欢迎大家参与本项目，贡献其他语言版本的代码，拥抱开源，让更多学习算法的小伙伴们收益！

> 简单的反转还不够，我要花式反转

541. 反转字符串II

https://leetcode-cn.com/problems/reverse-string-ii/

给定一个字符串 s 和一个整数 k，你需要对从字符串开头算起的每隔 2k 个字符的前 k 个字符进行反转。

如果剩余字符少于 k 个，则将剩余字符全部反转。

如果剩余字符小于 2k 但大于或等于 k 个，则反转前 k 个字符，其余字符保持原样。

示例:

输入: s = "abcdefg", k = 2
输出: "bacdfeg"

思路

这道题目其实也是模拟，实现题目中规定的反转规则就可以了。

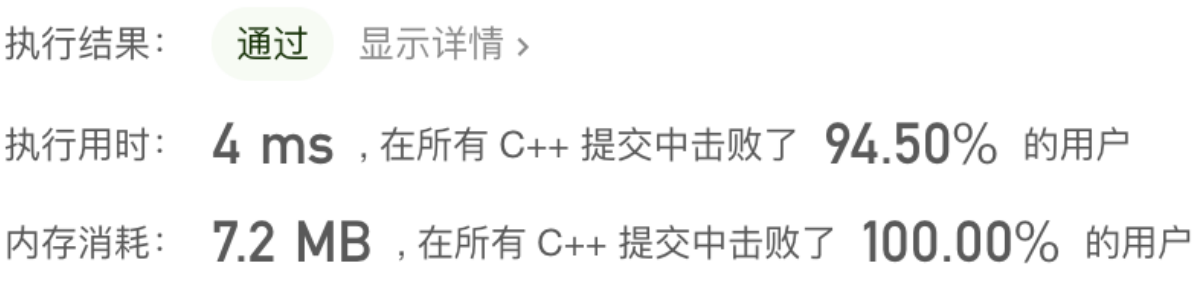
一些同学可能为了处理逻辑：每隔2k个字符的前k的字符，写了一堆逻辑代码或者再搞一个计数器，来统计2k，再统计前k个字符。

其实在遍历字符串的过程中，只要让 i += (2 * k)，i 每次移动 2 * k 就可以了，然后判断是否需要有反转的区间。

因为要找的也就是每2 * k 区间的起点，这样写，程序会高效很多。

所以当需要固定规律一段一段去处理字符串的时候，要想想在在for循环的表达式上做做文章。

性能如下：



那么这里具体反转的逻辑我们要不要使用库函数呢，其实用不用都可以，使用reverse来实现反转也没毛病，毕竟不是解题关键部分。

C++代码

使用C++库函数reverse的版本如下：

```
class Solution {
public:
    string reverseStr(string s, int k) {
        for (int i = 0; i < s.size(); i += (2 * k)) {
            // 1. 每隔 2k 个字符的前 k 个字符进行反转
            // 2. 剩余字符小于 2k 但大于或等于 k 个，则反转前 k 个字符
            if (i + k <= s.size()) {
                reverse(s.begin() + i, s.begin() + i + k );
                continue;
            }
            // 3. 剩余字符少于 k 个，则将剩余字符全部反转。
            reverse(s.begin() + i, s.begin() + s.size());
        }
        return s;
    }
};
```

i=0, 2k, 4k, n*2k

Reverses the order of the elements in the range [first,last).

那么我们也可以实现自己的reverse函数，其实和题目344. 反转字符串道理是一样的。

下面我实现的reverse函数区间是左闭右闭区间，代码如下：

```
class Solution {
public:
    void reverse(string& s, int start, int end) {
        for (int i = start, j = end; i < j; i++, j--) {
            swap(s[i], s[j]);
        }
    }
    string reverseStr(string s, int k) {
        for (int i = 0; i < s.size(); i += (2 * k)) {
            // 1. 每隔 2k 个字符的前 k 个字符进行反转
```

```
            // 2. 剩余字符小于 2k 但大于或等于 k 个，则反转前 k 个字符
            if (i + k <= s.size()) {
                reverse(s, i, i + k - 1);
                continue;
            }
            // 3. 剩余字符少于 k 个，则将剩余字符全部反转。
            reverse(s, i, s.size() - 1);
        }
        return s;
    }
};
```

其他语言版本

Java：

```
//解法一
class Solution {
    public String reverseStr(String s, int k) {
        StringBuffer res = new StringBuffer();
        int length = s.length();
        int start = 0;
        while (start < length) {
            // 找到k处和2k处
            StringBuffer temp = new StringBuffer();
            // 与length进行判断，如果大于length了，那就将其置为length
            int firstK = (start + k > length) ? length : start + k;
            int secondK = (start + (2 * k) > length) ? length : start + (2 * k);

            //无论start所处位置，至少会反转一次
            temp.append(s.substring(start, firstK));
            res.append(temp.reverse());

            // 如果firstK到secondK之间有元素，这些元素直接放入res里即可。
            if (firstK < secondK) { //此时剩余长度一定大于k。
                res.append(s.substring(firstK, secondK));
            }
            start += (2 * k);
        }
        return res.toString();
    }
}

//解法二（似乎更容易理解点）
//题目的意思其实概括为 每隔2k个反转前k个，尾数不够k个时候全部反转
class Solution {
    public String reverseStr(String s, int k) {
        char[] ch = s.toCharArray();
        for(int i = 0; i < ch.length; i += 2 * k){
            int start = i;
            //这里是判断尾数够不够k个来取决end指针的位置
            int end = Math.min(ch.length - 1, start + k - 1);
            //用异或运算反转
            while(start < end){
                ch[start] ^= ch[end];
                ch[end] ^= ch[start];
```

```
                ch[start] ^= ch[end];
                start++;
                end--;
            }
        }
        return new String(ch);
    }
}
```

Python:

```
class Solution(object):
    def reverseStr(self, s, k):
        """
        :type s: str
        :type k: int
        :rtype: str
        """
        from functools import reduce
        # turn s into a list
        s = list(s)

        # another way to simply use a[::-1], but i feel this is easier to understand
        def reverse(s):
            left, right = 0, len(s) - 1
            while left < right:
                s[left], s[right] = s[right], s[left]
                left += 1
                right -= 1
            return s

        # make sure we reverse each 2k elements
        for i in range(0, len(s), 2*k):
            s[i:(i+k)] = reverse(s[i:(i+k)])

        # combine list into str.
        return reduce(lambda a, b: a+b, s)
```

Go:

```
func reverseStr(s string, k int) string {
    ss := []byte(s)
    length := len(s)
    for i := 0; i < length; i += 2 * k {
        if i + k <= length {
            reverse(ss[i:i+k])
        } else {
            reverse(ss[i:length])
        }
    }
    return string(ss)
}

func reverse(b []byte) {
```

```
    left := 0
    right := len(b) - 1
    for left < right {
        b[left], b[right] = b[right], b[left]
        left++
        right--
    }
}
```

javaScript:

```
/**
 * @param {string} s
 * @param {number} k
 * @return {string}
 */
var reverseStr = function(s, k) {
    const len = s.length;
    let resArr = s.split("");
    for(let i = 0; i < len; i += 2 * k) {
        let l = i - 1, r = i + k > len ? len : i + k;
        while(++l < --r) [resArr[l], resArr[r]] = [resArr[r], resArr[l]];
    }
    return resArr.join("");
};
```

- 作者微信：程序员Carl
- B站视频：代码随想录
- 知识星球：代码随想录