AC Automation

```
#include <cstdio>
#include <cstring>
#include <cstdlib>
#include <iostream>
#include <algorithm>
#define N 100005
#define C 26
using namespace std;
int n, cnt=1, root=1;
int go[N][C], flag[N], fail[N];
char s[N], st[N];
void init(){
    scanf("%s", st+1);
    cin>>n;
    for (int i=1;i<=n;i++){
        scanf("%s", s+1);
        int len=strlen(s+1);
        for (int j=1, u=root;j<=len;j++){
            int v=s[j]-'a';
            if (!go[u][v]) go[u][v]=++cnt;
            u=go[u][v];
        }
        flag[cnt]=len;
    }
}
int l[N];
void get_fail(){
    l[1]=root;
    for (int j=0;j<C;j++) go[0][j]=root;
    for (int le=1, ri=1;le<=ri;le++){
        int u=l[le];
        for (int j=0;j<C;j++)
            if (go[u][j])
                l[++ri]=go[u][j],
                fail[go[u][j]]=go[fail[u]][j];
            else go[u][j]=go[fail[u]][j];
    }
}
int top, g[N];
void solve(){
```

```
    int len=strlen(st+1);
    g[0]=root;
    int top=1;
    for (int u=root, i=1;i<=len;i++){
        g[top]=go[u][st[i]-'a'];
        s[top]=st[i];
        top-=flag[g[top]];
        u=g[top++];
    }
    s[top]='\0';
    printf("%s\n", s+1);
}
int main(){
    init();
    get_fail();
    solve();
    return 0;
}
```

