KMP

border

对于字符串满足$Prefix[i] == Suffix[i]$,即前缀和后缀完全相同，则称为一个border

周期和循环节

循环周期

对于字符串S和正整数p，如果有`S[i]=S[i-p]`,对于`p<i<=|S|`成立，则称p为字符串S的一个周期，特殊的`p=|S|`一定是S的周期

循环节

若S的周期满足$p \mid |S|$,周期整除总长度，则为循环节.

重要性质

> 1.p是S的周期 等价于 |S| -p是S的一个`Border`，但是border不具有二分性
>
> 2.S的Border的Border也是S的Border
>
> 3.p,q为S的周期，则gcd(p,q)也为S的周期
>
> 4.一个串的Border数量为O(n)个，但是组成了O(logn)个等差数列

Next数组的求解(下标从1开始)

`next[1]=0`

前缀i的border一定是前缀i-1的border，因此可以通过next数组的border链

```
struct KMP{
    #define N 1000010
    int p[N],ne[N],s[N];
    void get_next(){
        for(int i=2,j=0;i<=n;i++){
            while(j&&p[i]!=p[j+1]) j=ne[j];
            ne[i]=p[i]==p[j+1]?++j:j;
        }
    }
    void kmp(){
        for(int i=1,j=0;i<=m;i++){
            while(j&&s[i]!=p[j+1]) j=ne[j];
            if(s[i]==p[j+1]) j++;
            if(j==n){
```

```
                printf("%d ",i-j);
            }
        }
    }
}kmp;
```

Border_Tree

```
#define N 2000010
struct Border_Tree{
    char p[N];int ne[N],n;
    vector<int> edge[N];
    int fa[N][20],dep[N],q[N],hh,tt;
    void read(){
        scanf("%s",p+1);
        n=strlen(p+1);
    }
    void get_next(){
        for(int i=2,j=0;i<=n;i++){
            while(j&&p[i]!=p[j+1]) j=ne[j];
            ne[i]=p[i]==p[j+1]?++j:j;
        }
    }
    void build(){
        get_next();
        for(int i=1;i<=n;++i){
            int j=ne[i];
            edge[i].push_back(j);
            edge[j].push_back(i);
        }
        bfs();
    }
    void bfs(){
        for(int i=1;i<=n;++i) dep[i]=-1;
        hh=0,tt=0;
        q[0]=0;dep[0]=0;
        while(hh<=tt){
            int u=q[hh++];
            for(auto v:edge[u]){
                if(dep[v]!=-1) continue;
                dep[v]=dep[u]+1;
                q[++tt]=v;
                fa[v][0]=u;
                for(int i=1;i<=19;++i){
                    fa[v][i]=fa[fa[v][i-1]][i-1];
                }
            }
        }
    }
    int lca(int a,int b){
        if(dep[a]<dep[b]) swap(a,b);
        for(int i=19;i>=0;--i){
            if(dep[fa[a][i]]>=dep[b])
                a=fa[a][i];
        }
        if(a==b) return a;
        for(int i=19;i>=0;--i){
```

```
            if(fa[a][i]!=fa[b][i])
                a=fa[a][i],b=fa[b][i];
        }
        return fa[a][0];
    }
}T;
```

exkmp

对于个长度为 n 的字符串 s。定义函数 $z[i]$ 表示 s 和 $s[i,n-1]$（即以 $s[i]$ 开头的后缀）的最长公共前缀（LCP）的长度。z 被称为 s 的 **Z 函数**。特别地，$z[0]=0$。

```
//约定：字符串下标以0为起点。

//定义z[i]表示s和s[i,n-1]的最长公共前缀的长度


// C++ Version
vector<int> z_function(string s) {
  int n = (int)s.length();
  vector<int> z(n);
  for (int i = 1, l = 0, r = 0; i < n; ++i) {
    if (i <= r && z[i - l] < r - i + 1) {
      z[i] = z[i - l];
    } else {
      z[i] = max(0, r - i + 1);
      while (i + z[i] < n && s[z[i]] == s[i + z[i]]) ++z[i];
    }
    if (i + z[i] - 1 > r) l = i, r = i + z[i] - 1;
  }
  return z;
}


//应用：
//字符串整周期：
/*
    给定一个长度为n的字符串s，找到其最短的整周期，
即寻找一个最短的字符串t使得s可以被t拼接而成。
    考虑计算s的z函数，其整周期的长度为最小的n的因数i满足i+z[i]=n;
*/
```

