# 题目

给定一个非空字符串s，最多删除一个字符。判断是否能成为回文字符串。

**示例 1:**

输入: "aba"

输出: True

**示例 2:**

输入: "abca"

输出: True

解释: 你可以删除c字符。

**注意:**

字符串只包含从 a-z 的小写字母。字符串的最大长度是50000。

# 分析

## 方法一：双指针

思路：

你可以使用双指针的方法来解决这个问题。首先，从字符串的两端开始向中间移动两个指针，一个指向字符串的起始位置，另一个指向字符串的末尾位置。如果两个指针指向的字符相同，就继续向中间移动；如果两个指针指向的字符不同，就尝试删除其中一个字符，然后判断剩余字符串是否是回文字符串。

具体步骤如下：

1、初始化两个指针 left 和 right，分别指向字符串的起始位置和末尾位置。

2、当 left < right 时，执行以下步骤：

- 如果 s[left] == s[right]，则继续向中间移动：left++，right--。

- 如果 s[left] != s[right]，则尝试删除其中一个字符，判断剩余字符串是否是回文字符串：

- 尝试删除左边的字符：left++，并将一个标志变量设为 true。

- 尝试删除右边的字符：right--，并将标志变量设为 true。

- 如果两次尝试都不是回文字符串，则返回 false。

3、如果循环结束后仍未返回 false，则返回 true。

这里需要注意的是，为了避免重复判断字符串是否是回文字符串，可以使用一个辅助函数来判断字符串是否是回文字符串。

你可以使用以下代码来实现这个算法：

class Solution {

public:

bool isPalindrome(string s, int start, int end) {

while (start < end) {

if (s[start] != s[end]) {

return false;

}

start++;

end--;

}

return true;

}

bool validPalindrome(string s) {

int left = 0, right = s.length() - 1;

while (left < right) {

if (s[left] != s[right]) {

return isPalindrome(s, left + 1, right) || isPalindrome(s, left, right - 1);

}

left++;

right--;

}

return true;

}

};

在这个实现中，isPalindrome 函数用于判断从指定位置开始到结束位置的子字符串是否是回文字符串。在 validPalindrome 函数中，使用双指针 left 和 right 来遍历字符串，如果遇到不相同的字符，则尝试删除其中一个字符并判断剩余部分是否是回文字符串。如果两次尝试都不是回文字符串，则返回 false。如果循环结束后仍未返回 false，则说明可以通过删除一个字符使字符串成为回文字符串，返回 true。

class Solution {

public:

bool checkPalindrome(const string& s, int low, int high) {

for (int i = low, j = high; i < j; ++i, --j) {

if (s[i] != s[j]) {

return false;

}

}

return true;

}

bool validPalindrome(string s) {

int low=0,high=s.size()-1;

while(low<high)

{

if(s[low]==s[high])

{

low++;

high--;

}

else

{

return checkPalindrome(s,low+1,high) || checkPalindrome(s,low,high-1);

}

}

return true;

}

};