# 题目

给你一个字符串 s，它由数字（'0' - '9'）和 '#' 组成。我们希望按下述规则将 s 映射为一些小写英文字符：

* 字符（'a' - 'i'）分别用（'1' - '9'）表示。
* 字符（'j' - 'z'）分别用（'10#' - '26#'）表示。

返回映射之后形成的新字符串。

题目数据保证映射始终唯一。

示例 1：

输入：s = "10#11#12"

输出："jkab"

解释："j" -> "10#" , "k" -> "11#" , "a" -> "1" , "b" -> "2".

示例 2：

输入：s = "1326#"

输出："acz"

提示：

1 <= s.length <= 1000

s[i] 只包含数字（'0'-'9'）和 '#' 字符。

s 是映射始终存在的有效字符串。

# 分析

## 方法一：遍历

可以通过逆向遍历字符串 s，并根据 '#' 符号的位置进行不同的处理来实现。具体的步骤如下：

1、初始化一个空字符串 result 用于存储最终的映射结果。

2、从字符串 s 的末尾开始向前遍历：

如果当前字符是 '#'，说明该字符是一个数字的结尾，需要将前两个字符转换为对应的字母，并将结果添加到 result 中，然后将遍历索引向前移动两位。

如果当前字符不是 '#'，说明该字符是一个单独的数字，将其转换为对应的字母，并将结果添加到 result 中，然后将遍历索引向前移动一位。

3、最后返回逆序后的 result。

代码：

class Solution {

public:

string freqAlphabets(string s) {

string result = "";

int n = s.size();

for (int i = n - 1; i >= 0; ) {

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

int num = (s[i - 2] - '0') \* 10 + (s[i - 1] - '0'); // 计算两位数字

result = char('a' + num - 1) + result; // 转换为字母并添加到结果中

i -= 3; // 向前移动两位

} else {

result = char('a' + (s[i] - '0' - 1)) + result; // 转换为字母并添加到结果中

i -= 1; // 向前移动一位

}

}

return result;

}

};