ACAM

```
//AC自动机种fail[u]指向的节点v为u的最长后缀,但是可能下一个节点并不存在ch的后继，此时会直接指向原点
//但是我们可以通过补全成Trie图,来直接获得trans[u][ch],即u后面+ch字符能够转移到的最长后缀二点位置
#define N 200010
```

```
struct ACAM{ //trans[i][j]为Trie图 tr[u][ch]表示在u节点后面添加ch节点会转移到哪个节点
    string s[N];
    int q[N*30],hh,tt,idx,n;
    int tr[N*30][26],dep[N*30],cnt[N*30],pos[N*30],fail[N*30],nxt[N*30],trans[N*30][26];
    int ans[N*30];
    void insert(string &s,int id){
        int p=0;
        for(auto ch:s){
            int u=ch-'a';
            if(!tr[p][u]) tr[p][u]=++idx;
            dep[tr[p][u]]=dep[p]+1;
            p=tr[p][u];
        }
        pos[id]=p, cnt[p]++;
    }
    void get_fail(){
        hh=0,tt=-1;
        for(int i=0;i<26;++i)
            if(tr[0][i]) q[++tt]=tr[0][i];
        while(hh<=tt){
            int p=q[hh++];
            for(int i=0;i<26;++i){
                if(tr[p][i]){
                    fail[tr[p][i]]=tr[fail[p]][i];
                    q[++tt]=tr[p][i];
                }
                else tr[p][i]=tr[fail[p]][i];
                nxt[tr[p][i]]=cnt[fail[tr[p][i]]] ? fail[tr[p][i]] : nxt[fail[tr[p][i]]];
                trans[p][i]=tr[p][i]?tr[p][i]:trans[fail[p]][i];
            }
        }
    }
    void build(){
        cin >> n;
        rep(i,1,n) cin>>s[i],insert(s[i], i);
        get_fail();
    }
    void match(string &t){
        for(int i=0,p=0;i<t.size();++i){
            p=tr[p][t[i]-'a'];
            ans[p]++;
        }
        for(int i=idx-1;i>=0;--i) ans[fail[q[i]]]+=ans[q[i]];
    }
    void clear(){
        for(int i=0;i<=idx;++i){
            for(int j=0;j<26;++j){
                trans[i][j]=nxt[tr[i][j]]=ans[tr[i][j]]=pos[tr[i][j]]=dep[tr[i][j]]=cnt[tr[i]
[j]]=fail[tr[i][j]]=0;
                tr[i][j]=0;
            }
        }
        idx=0;
    }
}AC;
//============================================================
//动态开点
#define N 200010
```

```
struct ACAM{
    struct Node{
        int fail,ans;
        int ch[26];
        Node(int _fail=0,int _ans=0):
            fail(_fail),ans(_ans){memset(ch,0,sizeof ch);};
    };
    string s[N];
    vector<Node> tr;
    int n,idx,q[N*30],hh,tt,pos[N*30];

    void insert(string &s,int num){
        if(!tr.size()) tr.push_back(Node(0,0));
        int p=0;
        for(auto ch:s){
            int u=ch-'a';
            if(tr[p].ch[u]==0){
                tr[p].ch[u]=++idx;
                tr.push_back(Node(0,0));
            }
            p=tr[p].ch[u];
        }
        pos[num]=p;
    }
    void get_fail(){
        hh=0,tt=-1;
        for(int i=0;i<26;++i)
            if(tr[0].ch[i]) q[++tt]=tr[0].ch[i];
        while(hh<=tt){
            int u=q[hh++];
            for(int i=0;i<26;++i){
                if(tr[u].ch[i]){
                    tr[tr[u].ch[i]].fail=tr[tr[u].fail].ch[i];
                    q[++tt]=tr[u].ch[i];
                }
                else tr[u].ch[i]=tr[tr[u].fail].ch[i];
            }
        }
    }
    void build(){
        cin >> n;
        for(int i=1;i<=n;++i) cin>>s[i],insert(s[i],i);
        get_fail();
    }
    void match(string &t){
        int p=0;
        for(auto ch:t){
            p=tr[p].ch[ch-'a'];
            tr[p].ans++;
        }
        for(int i=idx-1;i>=0;--i) tr[tr[q[i]].fail].ans+=tr[q[i]].ans;
    }
    void clear(){
        for(int i=1;i<=n;++i) pos[i]=0;
        tr.clear(); idx=n=0;
    }
    void print_ans(){
        for(int i=1;i<=n;++i)
```

```
            print(tr[pos[i]].ans);
    }
}AC;
//============================AC自动机板子
struct ACAM{
    string s[N];
    int q[N*30],hh,tt,idx,n;
    int tr[N*30][26],dep[N*30],fail[N*30];
    int ans[N*30],pos[N*30],cnt[N*30],din[N*30];
    void insert(string &s,int id){
        int p=0;
        for(auto ch:s){
            int u=ch-'a';
            if(!tr[p][u]) tr[p][u]=++idx;
            dep[tr[p][u]]=dep[p]+1;
            p=tr[p][u];
        }
        pos[id]=p,cnt[p]++;
    }
    void get_fail(){
        hh=0,tt=-1;
        for(int i=0;i<26;++i)
            if(tr[0][i]) q[++tt]=tr[0][i];
        while(hh<=tt){
            int p=q[hh++];
            for(int i=0;i<26;++i){
                if(tr[p][i]){
                    fail[tr[p][i]]=tr[fail[p]][i];
                    din[tr[fail[p]][i]] ++;
                    q[++tt]=tr[p][i];
                }
                else tr[p][i]=tr[fail[p]][i];
            }
        }
    }
    void build(){
        cin >> n;
        rep(i,1,n) cin >> s[i], insert(s[i], i);
        get_fail();
    }
    void match(string &t){
        int p=0;
        for(auto ch:t){
            int u=ch-'a';
            p=tr[p][u];
            ans[p] ++;
        }
    }
    void get_ans(){
        queue<int> q;
        for(int i=0;i<=idx;++i){
            if(!din[i])
                q.push(i);
        }

        while(q.size()){
            auto u=q.front();
            q.pop();
```

```
            ans[fail[u]] += ans[u];
            din[fail[u]] --;
            if(din[fail[u]]==0) q.push(fail[u]);
        }
    }
    void clear(){
        for(int i=0;i<=idx;++i){
            for(int j=0;j<26;++j){
                din[tr[i][j]]=ans[tr[i][j]]=pos[tr[i][j]]=0;
                dep[tr[i][j]]=cnt[tr[i][j]]=fail[tr[i][j]]=0;
                tr[i][j]=0;
            }
        }
        idx=0;
    }
}T;
```

