字符串高频面试题精讲

七月算法　**曹鹏**

2015年4月21日

提纲

- 字符串简介
- 面试题总体分析
- 一些例题
  - 例1 0-1串交换排序
  - 例2 字符的替换和复制
  - 例3 交换星号
  - 例4 子串变位词
  - 例5 单词（字符串）翻转
- 总结

字符串简介

- 字符串(String)
  - 通常把它作为字符数组
  - java : String内置类型，不可更改，要更改的话可考虑转StringBuffer, StringBuilder, char []之类
  - C++ : std::string可更改，也可以考虑用char[] (char*)
  - C：只有char[]
  - 注意
    - C++中“＋”运算符，复杂度未定义，但通常认为是线性的
    - C++ std::string substr和java的String的subString参数不同
    - 字符范围:
      - C/C++ [-128..+127], 我们通常转化为unsigned 变为[0..+255]
      - Java: [0..65535]

面试题总体分析

- 和数组相关，内容广泛
  - 概念理解：字典序
  - 简单操作：插入、删除字符，旋转
  - 规则判断（罗马数字转换 是否是合法的整数、浮点数）
  - 数字运算（大数加法、二进制加法）
  - 排序、交换（partition过程）
  - 字符计数（hash)：变位词
  - 匹配（正则表达式、全串匹配、KMP、周期判断）
  - 动态规划 (LCS、编辑距离、最长回文子串）
  - 搜索（单词变换、排列组合）

例1 0-1交换

- 把一个0-1串（只包含0和1的串）进行排序，你可以交换任意两个位置，问最少交换的次数？（国内某公司最新在线笔试题）
  - 分析：快排partition?最左边的那些0和最右边的那些1都可以不管
    - 00…0001…….0111….1

```
int answer = 0;
for (int i = 0, j = len – 1; i < j; ++i, --j) {
  for (;(i < j) && (a[i] == ‘0’);++i);
  for (;(j > i) && (a[j] == ‘1’); --j);
  if (i < j) ++answer;
}
```

例2 字符替换和复制

- 删除一个字符串所有的a,并且复制所有的b。注：字符数组足够大
  - 分析：
    - 先删除a,可以利用原来字符串的空间

```
int n = 0, numb = 0;
for (int i = 0; s[i]; ++i) {
        if (s[i] != 'a') { s[n++] = s[i];}
        if (s[i] == 'b') { ++numb;}
}
s[n] = 0;
```

    - 再复制b，注意字符串要加长
      - 先计算字符串里有几个b，得到复制后的长度
      - 然后“倒着”复制——惯用技巧

例2——续

```
int newLength = n + numb;
s[newLength] = 0;
for (int i = newLength - 1, j = n – 1; j >=0; --j) {
        s[i--] = s[j];
        if (s[j] == ‘b’) s[i--] = ‘b’;
}
```

- 思考题：如何把字符串的空格变成”%20”?同样，字符数组足够大！

例3 交换星号

- 例3 一个字符串只包含*和数字，请把它的*号都放开头。
  - 方法1 快排partition——数字相对顺序会变化
    - 循环不变式：[0..i – 1]都是*, [i..j – 1]是数字，[j..n – 1]未探测

```
for (int i = 0, j = 0; j < n; ++j)
        if (s[j] == ‘*’) swap(s[i++], s[j]);
```

例3 续1

- 样例 *01*2*4
  - i = 0, j = 0, *01*2*4 交换s[0]，不变，i = 1
  - i = 1, j = 1, *01*2*4 不变
  - i = 1, j = 2, *01*2*4 不变
  - i = 1, j = 3, 交换s[1],s[3]变为 **102*4 并且i = 2
  - i = 2, j = 4， **102*4不变
  - i = 2, j = 5， 交换s[2],s[5]变为***0214 且i = 3
  - 再往后没变化了

例3 续2

- 方法2 数字相对顺序不变
  - “倒着”

```
int j = n – 1;
for (int i = n – 1; i >= 0; --i)
        if (isdigit(s[i])) s[j--] = s[i];
for (; j >= 0; --j) s[j] = ‘*’;
```

例4 子串变位词

- 给定两个串a和b，问b是否是a的子串的变位词。例如输入a = hello, b = lel, lle, ello都是true,但是b = elo是false。（国外某公司最新面试题）
  - 滑动窗口的思想
    - 动态维护一个“窗口”。
    - 比如b的长度是3，我们考察a[0..2], [1..3],[2..4]是否和b是变位词
    - 如何与b比较？

例4——续1

- 我们用一个hash,基于字符串的特殊性，我们可以用[0..255]或者[0..65535]的数组，我们暂且认为它们都是小写英文字母，用[0..25]来表示b中每个单词出现多少次。
- 我们可以存一下有多少个非0次出现的，以后有用

```
    int nonZero = 0;
for (int i = 0; i < lenb; ++i)
    if (++num[b[i] – ‘a’] == 1) ++nonZero;
```

例4——续2

- 我们用b中的次数减去a中一个“窗口”内的字符种类，如果结果全是0，则找到这样的子串了。注意num[]的含义变为了字符种类差
- 第一个窗口 [0..lenb – 1] (注意lena < lenb无解)

```
for (int i = 0;  i < lenb; ++i) {
        int c = a[i] – ‘a’;
        --num[c];
        if (num[c] == 0) --nonZero;
        else if (num[c] == -1) ++nonZero;
}
if (nonZero == 0) return true;
```

例4——续3

- 窗口如何滑动？向右移动一位
  - 新窗口a[i – lenb + 1..i]
  - 旧窗口a[i – lenb.. i – 1]
    - 扔掉a[i – lenb]
    - 加入a[i]

```
for (int i = lenb; i < lena; ++i) {
        int c = a[i – lenb] – ‘a’;
        ++num[c];
        if (num[c] == 1) ++nonZero;
        else if (num[c] == 0) --nonZero;
        c = a[i] – ‘a’;
        --num[c];
        if (num[c] == 0) --nonZero;
        else if (num[c] == -1) ++nonZero;
        if (nonZero == 0) return true;
}
```

- 思考题 Leetcode 3

例5 单词翻转

- 翻转句子中全部的单词，单词内容不变
  - 例如I’m a student. 变为student. a I’m
  - in-place翻转 字符串第i位到第j位
    - while (i < j) swap(s[i++], s[j--]);
  - 有什么用？
    - 翻转整个句子：.tneduts a m’I
    - 每个单词单独翻转：student. a I’m
  - 难点？如何区分单词？找空格，split
  - 思考题：字符串循环移位abcd
    - 移动1次变为bcda
    - 移动2次变为cdab
    - 移动3次变为dabc
    - 结论：长度为n，移动m次，相当于移动m % n次
      - 前m % n位翻转，后n – m % n位翻转
      - 总体再翻转一次 试验一下？

总结

- 我理解的in-place (原地）
  - 本身O(1)空间
  - 递归，堆栈空间可以不考虑
- 原地相关的问题
  - 字符串循环左移、右移动
  - 快排partition相关
- 滑动窗口
  - 能达到O(n)的的时间复杂度
  - O(1)的空间复杂度
- 规则相关——细致
- 匹配（暴力）:KMP比较少见
- Manacher——要求比较高的笔试