对称字符串

21351023 李永琦

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。

解法 1

> s 下标从 0 开始，$s[i\ldots j]$ 表示首字符下标为 i、尾字符下标为 j 的子串。

从大到小考虑答案，记 s 的长度为 n，显然 n 为符合条件的解。

注意到 KMP 算法步骤中求出的前缀函数：$\pi[i]$ 为子串 $s[0\ldots i]$ 最长的相等的真前缀与真后缀的长度。那么 $\pi[n-1]$ 也符合条件。

再寻找比 $\pi[n-1]$ 小的解：若 j 符合题意，即 $s[0\ldots j-1]=s[n-j\ldots n-1]$，则 $s[0\ldots \pi[n-1]]$ 的长度为 j 的前缀和后缀也相等。故下一个解为 $\pi[\pi[n-1]-1]$。

根据以上分析递推，直至解为 0 即可。前缀函数可在 $O(n)$ 内求得。

解法 2

可枚举长度为 1 至 n 的前缀和后缀，使用字符串哈希判断前后缀是否相等。

> s 下标从 1 开始。

定义一个把字符串映射到整数的函数 f，通常采用多项式 Hash 的方法，将哈希值看成一个 b 进制的数对 M 取模的结果：对于一个长度为 l 的字符串 s，$f(s)=\sum_{i=1}^{l}s[i]\times b^{l-i}\pmod M$。例如，字符串 xyz 的哈希函数值为 xb^2+yb+z。为减小哈希碰撞的概率，M 需为素数，如 998244353。

求字符串子串的哈希值时，一般采取的方法是：

首先预处理 b 的幂，$p[i]=b^i \mod M$：

$$p[i]=\begin{cases}1, & i=0\\ bp[i-1] \mod M, & i\geq 1\end{cases}$$

对字符串 $O(n)$ 地预处理出每个前缀的哈希值，令 $g[i]$ 表示 $f(s[1\ldots i])$，即原串长度为 i 的前缀的哈希值，规定 $g[i]=0$，则有：

$$g[i]=\begin{cases}0, & i=0\\ (bg[i-1]+s[i]) \mod M, & i\geq 1\end{cases}$$

现在，我们想要用类似前缀和的方式快速求出 $f(s[l\ldots r])$。按照定义发现 $f(s[l\ldots r])=f[1\ldots s]-b^{r-l+1}f[1\ldots l-1]=g[s]-p[r-l+1]g[l-1]$。

代码 1

```cpp
#include <algorithm>
#include <iostream>
#include <vector>
using namespace std;
vector<int> prefix(string s)
{
    int n = (int)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;
}
void solve(string &s)
{
    vector<int> pi = prefix(s), ans;
    int x = (int)s.length();
    while (x)
    {
        ans.push_back(x);
        x = pi[x - 1];
    }
    for_each(ans.rbegin(), ans.rend(), [](const int &x) { cout << x << ' '; });
    cout << endl;
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    int t;
    string s;
    cin >> t;
    while (t--)
    {
        cin >> s;
        solve(s);
    }
    return 0;
}
```

代码 2

```
#include <cstring>
#include <iostream>
using namespace std;
typedef long long ll;
const int N = 2e6, B = 31, M = 998244353;
int p[N], g[N];
char s[N];
int Hash(int l, int r)
{
    return (ll)(g[r] - (ll)g[l - 1] * p[r - l + 1] % M + M) % M;
}
int main()
{
    int t;
    scanf("%d", &t);
    while (t--)
    {
        scanf("%s", s + 1);
        int n = strlen(s + 1);
        for (int i = p[0] = 1; i <= n; i++)
        {
            p[i] = (ll)p[i - 1] * B % M;
        }
        for (int i = 1; i <= n; i++)
        {
            g[i] = (ll)((ll)g[i - 1] * B + s[i]) % M;
        }
        for (int i = 1; i <= n; i++)
        {
            if (Hash(1, i) == Hash(n - i + 1, n))
            {
                printf("%d ", i);
            }
        }
        puts("");
    }
}
```