SA

SA数组：

sa[i]表示排第i的后缀id（后缀id指后缀开始的位置）。

Rank数组：

rk[i]表示从i开始的后缀的rank

rk和sa具有特点：sa[rk[i]]=i; rk[sa[i]]=i;

Height数组：

广义上的height[i] (ht数组)表示的是第i个后缀s[i...n]和排名在他前面的后缀的LCP

`height[i]=LCP(S[i,n], S[sa[rk[i]-1], n])`

结论：$height[i] \geq height[i-1]-1$

实际在使用的时候我们使用`H[i]=height[sa[i]]`表示排名为i的后缀和排名为i-1的后缀的LCP

那么`H[i]=LCP(S[sa[i], n], S[sa[i]-1, n])`;

而实际上，我们板子中处理出来的**height[i]就相当于H[i]**数组，height[i]表示排名为i的后缀和排名为i-1的后缀的LCP，height[0]=0.

应用

1.求本质不同子串

$\sum n - sa[i] + 1 - H[i]$

$n - sa[i] + 1$表示排名为i的后缀的长度

2.给出两个字符串S和T，求有多少长度大于K的公共子串

做法：用特殊字符连接两个串，进行后缀排序，问题转化为对于所有属于S的后缀1和属于T的后缀r组成的区间`[l,r]`,求`max(0, min(H[l...r])-K)`的和

