目录

[头文件 1](#_Toc13668560)

[数学 2](#_Toc13668561)

[波利亚定理 2](#_Toc13668562)

[快速沃尔什变换 6](#_Toc13668563)

[类欧几里得 8](#_Toc13668564)

[欧拉函数、欧拉降幂 11](#_Toc13668565)

[矩阵相关 11](#_Toc13668566)

[数据结构 12](#_Toc13668567)

[CDQ分治 12](#_Toc13668568)

[KD-TREE 22](#_Toc13668569)

[分块 26](#_Toc13668570)

[笛卡尔树 30](#_Toc13668571)

[分块 31](#_Toc13668572)

[莫队 34](#_Toc13668573)

[线段树 36](#_Toc13668574)

[整体二分 37](#_Toc13668575)

[字典树 38](#_Toc13668576)

[主席树 44](#_Toc13668577)

[并查集 48](#_Toc13668578)

[splay 48](#_Toc13668579)

[link-cut tree 54](#_Toc13668580)

[图论 60](#_Toc13668581)

[2-SAT 60](#_Toc13668582)

[LCA 62](#_Toc13668583)

[点分治 62](#_Toc13668584)

[二分图 66](#_Toc13668585)

[三元环、四元环 67](#_Toc13668586)

[树剖 69](#_Toc13668587)

[虚树 73](#_Toc13668588)

[树分块 76](#_Toc13668589)

[强连通分量 78](#_Toc13668590)

[树的同构 79](#_Toc13668591)

[字符串 80](#_Toc13668592)

[KMP 80](#_Toc13668593)

[最小表示法 80](#_Toc13668594)

[AC自动机 81](#_Toc13668595)

[后缀自动机 85](#_Toc13668596)

[hiho版 85](#_Toc13668597)

[SAM(hiho-struct版-无说明) 90](#_Toc13668598)

[后缀数组 92](#_Toc13668599)

[序列自动机 93](#_Toc13668600)

[DP 94](#_Toc13668601)

[dp套dp 94](#_Toc13668602)

[决策单调性 95](#_Toc13668603)

[树形依赖DP 96](#_Toc13668604)

[斯坦纳树 97](#_Toc13668605)

[四边形不等式 99](#_Toc13668606)

[斜率优化 100](#_Toc13668607)

[树形DP 102](#_Toc13668608)

[概率DP 104](#_Toc13668609)

[数位DP 105](#_Toc13668610)

[插头DP 106](#_Toc13668611)

[nlogn最长上升子序列 112](#_Toc13668612)

[网络流 112](#_Toc13668613)

[上下界网络流 112](#_Toc13668614)

[最小割 116](#_Toc13668615)

[随机化 118](#_Toc13668616)

[图随机化 118](#_Toc13668617)

[基础&常用 119](#_Toc13668618)

[二分模拟乘法 119](#_Toc13668619)

[二维单调队列 119](#_Toc13668620)

[离散化 119](#_Toc13668621)

