# 哈希姿势

**using** **namespace** std**;**

**typedef** long long LL**;**

const int INF **=** 0x3f3f3f3f**;**

const int N **=** 2e5 **+** 10**;**

const LL SEED **=** 2009**;**

**//13331 12323 53077**

vector**<**int**>** ve**[**N**];**

map**<**LL**,**int**>** mymap**;**

LL dfs**(**int x**)**

**{**

LL p **=** 1**;**

**for(**int i **=** 0**;** i **<** ve**[**x**].**size**();** i**++)** **{**

int y **=** ve**[**x**][**i**];**

LL tmp **=** dfs**(**y**);**

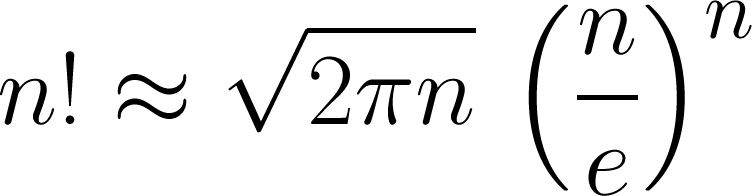
p **+=** tmp **\*** SEED**;**

**}**

mymap**[**p**]++;**

**return** p**;**

**}**



Lucas

LL Power\_mod**(**LL a**,** LL b**,** LL p**)**

**{**

LL res **=** 1**;**

**while(**b**!=**0**)**

**{**

**if(**b**&**1**)** res **=** **(**res**\***a**)%**p**;**

a **=** **(**a**\***a**)%**p**;**

b **>>=** 1**;**

**}**

**return** res**;**

**}**

LL Comb**(**LL a**,**LL b**,** LL p**)**

**{**

**if(**a **<** b**)** **return** 0**;**

**if(**a **==** b**)** **return** 1**;**

**if(**b **>** a**-**b**)** b **=** a**-**b**;**

LL ans **=** 1**,** ca **=** 1**,** cb **=** 1**;**

**for(**LL i**=**0**;** i**<**b**;** **++**i**)**

**{**

ca **=** **(**ca**\*(**a**-**i**))%**p**;**

cb **=** **(**cb**\*(**b**-**i**))%**p**;**

**}**

ans **=** **(**ca**\***Power\_mod**(**cb**,** p**-**2**,** p**))%**p**;**

**return** ans**;**

**}**

LL Lucas**(**int n**,** int m**,** int p**)**

**{**

LL ans **=** 1**;**

**while(**n **&&** m **&&** ans**)**

**{**

ans **=** **(**ans **\*** Comb**(**n**%**p**,** m**%**p**,** p**))%**p**;**

n **/=** p**;**

m **/=** p**;**

**}**

**return** ans**;**

**}**

# FFT

const double eps**(**1e-8**);**

**typedef** long long lint**;**

const double PI **=** acos**(-**1.0**);**

struct Complex

**{**

double real**,** image**;**

Complex**(**double \_real**,** double \_image**)**

**{**

real **=** \_real**;**

image **=** \_image**;**

**}**

Complex**(){}**

**};**

Complex **operator** **+** **(**const Complex **&**c1**,** const Complex **&**c2**)**

**{**

**return** Complex**(**c1**.**real **+** c2**.**real**,** c1**.**image **+** c2**.**image**);**

**}**

Complex **operator** **-** **(**const Complex **&**c1**,** const Complex **&**c2**)**

**{**

**return** Complex**(**c1**.**real **-** c2**.**real**,** c1**.**image **-** c2**.**image**);**

**}**

Complex **operator** **\*** **(**const Complex **&**c1**,** const Complex **&**c2**)**

**{**

**return** Complex**(**c1**.**real**\***c2**.**real **-** c1**.**image**\***c2**.**image**,** c1**.**real**\***c2**.**image **+** c1**.**image**\***c2**.**real**);**

**}**

int rev**(**int id**,** int len**)**

**{**

int ret **=** 0**;**

**for(**int i **=** 0**;** **(**1 **<<** i**)** **<** len**;** i**++)**

**{**

ret **<<=** 1**;**

**if(**id **&** **(**1 **<<** i**))** ret **|=** 1**;**

**}**

**return** ret**;**

**}**

Complex A**[**140000**];**

void FFT**(**Complex**\*** a**,** int len**,** int DFT**)**//对a进行DFT或者逆DFT, 结果存在a当中

**{**

//Complex\* A = new Complex[len]; 这么写会爆栈

**for(**int i **=** 0**;** i **<** len**;** i**++)**

A**[**rev**(**i**,** len**)]** **=** a**[**i**];**

**for(**int s **=** 1**;** **(**1 **<<** s**)** **<=** len**;** s**++)**

**{**

int m **=** **(**1 **<<** s**);**

Complex wm **=** Complex**(**cos**(**DFT**\***2**\***PI**/**m**),** sin**(**DFT**\***2**\***PI**/**m**));**