3.给出两个串S和T，求最长公共子串的长度

用特殊字符连接两个串，进行后缀排序,得到H数组。求每一个T的后缀和所有S后缀的LCP，最大值为答案。

> 枚举每个属于T的后缀，向左向右找到第一个属于S的后缀S_l,S_r
>
> $ans = max\{LCP(T, S_l), LCP(T, S_r)\}$

4.给出两个S和T，求有多少个本质不同公共子串

```
/*懵哥的SA
//懵哥的板子中sa[i]表示的id从1开始
#include <bits/stdc++.h>
using namespace std;
const int N = 1000010;
class SA
{
    vector<int> rk, sa, cnt, height, oldrk, px, id;
    int n;
    bool cmp(int x, int y, int w)
    {
        return oldrk[x] == oldrk[y] && oldrk[x + w] == oldrk[y + w];
    }

public:
    SA() = default;
    SA(string s)
    {
        int n = s.length(), m = 300;
        this->n = n;
        oldrk.resize(max(m + 1, 2 * n + 1));
        sa.resize(max(m + 1, n + 1));
        rk.resize(max(m + 1, n + 1));
        cnt.resize(max(m + 1, n + 1));
        height.resize(max(m + 1, n + 1));
        px.resize(max(m + 1, n + 1));
        id.resize(max(m + 1, n + 1));
        s = " " + s;
        for (int i = 1; i <= n; ++i)
            ++cnt[rk[i] = s[i]];
        for (int i = 1; i <= m; ++i)
            cnt[i] += cnt[i - 1];
        for (int i = n; i >= 1; --i)
            sa[cnt[rk[i]]--] = i;
```

```
        for (int w = 1, p;; w <<= 1, m = p)
        {
            p = 0;
            for (int i = n; i > n - w; --i)
                id[++p] = i;
            for (int i = 1; i <= n; ++i)
                if (sa[i] > w)
                    id[++p] = sa[i] - w;
            fill(cnt.begin(), cnt.end(), 0);
            for (int i = 1; i <= n; ++i)
                ++cnt[px[i] = rk[id[i]]];
            for (int i = 1; i <= m; ++i)
                cnt[i] += cnt[i - 1];
            for (int i = n; i >= 1; --i)
                sa[cnt[px[i]]--] = id[i];
            copy(rk.begin(), rk.end(), oldrk.begin());
            p = 0;
            for (int i = 1; i <= n; ++i)
                rk[sa[i]] = cmp(sa[i], sa[i - 1], w) ? p : ++p;
            if (p == n)
            {
                for (int i = 1; i <= n; ++i)
                    sa[rk[i]] = i;
                break;
            }
        }
        for (int i = 1, k = 0; i <= n; ++i)
        {
            if (rk[i] == 0)
                continue;
            if (k)
                --k;
            while (s[i + k] == s[sa[rk[i] - 1] + k])
                ++k;
            height[rk[i]] = k;
        }
    }
    void print()
    {
        for (int i = 1; i <= n;i++)
            cout << sa[i] << " ";
        cout << "\n";
        for (int i = 1; i <= n;i++)
            cout << height[i] << " ";
        cout << "\n";
    }
};
int main()
{
    cin.tie(0)->sync_with_stdio(0);
    cin.exceptions(cin.failbit);
    cin.tie(NULL);
    cout.tie(NULL);
    int t;
    cin >> t;
    while(t--)
    {
        string s;
```

```
        cin >> s;
        SA t(s);
        t.print();
    }
    return 0;
}

*/
/*
int n,m;
char s[N];
int sa[N],rk[N],x[N],y[N],c[N],height[N];

void get_sa(){
    for(int i=1;i<=n;++i) c[x[i]=s[i]] ++;
    for(int i=2;i<=m;++i) c[i]+=c[i-1];
