题目

E 对称字符串

时间限制：2000ms 内存限制：65536kb

通过率：61/124 (49.19%) 正确率：61/350 (17.43%)

题目描述

给定字符串 s，请你求出对于 $i = 1, 2, \ldots, |s|$ 而言，长度为 i 的前缀和长度为 i 后缀是否相同。

输入格式

第一行一个正整数 t（$1 \leq t \leq 10$），表示数据组数。

对于每组数据，一行一个由小写字母、大写字母和数字组成的字符串 s（$1 \leq |s| \leq 10^6$），含义同题目描述。

输出格式

对于每组数据，输出一行从小到大若干个正整数，表示符合条件的 i。

思路

使用KMP算法，具体来说，是找出字符串中每个位置的最长前缀后缀匹配长度。

1. `vector<int> prefix_function(const string &s)`：返回一个包含每个位置的最长前缀后缀匹配长度。
2. `int cnt = 0;`：初始化计数器，用于记录后续的循环中找到的匹配长度。
3. `int x = kmp.back();`：取`kmp`数组的最后一个元素，即字符串`s`的最长前缀后缀匹配长度。
4. `while (x != 0)`：循环，每次将找到的匹配长度存储在数组`ans`中，并将`x`更新为下一个匹配位置的值。
5. `for (int i = cnt - 1; i >= 0; i--)`：反向遍历数组`ans`，输出匹配长度。
6. `cout << s.length();`：输出字符串的长度，即整个字符串作为一个匹配。

代码

```
#include <iostream>
#include <vector>
#include<iostream>
#include<cmath>
#include<cstdio>
#include<string>
#include<cstdlib>
#include<algorithm>
#include<vector>
using namespace std;
const int N = 1000050;
int ans[N];
vector<int> prefix_function(const string &s) {
    int n = s.length();
    vector<int> pi(n);

    for (int i = 1; i < n; i++) {
        int j = pi[i - 1];
        while (j > 0 && s[i] != s[j]) {
            j = pi[j - 1];
        }
        if (s[i] == s[j]) {
            j++;
        }
        pi[i] = j;
    }

    return pi;
}

int main() {
    int t;
    cin >> t;

    while (t--) {
        string s;
        cin >> s;

        int n = s.length();
        vector<int> kmp = prefix_function(s);
        int cnt = 0;
        int x = kmp.back();
        while(x != 0){
        ans[cnt++] = x;
        x = kmp[x-1];
        }
        for(int i = cnt-1; i>=0; i--){
            cout<<ans[i]<<" ";
        }
        cout<<s.length();

        cout << endl;
```

```
    }

    return 0;
}
```

要注意的细节

别忘了最后输出字符串的长度，长度为 s 的前缀和长度为 s后缀就是字符串本身，肯定相等