本次笔记记录了LeetCode中关于字符串的一些问题，并给出了相应的思路说明和代码。题目编号与LeetCode对应，方便查找。

题目1：LeetCode 13. Roman to Integer

Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M.

Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

For example, The number twenty seven is written as XXVII, which is XX + V + II.

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX. There are six instances where subtraction is used:

I can be placed before V (5) and X (10) to make 4 and 9.
X can be placed before L (50) and C (100) to make 40 and 90.
C can be placed before D (500) and M (1000) to make 400 and 900.

Example :
Input: "MCMXCIV"
Output: 1994
Explanation: M = 1000, CM = 900, XC = 90 and IV = 4.

```
"""
思路：
1.构建将罗马数字映射为阿拉伯数字的字典
2.依次读取字符串中的字符，并判断当前字符的值是否小于后一字符的值；
3.若符合，说明有“抽象”，因为循环到下一位时会求和，所以当前应该求差（即把两个字符抽象成一个）；
4.若不符，则把当前的数和之前的和相加就行；
5.当然，如果当前的i是字符串的末尾，i+1肯定不存在，直接把当前值加入到z中求和。
"""
class Solution(object):
    def romanToInt(self, s):
        """
        :type s: str
        :rtype: int
        """
        roman = {'M': 1000,'D': 500 ,'C': 100,'L': 50,'X': 10,'V': 5,'I': 1}
        z = 0
        k = len(s)
        for i in range(0, k):
            if i == k-1:
                z += roman[s[i]]
            else:
                if roman[s[i]] < roman[s[i+1]]:
                    z -= roman[s[i]]
                else:
```

```
                z += roman[s[i]]
        return z
```

题目2：LeetCode 14. Longest Common Prefix

Write a function to find the longest common prefix string amongst an array of strings.If there is no common prefix, return an empty string "".

Example 1:
Input: ["flower","flow","flight"]
Output: "fl"
Example 2:
input: ["dog","racecar","car"]
Output: ""

```
"""
思路：
1.目标是找共同的最长前缀。
2.如果找出最小的串和最大的串，那么它们的共同前缀就是表里所有串的共同前缀（反证可得）
3.然后对比最小串和最大串前缀，如果第i位不一样，返回前i-1位
4.如果遍历完了最小串，还是相同，则最小串本身就是共同前缀
"""
class Solution(object):
    def longestCommonPrefix(self, strs):
        """
        :type strs: List[str]
        :rtype: str
        """
        if not strs:
            return ""
        min_s = min(strs)
        max_s = max(strs)
        for i in range(0,len(min_s)):
            if max_s[i] != min_s[i]:
                return min_s[:i] #切片，是不包括i的，到i-1为止
        return min_s
```

题目3：LeetCode 20. Valid Parentheses

Given a string containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

An input string is valid if:

- Open brackets must be closed by the same type of brackets.
- Open brackets must be closed in the correct order.
- Note that an empty string is also considered valid.

Example :

Input: "()[]{}" ________Output: true
Input: "([)]" _________Output: false
Input: "{[]()}"_________Output: true

```
"""
思路：用栈入栈出来做括号的匹配
1.先把括号用字典进行配对，方便比较操作
2.读取s中的字符，如果不是任何一个左右括号，返回False
3.如果是左括号，存入栈中；
4.如果是右括号，判断栈中是否有左括号
    1rd若没有,说明S中的右括号出现在左括号之前，返回False;
    2rd若有，是否能和最后一个进入栈中的左括号匹配，不成功则返回False,成功则读取s的下一个字符
```

```
5.读取完所有的字符后，若stack中没有任何字符，说明匹配了；反之则未匹配
"""
class Solution(object):
    def isValid(self, s):
        """
        :type s: str
        :rtype: bool
        """
        stack = []
        dict = {"]":"[", "}":"{", ")":"("}
        for char in s:
            if char in dict.values():
                stack.append(char)
            elif char in dict.keys():
                if  stack == []:
                    return False
                elif dict[char] != stack.pop():
                    return False
            else: #s中有不在字典中的字符
                return False
        if stack == []:
            return True
        else:
            return False
```

题目4：LeetCode 28. Implement strStr()

Return the index of the first occurrence of needle in haystack, or -1 if needle is not part of haystack.

