ZKY个人模板

[ZKY个人模板 1](#_Toc4801)

[数论 2](#_Toc23058)

[线性筛 2](#_Toc31388)

[线性逆元 2](#_Toc23672)

[扩展欧几里得 2](#_Toc14248)

[求Phi 3](#_Toc570)

[快速乘 3](#_Toc13008)

[快速幂 3](#_Toc16922)

[求逆元 3](#_Toc31559)

[中国剩余定理 3](#_Toc31764)

[离散对数 4](#_Toc4196)

[卢卡斯定理 4](#_Toc31809)

[伪素性检验 4](#_Toc14196)

[Prho 4](#_Toc24043)

[扩展离散对数 5](#_Toc9474)

[不定方程 5](#_Toc6199)

[垃圾桶算法 5](#_Toc15710)

[组合数取模 5](#_Toc2308)

[MR 6](#_Toc3697)

[数值算法 7](#_Toc14330)

[FFT 7](#_Toc29985)

[NTT 8](#_Toc6876)

[高斯消元 9](#_Toc13000)

[数据结构 10](#_Toc11674)

[fhqTreap 10](#_Toc7063)

[LCT 11](#_Toc946)

[SkewHeap 13](#_Toc3291)

[Treap 14](#_Toc26083)

[点分治 15](#_Toc4721)

[HLD 17](#_Toc26181)

[左偏树 17](#_Toc7418)

[莫队 18](#_Toc26172)

[树上莫队 19](#_Toc30343)

[KDtree 21](#_Toc9439)

[图论 23](#_Toc20955)

[割点 23](#_Toc25039)

[Dinic 23](#_Toc28781)

[HK 24](#_Toc16274)

[匈牙利 26](#_Toc32212)

[费用流 26](#_Toc10730)

[SCC 27](#_Toc29511)

[KM 27](#_Toc26562)

[最大团 28](#_Toc11341)

[字符串 29](#_Toc32165)

[AC自动机 29](#_Toc24030)

[后缀数组 30](#_Toc27876)

[后缀自动机 31](#_Toc27510)

[KMP 32](#_Toc31942)

[扩展SAM 32](#_Toc12787)

[Manacher 34](#_Toc26244)

[最小表示法 34](#_Toc24117)

[IBWT 35](#_Toc16865)

[其他 35](#_Toc21971)

[平面最近点对 35](#_Toc16269)

[整体二分 36](#_Toc10851)

[最小圆覆盖(随机增量) 37](#_Toc26564)

[插头DP 38](#_Toc5163)

[线性规划 40](#_Toc1733)

[快速乘 41](#_Toc5943)

[Hints 41](#_Toc32274)

[积分表 41](#_Toc19998)

[数学公式 41](#_Toc10391)

[网络流二分图 41](#_Toc28674)

[待补 41](#_Toc12589)

# 数论

### 线性筛

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

void get\_p**(){**

phi**[**1**]=**mu**[**1**]=**1**;**

**for(**int i**=**2**;**i**<=**n**;**i**++){**

**if(!**p**[**i**]){**

prime**[++**prime**[**0**]]=**i**;**

phi**[**i**]=**i**-**1**;**mu**[**i**]=-**1**;**

**}for(**int j**=**1**;**j**<=**prime**[**0**];**j**++){**

**if(**i**\***prime**[**j**]>**n**)break;**

p**[**i**\***prime**[**j**]]=**1**;**

**if(**i**%**prime**[**j**]==**0**){**

mu**[**i**\***prime**[**j**]]=**0**;**

phi**[**i**\***prime**[**j**]]=**phi**[**i**]\***prime**[**j**];**

**break;**

**}else{**

phi**[**i**\***prime**[**j**]]=**phi**[**i**]\*(**prime**[**j**]-**1**);**

mu**[**i**\***prime**[**j**]]=-**mu**[**i**];**

**}**

**}**

**}**

**}**

### 线性逆元

void get\_fac**(**LL p**){**

fac**[**0**]=**1**;**

**for(**int i**=**1**;**i**<=**p**;**i**++)**fac**[**i**]=**fac**[**i**-**1**]\***i**%**p**;**

**}**

void get\_inv**(**LL p**){**

\_inv**[**1**]=**1**;**

**for(**int i**=**2**;**i**<=**p**;**i**++)**\_inv**[**i**]=(**p**-**p**/**i**)\***\_inv**[**p**%**i**]%**p**;**

**}**

### 扩展欧几里得

LL gcd**(**LL a**,**LL b**){**

**while(**b**){**

LL t**=**a**%**b**;**

a**=**b**;**

b**=**t**;**

**}return** a**;**

**}**

LL exgcd**(**LL a**,**LL b**,**LL **&**x**,**LL **&**y**){**

**if(!**b**){**

x**=**1**;**y**=**0**;return** a**;**

**}else{**

LL d**=**exgcd**(**b**,**a**%**b**,**x**,**y**);**

LL t**=**x**;**x**=**y**;**y**=**t**-**a**/**b**\***y**;**

**return** d**;**

**}**

**}**

### 求Phi

LL \_phi**(**LL x**){**

LL ans**=**x**;**

**for(**int i**=**2**;**i**\***i**<=**x**;**i**++){**

**if(**x**%**i**==**0**){**

ans**=**ans**\***i**/(**i**-**1**);**

**while(**x**%**i**==**0**)**x**/=**i**;**

**}**

**}if(**x**>**1**)**ans**=**ans**/**x**\*(**x**-**1**);**

**return** ans**;**

**}**

### 快速乘

LL \_mul**(**LL x**,**LL y**,**LL p**){**

LL ans**=**0**;**

**for(;**y**;**y**>>=**1**){**

**if(**y**&**1**)**ans**=(**ans**+**x**)%**p**;**

x**=(**x**+**x**)%**p**;**

**}return** ans**;**

**}**

LL mul**(**LL x**,**LL y**,**LL p**){**

**if(**y**==**0**)return** 0**;**

**if(**y**==**1**)return** x**%**p**;**

LL res**=**mul**(**x**,**y**>>**1**,**p**);**

**if((**y**&**1**)==**1**)return** **(**res**+**res**+**x**)%**p**;**

**else** **return** **(**res**+**res**)%**p**;**

**}**

### 快速幂

LL power**(**LL x**,**LL k**,**LL p**){**

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

**for(;**k**;**k**>>=**1**){**

**if(**k**&**1**)**ans**=**mul**(**ans**,**x**,**p**);**

x**=**mul**(**x**,**x**,**p**);**

**}return** ans**;**

**}**

### 求逆元

LL inv**(**LL a**,**LL p**){**

LL d**,**x**,**y**;**

d**=**exgcd**(**a**,**p**,**x**,**y**);**

**return** d**==**1**?(**x**+**p**)%**p**:-**1**;**

**}**

### 中国剩余定理

LL china**(**int n**,**int **\***a**,**int **\***m**){**

LL M**=**1**,**d**,**x**=**0**,**y**;**

**for(**int i**=**0**;**i**<**n**;**i**++)**M**\*=**m**[**i**];**

**for(**int i**=**0**;**i**<**n**;**i**++){**

LL w**=**M**/**m**[**i**];**

d**=**exgcd**(**m**[**i**],**w**,**d**,**y**);**

x**=(**x**+**y**\***w**\***a**[**i**])%**M**;**

**}while(**x**<=**0**)**x**+=**M**;return** x**;**

**}**

### 离散对数

LL BSGS**(**LL a**,**LL b**,**LL p**){**

LL m**=**sqrt**(**p**)+**.5**,**v**=**inv**(**power**(**a**,**m**,**p**),**p**),**e**=**1**;**

map**<**LL**,**LL**>**hash**;**hash**[**1**]=**0**;**

**for(**int i**=**1**;**i**<**m**;**i**++)**

e**=**e**\***a**%**p**,**hash**[**e**]=**i**;**

**for(**int i**=**0**;**i**<=**m**;**i**++){**

**if(**hash**.**count**(**b**))return** i**\***m**+**hash**[**b**];**

b**=**b**\***v**%**p**;**

**}return** **-**1**;**

**}**

### 卢卡斯定理

LL C**(**LL n**,**LL m**,**LL p**){**

**if(**n**<**m**)return** 0**;**

**return** fac**[**n**]\***inv**(**fac**[**m**]\***fac**[**n**-**m**]%**p**,**p**)%**p**;**

**}**

LL \_C**(**LL n**,**LL m**,**LL p**){**

**if(**n**<**m**)return** 0**;**

LL ans**=**1**,**res**=**1**;**

**for(**int i**=**1**;**i**<=**n**;**i**++)**ans**=**ans**\***i**%**p**;**

**for(**int i**=**1**;**i**<=**n**-**m**;**i**++)**res**=**res**\***i**%**p**;**

**for(**int i**=**1**;**i**<=**m**;**i**++)**res**=**res**\***i**%**p**;**

res**=**inv**(**res**,**p**);**

ans**=(**ans**\***res**)%**p**;**

**}**

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

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

**while(**n**&&**m**){**

LL a**=**n**%**p**,**b**=**m**%**p**;**

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

ans**=(**ans**\***C**(**a**,**b**,**p**))%**p**;**

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

**}return** ans**;**

**}**

### 伪素性检验

bool isprime**(**LL x**){**

LL sqrtx**=**sqrt**(**x**);**

**if(**x**==**1**)return** **false;**

**for(**int i**=**1**;**i**<=**7**;**i**++){**

LL a**=**rand**()%(**x**-**1**)+**1**;**

**if(**power**(**a**,**x**-**1**,**x**)!=**1**)return** **false;**

**}return** **true;**

**}**

### Prho

LL prho**(**LL n**,**LL c**){**

LL i**=**1**,**k**=**2**,**x**=**rand**()%(**n**-**1**)+**1**,**y**=**x**;**

**while(**1**){**

i**++;**x**=(**x**\***x**%**n**+**c**)%**n**;**

LL d**=**gcd**((**y**-**x**+**n**)%**n**,**n**);**

**if(**d**>**1**&&**d**<**n**)return** d**;**

**if(**y**==**x**)return** n**;**

**if(**i**==**k**)**y**=**x**,**k**<<=**1**;**

**}**

**}**

void factor**(**LL n**,**LL c**){**

**if(**n**==**1**)return;**

**if(**isprime**(**n**)){**

fat**[++**fat**[**0**]]=**n**;**

**return;**

**}**LL p**=**n**,**k**=**c**;**

**while(**p**>=**n**)**p**=**prho**(**p**,**c**--);**

factor**(**p**,**k**);**

factor**(**n**/**p**,**k**);**

**}**

### 扩展离散对数

void solve2**(){**

LL a**,**b**,**p**;**

scanf**(**"%lld%lld%lld"**,&**a**,&**b**,&**p**);**

b**%=**p**;**

LL e**=**1**%**p**;**

**for(**int i**=**0**;**i**<**100**;**i**++){**

**if(**e**==**b**)return** **(**void**)**printf**(**"%d\n"**,**i**);**

e**=**e**\***a**%**p**;**

**}**

int r**=**0**;**

**while(**gcd**(**a**,**p**)!=**1**){**

LL d**=**gcd**(**a**,**p**);**

**if(**b**%**d**)return** **(**void**)**puts**(**"Math Error"**);**

p**/=**d**;**b**/=**d**;**b**=**b**\***inv**(**a**/**d**,**p**);**

r**++;**

**}**LL res**=**BSGS**(**a**,**b**,**p**);**

**if(**res**==-**1**)return** **(**void**)**puts**(**"Math Error"**);**

printf**(**"%lld\n"**,**res**+**r**);**

**}**

### 不定方程

void sol**(**LL a**,**LL b**,**LL c**,**LL **&**x**,**LL **&**y**){**

LL d**=**exgcd**(**a**,**b**,**x**,**y**);**

x**=**x**\***c**/**d**;**y**=**y**\***c**/**d**;**

**}**

### 垃圾桶算法

void merge**(**LL **&**A**,**LL **&**B**,**LL a**,**LL b**){**

LL x**,**y**;**

sol**(**A**,-**a**,**b**-**B**,**x**,**y**);**

A**=**lcm**(**A**,**a**);**

B**=(**a**\***y**+**b**)%**A**;**

B**=(**B**+**A**)%**A**;**

**}**

### 组合数取模

LL prod**=**1**,**P**;**

pair**<**LL**,**LL**>** comput**(**LL n**,**LL p**,**LL k**){**

**if(**n**<=**1**)return** make\_pair**(**0**,**1**);**

LL ans**=**1**,**cnt**=**0**;**

ans**=**pow**(**prod**,**n**/**P**,**P**);**

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

pair**<**LL**,**LL**>**res**=**comput**(**n**/**p**,**p**,**k**);**

cnt**+=**res**.**first**;**

ans**=**ans**\***res**.**second**%**P**;**

**for(**int i**=**n**-**n**%**P**+**1**;**i**<=**n**;**i**++)if(**i**%**p**){**

ans**=**ans**\***i**%**P**;**

**}**

**return** make\_pair**(**cnt**,**ans**);**

**}**

pair**<**LL**,**LL**>** calc**(**LL n**,**LL p**,**LL k**){**

prod**=**1**;**P**=**pow**(**p**,**k**,**1e18**);**

**for(**int i**=**1**;**i**<**P**;**i**++)if(**i**%**p**)**prod**=**prod**\***i**%**P**;**

pair**<**LL**,**LL**>** res**=**comput**(**n**,**p**,**k**);**

// res.second=res.second\*pow(p,res.first%k,P)%P;

// res.first-=res.first%k;

**return** res**;**

**}**

LL calc**(**LL n**,**LL m**,**LL p**,**LL k**){**

pair**<**LL**,**LL**>**A**,**B**,**C**;**

LL P**=**pow**(**p**,**k**,**1e18**);**

A**=**calc**(**n**,**p**,**k**);**

B**=**calc**(**m**,**p**,**k**);**

C**=**calc**(**n**-**m**,**p**,**k**);**

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

ans**=**pow**(**p**,**A**.**first**-**B**.**first**-**C**.**first**,**P**);**

ans**=**ans**\***A**.**second**%**P**\***inv**(**B**.**second**,**P**)%**P**\***inv**(**C**.**second**,**P**)%**P**;**

**return** ans**;**

**}**

void solve3**(){**

LL n**,**m**,**P**;**scanf**(**"%lld%lld%lld"**,&**m**,&**n**,&**P**);**

**if(**n**<**m**)return** **(**void**)**puts**(**"0"**);**

//C(n,m) mod P

//P=\prod p\_i^{a\_i}

//n! = p^x \times s

//CRT ?

vector**<**pair**<**int**,**int**>** **>**vec**;**

**for(**int i**=**2**;**i**<=**P**;**i**++){**

**if(**P**%**i**==**0**){**

vec**.**push\_back**(**make\_pair**(**i**,**1**));**

P**/=**i**;**

**while(**P**%**i**==**0**){**

vec**.**back**().**second**++;**

P**/=**i**;**

**}**

**}**

**}**

LL A**=**1**,**B**=**0**;**

**for(**int i**=**0**;**i**<**vec**.**size**();**i**++)**

merge**(**A**,**B**,**pow**(**vec**[**i**].**first**,**vec**[**i**].**second**),**calc**(**n**,**m**,**vec**[**i**].**first**,**vec**[**i**].**second**));**

printf**(**"%lld\n"**,**B**);**

**}**

### MR

int strong\_pseudo\_primetest**(**long long n**,**int base**)** **{**

long long n2**=**n**-**1**,**res**;**

int s**;** s**=**0**;**

**while(**n2**%**2**==**0**)** n2**>>=**1**,**s**++;**

res**=**power**(**base**,**n2**,**n**);**

**if((**res**==**1**)||(**res**==**n**-**1**))** **return** 1**;**

s**--;**

**while(**s**>=**0**)** **{**

res**=**\_mul**(**res**,**res**,**n**);**

**if(**res**==**n**-**1**)** **return** 1**;**

s**--;**

**}**

**return** 0**;** // n is not a strong pseudo prime

**}**

int isprime**(**long long n**)** **{**

**if(**n**<**2**)** **return** 0**;**

**if(**n**<**4**)** **return** 1**;**

**if(**strong\_pseudo\_primetest**(**n**,**2**)==**0**)** **return** 0**;**

**if(**strong\_pseudo\_primetest**(**n**,**3**)==**0**)** **return** 0**;**

**if(**n**<**1373653LL**)** **return** 1**;**

**if(**strong\_pseudo\_primetest**(**n**,**5**)==**0**)** **return** 0**;**

**if(**n**<**25326001LL**)** **return** 1**;**

**if(**strong\_pseudo\_primetest**(**n**,**7**)==**0**)** **return** 0**;**

**if(**n**==**3215031751LL**)** **return** 0**;**

**if(**n**<**25000000000LL**)** **return** 1**;**

**if(**strong\_pseudo\_primetest**(**n**,**11**)==**0**)** **return** 0**;**

**if(**n**<**2152302898747LL**)** **return** 1**;**

**if(**strong\_pseudo\_primetest**(**n**,**13**)==**0**)** **return** 0**;**

**if(**n**<**3474749660383LL**)** **return** 1**;**

**if(**strong\_pseudo\_primetest**(**n**,**17**)==**0**)** **return** 0**;**

**if(**n**<**341550071728321LL**)** **return** 1**;**

**if(**strong\_pseudo\_primetest**(**n**,**19**)==**0**)** **return** 0**;**

**if(**strong\_pseudo\_primetest**(**n**,**23**)==**0**)** **return** 0**;**

**if(**strong\_pseudo\_primetest**(**n**,**29**)==**0**)** **return** 0**;**

**if(**strong\_pseudo\_primetest**(**n**,**31**)==**0**)** **return** 0**;**

**if(**strong\_pseudo\_primetest**(**n**,**37**)==**0**)** **return** 0**;**

**return** 1**;**

**}**

# 数值算法

### FFT

**typedef** complex**<**double**>** cp**;**

const int maxn**=**1**<<**18**;**

const double pi**=**acos**(-**1**);**

cp a**[**maxn**],**b**[**maxn**],**tmp**[**maxn**];**

int n**,**N**;**

void FFT**(**cp **\***a**,**int n**,**int f**){**

**if(**n**==**1**)return** **;**

copy**(**a**,**a**+**n**,**tmp**);**

**for(**int i**=**0**;**i**<**n**;**i**++)if(**i**&**1**)**a**[(**i**>>**1**)+**n**/**2**]=**tmp**[**i**];else** a**[**i**>>**1**]=**tmp**[**i**];**