## 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

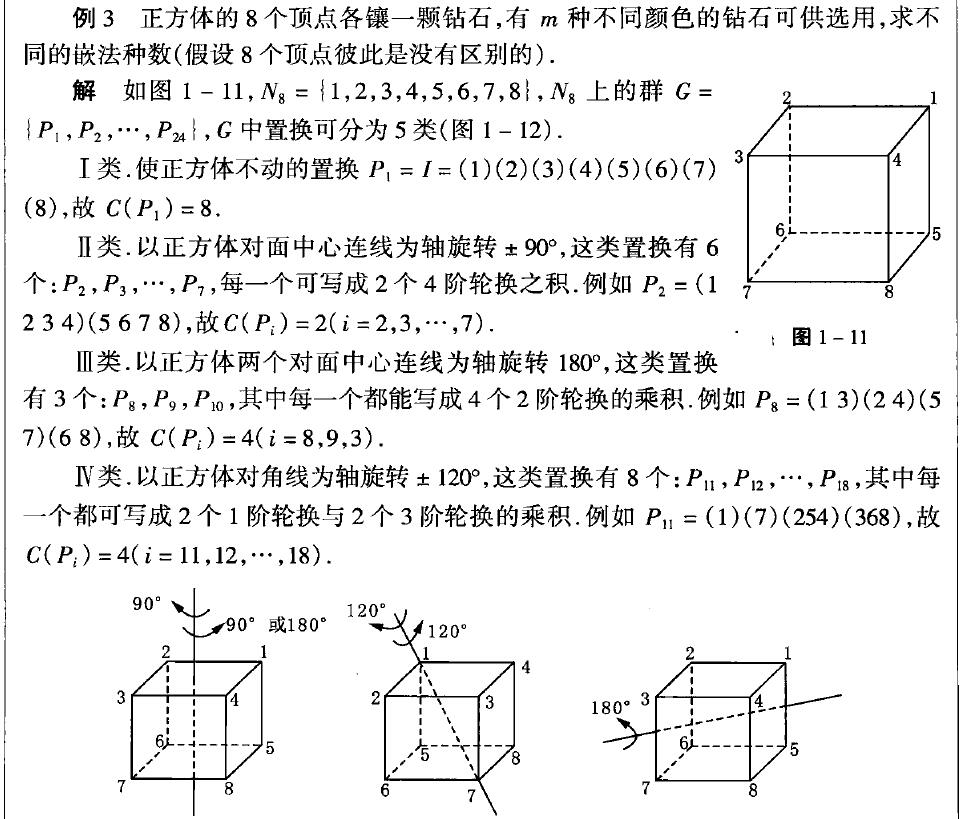
在每次旋转90度和270度的时候，可以发现分为3个循环，每个循环节长度为4

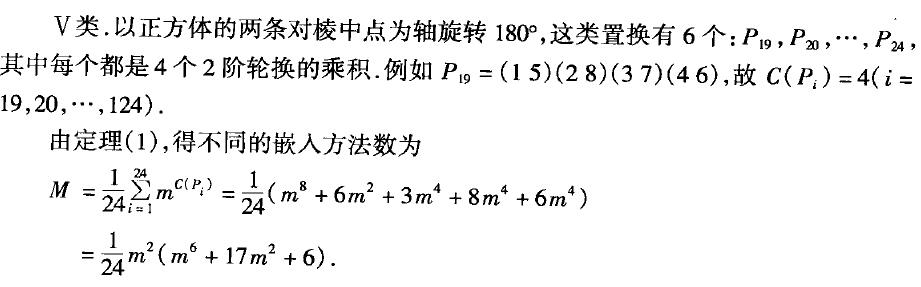
在每次旋转180度的时候，可以发现分为6个循环，每个循环节长度为2

4、通过两条对立的棱的中心，分别旋转180度，有6组棱

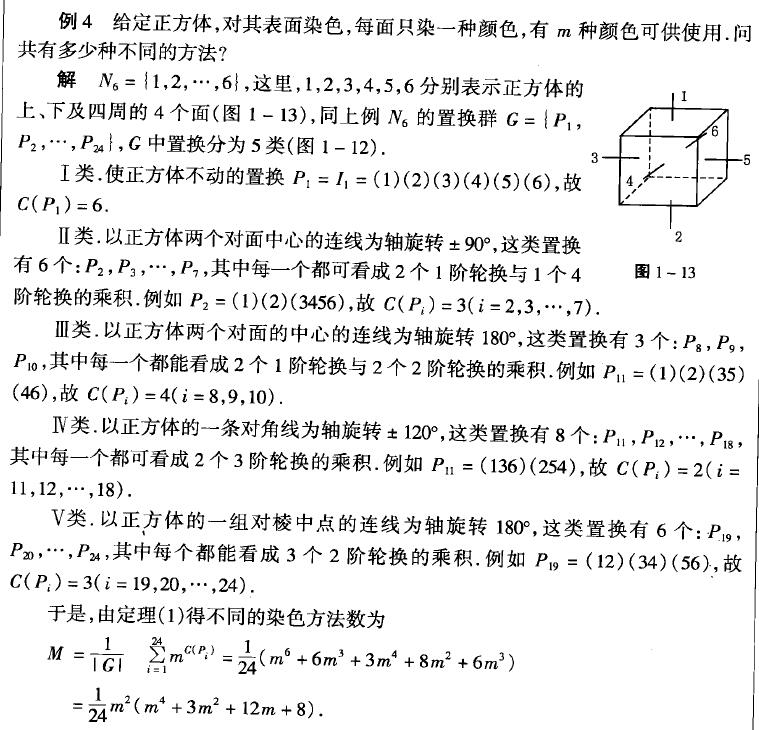
在每次旋转的时候，分为6个循环，每个循环节长度为2

**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);

}

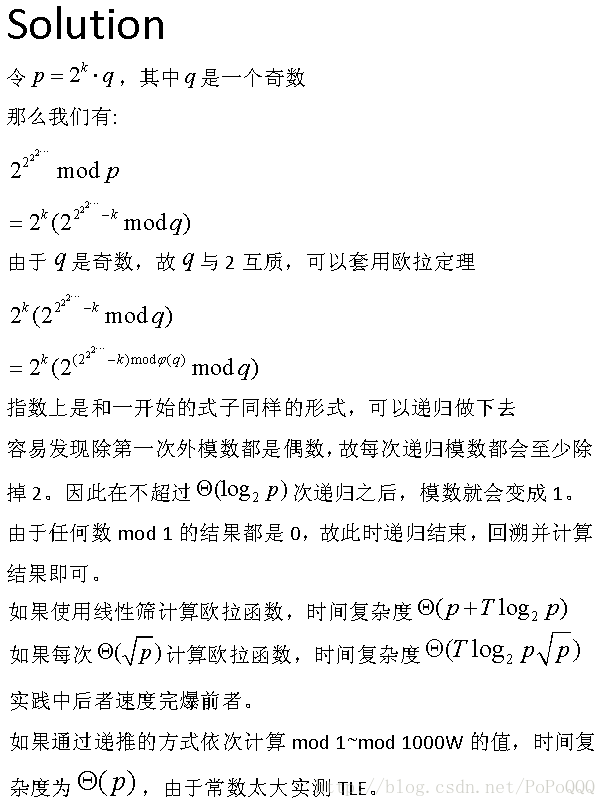
return 0;

}

## 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;

}

## 矩阵相关

const int m\_num=5;//矩阵的大小

/\*

矩阵 e[i][j]表示第i行第j列的矩阵元素的值

\*/

struct matrix

{

double 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);//新形成的矩阵规模为 左行右列

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

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

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

ret.e[i][j]+=1LL\*e[i][k]\*tem.e[k][j];

return ret;

}

matrix operator + (const matrix &tem)const

{

matrix ret=matrix(row,col);

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

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

return ret;

}

void getE()//化为单位阵

{

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

for(int j=1;j<=col;j++)e[i][j]=(i==j?1:0);

}

};

matrix m\_qpow(matrix tem,int x)//矩阵快速幂

{

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

ret.getE();

while(x)

{

