A POJ - 1509（最小表示法）

模板题 直接套

```
#include<bits/stdc++.h>
using namespace std;
char str[1000005];
int mininum(string str) {
    int n = str.length();
    int k = 0, i = 0, j = 1;
    while (k < n && i < n && j < n) {
        if (str[(i + k) % n] == str[(j + k) % n]) {
            k++;
        } else {
            str[(i + k) % n] > str[(j + k) % n] ? i = i + k + 1 : j = j + k + 1;
            if (i == j) i++;
            k = 0;
        }
    }
    i = min(i, j);
    return i;
}
int main() {
    int T;
    read(T);
    while (T--) {
        cin >> str;
        cout << mininum(str) +1 << endl;
    }

}
```

B HDU - 1711 （KMP）

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<stack>
#include<cstdio>
#include<string>
#include<cstring>
#include<vector>
// #include<map>
#include<cmath>
#include<list>
#define ll long long
#define N 1000005
#define pb push_back
#define Sca(a) scanf("%d",&a)
#define mem0(a) memset(a,0,sizeof(a))
#define mem1(a) memset(a,-1,sizeof(a))
#define Scal(a) scanf("%ld",&a)
#define Scall(a) scanf("%lld",&a)
```

```
#define Pri(a) printf("%d",a)
#define Pril(a) printf("%ld",a)
#define Prill(a) printf("%lld",a)
#define FAST_IO ios::sync_with_stdio(false)
using namespace std;
const int INF = 0x3f3f3f3f;
const ll INFL = 0x3f3f3f3f3f3f3f3f;
using namespace std;
template <class T>void tomax(T&a, T b) {
    a = max(a, b);
}
template <class T>void tomin(T&a, T b) {
    a = min(a, b);
}

int s[N];
int p[N];
int Next[N];
int m,n;
void getNext(){
    int j=0,k=-1;
    Next[0]=-1;
    while(j<m){
        if(k==-1||p[j]==p[k]){
            j++;
            k++;
            Next[j]=k;
        }
        else
            k=Next[k];
    }
}
int kmp(){
    int i=0,j=0;
    getNext();
    while(i<n){
        if(j==-1||s[i]==p[j]){
            i++;
            j++;
        }
        else
            j=Next[j];
        if(j==m)
            return i;
    }
    return -1;
}
int a[10000005],b[10005];
int main(){
    int T;
    cin>>T;
    while(T--){
        mem0(Next);
        Sca(n);Sca(m);
        for(int i=0;i<n;i++) Sca(s[i]);
        for(int i=0;i<m;i++) Sca(p[i]);
        getNext();
        int flag=kmp();
        if(flag==-1) cout<<"-1"<<endl;
        else cout<<flag-m+1<<endl;
    }
}
```

C HDU - 1686 (KMP)

```
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <cmath>
#include <ctime>
#include <iostream>
#include <algorithm>
#include <string>
#include <vector>
#include <deque>
#include <list>
#include <set>
#include <map>
#include <stack>
#include <queue>
#include <numeric>
#include <iomanip>
#include <bitset>
#include <sstream>
#include <fstream>
#define pi (acos(-1.0))
#define eps (1e-8)
#define inf (1<<30)
typedef long long ll;
typedef long double ld;
using namespace std;
inline void read(ll &ans) {
    ll x = 0, f = 1;
    char ch = getchar();
    while (ch < '0' || ch > '9') {
        if (ch == '-')
            f = -1;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9') {
        x = (x << 1) + (x << 3) + (ch ^ 48);
        ch = getchar();
    }
    ans = x * f;
}
inline void read(int &ans) {
    int x = 0, f = 1;
    char ch = getchar();
    while (ch < '0' || ch > '9') {
        if (ch == '-')
            f = -1;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9') {
        x = (x << 1) + (x << 3) + (ch ^ 48);
        ch = getchar();
    }
    ans = x * f;
}
int nxt[100005];
vector<int> v;
string a,b;
int ans=0;
inline void KMP(string str, string p) //KMP
{
    int t1=0,t2=0,len1=str.length(),len2=p.length();//从0位开始匹配
    while(t1<len1) //临界值
    {
        if(t2==-1 || str[t1]==p[t2]) //匹配成功，继续
            t1++,t2++;
        else t2=nxt[t2]; //失配
        if(t2==len2) ans++,t2=nxt[t2];//t2==lenn2时，匹配成功；t1-len2+1即为第一个字
母的位置
    } //匹配成功后，t2置为next[t2]
}
inline void Getnext(string str) { //求出next数组//next数组是从 S[0到i-1]前子串 的前缀
后缀最大值
    int t1 = 0, t2, len2 = str.length();
    nxt[0] = t2 = -1;
    while (t1 < len2){
        if (t2 == -1 || str[t1] == str[t2]) //类似于KMP的匹配
            nxt[++t1] = ++t2;
        else t2 = nxt[t2]; //失配
    }

