线性基

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
struct LinearBasis{
    int sz,cnt;
    ll bs[74],p[74];
    set<int>num;
    LinearBasis(int Sz=63){
        sz=Sz;cnt=0;
        memset(p,0,sizeof(p));
        memset(bs,0,sizeof(bs));
        num.clear();
    }
    bool ins(ll x,int sit){ // 插入
        for(ll i=sz-1,bin=(1LL<<(sz-1));i>=0;i--,bin>>=1){
            if(x&bin){
                if(!bs[i]){
                    bs[i]=x;
                    num.insert(sit);
                    break;
                }
                x^=bs[i];
            }
        }
        return x>0;
    }
    /**
        如果要验证一个数字是否属于这个线性空间的话，只需要将这个数字插入线性基，
        如果返回false，则这个数字存在，否则不存在。
    **/
    ll getmax(){ // 获取线性异或空间内的最大值
        ll res=0;
        for(ll i=sz-1;i>=0;i--){
            if((res^bs[i])>res)res^=bs[i];
        }
        return res;
    }
    ll getmin(){ // 获取线性异或空间内的最小值
        for(ll i=0;i<sz;i++){
            if(bs[i])return bs[i];
        }
        return 0;
    }
    void rebuild(){ // 重构线性基，用于查询第k小
        for(int i=sz-1;i>=0;i--)
            for(int j=i-1;j>=0;j--)
                if(bs[i]&(1LL<<j))bs[i]^=bs[j];
        for(int i=0;i<sz;i++)
        if(bs[i])p[cnt++]=bs[i];
    }
    ll getK(ll k){// 获取线性异或空间中第k小的值
```

```
        ll ret=0;
        if(k>=(1LL<<cnt))return -1;
        for(int i=sz-1;i>=0;i--)
            if(k&(1LL<<i))ret^=p[i];
        return ret;
    }
};
// 合并两个基底
LinearBasis Merge(const LinearBasis &n1,const LinearBasis &n2){
    LinearBasis ret=n1;
    for(int i=n2.sz-1;i>=0;i--)
        if(n2.bs[i])ret.ins(n2.bs[i],i);
    return ret;
}

// 线性基求交集
LinearBasis Inter(LinearBasis n1,LinearBasis n2){
    LinearBasis a=n2,b=n2;
    LinearBasis ret=LinearBasis();
    for(int i=0;i<n1.sz;i++){
        if(!n1.bs[i])continue;
        ll temp=0,x=n1.bs[i];
        int j;
        for(j=i;j>=0;j--){
            if(x&(1LL<<j)){
                if(a.bs[j]){
                    x^=a.bs[j];
                    temp^=b.bs[j];
                }
                else break;
            }
        }
        if(x==0)ret.bs[i]=temp;
        else{
            a.bs[j]=x;
            b.bs[j]=temp;
        }
    }
    return ret;
}
int main(){

    return 0;
}
```

floyd求最小环

```
/**
    求最小环
*/
ll floyd(){
    ll ans=INF;
    for(int k=1;k<=cnt;k++){
        for(int i=1;i<k;i++){
```

```
            for(int j=i+1;j<k;j++){
                ans=min(ans,roc[i][k]+roc[k][j]+mp[j][i]);
            }
        }
        for(int i=1;i<=cnt;i++){
            for(int j=1;j<=cnt;j++){
                mp[i][j]=min(mp[i][j],mp[i][k]+mp[k][j]);
            }
        }
    }
    return ans;
}
```

快速矩阵乘

```
    /**
        牛客网多校2019年第二场
        Eddy Walker 2
    **/

    #include<bits/stdc++.h>
    using namespace std;

    typedef long long LL;
    const int MAXN=1e3+10;
    const int MAXM=2060;
    const int mod=1e9+7;

    LL power(LL a,LL b){
        LL c=1;
        while(b){
            if (b%2==1){
                c=(c*a)%mod;
            }
            a=(a*a)%mod;
            b/=2;
        }
        return c;
    }

    LL n,m,invM;

    struct Matrix {
        /**
            m^2 的快速矩阵乘法，适用于线性递推方程生成的矩阵
        **/
        Matrix(){
            init();
        }
        void init(){
            for(int i=1;i<m;i++) x[i]=0;
            x[0]=1;
        }
        Matrix operator * (const Matrix &mm) const
```

```
        {
            Matrix ret;
            for (int i=0;i<=2*m-2;i++) ret.x[i]=0;
            for (int i=0;i<m;i++)
                for (int j=0;j<m;j++)
                    ret.x[i+j]=(ret.x[i+j]+(LL)x[i]*mm.x[j])%mod;
            for (int i=2*m-2;i>=m;i--){
                for (int j=1;j<=m;j++)
                    ret.x[i-j]=(ret.x[i-j]+(LL)ret.x[i]*invM)%mod;
                ret.x[i]=0;
            }
            return ret;
        }