    for(int i=n;i>=1;--i) sa[c[x[i]] --]=i;
    for(int k=1;k<=n;k<<=1){
        int num=0;
        for(int i=n-k+1;i<=n;++i) y[++num]=i;
        for(int i=1;i<=n;++i)
            if(sa[i]>k)
                y[++num]=sa[i]-k;
        for(int i=1;i<=m;++i) c[i]=0;
        for(int i=1;i<=n;++i) c[x[i]] ++;
        for(int i=2;i<=m;++i) c[i]+=c[i-1];
        for(int i=n;i>=1;--i) sa[c[x[y[i]]]--]=y[i],y[i]=0;
        swap(x,y);
        x[sa[1]]=1,num=1;
        for(int i=2;i<=n;++i)
            x[sa[i]]=((y[sa[i]]==y[sa[i-1]])&&(y[sa[i]+k]==y[sa[i-1]+k]))?num:++num;  //绗 竴鍏抽
敭瀛楁湁绛変竴绗 簩鍏抽敭瀛楁湁绛?
        if(num==n) break;
        m=num;
    }
}

void get_height(){
    for(int i=1;i<=n;++i) rk[sa[i]]=i;
    for(int i=1,k=0;i<=n;++i){
        if(rk[i]==1) {height[rk[i]]=0;continue;}
        if(k) k--; //鐢变簬绉诲姩鍑 涓€涓 瓧绗︿覆?
        int j=sa[rk[i]-1];
        while(i+k<=n&&j+k<=n&&s[i+k]==s[j+k]) k++;
        height[rk[i]]=k;
    }
}

void solve(){
    scanf("%s",s+1);
    n=strlen(s+1), m=122;
    get_sa();
    get_height();
    for(int i=1;i<=n;++i) printf("%d ",sa[i]);
    printf("\n");
    for(int i=1;i<=n;++i) printf("%d ",height[i]);
}
*/
```

```
//====================================更快的板子
int sa[N],t1[N],t2[N],c[N],x[N],height[N],n,s[N],rk[N];
void get_sa(int m)
{
    int *x=t1,*y=t2;
    for(int i=0;i<m;i++) c[i]=0;
    for(int i=0;i<n;i++) c[x[i]=s[i]]++;
    for(int i=1;i<m;i++) c[i]+=c[i-1];
    for(int i=n-1;i>=0;i--) sa[--c[x[i]]]=i;
    for(int k=1;k<=n;k<<=1)
    {
        int p=0;
        for(int i=n-k;i<n;i++) y[p++]=i;
        for(int i=0;i<n;i++) if(sa[i]>=k) y[p++]=sa[i]-k;
        for(int i=0;i<m;i++) c[i]=0;
        for(int i=0;i<n;i++) c[x[y[i]]]++;
        for(int i=1;i<m;i++) c[i]+=c[i-1];
        for(int i=n-1;i>=0;i--) sa[--c[x[y[i]]]]=y[i];
        swap(x,y);
        p=1;x[sa[0]]=0;
        for(int i=1;i<n;i++)
            x[sa[i]]= y[sa[i]]==y[sa[i-1]]&&y[sa[i]+k]==y[sa[i-1]+k]?p-1:p++;
        if(p>=n) break;
        m=p;
    }
}
void get_height()
{
    for(int i=0;i<n;i++) rk[sa[i]]=i;
    for(int i=0,k=0;i<n;i++)
    {
        if(rk[i])
        {
            if(k) k--;
            int j=sa[rk[i]-1];
            while(s[i+k]==s[j+k]) k++;
            height[rk[i]]=k;
        }
    }
}

//SAIS============================================
/*
//注意此处的sa[i]表示排名为i的id，从0开始
const int N = 300000;

int n;
namespace SA {
    int sa[N], rank[N], height[N], s[N<<1], t[N<<1], p[N], cnt[N], cur[N];
    int MIN[N][30];
    #define pushS(x) sa[cur[s[x]]--] = x
    #define pushL(x) sa[cur[s[x]]++] = x
    #define inducedSort(v) fill_n(sa, n, -1); fill_n(cnt, m, 0);                \
        for (int i = 0; i < n; i++) cnt[s[i]]++;                                \
        for (int i = 1; i < m; i++) cnt[i] += cnt[i-1];                         \
        for (int i = 0; i < m; i++) cur[i] = cnt[i]-1;                          \
        for (int i = n1-1; ~i; i--) pushS(v[i]);                                \