}

int main() {
    int T,n,m;
    read(T);

    while(T--){
        cin>>b>>a;
        // memset(nxt,0,sizeof(nxt));
        Getnext(b);
        KMP(a, b);
        cout<<ans<<endl;
        ans=0;
    }
}
```

D HDU - 2087 (KMP)

```
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <cmath>
#include <ctime>
#include <iostream>
#include <algorithm>
#include <string>
```

```
#include <vector>
#include <deque>
#include <list>
#include <set>
#include <map>
#include <stack>
#include <queue>
#include <numeric>
#include <iomanip>
#include <bitset>
#include <sstream>
#include <fstream>
#define pi (acos(-1.0))
#define eps (1e-8)
#define inf (1<<30)
typedef long long ll;
typedef long double ld;
using namespace std;
inline void read(ll &ans) {
    ll x = 0, f = 1;
    char ch = getchar();
    while (ch < '0' || ch > '9') {
        if (ch == '-')
            f = -1;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9') {
        x = (x << 1) + (x << 3) + (ch ^ 48);
        ch = getchar();
    }
    ans = x * f;
}
inline void read(int &ans) {
    int x = 0, f = 1;
    char ch = getchar();
    while (ch < '0' || ch > '9') {
        if (ch == '-')
            f = -1;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9') {
        x = (x << 1) + (x << 3) + (ch ^ 48);
        ch = getchar();
    }
    ans = x * f;
}
int nxt[100005];
vector<int> v;
string a,b;
int ans=0;
inline void KMP(string str, string p) //KMP
{
    int t1=0,t2=0,len1=str.length(),len2=p.length();//从0位开始匹配
    while(t1<len1) //临界值
    {
        if(t2==-1 || str[t1]==p[t2]) //匹配成功，继续
            t1++,t2++;
        else t2=nxt[t2]; //失配
        if(t2==len2) ans++,t2=0;//t2==lenn2时，匹配成功；t1-len2+1即为第一个字母的位置
    } //匹配成功后，t2置为next[t2]
}
inline void Getnext(string str) { //求出next数组//next数组是从 S[0到i-1]前子串 的前缀后缀最大值
    int t1 = 0, t2, len2 = str.length();
    nxt[0] = t2 = -1;
    while (t1 < len2){
        if (t2 == -1 || str[t1] == str[t2]) //类似于KMP的匹配
            nxt[++t1] = ++t2;
        else t2 = nxt[t2]; //失配
    }

}

int main() {
    int T,n,m;

    while(cin>>a && !(a[0]=='#'&& a.size()==1)){
        cin>>b;

        // memset(nxt,0,sizeof(nxt));
        Getnext(b);
        KMP(a, b);
        cout<<ans<<endl;
        ans=0;
    }
}
```

E HDU - 3746 (KMP)

```
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <cmath>
#include <ctime>
#include <iostream>
#include <algorithm>
#include <string>
#include <vector>
#include <deque>
#include <list>
#include <set>
#include <map>
#include <stack>
#include <queue>
#include <numeric>
#include <iomanip>
#include <bitset>
#include <sstream>
#include <fstream>
#define pi (acos(-1.0))
#define eps (1e-8)
#define inf (1<<30)
typedef long long ll;
typedef long double ld;
```

```
using namespace std;
inline void read(ll &ans) {
    ll x = 0, f = 1;
    char ch = getchar();
    while (ch < '0' || ch > '9') {
        if (ch == '-')
            f = -1;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9') {
        x = (x << 1) + (x << 3) + (ch ^ 48);
        ch = getchar();
    }
    ans = x * f;
}
inline void read(int &ans) {
    int x = 0, f = 1;
    char ch = getchar();
    while (ch < '0' || ch > '9') {
        if (ch == '-')
            f = -1;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9') {
        x = (x << 1) + (x << 3) + (ch ^ 48);
        ch = getchar();
    }
    ans = x * f;
}
int nxt[100005];
vector<int> v;
string a, b;
int ans = 0;
inline void KMP(string str, string p) { //KMP
    int t1 = 0, t2 = 0, len1 = str.length(), len2 = p.length(); //从0位开始匹配
    while (t1 < len1) { //临界值
        if (t2 == -1 || str[t1] == p[t2]) //匹配成功，继续
            t1++, t2++;
        else t2 = nxt[t2]; //失配
        if (t2 == len2) ans++, t2 = 0; //t2==lenn2时，匹配成功；t1-len2+1即为第一个
字母的位置
    } //匹配成功后，t2置为next[t2]
}
inline void Getnext(string str) { //求出next数组//next数组是从 S[0到i-1]前子串 的前缀
后缀最大值
    int t1 = 0, t2, len2 = str.length();
    nxt[0] = t2 = -1;
    while (t1 < len2) {
        if (t2 == -1 || str[t1] == str[t2]) //类似于KMP的匹配
            nxt[++t1] = ++t2;
        else t2 = nxt[t2]; //失配
    }