        LL x[MAXM];
    }bas,res;

    LL f[MAXM];

    LL solve(LL n){
        if (m==1) return 1;
        if (n==-1){
            LL ret=2;
            ret*=power(m+1,mod-2);
            ret%=mod;
            return ret;
        }
        f[0]=1;
        for(int i=1;i<=2*m;i++){
            f[i]=0;
            for(int j=1;j<=m && i-j>=0;j++){
                f[i]=(f[i]+f[i-j])%mod;
            }
            f[i]=(f[i]*invM)%mod;
        }
        if (n<m) return f[n];
        bas.init(); bas.x[0]=0; bas.x[1]=1;
        res.init();
        for(n-=m-1;n;n/=2,bas=bas*bas){
            if (n%2==1){
                res=res*bas;
            }
        }
        LL ans=0;
        for (int i=0;i<m;i++)
            ans=(ans+res.x[i]*f[m+i-1]%mod)%mod;
        return ans;
    }

    //long double g[10000005];

    int main(){
        int T; scanf("%d",&T);
        while(T--){
            scanf("%lld%lld",&m,&n);
            invM=power(m,mod-2);
            LL ans=solve(n);
            printf("%lld\n",ans);
```

```
        }
        return 0;
    }
```

回文自动机

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
const int MAXN=300004;
const int D=27;
char s[300004];
int n;
/**
这里是统计回文子串的最大的出现值
出现值定义：出现次数*回文串长度
**/
struct PAM{
    int last,cnt;
    int fa[MAXN],sz[MAXN],len[MAXN],ch[MAXN][D];
    int create(int Len,int Fa){
        len[cnt]=Len;
        fa[cnt]=Fa;
        return cnt++;
    }
    void CLEAR(){
        for(int i=0;i<cnt;i++){
            fa[i]=0;sz[i]=0;len[i]=0;
            memset(ch[i],0,sizeof(ch[i]));
        }
        cnt=0;
        last=0;
        create(0,1);
        create(-1,0);
    }
    int getfail(int p,int n){
        for(;s[n-len[p]-1]!=s[n];p=fa[p]);
        return p;
    }
    int add(int c,int pos){
        int p=getfail(last,pos);
        if(!ch[p][c]){
            ch[p][c]=create(len[p]+2,ch[getfail(fa[p],pos)][c]);
        }
        last=ch[p][c];
        sz[last]++;
        return last;
    }
    ll getans(){
        ll res=0;
        for(int i=cnt-1;i>=2;i--){
            sz[fa[i]]+=sz[i];
            /**
                这里需要将每种串的出现次数统计清楚。
            **/
        }
        for(int i=cnt-1;i>=2;i--){
```

```
            res=max(res,1LL*sz[i]*len[i]);
        }
        return res;
    }
}pam;
int w33ha(){
    n=strlen(s+1);
    pam.CLEAR();
    for(int i=1;i<=n;i++){
        pam.add(s[i]-'a',i);
    }
    printf("%lld\n",pam.getans());
}
int main(){
    while(scanf("%s",s+1)!=EOF)w33ha();
    return 0;
}
```

双端回文自动机

```
/**
你有四种操作：
1.往字符串的开头加一个字符
2.往字符串的结尾加一个字符
3.询问这个字符串的子串中本质不同的回文串的数量
4.询问这个字符串的子串中回文串的总数（相同的也要计算）
刚开始字符串长度是0

双端回文自动机
由于回文自动机的插入是O(1)的，所以可以借此来拓展一下回文自动机的应用范围。使其能够支持前后同时插入字符。
需要两个pos指针和两个last指针，分别左边界和右边界，last指针表示左边插入之后的最后一个节点，和右边插入之后的最后一个节点。
每个节点记录一个ca[x]表示当前节点所表示的字符串的数量。
对于询问3，我们只需要输出节点个数-2即可。
对于询问4，我们需要在插入的时候统计所有节点的ca[x]总和
当插入当前字符串之后，总串是个回文串的时候，这次插入操作的last可以被赋值给在另一个方向插入的last
**/
#include<bits/stdc++.h>
#define ll long long
using namespace std;
const int MAXN=200014;
const int D=27;
int m,s[MAXN];
struct dePAM{
    int cnt;
    int last[2],pos[2];
    ll sum;
    int ch[MAXN][D],len[MAXN],ca[MAXN],fa[MAXN];
    int create(int Len,int Fa){
        len[cnt]=Len;
        fa[cnt]=Fa;
        return cnt++;
    }
    void CLEAR(){
        for(int i=0;i<cnt;i++){
```

```
            len[i]=0;ca[i]=0;fa[i]=0;
            memset(ch[i],0,sizeof(ch[i]));
        }
        memset(s,-1,sizeof(s));
        last[0]=0;last[1]=0;
        pos[0]=m;pos[1]=m-1;
        cnt=0;sum=0;
        create(0,1);
        create(-1,0);
    }
    int getfail(int p,int n,int op){
        int g=-1;if(!op)g=1;
        for(;s[n+g*(len[p]+1)]!=s[n];p=fa[p]);
        return p;
    }
    int add(int c,int op){
        if(!op){
            s[--pos[op]]=c;
        }
        else{
            s[++pos[op]]=c;
        }
        int p=getfail(last[op],pos[op],op);
        if(!ch[p][c]){
            int nt=create(len[p]+2,ch[getfail(fa[p],pos[op],op)][c]);
            ca[nt]=ca[fa[nt]]+1;
            ch[p][c]=nt;
        }
        last[op]=ch[p][c];
        if(pos[1]-pos[0]+1==len[last[op]])last[op^1]=last[op];
        sum+=ca[last[op]];
        return last[op];
    }
}pam;

int w33ha(){
    pam.CLEAR();
    while(m--){
        int op;
        char ch[4];
        scanf("%d",&op);
        if(op<=2){
            scanf("%s",ch+1);
            pam.add(ch[1]-'a',op-1);
        }
        else if(op==3){
            printf("%d\n",pam.cnt-2);
        }
        else{
            printf("%lld\n",pam.sum);
        }
    }
    return 0;
}

int main(){
    while(scanf("%d",&m)!=EOF)w33ha();
    return 0;
```

```
}
```