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

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

}

return ret;

}

double det(double a[][MAXN],int n)

{

int i, j, k, sign = 0;

double ret = 1;

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

for(j = 0;j < n;j++) b[i][j] = a[i][j];

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

{

if(sgn(b[i][i]) == 0)

{

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

if(sgn(b[j][i]) != 0)

break;

if(j == n)return 0;

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

swap(b[i][k],b[j][k]);

sign++;

}

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

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

b[i][k]/=b[i][i];

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

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

b[j][k] -= b[j][i]\*b[i][k];

}

if(sign & 1)ret = -ret;

return ret;

}

# 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;

}

## 分块

**例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;

}

**例2-CF1129D-将数组划分成若干段每段只出现1次的数的个数不大于k的方案数**

const int B=318,Bcnt=318;

int n,k,ans;

int pre[MAX],dp[MAX],tag[Bcnt],las[MAX],sum[MAX];//pre[i] 下标i前一个出现该位置值的位置

int val[Bcnt][MAX];//某块i中只作为后缀的开始出现一次的数为j的位置的dp值之和

int bel(int x){return (x-1)/B+1;}//返回x在的块的ID

void Ins(int u,int v){

int id=bel(u);

sum[u]-=tag[id];

addi(ans,v);

addi(val[id][n+sum[u]],v);