}

int main() {
    int T, n, m;
    read(T);
    while (T--) {
        memset(nxt, 0, sizeof(nxt));
        cin >> a;
        Getnext(a);
        int len = a.length();
        int sum = len - nxt[len];
        if (!nxt[len]) printf("%d\n", len );
        else if (len % sum == 0) printf("0\n");
        else printf("%d\n", sum - len % sum );
    }
}
```

F 计蒜客 - 38232 (序列自动机)

```
#include<bits/stdc++.h>
using namespace std;
# define ll long long
# define inf 0x3f3f3f3f
const int maxn = 2e5 + 100;
const int mod = 1e9 + 7;
char str[maxn];
int now[30];
int nex[maxn][30];
void init() {
    memset(now, -1, sizeof(now));
    int len = strlen(str);
    for (int i = len - 1; i >= 0; i--) {
        for (int j = 0; j < 26; j++) {
            nex[i][j] = now[j];
        }
        now[str[i] - 'a'] = i;
    }
}
int main() {
    scanf("%s", str);
    init();
    int m;
    scanf("%d", &m);
    while (m--) {
        scanf("%s", str);
        int len = strlen(str);
        int pos = now[str[0] - 'a'];
        int flag = 1;
        if (pos == -1)
            flag = 0;
        for (int i = 1; i < len; i++) {
            pos = nex[pos][str[i] - 'a'];
            if (pos == -1) {
                flag = 0;
                break;
            }
        }
        if (!flag)
            printf("NO\n");
        else
            printf("YES\n");
    }
    return 0;
}
```

G CF-1183E/H

我们定义dp[i][len]为以第i个字母为子串最后一个字符，长度为len的子串有多少个。

每一个字母由上一个字母转移得到。

如：如果第i个字符为'y',那么 dp[i][len] += dp[前i-1个字符中最后的'a'-'z'] [len-1];

递归边界是dp[i][1] = 1;

```
#include<iostream>
#include<cstdio>
#include<queue>
#include<cstring>
#include<unordered_map>
using namespace std;

int last[105][30];
long long dp[105][105];
char str[105];

int main() {
    int n;
    long long k;
    scanf("%d%lld", &n, &k);
    scanf("%s", str + 1);

    memset(last, -1, sizeof(last));
    for (int i = 1; i <= n; i++) {
        for (int j = 0; j < 26; j++) {
            last[i][j] = last[i - 1][j];
        }
        last[i][(int)(str[i] - 'a')] = i;
    }

    memset(dp, 0, sizeof(dp));
    for (int i = 1; i <= n; i++)
        dp[i][1] = 1;

    for (int len = 2; len < n; len++) {
        for (int i = 2; i <= n; i++) {
            for (int j = 0; j < 26; j++) {
                if (last[i - 1][j] != -1)
                    dp[i][len] = dp[i][len] + dp[last[i - 1][j]][len - 1];
            }
        }
    }

    long long ans = 0;
    k--;  //有一个长度为n的串是不需要代价的
    for (int len = n - 1; len > 0; len--) {
        long long cnt = 0;
        for (int i = 0; i < 26; i++) {
            if (last[n][i] != -1)
                cnt += dp[last[n][i]][len];
        }
        if (cnt > k) {
            ans += (n - len) * k;
            k = 0;
            break;
        } else {
            ans += (n - len) * cnt;
            k -= cnt;
        }
    }
    if (k == 1) {
        ans += n;
        k = 0;
    }

    if (k == 0)
        printf("%lld\n", ans);
    else
        printf("-1\n");
}
```

H [HDU - 1358]

1. *本题利用KMP算法中求Next[]数组的性质可以解决;*
2. *//即如果一个字符串为循环串时，（例如adcabcabc）那么它的next[]数组满足下面性质:*

   1、len%(len-next[len])==0;

   2、len/(len-next[len])就是循环的次数

```
#include <bits/stdc++.h>
#define pi (acos(-1.0))
#define eps (1e-8)
#define inf (1<<30)
typedef long long ll;
typedef long double ld;
using namespace std;
int nxt[1000010];
inline void Getnext(string str) { //求出next数组//next数组是从 S[0到i-1]前子串 的前缀后缀最大值
    int t1 = 0, t2, len2 = str.length();
    nxt[0] = t2 = -1;
    while (t1 < len2) {
        if (t2 == -1 || str[t1] == str[t2]) //类似于KMP的匹配
            nxt[++t1] = ++t2;
        else t2 = nxt[t2]; //失配
    }
}
string str;
int main() {
    int n, i, j, c, ca = 1;
    while (cin >> n && n) {
        cin >> str;
        Getnext(str);
        j = 0;
        printf("Test case #%d\n", ca++);
        for (i = 2; i <= n; i++) {
            if (i % (i - nxt[i]) == 0 && i / (i - nxt[i]) != 1) { //判断是否为周期
串且循环次数是否为1次；
                printf("%d %d\n", i, i / (i - nxt[i]));
            }
        }
        printf("\n");

    }
}
```