题解：回文串

问题描述

本问题的目标是确定将一个字符串转化为回文串所需插入的最小字符数。回文串是指从前向后和从后向前读都相同的字符串。该问题需要分析多个测试用例，每个测试用例包含一个字符串。

方法

1. 动态规划 (DP) 解法

该解法使用动态规划的方法来找出将给定字符串变为回文串所需的最小插入次数。

关键概念

- **动态规划表**：我们使用一个二维数组 `f`，其中 `f[i][j]` 表示将子串 `s[i..j]` 转换为回文串所需的最小插入次数。
- **基础情况**：单个字符始终是回文串，因此 `f[i][i] = 0`。
- **递推关系**：
  - 如果子串两端的字符相等（`s[i] == s[j]`），则此时无需插入：[f[i][j] = f[i+1][j-1]]
  - 如果两端字符不相等，我们需要考虑两种情况：
    - 在 `s[i]` 后插入一个字符（这导致 `f[i+1][j] + 1`）
    - 在 `s[j]` 前插入一个字符（这导致 `f[i][j-1] + 1`） 因此我们有：[f[i][j] = \min(f[i+1][j], f[i][j-1]) + 1]

2. 实现

该实现读取多个测试用例，处理每个字符串以填充动态规划表，并输出结果。以下是实现代码：

```
#include<iostream>
#include<bits/stdc++.h>
using namespace std;

char s[5005];
int t;
int l;
int f[5100][5100];

void dp() {
    for (int len = 2; len <= l; len++) {
        for (int i = 0; i + len - 1 < l; i++) {
            int j = i + len - 1;
            if (s[i] == s[j]) {
                f[i][j] = f[i + 1][j - 1];
            } else {
                f[i][j] = min(f[i + 1][j], f[i][j - 1]) + 1;
            }
        }
```

```
    }
    printf("%d\n", f[0][l - 1] + l);
}

int main() {
    cin >> t;
    while (t--) {
        scanf("%d%s", &l, s);
        d
```