Example :

Input: haystack = "hello", needle = "ll"

Output: 2

Input: haystack = "aaaaa", needle = "bba"

Output: -1

```
"""
若haystack长度为n,needle长度为m
方法一：利用循环 比较
    这种方法的时间复杂度：O(m*n)
方法二：利用python切片
    这种方法的时间复杂度：O((m-n )* n)
方法三：利用KMP算法
    这种方法的时间复杂度：O（m+n）
"""
class Solution(object):
    def strStr(self, haystack, needle):
        """
        :type haystack: str
        :type needle: str
        :rtype: int
        """

        # method 1: use loop
        """
        if needle == None:
            return -1
        n = len(haystack)
        m = len(needle)
        i, j = 0, 0
        while i < n and j < m:
            if haystack[i] == needle[j]:
                i = i+1
                j = j+1
            else:
                i = i-j+1
                j = 0
        if  j == m:
            return i-m
        return -1
        """

        #method2:  slice
        """
        for i in range(len(haystack) - len(needle)+1):
            if haystack[i:i+len(needle)] == needle:
                return i
        return -1
        """

        #method 3 KMP
        def kmp(str_):
            b, prefix = 0, [0]
            for i in range(1, len(str_)):
                while b > 0 and str_[i] != str_[b]:
                    b = prefix[b - 1]
                if str_[b] == str_[i]:
                    b += 1
                else:
                    b = 0
                prefix.append(b)
            return prefix

        str_ = kmp(needle + '#' + haystack)
        n = len(needle)
        if n == 0:
            return n
        for i in range(n + 1, len(str_)):
            if str_[i] == n:
                return i - 2 * n
        return -1
```

题目5：LeetCode 38. Count and Say

The count-and-say sequence is the sequence of integers with the first five terms as following:

1. 1
2. 11
3. 21
4. 1211
5. 111221
   1 is read off as "one 1" or 11.
   11 is read off as "two 1s" or 21.
   21 is read off as "one 2, then one 1" or 1211.

Given an integer n where 1 ≤ n ≤ 30, generate the nth term of the count-and-say sequence.
Note: Each term of the sequence of integers will be represented as a string.

```
"""
思路:
0.特殊情况，n = 1 ,n=2 手动设置
1.循环整个串，初始化标记值marker为串的头部,临时变量temp,计数器count=1
2.获取当前循环位置的值，记为current
3.如果当前值current和上一个位置的值（marker的值）相等，说明marker重复一次，令count = count + 1
4.如果不相等，说明marker只出现了一次，temp = temp + 有几个 marker 令count = 1，且marker后移
5.内循环退出后，将temp的值赋给s
6.继续外循环，退出后，返回s
"""
class Solution(object):
    def countAndSay(self, n):
        """
        :type n: int
        :rtype: str
        """
        if n == 1:
            s = '1'
            return s
        s = '11'
        for i in range(1,n-1):
            marker, temp, count = s[0], '', 1 #初始化
            for current in s[1:]: #读取当前位置的值(第二个位置开始)
                if current == marker: #如果当前值==前一个值，计数器count增加1
                    count += 1
                else: #如果不相等，说明未重复，计数器置为1，且让标记marker等于current
                    temp += str(count)+marker #原temp+（有count个marker），用数字表示
                    count = 1
                    marker = current
                    # 相当于marker后移（下个内循环current后移，保证marker在current前一个位置）
            temp += str(count)+current #temp临时变量每次存下s最新的值
            s = temp #赋值
        return s
```

题目6：LeetCode 58. Length of Last Word

Given a string s consists of upper/lower-case alphabets and empty space characters ' ', return the length of last word in the string.If the last word does not exist, return 0.

Note: A word is defined as a character sequence consists of non-space characters only.

Example:
Input: "Hello World"
Output: 5

```
"""
方法1：利用python内置函数
    strip()去除字符串头和尾的空格，split()去除中间的空格并生成一个list
    返回list中最后一个元素，并求长度

"""
class Solution(object):
    def lengthOfLastWord(self, s):
        """
        :type s: str
        :rtype: int
        """
        #方法1

        s = s.strip(' ').split(' ')
        return len(s[-1])
```

题目7：LeetCode 67. Add Binary

Given two binary strings, return their sum (also a binary string).The input strings are both non-empty and contains only characters 1 or 0.

Example 1:
Input: a = "11", b = "1"
Output: "100"

```
"""
方法1使用内置函数
"""
class Solution(object):
    def addBinary(self, a, b):
        """
        :type a: str
        :type b: str
        :rtype: str
        """

        #方法1 内置函数
        return bin(int(a, 2) + int(b, 2))[2:]
```

题目8：LeetCode 125. Valid Palindrome

Given a string, determine if it is a palindrome, considering only alphanumeric characters and ignoring cases.
Note: For the purpose of this problem, we define empty string as valid palindrome.
Example :

