字典序算法

转自：http://www.sohu.com/a/228117420_453160

算法题目：

给定一个正整数，实现一个方法来求出离该整数最近的大于自身的“换位数”。

什么是换位数呢？就是把一个整数各个数位的数字进行全排列，从而得到新的整数。例如53241和23541。

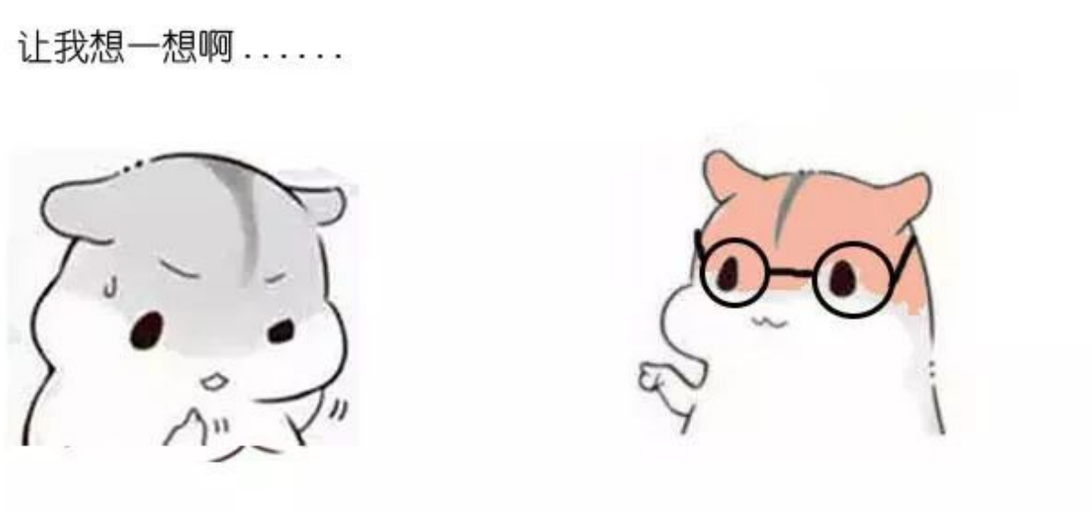
小灰也不知道这种经过换位的整数应该如何称呼，所以姑且称其为“换位数”。

题目要求写一个方法来寻找最近的且大于自身的换位数。比如下面这样：

输入**12345**，返回**12354**

输入**12354**，返回**12435**

