是 $\mathrm{O}(1)$，即 $n^d = 1$，因此 $d = 0$

使用主定理推到复杂度

主定理本身表示的复杂度难以分析，但是可以根据具体项的关系确定大致的复杂度

当 $d < log_b^a$ 时，递归时间复杂度为：$\mathrm{O}(nlog_b^a)$

当 $d = log_b^a$ 时，递归时间复杂度为：$\mathrm{O}(n^d * log_2^n)$

当 $d > log_b^a$ 时，递归时间复杂度为：$\mathrm{O}(n^d)$

对于那些不太正常的公式

例如

$$\mathrm{T}(n) = \mathrm{T}(n-1) + 0.5n$$

我们要这样分析，推式子

$$T(N) = T(1) + \sum k = 2^N 0.5k = 1 + 0.5 \times [\sum k = 1^N k - 1] = 1 + 0.5 \times [N(N+1)/2 - 1] = \frac{N(N+1)+2}{4}。$$

参考文献

> 带你彻底搞懂递归时间复杂度的Master公式