SAM

```
#define C 27
#define N 200005
#define MOD 1000000007
#define pii pair<int, int>
typedef long long ll;
using namespace std;
int read(){
    int p=0, q=1;
    char ch=getchar();
    while (ch<'0' || ch>'9') (ch=='-'?q=-1:0), ch=getchar();
    while (ch>='0' && ch<='9') p=p*10+ch-'0', ch=getchar();
    return p*q;
}
int output_len;
char output[N*15];
void printi(int x){
    char s[20];
    int ct=0;
    while (x){
        s[ct++]=x%10+'0';
        x/=10;
    }
    for (int i=0;i<ct/2;++i) swap(s[i],s[ct-i-1]);
    s[ct]='\n';
    for (int i=0;i<=ct;i++) output[output_len++]=s[i];
}
void printll(ll x){
    char s[22];
    int ct=0;
    while (x){
        s[ct++]=x%10+'0';
        x/=10;
    }
    for (int i=0;i<ct/2;++i) swap(s[i],s[ct-i-1]);
    s[ct]='\n';
    for (int i=0;i<=ct;i++) output[output_len++]=s[i];
}
void add(int &a, int b){
    a+=b;
    if (a>=MOD) a-=MOD;
}
```

```
class Sam{
    public:
        int n, cnt, last;
        int maxlen[N], minlen[N];
        int link[N], trans[N][C];
        int in[N], size[N], q[N];
        int f[N];
        char st[N];
        int dp[N];
        int tot[N];
        int flag[N];
        int sg[N];
        ll g[N][C][2];
        friend void printi(int x);
        friend void add(int &a, int b);
        void clear();
        void build(char *s);
        void extend(int ch);
        ll subst_diff();
        void get_size();
        ll get_sum();
        void lcs();
        void get_sg(int p);
        void solve();
};
void Sam::clear(){
    last=cnt=1;
    memset(sg,-1,sizeof(sg));
//  memset(f,0,sizeof(f));
//  memset(q,0,sizeof(q));
//  memset(in,0,sizeof(in));
//  memset(size,0,sizeof(size));
//  memset(trans,0,sizeof(trans));
//  memset(minlen,0,sizeof(minlen));
//  memset(maxlen,0,sizeof(maxlen));
//  memset(link,0,sizeof(link));
}
void Sam::build(char *s){
    clear();
    n=strlen(s);
    for (int i=0;i<n;++i) extend(s[i]-'a');
    for (int i=1;i<=cnt;++i) minlen[i]=maxlen[link[i]]+1;
    get_size();
}
```

```
void Sam::extend(int ch){
    int cur=++cnt;
    int p=last;
    maxlen[cur]=maxlen[p]+1;
    size[cur]=1;
    for (;p && !trans[p][ch];p=link[p]) trans[p][ch]=cur;
    if (!p) link[cur]=1;
    else{
        int q=trans[p][ch];
        if (maxlen[q]==maxlen[p]+1) link[cur]=q;
        else{
            int y=++cnt;
            maxlen[y]=maxlen[p]+1;
            link[y]=link[q];
            size[y]=0;
            memcpy(trans[y], trans[q], sizeof(trans[y]));
            for (;p && trans[p][ch]==q;p=link[p]) trans[p][ch]=y;
            link[q]=link[cur]=y;
        }
    }
    last=cur;
}
void Sam::get_size(){
    int r=0;
    for (int i=1;i<=cnt;++i) in[link[i]]++;
    for (int i=1;i<=cnt;++i) if (!in[i]) q[++r]=i;
    for (int i=1;i<=r;++i){
        int u=q[i], v=link[u];
        size[v]+=size[u];
        if (!(--in[v])) q[++r]=v;
    }
}
ll Sam::subst_diff(){
    ll res=0;
    for (int i=1;i<=cnt;++i) res+=maxlen[i]-minlen[i]+1;
    return res;
}
void Sam::lcs(){
    int n=read();
    for (int i=1;i<=n;i++){
        scanf("%s", st+1);
        int len=strlen(st+1);
        int ans=0;
        for (int j=1, p=1, lcs=0;j<=2*len;j++){
```

```
            int ch=st[j>len?j-len:j]-'a';
            while (p>1 && !trans[p][ch]) p=link[p], lcs=maxlen[p];
            if (trans[p][ch]) p=trans[p][ch], ++lcs;
            else lcs=0;
            while (maxlen[link[p]]>=len) p=link[p], lcs=maxlen[p];
            if (lcs>=len && flag[p]!=i) flag[p]=i, ans+=size[p];
        }
        printf("%d\n", ans);
    }
}
void Sam::get_sg(int p){
    sg[p]=0;
    int flag[C];
    for (int j=0;j<C;j++) flag[j]=0;
    for (int j=0;j<C;j++){
        int v=trans[p][j];
        if (v){
            if (sg[v]==-1) get_sg(v);
            flag[sg[v]]=1;
            for (int k=0;k<C;k++)
                g[p][k][1]+=g[v][k][1];
        }
    }
    for (int j=0;j<C;j++)
        if (!flag[j]){
            sg[p]=j;
            break;
        }
    ++g[p][sg[p]][1];
    ll sum=0;
    for (int j=0;j<C;j++) sum+=g[p][j][1];
    for (int j=0;j<C;j++) g[p][j][0]=sum-g[p][j][1];
}
void Sam::solve(){
}
Sam sam, sam0, sam1;
char st[N], sta[N], stb[N];
ll K;
void solve(){
    int len0=0, len1=0;
    ll sum=0;
    for (int j=0;j<C;j++){
        sum+=sam0.g[1][j][1]*sam1.g[1][j][0];
        if (sum>=K) break;
```

```
    }
    if (sum<K){
        puts("NO");
        return;
    }
    int s1=1, s2=1, ts1=0, ts2=0;
    while (s1!=ts1 && K>sam1.g[1][sam0.sg[s1]][0]){
        ts1=s1;
        K-=sam1.g[1][sam0.sg[s1]][0];
        for (int j=0;j<C;j++){
            int v=sam0.trans[s1][j];
            if (v){
                ll sum=0;
               for (int k=0;k<C;k++) sum+=sam0.g[v][k][1]*sam1.g[1][k][0];
                if (sum<K) K-=sum;
                else{
                    sta[len0++]=j+'a';
                    s1=v;
                    break;
                }
            }
        }
    }
    int sg=sam0.sg[s1];
    while (s2!=ts2 && K>(sg!=sam1.sg[s2])){
        ts2=s2;
        K-=sg!=sam1.sg[s2];
        for (int j=0;j<C;j++){
            int v=sam1.trans[s2][j];
            if (v)
                if (sam1.g[v][sg][0]>=K){
                    stb[len1++]=j+'a';
                    s2=v;
                    break;
                }
                else K-=sam1.g[v][sg][0];
        }
    }
    sta[len0]='\0';
    stb[len1]='\0';
    printf("%s\n%s\n", sta, stb);
}
int main(){
    cin>>K;
```

```
    scanf("%s", st);
    sam0.build(st);
    sam0.get_sg(1);
    scanf("%s", st);
    sam1.build(st);
    sam1.get_sg(1);
    solve();
    return 0;
}
```

Suffix Arrary

```
int ls, a[3000], wv[3000], sa[3000], rk[3000], y[3000], r[3000],
h[3000];
char s[3000];
int main(){
    while (scanf("%s", s)){
        ls=strlen(s); int m=max(ls,26);
        for (int i=0;i<2*ls;i++) rk[i]=-1;
        for (int i=0;i<m;i++) wv[i]=0;
        for (int i=0;i<ls;i++) a[i]=s[i]-'a';
        for (int i=0;i<ls;i++) wv[a[i]]++;
        for (int i=1;i<m;i++) wv[i]+=wv[i-1];
        for (int i=0;i<ls;i++) sa[--wv[a[i]]]=i;
        rk[sa[0]]=0;
        for (int i=1;i<ls;i++) rk[sa[i]]=rk[sa[i-1]]+(a[sa[i]]!=a[sa[i-
1]]);
        for (int j=1;j<ls;j*=2){
            int p=0;
            for (int i=ls-j;i<ls;i++) y[++p]=i;
            for (int i=0;i<ls;i++)
                if (sa[i]>=j) y[++p]=sa[i]-j;
            for (int i=0;i<m;i++)  wv[i]=0;
            for (int i=0;i<ls;i++) wv[rk[i]]++;
            for (int i=1;i<m;i++) wv[i]+=wv[i-1];
            for (int i=ls;i;i--)   sa[--wv[rk[y[i]]]]=y[i];
            r[sa[0]]=0;
            for (int i=1;i<ls;i++)
                r[sa[i]]=r[sa[i-1]]+(rk[sa[i-1]]!=rk[sa[i]] || rk[j+sa[i-
1]]!=rk[j+sa[i]]);
            for (int i=0;i<ls;i++)
                rk[i]=r[i];
        }
        int j=0;
        for (int i=0;i<ls;i++)
        if (rk[i]<ls-1){
            for (;j+sa[rk[i]+1]<ls && j+i<ls &&
a[j+sa[rk[i]+1]]==a[i+j];++j);
            h[rk[i]]=j?j--:0;
        }
        for (int i=0;i<ls-1;i++) cout<<h[i]<<endl;
    }
}
```

