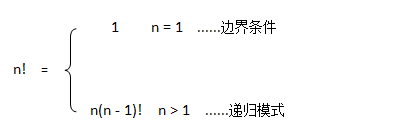
**递归算法**

**http://blog.csdn.net/SeeTheWorld518/article/details/47957183**

**1、递归的定义**   
递归就是子程序（或函数）直接调用自己或通过一系列调用语句间接调用自己，是一种描述问题和解决问题的基本方法。   
递归常与分治思想同时使用，能产生许多高校的算法。递归常用来解决结构相似的问题。所谓结构相似，是指构成原问题的子问题与原问题在结构上相似，可以用类似的方法解决。具体地，整个问题的解决，可以分为两部分：第一部分是一些特殊情况，有直接的解法；第二部分与原问题相似，但比原问题的规模小，并且依赖第一部分的结果。。实际上，递归是把一个不能或不好解决的大问题转化成一个或几个小问题，再把这些小问题进一步分解成更小的小问题，直至每个小问题都可以直接解决。因此，递归有两个基本要素：   
（1） 边界条件：确定递归到何时终止，也称为递归出口。   
（2） 递归模式：大问题是如何分解为小问题的，也称为递归体。   
递归函数只有具备了这两个要素，才能在有限次计算后得出结果。   
**2、递归算法实例**

2.1求一个整数n的阶乘   
阶乘的定义如下图：



根据阶乘的递归定义，很容易就能写出求阶乘的递归算法。

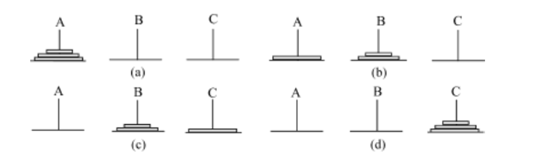
def factorial(n) :

if n == 1 :

return 1 #递归结束

return n \* factorial(n - 1) #问题规模减1，递归调用

2.2汉诺塔   
汉诺塔问题是递归函数的经典应用，它来自一个古老传说：在世界刚被创建的时候有一座钻石宝塔A，其上有64个金蝶。所有碟子按从大到小的次序从塔底堆放至塔顶。紧挨着这座塔有另外两个钻石宝塔B和C。从世界创始之日起，波罗门的牧师就一直在试图把塔A上的碟子移动到C上去，其间借助于塔B的帮助。每次只能移动一个碟子，任何时候都不能把一个碟子放在比它小的碟子上面。当牧师们完成这个任务时，世界末日也就到了。   
对于汉诺塔问题的求解，可以通过以下3步实现：   
（1）将塔A上的n -1个碟子借助C塔先移动到B塔上；   
（2）把塔A上剩下的一个碟子移动到塔C上；   
（3）将n - 1个碟子从B塔借助塔A移动到塔C上。   
很显然，这是一个递归求解的过程，假设碟子数n=3时，汉诺塔问题的求解过程如下图所示：



汉诺塔的递归算法（Python实现）：

def Hanoi(n, A, B, C) :

if (n == 1) :

move(A, c) #表示只有一个碟子时，直接从A塔移动到C塔

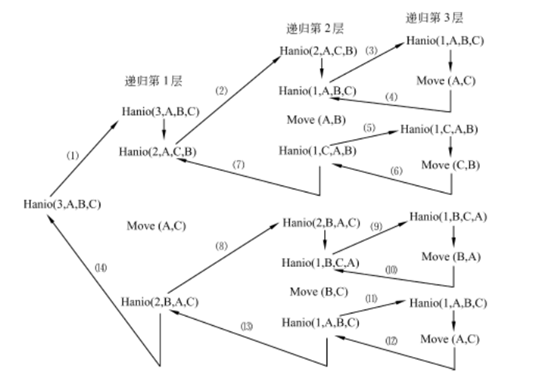
else :

Hanoi(n - 1, A, C, B) #将剩下的A塔上的n-1借助C塔移动到B塔

move(A, C) #将A上最后一个直接移动到C塔上

Hanoi(n - 1, B, A, C) #将B塔上的n-1个碟子借助A塔移动到C塔

**递归函数的运行轨迹**   
借助汉诺塔这个实例，来讲解一下递归函数的运行轨迹。在递归函数中，调用函数和被调用函数都是同一个函数，需要注意的是函数的调用层次，如果把调用递归函数的主函数称为第0层，进入函数后，首次递归调用自身称为第1层调用；从第i层递归调用自身称为第i+1层。反之退出i+1层调用应该返回第i层。下图是n=3时汉诺塔算法的运行轨迹，有向弧上的数字表示递归调用和返回的执行顺序。



汉诺塔的递归算法代码实现：

#coding=utf-8

i = 1

def move(n, mfrom, mto) :

global i

print "第%d步:将%d号盘子从%s -> %s" %(i, n, mfrom, mto)

i += 1

def hanoi(n, A, B, C) :

if n == 1 :

move(1, A, C)

else :

hanoi(n - 1, A, C, B)

move(n, A, C)

hanoi(n - 1, B, A, C)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*程序入口\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

try :

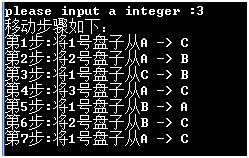
n = int(raw\_input("please input a integer :"))

print "移动步骤如下："

hanoi(n, 'A', 'B', 'C')

except ValueError:

print "please input a integer n(n > 0)!"

执行结果：   


**2.3 斐波拉契数列**   
斐波拉契数列，是这样的一个数列：0、1、1、2、3、5、8、13、21、……。   
斐波拉契数列的核心思想是:   
从第三项起，每一项都等于前两项的和，即F(N) = F(N - 1) + F(N - 2) (N >= 2)   
并且规定F(0) = 0，F(1) = 1

要求:利用递归算法获得指定项的斐波拉契数列。

#!/usr/bin/python

#coding=utf-8

def fib\_list(n) :

if n == 1 or n == 2 :

return 1

else :

m = fib\_list(n - 1) + fib\_list(n - 2)

return m

print "\*\*\*\*\*\*\*\*\*\*请输入要打印的斐波拉契数列项数n的值\*\*\*\*\*\*\*\*\*\*\*"

try :

n = int(raw\_input("enter:"))

except ValueError :

print "请输入一个整数！"

exit()

list2 = [0]

tmp = 1

while(tmp <= n):

list2.append(fib\_list(tmp))

tmp += 1

print list2

执行结果：   
