# 题目

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 1：

输入：s = "abcdefg", k = 2

输出："bacdfeg"

示例 2：

输入：s = "abcd", k = 2

输出："bacd"

提示：

1 <= s.length <= 104

s 仅由小写英文组成

1 <= k <= 104

# 分析

## 方法一：模拟

思路：

你可以按照题目描述的要求对字符串进行处理。可以通过循环遍历字符串，每次处理 2k 个字符，对其中的前 k 个字符进行反转。

具体步骤如下：

1、初始化变量 i 为 0，表示当前处理的起始位置。

2、循环遍历字符串，每次处理 2k 个字符：

- 反转从 i 开始的前 k 个字符。

- 如果剩余字符少于 k 个，则将剩余字符全部反转。

- 更新 i 的值为 i + 2k，继续下一轮循环。

3、返回处理后的字符串。

以下是实现这个算法的代码：

class Solution {

public:

string reverseStr(string s, int k) {

int n = s.length();

for (int i = 0; i < n; i += 2 \* k) {

int left = i;

int right = min(i + k - 1, n - 1);

while (left < right) {

swap(s[left++], s[right--]);

}

}

return s;

}

};

在这个实现中，每次循环处理 2k 个字符，其中 `left` 表示当前处理的起始位置，`right` 表示当前处理的结束位置，根据题目要求对其中的前 k 个字符进行反转。

## 方法二：双指针

思路：

你可以使用双指针来实现这个算法。具体步骤如下：

1、初始化变量 i 和 j 分别为 0 和 k-1，表示当前处理的区间为 [i, j]。

2、循环遍历字符串，每次处理一个区间 [i, j]：

- 反转区间 [i, j] 的字符。

- 更新 i 和 j 的值为 j + 1 和 min(j + k, n - 1)，继续下一轮循环。

3、返回处理后的字符串。

以下是使用双指针实现的代码：

class Solution {

public:

string reverseStr(string s, int k) {

int n = s.length();

for (int i = 0; i < n; i += 2 \* k) {

int j = min(i + k - 1, n - 1);

reverse(s.begin() + i, s.begin() + j + 1);

}

return s;

}

};

在这个实现中，每次循环处理一个区间 `[i, j]`，并使用 `reverse` 函数对该区间进行反转。

class Solution {

public:

string reverseStr(string s, int k) {

int n = s.length();

for (int i = 0; i < n; i += 2 \* k) {

reverse(s.begin() + i, s.begin() + min(i + k, n));

}

return s;

}

};