# 题目

给定 S 和 T 两个字符串，当它们分别被输入到空白的文本编辑器后，判断二者是否相等，并返回结果。 # 代表退格字符。

注意：如果对空文本输入退格字符，文本继续为空。

示例 1：

输入：S = "ab#c", T = "ad#c"

输出：true

解释：S 和 T 都会变成 “ac”。

示例 2：

输入：S = "ab##", T = "c#d#"

输出：true

解释：S 和 T 都会变成 “”。

示例 3：

输入：S = "a##c", T = "#a#c"

输出：true

解释：S 和 T 都会变成 “c”。

示例 4：

输入：S = "a#c", T = "b"

输出：false

解释：S 会变成 “c”，但 T 仍然是 “b”。

提示：

1 <= S.length <= 200

1 <= T.length <= 200

S 和 T 只含有小写字母以及字符 '#'。

进阶：

你可以用 O(N) 的时间复杂度和 O(1) 的空间复杂度解决该问题吗？

# 分析

## 方法一：字符串

代码：

class Solution {

public:

bool backspaceCompare(string S, string T)

{

string stack1 = "", stack2 = "";

for (auto s : S)

if (s == '#' && stack1 != "")

stack1.pop\_back();

else if (s != '#') stack1.push\_back(s);

for (auto t : T)

if (t == '#' && stack2 != "")

stack2.pop\_back();

else if (t != '#') stack2.push\_back(t);

return stack1 == stack2;

}

};

## 方法二：栈

**思路：**

用两个栈，不断添加字符，遇上#就删除尾部字符，最后对比两个栈是否相同。

class Solution {

public:

bool backspaceCompare(string S, string T) {

stack<char> s1;

stack<char> s2;

//初始化s1

for(int i=0;i<S.length();i++)

{

if(S[i]=='#' && !s1.empty())

{

s1.pop();//遇到#,退出一个元素

}

else if(S[i]!='#')

{

s1.push(S[i]);

}

}

for(int i=0;i<T.length();i++)

{

if(T[i]=='#' && !s2.empty())

{

s2.pop();

}

else if(T[i]!='#')

{

s2.push(T[i]);

}

}

if(s1.size()==s2.size())

{

while(!s1.empty() && !s2.empty())

{

if(s1.top()==s2.top())

{

s1.pop();

s2.pop();

}

else

{

return false;

}

}

return true;

}

else

{

return false;

}

}

};