# 替换空格

请实现一个函数，把字符串中的每个空格替换成 %20, 例如输入 “Hello my friend”,则输出 “Hello%20my%20friend”

## 解法一：

从头到尾遍历字符串，遇到空格则把后面的字符向右移动，然后把空格替换成 %20。 这样假设字符串的长度是 n，对每个空格字符，需要移动后面 O(n) 个字符，因此对于含有 n 个空格字符的字符串而言，时间复杂度是 O(n2)

## 解法二：时间复杂度为 O(n)

- 首先遍历一遍字符串，统计空格的数量，然后使用空格数量乘以2加上原来的长度就计算出了最终字符串的长度。
- 从字符串的后面开始复制，准备两个下标：P1 和 P2， P1 指向原字符串的末尾，P2 指向新字符串的末尾。
- 接下来移动 P1 逐个把 P1 复制到 P2 位置，直到碰到第一个空格，P1 需要移动 1 位，而 P2 之前需要插入 %20 P2 需要移动 3 位。
- 依次向前重复上面的操作，直到 P1 等于 P2，复制完毕。

```
public class ReplaceSpaces_05_01 {
    public static void main(String[] args) {
        String[] test = {"Hello", " ", "my", " ", "friend"};
        replaceBlank(test);
    }

    private static void replaceBlank(String[] strs) {
        if (strs == null || strs.length <= 0) return;
        // 原始字符串的长度
        int originalLength = strs.length;
        // 空格的个数
        int numberOfBlank = 0;
        int i = 0;
        while (i < strs.length) {
            if (" ".equals(strs[i]))
                ++numberOfBlank;
            ++i;
        }
        // 创建一个新的数组，长度为原始长度加上空格 * 2
        int newLength = originalLength + numberOfBlank * 2;
        String[] newStrs = new String[newLength];

        int indexOfOriginal = originalLength - 1;
        int indexOfNew = newLength - 1;
        while (indexOfOriginal >= 0 && indexOfNew >= indexOfOriginal) {
            if (" ".equals(strs[indexOfOriginal])) {
                newStrs[indexOfNew--] = "0";
```

```
                newStrs[indexOfNew--] = "2";
                newStrs[indexOfNew--] = "%";
            } else {
                newStrs[indexOfNew--] = strs[indexOfOriginal];
            }
            --indexOfOriginal;
        }
        System.out.println("original strs: " + printStringArray(strs));
        System.out.println("new strs: " + printStringArray(newStrs));
    }

    private static String printStringArray(String[] array) {
        StringBuilder stringBuilder = new StringBuilder();
        for(String str : array) {
            stringBuilder.append(str);
        }
        return stringBuilder.toString();
    }
}
```