FFT**(**a**,**n**/**2**,**f**);**

FFT**(**a**+**n**/**2**,**n**/**2**,**f**);**

cp w**(**1**,**0**),**wn**(**cos**(**2**\***pi**/**n**),**f**\***sin**(**2**\***pi**/**n**));**

**for(**int i**=**0**;**i**<**n**/**2**;**i**++){**

cp x**=**a**[**i**],**y**=**a**[**i**+**n**/**2**];**

a**[**i**]=**x**+**y**\***w**;**

a**[**i**+**n**/**2**]=**x**-**y**\***w**;**

w**=**w**\***wn**;**

**}**

**}**

int main**(){**

// freopen("bzoj2194.in","r",stdin);

scanf**(**"%d"**,&**n**);**N**=**1**;**

**while(**N**<**n**)**N**<<=**1**;**N**<<=**1**;**

**for(**int i**=**0**;**i**<**n**;**i**++){**

double x**,**y**;**scanf**(**"%lf%lf"**,&**x**,&**y**);**

a**[**n**-**i**-**1**]=**x**;**

b**[**i**]=**y**;**

**}**

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

FFT**(**b**,**N**,**1**);**

**for(**int i**=**0**;**i**<**N**;**i**++)**a**[**i**]\*=**b**[**i**];**

FFT**(**a**,**N**,-**1**);**

**for(**int i**=**0**;**i**<**N**;**i**++)**a**[**i**]/=**N**;**

**for(**int i**=**n**-**1**;**i**>=**0**;**i**--)**

printf**(**"%d\n"**,**int**(**a**[**i**].**real**()+**.5**));**

**return** 0**;**

**}**

### NTT

const int M**=**998244353**;**

const int mo**=**1e9**+**7**;**

const int G**=**3**;**

const int maxn**=(**1**<<**18**)+**5**;**

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

LL pw**(**LL x**,**LL k**,**LL p**){**LL a**=**1**;for(;**k**;**k**>>=**1**,**x**=**x**\***x**%**p**)if(**k**&**1**)**a**=**a**\***x**%**p**;return** a**;}**

int n**,**m**,**N**,**flag**,**inv**[**maxn**];**

int a**[**maxn**],**b**[**maxn**],**tmp**[**maxn**];**

void NTT**(**int **\***a**,**int n**){**

**if(**n**==**1**)return;**

register int i**;**

**for(**i**=**0**;**i**<**n**;**i**++)**

**if(**i**&**1**)**tmp**[**i**/**2**+**n**/**2**]=**a**[**i**];**

**else** tmp**[**i**/**2**]=**a**[**i**];**

**for(**i**=**0**;**i**<**n**;**i**++)**a**[**i**]=**tmp**[**i**];**

NTT**(**a**,**n**>>**1**);**

NTT**(**a**+**n**/**2**,**n**>>**1**);**

register int w**=**1**,**wn**=**pw**(**G**,(**M**-**1**)/**n**,**M**),**x**,**y**;**

**if(**flag**==-**1**)**wn**=**pw**(**wn**,**M**-**2**,**M**);**

**for(**i**=**0**;**i**<**n**/**2**;**i**++){**

x**=**a**[**i**],**y**=**a**[**i**+**n**/**2**];**

a**[**i**]=(**x**+(**LL**)**y**\***w**%**M**)%**M**;**

a**[**i**+**n**/**2**]=(**x**-(**LL**)**y**\***w**%**M**+**M**)%**M**;**

w**=(**LL**)**w**\***wn**%**M**;**

**}**

**}**

int main**(){**

scanf**(**"%d%d"**,&**n**,&**m**);**n**++;**m**++;**

N**=**1**;while(**N**<**max**(**n**,**m**))**N**<<=**1**;**N**<<=**1**;**

**for(**int i**=**0**;**i**<**n**;**i**++)**scanf**(**"%d"**,&**a**[**i**]);**

**for(**int i**=**0**;**i**<**m**;**i**++)**scanf**(**"%d"**,&**b**[**i**]);**

flag**=**1**;**

NTT**(**a**,**N**);**

NTT**(**b**,**N**);**

**for(**int i**=**0**;**i**<**N**;**i**++)**a**[**i**]=(**LL**)**a**[**i**]\***b**[**i**]%**M**;**

flag**=-**1**;**

NTT**(**a**,**N**);**

LL invN**=**pw**(**N**,**M**-**2**,**M**);**

**for(**int i**=**0**;**i**<**N**;**i**++)**a**[**i**]=(**LL**)**a**[**i**]\***invN**%**M**;**

**for(**int i**=**0**;**i**<**n**+**m**-**1**;**i**++)**

printf**(**"%d%c"**,**a**[**i**],**" \n"**[**i**+**1**==**n**+**m**-**1**]);**

**return** 0**;**

**}**

### 高斯消元

void Gauss**(){**

int r**,**k**;**

**for(**int i**=**0**;**i**<**n**;**i**++){**

r**=**i**;**

**for(**int j**=**i**+**1**;**j**<**n**;**j**++)**

**if(**fabs**(**A**[**j**][**i**])>**fabs**(**A**[**r**][**i**]))**r**=**j**;**

**if(**r**!=**i**)for(**int j**=**0**;**j**<=**n**;**j**++)**swap**(**A**[**i**][**j**],**A**[**r**][**j**]);**

**for(**int k**=**i**+**1**;**k**<**n**;**k**++){**

double f**=**A**[**k**][**i**]/**A**[**i**][**i**];**

**for(**int j**=**i**;**j**<=**n**;**j**++)**A**[**k**][**j**]-=**f**\***A**[**i**][**j**];**

**}**

**}**

**for(**int i**=**n**-**1**;**i**>=**0**;**i**--){**

**for(**int j**=**i**+**1**;**j**<**n**;**j**++)**

A**[**i**][**n**]-=**A**[**j**][**n**]\***A**[**i**][**j**];**

A**[**i**][**n**]/=**A**[**i**][**i**];**

**}**

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

cout**<<**fixed**<<**setprecision**(**3**)<<**A**[**i**][**n**]<<**" "**;**

cout**<<**fixed**<<**setprecision**(**3**)<<**A**[**n**-**1**][**n**];**

**}**

bool Gauss**(){**

**for(**int i**=**1**;**i**<=**n**;**i**++){**

int r**=**0**;**

**for(**int j**=**i**;**j**<=**m**;**j**++)**

**if(**a**[**j**][**i**]){**r**=**j**;break;}**

**if(!**r**)return** 0**;**

ans**=**max**(**ans**,**r**);**

swap**(**a**[**i**],**a**[**r**]);**

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

**if(**a**[**j**][**i**])**a**[**j**]^=**a**[**i**];**

**}for(**int i**=**n**;**i**>=**1**;**i**--){**

**for(**int j**=**i**+**1**;**j**<=**n**;**j**++)if(**a**[**i**][**j**])**

a**[**i**][**n**+**1**]=**a**[**i**][**n**+**1**]^**a**[**j**][**n**+**1**];**

**}return** 1**;**

**}**

LL Gauss**(){**

**for(**int i**=**0**;**i**<**n**;**i**++)for(**int j**=**0**;**j**<**n**;**j**++)**A**[**i**][**j**]%=**m**;**

**for(**int i**=**0**;**i**<**n**;**i**++)for(**int j**=**0**;**j**<**n**;**j**++)**A**[**i**][**j**]=(**A**[**i**][**j**]+**m**)%**m**;**

LL ans**=**n**%**2**?-**1**:**1**;**

**for(**int i**=**0**;**i**<**n**;**i**++){**

**for(**int j**=**i**+**1**;**j**<**n**;**j**++){**

**while(**A**[**j**][**i**]){**

LL t**=**A**[**i**][**i**]/**A**[**j**][**i**];**

**for(**int k**=**0**;**k**<**n**;**k**++)**

A**[**i**][**k**]=(**A**[**i**][**k**]-**A**[**j**][**k**]\***t**%**m**+**m**)%**m**;**

swap**(**A**[**i**],**A**[**j**]);**

ans**=-**ans**;**

**}**

**}**ans**=**ans**\***A**[**i**][**i**]%**m**;**

**}return** **(**ans**%**m**+**m**)%**m**;**

**}**

# 数据结构

### fhqTreap

int n**,**m**;**

#define sz(x) (x?x->siz:0)

struct node**{**

int siz**,**key**;**

LL val**,**sum**;**

LL mu**,**a**,**d**;**

node **\***c**[**2**],\***f**;**

void split**(**int ned**,**node **\*&**p**,**node **\*&**q**);**

node**\*** rz**(){**

sum**=**val**;**siz**=**1**;**

**if(**c**[**0**])**sum**+=**c**[**0**]->**sum**,**siz**+=**c**[**0**]->**siz**;**

**if(**c**[**1**])**sum**+=**c**[**1**]->**sum**,**siz**+=**c**[**1**]->**siz**;**

**return** **this;**

**}**

void make**(**LL \_mu**,**LL \_a**,**LL \_d**){**

sum**=**sum**\***\_mu**+**\_a**\***siz**+**\_d**\***siz**\*(**siz**-**1**)/**2**;**

val**=**val**\***\_mu**+**\_a**+**\_d**\***sz**(**c**[**0**]);**

mu**\*=**\_mu**;**a**=**a**\***\_mu**+**\_a**;**d**=**d**\***\_mu**+**\_d**;**

**}**

void pd**(){**

**if(**mu**==**1**&&**a**==**0**&&**d**==**0**)return;**

**if(**c**[**0**])**c**[**0**]->**make**(**mu**,**a**,**d**);**

**if(**c**[**1**])**c**[**1**]->**make**(**mu**,**a**+**d**+**d**\***sz**(**c**[**0**]),**d**);**

mu**=**1**;**a**=**d**=**0**;**

**}**

node**(){**mu**=**1**;}**

**}**nd**[**maxn**\***2**],\***root**;**

node **\***merge**(**node **\***p**,**node **\***q**){**

**if(!**p**||!**q**)return** p**?**p**->**rz**():(**q**?**q**->**rz**():**0**);**

p**->**pd**();**q**->**pd**();**

**if(**p**->**key**<**q**->**key**){**

p**->**c**[**1**]=**merge**(**p**->**c**[**1**],**q**);**

**return** p**->**rz**();**

**}else{**

q**->**c**[**0**]=**merge**(**p**,**q**->**c**[**0**]);**

**return** q**->**rz**();**

**}**

**}**

void node**::**split**(**int ned**,**node **\*&**p**,**node **\*&**q**){**

**if(!**ned**){**p**=**0**;**q**=this;return;}**

**if(**ned**==**siz**){**p**=this;**q**=**0**;return;}**

pd**();**

**if(**sz**(**c**[**0**])>=**ned**){**

c**[**0**]->**split**(**ned**,**p**,**q**);**c**[**0**]=**0**;**rz**();**

q**=**merge**(**q**,this);**

**}else{**

c**[**1**]->**split**(**ned**-**sz**(**c**[**0**])-**1**,**p**,**q**);**c**[**1**]=**0**;**rz**();**

p**=**merge**(this,**p**);**

**}**

**}**

int tot**;**

void C**(**int l**,**int r**,**int v**){**

node **\***p**,\***q**,\***x**,\***y**;**

root**->**split**(**l**-**1**,**p**,**q**);**

q**->**split**(**r**-**l**+**1**,**x**,**y**);**

x**->**make**(**0**,**v**,**0**);**x**->**pd**();**

root**=**merge**(**p**,**merge**(**x**,**y**));**

**}**

void A**(**int l**,**int r**,**int d**){**

node **\***p**,\***q**,\***x**,\***y**;**

root**->**split**(**l**-**1**,**p**,**q**);**

q**->**split**(**r**-**l**+**1**,**x**,**y**);**

x**->**make**(**1**,**d**,**d**);**x**->**pd**();**

root**=**merge**(**p**,**merge**(**x**,**y**));**

**}**

void I**(**int ps**,**int v**){**

node **\***p**,\***q**;**

root**->**split**(**ps**-**1**,**p**,**q**);**

node **\***x**=**nd**+(++**tot**);**

x**->**key**=**rand**();**x**->**val**=**v**;**x**->**rz**();**

root**=**merge**(**merge**(**p**,**x**),**q**);**

**}**

LL Q**(**int l**,**int r**){**

node **\***p**,\***q**,\***x**,\***y**;**

root**->**split**(**l**-**1**,**p**,**q**);**

q**->**split**(**r**-**l**+**1**,**x**,**y**);**

LL ans**=**x**->**sum**;**

root**=**merge**(**p**,**merge**(**x**,**y**));**

**return** ans**;**

**}**

int main**(){**

// freopen("bzoj3188.in","r",stdin);

n**=**in**();**m**=**in**();**

**for(**int i**=**1**;**i**<=**n**;**i**++){**

nd**[**i**].**val**=**in**();**

nd**[**i**].**key**=**rand**();**

nd**[**i**].**rz**();**

root**=**merge**(**root**,**nd**+**i**);**

**}**tot**=**n**;**

**while(**m**--){**

int ty**=**in**();**

int l**,**r**;**

**if(**ty**==**1**){**

l**=**in**();**r**=**in**();**

C**(**l**,**r**,**in**());**

**}else** **if(**ty**==**2**){**

l**=**in**();**r**=**in**();**

A**(**l**,**r**,**in**());**

**}else** **if(**ty**==**3**){**

int ps**=**in**();**

I**(**ps**,**in**());**

**}else** **if(**ty**==**4**){**

l**=**in**();**r**=**in**();**

printf**(**"%lld\n"**,**Q**(**l**,**r**));**

**}**

**}**

**return** 0**;**

**}**

### LCT

struct LCT**{**

struct node**{**

bool rev**;**

int mx**,**val**;**

node **\***f**,\***c**[**2**];**

bool d**(){return** **this==**f**->**c**[**1**];}**

bool rt**(){return** **!**f**||(**f**->**c**[**0**]!=this&&**f**->**c**[**1**]!=this);}**

void sets**(**node **\***x**,**int d**){**pd**();if(**x**)**x**->**f**=this;**c**[**d**]=**x**;**rz**();}**

void makerv**(){**rev**^=**1**;**swap**(**c**[**0**],**c**[**1**]);}**

void pd**(){**

**if(**rev**){**

**if(**c**[**0**])**c**[**0**]->**makerv**();**

**if(**c**[**1**])**c**[**1**]->**makerv**();**

rev**=**0**;**

**}**

**}**

void rz**(){**

mx**=**val**;**

**if(**c**[**0**])**mx**=**max**(**mx**,**c**[**0**]->**mx**);**

**if(**c**[**1**])**mx**=**max**(**mx**,**c**[**1**]->**mx**);**

**}**

**}**nd**[**int**(**1e4**)+**1**];**

void rot**(**node **\***x**){**

node **\***y**=**x**->**f**;if(!**y**->**rt**())**y**->**f**->**pd**();**

y**->**pd**();**x**->**pd**();**bool d**=**x**->**d**();**

y**->**sets**(**x**->**c**[!**d**],**d**);**

**if(**y**->**rt**())**x**->**f**=**y**->**f**;**

**else** y**->**f**->**sets**(**x**,**y**->**d**());**

x**->**sets**(**y**,!**d**);**

**}**

void splay**(**node **\***x**){**

**while(!**x**->**rt**())**

**if(**x**->**f**->**rt**())**rot**(**x**);**

**else** **if(**x**->**d**()==**x**->**f**->**d**())**rot**(**x**->**f**),**rot**(**x**);**

**else** rot**(**x**),**rot**(**x**);**

**}**

node**\*** access**(**node **\***x**){**

node **\***y**=**0**;**

**for(;**x**;**x**=**x**->**f**){**

splay**(**x**);**

x**->**sets**(**y**,**1**);**y**=**x**;**

**}return** y**;**

**}**

void makert**(**node **\***x**){**

access**(**x**)->**makerv**();**

splay**(**x**);**

**}**

void link**(**node **\***x**,**node **\***y**){**

makert**(**x**);**

x**->**f**=**y**;**

access**(**x**);**

**}**

void cut**(**node **\***x**,**node **\***y**){**

makert**(**x**);**access**(**y**);**splay**(**y**);**

y**->**c**[**0**]=**x**->**f**=**0**;**

y**->**rz**();**

**}**

void link**(**int x**,**int y**){**link**(**nd**+**x**,**nd**+**y**);}**

void cut**(**int x**,**int y**){**cut**(**nd**+**x**,**nd**+**y**);}**

**}**T**;**

### SkewHeap

const int maxn**=**1000010**;**

int fa**[**maxn**],**n**,**m**;**

int find**(**int x**){**

**if(**fa**[**x**]!=**x**)return** fa**[**x**]=**find**(**fa**[**x**]);**

**return** x**;**

**}**

bool die**[**maxn**];**

struct node**;**

node **\***Null**,\***root**[**maxn**];**

struct node**{**

node**\*** c**[**2**];**

int val**,**ind**;**

node**(**int \_val**=**0**,**int \_ind**=**0**){**

val**=**\_val**;**c**[**0**]=**c**[**1**]=**Null**;**ind**=**\_ind**;**

**}**

**};**

node**\*** merge**(**node **\***p**,**node **\***q**){**

**if(**p**==**Null**)return** q**;**

**if(**q**==**Null**)return** p**;**

**if(**p**->**val**>**q**->**val**)**swap**(**p**,**q**);**

p**->**c**[**1**]=**merge**(**p**->**c**[**1**],**q**);**

swap**(**p**->**c**[**0**],**p**->**c**[**1**]);**

**return** p**;**

**}**

void deb**(**node **\***x**){**

**if(**x**==**Null**)return;**

printf**(**"%d "**,**x**->**val**);**

deb**(**x**->**c**[**0**]);**

deb**(**x**->**c**[**1**]);**

**}**

int main**(){**

scanf**(**"%d"**,&**n**);**

Null**=new** node**(**0**);**

Null**->**c**[**0**]=**Null**->**c**[**1**]=**Null**;**

**for(**int i**=**1**;**i**<=**n**;**i**++)**fa**[**i**]=**i**;**

**for(**int i**=**1**;**i**<=**n**;**i**++){**

int x**;**scanf**(**"%d"**,&**x**);**

root**[**i**]=new** node**(**x**,**i**);**

**}**scanf**(**"%d"**,&**m**);**

**while(**m**--){**

char opt**[**2**];**

scanf**(**"%s"**,&**opt**);**

**if(**opt**[**0**]==**'M'**){**

int a**,**b**;**scanf**(**"%d%d"**,&**a**,&**b**);**

**if(**die**[**a**]||**die**[**b**])continue;**

a**=**find**(**a**);**b**=**find**(**b**);**

**if(**a**==**b**)continue;**

root**[**a**]=**root**[**b**]=**merge**(**root**[**a**],**root**[**b**]);**

fa**[**find**(**b**)]=**find**(**a**);**

**}else{**

int x**;**scanf**(**"%d"**,&**x**);**

**if(**die**[**x**]||**root**[**find**(**x**)]==**Null**)**

puts**(**"0"**);**

**else{**

x**=**find**(**x**);**

die**[**root**[**x**]->**ind**]=**1**;**

printf**(**"%d\n"**,**root**[**x**]->**val**);**

root**[**x**]=**merge**(**root**[**x**]->**c**[**0**],**root**[**x**]->**c**[**1**]);**

