多项式练习 1

习题一

洛谷p3702

题意

询问有多少序列满足下列条件：

1. 序列长度等于 n
2. 序列由不超过 m 的正整数构成且至少含有一个素数
3. 序列所有数之和是 p 的倍数

题解

考虑容斥，于是答案为由不超过 m 的正整数构成的序列数减去不超过 m 的合数构成的序列数。

不超过 m 且模 p 为 i 的正整数个数，而利用线性筛可以计算出不超过 m 且模 p 为 i 的合数个数。

设 $\text{dp}(i,j)$ 表示长度为 i 且和模 p 为 j 的序列数，发现可以利用倍增 $+$ NTT 转移。时间复杂度 $O(m+p\log n\log p)$

本题 $p\le 100$ 且模数为合数，所以考虑直接暴力转移，时间复杂度 $O(m+p^2\log n)$

```
const int MAXM=2e7+5,MAXP=105,Mod=20170408;
int prime[MAXM],p_cnt,p;
bool vis[MAXM];
void get_prime(int n){
    vis[1]=true;
    _rep(i,2,n){
        if(!vis[i])prime[p_cnt++]=i;
        for(int j=0;i*prime[j]<=n&&j<p_cnt;j++){
            vis[i*prime[j]]=true;
            if(i%prime[j]==0)break;
        }
    }
}
void mul(int *a,int *b){
    static int temp[MAXP];
    mem(temp,0);
    _for(i,0,p)_for(j,0,p)
    temp[(i+j)%p]=(temp[(i+j)%p]+1LL*a[i]*b[j])%Mod;
    _for(i,0,p)a[i]=temp[i];
}
```

```
int cnt1[MAXP],cnt2[MAXP],dp1[MAXP],dp2[MAXP];
int main()
{
    int n=read_int(),m=read_int();
    p=read_int();
    get_prime(m);
    _rep(i,1,m){
        cnt1[i%p]++;
        cnt2[i%p]+=vis[i];
    }
    dp1[0]=dp2[0]=1;
    for(int i=30;i>=0;i--){
        if(n&(1<<i)){
            mul(dp1,cnt1);
            mul(dp2,cnt2);
        }
        if(i){
            mul(dp1,dp1);
            mul(dp2,dp2);
        }
    }
    enter((dp1[0]-dp2[0]+Mod)%Mod);
    return 0;
}
```

习题二

洛谷p4173

题意

给定只包含小写字母和通配符的字符串 A,B□其中 A 为模板串□$|A|=m,|B|=n$□询问所有匹配位置。

题解

构成字符集到数值的映射

$$ f(x)= \begin{cases} x-a &x= a\sim z\\ 0 &x=* \end{cases} $$

令 $c_i=\sum_{j=0}^{m-1}\left(f(a_j)-f(b_{i-m+1+j})\right)^2f(a_j)f(b_{i-m+1+j})$□

于是 B 以 b_i 结尾的长度为 m 的字符串与 A 匹配当且仅当 $c_i=0$□令 $t_i=a_{m-1-i}$□有

$$\sum_{j=0}^{m-1}\left(f(t_{m-1-j})-f(b_{i-m+1+j})\right)^2f(t_{m-1-j})f(b_{i-m+1+j})=\sum_{x+y=i}\left(f(t_x)-f(b_y)\right)^2f(t_x)f(b_y)$$

于是直接 FFT 即可，时间复杂度 $O(n\log n)$□

```
const int MAXN=3e5+5;
const double pi=acos(-1.0);
struct complex{
    double x,y;
    complex(double x=0.0,double y=0.0):x(x),y(y){}
    complex operator + (const complex &b){
        return complex(x+b.x,y+b.y);
    }
    complex operator - (const complex &b){
        return complex(x-b.x,y-b.y);
    }
    complex operator * (const complex &b){
        return complex(x*b.x-y*b.y,x*b.y+y*b.x);
    }
    complex operator * (const int &b){
        return complex(x*b,y*b);
    }
}w[32][2];
int rev[MAXN<<2];
int build(int k){
    int n,pos=0;
    while((1<<pos)<=k)pos++;
    n=1<<pos;
    _for(i,0,n)rev[i]=(rev[i>>1]>>1)|((i&1)<<(pos-1));
    for(int i=1,t=0;i<n;i<<=1,t++)
    w[t][1]=complex(cos(pi/i),sin(pi/i)),w[t][0]=complex(cos(pi/i),-
sin(pi/i));
    return n;
}
void FFT(complex *f,int n,int type){
    _for(i,0,n)if(i<rev[i])
    swap(f[i],f[rev[i]]);
    complex t1,t2;
    for(int i=1,t=0;i<n;i<<=1,t++){
        for(int j=0;j<n;j+=(i<<1)){
            complex cur(1.0,0.0);
            _for(k,j,j+i){
                t1=f[k],t2=cur*f[k+i];
                f[k]=t1+t2,f[k+i]=t1-t2;
                cur=cur*w[t][type];
            }
        }
    }
    if(!type)_for(i,0,n)
    f[i].x/=n;
}
complex sum[MAXN<<2];
void Mul(int *a,int _n,int *b,int _m,int k,int n){
    static complex f[MAXN<<2],g[MAXN<<2];
    _for(i,0,_n)f[i]=complex(a[i],0.0);
```

```
    _for(i,_n,n)f[i]=complex(0.0,0.0);
    _for(i,0,_m)g[i]=complex(b[i],0.0);
    _for(i,_m,n)g[i]=complex(0.0,0.0);
    FFT(f,n,1);FFT(g,n,1);
    _for(i,0,n)f[i]=f[i]*g[i];
    _for(i,0,n)sum[i]=sum[i]+f[i]*k;
}
char s1[MAXN],s2[MAXN];
int v1[MAXN],v2[MAXN],t1[MAXN],t2[MAXN];
int main()
{
    int m=read_int(),n=read_int();
    scanf("%s%s",s1,s2);
    _for(i,0,m){
        if(s1[i]=='*')v1[i]=0;
        else
        v1[i]=s1[i]-'a'+1;
    }
    reverse(v1,v1+m);
    _for(i,0,n){
        if(s2[i]=='*')v2[i]=0;
        else
        v2[i]=s2[i]-'a'+1;
    }
    int __n=build(n+m-2);
    _for(i,0,m)t1[i]=v1[i]*v1[i]*v1[i];
    _for(i,0,n)t2[i]=v2[i];
    Mul(t1,m,t2,n,1,__n);
    _for(i,0,m)t1[i]=v1[i]*v1[i];
    _for(i,0,n)t2[i]=v2[i]*v2[i];
    Mul(t1,m,t2,n,-2,__n);
    _for(i,0,m)t1[i]=v1[i];
    _for(i,0,n)t2[i]=v2[i]*v2[i]*v2[i];
    Mul(t1,m,t2,n,1,__n);
    FFT(sum,__n,0);
    int cnt=0;
    _for(i,m-1,n){
        if(fabs(sum[i].x)<0.5)
        cnt++;
    }
    enter(cnt);
    _for(i,m-1,n){
        if(fabs(sum[i].x)<0.5)
        space(i-m+2);
    }
    return 0;
}
```