        for (int i = 1; i < m; i++) cur[i] = cnt[i-1];                          \
```

```
    for (int i = 0; i < n; i++) if (sa[i] > 0 &&  t[sa[i]-1]) pushL(sa[i]-1); \
    for (int i = 0; i < m; i++) cur[i] = cnt[i]-1;                            \
    for (int i = n-1;  ~i; i--) if (sa[i] > 0 && !t[sa[i]-1]) pushS(sa[i]-1)
void sais(int n, int m, int *s, int *t, int *p) {
    int n1 = t[n-1] = 0, ch = rank[0] = -1, *s1 = s+n;
    for (int i = n-2; ~i; i--) t[i] = s[i] == s[i+1] ? t[i+1] : s[i] > s[i+1];
    for (int i = 1; i < n; i++) rank[i] = t[i-1] && !t[i] ? (p[n1] = i, n1++) : -1;
    inducedSort(p);
    for (int i = 0, x, y; i < n; i++) if (~(x = rank[sa[i]])) {
        if (ch < 1 || p[x+1] - p[x] != p[y+1] - p[y]) ch++;
        else for (int j = p[x], k = p[y]; j <= p[x+1]; j++, k++)
            if ((s[j]<<1|t[j]) != (s[k]<<1|t[k])) {ch++; break;}
        s1[y = x] = ch;
    }
    if (ch+1 < n1) sais(n1, ch+1, s1, t+n, p+n1);
    else for (int i = 0; i < n1; i++) sa[s1[i]] = i;
    for (int i = 0; i < n1; i++) s1[i] = p[sa[i]];
    inducedSort(s1);
}
template<typename T>
int mapCharToInt(int n, const T *str) {
    int m = *max_element(str, str+n);
    fill_n(rank, m+1, 0);
    for (int i = 0; i < n; i++) rank[str[i]] = 1;
    for (int i = 0; i < m; i++) rank[i+1] += rank[i];
    for (int i = 0; i < n; i++) s[i] = rank[str[i]] - 1;
    return rank[m];
}
template<typename T>
void suffixArray(int n, const T *str) {
    int m = mapCharToInt(++n, str);
    sais(n, m, s, t, p);
    for (int i = 0; i < n; i++) rank[sa[i]] = i;
    for (int i = 0, h = height[0] = 0; i < n-1; i++) {
        int j = sa[rank[i]-1];
        while (i+h < n && j+h < n && s[i+h] == s[j+h]) h++;
        if (height[rank[i]] = h) h--;
    }
}
void RMQ_init(){
    for(int i=0; i<n; i++) MIN[i][0] = height[i+1];
    for(int j=1; (1<<j)<=n; j++){
        for(int i=0; i+(1<<j)<=n; i++){
            MIN[i][j] = min(MIN[i][j-1], MIN[i+(1<<(j-1))][j-1]);
        }
    }
}
int RMQ(int L, int R){
    int k = 0;
    while((1<<(k+1)) <= R-L+1) k++;
    return min(MIN[L][k], MIN[R-(1<<k)+1][k]);
}
int LCP(int i, int j){
     if(rank[i] > rank[j]) swap(i, j);
    return RMQ(rank[i], rank[j]-1);
}
void init(char *str, int _n){
    n = _n;
```

```
        suffixArray(n, str);
        RMQ_init();
    }
    void solve(int len) {
        int ans = 0;
        for(int i = 2; i <= n; i++) {
            int u = sa[i-1], v = sa[i];
            if(u > v) swap(u, v);
            if(u <= len && v > len)
                ans = max(ans, height[i]);
        }
        printf("%d\n", ans);
    }
};

int t;
char str[220000], str2[110000];

int main() {
    scanf("%s", str);
    strcpy(str2, str);
    int len1 = strlen(str);
    int len2 = len1;
    reverse(str2, str2+len2);
    str[len1] = '@';
    str[len1+1] = '\0';
    len1++;
    strcat(str, str2);
    SA::init(str, len1+len2);
    SA::solve(len1-2);
    return 0;
}

*/
```

