344. 反转字符串

提示

编写一个函数，其作用是将输入的字符串反转过来。输入字符串以字符数组 s 的形式给出。

不要给另外的数组分配额外的空间，你必须**原地修改**输入数组，使用 $O(1)$ 的额外空间解决这一问题。

示例

示例 1:

```
输入：s = ["h","e","l","l","o"]
输出：["o","l","l","e","h"]
```

示例 2:

```
输入：s = ["H","a","n","n","a","h"]
输出：["h","a","n","n","a","H"]
```

提示

- $1 \leq s.length \leq 10^5$
- $s[i]$ 都是 ASCII 码表中的可打印字符

In [1]:
```python
from typing import List
class Solution:
    def reverseString(self, s: List[str]) -> None:
        """
        Do not return anything, modify s in-place instead.
        """
        length = len(s)
        temp = None
        for i in range(1, length//2+1):
            temp = s[i-1]
            s[i-1] = s[-i]
            s[-i] = temp

        # s[i], s[n - i - 1] = s[n - i - 1], s[i] 用这一句就能同时调换，无需用temp
```

In [ ]:
```python

```

541. 反转字符串 II

题目描述

给定一个字符串 s 和一个整数 k，从字符串开头算起，每计数至 $2k$ 个字符，就反转这 $2k$ 字符中的前 k 个字符。

- 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

示例 1:

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

示例 2:

```
输入：s = "abcd", k = 2
输出："bacd"
```

In [2]:
```python
# 第一次尝试，反转不成功

class Solution:
    def reverseStr(self, s: str, k: int) -> str:
        left = 0
        right = 2*k

        while right < len(s):
            remaining = s[left:]
            remaining_length = len(remaining)

            if remaining_length< k:
                s[left:].reverse()
```

```
            elif k <= remaining_length < 2*k:
                s[left:left+k].reverse()

            left += 2*k
            right += 2*k
        return s
```

字符串不可变： 在 Python 中，字符串是不可变的类型，无法直接通过切片（如 s[left:].reverse() 或 s[left:left+k].reverse()）来修改字符串。.reverse() 是列表的方法，而不是字符串的方法。

逻辑错误：

right 的计算和循环条件有问题。如果字符串长度小于 2k 时，right 可能永远无法满足循环条件。 即使 remaining 和 remaining_length 的逻辑正确，循环的核心功能并没有修改字符串。

无实际修改： 在 s[left:].reverse() 或 s[left:left+k].reverse() 这种操作中，切片只是生成了新的字符串或子列表，实际的 s 并未改变。

In [ ]:
```
# 第二次尝试，转换成列表，还是没有对列表操作
# 因为切片之后会产生新的列表

class Solution:
    def reverseStr(self, s: str, k: int) -> str:
        s = list(s)
        left = 0
        right = 2*k

        while right < len(s):
            remaining = s[left:]
            remaining_length = len(remaining)

            if remaining_length< k:
                s[left:].reverse()
            elif k <= remaining_length < 2*k:
                s[left:left+k].reverse()

            left += 2*k
            right += 2*k

        return ''.join(s) # list转换成列表的方法
```

知识点：

- str不可变，不能通过.reverse()反转
- str = ''.join(list) # 可以转化list为string
- s[left:].reverse() 会尝试反转从 left 开始的切片，但这并不会修改原列表的内容，而是创建了一个新的切片。你需要明确对原列表操作。
- s[left:] = reversed(s[left:]) 才能反转原列表

In [ ]:
```
# 根据gpt提示下的答案，因为这里有知识点不知道

class Solution:
    def reverseStr(self, s: str, k: int) -> str:
        s = list(s)
        left = 0

        while left < len(s):
            remaining_length = len(s) - left

            if remaining_length< k:
                s[left:] = reversed(s[left:])
            elif k <= remaining_length < 2*k:
                s[left:left + k] = reversed(s[left:left + k])
            else:  # 剩余字符多于 2k，正常处理前 k 个字符
                s[left:left + k] = reversed(s[left:left + k])

            left += 2*k

        return ''.join(s) # list转换成列表的方法
```

In [ ]:
```
# 答案
class Solution:
    def reverseStr(self, s: str, k: int) -> str:
        i = 0
        chars = list(s)

        while i < len(chars):
            chars[i:i + k] = chars[i:i + k][::-1] # 反转后，更改原值为反转后值
            i += k * 2

        return ''.join(chars)
```

