递归

递归的定义：

在定义一个过程或函数时出现调用本过程或本函数的成分称为递归（recursion）。

递归策略只需少量的代码就能描述出解题过程中所需的多次重复计算，大大减少了算法的代码量。

递归执行过程：分解和求值

分解：用递归体将“大问题”分解成“小问题”，直至递归出口，然后进行值过程

求值：已知“小问题”计算“大问题”

计算技术中包含但不限于的递归：

递归过程：直接或间接调用自身的过程

递归算法：包含递归过程的算法

直接递归：调用自身

间接递归：通过另一个过程或函数调用自身

尾递归：一个递归过程或递归函数中的递归调用语句是最后一条执行语句的递归调用。

递归调用要满足的条件：

1. 子问题，子问题的求解方法与原问题完全相同
2. 有限

递归适用情况：

1. 概念的定义是递归的，这时可以直接转化成对应的递归算法。如Fibonacci数列
2. 数据结构是递归的，如链表
3. 问题的求解方法是递归的，如Hanoi问题

递归的思路：

将不能或不好直接解决的大问题转化为一个或几个小问题

递归执行过程：

分解 🡪 递归体开始分解

求值 🡪 递归出口开始求值

递归出口：由递归终止条件和执行语句组成

递归体：由递归执行条件和递归语句组成

这也意味着通常递归出口和递归体是if和else之间的关系

递归模型是递归算法的抽象，它描述了递归问题的递归结构，用于在代码实现前理清思路

描述方法：

f(n) = C ， 递归出口的条件

f(n) = f(m) + C ，递归体的条件

代码实现算法的两种情况：

1. 返回值类型为 void ，则不需为递归出口编写代码，比如函数体内只用一条if语句：if(btree\_p != NULL);
2. 返回值类型为非void，则需分别为递归出口和递归体编写代码，即进行两次或以上的条件判断

以上原则能让代码更加简洁

递归中计数：

1. 全局变量
2. 传参

递归调用深度：即递归调用链长，如线性递归链（以结点计）、递归树（以树高计）

递归算法的时间复杂度分析：递推式