# 递归原理

1. 定义

在递归过程中, 函数将自身作为子例程调用.

简单说程序调用自身的编程技巧叫递归. 递归的思想是**把一个大型复杂问题层层转化为一个与原问题规模更小的问题, 问题被拆解成子问题后, 递归调用继续进行, 直到子问题无需进一步递归就可以解决的地步为止.**

使用递归需要避免出现死循环, 为了确保递归正确工作, 递归程序应该包含2个属性:

1. 基本情况(bottom cases), 基本情况用于保证程序调用及时返回, 不在继续递归, 保证了程序可终止.
2. 递推关系(recurrentce relation), 可将所有其它情况拆分到基本案例.
3. 特征

**优雅性:**

相比其它解法(比如迭代法), 使用递归法, 会发现只需要少量程序就可以描述出解题过程, 大大减少了程序的代码量, 而且很好理解. 递归的能力在于用有限的语句来定义对象的无限集合.

**反向性:**

由于递归调用程序需要维护调用栈, 而栈具有后进先出的特征, 因此递归程序适合满足取反类需求. 比如字符串取反, 链表取反等.

**递推关系:**

递归程序可以较明显的发现递推关系, 反过来也可以这么说, 具有递推关系的问题基本都可以通过递归求解(当然也许有性能更佳的解法,). 递推关系常见问题有杨辉三角、阶乘计算.

1. 考虑递归

具有以下特征的问题可考虑递归求解:

--递推关系: 一个问题的结果与其子问题的结果之间的关系.

--基本情况: 不需要进一步的递归调用就可以直接计算答案的情况. 往往是问题被减少到最小规模的情况, 也就是如果我们认为将问题划分为子问题是一种自上而下的方式的最下层.

一旦计算出以上两个元素, 再想要实现一个递归函数, 就只需要根据递推关系调用函数本身, 直到其抵达基本情况.