}

void update(int pos,int v){//单点加val

int id=bel(pos);

if(sum[pos]+tag[id]<=k)//原本是<=k的

subi(ans,dp[pos-1]);//修改原先对ans的影响

subi(val[id][sum[pos]+n],dp[pos-1]);//修改原先对val的影响

sum[pos]+=v;

if(sum[pos]+tag[id]<=k)addi(ans,dp[pos-1]);

addi(val[id][sum[pos]+n],dp[pos-1]);

}

void update\_b(int L,int R,int v){//整个区间[L,R]进行更新

if(L>R)return;

int bl=bel(L),br=bel(R);

if(bl+1>=br)//在同一个块中

for(int i=L;i<=R;i++)update(i,v);

else{

for(int i=L;i<=bl\*B;i++)update(i,v);//修改区间的头部非完整块

for(int i=(br-1)\*B+1;i<=R;i++)update(i,v);//修改区间的尾部非完整块

for(int i=bl+1;i<br;i++){

if(~v)subi(ans,val[i][k-tag[i]+n]);

else addi(ans,val[i][k-tag[i]+1+n]);

tag[i]+=v;

}

}

}

int main(){

read(n);read(k);

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

read(u);pre[i]=las[u];las[u]=i;

}

dp[0]=1;//从1转移

Ins(1,1);

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

update\_b(pre[i]+1,i,1);

update\_b(pre[pre[i]]+1,pre[i],-1);

Ins(i+1,dp[i]=ans);

}

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

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;

}

**例3-wannafly-wintercamp-2019-Day7-F**

int n,m;

int dcnt,ans;//字典树结点个数

int cnt[MAX<<1],dp[105][2],nxt[MAX][2];//字典树

int mcnt[100][2];//小于等于m的数中二进制下第i位为j的数的个数

void insert(int k){

int lo=0;//字典树的位置

for(int i=31;i>=0;i--){

int val=(k>>i)&1;

if(!nxt[lo][val])nxt[lo][val]=++dcnt;

addi(dp[i][val],cnt[nxt[lo][!val]]);

lo=nxt[lo][val];++cnt[lo];

}

}

inline ll cal\_1(ll x){//小于等于x的奇数个数

return (x+1)/2;

}

void get\_mcnt(ll val){//对于val计算mcnt

for(int i=0;i<=31;i++){

ll ge=(val>>i);//数位dp无lim的部分(包含0)

if(ge==0)break;

ll num=cal\_1(ge-1);//无lim部分该位置为1的个数

addi(mcnt[i][1],(num<<i)%MOD);

addi(mcnt[i][0],((ge-num)<<i)%MOD);

ll limcnt=(val-(ge<<i)+1)%MOD;//有lim的数的个数

if(val&(1<<i))addi(mcnt[i][1],limcnt);

else addi(mcnt[i][0],limcnt);

sub(mcnt[i][0],1);

}

}

int main(){

read(n);read(m);

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

int k;read(k);insert(k);

}

get\_mcnt(m);

for(int i=0;i<=31;i++)

addi(ans,add(mul(m-mcnt[i][1],dp[i][0]),mul(dp[i][1],mcnt[i][1]) ));

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

return 0;

}

**例4-LOJ517**

const int MAXN=2e5+5;

int n,m,k,sort\_val,tot\_val;

int tcnt,rt,pcnt;

int opt;

int cnt[MAX][31],pre[MAXN][31],sum[31],bval[MAXN];

vector<int>pts;

struct Trie{

int ls,rs,ncnt;

Trie(int \_ls=0,int \_rs=0,int \_ncnt=0):ls(\_ls),rs(\_rs),ncnt(\_ncnt){}

}t[MAX];

inline void insert(int dep,int val,int& lo){

if(!lo){

lo=++tcnt;t[lo]=Trie(0,0,0);

