I 回文串串文回

题目描述

给定一个长度为n的字符串 S，S仅有26个小写字母组成。你每次操作可以向 S的任意位置插入任意一个字符。现在你需要向 S中插入一些字符（也可以不插入），使得 S变为回文串 S，请你求出 S'的最小长度。

一个字符串 S称为回文串当且仅当 S与 S的逆序字符串S（rev）完全相同，如 `ABCBA` 是回文串，但 `ABCA` 不是回文串。

题目分析

要使字符串 S 变为回文串 S′，最直接的方法是在 S 的末尾添加 S 的逆序字符串 S_rev。这样，S′ 就是一个回文串。因此，S′ 的最小长度为 2n，其中 n 是字符串 S 的长度。

即使我们不添加 S_rev，S′ 仍然可以是回文串。这是因为在 S′ 的前半部分可以添加 S 的逆序字符串的一部分，使得 S′ 的后半部分与 S 的逆序字符串的剩余部分相同。这样，S′ 仍然是一个回文串。

因此，S′ 的最小长度是 2n 减去 S 的最长回文子序列的长度。为了计算 S 的最长回文子序列的长度，我们可以使用动态规划算法。

代码实现

```c
#include <stdio.h>
#include <string.h>
#include <math.h>

int min(int a, int b) {
    return a < b ? a : b;
}

int dp[5050][5050]; //在循环体外创建二维数组存储字符串可以有效避免REG

int longestCommonSubsequence(char *s1, char *s2) {
    int m = strlen(s1), n = strlen(s2);
    //int dp[m + 1][n + 1];
    //memset(dp, 0, sizeof(dp));
    for (int i = 1; i <= m; i++) {
        for (int j = 1; j <= n; j++) {
            if (s1[i - 1] == s2[j - 1]) {
                dp[i][j] = dp[i - 1][j - 1] + 1;
            } else {
                dp[i][j] = fmax(dp[i - 1][j], dp[i][j - 1]);
            }
        }
    }
    return dp[m][n];
}
//动态规划算法计算两个字符串的最长公共子序列，即LCS算法
char s[5050];
char s2[5050];
```

```
int main() {

    int T;
    scanf("%d", &T);
    while (T--) {

        int n;
        scanf("%d", &n);
        scanf("%s", s);
        for (int i = 0; i < n; i++) {
            s2[i]
                = s[n - i - 1];
        }
        s2[n] = '\0'; //此时s2即为s的逆序列
        int lcs = longestCommonSubsequence(s, s2); //找出s和其逆序列s2的最长公共子序列
的长度
        printf("%d\n", 2 * n - lcs); //2n-lcs即为所求的值
    }
    return 0;
}
```