倍儿亲切的字符串算法

今日目标：

1：完成今天面试题

面试实战

关于字符串是否可变的问题：https://lemire.me/blog/2017/07/07/are-your-strings-immutable/

709.转换成小写字母

网易游戏，亚马逊面试题，709.转换成小写字母

解法1：使用内置API，不推荐

解法2：ASCCII码操作

在线ASCCII对照表

发现：

a-z：97-122

A-Z：65-90

0-9：48-57

大写字母和小写字母之间差了32，故，遍历字符串中的每个字符，如果是大小字母，加上32即可

```
class Solution {
    //ASCCII码表操作
    public String toLowerCase(String str) {
        //特殊
        if (str==null || str.length() <2) {
            return str;
        }
        char[] strs = str.toCharArray();
        for (int i=0;i<strs.length;i++) {
            if (strs[i] >='A' && strs[i] <= 'Z') {
                strs[i] += 32;
            }
        }
        return new String(strs);
    }
}
```

解法3：位运算

基于ASCCII码的区别，使用位运算的技巧。

- 大写变小写、小写变大写：字符 ^= 32;

> 异或操作的另一种理解方式：不进位加法

> 32是2的幂次方，二进制中只有1位为1
>
> A 65：0100 0001
>
> 32：0010 0000
>
> -- 异或----------
>
> 0110 0001
>
> (对大写字母加了32)
>
> a 97：0110 0001
>
> 32：0010 0000
>
> -- 异或----------
>
> 0100 0001
>
> (对小写字母减了32)

- 大写变小写、小写变小写：字符 |= 32;

> A 65：0100 0001
>
> 32：0010 0000
>
> ------或----------
>
> 0110 0001
>
> (对大写字母加了32)
>
> a 97：0110 0001
>
> 32：0010 0000
>
> -- 或----------
>
> 0110 0001
>
> (对小写字母不变)

- 大写变大写、小写变大写：字符 &= 33;

> A 65：0100 0001
>
> 33：0010 0001
>
> ------与----------
>
> 0100 0001
>
> (对大写字母不变)
>
> a 97：0110 0001
>
> 33：0010 0001
>
> -- 或----------
>
> 0010 0001
>
> (对小写字母减了32)

```java
class Solution {
    //ASCCII码+位运算
    public String toLowerCase(String str) {
        //特殊
        if (str==null || str.length() <2) {
            return str;
        }
        char[] strs = str.toCharArray();
        for (int i=0;i<strs.length;i++) {
            strs[i] |= 32;
        }
        return new String(strs);
    }
}
```

58.最后一个单词的长度

字节，唯品会面试题，58.最后一个单词的长度

比较简单！

> 参考题解：https://leetcode-cn.com/problems/length-of-last-word/solution/hua-jie-suan-fa-58-zui-hou-yi-ge-dan-ci-de-chang-d/

```
class Solution {
    public int lengthOfLastWord(String s) {
        int end = s.length() -1;
        while (end >=0 && s.charAt(end) == ' ') {
            end--;
        }
        if (end<0) {
            return 0;
        }
        int start = end;
        while (start >=0 && s.charAt(start) != ' ') {
            start--;
        }
        return end -start;
    }
}
```

771. 宝石与石头

百度，字节，亚马逊面试题，771. 宝石与石头

解法1：暴力，遍历S，对于 S 中的每个字符，遍历一次字符串 J，如果其和 J 中的某一个字符相同，则是宝石。---不推荐

解法2：哈希集合

此处只需要用一个基础的数组即可，因为题设条件说了：`J` 中的字母不重复，`J` 和 `S` 中的所有字符都是字母。字母区分大小写。

```
class Solution {
    public int numJewelsInStones(String jewels, String stones) {
        int[] hash = new int[64];//能包含大小写字母的数组不超过64个
        char[] js = jewels.toCharArray();
        for (char cr:js) {
            hash[cr-'A']++;
        }
        int count = 0;
        char[] st = stones.toCharArray();
        for (char c:st) {
            if (hash[c-'A'] > 0) {
                count++;
            }
        }
        return count;
    }
```

```
}
```