54. 替换数字

题目描述

给定一个字符串 s，它包含小写字母和数字字符，请编写一个函数，将字符串中的字母字符保持不变，而将每个数字字符替换为 `number`。

例如，对于输入字符串 `"a1b2c3"`，函数应该将其转换为 `"anumberbnumbercnumber"`。

输入描述

输入一个字符串 s，s 仅包含小写字母和数字字符。

输出描述

打印一个新的字符串，其中每个数字字符都被替换为了 `number`。

示例

输入：

```
a1b2c3
```

输出：

```
anumberbnumbercnumber
```

In [5]:
```python
class Solution:
    def replaceNumber(self, s: str) -> str:
        list_s = list(s)
        for i in range(len(list_s)):
            if list_s[i] in "0123456789":
                list_s[i] = 'number'
            else:
                continue
        return "".join(list_s)
```

In [ ]:
```python
# GPT优化版（差不多）
def replaceNumber(s: str) -> str:
    result = []
    for char in s:
        if '0' <= char <= '9':
            result.append('number')  # 替换数字
        else:
            result.append(char)  # 保持原字符
    return ''.join(result)  # 拼接结果
```

虽然用python比较简单，但是注意list的添加复杂度（并不会增加，因为相当于hashmap） Python 中字符串是不可变的，最后还是要转换为列表合并

时间复杂度

1. **字符串转列表**：
   - `list(s)` 将字符串转换为列表，时间复杂度为 $O(n)$，其中 n 是字符串的长度。
2. **遍历列表**：
   - `for i in range(len(list_s))` 遍历列表中的每个元素，执行 n 次循环。
   - 每次判断 `if list_s[i] in "0123456789"`：
     - `"0123456789"` 是一个长度为 10 的字符串。
     - Python 的 `in` 操作符对字符串的复杂度为 $O(m)$，其中 m 是目标字符串的长度。
     - 在本例中，$m = 10$，这是一个常数，因此可以视为 $O(1)$。
3. **拼接字符串**：
   - `''.join(list_s)` 将列表重新拼接为字符串，时间复杂度为 $O(n)$。

综合上述步骤，总的时间复杂度为：

$$O(n) + O(n \cdot 1) + O(n) = O(n)$$

空间复杂度

1. **列表存储**：
   - `list(s)` 创建了一个与输入字符串等长的列表，空间复杂度为 $O(n)$。
2. **返回新字符串**：
   - `''.join(list_s)` 会创建一个新的字符串，长度与原字符串相同，空间复杂度为 $O(n)$。

综合上述，空间复杂度为：

$$O(n)$$

在 Python 中，**替换列表中的元素不会有额外复杂度**，因为列表的索引操作和赋值操作都是常数时间复杂度 $O(1)$。

具体来说：

替换列表元素的复杂度

当你执行以下代码：

```
list_s[i] = 'number'
```

1. **定位元素**：
   - `list_s[i]` 使用索引访问列表中的元素。
   - 在 Python 的底层实现中，列表是一个动态数组，访问元素的复杂度为 $O(1)$。
2. **替换元素**：
   - 替换操作只是将新值 `'number'` 写入已经分配好的内存地址，不涉及其他操作。
   - 替换操作的复杂度也是 $O(1)$。

因此，**单次替换的复杂度是 $O(1)$**。

In [6]:
```python
# 测试用例
solution = Solution()

# 示例测试
s1 = "a1b2c3"
output1 = solution.replaceNumber(s1)
print(output1)  # 期望输出: "anumberbnumbercnumber"

# 边界测试：空字符串
s2 = ""
output2 = solution.replaceNumber(s2)
print(output2)  # 期望输出: ""

# 全是数字
s3 = "123456"
output3 = solution.replaceNumber(s3)
print(output3)  # 期望输出: "numbernumbernumbernumbernumbernumber"

# 全是字母
s4 = "abcdef"
output4 = solution.replaceNumber(s4)
print(output4)  # 期望输出: "abcdef"

# 混合测试
s5 = "abc123xyz456"
output5 = solution.replaceNumber(s5)
print(output5)  # 期望输出: "abcnumbernumbernumberxyznumbernumbernumber"
```

```
anumberbnumbercnumber

numbernumbernumbernumbernumbernumber
abcdef
abcnumbernumbernumberxyznumbernumbernumber
```

In [ ]:
```python

```