**}**

**}**

**}**

**return** 0**;**

**}**

### Treap

#define sz(x) (x?x->siz:0)

struct Treap**{**

struct node**{**

int key**,**val**;**

int siz**,**s**;**

node **\***c**[**2**];**

node**(**int v**=**0**){**

val**=**v**;**

key**=**rand**();**

siz**=**1**,**s**=**1**;**

c**[**0**]=**c**[**1**]=**0**;**

**}**

void rz**(){**siz**=**s**;if(**c**[**0**])**siz**+=**c**[**0**]->**siz**;if(**c**[**1**])**siz**+=**c**[**1**]->**siz**;}**

**}**pool**[**maxn**],\***cur**,\***root**;**

Treap**(){**cur**=**pool**;}**

node**\*** newnode**(**int val**){return** **\***cur**=**node**(**val**),**cur**++;}**

void rot**(**node **\*&**t**,**int d**){**

**if(!**t**->**c**[**d**])**t**=**t**->**c**[!**d**];**

**else{**

node **\***p**=**t**->**c**[**d**];**t**->**c**[**d**]=**p**->**c**[!**d**];**

p**->**c**[!**d**]=**t**;**t**->**rz**();**p**->**rz**();**t**=**p**;**

**}**

**}**

void insert**(**node **\*&**t**,**int x**){**

**if(!**t**){**t**=**newnode**(**x**);return;}**

**if(**t**->**val**==**x**){**t**->**s**++;**t**->**siz**++;return;}**

insert**(**t**->**c**[**x**>**t**->**val**],**x**);**

**if(**t**->**key**<**t**->**c**[**x**>**t**->**val**]->**key**)**

rot**(**t**,**x**>**t**->**val**);**

**else** t**->**rz**();**

**}**

void del**(**node **\*&**t**,**int x**){**

**if(!**t**)return;**

**if(**t**->**val**==**x**){**

**if(**t**->**s**>**1**){**t**->**s**--;**t**->**siz**--;return;}**

**if(!**t**->**c**[**0**]||!**t**->**c**[**1**]){**

**if(!**t**->**c**[**0**])**t**=**t**->**c**[**1**];**

**else** t**=**t**->**c**[**0**];**

**return;**

**}**

int d**=**t**->**c**[**0**]->**key**<**t**->**c**[**1**]->**key**;**

rot**(**t**,**d**);**

del**(**t**,**x**);**

**return;**

**}**

del**(**t**->**c**[**x**>**t**->**val**],**x**);**

t**->**rz**();**

**}**

int pre**(**node **\***t**,**int x**){**

**if(!**t**)return** INT\_MIN**;**

int ans**=**pre**(**t**->**c**[**x**>**t**->**val**],**x**);**

**if(**t**->**val**<**x**)**ans**=**max**(**ans**,**t**->**val**);**

**return** ans**;**

**}**

int nxt**(**node **\***t**,**int x**){**

**if(!**t**)return** INT\_MAX**;**

int ans**=**nxt**(**t**->**c**[**x**>=**t**->**val**],**x**);**

**if(**t**->**val**>**x**)**ans**=**min**(**ans**,**t**->**val**);**

**return** ans**;**

**}**

int rank**(**node **\***t**,**int x**){**

**if(!**t**)return** 0**;**

**if(**t**->**val**==**x**)return** sz**(**t**->**c**[**0**]);**

**if(**t**->**val**<**x**)return** sz**(**t**->**c**[**0**])+**t**->**s**+**rank**(**t**->**c**[**1**],**x**);**

**if(**t**->**val**>**x**)return** rank**(**t**->**c**[**0**],**x**);**

**}**

int kth**(**node **\***t**,**int x**){**

**if(**sz**(**t**->**c**[**0**])>=**x**)return** kth**(**t**->**c**[**0**],**x**);**

**if(**sz**(**t**->**c**[**0**])+**t**->**s**>=**x**)return** t**->**val**;**

**return** kth**(**t**->**c**[**1**],**x**-**t**->**s**-**sz**(**t**->**c**[**0**]));**

**}**

void deb**(**node **\***t**){**

**if(!**t**)return;**

deb**(**t**->**c**[**0**]);**

printf**(**"%d "**,**t**->**val**);**

deb**(**t**->**c**[**1**]);**

**}**

void insert**(**int x**){**insert**(**root**,**x**);}**

void del**(**int x**){**del**(**root**,**x**);}**

int pre**(**int x**){return** pre**(**root**,**x**);}**

int nxt**(**int x**){return** nxt**(**root**,**x**);}**

int rank**(**int x**){return** rank**(**root**,**x**);}**

int kth**(**int x**){return** kth**(**root**,**x**);}**

void deb**(){**deb**(**root**);**puts**(**""**);}**

**}**T**;**

### 点分治

int dcmp**(**double x**){return** **(**x**>**eps**)-(**x**<-**eps**);}**

int n**,**L**,**U**;**

double lim**;**

int getint**(){**

int res**=**0**;**char c**=**getchar**();**

**while(!**isdigit**(**c**))**c**=**getchar**();**

**while(**isdigit**(**c**))**res**=**res**\***10**+**c**-**'0'**,**c**=**getchar**();**

**return** res**;**

**}**

double ans**;**

struct edge**{**int u**,**v**,**w**;};**

vector**<**edge**>**G**[**maxn**];**

int root**,**All**;**

short cant**[**maxn**];**

int siz**[**maxn**],**f**[**maxn**],**fa**[**maxn**];**

void makert**(**int u**,**int **&**root**){**

siz**[**u**]=**1**;**f**[**u**]=**0**;**

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

edge e**=**G**[**u**][**i**];if(**cant**[**e**.**v**]||**e**.**v**==**fa**[**u**])continue;**

fa**[**e**.**v**]=**u**;**

makert**(**e**.**v**,**root**);**

siz**[**u**]+=**siz**[**e**.**v**];**

f**[**u**]=**max**(**f**[**u**],**siz**[**e**.**v**]);**

**}**f**[**u**]=**max**(**f**[**u**],**All**-**f**[**u**]);**

**if(**f**[**root**]>**f**[**u**])**root**=**u**;**

**}**

double mx**[**maxn**],**d**[**maxn**];**

int data**[**maxn**],**dep**[**maxn**];**

bool ok**(**int u**,**double mid**){**

int size**=**0**;**

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

edge e**=**G**[**u**][**i**];if(**cant**[**e**.**v**])continue;**data**[**0**]=**0**;**fa**[**e**.**v**]=**u**;**

queue**<**int**>**q**;**q**.**push**(**e**.**v**);**d**[**e**.**v**]=**e**.**w**-**mid**;**dep**[**e**.**v**]=**1**;**

**while(!**q**.**empty**()){**

int u**=**q**.**front**();**q**.**pop**();**data**[++**data**[**0**]]=**u**;**

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

edge e**=**G**[**u**][**i**];if(**cant**[**e**.**v**]||**fa**[**u**]==**e**.**v**)continue;**

fa**[**e**.**v**]=**u**;**dep**[**e**.**v**]=**dep**[**u**]+**1**;**q**.**push**(**e**.**v**);**

d**[**e**.**v**]=**d**[**u**]+**e**.**w**-**mid**;**

**}**

**}**deque**<**int**>**dq**;**int now**=**size**;**

**for(**int j**=**1**;**j**<=**data**[**0**];**j**++){**

int u**=**data**[**j**];**

**while(**now**>=**0**&&**dep**[**u**]+**now**>=**L**){**

**while(**dq**.**size**()&&**mx**[**dq**.**back**()]<**mx**[**now**])**dq**.**pop\_back**();**

dq**.**push\_back**(**now**--);**

**}while(**dq**.**size**()&&**dq**.**front**()+**dep**[**u**]>**U**)**dq**.**pop\_front**();**

**if(**dq**.**size**()&&**d**[**u**]+**mx**[**dq**.**front**()]>=**0**)return** 1**;**

**}for(**int j**=**size**+**1**;**j**<=**dep**[**data**[**data**[**0**]]];**j**++)**mx**[**j**]=-**1e10**;**

**for(**int j**=**1**;**j**<=**data**[**0**];**j**++){**

int u**=**data**[**j**];**

mx**[**dep**[**u**]]=**max**(**mx**[**dep**[**u**]],**d**[**u**]);**

**}**size**=**max**(**size**,**dep**[**data**[**data**[**0**]]]);**

**}return** 0**;**

**}**

void calc**(**int u**){**

double l**=**ans**,**r**=**lim**;**

**while(**dcmp**(**l**-**r**)){**

double mid**=(**l**+**r**)\***.5**;**

**if(**ok**(**u**,**mid**))**l**=**mid**;**

**else** r**=**mid**;**

**}**ans**=**l**;**

**}**

void solve**(**int u**){**

calc**(**u**);**cant**[**u**]=**1**;**

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

edge e**=**G**[**u**][**i**];if(**cant**[**e**.**v**]||**siz**[**e**.**v**]<**L**)continue;**

f**[**root**=**0**]=**n**+**1**;**All**=**siz**[**e**.**v**];**

makert**(**e**.**v**,**root**);**

solve**(**root**);**

**}**

**}**

int main**(){**

n**=**getint**();**L**=**getint**();**U**=**getint**();**All**=**n**;**

**for(**int i**=**1**;**i**<**n**;**i**++){**

int u**=**getint**(),**v**=**getint**(),**w**=**getint**();**

lim**=**max**(**lim**,**double**(**w**));**

G**[**u**].**push\_back**((**edge**){**u**,**v**,**w**});**

G**[**v**].**push\_back**((**edge**){**v**,**u**,**w**});**

**}**f**[**root**=**0**]=**n**+**1**;**

makert**(**1**,**root**);**

solve**(**root**);**

cout**<<**fixed**<<**setprecision**(**3**)<<**ans**<<**endl**;**

**return** 0**;**

**}**

### HLD

vector**<**int**>**G**[**maxn**];**

int fa**[**maxn**],**top**[**maxn**],**siz**[**maxn**],**son**[**maxn**],**mp**[**maxn**],**z**,**dep**[**maxn**];**

void dfs**(**int u**){**

siz**[**u**]=**1**;**

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

int v**=**G**[**u**][**i**];**

**if(**v**!=**fa**[**u**]){**

fa**[**v**]=**u**;**dep**[**v**]=**dep**[**u**]+**1**;**

dfs**(**v**);**

siz**[**u**]+=**siz**[**v**];**

**if(**siz**[**son**[**u**]]<**siz**[**v**])**son**[**u**]=**v**;**

**}**

**}**

**}**

void build**(**int u**,**int tp**){**

top**[**u**]=**tp**;**mp**[**u**]=++**z**;**

**if(**son**[**u**])**build**(**son**[**u**],**tp**);**

**for(**int v**,**i**=**0**;**i**<**G**[**u**].**size**();**i**++)**

**if((**v**=**G**[**u**][**i**])!=**son**[**u**]&&**v**!=**fa**[**u**])**build**(**v**,**v**);**

**}**

int main**(){**

**return** 0**;**

**}**

### 左偏树

struct node**{**

int val**,**dis**;**

node **\***c**[**2**];**

**}\***Null**,**poor**[**maxn**<<**2**];**

**namespace** TRT**{**

struct LeftistTree**{**

node **\***root**;**

int sum**,**size**;**

void init**(){**

sum**=**size**=**0**;**

root**=**Null**;**

**}**

node **\***merge**(**node **\*&**A**,**node **\*&**B**){**

**if(**A**==**Null**)return** B**;**

**if(**B**==**Null**)return** A**;**

**if(**A**->**val**<**B**->**val**)**

swap**(**A**,**B**);**

A**->**c**[**1**]=**merge**(**A**->**c**[**1**],**B**);**

**if(**A**->**c**[**0**]->**dis**<**A**->**c**[**1**]->**dis**)**

swap**(**A**->**c**[**0**],**A**->**c**[**1**]);**

A**->**dis**=**A**->**c**[**1**]->**dis**+**1**;**

**return** A**;**

**}**

void merge**(**LeftistTree **&**B**){**

size**+=**B**.**size**;**

sum**+=**B**.**sum**;**

root**=**merge**(**root**,**B**.**root**);**

**}**

void insert**(**int x**)** **{**

node **\***p **=** **&**poor**[++**tot**];**

**++**size**;**

sum **+=** x**;**

p**->**val **=** x**;**

p**->**dis **=** 0**;**

p**->**c**[**0**]** **=** p**->**c**[**1**]** **=** Null**;**

root **=** merge**(**root**,** p**);**

**}**

int top**(){**

**return** root**->**val**;**

**}**

void pop**(){**

size**--;**

sum**-=**root**->**val**;**

root**=**merge**(**root**->**c**[**0**],**root**->**c**[**1**]);**

**}**

**}**t**[**maxn**];**

**}**

### 莫队

const int maxn**=**50010**;**

**typedef** long long lld**;**

lld n**,**m**;**

lld col**[**maxn**];**

lld a**[**maxn**],**sqrn**;**

lld bel**(**lld x**){return** **(**x**-**1**)/**sqrn**;}**

lld getlld**(){**

lld res**=**0**,**ok**=**0**;**char ch**;**

**while(**1**){**

ch**=**getchar**();**

**if(**isdigit**(**ch**)){**

res**\*=**10**;**res**+=**ch**-**'0'**;**ok**=**1**;**

**}else** **if(**ok**)break;**

**}return** res**;**

**}**

struct qes**{**

lld ind**,**l**,**r**;**

bool **operator** **<** **(**const qes **&**a**)**const**{**

**return** bel**(**l**)<**bel**(**a**.**l**)||**bel**(**l**)==**bel**(**a**.**l**)&&**r**<**a**.**r**;**

**}**

**};**

lld anss**[**maxn**];**

lld ans**=**0**;**

void malldain**(**lld j**,**lld i**){**

ans**-=**col**[**a**[**j**]]\*(**col**[**a**[**j**]]-**1**);**

col**[**a**[**j**]]+=**i**;**

ans**+=**col**[**a**[**j**]]\*(**col**[**a**[**j**]]-**1**);**

**}**

bool cmp**(**const qes **&**a**,**const qes **&**b**){**

**return** a**.**ind**<**b**.**ind**;**

**}**

lld gcd**(**lld a**,**lld b**){**

**while(**b**){**

lld t**=**a**%**b**;**

a**=**b**;**

b**=**t**;**

**}return** a**;**

**}**

qes q**[**maxn**];**

int main**(){**

n**=**getlld**();**m**=**getlld**();**

sqrn**=**sqrt**(**n**)+**1**;**

**for(**lld i**=**1**;**i**<=**n**;**i**++)**a**[**i**]=**getlld**();**

**for(**lld i**=**1**;**i**<=**m**;**i**++){**

q**[**i**].**ind**=**i**;**q**[**i**].**l**=**getlld**();**q**[**i**].**r**=**getlld**();**

**}**sort**(**q**+**1**,**q**+**1**+**m**);**

lld i**=**1**;**

**while(**i**<=**m**){**

memset**(**col**,**0**,sizeof(**col**));**

ans**=**0**;**

**for(**lld j**=**q**[**i**].**l**;**j**<=**q**[**i**].**r**;**j**++)**

malldain**(**j**,**1**);**

anss**[**q**[**i**++].**ind**]=**ans**;**

**while(**i**<=**m**&&**bel**(**q**[**i**].**l**)==**bel**(**q**[**i**-**1**].**l**)){**

**for(**lld j**=**q**[**i**-**1**].**r**+**1**;**j**<=**q**[**i**].**r**;**j**++)**

malldain**(**j**,**1**);**

**if(**q**[**i**-**1**].**l**<**q**[**i**].**l**){**

**for(**lld j**=**q**[**i**-**1**].**l**;**j**<**q**[**i**].**l**;**j**++)**

malldain**(**j**,-**1**);**

**}else{**

**for(**lld j**=**q**[**i**].**l**;**j**<**q**[**i**-**1**].**l**;**j**++)**

malldain**(**j**,**1**);**

**}**

anss**[**q**[**i**++].**ind**]=**ans**;**

**}**

**}**

sort**(**q**+**1**,**q**+**1**+**m**,**cmp**);**

**for(**lld i**=**1**;**i**<=**m**;**i**++){**

lld k**=(**q**[**i**].**r**-**q**[**i**].**l**+**1**)\*(**q**[**i**].**r**-**q**[**i**].**l**);**

**if(**k**==**0**||**anss**[**i**]==**0**){**

puts**(**"0/1"**);continue;**

**}**

lld d**=**gcd**(**k**,**anss**[**i**]);**

printf**(**"%lld/%lld\n"**,**anss**[**i**]/**d**,**k**/**d**);**

**}**

**return** 0**;**

**}**

### 树上莫队

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

int n**,**m**,**q**,**col**[**maxn**],**Qsize**,**Csize**,**B**,**dfn**[**maxn**],**pre**[**maxn**],**vis**[**maxn**],**cnt**[**maxn**],**now**;**

