



数据结构与算法 (Python版)

递归的应用：任意进制转换

陈斌 北京大学 gischen@pku.edu.cn

整数转换为任意进制

❖ 这个在数据结构栈里讨论过的算法，又回来了！

递归和栈，一定有关联

❖ 如果上次你被“入栈”“出栈”搞得挺晕乎的话，这次递归算法一定会让你感到清新

而且这次我们要解决从二进制到十六进制的任意进制转换

整数转换为任意进制

❖ 我们用最熟悉的十进制分析下这个问题

十进制有十个不同符号：`convString = "0123456789"`

比十小的整数，转换成十进制，直接查表就可以了：`convString[n]`

想办法把比十大的整数，拆成一系列比十小的整数，逐个查表

比如七百六十九，拆成七、六、九，查表得到769就可以了

整数转换为任意进制

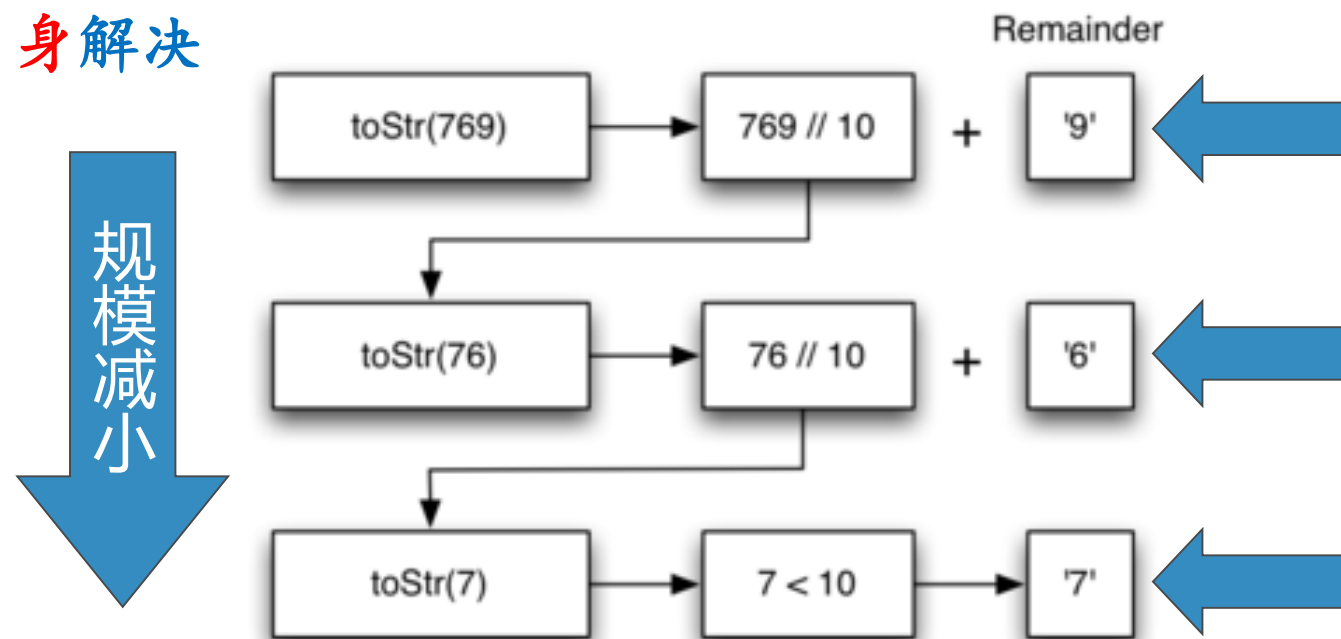
- ❖ 所以，在递归三定律里，我们找到了“基本结束条件”，就是小于十的整数
拆解整数的过程就是向“基本结束条件”演进的过程
- ❖ 我们用**整数除**，和**求余数**两个计算来将整数一步步拆开
除以“进制基base” (`// base`)
对“进制基”求余数 (`% base`)

整数转换为任意进制

❖ 问题就分解为：

余数总小于“进制基base”，是“基本结束条件”，可直接进行查表转换

整数商成为“更小规模”问题，通过递归调用自身解决



整数转换为任意进制：代码

❖ 下面就是递归算法的代码

```
def toStr(n, base):  
    convertString = "0123456789ABCDEF"  
    if n < base:  
        return convertString[n]  
    else:  
        return toStr(n//base, base) + convertString[n%base]  
  
print(toStr(1453, 16))
```

最小规模

减小规模，调用自身

