# 题目

给你一个字符串 s 和两个整数 x 和 y 。你可以执行下面两种操作任意次。

删除子字符串 "ab" 并得到 x 分。

比方说，从 "cabxbae" 删除 ab ，得到 "cxbae" 。

删除子字符串"ba" 并得到 y 分。

比方说，从 "cabxbae" 删除 ba ，得到 "cabxe" 。

请返回对 s 字符串执行上面操作若干次能得到的最大得分。

示例 1：

输入：s = "cdbcbbaaabab", x = 4, y = 5

输出：19

解释：

- 删除 "cdbcbbaaabab" 中加粗的 "ba" ，得到 s = "cdbcbbaaab" ，加 5 分。

- 删除 "cdbcbbaaab" 中加粗的 "ab" ，得到 s = "cdbcbbaa" ，加 4 分。

- 删除 "cdbcbbaa" 中加粗的 "ba" ，得到 s = "cdbcba" ，加 5 分。

- 删除 "cdbcba" 中加粗的 "ba" ，得到 s = "cdbc" ，加 5 分。

总得分为 5 + 4 + 5 + 5 = 19 。

示例 2：

输入：s = "aabbaaxybbaabb", x = 5, y = 4

输出：20

提示：

1 <= s.length <= 10^5

1 <= x, y <= 10^4

s 只包含小写英文字母。

# 分析

## 方法一：辅助栈

思路：

使用两个栈来模拟操作，首先处理了 ba 这种情况，然后再处理 ab 这种情况，最终计算得分。

代码：

class Solution {

public:

int maximumGain(std::string a, int x, int y) {

std::stack<char> s, t; // 用两个栈来模拟操作

int ret = 0; // 最终得分

// 优化：如果 x > y，交换 x 和 y，并反转字符串 a，使得处理 ba 的操作更加高效

if (x > y) {

std::swap(x, y);

std::reverse(a.begin(), a.end());

}

// 先处理 ba 情况

for (char c : a) {

if (c != 'a') {

s.push(c); // 不是 a 的字符直接入栈

} else {

// 遇到 a，判断栈顶是否为 b，如果是则可以得分，否则继续入栈

if (!s.empty() && s.top() == 'b') {

s.pop(); // 弹出 b

ret += y; // 得分加上 y

} else {

s.push(c); // 否则继续入栈 a

}

}

}

// 处理完 ba 情况后，再处理 ab 情况

while (!s.empty()) {

char c = s.top();

s.pop();

if (c != 'a') {

t.push(c); // 不是 a 的字符直接入栈

} else {

// 遇到 a，判断栈顶是否为 b，如果是则可以得分，否则继续入栈

if (!t.empty() && t.top() == 'b') {

t.pop(); // 弹出 b

ret += x; // 得分加上 x

} else {

t.push(c); // 否则继续入栈 a

}

}

}

return ret; // 返回最终得分

}

};

这段代码使用了两个栈 s 和 t 分别处理 ba 和 ab 的情况，遍历字符串，如果遇到 a，则判断栈顶是否为 b，如果是则得分并弹出 b，否则继续入栈 a。处理完 ba 情况后，再处理 ab 情况，最终返回得分。