LL V**[**maxn**],**W**[**maxn**],**anss**[**maxn**],**ans**;**

struct qes**{**int x**,**y**,**id**,**tm**;}**Q**[**maxn**];**

struct oper**{**int x**,**y**,**pre**,**tm**;}**C**[**maxn**];**

bool **operator<(**qes a**,**qes b**){**

**if(**dfn**[**a**.**x**]/**B**!=**dfn**[**b**.**x**]/**B**)return** dfn**[**a**.**x**]/**B**<**dfn**[**b**.**x**]/**B**;**

**if(**dfn**[**a**.**y**]/**B**!=**dfn**[**b**.**y**]/**B**)return** dfn**[**a**.**y**]/**B**<**dfn**[**b**.**y**]/**B**;**

**if(**a**.**tm**/**B**!=**b**.**tm**/**B**)return** a**.**tm**/**B**<**b**.**tm**/**B**;**

**return** a**.**tm**<**b**.**tm**;**

**}**

vector**<**int**>**G**[**maxn**];**

int tot**=**0**;**

int siz**[**maxn**],**son**[**maxn**],**top**[**maxn**],**z**,**mp**[**maxn**],**fa**[**maxn**],**dep**[**maxn**];**

void dfs**(**int u**){**

dfn**[**u**]=++**tot**;**siz**[**u**]=**1**;**

**for(**int i**=**0**,**v**;**i**<**G**[**u**].**size**();**i**++){**

**if((**v**=**G**[**u**][**i**])!=**fa**[**u**]){**

fa**[**v**]=**u**;**dep**[**v**]=**dep**[**u**]+**1**;**

dfs**(**v**);**siz**[**u**]+=**siz**[**v**];**

**if(**siz**[**v**]>**siz**[**son**[**u**]])**son**[**u**]=**v**;**

**}**

**}**

**}**

void build**(**int u**,**int tp**){**

mp**[**u**]=++**z**;**top**[**u**]=**tp**;**

**if(**son**[**u**])**build**(**son**[**u**],**tp**);**

**for(**int i**=**0**,**v**;**i**<**G**[**u**].**size**();**i**++)if((**v**=**G**[**u**][**i**])!=**fa**[**u**]&&**v**!=**son**[**u**])**

build**(**v**,**v**);**

**}**

int lca**(**int u**,**int v**){**

**while(**top**[**u**]!=**top**[**v**]){**

**if(**dep**[**top**[**u**]]<**dep**[**top**[**v**]])**swap**(**u**,**v**);**

u**=**fa**[**top**[**u**]];**

**}return** dep**[**u**]<**dep**[**v**]?**u**:**v**;**

**}**

void vxor**(**int x**){**

**if(**vis**[**x**])**ans**-=(**LL**)**W**[**cnt**[**col**[**x**]]]\***V**[**col**[**x**]],**cnt**[**col**[**x**]]--;**

**else** cnt**[**col**[**x**]]++,**ans**+=(**LL**)**W**[**cnt**[**col**[**x**]]]\***V**[**col**[**x**]];**

vis**[**x**]^=**1**;**

**}**

void change**(**int x**,**int y**){**

**if(**vis**[**x**]){**

vxor**(**x**);**col**[**x**]=**y**;**vxor**(**x**);**

**}else** col**[**x**]=**y**;**

**}**

void TimeMachine**(**int tar**){**//XD

**for(**int i**=**now**+**1**;**i**<=**tar**;**i**++)**change**(**C**[**i**].**x**,**C**[**i**].**y**);**

**for(**int i**=**now**;**i**>**tar**;**i**--)**change**(**C**[**i**].**x**,**C**[**i**].**pre**);**

now**=**tar**;**

**}**

void vxor**(**int x**,**int y**){**

**while(**x**!=**y**)if(**dep**[**x**]>**dep**[**y**])**vxor**(**x**),**x**=**fa**[**x**];**

**else** vxor**(**y**),**y**=**fa**[**y**];**

**}**

int getint**(){**

int res**=**0**;**char c**=**getchar**();**

**while(!**isdigit**(**c**))**c**=**getchar**();**

**while(**isdigit**(**c**))**res**=**res**\***10**+**c**-**'0'**,**c**=**getchar**();**

**return** res**;**

**}**

int main**(){**

n**=**getint**();**m**=**getint**();**q**=**getint**();**B**=**pow**(**n**,**2.0**/**3.0**);**

**for(**int i**=**1**;**i**<=**m**;**i**++)**V**[**i**]=**getint**();**

**for(**int i**=**1**;**i**<=**n**;**i**++)**W**[**i**]=**getint**();**

**for(**int i**=**1**;**i**<**n**;**i**++){**

int u**=**getint**(),**v**=**getint**();**

G**[**u**].**push\_back**(**v**);**

G**[**v**].**push\_back**(**u**);**

**}**

dfs**(**1**);**

build**(**1**,**1**);**

**for(**int i**=**1**;**i**<=**n**;**i**++)**pre**[**i**]=**col**[**i**]=**getint**();**

**for(**int i**=**1**;**i**<=**q**;**i**++){**

int ty**=**getint**(),**x**=**getint**(),**y**=**getint**();**

**if(**ty**&&**dfn**[**x**]>**dfn**[**y**])**swap**(**x**,**y**);**

**if(**ty**==**0**)** C**[++**Csize**]=(**oper**){**x**,**y**,**pre**[**x**],**i**},**pre**[**x**]=**y**;**

**else** Q**[**Qsize**+**1**]=(**qes**){**x**,**y**,**Qsize**+**1**,**Csize**},**Qsize**++;**

**}**sort**(**Q**+**1**,**Q**+**1**+**Qsize**);**

int u**=**Q**[**1**].**x**,**v**=**Q**[**1**].**y**;**

TimeMachine**(**Q**[**1**].**tm**);**

vxor**(**Q**[**1**].**x**,**Q**[**1**].**y**);**

int LCA**=**lca**(**Q**[**1**].**x**,**Q**[**1**].**y**);**

vxor**(**LCA**);**anss**[**Q**[**1**].**id**]=**ans**;**vxor**(**LCA**);**

**for(**int i**=**2**;**i**<=**Qsize**;**i**++){**

TimeMachine**(**Q**[**i**].**tm**);**

vxor**(**Q**[**i**-**1**].**x**,**Q**[**i**].**x**);**

vxor**(**Q**[**i**-**1**].**y**,**Q**[**i**].**y**);**

int LCA**=**lca**(**Q**[**i**].**x**,**Q**[**i**].**y**);**

vxor**(**LCA**);**

anss**[**Q**[**i**].**id**]=**ans**;**

vxor**(**LCA**);**

**}for(**int i**=**1**;**i**<=**Qsize**;**i**++)**printf**(**"%lld\n"**,**anss**[**i**]);**

**return** 0**;**

**}**

### KDtree

int n**;**

struct point**{**

int x**,**y**,**val**;**

int**&** **operator[](**const int s**){return** s**==**0**?**x**:**y**;}**

**}**p**[**maxn**];**

struct rec**{**

int x**[**2**],**y**[**2**];**

rec**(){}**

rec**(**point p**){**x**[**0**]=**x**[**1**]=**p**.**x**;**y**[**0**]=**y**[**1**]=**p**.**y**;}**

**};**

bool **operator==(**const point **&**A**,**const point **&**B**){**

**return** A**.**x**==**B**.**x**&&**A**.**y**==**B**.**y**;**

**}**

inline rec **operator+(**const rec **&**ls**,**const rec **&**rs**){**

static rec R**;**R**=**ls**;**

R**.**x**[**0**]=**min**(**R**.**x**[**0**],**rs**.**x**[**0**]);**

R**.**x**[**1**]=**max**(**R**.**x**[**1**],**rs**.**x**[**1**]);**

R**.**y**[**0**]=**min**(**R**.**y**[**0**],**rs**.**y**[**0**]);**

R**.**y**[**1**]=**max**(**R**.**y**[**1**],**rs**.**y**[**1**]);**

**return** R**;**

**}**

inline bool **operator\*(**const point **&**p**,**const rec **&**R**){**

**return** R**.**x**[**0**]<=**p**.**x**&&**p**.**x**<=**R**.**x**[**1**]&&**R**.**y**[**0**]<=**p**.**y**&&**p**.**y**<=**R**.**y**[**1**];**

**}**

inline bool In**(**const rec **&**A**,**const rec **&**B**){**

**return** B**.**x**[**0**]<=**A**.**x**[**0**]&&**A**.**x**[**1**]<=**B**.**x**[**1**]&&**B**.**y**[**0**]<=**A**.**y**[**0**]&&**A**.**y**[**1**]<=**B**.**y**[**1**];**

**}**

inline bool Out**(**const rec **&**A**,**const rec **&**B**){**

**return** B**.**x**[**0**]>**A**.**x**[**1**]||**A**.**x**[**0**]>**B**.**x**[**1**]||**B**.**y**[**0**]>**A**.**y**[**1**]||**A**.**y**[**0**]>**B**.**y**[**1**];**

**}**

struct node**{**

rec R**;**point p**;**

int sum**,**siz**;**

node **\***c**[**2**];**

node **\***rz**(){**

sum**=**p**.**val**;**R**=**rec**(**p**);**siz**=**1**;**

**if(**c**[**0**])**sum**+=**c**[**0**]->**sum**,**R**=**R**+**c**[**0**]->**R**,**siz**+=**c**[**0**]->**siz**;**

**if(**c**[**1**])**sum**+=**c**[**1**]->**sum**,**R**=**R**+**c**[**1**]->**R**,**siz**+=**c**[**1**]->**siz**;**

**return** **this;**

**}**

node**(){**sum**=**0**;**siz**=**1**;**c**[**0**]=**c**[**1**]=**0**;}**

**}\***root**,\***re**,**pool**[**maxn**],\***cur**=**pool**;**

node **\***sta**[**maxn**];**

int D**,**si**;**

bool cmp**(**const point **&**A**,**const point **&**B**){**

**if(**D**)return** A**.**x**<**B**.**x**||(**A**.**x**==**B**.**x**&&**A**.**y**<**B**.**y**);**

**return** A**.**y**<**B**.**y**||(**A**.**y**==**B**.**y**&&**A**.**x**<**B**.**x**);**

**}**

int top**;**

node **\***newnode**(){**

**if(**si**)return** sta**[**si**--];**

**return** cur**++;**

**}**

node**\*** build**(**int l**,**int r**,**int d**){**

int mid**=(**l**+**r**)>>**1**;**D**=**d**;**

nth\_element**(**p**+**l**,**p**+**mid**,**p**+**r**+**1**,**cmp**);**

node **\***t**=**newnode**();**t**->**p**=**p**[**mid**];**

**if(**l**<=**mid**-**1**)**t**->**c**[**0**]=**build**(**l**,**mid**-**1**,**d**^**1**);**

**if(**mid**+**1**<=**r**)**t**->**c**[**1**]=**build**(**mid**+**1**,**r**,**d**^**1**);**

**return** t**->**rz**();**

**}**

void dfs**(**node **\*&**t**){**

**if(**t**->**c**[**0**])**dfs**(**t**->**c**[**0**]);**

p**[++**top**]=**t**->**p**;**

**if(**t**->**c**[**1**])**dfs**(**t**->**c**[**1**]);**

sta**[++**si**]=**t**;\***t**=**node**();**

//delete t;

**}**

node**\*** rebuild**(**node **\*&**t**){**

**if(!**t**)return** 0**;**

top**=**0**;**dfs**(**t**);**

**return** build**(**1**,**top**,**0**);**

**}**

#define siz(x) (x?x->siz:0)

void Add**(**node **\*&**t**,**point p**){**

D**^=**1**;**

**if(!**t**){**t**=**newnode**(),**t**->**p**=**p**;**t**->**rz**();return;}**

**if(**t**->**p**==**p**){**t**->**p**.**val**+=**p**.**val**;**t**->**rz**();return;}**

**if(**p**[**D**]<**t**->**p**[**D**])**Add**(**t**->**c**[**0**],**p**);**

**else** Add**(**t**->**c**[**1**],**p**);**t**->**rz**();**

**if(**max**(**siz**(**t**->**c**[**0**]),**siz**(**t**->**c**[**1**]))>**0.8**\***t**->**siz**)**

re**=**t**;**

**}**

int ans**;**

void Q**(**const node **\***t**,**const rec **&**R**){**

**if(**Out**(**t**->**R**,**R**))return** **;**

**if(**In**(**t**->**R**,**R**)){**ans**+=**t**->**sum**;return;}**

**if(**t**->**p**\***R**)**ans**+=**t**->**p**.**val**;**

**if(**t**->**c**[**0**])**Q**(**t**->**c**[**0**],**R**);**

**if(**t**->**c**[**1**])**Q**(**t**->**c**[**1**],**R**);**

**}**

int main**(){**

n**=**in**();**

**while(**1**){**

int ty**=**in**();if(**ty**==**3**)break;**

**if(**ty**==**1**){**

int x**=**in**()^**lans**,**y**=**in**()^**lans**,**A**=**in**()^**lans**;**

re**=**0**;**D**=**1**;**

Add**(**root**,(**point**){**x**,**y**,**A**});**

**if(**re**)**re**=**rebuild**(**re**);**

**}else{**

rec R**;**

R**.**x**[**0**]=**in**()^**lans**;**R**.**y**[**0**]=**in**()^**lans**;**

R**.**x**[**1**]=**in**()^**lans**;**R**.**y**[**1**]=**in**()^**lans**;**

ans**=**0**;if(**root**)**Q**(**root**,**R**);**

printf**(**"%d\n"**,**ans**);**

lans**=**ans**;**

**}**

**}**

**return** 0**;**

**}**

### 虚树

int n**,**m**;**

vector**<**pair**<**int**,**int**>** **>**G**[**maxn**];**

vector**<**int**>**Ge**[**maxn**];**

LL mn**[**maxn**];**

int fa**[**maxn**][**BIT**],**dep**[**maxn**],**z**,**dfn**[**maxn**],**w**[**maxn**],**vis**[**maxn**];**

void dfs**(**int u**){**

dfn**[**u**]=++**z**;**

**for(**int i**=**1**;**i**<**BIT**;**i**++)**fa**[**u**][**i**]=**fa**[**fa**[**u**][**i**-**1**]][**i**-**1**];**

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

int v**=**G**[**u**][**i**].**first**,**w**=**G**[**u**][**i**].**second**;**

**if(**fa**[**u**][**0**]==**v**)continue;**

fa**[**v**][**0**]=**u**;**dep**[**v**]=**dep**[**u**]+**1**;**mn**[**v**]=**min**(**mn**[**u**],(**LL**)**w**);**

dfs**(**v**);**

**}**

**}**

LL dp**(**int u**,**int fa**){**

LL tmp**=**0**;**

**for(**int v**,**i**=**0**;**i**<**Ge**[**u**].**size**();**i**++){**

**if((**v**=**Ge**[**u**][**i**])!=**fa**)**

tmp**+=**dp**(**v**,**u**);**

**}**Ge**[**u**].**clear**();**

**return** tmp**?**min**(**tmp**,(**LL**)**mn**[**u**]):**mn**[**u**];**

**}**

bool cmp**(**int x**,**int y**){return** dfn**[**x**]<**dfn**[**y**];}**

int h**[**maxn**];**

int lca**(**int u**,**int v**){**

**if(**dep**[**u**]<**dep**[**v**])**swap**(**u**,**v**);**

int d**=**dep**[**u**]-**dep**[**v**];**

**for(**int i**=**BIT**-**1**;**i**>=**0**;**i**--)if(**d**&(**1**<<**i**))**u**=**fa**[**u**][**i**];**

**if(**u**==**v**)return** u**;**

**for(**int i**=**BIT**-**1**;**i**>=**0**;**i**--)if(**fa**[**u**][**i**]!=**fa**[**v**][**i**])**u**=**fa**[**u**][**i**],**v**=**fa**[**v**][**i**];**

**return** fa**[**u**][**0**];**

**}**

int sta**[**maxn**],**top**;**

void add**(**int u**,**int v**){**

**if(**u**==**v**)return;**

// cerr<<u<<" "<<v<<endl;

