# 题目

给你一个由若干0和1组成的字符串s，请你计算并返回将该字符串分割成两个非空子字符串（即左子字符串和右子字符串）所能获得的最大得分。

「分割字符串的得分」为左子字符串中0的数量加上右子字符串中1的数量。

示例 1：

输入：s = "011101"

输出：5

解释：

将字符串 s 划分为两个非空子字符串的可行方案有：

左子字符串 = "0" 且 右子字符串 = "11101"，得分 = 1 + 4 = 5

左子字符串 = "01" 且 右子字符串 = "1101"，得分 = 1 + 3 = 4

左子字符串 = "011" 且 右子字符串 = "101"，得分 = 1 + 2 = 3

左子字符串 = "0111" 且 右子字符串 = "01"，得分 = 1 + 1 = 2

左子字符串 = "01110" 且 右子字符串 = "1"，得分 = 2 + 1 = 3

示例 2：

输入：s = "00111"

输出：5

解释：当 左子字符串 = "00" 且 右子字符串 = "111" 时，我们得到最大得分 = 2 + 3 = 5

示例 3：

输入：s = "1111"

输出：3

提示：

2 <= s.length <= 500

字符串 s 仅由字符 '0' 和 '1' 组成。

# 分析

## 方法一：枚举

思路：

可以通过遍历字符串 s 一次来解决。我们可以考虑将字符串分成两部分，左边的部分只包含 0，右边的部分只包含 1。我们可以用两个变量 zeros 和 ones 来分别表示左边部分的 0 的数量和右边部分的 1 的数量。初始化时，zeros 为左边部分的第一个字符是 0 的数量，ones 为右边部分的所有字符是 1 的数量。然后，我们从第二个字符开始遍历字符串 s，如果遇到 0，则 zeros 加一；如果遇到 1，则 ones 减一。在遍历过程中，我们不断更新最大得分，即 zeros + ones 的最大值。最后返回最大得分即可。

代码：

class Solution {

public:

int maxScore(string s) {

int zeros = (s[0] == '0') ? 1 : 0; // 左边部分的 0 的数量

int ones = count(s.begin() + 1, s.end(), '1'); // 右边部分的 1 的数量

int maxScore = zeros + ones; // 初始化最大得分为左右两部分数量之和

for (int i = 1; i < s.size() - 1; ++i) {

if (s[i] == '0') {

zeros++;

} else {

ones--;

}

maxScore = max(maxScore, zeros + ones);

}

return maxScore;

}

};