}

++t[lo].ncnt;

for(register int i=0;i<31;i++)

if(val&(1<<i))++cnt[lo][i];

if(dep==-1)

return;

if(val&(1<<dep))

insert(dep-1,val,t[lo].rs);

else insert(dep-1,val,t[lo].ls);

}

inline ll query\_pre(int x){

ll re=0;

for(register int i=0;i<31;i++){

if(tot\_val&(1<<i))re+=(1LL\*(x-pre[x][i])<<(1LL\*i));

else re+=1LL\*pre[x][i]<<(1LL\*i);

}

return re;

}

inline ll query\_pts(){

for(int u:pts)

for(register int i=0;i<31;i++)sum[i]+=cnt[u][i];

ll re=0;

for(register int i=0;i<31;i++){

if(tot\_val&(1<<i))re+=(1LL\*(k-sum[i]))<<i;

else re+=1LL\*sum[i]<<i;

}

return re;

}

void dfs(int dep,int lo,int qcnt){

if(qcnt<=0)return ;

if(qcnt==t[lo].ncnt){

pts.pb(lo);return;

}

if(dep==-1){

for(int i=0;i<31;i++){

sum[i]+=qcnt\*(cnt[lo][i]?1:0);

}

return;

}

if(sort\_val&(1<<dep))swap(t[lo].ls,t[lo].rs);

if(t[lo].ls){

int ls=t[lo].ls;

if(t[ls].ncnt<qcnt){pts.eb(ls);dfs(dep-1,t[lo].rs,qcnt-t[ls].ncnt);}

else dfs(dep-1,t[lo].ls,qcnt);

}

else dfs(dep-1,t[lo].rs,qcnt);

if(sort\_val&(1<<dep))swap(t[lo].ls,t[lo].rs);

}

inline ll get\_pre(int x){

memset(sum,0,sizeof(sum));

if(!x)return 0LL;

k=x;

pts.clear();

ll re=0;

if(x>t[rt].ncnt){

if(rt)

pts.pb(rt);

k=t[rt].ncnt;

re+=query\_pre(x-t[rt].ncnt);

}

else dfs(30,rt,x);

re+=query\_pts();

return re;

}

int main(){

read(n);

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

read(u);

bval[++pcnt]=u;

for(register int i=0;i<31;i++)

pre[pcnt][i]=pre[pcnt-1][i]+((u>>i)&1);

}

read(m);

while(m--){

read(opt);

if(opt==1){

int u;read(u);

bval[++pcnt]=(u^=tot\_val);

for(register int i=0;i<31;i++)

pre[pcnt][i]=pre[pcnt-1][i]+((u>>i)&1);

}

else if(opt==2){

int u,v;read(v);read(u);

printf("%lld\n",get\_pre(u)-get\_pre(v-1));

}

else if(opt==3){

int u;read(u);tot\_val^=u;

}

else if(opt==4){

sort\_val=tot\_val;

while(pcnt)

insert(30,bval[pcnt--],rt);

}

}

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;

}

**例1-wannafly-wintercamp-2019-Day5-C-查询区间最多k次某数除2后的最小和**

struct node{

int l,r,cnt;

ll sum;

}T[MAX\*100];

int tcnt,lim;

int rt[MAX];

void insert(int &k,int lo,int l,int r){

T[++tcnt]=T[k];k=tcnt;//可以避免初始化整个主席树结点池

++T[k].cnt;T[k].sum+=lo;

if(l==r)return;

int mid=(l+r)/2;

if(lo<=mid)insert(T[k].l,lo,l,mid);

else insert(T[k].r,lo,mid+1,r);

}

ll query(int rl,int rr,int &cnt,int l,int r){

if(!cnt||rl==rr)return 0LL;

if(T[rr].cnt-T[rl].cnt<=cnt){

cnt-=T[rr].cnt-T[rl].cnt;

return T[rr].sum-T[rl].sum;

}

if(l==r){

int ge=min(cnt,T[rr].cnt-T[rl].cnt);

cnt-=ge;

return 1LL\*ge\*l;