Ge**[**u**].**push\_back**(**v**);**

Ge**[**v**].**push\_back**(**u**);**

**}**

int main**(){**

scanf**(**"%d"**,&**n**);**

**for(**int i**=**1**;**i**<**n**;**i**++){**

int u**,**v**,**w**;**scanf**(**"%d%d%d"**,&**u**,&**v**,&**w**);**

G**[**u**].**push\_back**(**make\_pair**(**v**,**w**));**

G**[**v**].**push\_back**(**make\_pair**(**u**,**w**));**

**}**mn**[**1**]=**1e11**;**dfs**(**1**);**

scanf**(**"%d"**,&**m**);**m**++;**

**while(**m**-->**1**){**

int k**;**scanf**(**"%d"**,&**k**);**z**=**1**;**

**for(**int i**=**1**;**i**<=**k**;**i**++)**scanf**(**"%d"**,&**h**[**i**]);**

sort**(**h**+**1**,**h**+**1**+**k**,**cmp**);**z**=**1**;**

**for(**int i**=**2**;**i**<=**k**;**i**++)if(**h**[**z**]!=**lca**(**h**[**z**],**h**[**i**]))**h**[++**z**]=**h**[**i**];**

top**=**0**;**sta**[++**top**]=**1**;**k**=**z**;**

**for(**int i**=**1**;**i**<=**k**;**i**++){**

int u**=**h**[**i**],**f**=**lca**(**u**,**sta**[**top**]);**

**for(;;){**

**if(**dep**[**f**]>=**dep**[**sta**[**top**-**1**]]){**

add**(**f**,**sta**[**top**--]);**

**if(**sta**[**top**]!=**f**)**sta**[++**top**]=**f**;**

**break;**

**}**add**(**sta**[**top**-**1**],**sta**[**top**]);**top**--;**

**}if(**sta**[**top**]!=**u**)**sta**[++**top**]=**u**;**

**}while(--**top**)**add**(**sta**[**top**],**sta**[**top**+**1**]);**

printf**(**"%lld\n"**,**dp**(**1**,**0**));**

**}**

**return** 0**;**

**}**

# 

# 图论

### 割点

int n**,**m**,**cnt**,**T**;**

vector**<**int**>**G**[**maxn**];**

int vis**[**maxn**];**

int iscut**[**maxn**],**dfn**[**maxn**],**low**[**maxn**],**tot**;**

void dfs**(**int u**,**int fa**){**

low**[**u**]=**dfn**[**u**]=++**tot**;**

int ch**=**0**;**

**for(**int v**,**i**=**0**;**i**<**G**[**u**].**size**();**i**++){**

**if(!**dfn**[**v**=**G**[**u**][**i**]]){**

ch**++;**

dfs**(**v**,**u**);**

low**[**u**]=**min**(**low**[**u**],**low**[**v**]);**

**if(**low**[**v**]>=**dfn**[**u**])**iscut**[**u**]=**1**;**

**}else{**

**if(**v**!=**fa**)**

low**[**u**]=**min**(**low**[**u**],**dfn**[**v**]);**

**}**

**}**

**if(!**fa**&&**ch**==**1**)**iscut**[**u**]=**0**;**

**}**

int main**(){**

// freopen("cut.in","r",stdin);

scanf**(**"%d%d"**,&**n**,&**m**);**

**for(**int i**=**1**;**i**<=**m**;**i**++){**

int u**,**v**;**scanf**(**"%d%d"**,&**u**,&**v**);**

G**[**u**].**push\_back**(**v**);**

G**[**v**].**push\_back**(**u**);**

**}**

**for(**int i**=**1**;**i**<=**n**;**i**++)if(!**dfn**[**i**])**

dfs**(**i**,**0**);**

**for(**int i**=**1**;**i**<=**n**;**i**++)if(**iscut**[**i**])**printf**(**"%d\n"**,**i**);**

**return** 0**;**

**}**

### Dinic

struct edge**{**int u**,**v**,**cap**,**flow**;};**

vector**<**edge**>**edges**;**

vector**<**int**>**G**[**maxn**];**

int s**,**t**;**

int cur**[**maxn**],**d**[**maxn**];**

void add**(**int u**,**int v**,**int cap**){**

edges**.**push\_back**((**edge**){**u**,**v**,**cap**,**0**});**

G**[**u**].**push\_back**(**edges**.**size**()-**1**);**

edges**.**push\_back**((**edge**){**v**,**u**,**0**,**0**});**

G**[**v**].**push\_back**(**edges**.**size**()-**1**);**

**}**

bool bfs**(){**

static int vis**[**maxn**];**

memset**(**vis**,**0**,sizeof** vis**);**vis**[**s**]=**1**;**

queue**<**int**>**q**;**q**.**push**(**s**);**d**[**s**]=**0**;**

**while(!**q**.**empty**()){**

int u**=**q**.**front**();**q**.**pop**();**

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

edge e**=**edges**[**G**[**u**][**i**]];if(**vis**[**e**.**v**]||**e**.**cap**==**e**.**flow**)continue;**

d**[**e**.**v**]=**d**[**u**]+**1**;**vis**[**e**.**v**]=**1**;**q**.**push**(**e**.**v**);**

**}**

**}return** vis**[**t**];**

**}**

int dfs**(**int u**,**int a**){**

**if(**u**==**t**||!**a**)return** a**;**

int flow**=**0**,**f**;**

**for(**int **&**i**=**cur**[**u**];**i**<**G**[**u**].**size**();**i**++){**

edge e**=**edges**[**G**[**u**][**i**]];**

**if(**d**[**e**.**v**]==**d**[**u**]+**1**&&(**f**=**dfs**(**e**.**v**,**min**(**a**,**e**.**cap**-**e**.**flow**)))>**0**){**

edges**[**G**[**u**][**i**]].**flow**+=**f**;**

edges**[**G**[**u**][**i**]^**1**].**flow**-=**f**;**

flow**+=**f**;**a**-=**f**;if(!**a**)break;**

**}**

**}return** flow**;**

**}**

int dinic**(){**

int flow**=**0**,**x**;**

**while(**bfs**()){**

memset**(**cur**,**0**,sizeof** cur**);**

**while(**x**=**dfs**(**s**,**INT\_MAX**)){**

flow**+=**x**;**

memset**(**cur**,**0**,sizeof** cur**);**

**}**

**}return** flow**;**

**}**

### HK

int n**,**m**;**

int lin**[**maxn**];**

vector**<**int**>**G**[**maxn**],**Ge**[**maxn**];**

int d**[**maxn**];**

bool bfs**(){**

memset**(**d**,**0**,sizeof** d**);**

queue**<**int**>**q**;**

**for(**int i**=**1**;**i**<=**n**;**i**++)if(!**lin**[**i**])**q**.**push**(**i**);**

int f**=**0**;**

**while(!**q**.**empty**()){**

int u**=**q**.**front**();**q**.**pop**();**

**if(**u**>**n**){**

int v**=**lin**[**u**];**

q**.**push**(**v**);**d**[**v**]=**d**[**u**]+**1**;**

**}else** **for(**int i**=**0**;**i**<**G**[**u**].**size**();**i**++){**

int v**=**G**[**u**][**i**];**

**if(**d**[**v**])continue;**

d**[**v**]=**d**[**u**]+**1**;**

**if(**lin**[**v**])**q**.**push**(**v**);**

**else** f**=**1**;**

**}**

**}return** f**;**

**}**

int find**(**int u**){**

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

int v**=**G**[**u**][**i**];**

**if(**d**[**u**]+**1**!=**d**[**v**])continue;**

d**[**v**]=-**1**;**

**if(!**lin**[**v**]||**find**(**lin**[**v**])){**

lin**[**v**]=**u**;**lin**[**u**]=**v**;return** 1**;**

**}**

**}return** 0**;**

**}**

int dfn**[**maxn**],**low**[**maxn**],**bel**[**maxn**],**z**,**B**,**ins**[**maxn**];**

stack**<**int**>**S**;**

void tarjan**(**int u**){**

low**[**u**]=**dfn**[**u**]=++**z**;**ins**[**u**]=**1**;**S**.**push**(**u**);**

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

int v**=**Ge**[**u**][**i**];**

**if(!**dfn**[**v**]){**

tarjan**(**v**);**

low**[**u**]=**min**(**low**[**u**],**low**[**v**]);**

**}else** **if(**ins**[**v**])**low**[**u**]=**min**(**low**[**u**],**dfn**[**v**]);**

**}if(**low**[**u**]==**dfn**[**u**]){**

int v**;**B**++;**

**do{**

v**=**S**.**top**();**S**.**pop**();**ins**[**v**]=**0**;**

bel**[**v**]=**B**;**

**}while(**u**!=**v**);**

**}**

**}**

pair**<**int**,**int**>** Q**[**int**(**6e5**+**5**)];**

int main**(){**

n**=**getn**();**m**=**getn**();**

**for(**int i**=**1**;**i**<=**m**;**i**++){**

int u**=**getn**(),**v**=**getn**();**

Q**[**i**].**first**=**u**;**

Q**[**i**].**second**=**v**+**n**;**

G**[**u**].**push\_back**(**v**+**n**);**

**}while(**bfs**())**

**for(**int i**=**1**;**i**<=**n**;**i**++)if(!**lin**[**i**])**

find**(**i**);**

**for(**int i**=**1**;**i**<=**n**\***2**;**i**++)for(**int j**=**0**;**j**<**G**[**i**].**size**();**j**++){**

int u**=**i**,**v**=**G**[**i**][**j**];**

**if(**lin**[**u**]==**v**)**Ge**[**v**].**push\_back**(**u**);**

**else** Ge**[**u**].**push\_back**(**v**);**

**}for(**int i**=**1**;**i**<=**n**\***2**;**i**++)if(!**dfn**[**i**])**tarjan**(**i**);**

**for(**int i**=**1**;**i**<=**m**;**i**++){**

int u**=**Q**[**i**].**first**,**v**=**Q**[**i**].**second**;**

**if(**bel**[**u**]!=**bel**[**v**]){**

**if(**lin**[**u**]==**v**)**putchar**(**'1'**);**

**else** putchar**(**'2'**);**

**}else** putchar**(**'0'**);**

**}**puts**(**""**);**

**return** 0**;**

**}**

### 匈牙利

vector**<**int**>**G**[**maxn**];**

int Link**[**maxn**],**vis**[**maxn**],**T**;**

bool find**(**int x**){**

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

int v**=**G**[**x**][**i**];**

**if(**vis**[**v**]==**T**)continue;**

vis**[**v**]=**T**;**

**if(!**Link**[**v**]||**find**(**Link**[**v**])){**

Link**[**v**]=**x**;**

**return** 1**;**

**}**

**}return** 0**;**

**}**

int Hungarian**(**int n**){**

int ans**=**0**;**

memset**(**Link**,**0**,sizeof** Link**);**

**for(**int i**=**1**;**i**<=**n**;**i**++){**

T**++;**

ans**+=**find**(**i**);**

**}return** ans**;**

**}**

### 费用流

int s**,**t**;**

struct edge**{**int u**,**v**,**ca**,**fl**,**co**;};**

vector**<**edge**>**E**;**

vector**<**int**>**G**[**maxn**];**

void add**(**int u**,**int v**,**int ca**,**int co**){**

E**.**push\_back**((**edge**){**u**,**v**,**ca**,**0**,**co**});**

G**[**u**].**push\_back**(**E**.**size**()-**1**);**

E**.**push\_back**((**edge**){**v**,**u**,**0**,**0**,-**co**});**

G**[**v**].**push\_back**(**E**.**size**()-**1**);**

**}**

int cost**,**flow**=**0**;**

bool spfa**(){**

static int vis**[**maxn**],**d**[**maxn**],**a**[**maxn**],**pre**[**maxn**];**

memset**(**d**,**0x7f**,sizeof** d**);**int B**=**d**[**0**];**a**[**s**]=**INT\_MAX**;**

queue**<**int**>**q**;**q**.**push**(**s**);**vis**[**s**]=**1**;**d**[**s**]=**0**;**

**while(!**q**.**empty**()){**

int u**=**q**.**front**();**q**.**pop**();**vis**[**u**]=**0**;**

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

edge e**=**E**[**G**[**u**][**i**]];**

**if(**e**.**ca**==**e**.**fl**||**d**[**e**.**v**]<=**d**[**u**]+**e**.**co**)continue;**

a**[**e**.**v**]=**min**(**e**.**ca**-**e**.**fl**,**a**[**u**]);**pre**[**e**.**v**]=**G**[**u**][**i**];**

d**[**e**.**v**]=**d**[**u**]+**e**.**co**;if(!**vis**[**e**.**v**])**q**.**push**(**e**.**v**),**vis**[**e**.**v**]=**1**;**

**}**

**}**

**if(**d**[**t**]==**B**)return** **false;**

flow**+=**a**[**t**];**

cost**+=**a**[**t**]\***d**[**t**];**

int u**=**t**;**

**while(**u**!=**s**){**

E**[**pre**[**u**]].**fl**+=**a**[**t**];**

E**[**pre**[**u**]^**1**].**fl**-=**a**[**t**];**

u**=**E**[**pre**[**u**]].**u**;**

**}return** **true;**

**}**

int main**(){**

int n**;**scanf**(**"%d"**,&**n**);**

s**=**0**;**t**=**n**+**1**;**

**for(**int i**=**1**;**i**<=**n**;**i**++){**

int m**;**scanf**(**"%d"**,&**m**);**

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

int v**,**z**;**scanf**(**"%d%d"**,&**v**,&**z**);**

add**(**s**,**v**,**1**,**z**);**

add**(**i**,**v**,**1e9**,**z**);**

**}**add**(**i**,**t**,**m**,**0**);**

**if(**i**!=**1**)**add**(**i**,**1**,**1e9**,**0**);**

**}while(**spfa**());**

cout**<<**cost**<<**endl**;**

**return** 0**;**

**}**

### SCC

void Tarjan**(**int u**){**

dfs**[**u**]=**low**[**u**]=++**tot**;**

ins**[**u**]=**1**;**

S**.**push**(**u**);**

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

edge e**=**edges**[**G**[**u**][**i**]];**

**if(!**dfs**[**e**.**v**]){**

Tarjan**(**e**.**v**);**

low**[**u**]=**min**(**low**[**u**],**low**[**e**.**v**]);**

**}else** **if(**ins**[**e**.**v**]){**

low**[**u**]=**min**(**low**[**u**],**dfs**[**e**.**v**]);**

**}**

**}**

**if(**dfs**[**u**]==**low**[**u**]){**

int v**;**

Scnt**++;**

**do{**

v**=**S**.**top**();**S**.**pop**();**

Belong**[**v**]=**Scnt**;**

ins**[**v**]=**0**;**

**}while(**u**!=**v**);**

**}**

**}**

### KM

int N**,** flag**;**

int w**[**MAX\_N**][**MAX\_N**];**

int fx**[**MAX\_N**],** fy**[**MAX\_N**],** lx**[**MAX\_N**],** ly**[**MAX\_N**],** slk**[**MAX\_N**],** mat**[**MAX\_N**];**

int DFS**(**int x**)** **{**

fx**[**x**]** **=** flag**;**

**for(**int i **=** 1**;** i **<=** N**;** **++** i**)**

**if** **(**lx**[**x**]** **+** ly**[**i**]** **!=** w**[**x**][**i**])**

slk**[**i**]** **=** min**(**slk**[**i**],** lx**[**x**]** **+** ly**[**i**]** **-** w**[**x**][**i**]);**

**else** **if** **(**fy**[**i**]** **!=** flag**)** **{**

fy**[**i**]** **=** flag**;**

**if** **(**mat**[**i**]** **<** 0 **||** DFS**(**mat**[**i**]))** **{**

mat**[**i**]** **=** x**;**

**return** **true;**

**}**

**}**

**return** **false;**

**}**

int KM**()** **{**

**for(**int i **=** 1**;** i **<=** N**;** **++** i**)** **{**

mat**[**i**]** **=** **-**1**;**

fx**[**i**]** **=** 0**;** fy**[**i**]** **=** 0**;**

ly**[**i**]** **=** 0**;** lx**[**i**]** **=** **-**INF**;**

**for(**int j **=** 1**;** j **<=** N**;** **++** j**)**

lx**[**i**]** **=** max**(**lx**[**i**],** w**[**i**][**j**]);**

**}**

**for(**int now **=** 1**;** now **<=** N**;** **++** now**)** **{**

**++** flag**;** **for(**int i **=** 1**;** i **<=** N**;** **++** i**)** slk**[**i**]** **=** INF**;**

**while** **(!** DFS**(**now**))** **{**

int p**(**INF**);** **for(**int i **=** 1**;** i **<=** N**;** **++** i**)**

**if** **(**fy**[**i**]** **!=** flag**)** p **=** min**(**p**,** slk**[**i**]);**

**for(**int i **=** 1**;** i **<=** N**;** **++** i**)** **{**

**if** **(**fx**[**i**]** **==** flag**)** lx**[**i**]** **-=** p**;**

**if** **(**fy**[**i**]** **==** flag**)** ly**[**i**]** **+=** p**;**

slk**[**i**]** **=** INF**;**

**}**

**++** flag**;**

**}**

**}**

long long ans **=** 0**;**

**for(**int i **=** 1**;** i **<=** N**;** **++** i**)** ans **+=** lx**[**i**],** ans **+=** ly**[**i**];**

**return** ans**;**

**}**

### 最大团

//Int g[][] 为图的邻接矩阵 标号由 1 至 n

//MC(V) 表示点集 V 的最大团 令 Si={vi, vi+1, ..., vn}, mc[i] 表示 MC(Si)

// 倒着算 mc[i], 那么显然 MC(V)=mc[1] 此外有 mc[i]=mc[i+1] or mc[i]=mc[i+1]+1

void dfs**(**int size**){**

**if** **(**len**[**size**]==**0**)** **{**

**if** **(**size**>**ans**)** ans**=**size**,** found**=true;** **return;**

**}** **for** **(**int k**=**0**,**i**,**j**;** k**<**len**[**size**]** **&&** **!**found**;** **++**k**)** **{**

**if** **(**size**+**len**[**size**]-**k**<=**ans**)** **break;**

i**=**list**[**size**][**k**];** **if** **(**size**+**mc**[**i**]<=**ans**)** **break;**

**for** **(**j**=**k**+**1**,** len**[**size**+**1**]=**0**;** j**<**len**[**size**];** **++**j**)**

**if** **(**g**[**i**][**list**[**size**][**j**]])** list**[**size**+**1**][**len**[**size**+**1**]++]=**list**[**size**][**j**];**

dfs**(**size**+**1**);**

**}}**

void work**(){**

mc**[**n**]=**ans**=**1**;**

**for** **(**int i**=**n**-**1**;** i**;** **--**i**)** **{**

found**=false;** len**[**1**]=**0**;**

**for** **(**int j**=**i**+**1**;** j**<=**n**;** **++**j**)** **if** **(**g**[**i**][**j**])** list**[**1**][**len**[**1**]++]=**j**;**

dfs**(**1**);** mc**[**i**]=**ans**;**

**}}**

# 字符串

### AC自动机

#define idx(c) (c-'a')

int n**,**len**,**anss**[**maxn**];**

char s**[**maxlen**],**str**[**maxlen**];**

struct node**{**

vector**<**int**>**val**;**

node **\***go**[**27**],\***fail**,\***last**;**

node**(**node **\***C**=**0**){**

**for(**int i**=**0**;**i**<**27**;**i**++)**go**[**i**]=**C**;**

fail**=**last**=**C**;**

**}**

**}\***root**,\***Null**;**

void insert**(**const char **\***s**,**int x**){**

node **\***u**=**root**;**int len**=**strlen**(**s**);**

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

int v**=**idx**(**s**[**i**]);**

**if(**u**->**go**[**v**]==**Null**)**u**->**go**[**v**]=new** node**(**Null**);**

u**=**u**->**go**[**v**];**

**}**u**->**val**.**push\_back**(**x**);**

**}**

void get\_fail**(){**

queue**<**node**\*>**q**;**

**for(**int i**=**0**;**i**<**27**;**i**++)if(**root**->**go**[**i**]!=**Null**)**

q**.**push**(**root**->**go**[**i**]),**root**->**go**[**i**]->**fail**=**root**->**go**[**i**]->**last**=**root**;**

