Omar And Strings

Attempted by: 1 / Accuracy: 100% / ★★★★★

Tag(s): Hashing, Medium, String Algorithms, Z-algorithm

There will be used terms Z-function, Manacher algorithm, hashes. You can read about it here and here.

Firstly we want to find longest palindromic prefix. It can be done using Manacher algorithm or using hashes. To find if some prefix is palindromic you can find hash of this prefix and hash of reversed prefix and check if they are equal. Same with the longest palindromic suffix.

We also need to find frequencies of each prefix in omeric string. Let's find for each **i** longest substring starting in position **i** which equals to the prefix of the string. It can be done using Z-function or hashes with binary search. Let Z[i] be the length of such substring. Now the number of times prefix with length L appears in the string equals the number of such indices **i** that Z[i] ≥ L.

Complexity of such solution is O(N).

Author Solution by omar khaled abdelaziz abdelnabi

```cpp
#include <bits/stdc++.h>

using namespace std;

int fail1 [200000 + 10];
int fail2 [200000 + 10];
int fail3 [200000 + 10];
int freq [200000 + 10];

int main()
{

    string s, c , prefix = "", suffix = "", omeric = "";

    cin >> s;
```

```
    c = s;

    reverse(c.begin(), c.end());
    prefix = s + "#" + c;
    suffix = c + "#" + s;

    for(int i = 1, k = 0; i < prefix.size(); i++){
        while(k > 0 && prefix[i] != prefix[k]) k = fail1[k - 1];
        if(prefix[i] == prefix[k]) k++;
        fail1[i] = k;
    }

    for(int i = 1, k = 0; i < suffix.size(); i++){
        while(k > 0 && suffix[i] != suffix[k]) k = fail2[k - 1];
        if(suffix[i] == suffix[k]) k++;
        fail2[i] = k;
    }

    string res1 = s.substr(s.size() - fail2[suffix.size() - 1], fail2[suffi
    string res2 = s.substr(0 , fail1[prefix.size() - 1]);

    omeric = res1 + res2;
    //cout << omeric << "\n";
    s = omeric;
    cout << omeric << "\n";

    for(int i = 1, k = 0; i < s.size(); i++){
        while(k > 0 && s[i] != s[k]) k = fail3[k - 1];
        if(s[i] == s[k]) k++;
        fail3[i] = k;
    }

    for(int i = 0; i < s.size(); i++) freq[ fail3[i] ]++;

    for(int i = s.size() - 1; i > 0; i--) freq[ fail3[i - 1] ] += freq[i];

    for(int i = s.size(); i > 0; i--) freq[i]++;

    for(int i = 1; i <= s.size(); i++) cout << freq[i] << " ";

    return 0;
}
```

Tester Solution

```
#define _CRT_SECURE_NO_WARNINGS
#pragma comment(linker, "/stack:16777216")
#include <string>
#include <vector>
#include <map>
#include <list>
#include <iterator>
```

```
#include <cassert>
#include <set>
#include <queue>
#include <iostream>
#include <sstream>
#include <stack>
#include <deque>
#include <cmath>
#include <memory.h>
#include <cstdlib>
#include <cstdio>
#include <cctype>
#include <algorithm>
#include <utility>
#include <time.h>
#include <complex>

using namespace std;

#define FOR(i,a,b) for(int i=(a);i<(b);++i)
#define RFOR(i,b,a) for(int i=(b)-1;i>=(a);--i)
#define FILL(A,val) memset(A,val,sizeof(A))

#define ALL(V) V.begin(),V.end()
#define SZ(V) (int)V.size()
#define PB push_back
#define MP make_pair

typedef long long Int;
typedef unsigned long long UInt;
typedef vector<int> VI;
typedef pair<int,int> PII;

const double Pi = acos(-1.0);
const int INF = 1000000000;
const int MAX = 200007;
const int MAX2 = 2000000;
const int BASE = 10;
const int ST = 1000000007;
const int CNT = 100;

const int MOD = 1000000007;

char s[MAX];
Int Hash[MAX];
Int RevHash[MAX];
Int pw[MAX];
int z[MAX];
int d[MAX];

int main()
{
        //freopen("out.txt" , "w" , stdout);
        //freopen("in.txt", "r", stdin);
```

```
62.         //freopen("puzzle.in", "r", stdin);
63.         //freopen("puzzle.out", "w", stdout);
64.
65.         pw[0] = 1;
66.         FOR(i,1,MAX)
67.                 pw[i] = pw[i - 1] * ST;
68.
69.         scanf("%s" , &s);
70.         int n = strlen(s);
71.
72.         assert(n >= 1 && n <= 100000);
73.
74.         FOR(i,0,n)
75.         {
76.                 assert(s[i] >= 'a' && s[i] <= 'z');
77.         }
78.
79.         Hash[0] = s[0];
80.         FOR(i,1,n)
81.         {
82.                 Hash[i] = Hash[i - 1] + s[i] * pw[i];
83.         }
84.         RevHash[n - 1] = s[n - 1];
85.         RFOR(i,n - 1, 0)
86.         {
87.                 RevHash[i] = RevHash[i + 1] + s[i] * pw[n - 1 - i];
88.         }
89.
90.         string t = "";
91.
92.         FOR(i,0,n)
93.         {
94.                 if ((Hash[n - 1] - (i ? Hash[i - 1] : 0)) == (RevHash[i] -
95.                 {
96.                         FOR(j,i,n)
97.                                 t += s[j];
98.                         break;
99.                 }
100.        }
101.
102.        RFOR(i, n, 0)
103.        {
104.                if (Hash[i] * pw[n - 1 - i] == (RevHash[0] - RevHash[i + 1]
105.                {
106.                        FOR(j,0,i + 1)
107.                                t += s[j];
108.                        break;
109.                }
110.        }
111.
112.        cout << t << endl;
113.
114.        n = t.length();
115.        int L = 0;
```

```
116.         int R = 0;
117.
118.         FOR(i,1,n)
119.         {
120.                 if (i <= R)
121.                 {
122.                         z[i] = min(R - i + 1, z[i - L]);
123.                 }
124.                 while (i + z[i] < n && t[z[i]] == t[i + z[i]])
125.                 {
126.                         ++z[i];
127.                 }
128.                 if (i + z[i] - 1 > R)
129.                 {
130.                         L = i;
131.                         R = i + z[i] - 1;
132.                 }
133.         }
134.
135.
136.         FOR(i,1,n)
137.         {
138.                 d[z[i]] ++;
139.         }
140.         RFOR(i,n,1)
141.         {
142.                 d[i - 1] += d[i];
143.         }
144.
145.         FOR(i,1,n + 1)
146.         {
147.                 cout << d[i] + 1 << ' ';
148.         }
149.         cout << endl;
150.
151.         return 0;
152. }
153.
```