输入**12435**，返回**12453**



小灰发现的“规律”：

输入**12345**，返回**12354**

12354 - 12345 = **9**

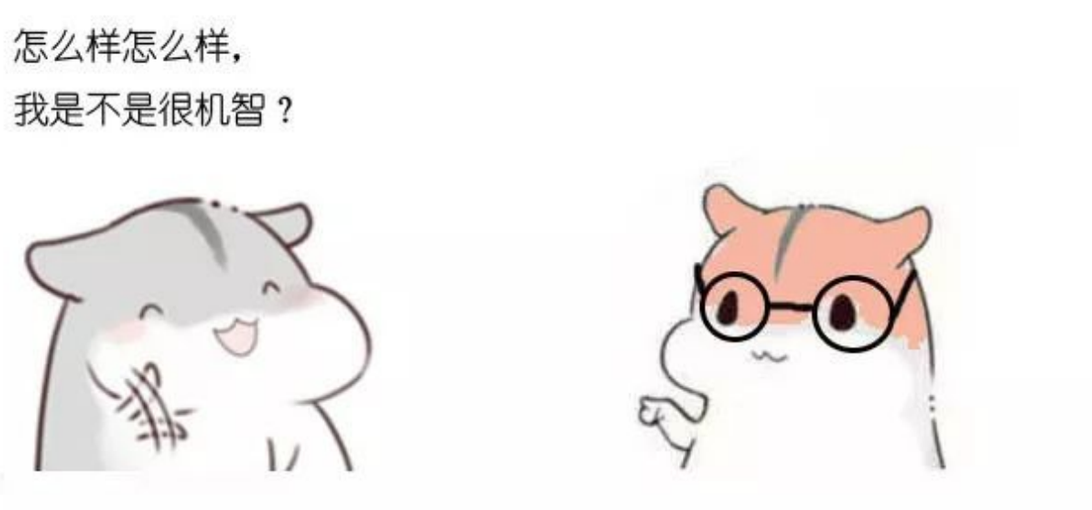
刚好相差9的一次方

输入**12354**，返回**12435**

12435 - 12354 = **81**

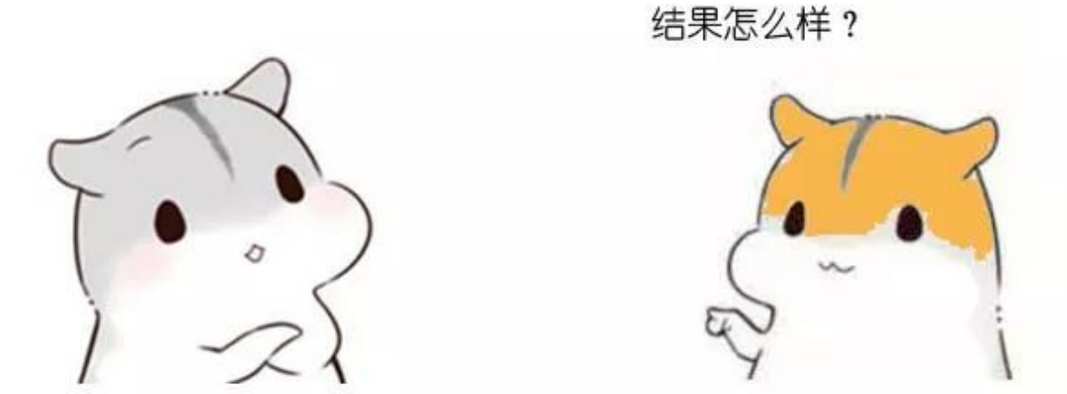
刚好相差9的二次方

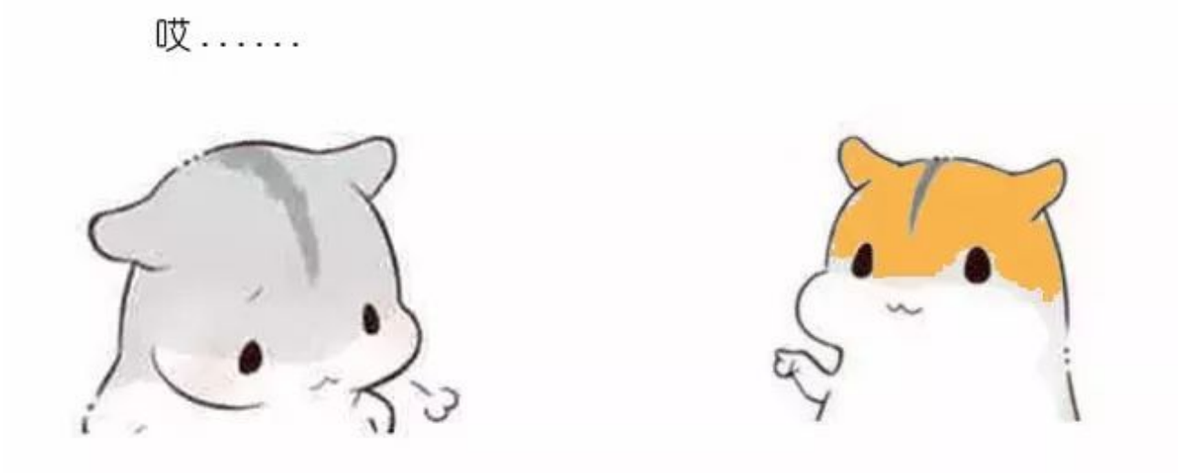
所以，每次计算最近的换位数，只需要加上9的N次方即可?