**while(!**q**.**empty**()){**

node **\***u**=**q**.**front**(),\***v**;**q**.**pop**();**

**for(**int i**=**0**;**i**<**27**;**i**++)if((**v**=**u**->**go**[**i**])!=**Null**){**

q**.**push**(**v**);**node **\***j**=**u**->**fail**;**

**while(**j**!=**Null**&&**j**->**go**[**i**]==**Null**)**j**=**j**->**fail**;**

v**->**fail**=**j**->**go**[**i**];**

v**->**last**=**v**->**fail**->**val**.**size**()?**v**->**fail**:**v**->**fail**->**last**;**

**}**

**}**

**}**

void calc**(**node **\***j**){**

**for(**int i**=**0**;**i**<**j**->**val**.**size**();**i**++)**

anss**[**j**->**val**[**i**]]++;**

**if(**j**->**last**!=**Null**)**calc**(**j**->**last**);**

**}**

void find**(){**

node **\***j**=**root**;**

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

int v**=**idx**(**str**[**i**]);**

**while(**j**!=**Null**&&**j**->**go**[**v**]==**Null**)**j**=**j**->**fail**;**

j**=**j**->**go**[**v**];**

**if(**j**->**val**.**size**())**calc**(**j**);**

**else** **if(**j**->**last**!=**Null**)**calc**(**j**->**last**);**

**}**

**}**

int main**(){**

Null**=new** node**();**root**=**Null**;**

**for(**int i**=**0**;**i**<**27**;**i**++)**Null**->**go**[**i**]=**Null**;**

Null**->**fail**=**Null**->**last**=**Null**;**

scanf**(**"%d"**,&**n**);**

**for(**int i**=**1**;**i**<=**n**;**i**++){**

scanf**(**"%s"**,**s**);**copy**(**s**,**s**+**strlen**(**s**),**str**+**len**);**len**+=**strlen**(**s**);**str**[**len**++]=**'{'**;**

insert**(**s**,**i**);**

**}**get\_fail**();**

find**();**

**for(**int i**=**1**;**i**<=**n**;++**i**)**printf**(**"%d\n"**,**anss**[**i**]);**

**return** 0**;**

**}**

### 后缀数组

#define fst first

#define sec second

**typedef** pair**<**int**,**int**>** pi**;**

pi tmp**[**maxn**];**

int sa**[**maxn**],**rank**[**maxn**],**height**[**maxn**],**n**,**len**,**len2**;**

char s**[**maxn**];**

int sum**[**maxn**],**lef**[**maxn**],**rig**[**maxn**];**

void radix\_sort**(**int n**,**pi **\***a**,**int **\***rank**,**int **\***sa**){**

int m**=**max**(**n**,**27**);**

memset**(**sum**,**0**,sizeof(**int**)\*(**m**+**1**));**

**for(**int i**=**1**;**i**<=**n**;**i**++)**sum**[**a**[**i**].**sec**]++;**

**for(**int i**=**1**;**i**<=**m**;**i**++)**sum**[**i**]+=**sum**[**i**-**1**];**

**for(**int i**=**n**;**i**>=**1**;**i**--)**rank**[**i**]=**sum**[**a**[**i**].**sec**]--;**

**for(**int i**=**1**;**i**<=**n**;**i**++)**sa**[**rank**[**i**]]=**i**;**

memset**(**sum**,**0**,sizeof(**int**)\*(**m**+**1**));**

**for(**int i**=**1**;**i**<=**n**;**i**++)**sum**[**a**[**i**].**fst**]++;**

**for(**int i**=**1**;**i**<=**m**;**i**++)**sum**[**i**]+=**sum**[**i**-**1**];**

**for(**int i**=**n**;**i**>=**1**;**i**--)**rank**[**sa**[**i**]]=**sum**[**a**[**sa**[**i**]].**fst**]--;**

**for(**int i**=**1**;**i**<=**n**;**i**++)**sa**[**rank**[**i**]]=**i**;**rank**[**sa**[**1**]]=**1**;**

**for(**int i**=**2**;**i**<=**n**;**i**++)**rank**[**sa**[**i**]]=**rank**[**sa**[**i**-**1**]]+(**a**[**sa**[**i**]]!=**a**[**sa**[**i**-**1**]]);**

**}**

void sa\_init**(**int n**){**

**for(**int i**=**1**;**i**<=**n**;**i**++)**tmp**[**i**].**fst**=**s**[**i**]-**'a'**+**1**,**tmp**[**i**].**sec**=**0**;**

radix\_sort**(**n**,**tmp**,**rank**,**sa**);**int mx**=**0**;**

**for(**int j**=**1**;**j**<=**n**;**j**<<=**1**){**

**for(**int i**=**1**;**i**<=**n**;**i**++)**tmp**[**i**].**fst**=**rank**[**i**],**tmp**[**i**].**sec**=**i**+**j**<=**n**?**rank**[**i**+**j**]:**0**,**mx**=**max**(**mx**,**rank**[**i**]);**

**if(**mx**==**n**)break;**radix\_sort**(**n**,**tmp**,**rank**,**sa**);**

**}for(**int i**=**1**,**j**,**k**=**0**;**i**<=**n**;**i**++){**

k**-=!!**k**;**j**=**sa**[**rank**[**i**]-**1**];**

**while(**s**[**i**+**k**]==**s**[**j**+**k**])**k**++;**

height**[**rank**[**i**]]=**k**;**

**}**

**}**

int Qsum**(**int L**,**int R**){**

int l**=**1**,**r**=**len**,**ans**=**0**;**

**while(**l**<**r**){**

int mid**=(**l**+**r**)>>**1**;**

int res**=**0**;**

**for(**int i**=**L**;**i**<=**R**;**i**++){**

**if(**sa**[**mid**]+**i**-**L**>**len**){**res**=**1**;break;}**

**if(**s**[**i**]<**s**[**sa**[**mid**]+**i**-**L**]){**res**=-**1**;break;}**

**if(**s**[**i**]>**s**[**sa**[**mid**]+**i**-**L**]){**res**=**1**;break;}**

**}if(**res**>=**0**)**l**=**mid**+**1**;**

**else** r**=**mid**;**

**}**ans**=**l**;**

l**=**1**,**r**=**len**;**

**while(**l**<**r**){**

int mid**=(**l**+**r**)>>**1**;**

int res**=**0**;**

**for(**int i**=**L**;**i**<=**R**;**i**++){**

**if(**sa**[**mid**]+**i**-**L**>**len**){**res**=**1**;break;}**

**if(**s**[**i**]<**s**[**sa**[**mid**]+**i**-**L**]){**res**=-**1**;break;}**

**if(**s**[**i**]>**s**[**sa**[**mid**]+**i**-**L**]){**res**=**1**;break;}**

**}if(**res**>**0**)**l**=**mid**+**1**;**

**else** r**=**mid**;**

**}return** ans**-**l**;**

**}**

int main**(){**

scanf**(**"%d"**,&**n**);**

**for(**int i**=**1**;**i**<=**n**;**i**++){**

lef**[**i**]=**len**+**1**;**

char c**=**getchar**();**

**while(!**isalpha**(**c**))**c**=**getchar**();**

**while(**isalpha**(**c**))**s**[++**len**]=**c**,**c**=**getchar**();**

rig**[**i**]=**len**;**s**[++**len**]=**'{'**;**

**}**sa\_init**(**len**);**

**for(**int i**=**1**;**i**<=**n**;**i**++)**

printf**(**"%d\n"**,**Qsum**(**lef**[**i**],**rig**[**i**]));**

**return** 0**;**

**}**

### 后缀自动机

int n**,**len**,**r**[**201**],**cnt**,**cur**=**1**;**

char s**[**maxn**];**

struct node**{**

int val**,**ans**,**id**;**

node **\***go**[**27**],\***f**;**

vector**<**node**\*>**rp**;**

node**(){**val**=**ans**=**0**;**memset**(**go**,**0**,sizeof** go**);**f**=**0**;}**

**}**pool**[**maxn**\***2**],\***root**=**pool**,\***last**=**root**;**

node **\***newnode**(){**

node **\***x**=&**pool**[**cur**++];**

x**->**id**=++**cnt**;**

**return** x**;**

**}**

void add**(**int w**){**

node **\***p**=**last**;**

node **\***np**=**newnode**();**np**->**val**=**p**->**val**+**1**;**

**while(**p**&&!**p**->**go**[**w**])**p**->**go**[**w**]=**np**,**p**=**p**->**f**;**

**if(!**p**)**np**->**f**=**root**;**

**else{**

node **\***q**=**p**->**go**[**w**];**

**if(**p**->**val**+**1**==**q**->**val**)**np**->**f**=**q**;**

**else{**

node **\***nq**=**newnode**();**nq**->**val**=**p**->**val**+**1**;**

**for(**int i**=**0**;**i**<**27**;**i**++)**nq**->**go**[**i**]=**q**->**go**[**i**];**

nq**->**f**=**q**->**f**;**

q**->**f**=**np**->**f**=**nq**;**

**while(**p**&&**p**->**go**[**w**]==**q**)**p**->**go**[**w**]=**nq**,**p**=**p**->**f**;**

**}**

**}**last**=**np**;**

**}**

vector**<**node**\*>**vec**;**

bool cmp**(**node **\***a**,**node **\***b**){return** a**->**val**>**b**->**val**;}**

int Q**(**const char **\***s**,**int l**,**int r**){**

node **\***u**=**root**;**

**for(**int i**=**l**;**i**<**r**;**i**++){**

int v**=**s**[**i**]-**'a'**;**

u**=**u**->**go**[**v**];**

**}return** u**->**ans**;**

**}**

int main**(){**

scanf**(**"%d"**,&**n**);**

**for(**int i**=**1**;**i**<=**n**;**i**++){**

scanf**(**"%s"**,**s**+**len**);**

**for(**int j**=**len**,**l**=**strlen**(**s**+**len**);**j**<**l**+**len**;**j**++)**

add**(**s**[**j**]-**'a'**);**

add**(**26**);**

len**+=**strlen**(**s**+**len**);**

s**[**len**++]=**26**+**'a'**;**r**[**i**]=**len**;**

**}**

node **\***u**=**root**;**

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

u**=**u**->**go**[**s**[**i**]-**'a'**],**u**->**ans**=**1**;**

**for(**int i**=**0**;**i**<**cnt**;**i**++)**vec**.**push\_back**(**pool**+**i**);**

sort**(**vec**.**begin**(),**vec**.**end**(),**cmp**);**

**for(**int i**=**0**;**i**<**vec**.**size**();**i**++)if(**vec**[**i**]->**f**)**

vec**[**i**]->**f**->**ans**+=**vec**[**i**]->**ans**;**

**for(**int i**=**1**;**i**<=**n**;**i**++)**

printf**(**"%d\n"**,**Q**(**s**,**r**[**i**-**1**],**r**[**i**]-**1**));**

**return** 0**;**

**}**

### KMP

int next**[**maxn**],**slen**,**tlen**;**

char s**[**maxn**],**t**[**maxn**];**

int kmp**(){**

int ans**=**0**;**

**for(**int i**=**2**,**j**=**next**[**1**]=**0**;**i**<=**tlen**;++**i**){**

**while(**j**&&**t**[**i**]!=**t**[**j**+**1**])**j**=**next**[**j**];**

next**[**i**]=**j**+=**t**[**i**]==**t**[**j**+**1**];**

**}for(**int i**=**1**,**j**=**0**;**i**<=**slen**;**i**++){**

**while(**j**&&**s**[**i**]!=**t**[**j**+**1**])**j**=**next**[**j**];**

j**+=**s**[**i**]==**t**[**j**+**1**];**

**if(**j**==**tlen**)**j**=**next**[**j**],**ans**++;**

**}return** ans**;**

**}**

int main**(){**

int T**;**scanf**(**"%d"**,&**T**);**

**while(**T**--){**

scanf**(**"%s%s"**,**t**+**1**,**s**+**1**);**

slen**=**strlen**(**s**+**1**);**

tlen**=**strlen**(**t**+**1**);**

printf**(**"%d\n"**,**kmp**());**

**}**

**return** 0**;**

**}**

### 扩展SAM

#include<bits/stdc++.h>

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

const int maxn**=**2e5**+**5**;**

int son**[**maxn**][**26**],**len**[**maxn**],**fa**[**maxn**];**

int root**=**1**,**last**=**1**,**tot**=**1**;**

int newnode**(**int x**){**

len**[++**tot**]=**x**;**

**return** tot**;**

**}**

void add**(**int w**){**

**if(**son**[**last**][**w**]){**

int p**=**last**;**

int q**=**son**[**last**][**w**];**

**if(**len**[**p**]+**1**==**len**[**q**])**last**=**q**;**

**else{**

int nq**=**newnode**(**len**[**p**]+**1**);**

memcpy**(**son**[**nq**],**son**[**q**],sizeof** son**[**nq**]);**

fa**[**nq**]=**fa**[**q**];**fa**[**q**]=**nq**;**

**while(**p**&&**son**[**p**][**w**]==**q**)**son**[**p**][**w**]=**nq**,**p**=**fa**[**p**];**

last**=**nq**;**

**}**

**}else{**

int p**=**last**;**

int np**=**newnode**(**len**[**p**]+**1**);**

**while(**p**&&!**son**[**p**][**w**])**son**[**p**][**w**]=**np**,**p**=**fa**[**p**];**

**if(!**p**)**fa**[**np**]=**root**;**

**else{**

int q**=**son**[**p**][**w**];**

**if(**len**[**p**]+**1**==**len**[**q**])**fa**[**np**]=**q**;**

**else{**

int nq**=**newnode**(**len**[**p**]+**1**);**

memcpy**(**son**[**nq**],**son**[**q**],sizeof** son**[**nq**]);**

fa**[**nq**]=**fa**[**q**];**

fa**[**q**]=**fa**[**np**]=**nq**;**

**while(**p**&&**son**[**p**][**w**]==**q**)**son**[**p**][**w**]=**nq**,**p**=**fa**[**p**];**

**}**

**}**last**=**np**;**

**}**

**}**

int n**,**K**,**now**,**vis**[**maxn**],**cnt**[**maxn**],**f**[**maxn**];**

string s**[**int**(**1e5**+**1**)];**

char str**[**maxn**];**

void make**(**int u**){**

**while(**u**&&**vis**[**u**]!=**now**){**

vis**[**u**]=**now**;**

cnt**[**u**]++;**

u**=**fa**[**u**];**

**}**

**}**

void up**(**int u**){**

**if(**u**==**1**||**vis**[**u**])return;**

vis**[**u**]=**1**;**

up**(**fa**[**u**]);**

f**[**u**]+=**f**[**fa**[**u**]];**

**}**

int main**(){**

freopen**(**"bzoj3473.in"**,**"r"**,**stdin**);**

scanf**(**"%d%d"**,&**n**,&**K**);**

**for(**int i**=**1**;**i**<=**n**;**i**++){**

scanf**(**"%s"**,**str**);**s**[**i**]=**string**(**str**);**

**for(**int j**=**0**;**j**<**s**[**i**].**length**();**j**++)**

add**(**s**[**i**][**j**]-**'a'**);**

last**=**root**;**

// deb();

**}**

// deb();

**for(**int i**=**1**;**i**<=**n**;**i**++){**

int u**=**root**;**now**=**i**;**

**for(**int j**=**0**;**j**<**s**[**i**].**length**();**j**++){**

u**=**son**[**u**][**s**[**i**][**j**]-**'a'**];**

make**(**u**);**

**}**

**}**

memset**(**vis**,**0**,sizeof** vis**);**

**for(**int i**=**1**;**i**<=**tot**;**i**++)**f**[**i**]=(**cnt**[**i**]>=**K**)\*(**len**[**i**]-**len**[**fa**[**i**]]);**

**for(**int i**=**1**;**i**<=**tot**;**i**++)**up**(**i**);**

**for(**int i**=**1**;**i**<=**n**;**i**++){**

int u**=**root**;**now**=**i**;**

long long ans**=**0**;**

**for(**int j**=**0**;**j**<**s**[**i**].**length**();**j**++){**

u**=**son**[**u**][**s**[**i**][**j**]-**'a'**];**

ans**+=**f**[**u**];**

**}**

printf**(**"%lld%c"**,**ans**,**" \n"**[**i**==**n**]);**

**}**

**return** 0**;**

**}**

### Manacher

int main**(){**

char c**=**getchar**();**s**[**0**]=**'+'**;**s**[++**n**]=**'#'**;**

**while(**isalpha**(**c**))**str**[++**m**]=**c**,**s**[++**n**]=**c**,**s**[++**n**]=**'#'**,**c**=**getchar**();**s**[++**n**]=**'-'**;**

f**[**1**]=**1**;**

**for(**int i**=**2**,**k**=**1**;**i**<=**n**;**i**++){**

**if(**f**[**k**]+**k**>**i**)**f**[**i**]=**min**(**f**[**k**+**k**-**i**],**f**[**k**]+**k**-**i**);else** f**[**i**]=**1**;**

**while(**s**[**i**+**f**[**i**]]==**s**[**i**-**f**[**i**]])**f**[**i**]++;**

**if(**f**[**k**]+**k**<**f**[**i**]+**i**)**k**=**i**;**

**}**

**return** 0**;**

**}**

### 最小表示法

string sol**(**char **\***s**){**

int n**=**strlen**(**s**);**

int i**=**0**,**j**=**1**,**k**=**0**,**p**;**

**while(**i**<**n**&&**j**<**n**&&**k**<**n**){**

int t**=**s**[(**i**+**k**)%**n**]-**s**[(**j**+**k**)%**n**];**

**if(**t**==**0**)**k**++;**

**else** **if(**t**<**0**)**j**+=**k**+**1**,**k**=**0**;**

**else** i**+=**k**+**1**,**k**=**0**;**

**if(**i**==**j**)**j**++;**

**}**p**=**min**(**i**,**j**);**

string S**;**

**for(**int i**=**p**;**i**<**p**+**n**;**i**++)**S**.**push\_back**(**s**[**i**%**n**]);**

**return** S**;**

**}**

### IBWT

int x**,**n**;**

char s**[**maxn**];**

string IBWT**(**int x**,**char **\***s**){**

static int nxt**[**maxn**],**n**;**

static char t**[**maxn**];**

deque**<**int**>**dq**[**256**];**

strcpy**(**t**+**1**,**s**+**1**);**

sort**(**t**+**1**,**t**+**1**+(**n**=**strlen**(**t**+**1**)));**

**for(**int i**=**1**;**i**<=**n**;**i**++)**

dq**[**s**[**i**]].**push\_back**(**i**);**

**for(**int i**=**1**;**i**<=**n**;**i**++){**

nxt**[**i**]=**dq**[**t**[**i**]].**front**();**

dq**[**t**[**i**]].**pop\_front**();**

**}**

string ans**;**

**for(**int i**=**1**;**i**<=**n**;**i**++){**

x**=**nxt**[**x**];**

ans**.**push\_back**(**s**[**x**]);**

**}return** ans**;**

**}**

# 计算几何

### 2D计算几何基础

struct Point **{**

double x**,**y**;**

Point rotate**(**const double ang**)** **{** // 逆时针旋转 ang 弧度

**return** Point**(**cos**(**ang**)** **\*** x **-** sin**(**ang**)** **\*** y**,** cos**(**ang**)** **\*** y **+** sin**(**ang**)** **\*** x**);**

**}**

Point turn90**()** **{** // 逆时针旋转 90 度

**return** Point**(-**y**,** x**);**

**}**

**};**

struct Line**{**

double x**,**y**;**

**};**

int sign**(**double x**){return** **(**x**>**eps**)-(**x**<-**eps**);}**

Point isLL**(**const Line **&**l1**,** const Line **&**l2**)** **{**

double s1 **=** det**(**l2**.**b **-** l2**.**a**,** l1**.**a **-** l2**.**a**),**

s2 **=** **-**det**(**l2**.**b **-** l2**.**a**,** l1**.**b **-** l2**.**a**);**

**return** **(**l1**.**a **\*** s2 **+** l1**.**b **\*** s1**)** **/** **(**s1 **+** s2**);**

**}**

bool onSeg**(**const Line **&**l**,** const Point **&**p**)** **{** // 点在线段上

**return** sign**(**det**(**p **-** l**.**a**,** l**.**b **-** l**.**a**))** **==** 0 **&&** sign**(**dot**(**p **-** l**.**a**,** p **-** l**.**b**))** **<=** 0**;**

**}**

Point projection**(**const Line **&**l**,** const Point **&**p**)** **{** // 点到直线投影

**return** l**.**a **+** **(**l**.**b **-** l**.**a**)** **\*** **(**dot**(**p **-** l**.**a**,** l**.**b **-** l**.**a**)** **/** **(**l**.**b **-** l**.**a**).**len2**());**

**}**

double disToLine**(**const Line **&**l**,** const Point **&**p**)** **{**

**return** abs**(**det**(**p **-** l**.**a**,** l**.**b **-** l**.**a**)** **/** **(**l**.**b **-** l**.**a**).**len**());**

**}**

double disToSeg**(**const Line **&**l**,** const Point **&**p**)** **{** // 点到线段距离

**return** sign**(**dot**(**p **-** l**.**a**,** l**.**b **-** l**.**a**))** **\*** sign**(**dot**(**p **-** l**.**b**,** l**.**a **-** l**.**b**))** **!=** 1 **?**

disToLine**(**l**,** p**)** **:** min**((**p **-** l**.**a**).**len**(),** **(**p **-** l**.**b**).**len**());**

**}**

Point symmetryPoint**(**const Point a**,** const Point b**)** **{** // 点 b 关于点 a 的中心对称点

**return** a **+** a **-** b**;**

**}**

Point reflection**(**const Line **&**l**,** const Point **&**p**)** **{** // 点关于直线的对称点

**return** symmetryPoint**(**projection**(**l**,** p**),** p**);**

**}**

// 求圆与直线的交点

bool isCL**(**Circle a**,** Line l**,** Point **&**p1**,** Point **&**p2**)** **{**

double x **=** dot**(**l**.**a **-** a**.**o**,** l**.**b **-** l**.**a**),**

y **=** **(**l**.**b **-** l**.**a**).**len2**(),**

d **=** x **\*** x **-** y **\*** **((**l**.**a **-** a**.**o**).**len2**()** **-** a**.**r **\*** a**.**r**);**

**if** **(**sign**(**d**)** **<** 0**)** **return** **false;**

d **=** max**(**d**,** 0.0**);**

Point p **=** l**.**a **-** **((**l**.**b **-** l**.**a**)** **\*** **(**x **/** y**)),** delta **=** **(**l**.**b **-** l**.**a**)** **\*** **(**sqrt**(**d**)** **/** y**);**

p1 **=** p **+** delta**,** p2 **=** p **-** delta**;**

**return** **true;**

**}**

// 求圆与圆的交面积

double areaCC**(**const Circle **&**c1**,** const Circle **&**c2**)** **{**

double d **=** **(**c1**.**o **-** c2**.**o**).**len**();**

**if** **(**sign**(**d **-** **(**c1**.**r **+** c2**.**r**))** **>=** 0**)** **{**

**return** 0**;**

**}**

**if** **(**sign**(**d **-** abs**(**c1**.**r **-** c2**.**r**))** **<=** 0**)** **{**

double r **=** min**(**c1**.**r**,** c2**.**r**);**

**return** r **\*** r **\*** PI**;**

**}**

double x **=** **(**d **\*** d **+** c1**.**r **\*** c1**.**r **-** c2**.**r **\*** c2**.**r**)** **/** **(**2 **\*** d**),**

t1 **=** acos**(**x **/** c1**.**r**),** t2 **=** acos**((**d **-** x**)** **/** c2**.**r**);**

**return** c1**.**r **\*** c1**.**r **\*** t1 **+** c2**.**r **\*** c2**.**r **\*** t2 **-** d **\*** c1**.**r **\*** sin**(**t1**);**

**}**

// 求圆与圆的交点，注意调用前要先判定重圆

bool isCC**(**Circle a**,** Circle b**,** Point **&**p1**,** Point **&**p2**)** **{**

double s1 **=** **(**a**.**o **-** b**.**o**).**len**();**

**if** **(**sign**(**s1 **-** a**.**r **-** b**.**r**)** **>** 0 **||** sign**(**s1 **-** abs**(**a**.**r **-** b**.**r**))** **<** 0**)** **return** **false;**

double s2 **=** **(**a**.**r **\*** a**.**r **-** b**.**r **\*** b**.**r**)** **/** s1**;**

double aa **=** **(**s1 **+** s2**)** **\*** 0.5**,** bb **=** **(**s1 **-** s2**)** **\*** 0.5**;**

Point o **=** **(**b**.**o **-** a**.**o**)** **\*** **(**aa **/** **(**aa **+** bb**))** **+** a**.**o**;**

Point delta **=** **(**b**.**o **-** a**.**o**).**unit**().**turn90**()** **\*** newSqrt**(**a**.**r **\*** a**.**r **-** aa **\*** aa**);**

p1 **=** o **+** delta**,** p2 **=** o **-** delta**;**

**return** **true;**

**}**

// 求点到圆的切点，按关于点的顺时针方向返回两个点

bool tanCP**(**const Circle **&**c**,** const Point **&**p0**,** Point **&**p1**,** Point **&**p2**)** **{**

double x **=** **(**p0 **-** c**.**o**).**len2**(),** d **=** x **-** c**.**r **\*** c**.**r**;**

**if** **(**d **<** EPS**)** **return** **false;** // 点在圆上认为没有切点

Point p **=** **(**p0 **-** c**.**o**)** **\*** **(**c**.**r **\*** c**.**r **/** x**);**

Point delta **=** **((**p0 **-** c**.**o**)** **\*** **(-**c**.**r **\*** sqrt**(**d**)** **/** x**)).**turn90**();**

p1 **=** c**.**o **+** p **+** delta**;**

p2 **=** c**.**o **+** p **-** delta**;**

**return** **true;**

**}**

// 求圆到圆的外共切线，按关于 c1.o 的顺时针方向返回两条线

vector**<**Line**>** extanCC**(**const Circle **&**c1**,** const Circle **&**c2**)** **{**

vector**<**Line**>** ret**;**

**if** **(**sign**(**c1**.**r **-** c2**.**r**)** **==** 0**)** **{**

Point dir **=** c2**.**o **-** c1**.**o**;**

dir **=** **(**dir **\*** **(**c1**.**r **/** dir**.**len**())).**turn90**();**

ret**.**push\_back**(**Line**(**c1**.**o **+** dir**,** c2**.**o **+** dir**));**

ret**.**push\_back**(**Line**(**c1**.**o **-** dir**,** c2**.**o **-** dir**));**

**}** **else** **{**

Point p **=** **(**c1**.**o **\*** **-**c2**.**r **+** c2**.**o **\*** c1**.**r**)** **/** **(**c1**.**r **-** c2**.**r**);**

Point p1**,** p2**,** q1**,** q2**;**

**if** **(**tanCP**(**c1**,** p**,** p1**,** p2**)** **&&** tanCP**(**c2**,** p**,** q1**,** q2**))** **{**

**if** **(**c1**.**r **<** c2**.**r**)** swap**(**p1**,** p2**),** swap**(**q1**,** q2**);**

ret**.**push\_back**(**Line**(**p1**,** q1**));**

ret**.**push\_back**(**Line**(**p2**,** q2**));**

**}**

**}**

**return** ret**;**

**}**

// 求圆到圆的内共切线，按关于 c1.o 的顺时针方向返回两条线

vector**<**Line**>** intanCC**(**const Circle **&**c1**,** const Circle **&**c2**)** **{**

vector**<**Line**>** ret**;**

Point p **=** **(**c1**.**o **\*** c2**.**r **+** c2**.**o **\*** c1**.**r**)** **/** **(**c1**.**r **+** c2**.**r**);**

Point p1**,** p2**,** q1**,** q2**;**

**if** **(**tanCP**(**c1**,** p**,** p1**,** p2**)** **&&** tanCP**(**c2**,** p**,** q1**,** q2**))** **{** // 两圆相切认为没有切线

ret**.**push\_back**(**Line**(**p1**,** q1**));**

ret**.**push\_back**(**Line**(**p2**,** q2**));**

**}**

**return** ret**;**

**}**

bool contain**(**vector**<**Point**>** polygon**,** Point p**)** **{** // 判断点 p 是否被多边形包含，包括落在边界上

int ret **=** 0**,** n **=** polygon**.**size**();**

**for(**int i **=** 0**;** i **<** n**;** **++** i**)** **{**

Point u **=** polygon**[**i**],** v **=** polygon**[(**i **+** 1**)** **%** n**];**

**if** **(**onSeg**(**Line**(**u**,** v**),** p**))** **return** **true;**

**if** **(**sign**(**u**.**y **-** v**.**y**)** **<=** 0**)** swap**(**u**,** v**);**

**if** **(**sign**(**p**.**y **-** u**.**y**)** **>** 0 **||** sign**(**p**.**y **-** v**.**y**)** **<=** 0**)** **continue;**

ret **+=** sign**(**det**(**p**,** v**,** u**))** **>** 0**;**

**}**

**return** ret **&** 1**;**

**}**

vector**<**Point**>** convexCut**(**const vector**<**Point**>&**ps**,** Line l**)** **{** // 用半平面 (q1,q2) 的逆时针方向去切凸多边形

vector**<**Point**>** qs**;**

int n **=** ps**.**size**();**

**for** **(**int i **=** 0**;** i **<** n**;** **++**i**)** **{**

Point p1 **=** ps**[**i**],** p2 **=** ps**[(**i **+** 1**)** **%** n**];**

int d1 **=** sign**(**det**(**l**.**a**,** l**.**b**,** p1**)),** d2 **=** sign**(**det**(**l**.**a**,** l**.**b**,** p2**));**

**if** **(**d1 **>=** 0**)** qs**.**push\_back**(**p1**);**

**if** **(**d1 **\*** d2 **<** 0**)** qs**.**push\_back**(**isLL**(**Line**(**p1**,** p2**),** l**));**

**}**

**return** qs**;**

**}**

vector**<**Point**>** convexHull**(**vector**<**Point**>** ps**)** **{** // 求点集 ps 组成的凸包

int n **=** ps**.**size**();** **if** **(**n **<=** 1**)** **return** ps**;**

sort**(**ps**.**begin**(),** ps**.**end**());**

vector**<**Point**>** qs**;**

**for** **(**int i **=** 0**;** i **<** n**;** qs**.**push\_back**(**ps**[**i**++]))**

**while** **(**qs**.**size**()** **>** 1 **&&** sign**(**det**(**qs**[**qs**.**size**()-**2**],**qs**.**back**(),**ps**[**i**]))** **<=** 0**)** qs**.**pop\_back**();**

**for** **(**int i **=** n **-** 2**,** t **=** qs**.**size**();** i **>=** 0**;** qs**.**push\_back**(**ps**[**i**--]))**

**while** **((**int**)**qs**.**size**()** **>** t **&&** sign**(**det**(**qs**[(**int**)**qs**.**size**()-**2**],**qs**.**back**(),**ps**[**i**]))** **<=** 0**)** qs**.**pop\_back**();**

qs**.**pop\_back**();** **return** qs**;**

**}**

### 半平面交

struct Point **{**

int quad**()** const **{** **return** sign**(**y**)** **==** 1 **||** **(**sign**(**y**)** **==** 0 **&&** sign**(**x**)** **>=** 0**);}**

**};**

struct Line **{**

bool include**(**const Point **&**p**)** const **{** **return** sign**(**det**(**b **-** a**,** p **-** a**))** **>** 0**;** **}**

Line push**()** const**{** // 将半平面向外推 eps

const double eps **=** 1e-6**;**

Point delta **=** **(**b **-** a**).**turn90**().**norm**()** **\*** eps**;**

**return** Line**(**a **-** delta**,** b **-** delta**);**

**}**

**};**

bool sameDir**(**const Line **&**l0**,** const Line **&**l1**)** **{** **return** parallel**(**l0**,** l1**)** **&&** sign**(**dot**(**l0**.**b **-** l0**.**a**,** l1**.**b **-** l1**.**a**))** **==** 1**;** **}**

bool **operator** **<** **(**const Point **&**a**,** const Point **&**b**)** **{**

**if** **(**a**.**quad**()** **!=** b**.**quad**())** **{**

**return** a**.**quad**()** **<** b**.**quad**();**

**}** **else** **{**

**return** sign**(**det**(**a**,** b**))** **>** 0**;**

**}**

**}**

bool **operator** **<** **(**const Line **&**l0**,** const Line **&**l1**)** **{**

**if** **(**sameDir**(**l0**,** l1**))** **{**

**return** l1**.**include**(**l0**.**a**);**

**}** **else** **{**

**return** **(**l0**.**b **-** l0**.**a**)** **<** **(**l1**.**b **-** l1**.**a**);**

**}**

**}**

bool check**(**const Line **&**u**,** const Line **&**v**,** const Line **&**w**)** **{** **return** w**.**include**(**intersect**(**u**,** v**));** **}**

vector**<**Point**>** intersection**(**vector**<**Line**>** **&**l**)** **{**

sort**(**l**.**begin**(),** l**.**end**());**

deque**<**Line**>** q**;**

**for** **(**int i **=** 0**;** i **<** **(**int**)**l**.**size**();** **++**i**)** **{**

**if** **(**i **&&** sameDir**(**l**[**i**],** l**[**i **-** 1**]))** **{**

**continue;**

**}**

**while** **(**q**.**size**()** **>** 1 **&&** **!**check**(**q**[**q**.**size**()** **-** 2**],** q**[**q**.**size**()** **-** 1**],** l**[**i**]))** q**.**pop\_back**();**

**while** **(**q**.**size**()** **>** 1 **&&** **!**check**(**q**[**1**],** q**[**0**],** l**[**i**]))** q**.**pop\_front**();**

q**.**push\_back**(**l**[**i**]);**

**}**

**while** **(**q**.**size**()** **>** 2 **&&** **!**check**(**q**[**q**.**size**()** **-** 2**],** q**[**q**.**size**()** **-** 1**],** q**[**0**]))** q**.**pop\_back**();**

**while** **(**q**.**size**()** **>** 2 **&&** **!**check**(**q**[**1**],** q**[**0**],** q**[**q**.**size**()** **-** 1**]))** q**.**pop\_front**();**

vector**<**Point**>** ret**;**

**for** **(**int i **=** 0**;** i **<** **(**int**)**q**.**size**();** **++**i**)** ret**.**push\_back**(**intersect**(**q**[**i**],** q**[(**i **+** 1**)** **%** q**.**size**()]));**

**return** ret**;**

**}**

# 其他

### 平面最近点对

const int maxn**=**1e5**+**5**;**

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

int n**;**

int a**[**maxn**];**

LL sum**[**maxn**];**

struct P**{**

LL x**,**y**;**

P**(**LL x**=**0**,**LL y**=**0**):**x**(**x**),**y**(**y**){}**

**}**p**[**maxn**];**

LL sqr**(**LL x**){return** x**\***x**;}**

bool byY**(**P a**,**P b**){return** a**.**y**<**b**.**y**;}**

LL dis2**(**P a**,**P b**){return** sqr**(**a**.**x**-**b**.**x**)+**sqr**(**a**.**y**-**b**.**y**);}**

LL solve**(**int l**,**int r**){**

LL d**=**1LL**<<**62**;**

**if(**l**==**r**)return** d**;**

**if(**l**+**1**==**r**)return** dis2**(**p**[**l**],**p**[**r**]);**

int mid**=(**l**+**r**)>>**1**;**

d**=**min**(**solve**(**l**,**mid**),**d**);**

d**=**min**(**solve**(**mid**+**1**,**r**),**d**);**

vector**<**P**>**tmp**;**

**for(**int i**=**l**;**i**<=**r**;**i**++)if(**sqr**(**p**[**mid**].**x**-**p**[**i**].**x**)<=**d**)**

tmp**.**push\_back**(**p**[**i**]);**

sort**(**tmp**.**begin**(),**tmp**.**end**(),**byY**);**

**for(**int i**=**0**;**i**<**tmp**.**size**();**i**++)**

**for(**int j**=**i**+**1**;**j**<**tmp**.**size**()&&**j**-**i**<**10**;**j**++)**

d**=**min**(**d**,**dis2**(**tmp**[**i**],**tmp**[**j**]));**

**return** d**;**

**}**

int main**(){**

scanf**(**"%d"**,&**n**);**

**for(**int i**=**1**;**i**<=**n**;**i**++)**scanf**(**"%d"**,&**a**[**i**]),**sum**[**i**]=**sum**[**i**-**1**]+**a**[**i**];**

**for(**int i**=**1**;**i**<=**n**;**i**++)**

p**[**i**].**x**=**i**,**p**[**i**].**y**=**sum**[**i**];**

cout**<<**solve**(**1**,**n**)<<**endl**;**

**return** 0**;**

**}**

### 整体二分

#include<bits/stdc++.h>

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

const int maxn**=**3e5**+**5**;**

int n**,**m**,**k**;**

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

vector**<**int**>**O**[**maxn**];**

int getint**(){**

int res**=**0**;**char c**=**getchar**();**

**while(!**isdigit**(**c**))**c**=**getchar**();**

**while(**isdigit**(**c**))**res**=**res**\***10**+**c**-**'0'**,**c**=**getchar**();**

**return** res**;**

**}**

struct BIT**{**

LL d**[**maxn**];**

inline int lowbit**(**int x**){return** x**&-**x**;}**

LL get**(**int x**){**

LL ans**=**0**;**

**while(**x**)**ans**+=**d**[**x**],**x**-=**lowbit**(**x**);**

**return** ans**;**

**}**

void updata**(**int x**,**LL f**){**

**while(**x**<=**m**)**d**[**x**]+=**f**,**x**+=**lowbit**(**x**);**

**}**

void add**(**int l**,**int r**,**LL f**){**

updata**(**l**,**f**);**

updata**(**r**+**1**,-**f**);**

**}**

**}**T**,**T2**;**

int anss**[**maxn**],**wana**[**maxn**];**

struct qes**{**

LL x**,**y**,**z**;**

qes**(**LL \_x**=**0**,**LL \_y**=**0**,**LL \_z**=**0**):**

x**(**\_x**),**y**(**\_y**),**z**(**\_z**){}**

**}**q**[**maxn**],**p**[**maxn**];**

bool part**(**qes **&**q**){**

**if(**q**.**y**+**q**.**z**>=**wana**[**q**.**x**])return** 1**;**

q**.**z**+=**q**.**y**;**q**.**y**=**0**;return** 0**;**

**}**

void solve**(**int lef**,**int rig**,**int l**,**int r**){**

**if(**l**==**r**){**

**for(**int i**=**lef**;**i**<=**rig**;**i**++)if(**anss**[**p**[**i**].**x**]!=-**1**)**

anss**[**p**[**i**].**x**]=**l**;return;**

**}**int mid**=(**l**+**r**)>>**1**;**

**for(**int i**=**l**;**i**<=**mid**;**i**++){**

**if(**q**[**i**].**x**<=**q**[**i**].**y**)**T**.**add**(**q**[**i**].**x**,**q**[**i**].**y**,**q**[**i**].**z**);**

**else** T**.**add**(**1**,**q**[**i**].**y**,**q**[**i**].**z**),**T**.**add**(**q**[**i**].**x**,**m**,**q**[**i**].**z**);**

**}for(**int i**=**lef**;**i**<=**rig**;**i**++){**

p**[**i**].**y**=**0**;**

**for(**int j**=**0**;**j**<**O**[**p**[**i**].**x**].**size**()&&**p**[**i**].**y**<=**int**(**1e9**)+**1**;**j**++)**

p**[**i**].**y**+=**T**.**get**(**O**[**p**[**i**].**x**][**j**]);**

**}for(**int i**=**l**;**i**<=**mid**;**i**++){**

**if(**q**[**i**].**x**<=**q**[**i**].**y**)**T**.**add**(**q**[**i**].**x**,**q**[**i**].**y**,-**q**[**i**].**z**);**

**else** T**.**add**(**1**,**q**[**i**].**y**,-**q**[**i**].**z**),**T**.**add**(**q**[**i**].**x**,**m**,-**q**[**i**].**z**);**

**}**int dv**=**stable\_partition**(**p**+**lef**,**p**+**rig**+**1**,**part**)-**p**-**1**;**

**if(**lef**<=**dv**)**

solve**(**lef**,**dv**,**l**,**mid**);**

**if(**dv**+**1**<=**rig**)**

solve**(**dv**+**1**,**rig**,**mid**+**1**,**r**);**

**}**

int main**(){**

n**=**getint**();**m**=**getint**();**

**for(**int i**=**1**;**i**<=**m**;**i**++){**

int x**=**getint**();**O**[**x**].**push\_back**(**i**);**

**}for(**int i**=**1**;**i**<=**n**;**i**++)**wana**[**i**]=**getint**(),**p**[**i**]=**qes**(**i**,**0**,**0**);**

k**=**getint**();**

**for(**int i**=**1**;**i**<=**k**;**i**++){**

int l**=**getint**(),**r**=**getint**(),**x**=**getint**();**

q**[**i**]=**qes**(**l**,**r**,**x**);**

**}**q**[++**k**]=**qes**(**1**,**m**,**int**(**1e9**)+**1**);**

solve**(**1**,**n**,**1**,**k**);**

**for(**int i**=**1**;**i**<=**n**;**i**++)if(**anss**[**i**]!=**k**)**

printf**(**"%d\n"**,**anss**[**i**]);else** puts**(**"NIE"**);**

**return** 0**;**

**}**

### 最小圆覆盖(随机增量)

#include<bits/stdc++.h>

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

const double eps**=**1e-6**;**

int dcmp**(**double x**){return** **(**x**>**eps**)-(**x**<-**eps**);}**

struct point**{**

double x**,**y**;**

point**(**double \_x**=**0**,**double \_y**=**0**):**x**(**\_x**),**y**(**\_y**){}**

point **operator-(**point o**){return** point**(**x**-**o**.**x**,**y**-**o**.**y**);}**

point **operator+(**point o**){return** point**(**x**+**o**.**x**,**y**+**o**.**y**);}**

double **operator\*(**point o**){return** x**\***o**.**y**-**y**\***o**.**x**;}**

double **operator^(**point o**){return** x**\***o**.**x**+**y**\***o**.**y**;}**

point **operator\*(**double p**){return** point**(**x**\***p**,**y**\***p**);}**

point **operator/(**double p**){return** point**(**x**/**p**,**y**/**p**);}**

bool **operator==(**point o**){return** **!**dcmp**(**x**-**o**.**x**)&&!**dcmp**(**y**-**o**.**y**);}**

**}**O**,**p**[**int**(**1000001**)];**

double sqr**(**double x**){return** x**\***x**;}**

double dis**(**point a**,**point b**){return** sqrt**(**sqr**(**a**.**x**-**b**.**x**)+**sqr**(**a**.**y**-**b**.**y**));}**

int n**;**

double r**;**

struct line**{**

point p**,**v**;**

**};**

point Rev**(**point v**){return** point**(-**v**.**y**,**v**.**x**);}**

point **operator\*(**line A**,**line B**){**

point u**=**B**.**p**-**A**.**p**;**

double t**=(**B**.**v**\***u**)/(**B**.**v**\***A**.**v**);**

**return** A**.**p**+**A**.**v**\***t**;**

**}**

point get**(**point a**,**point b**){**

**return** **(**a**+**b**)/**2**;**

**}**

point get**(**point a**,**point b**,**point c**){**

**if(**a**==**b**)return** get**(**a**,**c**);**

**if(**a**==**c**)return** get**(**a**,**b**);**

**if(**b**==**c**)return** get**(**a**,**b**);**

line ABO**=(**line**){(**a**+**b**)/**2**,**Rev**(**a**-**b**)};**

line BCO**=(**line**){(**c**+**b**)/**2**,**Rev**(**b**-**c**)};**

**return** ABO**\***BCO**;**

**}**

int main**(){**

scanf**(**"%d"**,&**n**);**

**for(**int i**=**1**;**i**<=**n**;**i**++)**scanf**(**"%lf%lf"**,&**p**[**i**].**x**,&**p**[**i**].**y**);**

random\_shuffle**(**p**+**1**,**p**+**1**+**n**);**

O**=**p**[**1**];**r**=**0**;**

**for(**int i**=**2**;**i**<=**n**;**i**++){**

**if(**dis**(**p**[**i**],**O**)<**r**+**1e-6**)continue;**

O**=**get**(**p**[**1**],**p**[**i**]);**r**=**dis**(**O**,**p**[**i**]);**

**for(**int j**=**1**;**j**<**i**;**j**++){**

**if(**dis**(**p**[**j**],**O**)<**r**+**1e-6**)continue;**

O**=**get**(**p**[**i**],**p**[**j**]);**r**=**dis**(**O**,**p**[**i**]);**

**for(**int k**=**1**;**k**<**j**;**k**++){**

**if(**dis**(**p**[**k**],**O**)<**r**+**1e-6**)continue;**

O**=**get**(**p**[**i**],**p**[**j**],**p**[**k**]);**r**=**dis**(**O**,**p**[**i**]);**

**}**

**}**

**}**printf**(**"%.2lf %.2lf %.2lf\n"**,**O**.**x**,**O**.**y**,**r**);**

**return** 0**;**

**}**

### 插头DP

int n**,**m**,**l**;**

struct L**{**

int d**[**11**];**

int**&** **operator[](**int x**){return** d**[**x**];}**

int mc**(**int x**){**

int an**=**1**;**

**if(**d**[**x**]==**1**){**

**for(**x**++;**x**<**l**;**x**++)if(**d**[**x**]){**

an**=**an**+(**d**[**x**]==**1**?**1**:-**1**);**

**if(!**an**)return** x**;**

**}**

**}else{**

**for(**x**--;**x**>=**0**;**x**--)if(**d**[**x**]){**

an**=**an**+(**d**[**x**]==**2**?**1**:-**1**);**

**if(!**an**)return** x**;**

**}**

**}**

**}**

int h**(){**int an**=**0**;for(**int i**=**l**-**1**;**i**>=**0**;**i**--)**an**=**an**\***3**+**d**[**i**];return** an**;}**

L s**(**int x**,**int y**){**

L S**=\*this;**

S**[**x**]=**y**;return** S**;**

**}**

L **operator>>(**int \_**){**

L S**=\*this;**

**for(**int i**=**l**-**1**;**i**>=**1**;**i**--)**S**[**i**]=**S**[**i**-**1**];**

S**[**0**]=**0**;return** S**;**

**}**

**};**

struct Int**{**

int len**;**

int a**[**40**];**

Int**(){**len**=**1**;**memset**(**a**,**0**,sizeof** a**);}**

Int **operator+=(**const Int **&**o**){**

int l**=**max**(**len**,**o**.**len**);**

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

a**[**i**]=**a**[**i**]+**o**.**a**[**i**];**

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

a**[**i**+**1**]+=**a**[**i**]/**10**,**a**[**i**]%=**10**;**

**if(**a**[**l**])**l**++;**len**=**l**;**

**return** **\*this;**

**}**

void print**(){**

**for(**int i**=**len**-**1**;**i**>=**0**;**i**--)**

printf**(**"%d"**,**a**[**i**]);**

puts**(**""**);**

**}**

**};**

struct hashtab**{**

int sz**;**

int tab**[**177147**];**

Int w**[**177147**];**

L s**[**177147**];**

hashtab**(){**memset**(**tab**,-**1**,sizeof** tab**);}**

void cl**(){**

**for(**int i**=**0**;**i**<**sz**;**i**++)**tab**[**s**[**i**].**h**()]=-**1**;**

sz**=**0**;**

**}**

Int**&** **operator[](**L S**){**

int h**=**S**.**h**();**

**if(**tab**[**h**]==-**1**)**tab**[**h**]=**sz**,**s**[**sz**]=**S**,**w**[**sz**]=**Int**(),**sz**++;**

**return** w**[**tab**[**h**]];**

**}**

**}**f**[**2**];**

bool check**(**L S**){**

int cn1**=**0**,**cn2**=**0**;**

**for(**int i**=**0**;**i**<**l**;**i**++){**

cn1**+=**S**[**i**]==**1**;**

cn2**+=**S**[**i**]==**2**;**

**}return** cn1**==**1**&&**cn2**==**1**;**

**}**

int main**(){**

Int One**;**One**.**a**[**0**]=**1**;**

scanf**(**"%d%d"**,&**n**,&**m**);if(**n**<**m**)**swap**(**n**,**m**);**l**=**m**+**1**;**

**if(**n**==**1**||**m**==**1**){**puts**(**"1"**);return** 0**;}**

int cur**=**0**;**f**[**cur**].**cl**();**

**for(**int i**=**1**;**i**<=**n**;**i**++){**

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

**if(**i**==**1**&&**j**==**1**){**

f**[**cur**][**L**().**s**(**0**,**1**).**s**(**1**,**2**)]+=**One**;**

**continue;**

**}**

cur**^=**1**;**f**[**cur**].**cl**();**

**for(**int k**=**0**;**k**<**f**[!**cur**].**sz**;**k**++){**

L S**=**f**[!**cur**].**s**[**k**];**Int w**=**f**[!**cur**][**S**];**

int d1**=**S**[**j**-**1**],**d2**=**S**[**j**];**

**if(**d1**==**0**&&**d2**==**0**){**

**if(**i**!=**n**&&**j**!=**m**)**f**[**cur**][**S**.**s**(**j**-**1**,**1**).**s**(**j**,**2**)]+=**w**;**

**}else**

**if(**d1**==**0**||**d2**==**0**){**

**if(**i**!=**n**)**f**[**cur**][**S**.**s**(**j**-**1**,**d1**|**d2**).**s**(**j**,**0**)]+=**w**;**

**if(**j**!=**m**)**f**[**cur**][**S**.**s**(**j**-**1**,**0**).**s**(**j**,**d1**|**d2**)]+=**w**;**

**}else**

**if(**d1**==**1**&&**d2**==**2**){**

**if(**i**==**n**&&**j**==**m**&&**check**(**S**))**

**(**w**+=**w**).**print**();**

**}else**

**if(**d1**==**2**&&**d2**==**1**){**

f**[**cur**][**S**.**s**(**j**-**1**,**0**).**s**(**j**,**0**)]+=**w**;**

**}else**

**if((**d1**==**1**&&**d2**==**1**)||(**d1**==**2**&&**d2**==**2**)){**

int m1**=**S**.**mc**(**j**),**m2**=**S**.**mc**(**j**-**1**);**

f**[**cur**][**S**.**s**(**j**-**1**,**0**).**s**(**j**,**0**).**s**(**m1**,**1**).**s**(**m2**,**2**)]+=**w**;**

**}**

**}**

**}**

cur**^=**1**;**f**[**cur**].**cl**();**

**for(**int k**=**0**;**k**<**f**[!**cur**].**sz**;**k**++){**

L S**=**f**[!**cur**].**s**[**k**];**Int w**=**f**[!**cur**][**S**];**

f**[**cur**][**S**>>**1**]=**w**;**

**}**

**}**

**return** 0**;**

**}**

### 线性规划

int dcmp**(**double x**){return** **(**x**>**1e-6**)-(**x**<-**1e-6**);}**

int n**,**m**,**X**;**

void pivot**(**int l**,**int e**){**

**for(**int i**=**0**;**i**<=**n**;**i**++)if(**i**!=**e**)**

a**[**l**][**i**]/=**a**[**l**][**e**];**

a**[**l**][**e**]=**1**/**a**[**l**][**e**];**

**for(**int i**=**0**;**i**<=**m**;**i**++)if(**i**!=**l**){**

**if(**dcmp**(**a**[**i**][**e**])==**0**)continue;**

**for(**int j**=**0**;**j**<=**n**;**j**++)if(**j**!=**e**)**

a**[**i**][**j**]-=**a**[**i**][**e**]\***a**[**l**][**j**];**

a**[**i**][**e**]\*=-**a**[**l**][**e**];**

**}**

**}**

double simplex**(){**

**while(**1**){**

int l**=**1**,**e**=**1**;**

**for(**int i**=**1**;**i**<=**m**;**i**++)if(**a**[**i**][**0**]<**a**[**l**][**0**])**l**=**i**;**

**if(**dcmp**(**a**[**l**][**0**])>=**0**)break;**

**for(**int i**=**1**;**i**<=**n**;**i**++)if(**a**[**l**][**i**]<**a**[**l**][**e**])**e**=**i**;**

**if(**dcmp**(**a**[**l**][**e**])>=**0**){**puts**(**"-1"**);**exit**(**0**);}**

pivot**(**l**,**e**);**

**}**

**while(**1**){**

int l**=**1**,**e**=**1**;**

**for(**int i**=**1**;**i**<=**n**;**i**++)if(**a**[**0**][**i**]>**a**[**0**][**e**])**e**=**i**;**

**if(**dcmp**(**a**[**0**][**e**])<=**0**)return** **-**a**[**0**][**0**];**

**for(**int i**=**1**;**i**<=**m**;**i**++)if(**dcmp**(**a**[**i**][**e**])>**0**&&(**a**[**l**][**e**]<=**0**||**a**[**i**][**0**]\***a**[**l**][**e**]<**a**[**l**][**0**]\***a**[**i**][**e**]))**l**=**i**;**

**if(**dcmp**(**a**[**l**][**e**])<=**0**){**puts**(**"-1"**);**exit**(**0**);}**

pivot**(**l**,**e**);**

**}**

**}**

//标准型

//a[0][1..n]表示z

//a[i][1..n]表示约束

//m为约束个数、n为变量数

### 快速乘

LL mul**(**LL a**,**LL b**,**LL p**){**//保证a,b非负

LL t**=(**a**\***b**-**LL**((**long double**)**a**/**p**\***b**+**1e-3**))%**p**;**

**return** t**<**0**?**t**+**p**:**t**;**

**}**

# Hints

### 积分表

### 数学公式

### 网络流二分图

# 待补

主席树、可持久化Trie、牛顿迭代、辛普森、DLX、FWT、后缀树、计算几何、Dominator Tree、对拍、vimrc、checker、高精度、最小树形图、带花树、二维hash(单base，矩阵平移到nm)、上下界网络流、矩乘