Problem I. Perfect Palindrome

Given a string $S = s_0 s_1 \cdots s_{n-1}$ of length n, let $f(S, d)$ be the string obtained by shifting S to the left d times. That is $f(S, d) = s_{(d+0) \bmod n} s_{(d+1) \bmod n} \cdots s_{(d+n-1) \bmod n}$. We say S is a perfect palindrome if for **all** non-negative integer d, $f(S, d)$ is a palindrome.

You're now given a string $A = a_0 a_1 \cdots a_{n-1}$ of length n consisting only of lower-cased English letters. You can perform the following operation on A any number of times (including zero times): Choose an integer i such that $0 \le i < n$ and change a_i to any lower-cased English letter.

Calculate the minimum number of operations needed to change A into a perfect palindrome.

We say a string $P = p_0 p_1 \cdots p_{n-1}$ of length n is a palindrome, if $p_i = p_{n-1-i}$ for all $0 \le i < n$.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first and only line contains a string $a_0 a_1 \cdots a_{n-1}$ ($1 \le n \le 10^5$) consisting only of lower-cased English letters.

It is guaranteed that the total length of strings of all test cases will not exceed 10^6.

Output

For each test case output one line containing one integer indicating the minimum number of operations needed to change A into a perfect palindrome.

Example

standard input	standard output
2 abcb xxx	2 0

Note

For the first sample test case, we can change the first and the third characters to 'b' so the string becomes "bbbb". It is easy to see that for all non-negative integer d, f("bbbb", d) = "bbbb" and "bbbb" is a palindrome, so "bbbb" is a perfect palindrome. These changes cost us 2 operations and it can be proven that this is the minimum number of operations needed.

For the second sample test case, "xxx" is already a perfect palindrome, so no changes are needed.