最小偶数长度回文分解

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
const int INF=1e9;
const int MAXN=1000004;
char s[MAXN];
struct PAT{
    int cnt,last;
    int ch[MAXN][26],len[MAXN],fa[MAXN],sz[MAXN];
    int slk[MAXN],diff[MAXN];
    int fp[MAXN];
    int create(int Len,int Fa){
        fa[cnt]=Fa;
        len[cnt]=Len;
        return cnt++;
    }
    void CLEAR(){
        for(int i=0;i<cnt;i++){
            fa[i]=0;len[i]=0;sz[i]=0;
            slk[i]=0;diff[i]=0;
            memset(ch[i],0,sizeof(ch[i]));
        }
        last=0;cnt=0;
        create(0,1);
        create(-1,0);
    }
    int getfail(int p,int n){
        for(;s[n-len[p]-1]!=s[n];p=fa[p]);
        return p;
    }
    int add(int c,int pos){
        int p=getfail(last,pos);
        if(!ch[p][c]){
            int nt=create(len[p]+2,ch[getfail(fa[p],pos)][c]);
            ch[p][c]=nt;
            diff[nt]=len[nt]-len[fa[nt]];
            slk[nt]=((diff[nt]==diff[fa[nt]])?slk[fa[nt]]:fa[nt]);
        }
        last=ch[p][c];
        return last;
    }

    /**
    对于字符串s，询问其最少能被表示成多少个偶数长度的回文串拼接的结果。
    dp[i]记录s[1...i]的答案，pre[i]记录dp[i]的转移前驱。
    pre[i] , i 为贡献答案的一个回文串。
    */
    void solve(int l,int *dp,int *pre){
        for(int i=0;i<=l;i++){
            pre[i]=0;
            dp[i]=INF;
        }
        CLEAR();
```

```
        dp[0]=0;
        fp[0]=1;
        for(int i=1;i<=l;i++){
            for(int j=add(s[i]-'a',i);j;j=slk[j]){
                fp[j]=i-len[slk[j]]-diff[j];
                if(diff[j]==diff[fa[j]]&&dp[fp[j]]>dp[fp[fa[j]]]){
                    fp[j]=fp[fa[j]];
                }
                if(i%2==0&&dp[i]>dp[fp[j]]+1){
                    dp[i]=dp[fp[j]]+1;
                    pre[i]=fp[j];
                }
            }
            if(i%2==0&&s[i-1]==s[i]&&dp[i]>=dp[i-2]){
                dp[i]=dp[i-2];
                pre[i]=i-2;
            }
        }
    }
}pat;

/**
下方不重要...
*/

char ss[MAXN];
int dp[MAXN],pre[MAXN];
int main(){
    int n;
    scanf("%s",ss+1);
    n=strlen(ss+1);
    n<<=1;
    for(int i=1;i<=n;i+=2)s[i]=ss[i/2+1];
    scanf("%s",ss+1);
    for(int i=2;i<=n;i+=2)s[i]=ss[i/2];
    pat.solve(n,dp,pre);
    if(dp[n]>n)return puts("-1"),0;
    else printf("%d\n",dp[n]);
    for(int i=n;i;i=pre[i]){
        if(i-pre[i]>2)printf("%d %d\n",pre[i]/2+1,i/2);
    }
    return 0;
}
```