剑指offer （python版） 49-52

49把字符串转换成整数

题目：将一个字符串转换成一个整数(实现Integer.valueOf(string)的功能，但是string不符合数字要求时返回0)，要求不能使用字符串转换整数的库函数。 数值为0或者字符串不是一个合法的数值则返回0。

**输入描述:**

输入一个字符串,包括数字字母符号,可以为空

**输出描述:**

如果是合法的数值表达则返回该数字，否则返回0

思路：本题其实没有什么思路难度，但是需要考虑周全，需考虑字符串首的符号位。1、空串检验，2、数字合法性检验，3、正负数数字区分，4、只有一个正负号时

代码：

1. # -\*- coding:utf-8 -\*-
2. **class** Solution:
3. **def** StrToInt(self, s):
4. # write code here
5. res,multi,flag=0,1,1
6. **if** **not** s:
7. **return** res
8. **if** s[0]=='-' **or** s[0]== '+':
9. **if** s[0]=='-':
10. flag=-1
11. s=s[1:]
12. **for** i **in** range(len(s)-1,-1,-1):
13. **if** s[i]>='0' **and** s[i]<='9':
14. res += (ord(s[i])-48)\*multi
15. multi=multi\*10
16. **else**:
17. **return** 0

50数组中重复的数字

题目：在一个长度为n的数组里的所有数字都在0到n-1的范围内。 数组中某些数字是重复的，但不知道有几个数字是重复的。也不知道每个数字重复几次。请找出数组中任意一个重复的数字。 例如，如果输入长度为7的数组{2,3,1,0,2,5,3}，那么对应的输出是第一个重复的数字2。

思路：还可以利用哈希表来解决这个问题。从头到尾按顺序扫描数组的每个数字，每扫描到一个数字的时候，判断哈希表里是否已经包含了该数字。如果哈希表里还没有这个数字，就把它加入哈希表。如果哈希表里已经存在该数字，就找到一个重复的数字。可以用python的字典（键值对）。

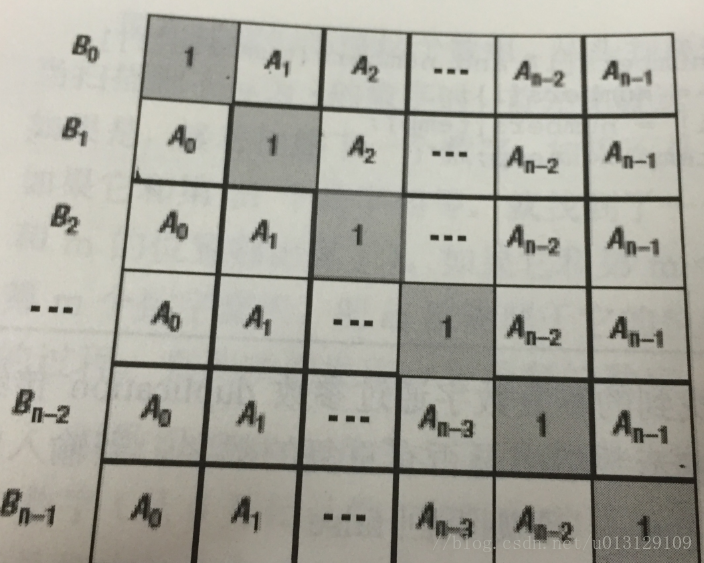
代码：

1. # -\*- coding:utf-8 -\*-
2. **class** Solution:
3. # 这里要特别注意~找到任意重复的一个值并赋值到duplication[0]
4. # 函数返回True/False
5. **def** duplicate(self, numbers, duplication):
6. # write code here
7. dict={}
8. **for** i **in** numbers:
9. **if** i **not** **in** dict:
10. dict[i]=1
11. **else**:
12. duplication[0]=i
13. **return** True
14. **return** False

51构建乘积数组

题目：给定一个数组A[0,1,...,n-1],请构建一个数组B[0,1,...,n-1],其中B中的元素B[i]=A[0]\*A[1]\*...\*A[i-1]\*A[i+1]\*...\*A[n-1]。不能使用除法。

思路：**B[i]的值可以看作下图的矩阵中每行的乘积。B[i]的意义是A数组不包括i位置的所有乘积，分为 i左边的元素乘积和 i右边的所有元素乘积。第一个for计算i左边的乘积，第二个for计算右边的。**



代码：

1. # -\*- coding:utf-8 -\*-
2. **class** Solution:
3. **def** multiply(self, A):
4. # write code here
5. **if** **not** A:
6. **return** []
7. num=len(A)
8. B=[None]\*num
9. B[0]=1
10. **for** i **in** range(1,num):
11. B[i]=B[i-1]\*A[i-1]
12. temp=1
13. **for** i **in** range(num-2,-1,-1):
14. temp \*=A[i+1]
15. B[i]=B[i]\*temp
16. **return** B

52正则表达式匹配

题目：请实现一个函数用来匹配包括'.'和'\*'的正则表达式。模式中的字符'.'表示任意一个字符，而'\*'表示它前面的字符可以出现任意次（包含0次）。 在本题中，匹配是指字符串的所有字符匹配整个模式。例如，字符串"aaa"与模式"a.a"和"ab\*ac\*a"匹配，但是与"aa.a"和"ab\*a"均不匹配

思路：

本题就是很直观的字符串的匹配，难点在于当遇到模式中两个特殊字符怎么处理和对于各种形式字符串的全面考虑。这里采用递归的方式，每次只匹配s中一个字符（模式串pattern， 被匹配字符串s）：

处理判断过程如下：

如果 s和pattern都为空，匹配成功

如果pattern是空串，而s不是，匹配失败

如果s，pattern均不是空串(长度至少为1)，考虑到pattern中‘ \* ’前字符可以出现0次，所以不能简单比较s和pattern的第一个字符是否相等，这里分为两种情况考虑：

如果pattern的第二个字符是‘ \* ’：

如果s与pattern的第一个字符匹配（即s与pattern的第一个字符相等或者pattern第一个字符为‘ . ’），剩余部分有两种匹配方式：(1). s后移一位，相当于认为‘ \* ’前的字符在s中出现不止一次，(2).pattern后移两位，相当于认为‘ \* ’前的字符在s中只出现一次。

否则模式串pattern后移两位，相当于认为‘ \* ’前的字符在s中出现了0次；

如果pattern的第二个字符不是‘ \* ’：如果s与pattern的第一个字符匹配（含义同上），s和pattern同时后移一位，继续匹配；否则匹配失败

代码：

1. **class** Solution:
2. # s, pattern都是字符串
3. **def** match(self, s, pattern):
4. # write code here
5. **if** len(s)==0 **and** len(pattern)==0:
6. **return** True
7. **if** len(s)>0 **and** len(pattern)==0:
8. **return** False
9. **if** len(pattern)>1 **and** pattern[1]=='\*':
10. **if** len(s)>0 **and** (s[0]==pattern[0] **or** pattern[0]=='.'):
11. **return** self.match(s,pattern[2:]) **or** self.match(s[1:],pattern)
12. **else**:
13. **return** self.match(s,pattern[2:])
14. **if** len(s)>0 **and** (s[0]==pattern[0] **or** pattern[0]=='.'):
15. **return** self.match(s[1:],pattern[1:])
16. **else**:
17. **return** False