}

int cnt\_r=T[T[rr].r].cnt-T[T[rl].r].cnt;

int mid=(l+r)/2;

ll re=query(T[rl].r,T[rr].r,cnt,mid+1,r);

re+=query(T[rl].l,T[rr].l,cnt,l,mid);

return re;

}

int n,q;

ll pre[MAX];//前缀和

int ql[MAX\*5],qr[MAX\*5],qk[MAX\*5];

ll ans[MAX\*5];

int a[MAX];

int main(){

read(n);read(q);

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

read(a[i]),pre[i]=pre[i-1]+a[i];

for(int i=1;i<=q;i++)read(ql[i]),read(qr[i]),read(qk[i]),ans[i]=pre[qr[i]]-pre[ql[i]-1];

for(int i=30;i>=0;i--){

tcnt=0;lim=1<<(i+1);

int minlim=1<<i;

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

rt[j]=rt[j-1];

while((a[j]-a[j]/2)>=minlim)

insert(rt[j],a[j]-a[j]/2,1,lim),a[j]/=2;

}

for(int j=1;j<=q;j++)

if(qk[j])

ans[j]-=query(rt[ql[j]-1],rt[qr[j]],qk[j],1,lim);

}

for(int j=1;j<=q;j++)printf("%lld\n",ans[j]);

return 0;

}

## 并查集

**例1-wannafly-wintercamp2019-Day3-I**

int fa[MAX];

int val[MAX];

inline int getfa(int x){

if(fa[x]==x)return x;

int y=getfa(fa[x]);

if(fa[x]!=y)

muli(val[x],val[fa[x]]);

fa[x]=y;

return y;

}

int n,m,mi;

int v1\_3,v2\_3;

int main(){

read(n);read(m);

mi=powMM(3,n,MOD);

v1\_3=powMM(3,MOD-2,MOD);

v2\_3=mul(2,powMM(3,MOD-2,MOD));

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

while(m--){

int opt;

read(opt);

if(opt==1){

int x,y;

read(x);read(y);

muli(val[x],v2\_3);muli(val[y],v1\_3);

muli(val[y],powMM(val[x],MOD-2,MOD));

fa[y]=x;

}

else{

int x;read(x);

int rt=getfa(x);

int ans=mul(val[x],mi);

if(x!=rt)muli(ans,val[rt]);

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

}

}

return 0;

}

## splay

**0.普通splay模版**

int ch[N][2], par[N], val[N], cnt[N], size[N], ncnt, root;

bool chk(int x) {

return ch[par[x]][1] == x;

}

void pushup(int x) {

size[x] = size[ch[x][0]] + size[ch[x][1]] + cnt[x];

}

void rotate(int x) {

int y = par[x], z = par[y], k = chk(x), w = ch[x][k^1];

ch[y][k] = w; par[w] = y;

ch[z][chk(y)] = x; par[x] = z;

ch[x][k^1] = y; par[y] = x;

pushup(y); pushup(x);

}

void splay(int x, int goal = 0) {

while (par[x] != goal) {

int y = par[x], z = par[y];

if (z != goal) {

if (chk(x) == chk(y)) rotate(y);

else rotate(x);

}

rotate(x);

}

if (!goal) root = x;

}

void insert(int x) {

int cur = root, p = 0;

while (cur && val[cur] != x) {

p = cur;

cur = ch[cur][x > val[cur]];

}

if (cur) {

cnt[cur]++;

} else {

cur = ++ncnt;

if (p) ch[p][x > val[p]] = cur;

ch[cur][0] = ch[cur][1] = 0;

par[cur] = p; val[cur] = x;

cnt[cur] = size[cur] = 1;

}

splay(cur);

}

void find(int x) {

int cur = root;

while (ch[cur][x > val[cur]] && x != val[cur]) {

cur = ch[cur][x > val[cur]];

}

splay(cur);

}

int kth(int k) {

int cur = root;

while (1) {

if (ch[cur][0] && k <= size[ch[cur][0]]) {

cur = ch[cur][0];

} else if (k > size[ch[cur][0]] + cnt[cur]) {

k -= size[ch[cur][0]] + cnt[cur];

cur = ch[cur][1];

} else {

return cur;

