第5篇

二进制中1的个数

问题

输入一个整数，输出该数二进制表示中1的个数。其中负数用补码表示。

思路

详见offer书。 较好的解法是，以1100为例，1100 减去 1 得到 1011，发现其实减去1就是将最右的1变为0，将右边的0变为1， 所以 1100 和 1011 做按位与运算，得到 1000 就相当于抹掉了最右边的1。 按照这种思路，只要一直减去1，就逐个从右边抹掉了1， 当整个数 = 0 时说明抹完了，循环次数就是1的个数。

代码

```
class Solution:
    def NumberOf1(self, n):
        count = 0
        # 很神奇，python要加上对n<0的处理，但是用c语言就不用?
        if n < 0:
            n = n & 0xffffffff
        while n:
            n = (n-1) & n
            count += 1
        return count
```

数值的整数次方

问题

给定一个double类型的浮点数base和int类型的整数exponent。求base的exponent次方。

代码

```
# 一行代码.....
class Solution:
    def Power(self, base, exponent):
        return base ** exponent
```

当然本题原意是不要调用库，自己来实现，关键是考虑到很多细节。 要点如下：

- 考虑到base为0，且要注意 base是double型，判断double型=0时要小心浮点数不精确
- 考虑到exp为0，则返回-1
- exp大于0，可通过循环累成来实现
- exp小于0，需将乘方结果取倒数
- 优化：例如32次方，可以通过先求16次方再将16次方结果平方得到，可用递归

打印1到最大的n位数

问题

给定n，要按顺序打印从1到最大的n位数，例如，n=3，则打印1,2,3.....999.

思路

（1）先求出最大的n位数，例如n=3，最大的是 10^3 - 1 = 999，然后循环从1开始打印 问题：如果n稍微大一点点，例如20，一般的整数类型就溢出了!

(2) 用字符串的形式，模拟手工加法进位的操作

（3）转换思路，其实就是输出n个0-9的全排列，按递归的方法，从高位逐个递归深入下去，就能实现按照从小到大的顺序逐个输出了!

> 好吧，这个题主要就是要提醒注意n的范围，也就是说，要仔细思考给定的输入值是不是0？正数？负数？会不会过大？采取的数据类型有没有问题?

O(1)的时间删除链表某节点

问题

单向链表，给定头指针和指向某个节点的指针，要求O(1)时间删除该节点

思路

常规的做法是，从头开始沿着链表向后遍历，直到找到目标节点，然后修改目标的前一个节点的指针，让next指向目标的后一个节点，释放空间，完成删除。

这种复杂度是O(n)，主要就是因为必须遍历才能找到目标节点的**前一个节点**!

另辟蹊径，可以O(1)找到目标的后一个节点，将后一个节点的内容复制到目标上，这样问题就转换成删除目标的后一个节点了！无需从头遍历!

需要特别注意！处理一些特殊情况，例如单节点链表、目标是尾节点等。

代码

简述一下结构吧

- 判断链表头和目标是否指针为空，不空才能继续
- 目标不是尾节点，可以操作，将目标的下一节点的内容和next指针复制过来，删掉下一节点
- 如果链表只有一个节点，则直接将头指针next置空，释放目标节点
- 如果链表不是只有一个节点，而且目标是尾节点，则必须从头遍历找到目标的上一节点，然后按正

常方式释放。

补充：（想这么多还能不能愉快的编程了！简直有点神经质啊！！）以上基于假设是目标一定在链表中，如果目标节点不在链表中，就gg了.....

调整数组顺序，使奇数位于偶数前面

问题

输入一个整数数组，实现一个函数来调整该数组中数字的顺序，使得所有的奇数位于数组的前半部分，所有的偶数位于位于数组的后半部分，并**保证奇数和奇数，偶数和偶数之间的相对位置不变。**

思路

需要注意到保证奇数和奇数、偶数和偶数的相对位置不变。

如果是允许做数组的delete和append操作，或者说允许建新数组的话： 1）另开两个数组，扫一遍分别存偶数和奇数，然后将偶数连接到奇数后面 2）扫一遍统计奇数的个数，新建一个等长的数组，奇数从头往后加，偶数从奇数长度下标处往后加（当允许另开空间时，此方法较好） 3）扫一遍，遇到偶数就从列表中删除，然后追加到列表末端。需注意结束扫描的条件

如果只允许在原数组上做交换操作： 4）利用冒泡排序的思想，从前往后扫，遇到奇数就往前冒泡（或者从后往前扫，遇到偶数往后冒泡），不开辟额外空间

代码

上面（1）用python的简洁写法：

```
def reOrderArray(self, array):
        # write code here
        odd,even=[],[]
        for i in array:
            odd.append(i) if i%2==1 else even.append(i)
        return odd+even
```

再写一下上面（4）的代码：

```
class Solution:
    def reOrderArray(self, array):
        # 冒泡思想。从前往后扫，遇到奇数就往前冒泡。
        last_odd = -1
        for i in range(0, len(array)):
            if array[i] % 2 == 1:
                for j in range(i, last_odd+1, -1):
                    array[j], array[j-1] = array[j-1], array[j]
                    print(array)
                last_odd += 1
        return array
```

扩展

如果不要求奇数和偶数的相对位置不变，只要能奇数在前，偶数在后。 1）从前往后扫，遇到偶数，就记下，把它之后的内容往前挪一位，把偶数放在最后的空位 2）维护两个指针，一个从头部往后，一个从尾部往前，如果第一个指向偶数，第二个指向奇数，就交换，当第二个位于第一个前面时说明交换完成。