Manacher

```
char a[N],b[N];
int p[N];
int idx,n,m;

void init(){
    b[idx++]='$';
    b[idx ++]='#';
    for(int i=0;a[i];++i) b[idx++]=a[i],b[idx++]='#';
    b[idx++]='^';
    n=idx;
}

void manacher(){
    int mr=0,mid;//之前扫过的半径最大的位置+1
    for(int i=0;i<n;++i){
        if(i<mr) p[i]=min(p[2*mid-i], mr-i);
        else p[i]=1;
        while(b[i-p[i]] == b[i+p[i]]) p[i]++;
```

```
        if(i+p[i]>mr){
            mr=i+p[i];
            mid=i;
        }
    }
}
```

PAM

```
/*=====================================
struct PAM{
    //sz分配指针/tot字符串长度/last指针
    int sz,tot,last;
    //cnt[]出现次数,len[]回文串长度
    int cnt[N],tr[N][26],len[N],fail[N];
    char s[N];
    //获取一个新节点，长度为len
    int get_node(int x){
        sz++;
        memset(tr[sz],0,sizeof tr[sz]);
        len[sz]=x;
        fail[sz]=cnt[sz]=0;
        return sz;
    }
    void init(){
        sz=-1,last=0;
        s[tot=0]='$';
        get_node(0),get_node(-1);
        fail[0]=1,fail[1]=0; //奇根偶根互指
    }
    int get_fail(int u){ //找到最长回文后缀代表节点
        while(s[tot-len[u]-1]!=s[tot])
            u=fail[u];
        return u;
    }
    void add(char ch){
        s[++tot]=ch;
        int u=ch-'a';
        int now=get_fail(last);
        if(!tr[now][u]){
            int x=get_node(len[now]+2);
            fail[x]=tr[get_fail(fail[now])][u];
            tr[now][u]=x;
        }
        last=tr[now][u];
        cnt[last]++;
    }
};
==========================================*/
#include<bits/stdc++.h>
using namespace std;
```

```
const int N = 5e5 + 10;
char s[N];
int n, lst, len[N];
int now, tot = 1, fail[N], cnt[N], t[N][26];
//tot 记得赋成 1，已经有两个根节点了

int getfail(int u, int p){//求 u 的fail指针 ，p表示当前插入的是哪个点
    while(p - len[u] - 1 <= 0 || s[p - len[u] - 1] != s[p]) u = fail[u];
    //如果当前位置比 u的回文串短或者字符不相等                    就一直跳
    return u;//最后不跳了，你就是答案了
}

int insert(char c, int id){
    int p = getfail(now, id);//找到那条路中有 id 位都一样的最长回文子串
    //这个点就是他爸爸
    if(!t[p][c - 'a']){//爸爸没它这个儿子，新认一个
        fail[++tot] = t[getfail(fail[p], id)][c - 'a'];
        //fail等于((((爸爸的fail那条路上)有 id 位都是一样的回文串)的节点)的同名儿子)
        t[p][c - 'a'] = tot;//跟trie树一样
        len[tot] = len[p] + 2;//长度等于爸爸+2
        cnt[tot] = cnt[fail[tot]] + 1;//回文串数量等于fail的数量加上它自己
    }
    return cnt[now = t[p][c - 'a']];//更新它现在的位置，并返回cnt作为答案
}

int main(){
    scanf("%s", s + 1);
    n = strlen(s + 1);
    fail[0] = 1, len[1] = -1;
    //len[i] 表示节点 i 所对应的回文串长度
    for(int i=1;i<=n;i++){
        if(i > 1)   s[i] = (s[i] - 'a' + lst) % 26 + 'a';
        printf("%d ", lst = insert(s[i], i));
    }
    return 0;
}

/*
namespace PAM {
    int size, tot, last;
    int cnt[N], tr[N][26], len[N], fail[N];
    char s[N];

    int node(int l) {  // 建立一个新节点，长度为 l
        size++;
        memset(tr[size], 0, sizeof(tr[size]));
        len[size] = l;
        fail[size] = cnt[size] = 0;
        return size;
    }

    void init() {  // 初始化
        size = -1; last = 0;
        s[tot = 0] = '$';
        node(0); node(-1);
        fail[0] = 1;
    }
```

```
    int getfail(int x) {  // 找后缀回文
        while (s[tot - len[x] - 1] != s[tot]) x = fail[x];
        return x;
    }

    void add(char c) {  // 建树
        s[++tot] = c;
        int now = getfail(last);
        if (!tr[now][c]) {
            int x = node(len[now] + 2);
            fail[x] = tr[getfail(fail[now])][c];
            tr[now][c] = x;
        }
        last = tr[now][c];
        cnt[last]++;
    }
}
*/
```

