第2篇

字符串——替换空格 链表——逆序输出链表

替换空格

要求 请实现一个函数，将一个字符串中的空格替换成“%20”。例如，当字符串为We Are Happy.则经过替换之后的字符串为We%20Are%20Happy。

思路 如果用Python或者java等高级点的语言，直接调replace是最简单的写法了，或者开一个新字符串，从前往后读原字符串，读到其他字符就直接追加，读到空格就追加%20。

如果用C语言，字符串以char[]来存，就要考虑更多的东西，比如数组的容量，挪动各个char，最后的\0等。

一种思路是从前往后读，碰到空格，就先把后面的全部内容往后挪两格，加入%20。这样会导致很多内容被多次挪动，O(n^2)效率。 另一种思路是，预先扫一遍看有多少个空格，从而知道最终结果的长度，然后从后往前进行替换，预留出恰好的空间。这样每个字符只需要挪动一次，O(n)效率即可。

需要特别注意\0、空指针、容量不足以及数组边界的问题。

```
class Solution {
public:
  void replaceSpace(char *str,int length) {
        //length是字符数组的总容量大小，而不是字符串的长度。
        //遍历一遍字符串找出空格的数量
        if(str==NULL||length<0)
            return ;
        int i=0;
        int oldnumber=0;//记录以前的长度
        int replacenumber=0;//记录空格的数量
        while(str[i]!='\0')
            {
               oldnumber++;
               if(str[i]==' ')
                   {
                     replacenumber++;
                   }
                  i++;
            }
        int newlength=oldnumber+replacenumber*2;//插入后的长度
        if(newlength>length)//如果计算后的长度大于总长度就无法插入
            return ;
        int pOldlength=oldnumber; //注意不要减一因为隐藏个‘\0’也要算里
        int pNewlength=newlength;
        while(pOldlength>=0&&pNewlength>pOldlength)//放字符
            {
```

```
            if(str[pOldlength]==' ') //碰到空格就替换
                {
                   str[pNewlength--]='0';
                   str[pNewlength--]='2';
                   str[pNewlength--]='%';
                }
             else //不是空格就把pOldlength指向的字符装入pNewlength指向的位置
             {
                  str[pNewlength--]=str[pOldlength];
             }
           pOldlength--; //不管是if还是elsr都要把pOldlength前移
        }
  }
};
```

这里感觉值得一提的是书中特别写出了测试用例：

- 输入的字符串中包含空格：空格在开头、最末、中间、连续多个空格
- 输入的字符串中没有空格
- 输入空指针、空串、只有一个空格、全为空格等特殊情况

在写代码时感觉 应该先想好这些各种情况，保证对各种特殊情况都考虑到了，才能使得写出的代码尽可能正确。

倒序输出链表

题目描述 输入一个链表，从尾到头打印链表每个节点的值。

思路 1）倒置指针——会改变原始输入信息，需注意是否允许改变原数据 2）基于递归实现——先输出内层的，注意当链表很长时会导致调用栈溢出 3）基于栈，用循环实现

code

```
class Solution:
    # 返回从尾部到头部的列表值序列，例如[1,2,3]
    def printListFromTailToHead(self, listNode):
        # write code here
        if not listNode:
            return []
        stack = []
        current = listNode
        while(current is not None):
            stack.append(current.val)
            current = current.next
        result = []

        # 注意range的使用
        for i in range(len(stack)-1, -1, -1):
```

```
            result.append(stack[i])
        return result
```