题目：剑指Offer 05.替换空格

https://leetcode-cn.com/problems/ti-huan-kong-ge-lcof/

请实现一个函数，把字符串 s 中的每个空格替换成"%20"。

示例 1： 输入：s = "We are happy."
输出："We%20are%20happy."

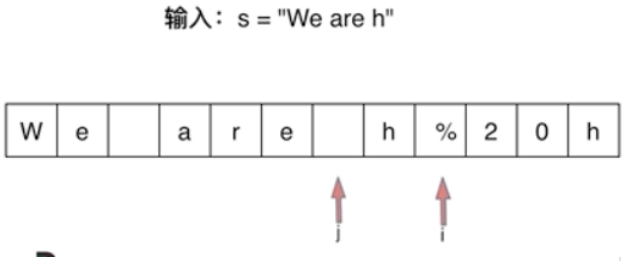
思路

如果想把这道题目做到极致，就不要只用额外的辅助空间了！

首先扩充数组到每个空格替换成"%20"之后的大小。

然后从后向前替换空格，也就是双指针法，过程如下：

i指向新长度的末尾，j指向旧长度的末尾。



有同学问了，为什么要从后向前填充，从前向后填充不行么？

从前向后填充就是O(n^2)的算法了，因为每次添加元素都要将添加元素之后的所有元素向后移动。

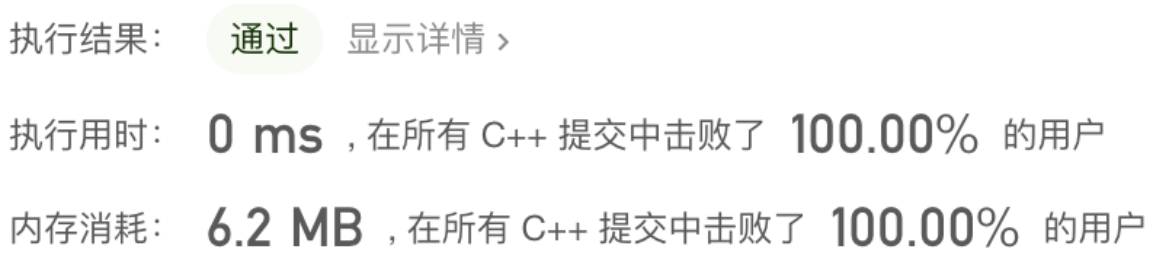
其实很多数组填充类的问题，都可以先预先给数组扩容带填充后的大小，然后在从后向前进行操作。

到

这么做有两个好处：

1. 不用申请新数组。
2. 从后向前填充元素，避免了从前先后填充元素要来的 每次添加元素都要将添加元素之后的所有元素向后移动。

时间复杂度，空间复杂度均超过100%的用户。



C++代码如下：

public member function
std::string::resize
void resize (size_t n);
void resize (size_t n, char c);
Resize string
Resizes the string to a length of *n* characters.
If *n* is smaller than the current string length, the current value is shortened to its first *n* character, removing the characters beyond the *n*th.
If *n* is greater than the current string length, the current content is extended by inserting at the end as many characters as needed to reach a size of *n*. If *c* is specified, the new elements are initialized as copies of *c*, otherwise, they are *value-initialized characters* (null characters).

```
class Solution {
public:
    string replaceSpace(string s) {
        int count = 0; // 统计空格的个数
        int sOldSize = s.size();
        for (int i = 0; i < s.size(); i++) {
            if (s[i] == ' ') {
                count++;
            }
        }
        // 扩充字符串s的大小，也就是每个空格替换成"%20"之后的大小
        s.resize(s.size() + count * 2);
        int sNewSize = s.size();
        // 从后先前将空格替换为"%20"
        for (int i = sNewSize - 1, j = sOldSize - 1; j < i; i--, j--) {
            if (s[j] != ' ') {
                s[i] = s[j];
            } else {
                s[i] = '0';
                s[i - 1] = '2';
                s[i - 2] = '%';
                i -= 2;
            }
        }
        return s;
    }
};
```

- 时间复杂度：O(n)
- 空间复杂度：O(1)

此时算上本题，我们已经做了七道双指针相关的题目了分别是：

- 27.移除元素
- 15.三数之和
- 18.四数之和
- 206.翻转链表
- 142.环形链表II
- 344.反转字符串

拓展

这里也给大家拓展一下字符串和数组有什么差别，

字符串是若干字符组成的有限序列，也可以理解为是一个字符数组，但是很多语言对字符串做了特殊的规定，接下来我来说一说C/C++中的字符串。

在C语言中，把一个字符串存入一个数组时，也把结束符 '\0'存入数组，并以此作为该字符串是否结束的标志。

例如这段代码：

```
char a[5] = "asd";
for (int i = 0; a[i] != '\0'; i++) {
}
```

在C++中，提供一个string类，string类会提供 size接口，可以用来判断string类字符串是否结束，就不用'\0'来判断是否结束。

例如这段代码:

```
string a = "asd";
for (int i = 0; i < a.size(); i++) {
}
```

那么vector< char > 和 string 又有什么区别呢?

其实在基本操作上没有区别，但是 string提供更多的字符串处理的相关接口，例如string 重载了+，而vector却没有。

所以想处理字符串，我们还是会定义一个string类型。

其他语言版本

Java：

```
//使用一个新的对象，复制 str，复制的过程对其判断，是空格则替换，否则直接复制，类似于数组复制
public static String replaceSpace(StringBuffer str) {
        if (str == null) {
            return null;
```

```
        }
                //选用 StringBuilder 单线程使用，比较快，选不选都行
        StringBuilder sb = new StringBuilder();
                //使用 sb 逐个复制 str ，碰到空格则替换，否则直接复制
        for (int i = 0; i < str.length(); i++) {
                //str.charAt(i) 为 char 类型，为了比较需要将其转为和 " " 相同的字符串类型
            if (" ".equals(String.valueOf(str.charAt(i)))){
                sb.append("%20");
            } else {
                sb.append(str.charAt(i));
            }
        }
        return sb.toString();
    }
```

Go:

```
// 遍历添加
func replaceSpace(s string) string {
    b := []byte(s)
    result := make([]byte, 0)
    for i := 0; i < len(b); i++ {
        if b[i] == ' ' {
            result = append(result, []byte("%20")...)
        } else {
            result = append(result, b[i])
        }
    }
    return string(result)
}

// 原地修改
func replaceSpace(s string) string {
    b := []byte(s)
    length := len(b)
    spaceCount := 0
    // 计算空格数量
    for _, v := range b {
        if v == ' ' {
            spaceCount++
        }
    }
    // 扩展原有切片
    resizeCount := spaceCount * 2
    tmp := make([]byte, resizeCount)
    b = append(b, tmp...)
    i := length - 1
    j := len(b) - 1
    for i >= 0 {
        if b[i] != ' ' {
            b[j] = b[i]
            i--
            j--
        } else {
            b[j] = '0'
            b[j-1] = '2'
            b[j-2] = '%'
```

```
            i--
            j = j - 3
        }
    }
    return string(b)
}
```

- 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