Input: "A man, a plan, a canal: Panama"
Output: true
Input: "race a car"
Output: false

```
"""
思路：
1.定义两个变量，一个从小变大，另一个从大变小
2.当小变量小于大变量时，判断以大小变量为下标位置的字符是否相等（不区分大小写）
3.特殊情况
    以小变量为下标位置遇到非字母或数字，小变量增加1；
    以大变量为下标位置遇到非字母或数字，大变量减小1；
4.正常情况则比较两个位置的值，一样的话，让小变量加1，大变量减1；不一样的话说明不是回文
5.最后比较完毕，退出while循环时，说明是回文
"""
class Solution(object):
    def isPalindrome(self, s):
        """
        :type s: str
        :rtype: bool
        """
        l, r = 0, len(s)-1
        while l < r:
            while l < r and not s[l].isalnum():#isalnum（）判断是否是字母或数字
                l += 1
            while l <r and not s[r].isalnum():
                r -= 1
            if s[l].lower() != s[r].lower():
                return False
            l +=1; r -= 1
        return True
```

题目9：LeetCode 344. Reverse String

Write a function that reverses a string. The input string is given as an array of characters char[].

Do not allocate extra space for another array, you must do this by modifying the input array in-place with O(1) extra memory.
You may assume all the characters consist of printable ascii characters.

Example :
Input: ["h","e","l","l","o"]
Output: ["o","l","l","e","h"]

```
"""
方法1:
交换前后半边对应位置的值，交换式用temp存，满足额外o(1)空间

"""
class Solution(object):
    def reverseString(self, s):
        """
        :type s: List[str]
        :rtype: None Do not return anything, modify s in-place instead.
        """
        i = 0
        j = len(s)-1
        while i < j:
            temp = s[i]
            s[i] = s[j]
            s[j] = temp
            i += 1
            j -= 1
```

题目10 LeetCode 345. Reverse Vowels of a String

Write a function that takes a string as input and reverse only the vowels of a string.

Example :
Input: "leetcode"
Output: "leotcede"

```
"""
方法1思路:
1.和反转列表相似, 分别设置两个变量, 一大一小, 小的增大, 大的减小
2.遇到非元音字符, 跳过 (令变量增大或减小)
3.当一边遇到元音字符, 停一下, 待另一边也遇到元音字符时, 交换两边的字符
注意: 由于字符串是不能进行元素修改的, 因此交换的步骤需要在列表中操作
所以, 一开始需要将str转为list,最后再将list转换回str。
"""
class Solution(object):
    def reverseVowels(self, s):
        """
        :type s: str
        :rtype: str
        """
        #方法1
        s = list(s)
        L = ['a', 'e', 'i', 'o', 'u','A','E','I','O','U']
        i = 0
        j = len(s)-1
        while i < j:
            while i < j and s[i] not in L:
                i = i + 1
            while i < j and s[j] not in L:
                j = j - 1
            while i < j and (s[i] in L) and (s[j] in L):
                temp = s[i]
                s[i] = s[j]
                s[j] = temp
                i += 1
                j -= 1
        return "".join(s)
```

题目11 LeetCode 383. Ransom Note

Given an arbitrary ransom note string and another string containing letters from all the magazines, write a function that will return true if the ransom note can be constructed from the magazines ; otherwise, it will return false.

Each letter in the magazine string can only be used once in your ransom note.
Note: You may assume that both strings contain only lowercase letters.

canConstruct("a", "b") -> false
canConstruct("aa", "ab") -> false
canConstruct("aa", "aba") -> true

```
"""
要求: 题目要求从magazine中抽取字符构建ransomNote,但不能重复抽取同一个位置的字符两次及以上
所以: 每次抽取完一个字符后, 就把这个字符从magazine中删除pop,这样避免重复
0.由于要进行pop操作, 需将str转换成list, 然后一些初始化操作
1. 构建循环, 条件为ransomNote 和  magazine都没有超出自己的索引范围
    i < mlen - n  有这个 -n 是因为每次删除一个字符后, magazine的长度会减1
    j < rlen 由于不会删除 ransomNote中的字符,无需-n
2.当ransomNote[j]==magazine[i]
    删除m中的当前元素, 并让计数器n增加1;
    j指向ransomNote的下一个位置
    i 置 0, 从头开始
3.如果j == rlen, 说明r中的所有字符都能在m中找到, 返回True, 反之返回False
"""
class Solution(object):
    def canConstruct(self, ransomNote, magazine):
        """
        :type ransomNote: str
        :type magazine: str
        :rtype: bool
        """
        ransomNote = list(ransomNote)
        magazine = list(magazine)
        rlen = len(ransomNote)
        mlen = len(magazine)
        i, j = 0, 0
        n = 0
        while i < mlen-n and j< rlen:
            if ransomNote[j] == magazine[i]:
                magazine.pop(i)
                n = n + 1
                j = j+1
                i = 0
            else:
                i = i+1
        if j == rlen:
            return True
        else:
            return False
        """如果要求，按顺序抽取m中的字符的话，得用这个
        rlen = len(ransomNote)
        mlen = len(magazine)
        i, j = 0, 0
        while i < mlen and j< rlen:
            if ransomNote[j] == magazine[i]:
                i = i+1
                j = j+1
            else:
                i = i+1
        if j == rlen:
            return True
        else:
            return False
        """
```