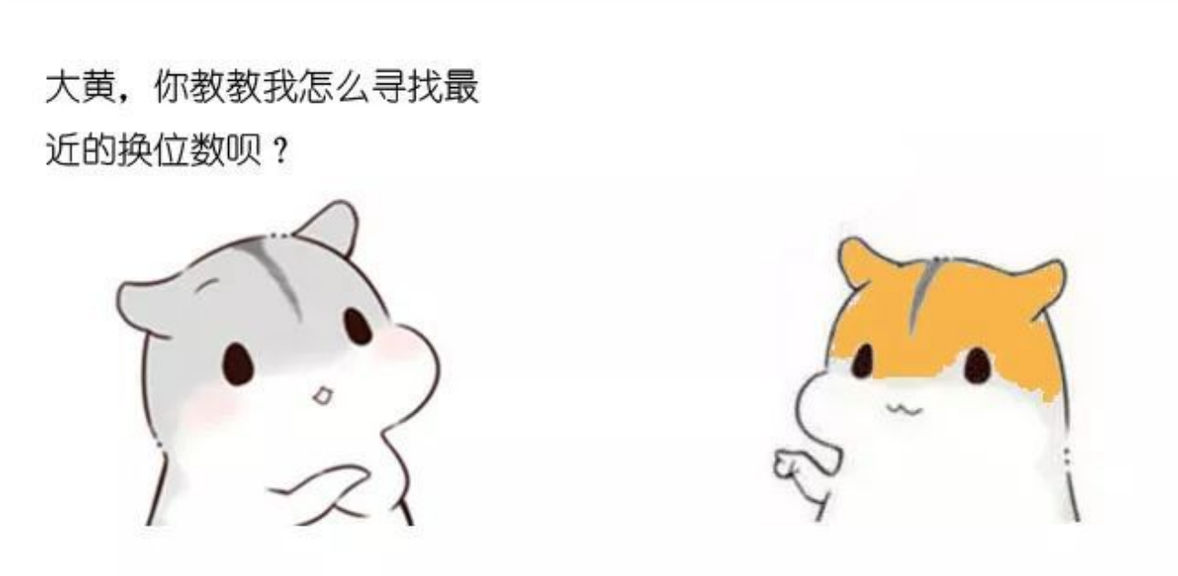



这算哪门子规律？12453-12435=18，24135-23541=594，也并不都是9幂啊！


呵呵！今天就先到这里，回家等通知去吧！


小灰，听说你去面试了？结果怎么样？


哎.....


大黄，你教教我怎么寻找最近的换位数呗？


好啊，在给出具体解法之前，小灰你先思考一个问题：由固定几个数字组成的整数，怎样排列最大，怎样排列最小？

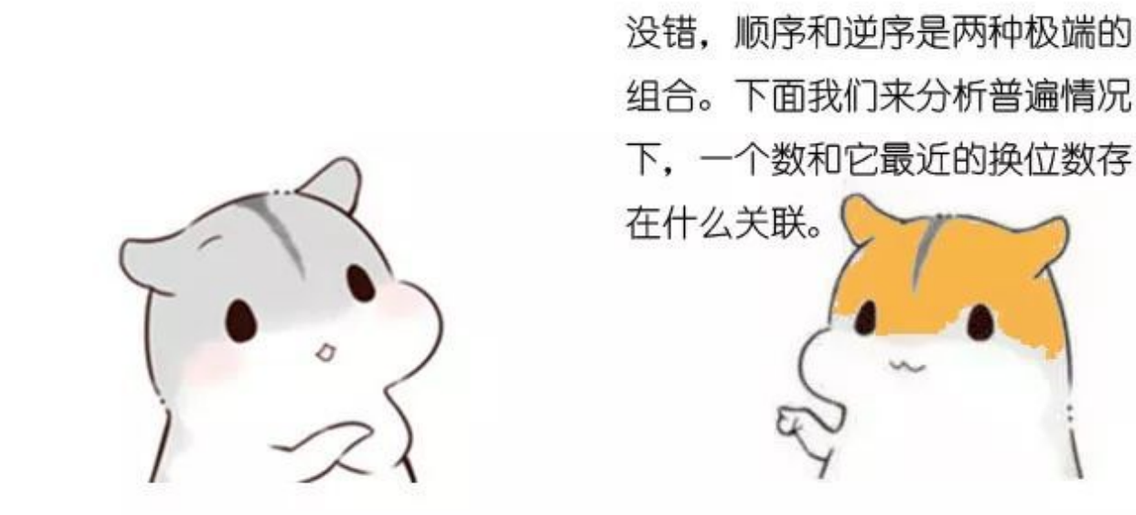

让我想想啊.....