**for(**int k **=** 0**;** k **<** len**;** k **+=** m**)**

**{**

Complex w **=** Complex**(**1**,** 0**);**

**for(**int j **=** 0**;** j **<** **(**m **>>** 1**);** j**++)**

**{**

Complex t **=** w**\***A**[**k **+** j **+** **(**m **>>** 1**)];**

Complex u **=** A**[**k **+** j**];**

A**[**k **+** j**]** **=** u **+** t**;**

A**[**k **+** j **+** **(**m **>>** 1**)]** **=** u **-** t**;**

w **=** w**\***wm**;**

**}**

**}**

**}**

**if(**DFT **==** **-**1**)** **for(**int i **=** 0**;** i **<** len**;** i**++)** A**[**i**].**real **/=** len**,** A**[**i**].**image **/=** len**;**

**for(**int i **=** 0**;** i **<** len**;** i**++)** a**[**i**]** **=** A**[**i**];**

**return;**

**}**

char numA**[**50010**],** numB**[**50010**];**//以每一位为系数, 那么多项式长度不超过50000

Complex a**[**140000**],** b**[**140000**];**//对应的乘积的长度不会超过100000, 也就是不超过(1 << 17) = 131072

int ans**[**140000**];**

int main**()**

**{**

**while(~**scanf**(**"%s"**,** numA**))**

**{**

int lenA **=** strlen**(**numA**);**

int sa **=** 0**;**

**while((**1 **<<** sa**)** **<** lenA**)** sa**++;**

scanf**(**"%s"**,** numB**);**

int lenB **=** strlen**(**numB**);**

int sb **=** 0**;**

**while((**1 **<<** sb**)** **<** lenB**)** sb**++;**

//那么乘积多项式的次数不会超过(1 << (max(sa, sb) + 1))

int len **=** **(**1 **<<** **(**max**(**sa**,** sb**)** **+** 1**));**

**for(**int i **=** 0**;** i **<** len**;** i**++)**

**{**

**if(**i **<** lenA**)** a**[**i**]** **=** Complex**(**numA**[**lenA **-** i **-** 1**]** **-** '0'**,** 0**);**

**else** a**[**i**]** **=** Complex**(**0**,** 0**);**

**if(**i **<** lenB**)** b**[**i**]** **=** Complex**(**numB**[**lenB **-** i **-** 1**]** **-** '0'**,** 0**);**

**else** b**[**i**]** **=** Complex**(**0**,** 0**);**

**}**

FFT**(**a**,** len**,** 1**);**

FFT**(**b**,** len**,** 1**);**//把A和B换成点值表达

**for(**int i **=** 0**;** i **<** len**;** i**++)**//做点值表达的成乘法

a**[**i**]** **=** a**[**i**]\***b**[**i**];**

FFT**(**a**,** len**,** **-**1**);**//逆DFT换回原来的系数, 虚部一定是0

**for(**int i **=** 0**;** i **<** len**;** i**++)**

ans**[**i**]** **=** **(**int**)(**a**[**i**].**real **+** 0.5**);**//取整误差的处理

**for(**int i **=** 0**;** i **<** len **-** 1**;** i**++)**//进位问题

**{**

ans**[**i **+** 1**]** **+=** ans**[**i**]** **/** 10**;**

ans**[**i**]** **%=** 10**;**

**}**

bool flag **=** 0**;**

**for(**int i **=** len **-** 1**;** i **>=** 0**;** i**--)**//注意输出格式的调整即可

**{**

**if(**ans**[**i**])** printf**(**"%d"**,** ans**[**i**]),** flag **=** 1**;**

**else** **if(**flag **||** i **==** 0**)** printf**(**"0"**);**

**}**

putchar**(**'\n'**);**

**}**

**return** 0**;**

**}**

FFT**(**b**,** len**,** 1**);**//把A和B换成点值表达

**for(**int i **=** 0**;** i **<** len**;** i**++)**//做点值表达的成乘法

a**[**i**]** **=** a**[**i**]\***b**[**i**];**

FFT**(**a**,** len**,** **-**1**);**//逆DFT换回原来的系数, 虚部一定是0

**for(**int i **=** 0**;** i **<** len**;** i**++)**

ans**[**i**]** **=** **(**int**)(**a**[**i**].**real **+** 0.5**);**//取整误差的处理

**for(**int i **=** 0**;** i **<** len **-** 1**;** i**++)**//进位问题

**{**

ans**[**i **+** 1**]** **+=** ans**[**i**]** **/** 10**;**

ans**[**i**]** **%=** 10**;**

**}**

bool flag **=** 0**;**

**for(**int i **=** len **-** 1**;** i **>=** 0**;** i**--)**//注意输出格式的调整即可

**{**

**if(**ans**[**i**])** printf**(**"%d"**,** ans**[**i**]),** flag **=** 1**;**

**else** **if(**flag **||** i **==** 0**)** printf**(**"0"**);**

**}**

putchar**(**'\n'**);**

**}**

**return** 0**;**

**}**