# 递归算法：

程序调用自身的编程技巧叫做递归。

程序调用自身的编程技巧称为递归（ recursion）。递归做为一种算法在程序设计语言中广泛应用。 一个过程或函数在其定义或说明中有直接或间接调用自身的一种方法，它通常把一个大型复杂的问题层层转化为一个与原问题相似的规模较小的问题来求解，递归策略只需少量的程序就可描述出解题过程所需要的多次重复计算，大大地减少了程序的代码量。递归的能力在于用有限的语句来定义对象的无限集合。

# 递归的条件

递归的三个条件：

边界条件

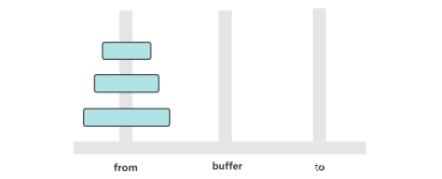
递归前进段

递归返回段

当边界条件不满足时，递归前进；当边界条件满足时，递归返回。

## 案例一：汉诺塔

有三个金刚石塔，第一个从小到大摞着64片黄金圆盘。现在把圆盘按大小顺序重新摆放在最后一个塔上。并且规定，在小圆盘上不能放大圆盘，在三个塔之间一次只能移动一个圆盘。



递归思想：

判断n 这个数字：

若是n=1,只有一个盘子，那么需要直接把这个盘子从from--》to

若是n>1,需要先把n-1个盘子从from，在to的帮助下，拿到buffer上，再将最底下的盘子拿到to上，最后，再将n-1个盘子从buffer在from的帮助下，拿到to上

算法实现：

public static void HanotaMove(int n, String from, String buffer, String to){

if(n==1){

System.out.println(from+"-->"+to);

return;

}else

{

HanotaMove(n-1,from,to,buffer);

System.out.println(from+"-->"+to);

HanotaMove(n-1,buffer,from,to);

}

}

## 案例二：欧几里得算法求两个数的最大公约数

欧几里德算法又称辗转相除法，是指用于计算两个正整数a，b的最大公约数。应用领域有数学和计算机两个方面。

假如需要求 1997 和 615 两个正整数的最大公约数,用欧几里德算法，是这样进行的：

1997 / 615 = 3 (余 152)

615 / 152 = 4(余7)

152 / 7 = 21(余5)

7 / 5 = 1 (余2)

5 / 2 = 2 (余1)

2 / 1 = 2 (余0)

至此，最大公约数为1

以除数和余数反复做除法运算，当余数为 0 时，取当前算式除数为最大公约数，所以就得出了 1997 和 615 的最大公约数 1。

算法实现：

private static void gcd(int a,int b) {

if (a >= b) {

if(a%b==0) {

System.out.println("最大公约数为"+b);

return;

}

gcd(b,a%b);

}else

gcd(b,a);

}

## 案例三：二分法查找

在计算机科学中，二分查找算法（英语：binary search algorithm），也称折半搜索算法（英语：half-interval search algorithm）[1]、对数搜索算法（英语：logarithmic search algorithm）[2]，是一种在有序数组中查找某一特定元素的搜索算法。搜索过程从数组的中间元素开始，如果中间元素正好是要查找的元素，则搜索过程结束；如果某一特定元素大于或者小于中间元素，则在数组大于或小于中间元素的那一半中查找，而且跟开始一样从中间元素开始比较。如果在某一步骤数组为空，则代表找不到。这种搜索算法每一次比较都使搜索范围缩小一半。

二分查找算法在情况下的复杂度是对数时间，进行{\displaystyle O(\log n)}O(\log n)次比较操作（{\displaystyle n}n在此处是数组的元素数量，{\displaystyle O}O是大O记号，{\displaystyle \log }\log 是对数）。二分查找算法使用常数空间，无论对任何大小的输入数据，算法使用的空间都是一样的。除非输入数据数量很少，否则二分查找算法比线性搜索更快，但数组必须事先被排序。尽管特定的、为了快速搜索而设计的数据结构更有效（比如哈希表），二分查找算法应用面更广。

算法实现：

private static int BSA(int[] arr,int key,int left,int right) {

if(left>right) return -1;

int mid=left+(right-left)/2;

if(arr[mid]==key)

return mid;

else if(arr[mid]<key)

return BSA(arr,key,mid+1,right);

else

return BSA(arr,key,left,mid-1);

}

## 案例四：递归求阶乘

利用递归思想实现阶乘算法 ，计算n的阶乘

算法实现：

private static long f(long n) {

if(n==1)

return 1;

else

return n\*f(n-1);

}