举一个栗子：

给定1,2,3,4,5这几个数字。

最大的组合：**54321**

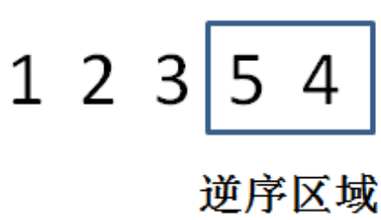
最小的组合：**12345**



比如给定整数12354，如何找到离它最近且大于它的换位数呢？

为了和原数接近，我们需要**尽量保持高位不变，低位在最小的范围内变换顺序**。

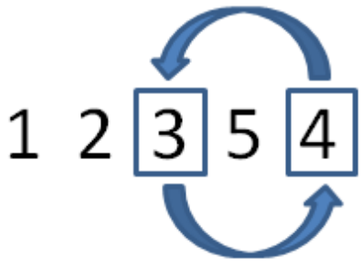
那么，究竟需要变换多少位呢？这取决于当前整数的**逆序区域**。



逆序区域

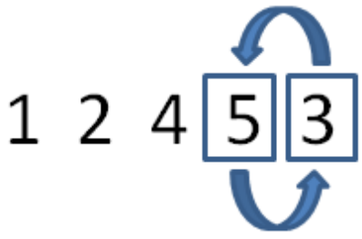
如果所示，12354的逆序区域是最后两位，仅看这两位已经是当前的最大组合。若想最接近原数，又比原数更大，必须从**倒数第3位**开始改变。

怎样改变呢？12345的倒数第3位是3，我们需要从后面的逆序区域中寻找到刚刚大于3的数字，和3的位置进行互换：



1 2 4 5 3

互换后的临时结果是12453，倒数第3位已经确定，这时候最后两位仍然是逆序状态。我们需要把最后两位**转变回顺序**，以此保证在倒数第3位数值为4的情况下，后后两位尽可能小：



1 2 4 3 5

这样一来，我们就得到了想要的结果**12435**。

获得最近换位数的三个步骤：

1.从后向前查看逆序区域，找到逆序区域的前一位，也就是数字置换的边界

2.把逆序区域的前一位和逆序区域中刚刚大于它的数字交换位置

3.把原来的逆序区域转为顺序

1.

```
/**
 *字典序
 * 1 2 3
 * 1 3 2
 * 2 1 3
 * 2 3 1
 * 3 1 2
 * 3 2 1
 */
function f(num){
  var numArr =  num.toString().split('');
  // 字符串转数字
  numArr = numArr.map(function(item){
    return parseInt(item);
  })
  // console.log('初始numArr:',numArr);
  var l = numArr.length;
  for(i=l-1;i>0;i--){

      if((numArr[i]-0)>(numArr[i-1]-0)){
```

```
        for(j=l-1;j>=i;j--){

        // if((numArr[i-1]-0)>(numArr[j]-0)){
          let reArr = numArr.slice(i,l);  //逆序区
          console.log('逆序区: ',reArr);
          //找到逆序区中大于临界值的所有数的最小值
          reArr = reArr.filter(function(item){
            return item>numArr[i-1]            });
           console.log('逆序区中大于临界值: ',reArr);
           // let x = numArr[i-1];  //浅拷贝
           let x = JSON.parse(JSON.stringify(numArr[i-1]))
             let index = numArr.lastIndexOf(Math.min.apply(Math, reArr));
             console.log('index',index);
             numArr[i-1] = numArr[index];
             numArr[index] = x;
           console.log('变换后numArr:',numArr);
           let before = numArr.slice(0,i).join('');
           let after = numArr.slice(i,l).sort().join('');
           return before+after;
         // }
        }
      }

    }

  }
var num = 123;
for(i=0;i<6;i++){
  console.log(f(num));
  num = f(num);
}
/**
132
213
231
312
321
*/
```

我自己理解的思路：首先自右向左选相邻两数比较，若左边数小于右边数取左边数数记为 x ，然后自末位向左至 x 位选取大于 x 的数中最小值记为 m ，将 x 与 m 位置互换;将 x 位右边数字升序排序．即得到当前数字字符串按字典序的下一位数字

这种解法拥有一个高大上的名字：**字典序算法**。

小灰，你说说这个解法的时间复杂度是多少？

三个步骤的每一步都是O(n)，
所以整体时间复杂度也是O(n)！

完全正确。对于这道算法题的解答就介绍到这里，如果大家有什么更优化的想法欢迎提出。感谢大家！