ExKmp

```
const int maxn=10086;   //字符串长度最大值
int next[maxn],ex[maxn]; //ex 数组即为 extend 数组
void GETNEXT(char *str){
    int i=0,j,po,len=strlen(str);
    next[0]=len;//初始化 next[0]
    while(str[i]==str[i+1]&&i+1<len)//计算 next[1]
        i++;
    next[1]=i;
    po=1;//初始化 po 的位置
    for(i=2; i<len; i++){
        if(next[i-po]+i<next[po]+po)//第一种情况，可以直接得到 next[i]的值
            next[i]=next[i-po];
        else{//第二种情况，要继续匹配才能得到 next[i]的值
            j=next[po]+po-i;
            if(j<0)j=0;//如果 i>po+next[po],则要从头开始匹配
            while(i+j<len&&str[j]==str[j+i])//计算 next[i]
                j++;
            next[i]=j;
            po=i;//更新 po 的位置
        }
    }
}
void EXKMP(char *s1,char *s2){
    int i=0,j,po,len=strlen(s1),l2=strlen(s2);
    GETNEXT(s2);//计算子串的 next 数组
    while(s1[i]==s2[i]&&i<l2&&i<len)//计算 ex[0]
        i++;
    ex[0]=i;
    po=0;//初始化 po 的位置
    for(i=1; i<len; i++){
        if(next[i-po]+i<ex[po]+po)//第一种情况，直接可以得到 ex[i]的值
            ex[i]=next[i-po];
        else{//第二种情况，要继续匹配才能得到 ex[i]的值
            j=ex[po]+po-i;
            if(j<0)j=0;//如果 i>ex[po]+po 则要从头开始匹配
            while(i+j<len&&j<l2&&s1[j+i]==s2[j])//计算 ex[i]
                j++;
            ex[i]=j;
            po=i;//更新 po 的位置
        }
    }
```

```
}
```

Manacher

```
const int maxn=1000010;
char str[maxn];//原字符串
char tmp[maxn<<1];//转换后的字符串
int Len[maxn<<1];\
int INIT(char *st)
{
    int i,len=strlen(st);
    tmp[0]='@';//字符串开头增加一个特殊字符，防止越界
    for(i=1;i<=2*len;i+=2)
    {
        tmp[i]='#';
        tmp[i+1]=st[i/2];
    }
    tmp[2*len+1]='#';
    tmp[2*len+2]='$';//字符串结尾加一个字符，防止越界
    tmp[2*len+3]=0;
    return 2*len+1;//返回转换字符串的长度
}
int MANACHER(char *st,int len)
{
     int mx=0,ans=0,po=0;//mx 即为当前计算回文串最右边字符的最大值
     for(int i=1;i<=len;i++)
     {
         if(mx>i)
         Len[i]=min(mx-i,Len[2*po-i]);//在 Len[j]和 mx-i 中取个小
         else
         Len[i]=1;//如果 i>=mx，要从头开始匹配
         while(st[i-Len[i]]==st[i+Len[i]])
         Len[i]++;
         if(Len[i]+i>mx)//若新计算的回文串右端点位置大于 mx，要更新 po 和 mx
         {
             mx=Len[i]+i;
             po=i;
         }
         ans=max(ans,Len[i]);
     }
     return ans-1;//返回 Len[i]中的最大值-1 即为原串的最长回文子串额长度
  }
```

Polar Angle Sort

```
struct point{
    ll x, y;
    int n, q;
    int xx;
}l[N+Q], p[N+Q];
ll cross(point a, point b){
    return a.x*b.y-a.y*b.x;
}
bool cmp(point a, point b){
    if (a.xx<b.xx) return 1;
    if (a.xx>b.xx) return 0;
    return cross(a,b)>0;
}
int getxx(int x, int y){
    if (x>0 && y>=0) return 1;
    if (x<=0 && y>0) return 2;
    if (x<0 && y<=0) return 3;
    if (x>=0 && y<0) return 4;
}
ll dot (point a, point b){
    return a.x*b.x+a.y*b.y;
}
```