题目12 LeetCode 387. First Unique Character in a String

Given a string, find the first non-repeating character in it and return it's index. If it doesn't exist, return -1.

Examples:
s = "leetcode"
return 0.
s = "loveleetcode",
return 2.

```
"""
1.找出未重复的元素和其下标  遍历字符串
    用字典，字典存下每个字符的出现的位置，下标i；
    如果在后面的字符中又出现这个字符，则令其值为-1
2.第二步就是找出字典中不等于-1的且最小的数，这个数就是满足目标的数
3.如果字符串有不重复的数，则返回其下标；反之返回-1
"""
class Solution(object):
    def firstUniqChar(self, s):
        """
        :type s: str
        :rtype: int
        """
        mapping = {}
        ans = len(s)
        temp = ans
        for i, c in enumerate(s):
            if c in mapping:
                mapping[c] = -1
            else:
                mapping[c] = i
        for c in mapping:
            if mapping[c] != -1:
                if mapping[c]<temp:
                    temp = mapping[c]
        if temp!=ans:
            return temp
        else:
            return -1
```

题目13 LeetCode 415. Add Strings

Given two non-negative integers num1 and num2 represented as string, return the sum of num1 and num2.

- The length of both num1 and num2 is < 5100.
- Both num1 and num2 contains only digits 0-9.
- Both num1 and num2 does not contain any leading zero.
- You must not use any built-in BigInteger library or convert the inputs to integer directly.

```
"""
目标：俩数相加
1. 从个位开始相加，依次向十位百位这样循环
    carry%10 可以知道是否有进位 carry //=10 可以返回当前的和的个位
2. 注意，返回的整数得是字符串形式，所以使用字符串拼接方式粘连每一次求的和的个位
3. s = str + s 而不是s+str,是因为相加求和是向左的，不是向右的。
"""
class Solution(object):
    def addStrings(self, num1, num2):
        """
        :type num1: str
        :type num2: str
        :rtype: str
        """
        a = len(num1)
        b = len(num2)
        s =''
        i, j = a-1, b-1
        carry = 0
        while i >= 0 or j >= 0 or carry:
            #or carry可以单独处理最高位之前还有进位的情况
            if i >= 0:
                carry = carry + int(num1[i])
            if j >= 0:
                carry = carry + int(num2[j])
            s = str(carry%10) + s
            carry //= 10
            i = i - 1
            j = j - 1
        return s
```

题目14 LeetCode 434. Number of Segments in a String

Count the number of segments in a string, where a segment is defined to be a contiguous sequence of non-space characters. Please note that the string does not contain any non-printable characters.

Example:
Input: "Hello, my name is John"
Output: 5

```
"""
方法1：利用内置函数
方法2: 利用flag和count计数
    遇到可打印字符开头，说明有单词，计数加1
    如遇到非可打印字符，说明单词结束，flag标记更新
"""
class Solution(object):
    def countSegments(self, s):
        """
        :type s: str
        :rtype: int
        """
        """
        #方法1
        return len(s.split())
        """
        #方法2
        flag = 0 # give fist time meet the not space char
        count = 0 # caculate the num
        s_length = len(s)
        for i in range(s_length):
            if s[i] != ' ':
                if not flag:
                    count = count + 1
                flag = 1
            else:
                flag = 0
        return count
```

题目15 LeetCode 459. Repeated Substring Pattern

Given a non-empty string check if it can be constructed by taking a substring of it and appending multiple copies of the substring together. You may assume the given string consists of lowercase English letters only and its length will not exceed 10000.

Example :
Input: "abab"
Output: True
Explanation: It's the substring "ab" twice.
Input: "aba"
Output: False

```
"""
1.将s的复制拼接到s，新串就是s+s
2.去掉新串ss的首位字符，然后查找s是否在ss中
3.如果在，说明s是重复字串构成的
4.如果不在，说明s不是由重复字串构成的
"""
class Solution(object):
    def repeatedSubstringPattern(self, s):
        """
        :type s: str
        :rtype: bool
        """
        if not s:
            return False
        ss = (s + s)[1:-1]
        if ss.find(s) != -1:
            return True
        else:
            return False
```

题目16 LeetCode 520. Detect Capital

Given a word, you need to judge whether the usage of capitals in it is right or not.
We define the usage of capitals in a word to be right when one of the following cases holds:

- All letters in this word are capitals, like "USA".
- All letters in this word are not capitals, like "leetcode".
- Only the first letter in this word is capital if it has more than one letter, like "Google".
- Otherwise, we define that this word doesn't use capitals in a right way.