387. 字符串中的第一个唯一字符

字节,华为,爱奇艺最近面试题,387. 字符串中的第一个唯一字符

题意说明: 第一个不重复的字符,并非指字符串中前后两个字符不重复, 这里的意思是: 对于某个字母,它在整个字符串中是否有重复的,

解法1: 哈希表辅助,两次遍历,一次遍历求出现的频率,二次遍历找第一个出现频率为1的字符

```
class Solution {
    public int firstUniqChar(String s) {
        //构造hash,字符串中字母ascii码值作为下标,对应位置存储出现的次数
        int[] hash = new int[26];//题设只有小写字母
        char[] cs = s.toCharArray();
        for (char c:cs) {
            hash[c-'a']++;
        }
        //二次遍历找第一个频率为1的字母
        for (int i=0;i<cs.length;i++) {
            if (hash [ cs[i] -'a' ] ==1) {
                return i;
            }
        }
        return -1;
    }
}
```

> 参看题解:https://leetcode-cn.com/problems/first-unique-character-in-a-string/solution/wei-zheng-ze-ha-xi-ji-he-dui-lie-yuan-sh-bupy/
>
> 注意多解法

14. 最长公共前缀

字节,快手,腾讯,百度最近面试题,14. 最长公共前缀

> 参考官方题解的多解法

1：分治法

```
```

```java
class Solution {
    public String longestCommonPrefix(String[] strs) {
        //特殊
        if (strs == null || strs.length ==0 ) {
            return "";
        }
        return longestCommonPrefix(strs,0,strs.length-1);
    }
    //求[begin,end]区间内的lcp
    public String longestCommonPrefix(String[] strs,int begin,int end) {
        if (begin == end) {
            return strs[begin];
        }
        //divide
        int mid = (end - begin) / 2 + begin;
        String lcpleft = longestCommonPrefix(strs,begin,mid);//左边区间的lcp
        String lcpright = longestCommonPrefix(strs,mid+1,end);//右边区间的lcp
        //conquer:合并
        return commonPrefix(lcpleft,lcpright);
    }

    public String commonPrefix(String lcpleft,String lcpright) {
        //lcp的长度不会超过两个字符串的最短长度
        int minLen = Math.min(lcpleft.length(),lcpright.length());
        if (minLen ==0) {
            return "";
        }
        //两个字符串按位比较即可
        StringBuilder sb = new StringBuilder();
        for (int i=0;i< minLen;i++) {
            char l = lcpleft.charAt(i);
            char r = lcpright.charAt(i);
            if (l!=r) {
                break;
            }
            sb.append(l);
        }
        return sb.toString();
    }

}
```

151. 翻转字符串里的单词

字节，腾讯，华为最近面试题，151. 翻转字符串里的单词

```java
class Solution {

    public String reverseWords(String s) {
        StringBuilder stringBuilder = new StringBuilder();
        getReverseWords(s, stringBuilder, 0);
        return stringBuilder.toString().trim();
    }
```

```
8
9       public void getReverseWords(String s, StringBuilder stringBuilder, int
    left) {
10          // 确定单词的左边界
11          while (left < s.length() && s.charAt(left) == ' ') {
12              left++;
13          }
14          if (left == s.length()) {
15              return;
16          }
17          int right = left;
18          // 确定单词的右边界
19          while (right < s.length() && s.charAt(right) != ' ') {
20              right++;
21          }
22          // 查找下一个单词
23          getReverseWords(s, stringBuilder, right);
24          // 添加单词
25          stringBuilder.append(s, left, right).append(" ");
26      }
27  }
```

其他题目：

344. 反转字符串

541. 反转字符串 II

557. 反转字符串中的单词 III

917. 仅仅反转字母

125. 验证回文串

680. 验证回文字符串 II