SAM

```
struct SAM{
    int sz,lst;
    vector<int> len,link;
    vector<vector<int>> trans;
    //strlen瀛楃 涓查暱搴 chsize瀛楃 闆嗗ぇ灏
    SAM(int strlen,int chsize):sz(1),lst(1){
        len.resize(0);
        link.resize(0);
        trans.resize(strlen, vector<int> (chsize, 0));
    }

    void extend(int c){
        int p, cur=++sz;
        len[cur]=len[lst]+1;
        for(p=lst;p&&!trans[p][c];p=link[p])
            trans[p][c]=cur;
        if(!p) link[cur]=1;
        else{
            int q=trans[p][c];
            if(len[q]==len[p]+1) link[cur]=q;
            else{
                int cl=++sz;
                len[cl]=len[p]+1;
                trans[cl]=trans[q], link[cl]=link[q];
                while(p&&trans[p][c]==q){
                    trans[p][c]=cl;
                    p=link[p];
                }
                link[cur]=link[q]=cl;
            }
        }
        lst=cur;
```

```
    }
};
```

最小表示法

字符串的最小表示法：

给定一个字符序列S会循环产生n个字符串，这n个字符串成为循环同构，其中字典序最小的被称为S的最小表示。

求解最小表示o(n)

> 1.初始化i=1,j=2
>
> 2.通过向后扫描的方法，判断分别以i和j开头的字符串的大小关系。
>
> 3.
>
> (1) while(s[i+k]==s[j+k]) k++;
>
> (2) while(s[i+k]<s[j+k]) j=j+k+1;
>
> (3) if(i>n || j>n) return min(i,j);

```
int get_min(char *s){
    int i=1,j=2,k=0;
    while(i<=n&&j<=n){
        for(k=0;k<=n&&s[i+k]==s[j+k];k++) ;//k++;
        if(k==n) break;
        if(s[i+k]>s[j+k]){
            i=i+k+1;
            if(i==j) i++;
        }
        else {
            j=j+k+1;
            if(i==j) j++;
        }
    }
    int pos=min(i,j);
    s[pos+n]='\0';
    return pos;
}
```