Example :
Input: "USA"
Output: True
Input: "FlaG"
Output: False

```
"""
方法1:
1.让word和word.upeper每个字符相比较，如果相等则s字符串当前位置置1，反之置0;
2.创建三个用于和s比较的字符串a,b,c 分别对应于若word是capital的三种情况下s的输出
3.最后，若s属于a,b,c其中之一，说明word是capital;反之不是。

方法2:
```

```
利用python内置函数...

"""
class Solution(object):
    def detectCapitalUse(self, word):
        """
        :type word: str
        :rtype: bool
        """
        #方法1
        n = len(word)
        s = ''
        a = ''
        b = ''
        c = ''
        w = word.upper()
        for i in range(n):
            if word[i] == w[i] :
                s = s + '1'
            else:
                s = s + '0'
        for j in range(n):
            a = a + '1'
        for k in range(n):
            b = b + '0'
        for l in range(1,n):
            c = c +'0'
        c = '1'+c
        if s == a or s ==b or s== c:
            return True
        else:
            return False

        #方法2
        """
        return word.isupper() or word.islower() or word.istitle()
        """
```

题目17 LeetCode 521. Longest Uncommon Subsequence I

Given a group of two strings, you need to find the longest uncommon subsequence of this group of two strings. The longest uncommon subsequence is defined as the longest subsequence of one of these strings and this subsequence should not be any subsequence of the other strings.

A subsequence is a sequence that can be derived from one sequence by deleting some characters without changing the order of the remaining elements. Trivially, any string is a subsequence of itself and an empty string is a subsequence of any string.

The input will be two strings, and the output needs to be the length of the longest uncommon subsequence. If the longest uncommon subsequence doesn't exist, return -1.

Example :
Input: "aba", "cdc"
Output: 3

```
"""
emmm,这题有点简单...就不说了吧
"""
class Solution(object):
    def findLUSlength(self, a, b):
        """
        :type a: str
        :type b: str
        :rtype: int
        """
        if a == b:
            return -1
        m =len(a)
        n =len(b)
        if m > n:
            return m
        else:
            return n
```

题目18 LeetCode 3. Longest Substring Without Repeating Characters

Given a string, find the length of the longest substring without repeating characters.

Example :

Input: "abcabcbb"
Output: 3
Explanation: The answer is "abc", with the length of 3.
Input: "bbbbb"
Output: 1
Explanation: The answer is "b", with the length of 1.
Input: "pwwkew"
Output: 3
Explanation: The answer is "wke", with the length of 3.

```
"""
方法1:
1.定义一个字典，用value去存储串s中每个字符的位置，用key去存储字符
2.当s中的字符未在字典中出现过，字串最大长度更新;
3.当s中的字符在字典中出现过，更新start值（统计长度的起始点）
4.最后返回经过比较得出的最大的maxLength
方法2:
暂未全部完成，待补充
"""
class Solution(object):
    def lengthOfLongestSubstring(self, s):
        """
        :type s: str
        :rtype: int
        """
        start = maxLength = 0
        usedChar = {}

        for i in range(len(s)):
            if s[i] in usedChar and start <= usedChar[s[i]]:
                start = usedChar[s[i]] + 1
            else:
                maxLength = max(maxLength, i - start + 1)

            usedChar[s[i]] = i

        return maxLength
        """
        #未完成版，待补充
        n = len(s)
        if len(s) == 1:
            return 1
        mapping = {}
        i = 0
        length_max =''
        j = i
        while i <len(s):
            c = s[i]
            if c in mapping and j <= i:
                mapping = {}
                length_max = length_max + str(i - j)
                i = j + 1
                j = i
            else:
                mapping[c] = 1
                i = i + 1
        length_max = length_max + str(i-j)
        if length_max == '':
            return n
        else:
            return int(max((length_max)))
        """
```

本次的刷题记录和分析就到这里~

代码github链接: https://github.com/LSayhi/Algorithms (https://github.com/LSayhi/Algorithms)

CSDN博客：https://blog.csdn.net/LSayhi (https://blog.csdn.net/LSayhi)

更多原创干货和最新资讯，请关注我吧

In [ ]: