目录

[头文件 1](#_Toc525171293)

[数学 2](#_Toc525171294)

[波利亚定理 2](#_Toc525171295)

[快速沃尔什变换 5](#_Toc525171296)

[类欧几里得 8](#_Toc525171297)

[欧拉函数、欧拉降幂 10](#_Toc525171298)

[数据结构 10](#_Toc525171299)

[CDQ分治 10](#_Toc525171300)

[KD-TREE 19](#_Toc525171301)

[笛卡尔树 23](#_Toc525171302)

[分块 25](#_Toc525171303)

[莫队 27](#_Toc525171304)

[线段树 29](#_Toc525171305)

[整体二分 30](#_Toc525171306)

[字典树 31](#_Toc525171307)

[主席树 35](#_Toc525171308)

[图论 37](#_Toc525171309)

[2-SAT 37](#_Toc525171310)

[LCA 39](#_Toc525171311)

[点分治 39](#_Toc525171312)

[三元环 43](#_Toc525171313)

[树剖 44](#_Toc525171314)

[虚树 47](#_Toc525171315)

[树分块 50](#_Toc525171316)

[字符串 52](#_Toc525171317)

[后缀自动机 52](#_Toc525171318)

[hiho版 52](#_Toc525171319)

[SAM(hiho-struct版-无说明) 56](#_Toc525171320)

[DP 58](#_Toc525171321)

[dp套dp 58](#_Toc525171322)

[决策单调性 59](#_Toc525171323)

[树形依赖DP 60](#_Toc525171324)

[斯坦纳树 61](#_Toc525171325)

[四边形不等式 63](#_Toc525171326)

[斜率优化 64](#_Toc525171327)

[随机化 64](#_Toc525171328)

[图随机化 64](#_Toc525171329)

[基础&常用 65](#_Toc525171330)

[二分模拟乘法 65](#_Toc525171331)

[二维单调队列 65](#_Toc525171332)

[离散化 66](#_Toc525171333)

## 

## 头文件

ll gcd(ll a,ll b){return b?gcd(b,a%b):a;}

template<typename T>inline T abs(T a) {return a>0?a:-a;}

template<class T> inline

void read(T& num) {

bool start=false,neg=false;

char c;

num=0;

while((c=getchar())!=EOF) {

if(c=='-') start=neg=true;

else if(c>='0' && c<='9') {

start=true;

num=num\*10+c-'0';

} else if(start) break;

}

if(neg) num=-num;

}

inline int powMM(int a,int b,int M){

int ret=1;

a%=M;

while (b){

if (b&1) ret=1LL\*ret\*a%M;

b>>=1;

a=1LL\*a\*a%M;

}

return ret;

}

inline int add(int x,int y){

x%=MOD;y%=MOD;return (1LL\*x+y)%MOD;}

inline int add(int x,int y){

x+=y;if(x>=MOD)x-=MOD;return x;}

inline void addi(int &x,int y)

{y%=MOD;x+=y;if(x>=MOD)x-=MOD;}

inline int mul(int x,int y){return 1LL\*x\*y%MOD;}

inline void addi(int &x,int y){

x%=MOD;y%=MOD;(x+=y)%=MOD;}

inline void muli(int &x,int y){

x%=MOD;y%=MOD;x=1LL\*x\*y%MOD;if(x<0)x+=MOD;}

inline void mod(int &x){if(x<0){

x%=MOD;x=(x+MOD)%MOD;}}

ll mul(ll a , ll b,ll Q){

return (a \* b - (ll) ((long double)a \* b / Q) \* Q) % Q;

}

#define debug

clock\_t t1 = clock();

fprintf(stderr, "%ld ms\n", clock() - t1);

# 数学

## 波利亚定理

计算置换的循环数,是这一算法的瓶颈.如果能够快速计算出各置换的循环数,就可以大大提高程序的运行效率

旋转:n个点顺时针(或逆时针)旋转i个位置的置换,循环数为gcd(n,i)

翻转:

n为偶数时,

对称轴不过顶点:循环数为n/2

对称轴过顶点:循环数为n/2+1

n为奇数时,循环数为(n+1)/2

正方体棱 UVA10601

首先重要的是要弄清楚正方体的旋转，共24种变换。

1、静止不动，那么就是12个循环，每个循环节长度为1

2、通过两个对立的顶点，分别旋转120，240，有4组顶点，在每一次旋转当中，可以发现分为4个循环，每个循环节长度为3，直观的说，就是有3条边是交换的，颜色必须一样。

3、通过两个对立面的中心，分别旋转90，180，270度。有3组面

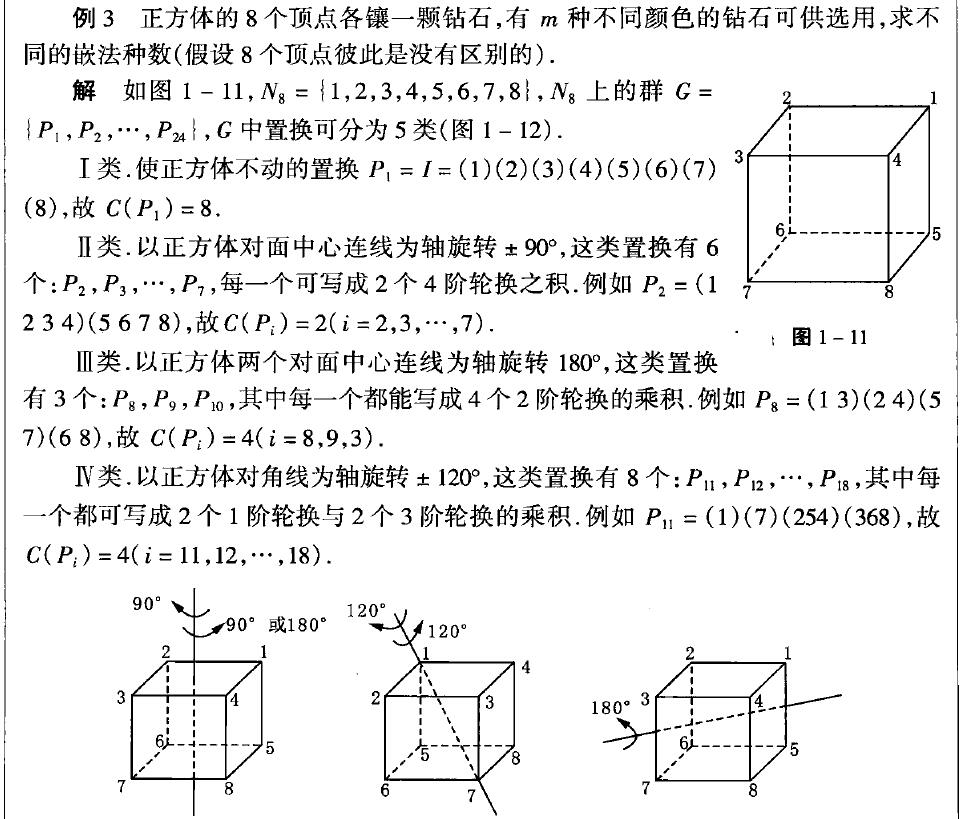
在每次旋转90度和270度的时候，可以发现分为3个循环，每个循环节长度为4

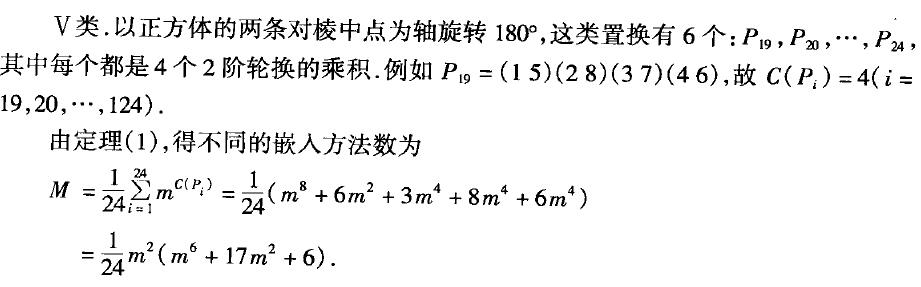
在每次旋转180度的时候，可以发现分为6个循环，每个循环节长度为2

4、通过两条对立的棱的中心，分别旋转180度，有6组棱

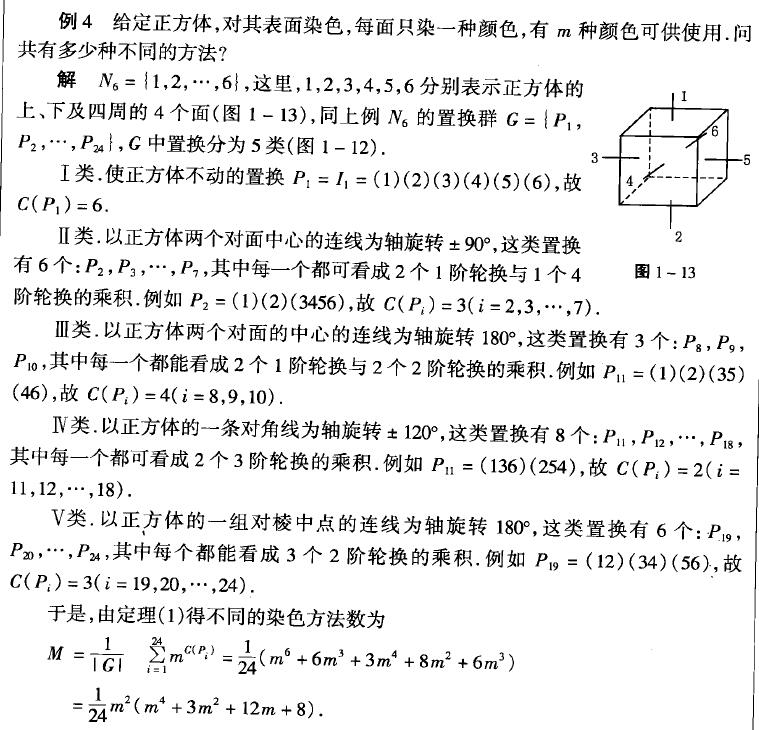
在每次旋转的时候，分为6个循环，每个循环节长度为2

**正方体顶点**





**正方体面**



1. **POJ1286-3色染可旋转可翻转的项链**

LL modxp(LL a,LL b) {

LL res=1;

while(b!=0) {

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

a=a\*a;

b>>=1;

}

return res;

}

int main() {

LL n,i;

LL ans;

while(~scanf("%lld",&n)) {

if(n==-1) break;

if(!n){

puts("0");

continue;

}//不要掉了这种情况

ans=0;

for(i=1; i<=n; i++)

ans+=modxp(3,gcd(n,i));

if(n&1) {

ans+=modxp(3,n/2+1)\*n;

} else {

ans+=modxp(3,n/2+1)\*(n/2);

ans+=modxp(3,n/2)\*(n/2);

}

printf("%lld\n",ans/(n\*2));

}

return 0;

}

1. **UVA 10601-立方体棱染色**

int T;

int a[10], b[10];

ll C[20][20];

void init()

{

C[0][0] = 1;

for(int i = 1; i < 15; ++i) {

C[i][0] = C[i][i] = 1;

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

C[i][j] = C[i - 1][j] + C[i - 1][j - 1];

}

}

}

ll work(int k)

{ //每k条边必须相同，分成12/k组(以对边中点为轴旋转180°是分成5组)

memcpy (b, a, sizeof(a));

int sum = 0;

for (int i = 1; i <= 6; ++i) {

if(b[i] % k) return 0;

b[i] /= k;

sum += b[i];

}

ll res = 1;

for (int i = 1; i <= 6; ++i) {

res \*= C[sum][b[i]];

sum -= b[i];

}

return res;

}

ll solve()

{

ll res = 0;

//静止

res += work(1);

//以相对面中心为轴

res += (ll)3 \* 2 \* work(4); //旋转90°和270°

res += (ll)3 \* work(2); //旋转180°

// 以对顶点为轴,可以旋转120°或240°

res += (ll)4 \* 2 \* work(3);

// 以对边种点为轴，只能旋转180°

for(int i = 1; i <= 6; ++i) {

for(int j = 1; j <= 6; ++j) {

if(a[i] == 0 || a[j] == 0) continue;

a[i]--; a[j]--; //将a[i]和a[j]设为选择的两条对边的颜色

res += (ll)6 \* work(2); //剩下的是5个循环长度为2的循环，6代表对边选择情况

a[i]++; a[j]++;

}

}

return res;

}

int main()

{

init();

scanf("%d", &T);

while (T--) {

memset (a, 0, sizeof(a));

for (int i = 0; i < 12; ++i) {

int tmp;

scanf ("%d", &tmp);

a[tmp]++;

}

printf("%lld\n", solve() / 24); // 最后还要除以总的置换数:24

}

return 0;

}

1. **UVA11255-限制个数的染可旋转可翻转项链-burnside**

int T;

LL C[MAXN][MAXN];

void getC(){

for(int i=0;i<MAXN;i++){

C[i][0]=C[i][i]=1;

for(int j=1;j<i;j++) C[i][j]=C[i-1][j]+C[i-1][j-1];

}

}

int gcd(int a,int b){

return a%b?gcd(b,a%b):b;

}

LL cal(int a,int b,int c,int l,int m){

if(a<0||b<0||c<0) return 0;

if(a%l||b%l||c%l) return 0;

a/=l;

b/=l;

return C[m][a]\*C[m-a][b]; //从m个循环中选a个用颜色a，从剩下的中选b个用颜色b，再剩下的用颜色c

}

int main(){

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

getC();

scanf("%d",&T);

while(T--){

int a,b,c;

scanf("%d%d%d",&a,&b,&c);

int n=a+b+c;

LL ans=0;

//旋转

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

int l=gcd(i,n);

ans+=cal(a,b,c,n/l,l);

}

//翻转

//n为奇数，分别选一种颜色的珠子穿过对称轴，这个珠子可以有n个位置

if(n%2){

ans+=cal(a-1,b,c,2,n/2)\*n;

ans+=cal(a,b-1,c,2,n/2)\*n;

ans+=cal(a,b,c-1,2,n/2)\*n;

}

//n为偶数

else{

ans+=cal(a,b,c,2,n/2)\*(n/2); //对称轴不穿过珠子，有n/2种对称轴

//对称轴穿过2种颜色珠子，有n种对称轴

ans+=cal(a-1,b-1,c,2,n/2-1)\*n;

ans+=cal(a-1,b,c-1,2,n/2-1)\*n;

ans+=cal(a,b-1,c-1,2,n/2-1)\*n;

//对称轴穿过2个同样颜色的珠子，有n/2种对称轴

ans+=cal(a-2,b,c,2,n/2-1)\*(n/2);

ans+=cal(a,b-2,c,2,n/2-1)\*(n/2);

ans+=cal(a,b,c-2,2,n/2-1)\*(n/2);

}

printf("%lld\n",ans/(2\*n));

}

return 0;

}

1. **HDU2865-k种颜色染中心有一点的环相邻颜色不同可旋转的方案数**

int n,k,f1,f2,f3,prime[MAX],phi[MAX],p[MAX];

void init(){

phi[1]=1;

for(int i=2;i<=100000;i++){

if(!p[i]){

prime[++prime[0]]=i;phi[i]=i-1;

}

for(int j=1;j<=prime[0]&&i\*prime[j]<=100000;j++){

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

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

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

}

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

}

}

}

ll getphi(int x){

if(x<=100000)return phi[x];

ll re=x;

for(int i=1;x>1&&i<=prime[0]&&prime[i]\*prime[i]<=x;++i)

if(x%prime[i]==0){

re=re\*(prime[i]-1)/prime[i];

while(x%prime[i]==0)x/=prime[i];

}

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

return re%MOD;

}

const int m\_num=10;

struct matrix{

int e[m\_num][m\_num];

int row,col;

matrix(){}

matrix(int \_r,int \_c):row(\_r),col(\_c){memset(e,0,sizeof(e));}

matrix operator \* (const matrix &tem)const{

matrix ret=matrix(row,tem.col);

for(int i=1;i<=ret.row;i++)

for(int j=1;j<=ret.col;j++)

for(int k=1;k<=ret.col;k++)

addi(ret.e[i][j],mul(e[i][k],tem.e[k][j]));

return ret;

}

void getE(){

for(int i=1;i<=row;i++)

for(int j=1;j<=col;j++)e[i][j]=(i==j);

}

}xi,an;

matrix m\_qpow(matrix tem,int x){

matrix ret=matrix(tem.row,tem.row);

ret.getE();

while(x){

if(x&1)ret=ret\*tem;

x>>=1;tem=tem\*tem;

}

return ret;

}

int getf(int d){

if(d==1)return f1;

else if(d==2)return f2;

else if(d==3)return f3;

else return (an\*m\_qpow(xi,d-3)).e[1][1];

}

int main(){

init();

xi.col=2;xi.row=2;

an.col=2;an.row=1;

while(~scanf("%d%d",&n,&k)){

int ans=0;

xi.e[1][1]=k-3;xi.e[1][2]=1;

xi.e[2][1]=k-2;xi.e[2][2]=0;

f2=mul(k-1,k-2);

f1=k-1LL;

f3=mul(f2,k-3);

an.e[1][1]=f3;an.e[1][2]=f2;

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

if(n%i==0){

addi(ans,mul(getphi(n/i),getf(i) ) );

if(i\*i!=n)

addi(ans,mul(getphi(i),getf(n/i)));

}

}

addi(ans,MOD-mul(getphi(n),k-1) );

muli(ans,mul(k,powMM(n,MOD-2,MOD ) ));

printf("%d\n",ans);

}

return 0;

}

## 快速沃尔什变换

/\*

以下的各个模板中 dwt中的/2 结合具体情况转化为2的逆元

也可以不在过程中除以2 转为在求完之后直接除以数组长度 (2^k)

并且有一个trick为 过程中取模就直接以 (2^k\*MOD) 为模

这样最后就可以直接除以2^k了

不过这样乘法取模过程中可能会溢出ll 需要使用常用模板文件夹下的防溢出乘法

\*/

/\*

最速版 非递归 速度远超其他 DWT只需加一个除2操作 这里FWT展示的是不取模的 DWT展示的是取模的

使用是 for(int i=0;i<n;i++)a[i]=a[i]\*b[i] 这里也可以加取模操作 在模意义下不影响结果

\*/

template<typename T>void FWT(T\* a,int len)

{

for (int hl = 1, l = 2; l <= len; hl = l, l <<= 1)

for (T i = 0; i < len; i += l)

for (register T t, j = 0, \*x = a + i, \*y = x + hl; j < hl; ++j, ++x, ++y) t = \*x + \*y, \*y = \*x - \*y, \*x = t; return;

}

template<typename T>void DWT(T\* a,int len,int inv)

{

for (int hl = 1, l = 2; l <= len; hl = l, l <<= 1)

for (T i = 0; i < len; i += l)

for (register T t, j = 0, \*x = a + i, \*y = x + hl; j < hl; ++j, ++x, ++y) t = mul(add(\*x , \*y),inv), \*y = mul(inv,add(\*x,MOD - \*y)), \*x = t; return;

}

/\*

不通过DWT 直接O(n)获取 某一下标的 DWT后的结果 cnt[i]表示i的二进制1的个数 可以预处理出来

addi(int &x,int y) 修改x的值 （模意义下）

模可能会产生问题（毕竟是模意义的hash处理 但概率很小）可以多hash解决

\*/

int cnt[MAX];

int get(int\* a,int len,int pos){

int re=0;

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

if(cnt[i&pos]&1)addi(re,MOD-a[i]);

else addi(re,a[i]);

return re;

}

/\*

递归 n稍大一点就很慢了

\*/

template<typename T> void fwt(T \*a,int l,int r)

{

if(l==r)return;

int n=(r-l+1)/2,mid=l+n-1;

fwt(a,l,mid);fwt(a,mid+1,r);

for(int i=1;i<=mid;i++)

{

T x=a[i],y=a[i+n];

a[i]=x+y;a[i+n]=x-y;

}

}

template<typename T> void dwt(T \*a,int l,int r)

{

if(l==r)return;

int n=(r-l+1)/2,mid=l+n-1;

dwt(a,l,mid);dwt(a,mid+1,r);

for(int i=1;i<=mid;i++)

{

T x=a[i],y=a[i+n];

a[i]=(x+y)/2;a[i+n]=(x-y)/2;

}

}

/\*

以上为^运算

&运算： fwt(a)=(fwt(a0+a1), fwt(a1) )

dwt(a)=(dwt(a0-a1), dwt(a1) )

or运算 fwt(a)=(fwt(a0), fwt(a0+a1) )

dwt(a)=(dwt(a0), dwt(a1-a0) )

\*/

/\*

非递归版

其中UFWT仅适用于a^k 这种 如果不是的话需要改一下

只需要把inv设成 2^(MOD-2)即可 即2的逆元

\*/

void FWT()

{

for(int d=1;d<len;d<<=1)

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

for(int j=0;j<d;j++)

{

int x=a[i+j],y=a[i+j+d];

a[i+j]=(x+y)%p,a[i+j+d]=(x-y+p)%p;

}

}

void UFWT()

{

for(int d=1;d<len;d<<=1)

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

for(int j=0;j<d;j++)

{

int x=a[i+j],y=a[i+j+d];

a[i+j]=1ll\*(x+y)\*inv%p,a[i+j+d]=(1ll\*(x-y)\*inv%p+p)%p;

}

}

/\*

非递归 传数组版 注意这个区间是左闭右开的！

\*/

void Transform(int l, int r, LL a[]) // [l, r)

{

if(l == r - 1) return ;

int len = (r - l) >> 1;

int m = l + len;

Transform(l, m, a);

Transform(m, r, a);

for(int i = l; i < m; i ++)

{

LL x1 = a[i], x2 = a[i + len];

a[i] = (x1 - x2 + MOD) % MOD;

a[i + len] = (x1 + x2) % MOD;

}

}

void ReTranform(int l, int r, LL a[])

{

if(l == r - 1) return ;

int len = (r - l) >> 1;

int m = l + len;

for(int i = l; i < m; i ++)

{

LL y1 = a[i], y2 = a[i + len];

a[i] = (y1 + y2) \* Inv2 % MOD;

a[i + len] = (y2 - y1 + MOD) \* Inv2 % MOD;

}

ReTranform(l, m, a);

ReTranform(m, r, a);

}

1. **hiho1230**

/\*

题意：有2\*n+1个人，第一遍给每个人先发 [0,m]中任意值的钱，第二遍再给每个人发x元，

x的范围为[L,R]。问有多少种第一遍发钱的方法使得：存在x使得这些人的钱的异或值在第二遍发完之后为0。

解法：首先暴力打表发现如果某一方案可行，则第二遍发钱的值x一定唯一。（并不会证，但发现2\*n+1为奇数时有该性质）

然后就可以转化为：先枚举x，然后求给每个人发[x, x+m]元钱，使得他们异或值为0的方案数。

\*/

ll inv;

int a[MAX];

void fwt(int \*a,int l,int r)//左闭右闭

{

if(l==r)return;

int n=(r-l+1)/2,mid=l+n-1;

fwt(a,l,mid);fwt(a,mid+1,r);

for(int i=l;i<=mid;i++)

{

ll x=a[i],y=a[i+n];

a[i]=(x+y)%MOD;a[i+n]=(x-y+MOD)%MOD;

}

}

void dwt(int \*a,int l,int r)

{

if(l==r)return;

int n=(r-l+1)/2,mid=l+n-1;

dwt(a,l,mid);dwt(a,mid+1,r);

for(int i=l;i<=mid;i++)

{

ll x=a[i],y=a[i+n];

a[i]=(x+y)%MOD\*inv%MOD;a[i+n]=(x-y+MOD)%MOD\*inv%MOD;

}

}

ll cal(int l,int r,int n)

{

int N=1;

while(N<=r)N<<=1;

for(int i=0;i<N;i++)a[i]=(i<=r&&i>=l)?1:0;

fwt(a,0,N-1);//左闭右闭 故需要-1

for(int i=0;i<N;i++)a[i]=powMM(a[i],n,MOD);

dwt(a,0,N-1);

return (a[0]+MOD)%MOD;

}

int n,m,l,r;

int main()

{

inv=powMM(2,MOD-2,MOD);

while(~scanf("%d%d%d%d",&n,&m,&l,&r))

{

n=2\*n+1;

ll ans=0;

for(int i=l;i<=r;i++)(ans+=cal(i,m+i,n))%=MOD;//枚举取的k

printf("%lld\n",ans);

}

return 0;

}

**例2-2018牛客暑期多校第九场A**

template<typename T>void FWT(T\* a,int len)

{

for (int hl = 1, l = 2; l <= len; hl = l, l <<= 1)

for (T i = 0; i < len; i += l)

for (register T t, j = 0, \*x = a + i, \*y = x + hl; j < hl; ++j, ++x, ++y) t = add(\*x , \*y), \*y = add(\*x,MOD - \*y), \*x = t; return;

}

template<typename T>void DWT(T\* a,int len,int inv)

{

for (int hl = 1, l = 2; l <= len; hl = l, l <<= 1)

for (T i = 0; i < len; i += l)

for (register T t, j = 0, \*x = a + i, \*y = x + hl; j < hl; ++j, ++x, ++y) t = mul(add(\*x , \*y),inv), \*y = mul(inv,add(\*x,MOD - \*y)), \*x = t; return;

}

int a[MAX],b[MAX],inv,n;

int main()

{

read(n);

for(int i=0;i<n;i++)read(a[i]);

for(int i=0;i<n;i++)read(b[i]);

FWT(a,n);FWT(b,n);

inv=powMM(2,MOD-2,MOD);

for(int i=0;i<n;i++)a[i]=mul(powMM(a[i],MOD-2,MOD),b[i]);

DWT(a,n,inv);

for(int i=0;i<n;i++)printf("%d\n",a[i]);

return 0;

}

## 类欧几里得

一定注意前面是a,后面是b,线段树一定要注意顺序

f(a,b,c,n)=sigma{(ai+b)/c}; (0->n)

g(a,b,c,n)=sigma{(ai+b)/c\*i}; (0->n)

h(a,b,c,n)=sigma{((ai+b)/c)^2}; (0->n)

let m=(a\*n+b)/c;

推导f:

a=0:

return b/c\*(n+1)

a>=c||b>=c:有一部分是规律的;

return (a/c)\*n(n+1)/2+(b/c)\*(n+1)+f(a%c,b%c,c,n)

else:直接算,这个东西是个梯形中的点数,反过来算就可以了

f(a,b,c,n)=∑i=0->n ∑j=0->m-1 [(ai+b)/c>=j+1]

f(a,b,c,n)=∑i=0->n ∑j=0->m-1 [ai>=cj+c−b]

f(a,b,c,n)=∑i=0->n ∑j=0->m-1 [ai>cj+c−b−1]

f(a,b,c,n)=∑i=0->n ∑j=0->m-1 [i>(cj+c−b−1)/a]

f(a,b,c,n)=∑j=0->m (n−(cj+c−b−1)/a)

f(a,b,c,n)=n\*m-f(c,c-b-1,a,m-1);

推导g:

a=0:

return b/c\*n(n+1)/2 (sigma的是i)

a>=c||b>=c:有一部分是规律的;

g(a,b,c,n)=(a/c)\*n(n+1)(2n+1)/6+(b/c)\*n(n+1)/2+g(a%c,b%c,c,n)

else:

g(a,b,c,n)=∑i=0->n i\*∑j=0->m [(ai+b)/c>=j]

g(a,b,c,n)=∑i=0->n i\*∑j=0->m-1 [i>(cj+c−b−1)/a]

然后把这个i放进去求和

g(a,b,c,n)=1/2\*∑j=0->m-1 (n+1+(cj+c−b−1)/a)\*(n−(cj+c−b−1)/a)

g(a,b,c,n)=1/2\*∑j=0->m-1 n(n+1)−(cj+c−b−1)/a−[(cj+c−b−1)/a]^2

g(a,b,c,n)=1/2\*[n(n+1)\*m−f(c,c−b−1,a,m−1)−h(c,c−b−1,a,m−1)]

推导h:

a=0:

return (b/c)^2\*(n+1) (sigma的是i)

a>=c||b>=c:有一部分是规律的;

h(a,b,c,n)=(a/c)^2\*n(n+1)(2n+1)/6+(b/c)^2\*(n+1)+(a/c)\*(b/c)\*n(n+1)

+h(a%c,b%c,c,n)+2\*(a/c)\*g(a%c,b%c,c,n)+2\*(b/c)\*f(a%c,b%c,c,n)

else:

n^2=2\*n(n+1)/2−n=2(∑i=0->n i)−n

有了思路我们来推h

h(a,b,c,n)=∑i=0->n (2(∑j=1->(ai+b)/c j)−(ai+b)/c)

可以想到交换主体。

h(a,b,c,n)=∑j=0->m-1 (j+1)\*∑i=0->n [(ai+b)/c>=j+1]−f(a,b,c,n)

h(a,b,c,n)=∑j=0->m-1 (j+1)\*∑i=0->n [i>(cj+c−b−1)/a]−f(a,b,c,n)

h(a,b,c,n)=∑j=0->m-1 (j+1)\*(n−(cj+c−b−1)/a)−f(a,b,c,n)

h(a,b,c,n)=n\*m(m+1)−2g(c,c−b−1,a,m−1)−2f(c,c−b−1,a,m−1)−f(a,b,c,n)

1. **BZOJ3817-无理数斜率的f函数求解**

int T,t,n,r;

double R;

int calc(int a,int b,int c,int n){

if(!n)return 0;

int \_gcd=gcd(a,gcd(b,c));

a/=\_gcd;b/=\_gcd;c/=\_gcd;//约分

ll m=((ll)a\*R+b)/c,sum=(ll)n\*(n+1)/2\*m;//a>=c 或b>=c的部分

b-=m\*c;//d-[d]

m=((ll)a\*R+b)/c\*n;//矩形的右上角交点的y值（下取整）

sum+=n\*m;//加上矩形的

return sum-calc(a\*c,-b\*c,a\*a\*r-b\*b,m);//之前的总值减去上三角形（将其竖置）的

//这里只考虑斜率为无理数的情况 因此不会有矩形对角线上的交点

}

int main()

{

read(T);

while(T--){

read(n);read(r);

R=sqrt(r);

t=(int)R;

if(t\*t==r)printf("%d\n",t&1?(n&1?-1:0):n);

else printf("%d\n",n-(((calc(1,0,1,n)-(calc(1,0,2,n)<<1))<<1)));

}

return 0;

}

1. **BZOJ2987-f函数的变形-方法1：转化为f函数**

ll calc(ll n,ll a,ll b,ll c){

ll re=0;

if(n<0)return re;

if(a>c||b>c){

re+=n\*(n+1)/2LL\*(a/c);

re+=(n+1)\*(b/c);

a%=c;b%=c;

}

ll new\_n=(a\*n+b)/c;

re+=new\_n\*n-calc(new\_n-1,c,c-b-1,a);

return re;

}

ll a,b,c,ans;

int main()

{

read(a);read(b);read(c);

ll minn=(c-a)/b,tmp=(b+a-1)/a;

ans-=tmp\*(minn+1)\*minn/2;

ans+=c/a+c/b+1;//某个为0的情况 + 均为0的情况

cout<<ans+calc(minn-1,tmp\*a-b,c+tmp\*a-b,a);

return 0;

}

1. **BZOJ2987-f函数的变形-方法2：现推迭代式**

ll calc(ll n,ll a,ll b,ll c){

ll re=0;

if(!n)return re;

if(c<0){

ll tem=(-c+a-1)/a;

re-=tem\*n;c+=tem\*a;

}

if(c/a>0||b/a>0){

re+=c/a\*n;re+=b/a\*n\*(n+1)/2;

c%=a;b%=a;

}

ll new\_n=(b\*n+c)/a;

re+=new\_n\*(n+1)-calc(new\_n,b,a,b-c-1);

return re;

}

ll a,b,c,ans;

int main()

{

read(a);read(b);read(c);

ll minn=(c-a)/b,tmp=(b+a-1)/a;

ans-=tmp\*(minn+1)\*minn/2;

ans+=c/a+c/b+1;//某个为0的情况 + 均为0的情况

cout<<ans+calc(minn,a,tmp\*a-b,c);

return 0;

}

**例4-BZOJ2187-非典型性类欧几里德**

inline void simplify(ll &a,ll &b){

ll g=gcd(a,b);a/=g;b/=g;

}

pll solve(ll p1,ll q1,ll p2,ll q2){

simplify(p1,q1);simplify(p2,q2);

ll a=p1/q1+1,b=(p2+q2-1)/q2-1;

if(a<=b)return mp(a,1);

if(p1==0)return mp(1,(ll)(q2/p2)+1);

if(p1<=q1&&p2<=q2){

pll re=solve(q2,p2,q1,p1);

swap(re.first,re.second);

return re;

}

ll t=p1/q1;

pll re=solve(p1%q1,q1,p2-t\*q2,q2);

re.first+=re.second\*t;

return re;

}

ll a,b,c,d;

int main()

{

while(~scanf("%lld%lld%lld%lld",&a,&b,&c,&d)){

pll ans=solve(a,b,c,d);

printf("%lld/%lld\n",ans.first,ans.second);

}

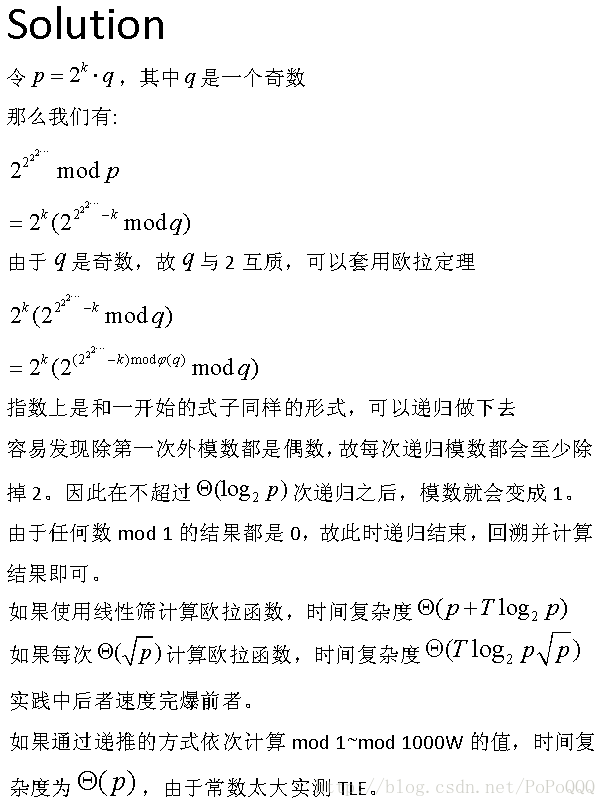
return 0;

}

## 欧拉函数、欧拉降幂

**欧拉降幂-例1-BZOJ3884**

题目大意：求2^(2^(2^(2^(2^...)))) mod p的值



int phi(int x){

int ans=x;

for(int i=2,lim=sqrt(x)+1;i<lim;i++)

if(!(x%i)){

ans-=ans/i;

while(!(x%i))x/=i;

}

return x>1?ans-ans/x:ans;

}

int f(int x){

if(x==1)return 0;

int p=phi(x);

return powMM(2,f(p)+p,x);

}

int t;

int main()

{

read(t);

while(t--){

int x;read(x);

printf("%d\n",f(x));

}

return 0;

}

# 数据结构

## CDQ分治

**CDQ分治-例1-二维偏序问题-洛谷P3374**

/\*

　　给定一个N个元素的序列a，初始值全部为0，对这个序列进行以下两种操作：

　　操作1：格式为1 x k，把位置x的元素加上k（位置从1标号到N）。

　　操作2：格式为2 x y，求出区间[x,y]内所有元素的和。

说明：

这是一个经典的树状数组问题，可以毫无压力地秒掉，现在，我们用CDQ分治解决它——带修改和查询的问题。

　　我们把他转化成一个二维偏序问题，每个操作用一个有序对(a,b)表示，其中a表示操作到来的时间，b表示操作的位置，

时间是默认有序的，所以我们在合并子问题的过程中，就按照b从小到大的顺序合并。

　　问题来了：如何表示修改与查询？

　　具体细节请参见代码，这里对代码做一些解释，请配合代码来看。

我们定义结构体Query包含3个元素：type，idx，val，其中idx表示操作的位置，type为1表示修改，val表示“加上的值”。

而对于查询，我们用前缀和的思想把他分解成两个操作：sum[1,y]-sum[1,x-1]，即分解成两次前缀和的查询。

在合并的过程中，type为2表示遇到了一个查询的左端点x-1，需要把该查询的结果减去当前“加上的值的前缀和”，

type为3表示遇到了一个查询的右端点y，需要把查询的结果加上当前“加上的值的前缀和”，val表示“是第几个查询”。

这样，我们就把每个操作转换成了带有附加信息的有序对(时间，位置)，然后对整个序列进行CDQ分治。

　　有几点需要注意：

对于位置相同的操作，要先修改后查询。

代码中为了方便，使用左闭右开区间。

合并问题的时候统计“加上的值的前缀和”，只能统计左边区间内的修改操作，改动查询结果的时候，只能修改右边区间内的查询结果。因为只有左边区间内的修改值对右边区间内的查询结果的影响还没有统计。

代码中，给定的数组是有初始值的，可以把每个初始值变为一个修改操作。

\*/

int n,m;

struct Query{

int type,idx;

ll val;

bool operator<(const Query &rhs)const{//按照位置从小到大排序，修改优先于查询

return idx==rhs.idx?type<rhs.type:idx<rhs.idx;

}

}query[MAX];

int qidx,aidx;

ll ans[MAX];//答案数组

Query tmp[MAX];//归并用临时数组

void cdq(int L,int R){//左闭右开

if(R-L<=1)return;

int M=(L+R)>>1;cdq(L,M);cdq(M,R);

ll sum=0;

int p=L,q=M,o=0;//由左侧向右侧合并

while(p<M &&q<R){

if(query[p]<query[q]){//只统计左边区间内的修改值

if(query[p].type==1)sum+=query[p].val;

tmp[o++]=query[p++];

}

else{//只修改右边区间内的查询结果

if(query[q].type==2)ans[query[q].val]-=sum;

else if(query[q].type==3)ans[query[q].val]+=sum;

tmp[o++]=query[q++];

}

}

while(p<M)tmp[o++]=query[p++];

while(q<R){

if(query[q].type==2)ans[query[q].val]-=sum;

else if(query[q].type==3)ans[query[q].val]+=sum;

tmp[o++]=query[q++];

}

for(int i=0;i<o;++i)query[i+L]=tmp[i];//顺便把序排了?

}

int main()

{

read(n);read(m);

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

query[qidx].idx=i;query[qidx].type=1;

scanf("%lld",&query[qidx].val);++qidx;

}

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

int type;read(type);

query[qidx].type=type;

if(type==1)scanf("%d%lld",&query[qidx].idx,&query[qidx].val);

else{//把查询操作分为两部分

int l,r;read(l);read(r);

query[qidx].idx=l-1;query[qidx].val=aidx;++qidx;

query[qidx].type=3;query[qidx].idx=r;query[qidx].val=aidx;

++aidx;

}

++qidx;

}

cdq(0,qidx);

for(int i=0;i<aidx;++i)printf("%lld\n",ans[i]);

return 0;

}

**CDQ分治-例2-三维偏序问题**

/\*

平面上有N个点，每个点的横纵坐标在[0,1e7]之间，有M个询问，

每个询问为查询在指定矩形之内有多少个点，矩形用(x1,y1,x2,y2)的方式给出，其中(x1,y1)为左下角坐标，(x2,y2)为右上角坐标。

\*/

struct Query{

int type,x,y,w,aid;//w表示对查询结果贡献（+还是-），aid是“第几个查询”

bool operator<(const Query &rhs)const{

return x==rhs.x?type<rhs.type:x<rhs.x;

}

}query[MAX];

int n,m,qidx,aidx,maxy;

void addq(int type,int x,int y,int w,int aid){

query[qidx++]=(Query){type,x,y,w,aid};

}

int ans[MAX];

//尽管BIT查询会出现坐标为0，而函数中都从1开始or结束 但无影响只要有0就返回0即可

namespace BIT{

int arr[MAX];

inline int lowbit(int num){return num&(-num);}

void add(int idx,int val){

while(idx<=maxy){

arr[idx]+=val;idx+=lowbit(idx);

}

}

int query(int idx){

int ans=0;

while(idx){

ans+=arr[idx];

idx-=lowbit(idx);

}

return ans;

}

void clear(int idx){

while(idx<=maxy){

if(arr[idx])arr[idx]=0;

else break;

idx+=lowbit(idx);

}

}

}

Query tmp[MAX];

void cdq(int L,int R){//左闭右开

if(R-L<=1)return;

int M=(R-L)>>1;cdq(L,M);cdq(M,R);

int p=L,q=M,o=L;

while(p<M&&q<R){

if(query[p]<query[q]){

if(query[p].type==0)BIT::add(query[p].y,1);

tmp[o++]=query[p++];

}

else{

if(query[q].type==1&&query[q].w)ans[query[q].aid]+=query[q].w\*BIT::query(query[q].y);

tmp[o++]=query[q++];

}

}

while(p<M)tmp[o++]=query[p++];

while(q<R){

if(query[q].type==1&&query[q].w)ans[query[q].aid]+=query[q].w\*BIT::query(query[q].y);

tmp[o++]=query[q++];

}

for(int i=L;i<R;i++){

BIT::clear(tmp[i].y);//清空树状数组

query[i]=tmp[i];

}

}

int main()

{

read(n);read(m);

while(n--){

int x,y;

read(x);read(y);++x;++y;//为了方便(防止出现负数)，把坐标转化为[1,1e7+1]

addq(0,x,y,0,0);maxy=max(maxy,y);//修改操作无附加信息

}

while(m--){

int xl,yl,xr,yr;

read(xl);read(yl);read(xr);read(yr);

++xl;++yl;++xr;++yr;

addq(1,xl-1,yl-1,1,aidx);

addq(1,xl-1,yr,-1,aidx);

addq(1,xr,yl-1,-1,aidx);

addq(1,xr,yr,1,aidx);

++aidx;

maxy=max(maxy,yr);

}

cdq(0,qidx);

for(int i=0;i<aidx;i++)printf("%d\n",ans[i]);

return 0;

}

**CDQ分治-例3-动态逆序对-BZOJ3295**

int n,m,cnt,qcnt;

int lo[MAX],id[MAX],val[MAX];

ll ans[MAX];

inline int lowbit(int x){return x&(-x);}

void add\_suf(int lo,int v){

while(lo<=n)val[lo]+=v,lo+=lowbit(lo);

}

void add\_pre(int lo,int v){

while(lo)val[lo]+=v,lo-=lowbit(lo);

}

int query\_pre(int lo){

--lo;int re=0;

while(lo)re+=val[lo],lo-=lowbit(lo);

return re;

}

int query\_suf(int lo){

++lo;int re=0;

while(lo<=n)re+=val[lo],lo+=lowbit(lo);

return re;

}

void clear\_suf(int lo){

while(lo<=n&&val[lo])val[lo]=0,lo+=lowbit(lo);

}

void clear\_pre(int lo){

while(lo&&val[lo])val[lo]=0,lo-=lowbit(lo);

}

struct Query{

int t,x,y,opt;

bool operator<(const Query &q)const{

return x<q.x;

}

}qs[MAX],tmp[MAX];

bool cmp(const Query &a,const Query &b){

return a.t<b.t;

}

void addq(int t,int x,int y,int opt){

qs[++qcnt]=Query{t,x,y,opt};

}

void cdq(int l,int r){

if(l==r)return;

int mid=(l+r)/2;

cdq(l,mid);cdq(mid+1,r);

int lp=l,rp=mid+1,tp=l;

while(lp<=mid&&rp<=r){

if(qs[lp]<qs[rp]){

add\_pre(qs[lp].y,1);

tmp[tp++]=qs[lp++];

}

else{

if(qs[rp].opt)

ans[qs[rp].opt]+=query\_suf(qs[rp].y);

tmp[tp++]=qs[rp++];

}

}

while(lp<=mid)tmp[tp++]=qs[lp++];

while(rp<=r){

if(qs[rp].opt)ans[qs[rp].opt]+=query\_suf(qs[rp].y);

tmp[tp++]=qs[rp++];

}

for(int i=l;i<=r;i++)clear\_pre(tmp[i].y);

for(int i=r;i>=l;i--){

if(tmp[i].t<=mid)

add\_suf(tmp[i].y,1);

else if(tmp[i].opt)

ans[tmp[i].opt]+=query\_pre(tmp[i].y);

}

for(int i=l;i<=r;i++)qs[i]=tmp[i],clear\_suf(tmp[i].y);

}

int main(){

read(n);read(m);

for(int u,i=1;i<=n;i++)read(u),lo[u]=i;

for(int u,i=0;i<m;i++)read(u),id[u]=n-i;

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

int temt=id[i];if(!temt)temt=++cnt;

addq(temt,lo[i],i,id[i]?id[i]:0);

}

sort(qs+1,qs+1+qcnt,cmp);

ll tot=0;

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

tmp[i]=qs[i];

sort(tmp+1,tmp+1+n-m);

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

{

tot+=query\_suf(tmp[i].y);

add\_pre(tmp[i].y,1);

}

for(int i=1;i<=n-m;i++)clear\_pre(qs[i].y);

cdq(1,qcnt);

ans[n-m]=tot;

for(int i=n-m+1;i<=n;i++)ans[i]+=ans[i-1];

for(int i=n;i>n-m;i--)printf("%lld\n",ans[i]);

return 0;

}

**CDQ分治-例4-CDQ斜率优化-BZOJ2726**

int n,s;

ll t[MAX],f[MAX],dp[MAX],T[MAX],F[MAX];

struct node{

ll k,x,y,pos,pd;//pd??

node(ll pos=0):pos(pos){

k=T[pos];

}

void get(){

x=F[pos];y=dp[pos]-F[n]\*T[pos]+F[pos]\*T[pos]-s\*F[pos];

}

bool operator <(const node&rhs)const{

return k<rhs.k;

}

}q[MAX],now[MAX],sta[MAX];

bool cmp(const node &lhs,const node &rhs){

return lhs.x<rhs.x||(lhs.x==rhs.x&&lhs.y<rhs.y);

}

ll dy(const node &a,const node &b){

return a.y-b.y;

}

ll dx(const node &a,const node &b){

return a.x-b.x;

}

void solve(int l,int r){

if(l==r){

q[l].get();return;

}

int mid=(l+r)/2,l1=l,l2=mid+1;

for(int i=l;i<=r;i++)

if(q[i].pos<=mid)now[l1++]=q[i];

else now[l2++]=q[i];

for(int i=l;i<=r;i++)q[i]=now[i];

solve(l,mid);

solve(mid+1,r);

int top=0;

for(int i=l;i<=mid;i++){//k is increasing 维护左侧点的下凸包

while(top>=2&&dy(q[i],sta[top])\*dx(sta[top],sta[top-1])<dy(sta[top],sta[top-1])\*dx(q[i],sta[top]))--top;

sta[++top]=q[i];

}

int j=1;

for(int i=mid+1;i<=r;i++){

while(j<top&&q[i].k\*dx(sta[j],sta[j+1])<dy(sta[j],sta[j+1]))++j;

dp[q[i].pos]=min(dp[q[i].pos],sta[j].y-sta[j].x\*q[i].k+F[n]\*(T[q[i].pos]+s));

}

l1=l,l2=mid+1;

for(int i=l;i<=r;i++)

if((cmp(q[l1],q[l2])||(l2>r))&&l1<=mid)now[i]=q[l1++];

else now[i]=q[l2++];

for(int i=l;i<=r;i++)q[i]=now[i];

}

int main()

{

read(n);read(s);

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

read(t[i]),read(f[i]),dp[i]=INFF;

for(int i=1;i<=n;i++)T[i]=T[i-1]+t[i],F[i]=F[i-1]+f[i];

for(int i=0;i<=n;i++)q[i]=node(i);

sort(q,q+n+1);

solve(0,n);

printf("%lld\n",dp[n]);

return 0;

}

**CDQ分治-例5-CDQ分治斜率优化-BZOJ1492**

const int N=1e5+10;

const double inf=1e20;

//const double eps=1e-8;

struct Pt{

double x,y,a,b,k,r;

int id;

bool operator <(const Pt &rhs)const{

return k>rhs.k;

}

}p[N],t[N];

double f[N];

int n,top,st[N];

double slop(int a,int b){

if(!b)return -inf;

if(fabs(p[a].x-p[b].x)<eps)return inf;

return (p[b].y-p[a].y)/(p[b].x-p[a].x);

}

void solve(int l,int r){

if(l==r){

f[l]=max(f[l],f[l-1]);

p[l].y=f[l]/(p[l].a\*p[l].r+p[l].b);

p[l].x=p[l].y\*p[l].r;

return ;

}

int mid=(l+r)>>1,j=1,l1=l,l2=mid+1;

for(int i=l;i<=r;i++){

if(p[i].id<=mid)t[l1++]=p[i];

else t[l2++]=p[i];

}

for(int i=l;i<=r;i++)p[i]=t[i];

solve(l,mid);

top=0;

for(int i=l;i<=mid;i++){

while(top>1&&slop(st[top-1],st[top])<slop(st[top-1],i)+eps)--top;

st[++top]=i;

}

st[++top]=0;

for(int i=mid+1;i<=r;i++){

while(j<top&&slop(st[j],st[j+1])+eps>p[i].k)++j;

f[p[i].id]=max(f[p[i].id],p[st[j]].x\*p[i].a+p[st[j]].y\*p[i].b);

}

solve(mid+1,r);

l1=l,l2=mid+1;

for(int i=l;i<=r;i++){

if((((p[l1].x<p[l2].x)||(fabs(p[l1].x-p[l2].x)<eps&&p[l1].y<p[l2].y ))||l2>r )&&l1<=mid)

t[i]=p[l1++];

else t[i]=p[l2++];

}

for(int i=l;i<=r;i++)p[i]=t[i];

}

int main()

{

scanf("%d%lf",&n,&f[0]);

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

scanf("%lf%lf%lf",&p[i].a,&p[i].b,&p[i].r);

p[i].k=-p[i].a/p[i].b;p[i].id=i;

}

sort(p+1,p+n+1);

solve(1,n);

printf("%.3f",f[n]);

return 0;

}

**CDQ分治-例7-分治FFT-HDU5730**

struct cpx {

double x,y;

cpx(double a=0,double b=0):x(a),y(b) {}

};

cpx operator + (cpx a,cpx b) {return cpx(a.x+b.x,a.y+b.y);}

cpx operator - (cpx a,cpx b) {return cpx(a.x-b.x,a.y-b.y);}

cpx operator \* (cpx a,cpx b) {return cpx(a.x\*b.x-a.y\*b.y,a.x\*b.y+a.y\*b.x);}

const int maxn=500005;

char buf[maxn];

int n;

cpx A[maxn],B[maxn];

int dp[MAX],cnt[MAX];

ll num[maxn];

void DFT(cpx\* a,int n,int d=1) {

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

if(i<j) swap(a[i],a[j]);

int k;for(k=(n>>1);i&k;i^=k,k>>=1);

i^=k;

}

for(int m=2;m<=n;m<<=1) {

cpx w=cpx(cos(pi\*2/m\*d),sin(pi\*2/m\*d));

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

cpx s=cpx(1,0);

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

cpx u=a[j],v=s\*a[j+(m>>1)];

a[j]=u+v;a[j+(m>>1)]=u-v;

s=s\*w;

}

}

}

if(d==-1) for(int i=0;i<n;i++) a[i].x=a[i].x/n;

}

void cdq(int l,int r){

if(l==r)return;

int mid=(l+r)/2;

cdq(l,mid);

int n=1;while(n<r-l+1)n<<=1;

for(int i=0;i<n;i++)A[i]=B[i]=cpx();

for(int i=l;i<=mid;i++)A[i-l].x=dp[i];

for(int i=0;i<=r-l;i++)B[i].x=cnt[i];

DFT(A,n);DFT(B,n);

for(int i=0;i<n;i++)A[i]=A[i]\*B[i];

DFT(A,n,-1);

for(int i=mid+1;i<=r;i++)

addi(dp[i],(int)trunc(A[i-l].x+0.5));

cdq(mid+1,r);

}

int main()

{

while(scanf("%d",&n)&&n){

for(int i=1;i<=n;i++)read(cnt[i]),cnt[i]%=MOD,dp[i]=cnt[i];

cdq(1,n);

printf("%d\n",dp[n]%MOD);

}

return 0;

}

**CDQ分治-例8-图连通性CDQ-BZOJ3237**

struct node{

int x,y,tim;

}a[MAX];

struct Query{

int num,c[5];

}qs[MAX];

int tcnt,tp;

bool ans[MAX];

int fa[MAX],stk1[MAX<<2],stk2[MAX<<2];

int find(int x){

if(fa[x]!=x){

int y=fa[x];

stk1[++tp]=x;stk2[tp]=y;

fa[x]=find(fa[x]);

}

return fa[x];

}

void solve(int l,int r){

int now=tp;

if(l==r){

bool re=1;

for(int i=1;i<=qs[l].num;i++)

if(find(a[qs[l].c[i]].x)!=find(a[qs[l].c[i]].y)){re=0;break;}

ans[l]=re;

while(tp!=now)fa[stk1[tp]]=stk2[tp],--tp;

return;

}

int mid=(l+r)>>1;

++tcnt;

for(int i=l;i<=mid;i++)

for(int j=1;j<=qs[i].num;j++)

a[qs[i].c[j]].tim=tcnt;

for(int i=mid+1;i<=r;i++)

for(int j=1;j<=qs[i].num;j++)

if(a[qs[i].c[j]].tim!=tcnt){

int f1=find(a[qs[i].c[j]].x),f2=find(a[qs[i].c[j]].y);

if(f1!=f2){

stk1[++tp]=f1;stk2[tp]=fa[f1];

fa[f1]=f2;

}

}

solve(l,mid);

while(tp!=now)fa[stk1[tp]]=stk2[tp],tp--;

tcnt++;

for(int i=mid+1;i<=r;i++)

for(int j=1;j<=qs[i].num;j++)

a[qs[i].c[j]].tim=tcnt;

for(int i=l;i<=mid;i++)

for(int j=1;j<=qs[i].num;j++)

if(a[qs[i].c[j]].tim!=tcnt){

int f1=find(a[qs[i].c[j]].x),f2=find(a[qs[i].c[j]].y);

if(f1!=f2){

stk1[++tp]=f1;stk2[tp]=fa[f1];

fa[f1]=f2;

}

}

solve(mid+1,r);

}

int main()

{

int n,m,k;

read(n);read(m);

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

for(int i=1;i<=m;i++){read(a[i].x);read(a[i].y);a[i].tim=0;}

read(k);

tcnt=1,tp=0;

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

read(qs[i].num);

for(int j=1;j<=qs[i].num;j++){

read(qs[i].c[j]);

a[qs[i].c[j]].tim=tcnt;

}

}

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

if(a[i].tim!=tcnt){

int f1=find(a[i].x),f2=find(a[i].y);

if(f1!=f2)fa[f1]=f2;

}

solve(1,k);

for(int i=1;i<=k;i++)

puts(ans[i]?"Connected":"Disconnected");

return 0;

}

**CDQ分治-例9-CDQ套CDQ四维偏序-COGS2479**

int n;

struct Item {

int d1,d2,d3,d4,part; // 分别表示每一维的数据，part为第一维重标号之后的值

}a[MAXN];

const int LEFT = 0;

const int RIGHT = 1;

namespace BIT { // 树状数组相关

int arr[MAXN];

inline int lowbit( int num ) { return num&(-num); }

void add( int idx ) {

for( ; idx <= n; idx += lowbit(idx) ) arr[idx]++;

}

int query( int idx ) {

int ans = 0;

for( ; idx; idx -= lowbit(idx) ) ans += arr[idx];

return ans;

}

void clear( int idx ) {

for( ; idx <= n; idx += lowbit(idx) ) arr[idx] = 0;

}

}

ll ans = 0;

Item tmp3d[MAXN];

Item tmp2d[MAXN];

void cdq3d( int L, int R ) { // 对第二维分治，按照第三维合并

if( R-L <= 1 ) return;

int M = (L+R)>>1; cdq3d(L,M); cdq3d(M,R);

int p = L, q = M, o = L;

while( p < M && q < R ) { // 因为第二维是“左边全都是L，右边全都是R”，所以略去第二维的标号

if( tmp2d[p].d3 < tmp2d[q].d3 ) {

if( tmp2d[p].part == LEFT ) BIT::add( tmp2d[p].d4 );

tmp3d[o++] = tmp2d[p++];

} else {

if( tmp2d[q].part == RIGHT ) ans += BIT::query( tmp2d[q].d4 );

tmp3d[o++] = tmp2d[q++];

}

}

while( p < M ) tmp3d[o++] = tmp2d[p++];

while( q < R ) {

if( tmp2d[q].part == RIGHT ) ans += BIT::query( tmp2d[q].d4 );

tmp3d[o++] = tmp2d[q++];

}

for( int i = L; i < R; ++i ) { // 清空树状数组

if( tmp3d[i].part == LEFT ) BIT::clear( tmp3d[i].d4 );

tmp2d[i] = tmp3d[i];

}

}

void cdq2d( int L, int R ) { // 对第一维分治，按照第二维合并

if( R-L <= 1 ) return;

int M = (L+R)>>1; cdq2d(L,M); cdq2d(M,R);

int p = L, q = M, o = L;

while( p < M && q < R ) {

if( a[p].d2 < a[q].d2 ) {

a[p].part = LEFT; // 重标号

tmp2d[o++] = a[p++];

} else {

a[q].part = RIGHT;

tmp2d[o++] = a[q++];

}

}

while( p < M ) {

a[p].part = LEFT;

tmp2d[o++] = a[p++];

}

while( q < R ) {

a[q].part = RIGHT;

tmp2d[o++] = a[q++];

}

for( int i = L; i < R; ++i ) a[i] = tmp2d[i]; // tmp2d为“复制的那一份”

cdq3d(L,R);

}

int main() {

freopen( "partial\_order.in", "r", stdin );

freopen( "partial\_order.out", "w", stdout );

scanf( "%d", &n );

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

a[i].d1 = i;

scanf( "%d", &a[i].d2 );

}

for( int i = 0; i < n; ++i ) scanf( "%d", &a[i].d3 );

for( int i = 0; i < n; ++i ) scanf( "%d", &a[i].d4 );

cdq2d(0,n);

printf( "%lld\n", ans );

return 0;

}

**CDQ分治-例10-带插入查询的四维偏序-HDU5126**

int n;

struct node{

int d1,d2,d3,d4,part,opt,val;

}a[MAX],tmp3d[MAX],tmp2d[MAX];

const int LEFT=0;

const int RIGHT=1;

namespace BIT{

int arr[MAX];

inline int lowbit(int x){return x&-x;}

void add(int idx){

for(;idx<=n;idx+=lowbit(idx))++arr[idx];

}

int query(int idx){

int re=0;

for(;idx;idx-=lowbit(idx))re+=arr[idx];

return re;

}

void clear(int idx){

for(;idx<=n&&arr[idx];idx+=lowbit(idx))arr[idx]=0;

}

}

int ans[MAX];

bool cmp1(int x,int y){

return tmp2d[x].d3<tmp2d[y].d3||(tmp2d[x].d3==tmp2d[y].d3&&tmp2d[x].opt<tmp2d[y].opt);

}

bool cmp2(int x,int y){

return a[x].d2<a[y].d2||(a[x].d2==a[y].d2&&a[x].opt<a[y].opt);

}

void cdq3d(int l,int r){

if(l==r)return;

int mid=(l+r)/2;cdq3d(l,mid);cdq3d(mid+1,r);

int p=l,q=mid+1,o=l;

while(p<=mid&&q<=r){

if(cmp1(p,q)){

if(tmp2d[p].part==LEFT&&!tmp2d[p].opt)

BIT::add(tmp2d[p].d4);

tmp3d[o++]=tmp2d[p++];

}

else{

if(tmp2d[q].part==RIGHT&&tmp2d[q].opt)

ans[tmp2d[q].opt]+=tmp2d[q].val\*BIT::query(tmp2d[q].d4);

tmp3d[o++]=tmp2d[q++];

}

}

while(p<=mid)tmp3d[o++]=tmp2d[p++];

while(q<=r){

if(tmp2d[q].part==RIGHT&&tmp2d[q].opt)

ans[tmp2d[q].opt]+=tmp2d[q].val\*BIT::query(tmp2d[q].d4);

tmp3d[o++]=tmp2d[q++];

}

for(int i=l;i<=r;i++)

{

if(tmp3d[i].part==LEFT&&!tmp3d[i].opt)

BIT::clear(tmp3d[i].d4);

tmp2d[i]=tmp3d[i];

}

}

void cdq2d(int l,int r){

if(l==r)return;

int mid=(l+r)/2;

cdq2d(l,mid);cdq2d(mid+1,r);

int p=l,q=mid+1,o=l;

while(p<=mid&&q<=r){

if(cmp2(p,q)){

a[p].part=LEFT;

tmp2d[o++]=a[p++];

}

else{

a[q].part=RIGHT;

tmp2d[o++]=a[q++];

}

}

while(p<=mid)

a[p].part=LEFT,tmp2d[o++]=a[p++];

while(q<=r)

a[q].part=RIGHT,tmp2d[o++]=a[q++];

for(int i=l;i<=r;i++)a[i]=tmp2d[i];

cdq3d(l,r);

}

bool cmp(const node& lhs,const node& rhs){

if(lhs.d1!=rhs.d1)return lhs.d1<rhs.d1;

else if(lhs.d2!=rhs.d2)return lhs.d2<rhs.d2;

else if(lhs.d3!=rhs.d3)return lhs.d3<rhs.d3;

else if(lhs.d4!=rhs.d4)return lhs.d4<rhs.d4;

else return lhs.opt<rhs.opt;

}

int t;

vector<int>v;

int cnt,opt,acnt;

int main()

{

read(t);

while(t--){

read(n);cnt=acnt=0;

v.clear();

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

read(opt);

if(opt==1){

a[cnt].d1=i;

read(a[cnt].d2);read(a[cnt].d3);read(a[cnt].d4);a[cnt].opt=0;

v.pb(a[cnt].d4);++cnt;

}

else{

int sx,sy,sz,ex,ey,ez;

read(sx);read(sy);read(sz);read(ex);read(ey);read(ez);

v.pb(sz-1);v.pb(ez);++acnt;

a[cnt++]=node{i,ex,ey,ez,0,acnt,1};

a[cnt++]=node{i,sx-1,ey,ez,0,acnt,-1};

a[cnt++]=node{i,ex,sy-1,ez,0,acnt,-1};

a[cnt++]=node{i,ex,ey,sz-1,0,acnt,-1};

a[cnt++]=node{i,sx-1,sy-1,ez,0,acnt,1};

a[cnt++]=node{i,sx-1,ey,sz-1,0,acnt,1};

a[cnt++]=node{i,ex,sy-1,sz-1,0,acnt,1};

a[cnt++]=node{i,sx-1,sy-1,sz-1,0,acnt,-1};

}

}

sort(v.begin(),v.end());

v.erase(unique(v.begin(),v.end()),v.end());

for(int i=0;i<cnt;i++)a[i].d4=lower\_bound(v.begin(),v.end(),a[i].d4)-v.begin()+1;

sort(a,a+cnt,cmp);

n=v.size();

cdq2d(0,cnt-1);

for(int i=1;i<=acnt;i++)printf("%d\n",ans[i]),ans[i]=0;

}

return 0;

}

## KD-TREE

1. **BZOJ2648-插点查询最近曼哈顿距离**

struct arr{

int d[2],min[2],max[2],l,r;

};

int D,root,x,y,ans,n,m,tot,op;

inline int cmp(arr a,arr b){

return a.d[D]<b.d[D]||(a.d[D]==b.d[D]&&a.d[D^1]<b.d[D^1]);

}

arr a[MAX];

inline void up(int k,int s){//ÓÃs¸üÐÂk

a[k].min[0]=min(a[k].min[0],a[s].min[0]);

a[k].max[0]=max(a[k].max[0],a[s].max[0]);

a[k].min[1]=min(a[k].min[1],a[s].min[1]);

a[k].max[1]=max(a[k].max[1],a[s].max[1]);

}

int build(int l,int r,int dd){

D=dd;int mid=(l+r)>>1;

nth\_element(a+l+1,a+mid+1,a+r+1,cmp);

a[mid].min[0]=a[mid].max[0]=a[mid].d[0];

a[mid].min[1]=a[mid].max[1]=a[mid].d[1];

if(l!=mid)a[mid].l=build(l,mid-1,dd^1);

if(mid!=r)a[mid].r=build(mid+1,r,dd^1);

if(a[mid].l)up(mid,a[mid].l);

if(a[mid].r)up(mid,a[mid].r);

return mid;

}

void insert(int k){

int p=root;D=0;

while(1){

up(p,k);

if(a[k].d[D]<=a[p].d[D]){

if(!a[p].l){a[p].l=k;return;}

p=a[p].l;

}

else if(!a[p].r){

a[p].r=k;return;

}

else p=a[p].r;

D^=1;

}

}

int getdis(int k){

int res=0;

if(x<a[k].min[0])res+=a[k].min[0]-x;

if(x>a[k].max[0])res+=x-a[k].max[0];

if(y<a[k].min[1])res+=a[k].min[1]-y;

if(y>a[k].max[1])res+=y-a[k].max[1];

return res;

}

void ask(int k){

int d0=abs(a[k].d[0]-x)+abs(a[k].d[1]-y);

if(d0<ans)ans=d0;

int dl=(a[k].l)?getdis(a[k].l):INF;

int dr=(a[k].r)?getdis(a[k].r):INF;

if(dl<dr){

if(dl<ans)ask(a[k].l);

if(dr<ans)ask(a[k].r);

}

else{

if(dr<ans)ask(a[k].r);

if(dl<ans)ask(a[k].l);

}

}

int main(){

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

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

scanf("%d%d",&a[i].d[0],&a[i].d[1]);

if(n)root=build(1,n,0);

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

scanf("%d",&op);

if(op==1){

++n;

scanf("%d%d",&a[n].d[0],&a[n].d[1]);

a[n].min[0]=a[n].max[0]=a[n].d[0];

a[n].min[1]=a[n].max[1]=a[n].d[1];

insert(n);

}

else if(op==2){

ans=INF;

scanf("%d%d",&x,&y);

ask(root);

printf("%d\n",ans);

}

}

return 0;

}

1. **BZOJ2850-多次询问ax+by≤c的点权值和**

struct kd{

int ls,rs,min[2],max[2],v[2];

ll sum,s;

kd(int a,int b,int c){ls=rs=0;min[0]=max[0]=v[0]=a;min[1]=max[1]=v[1]=b;s=sum=c;}

kd(){}

}t[MAX];

int n,m,D,root,a,b,c;

ll A,B,C;

bool cmp(kd a,kd b){

return (a.v[D]==b.v[D])?(a.v[D^1]<b.v[D^1]):(a.v[D]<b.v[D]);

}

void pushup(int x,int y){

for(int i=0;i<=1;i++){

t[x].max[i]=max(t[x].max[i],t[y].max[i]);

t[x].min[i]=min(t[x].min[i],t[y].min[i]);

}

t[x].sum+=t[y].sum;

}

int build(int l,int r,int d){

if(l>r)return 0;

int mid=(l+r)>>1;

D=d;

nth\_element(t+l,t+mid,t+r,cmp);

t[mid].ls=build(l,mid-1,d^1);t[mid].rs=build(mid+1,r,d^1);

if(t[mid].ls)pushup(mid,t[mid].ls);

if(t[mid].rs)pushup(mid,t[mid].rs);

return mid;

}

int check(int x){

int re=0;

re+=(A\*t[x].min[0]+B\*t[x].min[1]<C);

re+=(A\*t[x].min[0]+B\*t[x].max[1]<C);

re+=(A\*t[x].max[0]+B\*t[x].min[1]<C);

re+=(A\*t[x].max[0]+B\*t[x].max[1]<C);

return re;

}

ll query(int x){

if(!x||!check(x))return 0;

if(check(x)==4)return t[x].sum;

ll re=0;

if(A\*t[x].v[0]+B\*t[x].v[1]<C)re+=t[x].s;

re+=query(t[x].ls)+query(t[x].rs);

return re;

}

int main(){

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

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

scanf("%d%d%d",&a,&b,&c);

t[i]=kd(a,b,c);

}

root=build(1,n,0);

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

scanf("%lld%lld%lld",&A,&B,&C);

printf("%lld\n",query(root));

}

return 0;

}

1. **BZOJ4066-强制在线询问矩形内点权值和**

int opt,D,m=10000;

int sx,sy,ex,ey;

ll lastans;

struct node{

int ls,rs,min[2],max[2],v[2];

ll sum,s;

node(int a,int b,int c){ls=rs=0;min[0]=max[0]=v[0]=a;min[1]=max[1]=v[1]=b;s=sum=c;}

node(){}

int &operator[](int x){

return v[x];

}

friend bool operator==(node a,node b){

return a[0]==b[0]&&a[1]==b[1];

}

friend bool operator<(node a,node b){

return a[D]!=b[D]?a[D]<b[D]:a[D^1]<b[D^1];

}

}tmp[MAX];

bool in(int X1,int Y1,int X2,int Y2)

{

return sx<=X1&&X2<=ex&&sy<=Y1&&Y2<=ey;

}

bool out(int X1,int Y1,int X2,int Y2)

{

return sx>X2||ex<X1||sy>Y2||ey<Y1;

}

struct kdt{

node t[MAX],tem;

int root,n;

void pushup(int x){

int l=t[x].ls,r=t[x].rs;

for(int i=0;i<=1;i++){

t[x].max[i]=t[x].min[i]=t[x][i];

if(l){

t[x].max[i]=max(t[x].max[i],t[l].max[i]);

t[x].min[i]=min(t[x].min[i],t[l].min[i]);

}

if(r){

t[x].max[i]=max(t[x].max[i],t[r].max[i]);

t[x].min[i]=min(t[x].min[i],t[r].min[i]);

}

}

t[x].sum=t[x].s+t[l].sum+t[r].sum;

}

int build(int l,int r,int d){

if(l>r)return 0;

int mid=(l+r)>>1;

D=d;

nth\_element(t+l,t+mid,t+r);

t[mid].ls=build(l,mid-1,d^1);

t[mid].rs=build(mid+1,r,d^1);

pushup(mid);

return mid;

}

void insert(int &k,bool D){

if(!k){

k=++n;

t[k][0]=t[k].min[0]=t[k].max[0]=tem[0];

t[k][1]=t[k].min[1]=t[k].max[1]=tem[1];

}

if(tem==t[k]){

t[k].s+=tem.s;t[k].sum+=tem.s;

return;

}

if(tem[D]<t[k][D])insert(t[k].ls,D^1);

else insert(t[k].rs,D^1);

pushup(k);

}

ll query(int k){

if(!k)return 0;

ll re=0;

if(in(t[k].min[0],t[k].min[1],t[k].max[0],t[k].max[1]))

return t[k].sum;

if(out(t[k].min[0],t[k].min[1],t[k].max[0],t[k].max[1]))

return 0;

if(in(t[k][0],t[k][1],t[k][0],t[k][1]))

re+=t[k].s;

re+=query(t[k].ls);

re+=query(t[k].rs);

return re;

}

}a;

int main(){

int ns;

read(ns);

while(1){

read(opt);

if(opt==3)break;

read(sx);read(sy);

sx^=lastans;sy^=lastans;

if(opt==1){

a.tem.min[0]=a.tem.max[0]=a.tem[0]=sx;

a.tem.min[1]=a.tem.max[1]=a.tem[1]=sy;

read(ex);

ex^=lastans;

a.tem.s=a.tem.sum=ex;

a.insert(a.root,0);

if(a.n==m){

a.root=a.build(1,a.n,0);

m+=10000;

}

}

else if(opt==2){

read(ex);read(ey);

ex^=lastans;ey^=lastans;

lastans=a.query(a.root);

printf("%lld\n",lastans);

}

}

return 0;

}

1. **HDU4347-m维查询曼哈顿最近k个点（非自己写的）**

const int N=50007;

const int K=6;

int n,m;

struct point{

int a[K];

int div; // 按哪个维度划分

bool lef; // 是否是叶子节点

ll dis; // 离询问点的距离。注意这个在读入建树时不会用到，在进入队列时才用到

void print(){

printf("%d",a[0]);

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

printf(" %d",a[i]);

puts("");

}

bool operator < (const point &t) const{

return dis<t.dis;

}

point(){}

point(point &t,ll d){

for (int i=0;i<m;i++) a[i]=t.a[i];

dis=d;

}

}p[N],tar;

int cmp\_NO;

inline bool cmp(point x,point y){

return x.a[cmp\_NO]<y.a[cmp\_NO];

}

inline ll dis(point x,point y){

ll ret=0;

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

ret+=(x.a[i]-y.a[i])\*(x.a[i]-y.a[i]);

return ret;

}

inline void bulid\_kdt(int L,int R,int d){

if (L>R) return;

int mid=(L+R)>>1;

cmp\_NO=d;

nth\_element(p+L,p+mid,p+R+1,cmp);

p[mid].div=d;

if (L==R){

p[L].lef=true;

return;

}

bulid\_kdt(L,mid-1,(d+1)%m);

bulid\_kdt(mid+1,R,(d+1)%m);

}

priority\_queue<point> que;

int num,nownum;

ll ans;

inline void find\_kd(int L,int R){

if (L>R) return;

int mid=(L+R)>>1;

ll d=dis(p[mid],tar);

if (p[mid].lef){//是叶子节点

if (nownum<num){

nownum++;

que.push(point(p[mid],d));

ans=max(ans,d);

}

else if (ans>d){

que.pop();

que.push(point(p[mid],d));

ans=que.top().dis;

}

return;

}

int t=tar.a[p[mid].div]-p[mid].a[p[mid].div];

if (t>0){//优先选这一侧的

find\_kd(mid+1,R);

if (nownum<num){

nownum++;

que.push(point(p[mid],d));

ans=max(ans,d);

find\_kd(L,mid-1);

}

else {

if (ans>d){

que.pop();

que.push(point(p[mid],d));

ans=que.top().dis;

}

if (ans>t\*t)//可能有更小的

find\_kd(L,mid-1);

}

}

else {

find\_kd(L,mid-1);

if (nownum<num){

nownum++;

que.push(point(p[mid],d));

ans=max(ans,d);

find\_kd(mid+1,R);

}

else{

if (ans>d){

que.pop();

que.push(point(p[mid],d));

ans=que.top().dis;

}

if (ans>t\*t)

find\_kd(mid+1,R);

}

}

}

inline void put(){

if (que.empty()) return;

point pp=que.top();

que.pop();

put();

pp.print();

}

int main(){

while (~scanf("%d%d",&n,&m)){

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

for (int j=0;j<m;j++)

scanf("%d",&p[i].a[j]);

p[i].lef=false;

}

bulid\_kdt(0,n-1,0); // 这一步相当于将原数组重新排了个序，先访问到的点放在中间

int q;

scanf("%d",&q);

while (q--){

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

scanf("%d",&tar.a[i]);

while (!que.empty()) que.pop();

scanf("%d",&num);

nownum=0;

ans=-1;

find\_kd(0,n-1);

printf("the closest %d points are:\n",num);

put();

}

}

return 0;

}

## 笛卡尔树

1. **HDU1506-最大子矩形**

int n,top;

int a[MAX],stk[MAX],ls[MAX],rs[MAX],fa[MAX],siz[MAX];

ll ans;

void dfs(int now){

siz[now]=1;

if(ls[now])dfs(ls[now]),siz[now]+=siz[ls[now]];

if(rs[now])dfs(rs[now]),siz[now]+=siz[rs[now]];

ans=max(ans,1LL\*siz[now]\*a[now]);

}

int main(){

while(scanf("%d",&n)&&n){

for(int i=1;i<=n;i++)scanf("%d",&a[i]);

for(int i=1;i<=n;i++)ls[i]=rs[i]=fa[i]=0;

top=0;ans=0;

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

if(!top||a[i]>=a[stk[top]]){

rs[stk[top]]=i;

fa[i]=stk[top];

stk[++top]=i;

}

else{

while(top>=1&&a[stk[top]]>a[i])--top;

rs[stk[top]]=i;fa[i]=stk[top];

ls[i]=stk[top+1];fa[stk[top+1]]=i;

rs[i]=0;

stk[++top]=i;

}

}

int rt=1;while(fa[rt])rt=fa[rt];

dfs(rt);

printf("%lld\n",ans);

}

return 0;

}

1. **HDU6044-给定作为最小的区间求合法排列数**

namespace fastIO {

#define BUF\_SIZE 100000

//fread -> read

bool IOerror = 0;

inline char nc() {

static char buf[BUF\_SIZE], \*p1 = buf + BUF\_SIZE, \*pend = buf + BUF\_SIZE;

if(p1 == pend) {

p1 = buf;

pend = buf + fread(buf, 1, BUF\_SIZE, stdin);

if(pend == p1) {

IOerror = 1;

return -1;

}

}

return \*p1++;

}

inline bool blank(char ch) {

return ch == ' ' || ch == '\n' || ch == '\r' || ch == '\t';

}

inline void read(int &x) {

char ch;

while(blank(ch = nc()));

if(IOerror)

return;

for(x = ch - '0'; (ch = nc()) >= '0' && ch <= '9'; x = x \* 10 + ch - '0');

}

#undef BUF\_SIZE

};

int fi[MAX],inv[MAX];

int Case,n,lo;

bool ava;

void init(){

fi[0]=1;

for(int i=1;i<MAX;i++)fi[i]=mul(fi[i-1],i);

inv[MAX-1]=powMM(fi[MAX-1],MOD-2,MOD);

for(int i=MAX-2;i>=0;i--)inv[i]=mul(inv[i+1],i+1);

}

int C(int a,int b){

return mul(fi[a],mul(inv[b],inv[a-b]));

}

struct node{

int l,r,id;

bool operator<(const node &z)const{

return l!=z.l?l<z.l:r>z.r;

}

}a[MAX];

int dfs(int l,int r){

if(!ava)return 0;

if(l>r)return 1;

if(a[lo].l!=l||a[lo].r!=r){ava=0;return 0;}

int mid=a[lo++].id;

int re=mul(C(r-l,mid-l),dfs(l,mid-1));

muli(re,dfs(mid+1,r));

return re;

}

int main(){

init();

while(1){

fastIO::read(n);

if(fastIO::IOerror)break;

for(int i=1;i<=n;i++)fastIO::read(a[i].l);

for(int i=1;i<=n;i++)fastIO::read(a[i].r),a[i].id=i;

sort(a+1,a+1+n);

ava=1;lo=1;

printf("Case #%d: %d\n", ++Case, dfs(1,n));

}

return 0;

}

## 分块

1. **ZOJ4053-nsqrtn查询区间逆序对个数**

int SIZ=150;

const int CNT=158;

int t,n,tot;

int a[MAX],bel[MAX],cnt[MAX],st[MAX],en[MAX];

int lim;

ll tp[MAX][CNT];

ll pre[MAX],bck[MAX];

ll dp[CNT][CNT];

ll ans;

int s1,s2;

ll q1[MAX],q2[MAX];

struct BIT{

int a[MAX];

inline int lowbit(int x){return x&-x;}

inline void ins\_bck(int x){//��׺��

for(;x<=lim;x+=lowbit(x))++a[x];

}

inline void ins\_pre(int x){

for(;x>0;x-=lowbit(x))++a[x];

}

inline void del\_pre(int x){

for(;x>0;x-=lowbit(x))--a[x];

}

inline int find\_pre(int x){//С�ڵ���x�����ֵ

while (x&&!a[x]) x^=lowbit(x);

if (!x) return 0;

int t=lowbit(x)>>1,y=a[x];

while (t){

if (y-a[x-t]) y-=a[x-t];

else{y=a[x-t];x=x-t;}

t>>=1;

}

return x;

}

inline int find\_bck(int x){//���ڵ���x����Сֵ

while (x<=lim&&!a[x]) x+=lowbit(x);

if (x>lim) return lim+1;

int t=lowbit(x)>>1,y=a[x];

while (t){

if (y-a[x+t]) y-=a[x+t];

else{y=a[x+t];x=x+t;}

t>>=1;

}

return x;

}

inline int query\_pre(int x){

int ans=0;

while(x){

ans+=a[x];x-=lowbit(x);

}

return ans;

}

inline int query\_bck(int x){

int ans=0;

while(x<=lim){

ans+=a[x];x+=lowbit(x);

}

return ans;

}

void clear\_bck(int x){

while(x<=lim){

if(a[x])a[x]=0;else break;

x+=lowbit(x);

}

}

void clear\_pre(int x){

while(x){

if(a[x])a[x]=0;else break;

x-=lowbit(x);

}

}

void clear(){

memset(a,0,sizeof(a));

}

}pts,bit,pts2;//pts为加的点

struct node{

int val,id;

bool operator<(const node &z)const{

if(val!=z.val)return val<z.val;

else return id<z.id;

}

}c[MAX];

void cal1(){

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

j=i;

while(j<=n&&c[j].val==c[i].val){

int who=c[j].id,st=bel[who]+1;

while(st<=tot)tp[who][st]=cnt[st],++st;

++j;

}

for(int s=i;s<j;++s)++cnt[bel[c[s].id]];

}

for(int i=1;i<=tot;i++)cnt[i]=0;

}

void cal2(){

for(int j,i=n;i>=1;i=j){

j=i;

while(j>=1&&c[j].val==c[i].val){

int who=c[j].id,st=bel[who]-1;

while(st>=1)tp[who][st]+=cnt[st],--st;

--j;

}

for(int s=j+1;s<=i;++s)++cnt[bel[c[s].id]];

}

for(int i=1;i<=tot;i++)cnt[i]=0;

}

int qs(int id1,int id2){

int l1=st[id1],r1=min(n,en[id1]),l2=st[id2],r2=min(n,en[id2]);

int lo1=l1,lo2=l2,re=0;

while(lo1<=r1&&lo2<=r2){

if(c[lo1].val<=c[lo2].val)++lo1;

else{++lo2;re+=r1-lo1+1;}

}

return re;

}

void init(){

for(int i=1;i<=tot;i++)

for(int j=st[i]-1;j>=1;j--)

if(j%SIZ)tp[j][i]+=tp[j+1][i];

for(int i=1;i<=tot;i++)

for(int j=en[i]+1;j<=n;j++)

if(j%SIZ!=1)tp[j][i]+=tp[j-1][i];

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

for(int j=1;j<=tot;j++){

tp[i][j]+=tp[i][j-1];

}

for(int i=1;i<=tot;i++){

bit.clear();

int l=0;

for(int j=st[i];j<=n&&j<=en[i];j++){

l+=bit.query\_bck(a[j]+1);

pre[j]=l;//sum记录前比其大的

bit.ins\_pre(a[j]);

}

bit.clear();

l=0;

for(int j=min(n,en[i]);j>=st[i];j--){

l+=bit.query\_pre(a[j]-1);

bck[j]=l;//sum2记录后比其小的

bit.ins\_bck(a[j]);

}

}

//每个块内排序

for(int i=1;i<=tot;i++){

int l=st[i],r=en[i]>n?n:en[i];

for(int j=l;j<=r;j++){

c[j].id=j;c[j].val=a[j];

}

sort(c+l,c+r+1);

}

for(int i=1;i<=tot;i++)dp[i][i]=pre[en[i]>n?n:en[i]];

for(int i=1;i<=tot-1;i++)

for(int j=i+1;j<=tot;j++)

dp[i][j]=qs(i,j);

for(int i=tot;i>=1;i--)

for(int j=i+1;j<=tot;j++)

dp[i][j]+=dp[i][j-1]+dp[i+1][j]-dp[i+1][j-1];

}

ll eme(){

int i=1,j=1;

ll t=0;

while(i<=s1&&j<=s2){

if(q1[i]<=q2[j])++i;

else{

++j;

t+=s1-i+1;

}

}

return t;

}

ll ask(int x,int l,int r){

s1=s2=0;

ll t=pre[r];

if(l!=st[x])t-=pre[l-1];

for(int i=st[x];i<=en[x]&&i<=n;i++)

if (c[i].id<l)q1[++s1]=c[i].val;

for(int i=st[x];i<=en[x]&&i<=n;i++)

if (c[i].id>=l&&c[i].id<=r) q2[++s2]=c[i].val;

t-=eme();

return t;

}

ll query(int l,int r){

if(l>r)return 0LL;

s1=s2=0;

int bl=bel[l],br=bel[r];

if(bl==br)

return ask(bl,l,r);

ll re=dp[bl+1][br-1];

re+=bck[l];re+=pre[r];

for(int i=st[bl];i<=en[bl]&&i<=n;i++)

if (c[i].id>=l) q1[++s1]=c[i].val;

for(int i=st[br];i<=en[br]&&i<=n;i++)

if (c[i].id<=r) q2[++s2]=c[i].val;

re+=tp[l][br-1]-tp[l][bl]+tp[r][br-1]-tp[r][bl]+eme();

return re;

}

multiset<ll>z;

multiset<ll>::iterator it;

int main(){

read(t);

while(t--){

read(n);lim=n+1;

SIZ=max(350,(n+CNT-2)/(CNT-1));

for(int i=1;i<=CNT-1;i++)st[i]=(i-1)\*SIZ+1,en[i]=i\*SIZ;

z.clear();

for(int i=1;i<=n;i++)read(a[i]),bel[i]=1+(i-1)/SIZ,c[i].val=a[i],c[i].id=i;

tot=1+(n-1)/SIZ;

sort(c+1,c+1+n);

cal1();cal2();

init();

pts.clear();pts2.clear();

ans=query(1,n);

printf("%lld",ans);

z.insert(-ans);

pts2.ins\_pre(n+1);

for(int q=1;q<=n;q++){

printf(" ");

ll tem;

read(tem);

if(q==n)break;

tem^=ans;

assert(tem>=1&&tem<=n);

int l=pts.find\_pre(tem),r=pts2.find\_bck(tem);

ll lin=-query(l+1,r-1);

it=lower\_bound(z.begin(),z.end(),lin);

z.erase(it);

if(l+1<=tem-1)

z.insert(-query(l+1,tem-1));

if(tem+1<=r-1)

z.insert(-query(tem+1,r-1));

ans=-(\*z.begin());

printf("%lld",ans);

pts.ins\_bck(tem);pts2.ins\_pre(tem);

}

printf("\n");

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

for(int j=1;j<=tot;j++)tp[i][j]=0;

for(int i=1;i<=tot;i++)

for(int j=1;j<=tot;j++)dp[i][j]=0;

for(int i=1;i<=n;i++)pre[i]=bck[i]=0;

}

return 0;

}

## 莫队

1. **求区间mex**

int n,m;

int a[MAX];

/\*

记录莫队中的每一个查询的结构体

\*/

struct node

{

int al,ar,bel,id;

}qs[MAX];

/\*

对查询排序

\*/

bool cmp(node x,node z)

{

return x.bel==z.bel?x.ar<z.ar:x.bel<z.bel;

}

int cnt[MAX],sum[MAX];

int ans[MAX];

/\*

莫队示例：

求数组区间mex

\*/

int main()

{

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

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

{

scanf("%d",&a[i]);

if(a[i]>n)a[i]=n;

}

int siz=sqrt(n);

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

{

scanf("%d%d",&qs[i].al,&qs[i].ar);

qs[i].bel=(qs[i].al-1)/siz;qs[i].id=i;

}

sort(qs+1,qs+1+m,cmp);

/\*

对于下标从1开始的情况

lol初始设置为1 lor初始设置为0

\*/

int lol=1,lor=0;siz=sqrt(n+1);

/\*

每次莫队调整l和r的位置

\*/

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

{

while(lol>qs[i].al)--lol,sum[a[lol]/siz]+=(!cnt[a[lol]]),++cnt[a[lol]];

while(lor<qs[i].ar)++lor,sum[a[lor]/siz]+=(!cnt[a[lor]]),++cnt[a[lor]];

while(lol<qs[i].al)--cnt[a[lol]],sum[a[lol]/siz]-=(!cnt[a[lol]]),++lol;

while(lor>qs[i].ar)--cnt[a[lor]],sum[a[lor]/siz]-=(!cnt[a[lor]]),--lor;

int st=0;

for(int i=0;i<=siz;i++,st+=siz)if(sum[i]<siz)break;

while(cnt[st])++st;

ans[qs[i].id]=st;

}

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

printf("%d\n",ans[i]);

}

1. **求区间的所有子区间gcd和-hdu5381**

int rgcd[MAX][15],a[MAX];

vector<pii>vr[MAX],vl[MAX];

int t,n,m;

ll an[MAX];

struct node

{

int al,ar,bel,id;

bool operator <(const node &t)const

{

if(bel==t.bel)return ar<t.ar;

else return bel<t.bel;

}

}qs[MAX];

void init()

{

for(int i=1;i<=n;i++)rgcd[i][0]=a[i];

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

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

if(j+(1<<i)-1<=n)rgcd[j][i]=gcd(rgcd[j][i-1],rgcd[j+(1<<i-1)][i-1]);

}

int query(int l,int r)

{

int k=(int)log2(r-l+1);

return gcd(rgcd[l][k],rgcd[r-(1<<k)+1][k]);

}

//s作为右端点 搜索向左gcd为t的最远位置

int Rsearch(int s,int l,int r,int t){

int re,mid;

while(l<=r)

{

mid=(l+r)/2;

if(query(mid,s)==t){

re=mid;r=mid-1;

}

else l=mid+1;

}

return re;

}

int Lsearch(int s,int l,int r,int t){

int re,mid;

while(l<=r)

{

mid=(l+r)/2;

if(query(s,mid)==t){

re=mid;l=mid+1;

}

else r=mid-1;

}

return re;

}

//计算s为右端点的贡献 t为当前区间的左端点

ll rcal(int s,int t)

{

ll re=0;

int lo=s;

for(int i=0;i<vr[s].size();i++){

re+=1LL\*(lo-max(t,vr[s][i].second)+1)\*(vr[s][i].first);

lo=vr[s][i].second-1;

if(lo<t)break;

}

return re;

}

ll lcal(int s,int t)

{

ll re=0;

int lo=s;

for(int i=0;i<vl[s].size();i++){

re+=1LL\*(min(t,vl[s][i].second)-lo+1)\*vl[s][i].first;

lo=vl[s][i].second+1;

if(lo>t)break;

}

return re;

}

int main()

{

read(t);

while(t--){

read(n);int siz=sqrt(n);

for(int i=1;i<=n;i++)read(a[i]);

init();

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

int r=i;

vl[i].clear();

while(r<=n){

int who=query(i,r);

r=Lsearch(i,r,n,who);

vl[i].pb(mp(who,r));

++r;

}

}

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

{

int l=i;

vr[i].clear();

while(l>=1){

int who=query(l,i);

l=Rsearch(i,1,l,who);

vr[i].pb(mp(who,l));--l;

}

}

read(m);ll sum=0;

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

read(qs[i].al);read(qs[i].ar);qs[i].id=i;

qs[i].bel=(qs[i].al-1)/siz;

}

sort(qs,qs+m);

int lol=1,lor=0;

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

while(lol>qs[i].al)--lol,sum+=lcal(lol,lor);

while(lor<qs[i].ar)++lor,sum+=rcal(lor,lol);

while(lol<qs[i].al)sum-=lcal(lol,lor),++lol;

while(lor>qs[i].ar)sum-=rcal(lor,lol),--lor;

an[qs[i].id]=sum;

}

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

printf("%lld\n",an[i]);

}

return 0;

}

## 线段树

**HDU6183-灵活运用线段树**

int rt[100],l[MAX],r[MAX],v[MAX];

int tot,y1,y2,X,opt;

bool flag;

void ins(int &t,int L,int R,int y,int x)

{

if(!t){

t=++tot;

v[t]=x;

}

if(v[t]>x)v[t]=x;

if(L==R)return;

int mid=(L+R)/2;

if(y<=mid)ins(l[t],L,mid,y,x);

else ins(r[t],mid+1,R,y,x);

}

void ask(int t,int L,int R)

{

if(flag||!t)return;

if(y1<=L&&R<=y2){

if(v[t]<=X)flag=1;

return;

}

int mid=(L+R)/2;

if(y1<=mid)ask(l[t],L,mid);

if(y2>mid)ask(r[t],mid+1,R);

}

int main()

{

while(~scanf("%d",&opt))

{

if(opt==3)return 0;

else if(opt==0){

for(int i=0;i<=50;i++)rt[i]=0;

for(int i=1;i<=tot;i++)l[i]=r[i]=0;

tot=0;

}

else if(opt==1){

int x,y,v;

read(x);read(y);read(v);ins(rt[v],1,1000000,y,x);

}

else if(opt==2){

read(X);read(y1);read(y2);

int ans=0;

for(int i=0;i<=50;i++){

flag=0;

ask(rt[i],1,1000000);

if(flag)++ans;

}

printf("%d\n",ans);

}

}

return 0;

}

## 整体二分

**整体二分-例1-静态第k小-POJ2104**

int n,m,ecnt;

int ans[MAX],ar[MAX];

inline int lowbit(int x){return x&(-x);}

void adds(int lo,int v){

while(lo<=n)ar[lo]+=v,lo+=lowbit(lo);

}

int sum(int lo){

int re=0;

while(lo)re+=ar[lo],lo-=lowbit(lo);

return re;

}

void clear(int lo){

while(lo<=n&&ar[lo])ar[lo]=0,lo+=lowbit(lo);

}

struct Query{

int x,y,k,id,opt;

}qs[MAX],ql[MAX],qr[MAX];

inline void addq(int x,int y,int k,int id,int opt){

qs[++ecnt]=Query{x,y,k,id,opt};

}

void solve(int sl,int sr,int l,int r){

if(sl>sr)return;

if(l==r){

for(int i=sl;i<=sr;i++)if(qs[i].opt)ans[qs[i].id]=l;

return;

}

int mid=(l+r)>>1;

int pl=0,pr=0;

for(int i=sl;i<=sr;i++){

if(!qs[i].opt){

if(qs[i].x<=mid)adds(qs[i].id,1),ql[pl++]=qs[i];

else qr[pr++]=qs[i];

}

else{

int cnt=sum(qs[i].y)-sum(qs[i].x-1);

if(cnt>=qs[i].k)ql[pl++]=qs[i];

else qs[i].k-=cnt,qr[pr++]=qs[i];

}

}

for(int i=0;i<pl;i++)if(!ql[i].opt)clear(ql[i].id);

for(int i=0;i<pl;i++)qs[sl+i]=ql[i];

for(int i=0;i<pr;i++)qs[sl+pl+i]=qr[i];

solve(sl,sl+pl-1,l,mid);solve(sl+pl,sr,mid+1,r);

}

int main(){

read(n);read(m);

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

read(u);addq(u,-1,-1,i,0);

}

for(int l,r,k,i=1;i<=m;i++){

read(l);read(r);read(k);

addq(l,r,k,i,1);

}

solve(1,ecnt,-INF,INF);

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

printf("%d\n",ans[i]);

return 0;

}

**整体二分-例2-动态第k小-ZOJ2112**

int n,m,ecnt;

char op[20];

int ans[MAX],ar[MAX];

inline int lowbit(int x){return x&(-x);}

void adds(int lo,int v){

while(lo<=n)ar[lo]+=v,lo+=lowbit(lo);

}

int sum(int lo){

int re=0;

while(lo)re+=ar[lo],lo-=lowbit(lo);

return re;

}

//void clear(int lo){

// while(lo<=n&&ar[lo])ar[lo]=0,lo+=lowbit(lo);

//}

struct Query{

int x,y,k,id,opt;

}qs[MAX],ql[MAX],qr[MAX];

inline void addq(int x,int y,int k,int id,int opt){

qs[++ecnt]=Query{x,y,k,id,opt};

}

void solve(int sl,int sr,int l,int r){

if(sl>sr)return;

if(l==r){

for(int i=sl;i<=sr;i++)if(qs[i].opt)ans[qs[i].id]=l;

return;

}

int mid=(l+r)>>1;

int pl=0,pr=0;

for(int i=sl;i<=sr;i++){

if(!qs[i].opt){

if(qs[i].x<=mid)adds(qs[i].id,qs[i].y),ql[pl++]=qs[i];

else qr[pr++]=qs[i];

}

else{

int cnt=sum(qs[i].y)-sum(qs[i].x-1);

if(cnt>=qs[i].k)ql[pl++]=qs[i];

else qs[i].k-=cnt,qr[pr++]=qs[i];

}

}

for(int i=0;i<pl;i++)if(!ql[i].opt)adds(ql[i].id,-ql[i].y);

for(int i=0;i<pl;i++)qs[sl+i]=ql[i];

for(int i=0;i<pr;i++)qs[sl+pl+i]=qr[i];

solve(sl,sl+pl-1,l,mid);solve(sl+pl,sr,mid+1,r);

}

int a[MAX];

int t,aidx;

int main(){

read(t);

while(t--)

{

ecnt=aidx=0;

read(n);read(m);

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

read(a[i]);addq(a[i],1,-1,i,0);

}

for(int l,r,k,i=1;i<=m;i++){

scanf("%s",op);

if(op[0]=='Q')

{

read(l);read(r);read(k);

addq(l,r,k,++aidx,1);

}

else{

read(l);read(r);

addq(a[l],-1,-1,l,0);

a[l]=r;

addq(a[l],1,-1,l,0);

}

}

solve(1,ecnt,0,INF);

for(int i=1;i<=aidx;i++)

printf("%d\n",ans[i]);

}

return 0;

}

## 字典树

**可持久化字典树**

int a[MAX],b[MAX],rt[MAX];

int bin[30];//2^i

struct trie{

int cnt;//结点id

int ch[MAX\*25][2],sum[MAX\*25];

int insert(int x,int val){//插入值val x为前版本的结点 返回新的根节点

int re,y;re=y=++cnt;

for(int i=23;i>=0;i--)//固定位数 保证位对齐

{

ch[y][0]=ch[x][0];ch[y][1]=ch[x][1];

sum[y]=sum[x]+1;

int t=val&bin[i];t>>=i;

x=ch[x][t];

ch[y][t]=++cnt;

y=ch[y][t];

}

sum[y]=sum[x]+1;

return re;

}

int query(int l,int r,int val){//询问在l、r结点之间 与val异或最大的结果

int re=0;

for(int i=23;i>=0;i--){

int t=val&bin[i];t>>=i;

if(sum[ch[r][t^1]]-sum[ch[l][t^1]])

re+=bin[i],r=ch[r][t^1],l=ch[l][t^1];

else r=ch[r][t],l=ch[l][t];

}

return re;

}

}trie;

**裸的字典树**

/\*极其容易MLE版本 \*/

struct Trie

{

int ch[MAX\_NODE][sigma\_size];//点数、“字母”种数

int val[MAX\_NODE];

int num;//结点总数

Trie(){num=1;memset(ch[0],0,sizeof(ch[0]));}//初始时仅有根节点

int idx(char c)//返回对应字符的编号

{

return c-'a';

}

/\*

clear函数，初始化trie

\*/

void clear() { num = 1; memset(ch[0], 0, sizeof(ch[0])); }

/\*

插入字符串s,附加信息为v。注意v必须非0，因为0代表：本结点不是单词结点

\*/

void insert(char \*s,int v)

{

int u=0,len=strlen(s);

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

{

int c=idx(s[i]);

if(!ch[u][c])//结点不存在

{

memset(ch[num],0,sizeof(ch[num]));

val[num]=0;//中间节点的附加信息为0

ch[u][c]=num++;//新建结点

}

u=ch[u][c];//往下走

}

val[u]=v;//字符串的最后一个字符的附加信息为v

}

/\*

查询字符串的“附加信息”

查询过程中间中断返回0

\*/

int check(char \*s)

{

int u=0,len=strlen(s);

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

{

int c=idx(s[i]);

if(!ch[u][c])

return 0;

u=ch[u][c];

}

return val[u];

}

/\*

找字符串s的长度不超过len的前缀

\*/

void find\_prefixes(const char \*s,int len,vector <int> &ans)

{

int u=0;

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

{

if(s[i]=='\0')

break;

int c=idx(s[i]);

if(!ch[u][c])

break;

u=ch[u][c];

if(val[u]!=0)//过程中所有找到的全都push进去

ans.push\_back(val[u]);

}

}

};

/\*不易MLE版本 \*/

struct Trie {

bool isWord;

Trie\* child[26];

Trie(bool isWord):isWord(isWord)

{

memset(child,0,sizeof(child));

}

void addWord(string &s)

{

Trie\*cur =this;

for(char c: s)

{

Trie\* next=cur->child[c-'a'];

if(next==nullptr)

next=cur->child[c-'a']=new Trie(false);

cur=next;

}

cur->isWord=true;

}

int checkstr(string s)//返回字符串在字典树中最长前缀

{

Trie\*cur=this;int re=0;

for(char c:s)

{

Trie\* next=cur->child[c-'a'];

if(next==nullptr)break;

else{cur=next;++re;}

}

return re;

}

~Trie()

{

for(int i=0;i<26;++i)

{

if(child[i])

delete child[i];

}

}

};

//Trie \*tri=new Trie(0); 以此来建Trie

**BZOJ3261-可持久化字典树例1**

/\*

题意：

给定一个非负整数序列 {a}，初始长度为 N。

有 M个操作，有以下两种操作类型：

1 、A x：添加操作，表示在序列末尾添加一个数 x，序列的长度 N+1。

2 、Q l r x：询问操作，你需要找到一个位置 p，满足 l<=p<=r，使得：

a[p] xor a[p+1] xor … xor a[N] xor x 最大，输出最大是多少。

每次加点将trie树的这条链的权都+1

修改当然是新建一个结点（类可持久化线段树）

然后查询的时候判断一个结点存在，只要做区间减法判权是否非0

即若sum[r]-sum[l-1]=0则该结点不存在

实现的时候数列开始加入一个数0会比较好处理

\*/

int bin[30];

int n,m;

int a[MAX],b[MAX],rt[MAX];

struct trie{

int cnt;//结点id

int ch[MAX\*25][2],sum[MAX\*25];

int insert(int x,int val){

int re,y;re=y=++cnt;

for(int i=23;i>=0;i--)//固定位数

{

ch[y][0]=ch[x][0];ch[y][1]=ch[x][1];

sum[y]=sum[x]+1;

int t=val&bin[i];t>>=i;

x=ch[x][t];

ch[y][t]=++cnt;

y=ch[y][t];

}

sum[y]=sum[x]+1;

return re;

}

int query(int l,int r,int val){

int tmp=0;

for(int i=23;i>=0;i--){

int t=val&bin[i];t>>=i;

if(sum[ch[r][t^1]]-sum[ch[l][t^1]])

tmp+=bin[i],r=ch[r][t^1],l=ch[l][t^1];

else r=ch[r][t],l=ch[l][t];

}

return tmp;

}

}trie;

int main()

{

bin[0]=1;

for(int i=1;i<30;i++)bin[i]=bin[i-1]<<1;

read(n);read(m);

++n;

for(int i=2;i<=n;i++)read(a[i]);

for(int i=1;i<=n;i++)b[i]=b[i-1]^a[i];

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

rt[i]=trie.insert(rt[i-1],b[i]);

char ch[5];

int l,r,x;

while(m--)

{

scanf("%s",ch);

if(ch[0]=='A'){

++n;

read(a[n]);b[n]=b[n-1]^a[n];

rt[n]=trie.insert(rt[n-1],b[n]);

}

else{

read(l);read(r);read(x);

printf("%d\n",trie.query(rt[l-1],rt[r],b[n]^x));

}

}

return 0;

}

**HDU4757-可持久化字典树例2**

/\*

题意：给定树 每次问(u,v)路径上的点权值与x异或中的最大值

做法就是按dfs序插点就好了

\*/

struct node

{

int to,nxt;

}edg[MAX<<2];

int n,m,ecnt;

int a[MAX],bin[25];

int fa[MAX][20],dep[MAX],rt[MAX];

int h[MAX];

void add\_edge(int u,int v)

{

edg[++ecnt]=node{v,h[u]};

h[u]=ecnt;

}

struct trie{

int cnt;

int ch[MAX\*20][2],sum[MAX\*20];

int insert(int pre,int val){

int re,now;re=now=++cnt;

for(int i=18;i>=0;i--){

ch[now][0]=ch[pre][0];ch[now][1]=ch[pre][1];

sum[now]=sum[pre]+1;

int t=val&bin[i];t>>=i;

pre=ch[pre][t];

ch[now][t]=++cnt;

now=ch[now][t];

}

sum[now]=sum[pre]+1;

return re;

}

int query(int u,int v,int anc,int w,int val){

int re=0;

for(int i=18;i>=0;i--){

int t=val&bin[i];t>>=i;

if(sum[ch[u][t^1]]+sum[ch[v][t^1]]-sum[ch[anc][!t]]-sum[ch[w][!t]]>0){

re+=bin[i];u=ch[u][!t];v=ch[v][!t];w=ch[w][!t];anc=ch[anc][!t];

}

else{

u=ch[u][t];v=ch[v][t];anc=ch[anc][t];w=ch[w][t];

}

}

return re;

}

}trie;

void dfs(int u,int pre)

{

fa[u][0]=pre;

rt[u]=trie.insert(rt[pre],a[u]);

for(int i=1;i<=18;i++)fa[u][i]=fa[fa[u][i-1]][i-1];

for(int i=h[u];i;i=edg[i].nxt){

int to=edg[i].to;

if(to==pre)continue;

dep[to]=dep[u]+1;

dfs(to,u);

}

}

int lca(int u,int v)

{

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

int cha=dep[u]-dep[v];

for(int i=18;i>=0;i--)if(cha&(1<<i))u=fa[u][i];

if(u==v)return u;

for(int i=18;i>=0;i--)

if(fa[u][i]!=fa[v][i])u=fa[u][i],v=fa[v][i];

return fa[u][0];

}

int main()

{

bin[0]=1;for(int i=1;i<=20;i++)bin[i]=bin[i-1]<<1;

while(~scanf("%d%d",&n,&m))

{

trie.cnt=0;

ecnt=0;memset(h,0,sizeof(h));

for(int i=1;i<=n;i++)read(a[i]);

for(int u,v,i=1;i<n;i++){

read(u);read(v);add\_edge(u,v);add\_edge(v,u);

}

dfs(1,0);

for(int u,v,x,i=1;i<=m;i++){

read(u);read(v);read(x);

int c=lca(u,v),anc=fa[c][0];

printf("%d\n",trie.query(rt[u],rt[v],rt[c],rt[anc],x));

}

}

return 0;

}

## 主席树

**主席树——区间第k大值**

const int maxn=1e5+5;

struct node

{

int l,r,sum;

}T[maxn\*40];

int rt[maxn];

int cnt;

int n,m;

void update(int l,int r,int &x,int y,int pos)

{

T[++cnt]=T[y];++T[cnt].sum;x=cnt;

if(l==r)return;

int mid=(l+r)>>1;

if(pos<=mid)update(l,mid,T[x].l,T[y].l,pos);

else update(mid+1,r,T[x].r,T[y].r,pos);

}

int query(int l,int r,int x,int y,int k)

{

if(l==r)return l;

int mid=(l+r)>>1;

int sum=T[T[y].l].sum-T[T[x].l].sum;

if(sum>=k)return query(l,mid,T[x].l,T[y].l,k);

else return query(mid+1,r,T[x].r,T[y].r,k-sum);

}

int a[maxn];

vector<int>v;

int getid(int x)//返回离散化后的排名（按值从小到大）

{

return lower\_bound(v.begin(),v.end(),x)-v.begin()+1;

}

int main()

{

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

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

{

scanf("%d",&a[i]);v.pb(a[i]);

}

sort(v.begin(),v.end());v.erase(unique(v.begin(),v.end()),v.end());

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

update(1,n,rt[i],rt[i-1],getid(a[i]));

for(int u,vs,k,i=1;i<=m;i++)

{

scanf("%d%d%d",&u,&vs,&k);

printf("%d\n",v[query(1,n,rt[u-1],rt[vs],k)-1]);

}

return 0;

}

**动态第k小**

struct node

{

int kind,x,y,z;

}query[MAX];

int ts,n,q,m,tot;

int a[MAX],od[MAX],t[MAX];

int T[MAX],lson[M],rson[M],s[M],bs[MAX];

int use[MAX];

char opt[10];

inline int lowbit(int x)

{

return x&(-x);

}

int build(int l,int r)

{

int rt=tot++;

s[rt]=0;

if(l!=r){

int mid=(l+r)/2;

//原本下一行没有lson\rson的赋值 感觉不对？

lson[rt]=build(l,mid);rson[rt]=build(mid+1,r);

}

return rt;

}

int Insert(int rt,int pos,int val)

{

int re=tot++,l=0,r=m-1,newrt=re;s[newrt]=s[rt]+val;

while(l<r)

{

int mid=(l+r)/2;

if(pos<=mid){

lson[newrt]=tot++;rson[newrt]=rson[rt];

newrt=lson[newrt];rt=lson[rt];r=mid;

}

else{

lson[newrt]=lson[rt];rson[newrt]=tot++;

newrt=rson[newrt];rt=rson[rt];l=mid+1;

}

s[newrt]=s[rt]+val;

}

return re;

}

int sum(int x)

{

int re=0;

for(int i=x;i>0;i-=lowbit(i))

{

re+=s[lson[use[i]]];

}

return re;

}

int Query(int ql,int qr,int k)

{

int l=0,r=m-1,mid,rtl=T[ql-1],rtr=T[qr];

for(int i=ql-1;i>0;i-=lowbit(i))use[i]=bs[i];

for(int i=qr;i>0;i-=lowbit(i))use[i]=bs[i];

while(l<r)

{

int mid=(l+r)/2;

int tem=sum(qr)-sum(ql-1)+s[lson[rtr]]-s[lson[rtl]];

if(tem>=k){

for(int i=ql-1;i>0;i-=lowbit(i))use[i]=lson[use[i]];

for(int i=qr;i>0;i-=lowbit(i))use[i]=lson[use[i]];

rtl=lson[rtl];rtr=lson[rtr];

r=mid;

}

else{

k-=tem;

for(int i=ql-1;i>0;i-=lowbit(i))use[i]=rson[use[i]];

for(int i=qr;i>0;i-=lowbit(i))use[i]=rson[use[i]];

rtl=rson[rtl];rtr=rson[rtr];l=mid+1;

}

}

return l;

}

void upd(int x,int who,int d)

{

while(x<=n)

{

bs[x]=Insert(bs[x],who,d);

x+=lowbit(x);

}

}

inline int lo(int x){return lower\_bound(t,t+m,x)-t;}

int main()

{

scanf("%d",&ts);

while(ts--)

{

scanf("%d%d",&n,&q);m=tot=0;

for(int i=1;i<=n;i++)scanf("%d",&a[i]),t[m++]=a[i];

for(int i=1;i<=q;i++)

{

scanf("%s",opt);

if(opt[0]=='Q'){

query[i].kind=1;scanf("%d%d%d",&query[i].x,&query[i].y,&query[i].z);

}

else{

query[i].kind=0;scanf("%d%d",&query[i].x,&query[i].y);t[m++]=query[i].y;

}

}

sort(t,t+m);m=unique(t,t+m)-t;

T[0]=build(0,m-1);

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

T[i]=Insert(T[i-1],lo(a[i]),1);

}

for(int i=1;i<=n;i++)bs[i]=T[0];

for(int i=1;i<=q;i++)

{

if(query[i].kind)printf("%d\n",t[Query(query[i].x,query[i].y,query[i].z)]);

else{

upd(query[i].x,lo(a[query[i].x]),-1);

upd(query[i].x,lo(query[i].y),1);

a[query[i].x]=query[i].y;

}

}

}

return 0;

}

**可持久化动态区间和**

int n,m,tot,TM;

int T[MAX<<2],lson[MAX\*40],rson[MAX\*40];

ll lazy[MAX\*40],sum[MAX\*40];

char opt[10];

int build(int l,int r)

{

int re=tot++;lazy[re]=sum[re]=0;

if(l==r){

scanf("%lld",&sum[re]);return re;

}

int mid=(l+r)/2;

lson[re]=build(l,mid);rson[re]=build(mid+1,r);

sum[re]=sum[lson[re]]+sum[rson[re]];

return re;

}

void pushup(int rt,int l,int r)

{

int mid=(l+r)/2;

sum[rt]=sum[lson[rt]]+sum[rson[rt]]+(1LL\*mid-l+1)\*lazy[lson[rt]]+(1LL\*r-mid)\*lazy[rson[rt]];

return ;

}

int Insert(int rt,int il,int ir,int l,int r,ll d)

{

int ntr=tot++;lazy[ntr]=lazy[rt];

if(l>=il&&r<=ir){

lazy[ntr]=lazy[rt]+d;sum[ntr]=sum[rt];lson[ntr]=lson[rt];rson[ntr]=rson[rt];return ntr;

}

int mid=(l+r)/2;

if(il<=mid)

lson[ntr]=Insert(lson[rt],il,ir,l,mid,d);

else lson[ntr]=lson[rt];

if(ir>mid)

rson[ntr]=Insert(rson[rt],il,ir,mid+1,r,d);

else rson[ntr]=rson[rt];

pushup(ntr,l,r);

return ntr;

}

ll query(int rt,int ql,int qr,int l,int r,ll add)

{

if(l>=ql&&r<=qr){

return (add+lazy[rt])\*(r-l+1LL)+sum[rt];

}

int mid=(l+r)/2;

ll re=0;

if(ql<=mid)re+=query(lson[rt],ql,qr,l,mid,add+lazy[rt]);

if(qr>mid)re+=query(rson[rt],ql,qr,mid+1,r,add+lazy[rt]);

return re;

}

bool st;

int main()

{

while(~scanf("%d%d",&n,&m))

{

if(st)printf("\n");st=1;

tot=TM=0;

T[0]=build(1,n);

while(m--){

scanf("%s",opt);

if(opt[0]=='C'){

int l,r;ll d;scanf("%d%d%lld",&l,&r,&d);T[TM+1]=Insert(T[TM],l,r,1,n,d);++TM;

}

else if(opt[0]=='Q'){

int l,r;scanf("%d%d",&l,&r);printf("%lld\n",query(T[TM],l,r,1,n,0));

}

else if(opt[0]=='H'){

int l,r,t;scanf("%d%d%d",&l,&r,&t);printf("%lld\n",query(T[t],l,r,1,n,0));

}

else{

int t;scanf("%d",&t);TM=t;tot=T[TM+1];

}

}

}

return 0;

}

# 图论

## 2-SAT

**2-SAT**

/\*

2-SAT问题是这样的：有n个布尔变量xi,另有m个需要满足的条件，每个条件的形式都是

“xi为真/假或者xj为真/假”。比如“x1为真或者x3为假”、“x7为假或者x2为假”都是合法的条件。

这里或指的是两个条件至少有一个是正确的

\*/

const int maxn=1e5+5;

struct TwoSAT

{

int n;

vector<int>G[maxn\*2];

bool mark[maxn\*2];

int S[maxn\*2],c;//s数组存储当前被标记的点

bool dfs(int x)

{

if(mark[x^1])return false;//真假同时被标记，逻辑矛盾

if(mark[x])return true;//记忆化

mark[x]=1;

S[c++]=x;

for(int i=0;i<G[x].size();i++)

if(!dfs(G[x][i]))return false;//同一个强连通分量应该同一种颜色

return true;

}

void init(int n)

{

this->n=n;

for(int i=0;i<n\*2;i++)G[i].clear();

memset(mark,0,sizeof(mark));

}

//x=xval or y=yval

/\*

x为xval 或者 y为yval需要有一个满足

\*/

void add\_clause(int x,int xval,int y,int yval)

{

x=x\*2+xval;y=y\*2+yval;

G[x^1].push\_back(y);

G[y^1].push\_back(x);

}

bool solve()

{

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

{

if(!mark[i]&&!mark[i+1])//真假都没标记 故需要dfs

{

c=0;//清零

if(!dfs(i))//尝试标记为true

{

while(c>0)mark[S[--c]]=false;

if(!dfs(i+1))return false;//尝试标记为false

}

}

}

return true;

}

};

**2-SAT-例题1-BZOJ3495前缀优化建图（使用Tarjan判2-SAT）**

/\*

有 n 个点， m 条边和 K 个国家（国家里的点已知）。

每个国家只能选一个点作为首都，并且要保证最后所有边的两端至少有一个点是首都，问是否存在方案。

做法：

每个点是首都或不是首都，只有两个状态，所以是2-SAT问题。m 条边的限制很容易转化，就是每个国家只能选一个点为首都比较奇怪。

其实这是典型的前后缀优化建图，这里以前缀优化建图为例：

首先我们先增加 n 个点，令 i 的新增节点为 i+n 。然后对于一个国家，

假设有 w 个点，那么该国家中的第 i 个点（设为 A[i] ）的新增点 A[i]+n

表示该国家中1~ i 的点是否有被选的节点，即代表前缀 i 中是否有被选的节点。

那么我们可以得到以下关系式：

1. A[i−1]+n 选了， A[i]+n 必定选了； A[i]+n 没选， A[i−1]+n 必定没选（前缀之间的关系）。

2. A[i] 选了， A[i−1]+n 必定没选； A[i−1]+n 选了， A[i] 必定没选（一个国家只能有一个首都）。

后缀优化建图原理是一样的，只不过这道题并不需要后缀优化。

最后刷2-SAT判断是否存在解即可。

\*/

struct node{

int to,nxt;

}edg[MAX<<1];

int n,m,k,dcnt,ecnt,top,scc\_cnt;

int dfn[MAX],h[MAX],low[MAX],scc[MAX],stk[MAX];

bool instk[MAX];

void add\_edge(int u,int v){

edg[++ecnt]=node{v,h[u]};

h[u]=ecnt;

}

void Tarjan(int x){

dfn[x]=low[x]=++dcnt;stk[++top]=x;instk[x]=1;

for(int i=h[x];i;i=edg[i].nxt){

int to=edg[i].to;

if(!dfn[to])Tarjan(to),low[x]=min(low[x],low[to]);

else if(instk[to])low[x]=min(low[x],dfn[to]);

}

if(dfn[x]==low[x]){

++scc\_cnt;int y;

do y=stk[top--],instk[y]=0,scc[y]=scc\_cnt;while(y!=x);

}

}

bool Two\_SAT(){

for(int i=0;i<4\*n;i++)if(!dfn[i])Tarjan(i);

for(int i=0;i<4\*n;i+=2)

if(scc[i]==scc[i^1])return false;

}

int main()

{

read(n);read(m);read(k);

for(int x,y,i=1;i<=m;i++){

read(x);read(y);--x;--y;

add\_edge(x<<1,y<<1^1);add\_edge(y<<1,x<<1^1);

}

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

add\_edge(i<<1^1,i+n<<1^1),add\_edge(i+n<<1,i<<1);

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

int w,x,lst;read(w);

for(int j=0;j<w;j++,lst=x){

read(x);--x;

if(j){

add\_edge(x+n<<1,lst+n<<1);

add\_edge(lst+n<<1^1,x+n<<1^1);

add\_edge(x<<1^1,lst+n<<1);

add\_edge(lst+n<<1^1,x<<1);

}

}

}

if(Two\_SAT())printf("TAK\n");

else printf("NIE\n");

return 0;

}

## LCA

vector <int> G[N];

int seq[N], tin[N], fa[N], dep[N], top[N], son[N], sz[N], label;

void dfs1(int u, int father) {

fa[u] = father;

son[u] = 0; sz[u] = 1;

for(auto v : G[u]) {

if(v == father) continue;

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

dfs1(v, u);

sz[u] += sz[v];

if(sz[v] > sz[son[u]])

son[u] = v;

}

}

void dfs2(int u, int anc) {

top[u] = anc; tin[u] = ++ label;

seq[label] = u;

if(son[u]) dfs2(son[u], anc);

for(auto v : G[u]) {

if(v == fa[u] || v == son[u])

continue;

dfs2(v, v);

}

}

inline 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;

}

## 点分治

**点分治-例1-POJ1987-树上两点距离小于等于k的对数**

/\*

POJ 1741

题意：

多组测试数据，每次输入n、m，和一棵n个点的有边权的树，

问你满足x到y距离小于等于m的无序点对(x,y)的个数是多少。

做法：

如果我们已经知道了此时所有点到根的距离a[i]，a[x] + a[y] <= k的(x, y)对数就是结果，

这个可以通过排序之后O(n)的复杂度求出。然后根据分治的思想，分别对所有的儿子求一遍即可，

但是这会出现重复的——当前情况下两个点位于一颗子树中，那么应该将其减掉

（显然这两个点是满足题意的，为什么减掉呢？因为在对子树进行求解的时候，会重新计算）。

在进行分治时，为了避免树退化成一条链而导致时间复杂度变为O(N^2)，每次都找树的重心，这样，

所有的子树规模就会变的很小了。时间复杂度O(Nlog^2N)。

树的重心的算法可以线性求解。

\*/

struct node

{

int to,dis,nxt;

}edg[MAX<<2];

int n,ecnt,tot,rt,k;//tot为某一时刻的树上总点数 rt为重心

int h[MAX],siz[MAX],f[MAX];//f[i]记录当前状态i点最大子树的大小

int a[MAX],d[MAX];//d数组记录某点到根节点的距离

bool vi[MAX];

int an;

void add\_edge(int u,int v,int d)

{

edg[++ecnt]=node{v,d,h[u]};

h[u]=ecnt;

}

void dfs(int now,int fa)//找重心 计算siz

{

siz[now]=1;f[now]=0;

for(int i=h[now];i;i=edg[i].nxt)

{

int to=edg[i].to;

if(!vi[to]&&to!=fa){

dfs(to,now);siz[now]+=siz[to];f[now]=max(f[now],siz[to]);

}

}

f[now]=max(tot-siz[now],f[now]);

if(f[now]<f[rt])rt=now;

}

void pre(int now,int prre)//计算每一点到根节点距离

{

a[++a[0]]=d[now];

for(int i=h[now];i;i=edg[i].nxt)

{

int to=edg[i].to;

if(!vi[to]&&to!=prre){

d[to]=d[now]+edg[i].dis;pre(to,now);

}

}

}

//初始d[now]=ds以此为基准(保证每点深度不变)

int cal(int now,int ds)//计算以某一点为根 有多少a[i]+a[j]<=k的点对（可能在一条链上）

{

int re=0;

d[now]=ds;a[0]=0;pre(now,0);

sort(a+1,a+a[0]+1);

int l=1,r=a[0];

while(l<r)

{

if(a[l]+a[r]<=k)re+=r-l,++l;

else --r;

}

return re;

}

void solve(int x)//求解以x为根的个数

{

an+=cal(x,0);vi[x]=1;//初始为所有个数（包含在一个子树内的） 标记vi[x] 切断树

for(int i=h[x];i;i=edg[i].nxt)

{

int to=edg[i].to;

if(!vi[to]){

an-=cal(to,edg[i].dis);tot=siz[to];rt=0;//去掉在一个子树的

dfs(to,0);solve(rt);

}

}

}

int main()

{

#ifdef debug

freopen("2.in","r",stdin);

#endif // debug

read(n);

for(int u,v,w,i=1;i<n;i++)

{

read(u);read(v);read(w);

add\_edge(u,v,w);add\_edge(v,u,w);

}

read(k);f[0]=n+1;

tot=n;

dfs(1,0);

solve(rt);

printf("%d\n",an);

return 0;

}

**点分治-例2-POJ2114-树上两点距离恰等于k的个数**

struct node

{

int to,nxt,dis;

node(){};

node(int \_to,int \_nxt,int \_dis):to(\_to),nxt(\_nxt),dis(\_dis){};

}edg[MAX<<2];

int n,m,ecnt,k,rt,an,tot;

int h[MAX],siz[MAX],f[MAX],d[MAX],a[MAX];

bool vi[MAX];

void add\_edge(int u,int v,int d)

{

edg[++ecnt]=node(v,h[u],d);

h[u]=ecnt;

}

void dfs(int now,int fa)//找重心

{

siz[now]=1;f[now]=0;

for(int i=h[now];i;i=edg[i].nxt)

{

int to=edg[i].to;

if(!vi[to]&&to!=fa)

dfs(to,now),siz[now]+=siz[to],f[now]=max(f[now],siz[to]);

}

f[now]=max(f[now],tot-siz[now]);

if(f[now]<f[rt])rt=now;

}

void cal\_dis(int now,int fa)

{

a[++a[0]]=d[now];

for(int i=h[now];i;i=edg[i].nxt)

{

int to=edg[i].to;

if(!vi[to]&&to!=fa)

d[to]=d[now]+edg[i].dis,cal\_dis(to,now);

}

}

int cal\_num(int now,int pred)

{

int re=0;

d[now]=pred;a[0]=0;cal\_dis(now,0);

sort(a+1,a+a[0]+1);

int l=1,r=a[0];

while(l<r)

{

if(a[l]+a[r]==k){//只统计和恰为k的

if(a[l]==a[r]){

re+=(r-l+1)\*(r-l)/2;break;//深度相等 深度为此的个数设为x 则有C(x,2)对

}

int i=l,j=r;

while(a[i]==a[l])++i;

while(a[j]==a[r])--j;

re+=(i-l)\*(r-j);//深度不同 乘法原理 两种个数的乘积

l=i;r=j;

}

else if(a[l]+a[r]<k)++l;

else --r;

}

return re;

}

void solve(int x)

{

an+=cal\_num(x,0);vi[x]=1;

for(int i=h[x];i;i=edg[i].nxt)

{

int to=edg[i].to;

if(!vi[to]){

an-=cal\_num(to,edg[i].dis);tot=siz[to];rt=0;

dfs(to,0);solve(rt);

}

}

}

int main()

{

#ifdef debug

freopen("2.in","r",stdin);

#endif // debug

while(scanf("%d",&n)&&n)

{

ecnt=0;memset(vi,0,sizeof(vi));

memset(h,0,sizeof(h));

for(int v,d,i=1;i<=n;i++)

{

while(scanf("%d",&v)&&v){

scanf("%d",&d);

add\_edge(i,v,d);add\_edge(v,i,d);

}

}

while(scanf("%d",&k)&&k){

an=0;rt=0;

memset(vi,0,sizeof(vi));

f[0]=n+1;tot=n;

dfs(1,0);solve(rt);

puts(an>0?"AYE":"NAY");

}

puts(".");

}

return 0;

}

**点分治-例3-字典序最小的两点使路径上点的积为k**

struct node

{

int to,nxt;

}edg[MAX<<2];

int inv[M],a[MAX],h[MAX],f[MAX],siz[MAX],id[MAX],temd[MAX],d[MAX],mp[M];

int n,k,ecnt,rt,tot,an1,an2,dcnt;

bool vi[MAX];

void add\_edge(int u,int v)

{

edg[++ecnt]=node{v,h[u]};

h[u]=ecnt;

}

void getrt(int now,int fa)

{

f[now]=0;siz[now]=1;

for(int i=h[now];i;i=edg[i].nxt)

{

int to=edg[i].to;

if(!vi[to]&&to!=fa){

getrt(to,now);siz[now]+=siz[to];f[now]=max(f[now],siz[to]);

}

}

f[now]=max(f[now],tot-siz[now]);

if(f[now]<f[rt])rt=now;

}

void dfs(int now,int fa)

{

temd[++dcnt]=d[now];id[dcnt]=now;

for(int i=h[now];i;i=edg[i].nxt)

{

int to=edg[i].to;

if(!vi[to]&&to!=fa)

{

d[to]=(1LL\*d[now]\*a[to])%MOD;

dfs(to,now);

}

}

}

void query(int x,int ID)

{

x=1LL\*inv[x]\*k%MOD;

int y=mp[x];

if(!y)return;

if(y>ID)swap(y,ID);

if(y<an1||(y==an1&&ID<an2))

an1=y,an2=ID;

}

void solve(int now)

{

vi[now]=1;mp[a[now]]=now;

for(int i=h[now];i;i=edg[i].nxt)

{

int to=edg[i].to;

if(!vi[to])

{

dcnt=0;d[to]=a[to];//先只从该子树走 这样能与之前处理过的其他子树连接起来

dfs(to,now);

for(int j=1;j<=dcnt;j++)

query(temd[j],id[j]);

dcnt=0;d[to]=(1LL\*a[now]\*a[to])%MOD;

dfs(to,now);

for(int j=1;j<=dcnt;j++)//枚举该子树中每一个点

{

int now=mp[temd[j]];

if(!now||now>id[j])mp[temd[j]]=id[j];

}

}

}

mp[a[now]]=0;//清空map

for(int i=h[now];i;i=edg[i].nxt)

{

int to=edg[i].to;

if(!vi[to])

{

dcnt=0;d[to]=(1LL\*a[now]\*a[to])%MOD;

dfs(to,now);

for(int j=1;j<=dcnt;j++)mp[temd[j]]=0;

}

}

for(int i=h[now];i;i=edg[i].nxt)

{

int to=edg[i].to;

if(!vi[to])

{

rt=0;tot=siz[to];

getrt(to,0);

solve(rt);//开始去处理某个子树的重心

}

}

}

int main()

{

inv[1]=1;

for(int i=2;i<1000009;i++)

inv[i]=1LL\*(MOD-inv[MOD%i])\*(MOD/i)%MOD;

while(~scanf("%d%d",&n,&k))

{

memset(h,0,sizeof(h));ecnt=0;an1=an2=INF;

memset(vi,0,sizeof(vi));

for(int i=1;i<=n;i++)read(a[i]);

for(int u,v,i=1;i<n;i++)

{

read(u);read(v);add\_edge(u,v);add\_edge(v,u);

}

rt=0;f[0]=n+1;tot=n;

getrt(1,0);solve(rt);

if(an1==INF)printf("No solution\n");

else printf("%d %d\n",an1,an2);

}

return 0;

}

## 三元环

nsqrtn版

#include <bits/stdc++.h>

#define LL long long

using namespace std;

const int bas = 1e5+1;

const int maxn = 1e5+5;

vector<int> G[maxn];

int n, m;

//unordered\_set<LL> \_hash;

set<LL> \_hash;

int deg[maxn], vis[maxn];

int bel[maxn];

void init()

{

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

{

deg[i] = bel[i] = vis[i] = 0;

G[i].clear();

}

}

void work()

{

int x = sqrt(1.0\*m);

LL ans = 0;

for(int a = 1; a <= n; ++a)

{

vis[a] = 1;

//扫一遍与a相连的所有点，为下面提供O(1)判两点是否存在连边

for(int i = 0; i < G[a].size(); ++i)

bel[G[a][i]] = a;

for(int i = 0; i < G[a].size(); ++i)

{

int b = G[a][i];

if(vis[b]) continue;

if(deg[b] <= x)

{

//如果b度数<=sqrt(m)，则枚举b的所有边

for(int j = 0; j < G[b].size(); ++j)

{

int c = G[b][j];

if(bel[c] == a) ++ans;

}

}

else

{

//如果b度数>sqrt(m)，则枚举a的所有边

for(int j = 0; j < G[a].size(); ++j)

{

int c = G[a][j];

if(\_hash.find(1ll\*b\*bas+c) != \_hash.end())

++ans;

}

}

}

}

//统计后每个三元环的每条边都会被统计一次，所以应该/3

printf("%lld\n", ans/3);

}

int main()

{

int t, u, v;

while(~scanf("%d %d", &n, &m))

{

init(); \_hash.clear();

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

{

scanf("%d %d", &u, &v);

++deg[u], ++deg[v];

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

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

\_hash.insert(1ll\*u\*bas+v);

\_hash.insert(1ll\*v\*bas+u);

}

work();

}

return 0;

}

## 树剖

**树剖+LCA**

vector <int> G[N];

int seq[N], tin[N], fa[N], dep[N], top[N], son[N], sz[N], label;

void dfs1(int u, int father) {

fa[u] = father;

son[u] = 0; sz[u] = 1;

for(auto v : G[u]) {

if(v == father) continue;

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

dfs1(v, u);

sz[u] += sz[v];

if(sz[v] > sz[son[u]])

son[u] = v;

}

}

void dfs2(int u, int anc) {

top[u] = anc; tin[u] = ++ label;

seq[label] = u;

if(son[u]) dfs2(son[u], anc);

for(auto v : G[u]) {

if(v == fa[u] || v == son[u])

continue;

dfs2(v, v);

}

}

inline 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;

}

**树剖-链修改查询**

/\*

HDU 3966

题意：给一棵树，并给定各个点权的值，然后有3种操作：

I C1 C2 K: 把C1与C2的路径上的所有点权值加上K

D C1 C2 K：把C1与C2的路径上的所有点权值减去K

Q C：查询节点编号为C的权值

线段树维护即可

\*/

struct node

{

int to,nxt;

}edg[MAX<<1];

int n,m,p,ecnt,dcnt;

int h[MAX],a[MAX],top[MAX],dep[MAX],son[MAX],fa[MAX],sz[MAX],dfn[MAX],who[MAX];

void add\_edge(int u,int v)

{

edg[++ecnt]=node{v,h[u]};

h[u]=ecnt;

}

void dfs1(int u,int pre)

{

fa[u]=pre;son[u]=0;sz[u]=1;

for(int i=h[u];i;i=edg[i].nxt){

int to=edg[i].to;if(to==pre)continue;

dep[to]=dep[u]+1;

dfs1(to,u);

sz[u]+=sz[to];

if(sz[to]>sz[son[u]])

son[u]=to;

}

}

void dfs2(int u,int anc){

top[u]=anc;dfn[u]=++dcnt;who[dcnt]=u;

if(son[u])dfs2(son[u],anc);

for(int i=h[u];i;i=edg[i].nxt){

int to=edg[i].to;

if(to==fa[u]||to==son[u])continue;

dfs2(to,to);

}

}

int sum[MAX\*4],lazy[MAX<<2];

void pushup(int x){

sum[x]=sum[x<<1]+sum[x<<1|1];

}

void build(int x,int l,int r){

lazy[x]=0;

if (l==r){

sum[x]=a[who[l]];

return;

}int mid=(l+r)/2;

build(x<<1,l,mid);

build(x<<1|1,mid+1,r);

pushup(x);

}

void pushdown(int k,int l,int r)

{

int mid=(l+r)/2;

sum[k<<1]+=lazy[k]\*(mid-l+1);sum[k<<1|1]+=lazy[k]\*(r-mid);

lazy[k<<1]+=lazy[k];lazy[k<<1|1]+=lazy[k];

lazy[k]=0;

}

void update(int x,int l,int r,int L,int R,int val){//negative

if (l<=L&&R<=r){

sum[x]+=val\*(R-L+1);

lazy[x]+=val;

return;

}

if(lazy[x])pushdown(x,L,R);

int mid=(L+R)/2;

if (mid>=l) update(x<<1,l,r,L,mid,val);

if (r>mid) update(x<<1|1,l,r,mid+1,R,val);

pushup(x);

}

//int query(int x,int l,int r,int L,int R){

// if (!sum[x]) return 0;

// if (l<=L&&R<=r) return sum[x];

// int mid=(L+R)/2,ret=0;

// if (mid>=l) ret+=query(x<<1,l,r,L,mid);

// if (r>mid) ret+=query(x<<1|1,l,r,mid+1,R);

// pushup(x);

// return ret;

//}

int query(int x,int lo,int l,int r)

{

if(l==r)return sum[x];

if(lazy[x])pushdown(x,l,r);

int mid=(l+r)/2,re;

if(lo<=mid)re=query(x<<1,lo,l,mid);

else re=query(x<<1|1,lo,mid+1,r);

pushup(x);

return re;

}

char opt[15];

int main()

{

while(~scanf("%d%d%d",&n,&m,&p))

{

ecnt=dcnt=0;

memset(h,0,sizeof(h));

for(int i=1;i<=n;i++)read(a[i]);

for(int u,v,i=1;i<n;i++){

read(u);read(v);add\_edge(u,v);add\_edge(v,u);

}

dfs1(1,0);dfs2(1,1);

build(1,1,n);

while(p--){

scanf("%s",opt);

if(opt[0]=='Q'){

int w;read(w);w=dfn[w];

printf("%d\n",query(1,w,1,n));

}

else{

int u,v,val;

read(u);read(v);read(val);

if(opt[0]=='D')val=-val;

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

{

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

update(1,dfn[top[u]],dfn[u],1,n,val);

u=fa[top[u]];

}

if(dep[u]>dep[v])swap(u,v);

update(1,dfn[u],dfn[v],1,n,val);

}

}

}

return 0;

}

**树剖-链最大值、修改、置负**

struct node

{

int to,nxt,val;

}edg[MAX<<1];

int t,ecnt,dcnt,n;

int h[MAX],top[MAX],dep[MAX],son[MAX],fa[MAX],sz[MAX],dfn[MAX],who[MAX];

int x[MAX],y[MAX],val[MAX],dis[MAX];

char opt[105];

void add\_edge(int u,int v,int val)

{

edg[++ecnt]=node{v,h[u],val};

h[u]=ecnt;

}

void dfs1(int u,int pre)

{

fa[u]=pre;son[u]=0;sz[u]=1;

for(int i=h[u];i;i=edg[i].nxt)

{

int to=edg[i].to;if(to==pre)continue;

dep[to]=dep[u]+1;

dis[to]=edg[i].val;

dfs1(to,u);

sz[u]+=sz[to];

if(sz[to]>sz[son[u]])

son[u]=to;

}

}

void dfs2(int u,int anc)

{

top[u]=anc;dfn[u]=++dcnt;who[dcnt]=u;

if(son[u])dfs2(son[u],anc);

for(int i=h[u];i;i=edg[i].nxt)

{

int to=edg[i].to;

if(to==fa[u]||to==son[u])continue;

dfs2(to,to);

}

}

inline 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;

}

int Max[MAX<<2],Min[MAX<<2],lazy[MAX<<2];

void push(int k)

{

if(!lazy[k])return;

lazy[k<<1]^=1;lazy[k<<1|1]^=1;lazy[k]=0;

int da=Max[k<<1],xiao=Min[k<<1];

Max[k<<1]=-xiao;Min[k<<1]=-da;

da=Max[k<<1|1],xiao=Min[k<<1|1];

Max[k<<1|1]=-xiao,Min[k<<1|1]=-da;

}

void build(int k,int l,int r)

{

if(l==r){Max[k]=Min[k]=dis[who[l]];lazy[k]=0;return;}

int mid=(l+r)/2;

lazy[k]=0;

build(k<<1,l,mid);build(k<<1|1,mid+1,r);

Min[k]=min(Min[k<<1],Min[k<<1|1]);

Max[k]=max(Max[k<<1],Max[k<<1|1]);

}

int query(int k,int l,int r,int ql,int qr)

{

if(l>=ql&&r<=qr)return Max[k];

int mid=(l+r)/2;

int re=-INF;

push(k);

if(ql<=mid)re=max(re,query(k<<1,l,mid,ql,qr));

if(qr>mid)re=max(re,query(k<<1|1,mid+1,r,ql,qr));

return re;

}

int Query(int u,int v)

{

int re=-INF;

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

{

re=max(re,query(1,2,n,dfn[top[u]],dfn[u]));

u=fa[top[u]];

}

if(dfn[v]+1<=dfn[u])

re=max(re,query(1,2,n,dfn[v]+1,dfn[u]));//同一条链上就是连续的!

return re;

}

void change(int k,int lo,int l,int r)

{

push(k);

if(l==r){

Max[k]=dis[who[l]];Min[k]=dis[who[l]];

return;

}

int mid=(l+r)/2;

if(lo<=mid)change(k<<1,lo,l,mid);

else change(k<<1|1,lo,mid+1,r);

Min[k]=min(Min[k<<1],Min[k<<1|1]);

Max[k]=max(Max[k<<1],Max[k<<1|1]);

}

void update(int k,int l,int r,int ul,int ur)

{

push(k);

if(l>=ul&&r<=ur){

lazy[k]^=1;

int da=Max[k],xiao=Min[k];

Max[k]=-xiao;Min[k]=-da;

return ;

}

int mid=(l+r)/2;

if(ul<=mid)update(k<<1,l,mid,ul,ur);

if(ur>mid)update(k<<1|1,mid+1,r,ul,ur);

Max[k]=max(Max[k<<1],Max[k<<1|1]);

Min[k]=min(Min[k<<1],Min[k<<1|1]);

}

void Update(int u,int v)

{

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

{

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

update(1,2,n,dfn[top[u]],dfn[u]);

u=fa[top[u]];

}

if(dfn[u]<dfn[v])swap(u,v);

if(dfn[v]+1<=dfn[u])

update(1,2,n,dfn[v]+1,dfn[u]);

}

int main()

{

#ifdef debug

freopen("in.txt","r",stdin);

freopen("out.txt","w",stdout);

#endif // debug

read(t);

while(t--){

memset(h,0,sizeof(h));

ecnt=dcnt=0;

read(n);

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

read(x[i]);read(y[i]);read(val[i]);

add\_edge(x[i],y[i],val[i]);

add\_edge(y[i],x[i],val[i]);

}

dfs1(1,0);dfs2(1,1);

build(1,2,n);

while(scanf("%s",opt)&&opt[0]!='D')

{

int u,v,c;read(u);read(v);

if(opt[0]=='Q'){

c=lca(u,v);

#ifdef debug

printf("c=%d\n",c);

#endif // debug

printf("%d\n",max(Query(u,c),Query(v,c)));

}

else if(opt[0]=='C'){

u=(dfn[x[u]]>dfn[y[u]]?x[u]:y[u]);

dis[u]=v;

change(1,dfn[u],2,n);

}

else

Update(u,v);

}

}

return 0;

}

## 虚树

**虚树BZOJ2286**

/\*

bzoj 2286

题意：

给定一棵树，每条边都有一定权值。若干次询问，每次给出m个点，要求去掉权值和最小的一些边，使得1与这些点不连通

\*/

struct node

{

node(){};

node(int \_to,int \_dis,int \_nxt):to(\_to),dis(\_dis),nxt(\_nxt){};

node(int \_to,int \_nxt):to(\_to),nxt(\_nxt){};

int to,dis,nxt;

}edg[MAX<<1];

int n,ecnt,c,m,tot,top;

int x,y,z;

int h[MAX],hd[MAX];//val:根节点到某点路径上最短的边长

int dep[MAX],fa[MAX][20],dfn[MAX],que[MAX],stk[MAX];

ll f[MAX],val[MAX];

void add\_edge(int u,int v,int cost)//原树的图

{

edg[++ecnt]=node(v,cost,h[u]);

h[u]=ecnt;

}

void dfs(int now,int pre)

{

fa[now][0]=pre;dfn[now]=++ecnt;

for(int i=1;i<=18;i++)fa[now][i]=fa[fa[now][i-1]][i-1];

for(int i=h[now];i;i=edg[i].nxt)

{

int to=edg[i].to;

if(to==pre)continue;

val[to]=min(val[now],1LL\*edg[i].dis);dep[to]=dep[now]+1;

dfs(to,now);

}

}

bool cmp(int a,int b){return dfn[a]<dfn[b];}

inline int lca(int x,int y)

{

if(dep[x]<dep[y])swap(x,y);

int cha=dep[x]-dep[y];

for(int i=18;i>=0;i--)

if(cha&(1<<i))x=fa[x][i];

if(x==y)return x;

for(int i=18;i>=0;i--)

if(fa[x][i]!=fa[y][i])x=fa[x][i],y=fa[y][i];

return x==y?x:fa[x][0];

}

inline void add(int u,int v){if(u==v)return ;edg[++ecnt]=node(v,hd[u]);hd[u]=ecnt;}

inline void dp(int x)

{

ll tem=0;f[x]=val[x];

for(int i=hd[x];i;i=edg[i].nxt){

dp(edg[i].to);

tem+=f[edg[i].to];

}

hd[x]=0;//退出时顺便清空

if(!tem)f[x]=val[x];//叶子节点 val[x]就是将该点去掉的最小花费

else if(tem<f[x])f[x]=tem;

}

void solve()//断绝根节点到1的路径？？

{

read(m);

for(int i=1;i<=m;i++)read(que[i]);

sort(que+1,que+1+m,cmp);//按dfs序排序

tot=0;que[++tot]=que[1];

for(int i=2;i<=m;i++)if(lca(que[i],que[tot])!=que[tot])que[++tot]=que[i];

//在下面的肯定不用计算 只要切断上部的即可

top=0;stk[++top]=1;

ecnt=0;//清空图

int grand;//LCA

for(int i=1;i<=tot;i++){

grand=lca(stk[top],que[i]);

while(1){

if(dep[stk[top-1]]<=dep[grand]){

add(grand,stk[top]);top--;

if(stk[top]!=grand)stk[++top]=grand;

break;

}

add(stk[top-1],stk[top]);--top;

}

if(stk[top]!=que[i])stk[++top]=que[i];//在同一子树

}

--top;

while(top)add(stk[top],stk[top+1]),--top;//剩余的记得连上

dp(1);

printf("%lld\n",f[1]);

}

int main()

{

read(n);

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

read(x);read(y);read(z);

add\_edge(x,y,z);add\_edge(y,x,z);

}

val[1]=INFF;ecnt=0;dep[1]=0;

dfs(1,0);

read(c);

while(c--)solve();

return 0;

}

**虚树-例2-BZOJ3572**

struct node{

int to,nxt;

}edg[MAX<<1],vtg[MAX<<1];

int n,q,ecnt,vcnt,dcnt,ccnt,top,tot,LCA;

int h[MAX],f[MAX],fa[MAX][20],siz[MAX],dfn[MAX],dep[MAX];

int bel[MAX],stk[MAX],a[MAX],b[MAX],g[MAX],c[MAX],ans[MAX];

void add\_edge(int u,int v){//原树加边

if(u==0||v==0||u==v)return;

edg[++ecnt]=node{v,h[u]};

h[u]=ecnt;

}

void addedge(int u,int v){//虚树加边

if(u==0||v==0||u==v)return;

vtg[++vcnt]=node{v,f[u]};

f[u]=vcnt;

}

void dfs(int now,int pre){

siz[now]=1;dfn[now]=++dcnt;

for(int i=1;i<=18;i++)fa[now][i]=fa[fa[now][i-1]][i-1];

for(int i=h[now];i;i=edg[i].nxt){

int to=edg[i].to;

if(to==pre)continue;

fa[to][0]=now;dep[to]=dep[now]+1;

dfs(to,now);siz[now]+=siz[to];

}

}

inline int lca(int u,int v){

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

int cha=dep[u]-dep[v];

for(int i=18;i>=0;i--)if(cha&(1<<i))u=fa[u][i];

if(u==v)return u;

for(int i=18;i>=0;i--)

if(fa[u][i]!=fa[v][i])u=fa[u][i],v=fa[v][i];

return fa[u][0];

}

inline int getdis(int x,int y){return dep[x]+dep[y]-2\*dep[lca(x,y)]; }

//使用子树更新该点

void dfs1(int now,int pre){

int d1,d2;g[now]=siz[now];

c[++ccnt]=now;

for(int i=f[now];i;i=vtg[i].nxt){

int to=vtg[i].to;

if(to==pre)continue;

dfs1(to,now);

if(bel[to]==0)continue;

if(bel[now]==0){bel[now]=bel[to];continue;}

d1=getdis(bel[to],now);d2=getdis(bel[now],now);

if(d1<d2||(d1==d2&&bel[to]<bel[now]))bel[now]=bel[to];

}

}

//使用父节点更新该点

void dfs2(int now,int pre){

int d1,d2;

for(int i=f[now];i;i=vtg[i].nxt){

int to=vtg[i].to;

if(to==pre)continue;

if(bel[to]==0){bel[to]=bel[now];}

else{

d1=getdis(bel[to],to);d2=getdis(bel[now],to);

if(d1>d2||(d1==d2&&bel[to]>bel[now]))

bel[to]=bel[now];

}

dfs2(to,now);

}

}

//处理每一条边

void solve(int pre,int now){

int son=now,mid=now,d1,d2,nex;

for(int i=18;i>=0;i--)

if(dep[fa[son][i]]>dep[pre])son=fa[son][i];

g[pre]-=siz[son];

if(bel[pre]==bel[now]){

ans[bel[pre]]+=siz[son]-siz[now];

return;

}

for(int i=18;i>=0;i--){

nex=fa[mid][i];

if(dep[nex]<=dep[pre])continue;

d1=getdis(nex,bel[pre]);d2=getdis(nex,bel[now]);

if(d1>d2||(d1==d2&&bel[now]<bel[pre]))

mid=nex;

}

ans[bel[pre]]+=siz[son]-siz[mid];

ans[bel[now]]+=siz[mid]-siz[now];

}

bool cmp(int x,int y){

return dfn[x]<dfn[y];

}

//建虚树 前几行读入关键点

void build(){

read(tot);

for(int i=1;i<=tot;i++)read(a[i]),b[i]=a[i];

for(int i=1;i<=tot;i++)bel[a[i]]=a[i];

sort(a+1,a+1+tot,cmp);

top=vcnt=0;

if(bel[1]!=1)stk[++top]=1;

for(int i=1;i<=tot;i++){

if(top==0){stk[++top]=a[i];continue;}

LCA=lca(stk[top],a[i]);

while(1){

if(dep[stk[top-1]]<=dep[LCA]){

addedge(LCA,stk[top]);--top;

if(stk[top]!=LCA)stk[++top]=LCA;

break;

}

addedge(stk[top-1],stk[top]);--top;

}

if(stk[top]!=a[i])stk[++top]=a[i];

}

while(top>1)addedge(stk[top-1],stk[top]),--top;

top=ccnt=0;

dfs1(1,0);dfs2(1,0);

for(int i=1;i<=ccnt;i++)

for(int j=f[c[i]];j;j=vtg[j].nxt)

solve(c[i],vtg[j].to);

for(int i=1;i<=ccnt;i++)

ans[bel[c[i]]]+=g[c[i]];

for(int i=1;i<=tot;i++)printf("%d ",ans[b[i]]);

printf("\n");

for(int i=1;i<=ccnt;i++)

ans[c[i]]=f[c[i]]=g[c[i]]=bel[c[i]]=0;

}

int main()

{

read(n);

for(int u,v,i=1;i<n;++i){

read(u);read(v);add\_edge(u,v);add\_edge(v,u);

}

dfs(1,0);

read(q);

while(q--)build();

return 0;

}

## 树分块

/\*

树分块

BZOJ 1086

给出一棵树，求一种分块方案，使得每个块的大小size∈[B,3B]。

每个块还要选一个省会，省会可以在块外，但是省会到块内任何一个点路径上的所有除了省会的点都必须属于这个块。n≤1000。

注意：分完块以后每个块并不一定连通

\*/

int n,b;

int tot;

struct node

{

int to,nxt;

}edg[MAX];

int h[MAX],id[MAX],cap[MAX],cnt;

void add(int u,int v)

{

edg[++tot]=node{v,h[u]};

h[u]=tot;

}

int stk[MAX],top;

void dfs(int now,int fa,int bot)

{

for(int i=h[now];i;i=edg[i].nxt)

{

int to=edg[i].to;

if(to!=fa)

{

dfs(to,now,top);

if(top-bot>=b)

{

cap[++cnt]=now;

while(top!=bot)id[stk[top--]]=cnt;

}

}

}

stk[++top]=now;

}

int main()

{

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

if(n<b)return 0\*printf("0\n");

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

{

int u,v;scanf("%d%d",&u,&v);add(u,v);add(v,u);

}

dfs(1,0,0);

while(top)

id[stk[top--]]=cnt;

printf("%d\n",cnt);

for(int i=1;i<=n;i++)printf("%d ",id[i]);printf("\n");

for(int i=1;i<=cnt;i++)printf("%d ",cap[i]);

}

**树分块莫队**

int n,m;

int bin[20],fa[MAX][20],bel[MAX],a[MAX],cnt[MAX],edg\_cnt,root,dep[MAX];

int dfn[MAX],dfs\_id;

int sum;

int siz,block\_cnt;

int stk[MAX],top;

bool vi[MAX];

struct node

{

int to,nxt;

}edg[MAX<<1];

int head[MAX];

void add(int u,int v)

{

edg[++edg\_cnt]=node{v,head[u]};

head[u]=edg\_cnt;

}

void dfs(int now,int bot)

{

dfn[now]=++dfs\_id;

// printf("now=%d\n",now);

// system("pause");

for(int i=1;i<=16&&dep[now]>=bin[i];i++)fa[now][i]=fa[fa[now][i-1]][i-1];

for(int i=head[now];i;i=edg[i].nxt)

{

int to=edg[i].to;

if(to!=fa[now][0])

{

dep[to]=dep[now]+1;

fa[to][0]=now;

dfs(to,top);

if(top-bot>=siz)

{

++block\_cnt;

while(top!=bot)

{

bel[stk[top--]]=block\_cnt;

}

}

}

}

stk[++top]=now;

}

int lca(int u,int v)

{

if(dep[u]>dep[v])swap(u,v);

int dif=dep[v]-dep[u];

for(int i=0;bin[i]<=dif;i++)

if(dif&bin[i])v=fa[v][i];

for(int i=16;i>=0;i--)

if(fa[u][i]!=fa[v][i])u=fa[u][i],v=fa[v][i];

if(u!=v)return fa[v][0];

return v;

}

int reverse(int u)

{

if(!vi[u])vi[u]=1,++cnt[a[u]],sum+=(cnt[a[u]]==1);

else vi[u]=0,--cnt[a[u]],sum-=(!cnt[a[u]]);

}

void solve(int u,int v)

{

while(u!=v)

{

// printf("%d %d %d %d\n",u,v,dep[u],dep[v]);

if(dep[u]>dep[v])reverse(u),u=fa[u][0];

else reverse(v),v=fa[v][0];

}

}

struct nod

{

int u,v,a,b,id;

bool operator<(const nod& z)const

{

if(bel[u]!=bel[z.u])return bel[u]<bel[z.u];

else return dfn[v]<dfn[z.v];

}

}qs[MAX];

int an[MAX];

**/\***

**BZOJ3757 求路径上不同颜色点个数 （a被视为b 除了a=b=0）**

**\*/**

int main()

{

// freopen("6.in","r",stdin);

// freopen("out.txt","w",stdout);

bin[0]=1;

for(int i=1;i<20;i++)bin[i]=bin[i-1]<<1;

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

siz=sqrt(n);

for(int i=1;i<=n;i++)scanf("%d",&a[i]);

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

{

int u,v;scanf("%d%d",&u,&v);if(u&&v){add(u,v);add(v,u);}else {if(!u)root=v;if(!v)root=u;}

}

// printf("root=%d\n",root);

// system("pause");

dfs(root,0);

// printf("??\n");

++block\_cnt;

while(top)

bel[stk[top--]]=block\_cnt;

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

{

scanf("%d%d%d%d",&qs[i].u,&qs[i].v,&qs[i].a,&qs[i].b),qs[i].id=i;

if(dfn[qs[i].u]>dfn[qs[i].v])swap(qs[i].u,qs[i].v);

}

sort(qs+1,qs+1+m);

int t=lca(qs[1].u,qs[1].v);

// printf("??\n");

solve(qs[1].u,qs[1].v);

reverse(t);

an[qs[1].id]=sum-(cnt[qs[1].a]&&cnt[qs[1].b]&&qs[1].a!=qs[1].b);

reverse(t);

for(int i=2;i<=m;i++)

{

// printf("i=%d\n",i);

solve(qs[i-1].u,qs[i].u);

solve(qs[i-1].v,qs[i].v);

t=lca(qs[i].u,qs[i].v);

reverse(t);

an[qs[i].id]=sum-(cnt[qs[i].a]&&cnt[qs[i].b]&&qs[i].a!=qs[i].b);

reverse(t);

}

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

printf("%d\n",an[i]);

}

# 字符串

## 后缀自动机

## hiho版

/\*

模版部分

\*/

const int MAXL=2000005;

const int MAXCH=26;

char s[MAXL];

char CH='a';

int tot=0;//tot为总结点数

int maxlen[MAXL],minlen[MAXL],trans[MAXL][MAXCH],slink[MAXL];

/\*

一些非必要的功能属性

若去掉 还需在函数中修改

\*/

int ind[MAXL];//树中子节点个数

int edpts[MAXL];//结点的endpos个数

bool pre[MAXL];//是否为前缀所在结点

/\*

声明新结点的函数 未知初值的 值可用-1代替 未知初值的引用 可用NULL代替

\*/

int new\_state(int \_maxlen,int \_minlen,int \*\_trans,int \_slink)

{

maxlen[tot]=\_maxlen;

minlen[tot]=\_minlen;

for(int i=0;i<MAXCH;i++)

{

if(\_trans==NULL)

trans[tot][i]=-1;

else trans[tot][i]=\_trans[i];

}

slink[tot]=\_slink;

return tot++;

}

/\*

在u结点后加入字符ch

\*/

int add\_char(char ch,int u)

{

int c=ch-CH;

/\*

建广义自动机即插完一个串后 回到root（0）

之后在这里下一行不做修改

或加入 (2018.8.3UPD:完全不用加这句话！)

if(trans[u][c]!=-1&&maxlen[trans[u][c]]==maxlen[u]+1)return trans[u][c];

若没有if成功则继续下述操作

（以下是之前的错误认识：个人感觉加上判断会好一些 不然会出现maxlen<minlen的奇怪的点（e.g. 加入两次"a"这个串））

\*/

int z=new\_state(maxlen[u]+1,-1,NULL,-1);

pre[z]=true;//该结点必为包含原串前缀的结点

int v=u;

while(v!=-1&&trans[v][c]==-1)

{

trans[v][c]=z;

v=slink[v];

}

if(v==-1)//最简单的情况，suffix-path(u->s)上都没有对应字符ch的转移

{

minlen[z]=1;

slink[z]=0;

++ind[0];

return z;

}

int x=trans[v][c];

if(maxlen[v]+1==maxlen[x])//较简单的情况，不用拆分x

{

minlen[z]=maxlen[x]+1;

slink[z]=x;

++ind[x];

return z;

}

int y=new\_state(maxlen[v]+1,-1,trans[x],slink[x]);//最复杂的情况，拆分x

slink[y]=slink[x];

ind[y]+=2;

minlen[x]=maxlen[y]+1;

slink[x]=y;

minlen[z]=maxlen[y]+1;

slink[z]=y;

int w=v;

while(w!=-1&&trans[w][c]==x)//该部分字符串长度越来越短 一定都是要转移到y的 因为y保留的是x原本较短的部分

{

trans[w][c]=y;

w=slink[w];

}

minlen[y]=maxlen[slink[y]]+1;

return z;

}

/\*

拓扑排序

获得每个结点的endpos个数

该种topsort是从叶子结点往上的排序 下文有从根节点往下的例子

\*/

void getEndPtCount() {

queue<int> q;

for(int i=1;i<tot;i++)

if(!ind[i])

q.push(i);

while( !q.empty() ) {

int u = q.front();

q.pop();

if(pre[u])edpts[u]++;

edpts[ slink[u]] += edpts[u];

if( !--ind[slink[u]] ) q.push(slink[u]);

}

}

/\*

另一种toposort

\*/

void toposort()

{

for(int i=0;i<=n;i++)cnt[i]=0;

for(int i=1;i<tot;i++)

++cnt[maxlen[i]];

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

cnt[i]+=cnt[i-1];

for(int i=tot-1;i>0;--i)

lo[cnt[maxlen[i]]--]=i;

}

/\*

使用示例：

求解字符串中所有不同子串个数 hihocoder 1445

& 求解字符串中每个长度子串出现最多的次数 hihocoder 1449

\*/

int anlen[MAXL];

int n;

ll an;

int main()

{

/\*

初始化部分

\*/

int pres=new\_state(0,0,NULL,-1);

// memset(trans,-1,sizeof(trans));

// memset(slink,-1,sizeof(slink));

/\*

通用过程代码部分

\*/

scanf("%s",s);

// tot=1;//tot=0的表示空串的结点 故需先设tot=1 使得新结点从1开始编号

n=strlen(s);

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

pres=add\_char(s[i],pres);//逐个加入结点

/\*

求解字符串中所有不同子串个数

\*/

for(int i=1;i<tot;i++)//所有结点下标范围为 [0,tot) 0为空串结点 故从1开始计算

an+=1LL\*maxlen[i]-minlen[i]+1;

printf("%lld\n",an);

/\*

求解字符串中每个长度子串出现最多的次数

\*/

getEndPtCount();

for(int i=1;i<tot;i++)//所有结点下标范围为 [0,tot) 0为空串结点 故从1开始计算

anlen[maxlen[i]]=max(anlen[maxlen[i]],edpts[i]);

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

anlen[i]=max(anlen[i],anlen[i+1]);

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

printf("%d\n",anlen[i]);

}

/\*

根节点向下topsort的例子

hihocoder 1457

\*/

const int MAXL=2000005;

const int MAXCH=11;

string s,tem;

char CH='0';

int tot=0;//tot为总结点数

int maxlen[MAXL],minlen[MAXL],trans[MAXL][MAXCH],slink[MAXL];

/\*

一些非必要的功能属性

若去掉 还需在函数中修改

\*/

int ind[MAXL];//树中子节点个数

int edpts[MAXL];//结点的endpos个数

bool pre[MAXL];//是否为前缀所在结点

/\*

声明新结点的函数 未知初值的 值可用-1代替 未知初值的引用 可用NULL代替

\*/

int new\_state(int \_maxlen,int \_minlen,int \*\_trans,int \_slink)

{

maxlen[tot]=\_maxlen;

minlen[tot]=\_minlen;

for(int i=0;i<MAXCH;i++)

{

if(\_trans==NULL)

trans[tot][i]=-1;

else trans[tot][i]=\_trans[i];

}

slink[tot]=\_slink;

return tot++;

}

/\*

在u结点后加入字符ch

\*/

int add\_char(char ch,int u)

{

int c=ch-CH;

int z=new\_state(maxlen[u]+1,-1,NULL,-1);

pre[z]=true;//该结点必为包含原串前缀的结点

int v=u;

while(v!=-1&&trans[v][c]==-1)

{

trans[v][c]=z;

v=slink[v];

}

if(v==-1)//最简单的情况，suffix-path(u->s)上都没有对应字符ch的转移

{

minlen[z]=1;

slink[z]=0;

++ind[0];

return z;

}

int x=trans[v][c];

if(maxlen[v]+1==maxlen[x])//较简单的情况，不用拆分x

{

minlen[z]=maxlen[x]+1;

slink[z]=x;

++ind[x];

return z;

}

int y=new\_state(maxlen[v]+1,-1,trans[x],slink[x]);//最复杂的情况，拆分x

slink[y]=slink[x];

ind[y]+=2;

minlen[x]=maxlen[y]+1;

slink[x]=y;

minlen[z]=maxlen[y]+1;

slink[z]=y;

int w=v;

while(w!=-1&&trans[w][c]==x)//该部分字符串长度越来越短 一定都是要转移到y的 因为y保留的是x原本较短的部分

{

trans[w][c]=y;

w=slink[w];

}

minlen[y]=maxlen[slink[y]]+1;

return z;

}

/\*

拓扑排序

获得每个结点的endpos个数

\*/

void getEndPtCount() {

queue<int> q;

for(int i=1;i<tot;i++)

if(!ind[i])

q.push(i);

while( !q.empty() ) {

int u = q.front();

q.pop();

if(pre[u])edpts[u]++;

edpts[ slink[u]] += edpts[u];

if( !--ind[slink[u]] ) q.push(slink[u]);

}

}

/\*

使用示例：

求解字符串中所有不同子串个数

& 求解字符串中每个长度子串出现最多的次数

\*/

int cnt[MAXL];

ll val\_num[MAXL];//到每个结点valid的个数

ll sum[MAXL];//每个结点的和

void init\_dag()

{

for(int i=0;i<=tot;i++)

for(int j=0;j<11;j++)

if(trans[i][j]>0)

++cnt[trans[i][j]];

}

void topsort()

{

queue<int>q;

for(int i=0;i<tot;i++)

{

if(cnt[i]==0)

{

q.push(i);

val\_num[i]=1;

sum[i]=0;

}

}

while(!q.empty())

{

int now=q.front();

q.pop();

for(int k=0;k<11;k++)

{

if(trans[now][k]>0)

{

int j=trans[now][k];

if(k<10)

{

val\_num[j]=(val\_num[j]+val\_num[now])%MOD;

sum[j]=(sum[j]+sum[now]\*10LL%MOD+k\*val\_num[now]%MOD)%MOD;

}

--cnt[j];

if(!cnt[j])q.push(j);

}

}

}

}

int n;

ll an;

int main()

{

/\*

初始化部分

\*/

int pres=new\_state(0,0,NULL,-1);

/\*

通用过程代码部分

\*/

cin>>n;

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

{

cin>>tem;

if(i)s+=":";

s+=tem;

}

n=s.length();

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

pres=add\_char(s[i],pres);//逐个加入结点

init\_dag();

topsort();

/\*

求解字符串中所有不同子串个数

\*/

for(int i=1;i<tot;i++)

an=(an+sum[i])%MOD;

printf("%lld\n",an);

}

## SAM(hiho-struct版-无说明)

const int MAXL=2e6+5;

const int MAXCH=26;

const char CH='a';

struct SAM

{

int maxlen[MAXL],minlen[MAXL],trans[MAXL][MAXCH],slink[MAXL];

int ind[MAXL];//树中子节点个数

int edpts[MAXL];//结点的endpos个数

bool pre[MAXL];//是否为前缀所在结点

int tot;

int new\_state(int \_maxlen,int \_minlen,int \*\_trans,int \_slink)

{

maxlen[tot]=\_maxlen;

minlen[tot]=\_minlen;

for(int i=0;i<MAXCH;i++)

{

if(\_trans==NULL)

trans[tot][i]=-1;

else trans[tot][i]=\_trans[i];

}

slink[tot]=\_slink;

return tot++;

}

int add\_char(char ch,int u)

{

int c=ch-CH;

/\*

建广义自动机即插完一个串后 回到root（0）

在这里下一行不做修改

或加入(2018.8.3UPD:完全不用加这句话！)

if(trans[u][c]!=-1&&maxlen[trans[u][c]]==maxlen[u]+1)return trans[u][c];

若没有if成功则继续下述操作

\*/

int z=new\_state(maxlen[u]+1,-1,NULL,-1);

pre[z]=true;//该结点必为包含原串前缀的结点

int v=u;

while(v!=-1&&trans[v][c]==-1)

{

trans[v][c]=z;

v=slink[v];

}

if(v==-1)//最简单的情况，suffix-path(u->s)上都没有对应字符ch的转移

{

minlen[z]=1;

slink[z]=0;

++ind[0];

return z;

}

int x=trans[v][c];

if(maxlen[v]+1==maxlen[x])//较简单的情况，不用拆分x

{

minlen[z]=maxlen[x]+1;

slink[z]=x;

++ind[x];

return z;

}

int y=new\_state(maxlen[v]+1,-1,trans[x],slink[x]);//最复杂的情况，拆分x

ind[y]+=2;

minlen[x]=maxlen[y]+1;

slink[x]=y;

minlen[z]=maxlen[y]+1;

slink[z]=y;

int w=v;

while(w!=-1&&trans[w][c]==x)//该部分字符串长度越来越短 一定都是要转移到y的 因为y保留的是x原本较短的部分

{

trans[w][c]=y;

w=slink[w];

}

minlen[y]=maxlen[slink[y]]+1;

return z;

}

};

**后缀自动机-例2（BZOJ3473）-onenote**

int n,k;

const int MAXL=2e6+5;

string s[MAXL];

const int MAXCH=26;

const char CH='a';

struct SAM{

int maxlen[MAXL],minlen[MAXL],trans[MAXL][MAXCH],slink[MAXL];

int c[MAXL],a[MAXL];

int last[MAXL],cnt[MAXL];

ll dp[MAXL];

int tot;

int new\_state(int \_maxlen,int \_minlen,int \*\_trans,int \_slink){

maxlen[tot]=\_maxlen;

minlen[tot]=\_minlen;

for(int i=0;i<MAXCH;i++){

if(\_trans==NULL)trans[tot][i]=-1;

else trans[tot][i]=\_trans[i];

}

slink[tot]=\_slink;

return tot++;

}

int add\_char(char ch,int u){

int c=ch-CH;

int z=new\_state(maxlen[u]+1,-1,NULL,-1);

int v=u;

while(v!=-1&&trans[v][c]==-1){

trans[v][c]=z;

v=slink[v];

}

if(v==-1){

minlen[z]=1;

slink[z]=0;

return z;

}

int x=trans[v][c];

if(maxlen[v]+1==maxlen[x]){

minlen[z]=maxlen[x]+1;

slink[z]=x;

return z;

}

int y=new\_state(maxlen[v]+1,-1,trans[x],slink[x]);

minlen[x]=maxlen[y]+1;

slink[x]=y;

minlen[z]=maxlen[y]+1;

slink[z]=y;

int w=v;

while(w!=-1&&trans[w][c]==x){

trans[w][c]=y;

w=slink[w];

}

minlen[y]=maxlen[slink[y]]+1;

return z;

}

void toposort(){

for(int i=1;i<tot;++i)++c[maxlen[i]];

for(int i=1;i<tot;i++)c[i]+=c[i-1];

for(int i=tot-1;i>=1;--i)a[c[maxlen[i]]--]=i;

}

void solve(){

ll ans=0;int u;

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

u=0;

for(int j=0;j<s[i].size();j++){

u=trans[u][s[i][j]-CH];

int now=u;

while(now!=-1&&last[now]!=i)

++cnt[now],last[now]=i,now=slink[now];

}

}

toposort();

cnt[0]=0;

for(int i=1;i<tot;++i)

u=a[i],dp[u]=dp[slink[u]]+(cnt[u]>=k?maxlen[u]-minlen[u]+1:0);

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

u=0;ans=0;

for(int j=0;j<s[i].size();j++){

u=trans[u][s[i][j]-CH];

ans+=dp[u];

}

printf("%lld ",ans);

}

}

}sam;

int main()

{

read(n);read(k);

int pres=sam.new\_state(0,0,NULL,-1);

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

cin>>s[i];

pres=0;

for(int j=0;j<s[i].size();j++)

pres=sam.add\_char(s[i][j],pres);

}

sam.solve();

return 0;

}

# DP

## dp套dp

1. **HDU5548-麻将胡牌情况数**

int t,n,m,tot;

int id[(1<<18)|5],nxt[1008][8],val[1008],dp[207][207][78];//dp[i][j][k] 看到第i种牌时 已选j张,状态集为k

queue<int>que;

inline int encode(int n\_2,int n\_1,int have2){//start from n-2 / n-1 /if have a pair

int re=0;

re=re\*3+n\_2;

re=re\*3+n\_1;

re=re\*2+have2;

return re;

}

inline void decode(int v,int &n\_2,int &n\_1,int &have2){

have2=v%2;v>>=1;

n\_1=v%3;v/=3;

n\_2=v;

}

int get\_nxt\_st(int st,int cnt){//得到 从st 该位选cnt个 能转移到的状态

int nxt\_st=0;

int n\_2,n\_1,have2;

int x\_2,x\_1,xave2;

for(int n=0;n<18;n++){

if(st&(1<<n)){

decode(n,n\_2,n\_1,have2);

x\_2=n\_1;x\_1=cnt-n\_2-n\_1;xave2=have2;

if(x\_1>=0){

int new\_st=encode(x\_2,x\_1%3,xave2);

nxt\_st|=(1<<new\_st);

if(!have2&&x\_1>=2){

new\_st=encode(x\_2,x\_1-2,1);

nxt\_st|=(1<<new\_st);

}

}

}

}

return nxt\_st;

}

void initDP(){//初始化状态转移

que.push(1);id[0]=++tot;

while(!que.empty()){

int st=que.front();que.pop();

for(int cnt=0;cnt<=4;cnt++){

int nxt\_st=get\_nxt\_st(st,cnt);

if(!id[nxt\_st])id[nxt\_st]=++tot,val[tot]=nxt\_st,que.push(nxt\_st);

nxt[id[st]][cnt]=id[nxt\_st];

}

}

}

int solve(){

memset(dp,0,sizeof(dp));

dp[0][0][1<<id[0]]=1;//0 0 0的状态encode结果就是0 所以id[0]

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

int lim=(i<n?4:0);//该位最多能取的数量

for(int j=0;j<=m;j++)//已取的数量

for(int t=1;t<=tot;t++)//所有的可能状态

if(dp[i][j][t])

for(int k=0;k<=lim;k++)//枚举这一位具体取多少

addi(dp[i+1][j+k][nxt[t][k]],dp[i][j][t] );

}

int re=0;

for(int i=1;i<=tot;i++)

if(val[i]&2)//出现了对子

addi(re,dp[n+3][m][i]);

return re;

}

int main(){

initDP();

// cout<<tot<<endl;//总共只有68种状态

read(t);

for(int Case=1;Case<=t;Case++){

read(n);read(m);

printf("Case #%d: %d\n",Case,solve());

}

return 0;

}

1. **HDU4899-LCS为m的序列个数**

int n,m,t;

char a[20],c[10]="AGCT";

int trans[(1<<15)|5][4],cnt[1<<15|5],dp[2][1<<15|5],ans[20],d[20],pre[20];

void init(){

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

if(i)cnt[i]=cnt[i^(i&-i)]+1;

for(int j=0;j<n;j++)d[j+1]=d[j]+(bool)(i&(1<<j));

for(int k=0;k<4;k++){

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

pre[j]=max(pre[j-1],d[j]);

if(c[k]==a[j])pre[j]=max(pre[j],d[j-1]+1);

}

trans[i][k]=0;

for(int j=0;j<n;j++)if(pre[j+1]-pre[j])trans[i][k]|=(1<<j);

}

}

}

int main(){

read(t);

while(t--){

scanf("%s",a+1);n=strlen(a+1);

init();

read(m);

memset(ans,0,sizeof(ans));memset(dp,0,sizeof(dp));

dp[0][0]=1;int p=0;

for(int i=1;i<=m;i++,p^=1){

memset(dp[p^1],0,sizeof(dp[p^1]));

for(int j=0;j<(1<<n);j++)

for(int k=0;k<4;k++)

addi(dp[p^1][trans[j][k]],dp[p][j]);

}

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

addi(ans[cnt[i]],dp[p][i]);

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

printf("%d\n",ans[i]);

}

return 0;

}

## 决策单调性

**决策单调性-例1-BZOJ1010**

int n,L;

int l[MAX],stk[MAX],st,en;

ll a[MAX],sum[MAX],dp[MAX];

//由j点dp更新i点

ll turn(int j,int i){

ll x=i-j+sum[i]-sum[j]-1;

return dp[j]+(x-L)\*(x-L);

}

//确定某点x应该在哪个决策区间中

int find(int x)

{

int left=st,right=en,mid,re=1;

while(left<=right){

mid=(left+right)/2;

if(l[stk[mid]]==x)return mid;

if(l[stk[mid]]<x)re=mid,left=mid+1;

else right=mid-1;

}

return re;

}

int main()

{

read(n);read(L);

for(int i=1;i<=n;i++)read(a[i]),sum[i]=sum[i-1]+a[i];

st=en=1;l[1]=1;

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

dp[i]=turn(stk[find(i)],i);//由对应的决策点确定该点

while(st<=en&& l[stk[en]]>i && turn(i,l[stk[en]])<turn(stk[en],l[stk[en]]) )--en;

// if(!en)//栈中已没有确定的决策区间 整个区间均为此 可以去掉

// {

// stk[++en]=i;l[i]=i;continue;

// }

//不然二分最后一个决策 寻找更新的位置

int left=max(i+1,l[stk[en]]),right=n;

while(left<=right){

int mid=(left+right)/2;

if(turn(i,mid)>turn(stk[en],mid))left=mid+1;

else right=mid-1;

}

//如果i整体比前一个决策差

if(turn(i,left)>turn(stk[en],left))continue;

stk[++en]=i,l[i]=left;

}

printf("%lld\n",dp[n]);

return 0;

}

## 树形依赖DP

1. **P2014-树上过根的最大权连通块背包**

#define SZ 5004

int n,m,dcnt,ecnt;

int h[SZ],siz[SZ],dp[SZ][SZ],fa[SZ],w[SZ],v[SZ];

int dfn[SZ],dwho[SZ];//dfsÐòµÄ±àºÅ

struct node{

int to,nxt;

node(){};

node(int \_to,int \_nxt):to(\_to),nxt(\_nxt){};

}edg[SZ<<1];

void add\_edge(int who){

int f=fa[who];

// if(!f)return;

edg[++ecnt]=node(who,h[f]);

h[f]=ecnt;

}

void dfs(int now){

siz[now]=1;dfn[now]=++dcnt;dwho[dcnt]=now;

for(int i=h[now];i;i=edg[i].nxt){

int to=edg[i].to;

dfs(to);siz[now]+=siz[to];

}

}

int main(){

read(n);read(m);

for(int i=1;i<=n;i++)read(fa[i]),read(v[i]),w[i]=1,add\_edge(i);

dfs(0);

for(int i=0;i<=m;i++)dp[dcnt+1][i]=dp[dcnt+2][i]=-1000000000;

dp[dcnt+1][0]=0;

for(int i=dcnt;i>=1;i--){

int lo=dwho[i];

for(int j=0;j<=m;j++){

if(j<w[lo])dp[i][j]=max(dp[i+siz[lo]][j],0);

else dp[i][j]=max(max(dp[i+siz[lo]][j],dp[i+1][j-w[lo]]+v[lo]),0);

}

}

printf("%d\n",dp[1][m]);

return 0;

}

1. **树上过根的最大权连通块背包-原版**

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <memory.h>

using namespace std;

//w=weight v=value

#define SZ 5004

int n,m,fa[SZ],w[SZ],v[SZ],fc[SZ],nc[SZ],siz[SZ],dp[SZ][SZ];

int st[SZ],t=0,dl[SZ];

void ass(int x) {int f=fa[x]; if(f) nc[x]=fc[f], fc[f]=x;}

void dfs(int p)

{

siz[p]=1; st[p]=++t; dl[t]=p;

for(int c=fc[p];c;c=nc[c])

{

dfs(c); siz[p]+=siz[c];

}

}

#define FO(x) {freopen(#x".in","r",stdin); freopen(#x".out","w",stdout);}

int main()

{

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

for(int i=1;i<=n;i++) scanf("%d%d%d",fa+i,w+i,v+i), ass(i);

dfs(1);

for(int p=0;p<=m;p++) dp[n+1][p]=dp[n+2][p]=-1000000000;

dp[n+1][0]=0; //原来这里忘了写

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

{

int x=dl[i];

for(int p=0;p<=m;p++)

{

if(p<w[x]) dp[i][p]=max(dp[i+siz[x]][p],0);

else dp[i][p]=max(max(dp[i+siz[x]][p],dp[i+1][p-w[x]]+v[x]),0);

}

}

printf("%d\n",dp[1][m]);

}

1. **树上过根的最大权连通块背包-复刻版**

#define SZ 5004

int n,m,dcnt,ecnt;

int h[SZ],siz[SZ],dp[SZ][SZ],fa[SZ],w[SZ],v[SZ];

int dfn[SZ],dwho[SZ];//dfsÐòµÄ±àºÅ

struct node{

int to,nxt;

node(){};

node(int \_to,int \_nxt):to(\_to),nxt(\_nxt){};

}edg[SZ<<1];

void add\_edge(int who){

int f=fa[who];if(!f)return;

edg[++ecnt]=node(who,h[f]);

h[f]=ecnt;

}

void dfs(int now){

siz[now]=1;dfn[now]=++dcnt;dwho[dcnt]=now;

for(int i=h[now];i;i=edg[i].nxt){

int to=edg[i].to;

dfs(to);siz[now]+=siz[to];

}

}

int main(){

read(n);read(m);

for(int i=1;i<=n;i++)read(fa[i]),read(w[i]),read(v[i]),add\_edge(i);

dfs(1);

for(int i=0;i<=m;i++)dp[n+1][i]=dp[n+2][i]=-1000000000;

dp[n+1][0]=0;

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

int lo=dwho[i];

for(int j=0;j<=m;j++){

if(j<w[lo])dp[i][j]=max(dp[i+siz[lo]][j],0);

else dp[i][j]=max(max(dp[i+siz[lo]][j],dp[i+1][j-w[lo]]+v[lo]),0);

}

}

printf("%d\n",dp[1][m]);

return 0;

}

## 斯坦纳树

1. **BZOJ2595-必须包含若干点的最小曼哈顿生成树**

int n,m,k;

int a[10][10],f[10][10][1<<10];

struct tup{

int first,second,thd;

tup(int \_fst=0,int \_sec=0,int \_thd=0):first(\_fst),second(\_sec),thd(\_thd){}

}pre[10][10][1<<10],zero;

queue<pii>que;

int vis[10][10];

void spfa(int sta){

const static int dx[4]={1,-1,0,0},dy[4]={0,0,1,-1};

while(!que.empty()){

pii u=que.front();que.pop();vis[u.first][u.second]=0;

for(int k=0;k<4;k++){

int x=dx[k]+u.first,y=dy[k]+u.second;

if(x<0||y<0||x>=n||y>=m)continue;

if(f[x][y][sta]>f[u.first][u.second][sta]+a[x][y]){

f[x][y][sta]=f[u.first][u.second][sta]+a[x][y];

pre[x][y][sta]=tup(u.first,u.second,sta);

if(!vis[x][y])que.push(mp(x,y)),vis[x][y]=1;

}

}

}

}

void dfs(int i,int j,int sta){

if(i==INF||pre[i][j][sta].thd==0)return;

vis[i][j]=1;tup tp=pre[i][j][sta];

dfs(tp.first,tp.second,tp.thd);

if(tp.first==i&&tp.second==j)dfs(i,j,sta-tp.thd);

}

int main(){

read(n);read(m);

memset(f,0x3f,sizeof(f));

// fill(&f[0][0][0],&f[10][10][1<<10],INF);

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

for(int j=0;j<m;j++){

scanf("%d",&a[i][j]);

if(!a[i][j])f[i][j][1<<(k++)]=0;

}

for(int sta=1;sta<(1<<k);sta++){

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

for(int j=0;j<m;j++){

for(int s=sta&(sta-1);s;s=sta&(s-1)){

if(f[i][j][sta]>f[i][j][s]+f[i][j][sta-s]-a[i][j]){

f[i][j][sta]=f[i][j][s]+f[i][j][sta-s]-a[i][j];

pre[i][j][sta]=tup(i,j,s);

}

}

if(f[i][j][sta]!=INF){

que.push(mp(i,j));vis[i][j]=1;

}

}

spfa(sta);

}

int x,y;

for(int i=0;i<n;i++)for(int j=0;j<m;j++)if(!a[i][j]){x=i;y=j;break;}

printf("%d\n",f[x][y][(1<<k)-1]);

dfs(x,y,(1<<k)-1);

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

for(int j=0;j<m;j++){

if(!a[i][j])putchar('x');//景点

else if(vis[i][j])putchar('o');//有人

else putchar('\_');

if(j==m-1)puts("");

}

return 0;

}

1. **BZOJ4774-给定图要求选边使得i与n-i+1连通(i≤d)**

struct node{

int to,nxt,weight;

node(){};

node(int \_to,int \_nxt,int \_weight):to(\_to),nxt(\_nxt),weight(\_weight){};

}edg[MAX<<1];

int n,m,d,ecnt,tot;

int h[MAX],dp[257][MAX],tp[20];

void add\_edge(int u,int v,int w){

edg[++ecnt]=node(v,h[u],w);

h[u]=ecnt;

}

struct qs{

int lo,val;

qs(){};

qs(int \_lo,int \_val):lo(\_lo),val(\_val){}

bool operator<(const qs& z)const{return val>z.val;}

};

priority\_queue<qs>que;

int main(){

read(n);read(m);read(d);

for(int u,v,w,i=1;i<=m;i++){

read(u);read(v);read(w);

add\_edge(u,v,w);add\_edge(v,u,w);

}

tot=(1<<d);

memset(dp,0x3f,sizeof(dp));

memset(tp,0x3f,sizeof(tp));

for(int i=1;i<=d;i++)

dp[1<<i-1][i]=dp[1<<i-1+d][n+1-i]=0;

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

int \*tem=dp[i];

for(int j=(i-1)&i;j>(i^j);j=(j-1)&i){//中间的条件永远取不到等号

for(int k=1,\*l1=dp[j],\*l2=dp[i^j];k<=n;++k)

tem[k]=min(tem[k],l1[k]+l2[k]);

}

for(int k=1;k<=n;k++)

if(tem[k]<INF)que.push(qs(k,tem[k]));

while(!que.empty()){

qs now=que.top();que.pop();

int pos=now.lo;

if(now.val>tem[pos])continue;

for(int j=h[pos];j;j=edg[j].nxt){

int to=edg[j].to,td=tem[pos]+edg[j].weight;

if(tem[to]>td)tem[to]=td,que.push(qs(to,tem[to]));

}

}

if((i&(tot-1))==(i>>d)){

int lin=INF;

for(int j=1;j<=n;j++)lin=min(lin,tem[j]);

tp[i>>d]=lin;

}

}

for(int i=1;i<tot;i++)

for(int j=(i-1)&i;j>(i^j);j=(j-1)&i)

tp[i]=min(tp[i],tp[j]+tp[i^j]);

printf("%d\n",tp[tot-1]!=INF?tp[tot-1]:-1);

return 0;

}

## 四边形不等式

**四边形不等式(HDU3516)**

struct point{

int x,y;

}a[MAX];

int dis(point a,point b,point c,point d){

return (c.x-a.x)+(b.y-d.y);

}

int n;

int dp[MAX][MAX],s[MAX][MAX];

int main()

{

while(~scanf("%d",&n)){

for(int i=1;i<=n;i++)read(a[i].x),read(a[i].y);

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

memset(dp,0,sizeof(dp));

for(int len=2;len<=n;len++)

for(int st=1;st+len-1<=n;st++){

int en=st+len-1;

dp[st][en]=INF;

for(int i=s[st][en-1];i<=s[st+1][en]&&i<en;i++){

if(dp[st][i]+dp[i+1][en]+dis(a[st],a[i],a[i+1],a[en])<dp[st][en]){

dp[st][en]=dp[st][i]+dp[i+1][en]+dis(a[st],a[i],a[i+1],a[en]);

// if(i==en)continue;

// cout<<st<<" "<<i<<" "<<i+1<<" "<<en<<endl;

s[st][en]=i;

}

}

}

printf("%d\n",dp[1][n]);

}

return 0;

}

**四边形不等式-例2-POJ1160**

int n,m;

int dp[32][303];

int a[305];

int w[303][303];

int s[32][303];

int getw(int i,int j){

if(i>=j)return 0;

if(w[i][j])return w[i][j];

return w[i][j]=a[j]-a[i]+getw(i+1,j-1);

}

int main()

{

while(~scanf("%d%d",&n,&m)){

for(int i=1;i<=n;i++)read(a[i]);

memset(dp,0x3f,sizeof(dp));

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

dp[1][i]=getw(1,i),s[1][i]=1;//此s实际不是最后位置 而是用于循环中确定范围

for(int i=2;i<=m;++i){

s[i][n+1]=n;//尽管无意义 但因为循环中出现s[i][j+1] 为划定范围 加上此

for(int j=n;j>i;--j){

for(int k=s[i-1][j];k<=s[i][j+1];++k)

{

if(dp[i-1][k]+getw(k+1,j)<dp[i][j]){

dp[i][j]=dp[i-1][k]+getw(k+1,j);

s[i][j]=k;

}

}

}

}

printf("%d\n",dp[m][n]);

}

return 0;

}

## 斜率优化

**斜率优化-例1-HDU2829**

int n,m;

int a[MAX],dp[MAX][2],cost[MAX],sum[MAX],q[MAX],st,en;

int now;

int getx(int i){return sum[i];}

int gety(int i){return dp[i][now^1]-cost[i]+sum[i]\*sum[i];}

int main()

{

now=0;

while(scanf("%d%d",&n,&m)&&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++)cost[i]=cost[i-1]+sum[i-1]\*a[i];

// st=en=1;q[st]=0;

for(int i=1;i<=n;i++)dp[i][now]=cost[i];

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

{

now^=1;

st=en=1;q[st]=0;

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

{

while(st<en&&(gety(q[st+1])-gety(q[st]))<=sum[j]\*(getx(q[st+1])-getx(q[st]) ))++st;

dp[j][now]=dp[q[st]][now^1]+cost[j]-cost[q[st]]-sum[q[st]]\*(sum[j]-sum[q[st]]);

// printf("%d!\n",dp[j][now]);

while(st<en&&( gety(j) -gety(q[en]) )\*( getx(q[en])-getx(q[en-1]) )<=( gety(q[en])-gety(q[en-1]) ) \*( getx(j)-getx(q[en]) ) )--en;

q[++en]=j;

}

}

printf("%d\n",dp[n][now]);

}

return 0;

}

**斜率优化-例2-POJ1180**

int n,s,st,en;

int que[MAX],f[MAX],t[MAX];

ll sumt[MAX],sumf[MAX],dp[MAX];

int main()

{

while(~scanf("%d%d",&n,&s)){

for(int i=1;i<=n;++i){read(t[i]);read(f[i]);}

sumt[n+1]=sumf[n+1]=dp[n+1]=0;

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

sumt[i]=sumt[i+1]+t[i];

sumf[i]=sumf[i+1]+f[i];

}

st=1,en=1;que[st]=n+1;

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

while(st<en&& (dp[que[st+1]]-dp[que[st]])<=sumf[i]\*(sumt[que[st+1]]-sumt[que[st]]))

++st;

int j=que[st];

dp[i]=dp[j]+(s+sumt[i]-sumt[j])\*sumf[i];

while(st<en&& (dp[i]-dp[que[en]])\*(sumt[que[en]]-sumt[que[en-1]])<=(dp[que[en]]-dp[que[en-1]])\*(sumt[i]-sumt[que[en]]))

--en;

que[++en]=i;

}

printf("%lld\n",dp[1]);

}

return 0;

}

# 随机化

## 图随机化

1. **随机化边权判连通图去掉一些边后是否连通-BZOJ3569**

int ecnt,n,m,q,k,cnt;

int dp[MAX],h[MAX],fa[MAX],eval[MAX],a[50];

bool v\_vi[MAX],e\_vi[MAX];//点是否使用过 边是否使用过

struct node{

int to,nxt,val;

}edg[MAX<<1];

void add\_edge(int u,int v){

edg[++ecnt]=node{v,h[u],0};

h[u]=ecnt;

}

void dfs(int now,int pre){

v\_vi[now]=1;

for(int i=h[now];i;i=edg[i].nxt){

int to=edg[i].to;

if(to==pre||v\_vi[to])continue;

fa[to]=now;e\_vi[i>>1]=1;

dfs(to,now);

}

}

void dfs2(int now){

for(int i=h[now];i;i=edg[i].nxt){

int to=edg[i].to;

if(fa[to]!=now)continue;

dfs2(to);

eval[i>>1]^=dp[to];

dp[now]^=dp[to];

}

}

int main()

{

read(n);read(m);

ecnt=1;

for(int u,v,i=1;i<=m;i++){

read(u);read(v);add\_edge(u,v);add\_edge(v,u);

}

dfs(1,0);

srand(19260817);

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

if(e\_vi[i])continue;

eval[i]=rand()%MOD+1;

dp[edg[i<<1].to]^=eval[i];

dp[edg[i<<1|1].to]^=eval[i];

}

dfs2(1);

read(q);

while(q--){

read(k);

bool ans=1;

memset(a,0,sizeof(a));

for(int u,i=1;i<=k;i++){

read(u);u^=cnt;u=eval[u];

for(int j=31;j>=0;j--){

if(!(u&(1<<j)))continue;

if(!a[j]){a[j]=u;break;}

u^=a[j];

}

ans&=(u!=0);

}

cnt+=ans;

puts(ans?"Connected":"Disconnected");

}

return 0;

}

# 基础&常用

二分模拟乘法-用于乘起来会超LL的取模意义乘法运算

ll multmod(ll a,ll b){

a%=MOD;b%=MOD;

if(b<0)b+=MOD;

if(a<0)a+=MOD;

ll re=0;

while(b){

if(b&1LL){

re+=a;if(re>=MOD)re-=MOD;

}

a=a<<1;

if(a>=MOD)a-=MOD;

b=b>>1;

}

return re;

}

二维单调队列-例：处理出所有nn方阵中最大值

int n,m,k;

int x[1005][1005];

int maxh[1005][1005],maxl[1005][1005];

int que[1005],st,en;

void getmax(int len,int z[1005][1005])//连续长度为len的区间

{

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

st=1,en=0;

for(int j=1;j<=m;j++){

while( st<=en&& que[st]+len-1<j ) ++st;

while( st<=en&& que[en]<=z[i][j] ) --en;

que[++en]=z[i][j];

maxh[i][j]=que[st];

}

}

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

st=1,en=0;

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

while( st<=en&& que[st]+len-1<j ) ++st;

while( st<=en&& que[en]<=maxh[j][i] )--en;

que[++en]=maxh[j][i];

maxl[j][i]=que[st];

}

}

}

int main()

{

read(n);read(m);read(k);

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

for(int j=1;j<=m;j++)read(x[i][j]);

getmax(k,x);

int an=0;

for(int i=k;i<=n;i++)

for(int j=k;j<=n;j++)an=max(an,maxl[i][j]);

printf("%d\n",an);

return 0;

}

## 离散化

int n;

int a[MAX];

vector<int>v;

int getid(int x){return lower\_bound(v.begin(),v.end(),x)-v.begin()+1;}

int main()

{

for(int i=1;i<=n;i++)v.pb(a[i]);

sort(v.begin(),v.end());v.erase(unique(v.begin(),v.end()),v.end());

}