}

}

}

int pre(int x) {

find(x);

if (val[root] < x) return root;

int cur = ch[root][0];

while (ch[cur][1]) cur = ch[cur][1];

return cur;

}

int succ(int x) {

find(x);

if (val[root] > x) return root;

int cur = ch[root][1];

while (ch[cur][0]) cur = ch[cur][0];

return cur;

}

void remove(int x) {

int last = pre(x), next = succ(x);

splay(last); splay(next, last);

int del = ch[next][0];

if (cnt[del] > 1) {

cnt[del]--;

splay(del);

}

else ch[next][0] = 0;

}

int n, op, x;

int main() {

scanf("%d", &n);

insert(0x3f3f3f3f);

insert(0xcfcfcfcf);

while (n--) {

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

switch (op) {

case 1: insert(x); break;

case 2: remove(x); break;

case 3: find(x); printf("%d\n", size[ch[root][0]]); break;

case 4: printf("%d\n", val[kth(x+1)]); break;

case 5: printf("%d\n", val[pre(x)]); break;

case 6: printf("%d\n", val[succ(x)]); break;

}

}

}

**例1-NOI2005-维修数列**

//以下splay先插入了两个-INF（一个在前一个在后） 所以下标实际（不算-INF）从1开始 但建线段树不包含两个INF

int size[MAX],sum[MAX],upd[MAX],rev[MAX],la[MAX],ra[MAX],gss[MAX];

int val[MAX],ch[MAX][2],par[MAX],ncnt,root;

queue<int>q;

//回收x子树（线段树中对应的子树？）

void recycle(int x){

if(ch[x][0])recycle(ch[x][0]);

if(ch[x][1])recycle(ch[x][1]);

q.push(x);

}

//使用垃圾回收机制的创建新结点

inline int newNode(int x){

int cur;

if(q.empty())cur=++ncnt;

else cur=q.front(),q.pop();

ch[cur][0]=ch[cur][1]=par[cur]=0;

val[cur]=sum[cur]=gss[cur]=x;

la[cur]=ra[cur]=max(0,x);

upd[cur]=rev[cur]=0;

size[cur]=1;

return cur;

}

//x在其fa的哪个方向上（0左 1右 儿子）

inline bool chk(int x){

return ch[par[x]][1]==x;

}

//线段树中的pushup

inline void pushup(int x){

int l=ch[x][0],r=ch[x][1];

size[x]=size[l]+size[r]+1;//??一个结点只有一个点？

sum[x]=sum[l]+sum[r]+val[x];

la[x]=max(la[l],sum[l]+val[x]+la[r]);

ra[x]=max(ra[r],sum[r]+val[x]+ra[l]);

gss[x]=max(ra[l]+val[x]+la[r],max(gss[l],gss[r]));

}

//单次旋转 将x旋转到其fa的位置

inline void rotate(int x){

int y=par[x],z=par[y],k=chk(x),w=ch[x][k^1];

ch[y][k]=w;par[w]=y;

ch[z][chk(y)]=x;par[x]=z;

ch[x][k^1]=y;par[y]=x;

pushup(y);pushup(x);

}

//下传lazy标记（这里是lazy和rev）

inline void pushdown(int x){

int l=ch[x][0],r=ch[x][1];

if(upd[x]){

upd[x]=rev[x]=0;

if(l){

upd[l]=1;val[l]=val[x];

sum[l]=val[x]\*size[l];

la[l]=ra[l]=max(sum[l],0);

gss[l]=val[x]<0?val[x]:sum[l];

}

if(r){

upd[r]=1;val[r]=val[x];

sum[r]=val[x]\*size[r];

la[r]=ra[r]=max(sum[r],0);

gss[r]=val[x]<0?val[x]:sum[r];

}

}

if(rev[x]){

rev[l]^=1;rev[r]^=1;rev[x]=0;

swap(la[l],ra[l]);swap(la[r],ra[r]);

swap(ch[l][0],ch[l][1]);

swap(ch[r][0],ch[r][1]);

}

}

//splay操作 将x结点移动到goal的子节点（默认goal是0表示将x移动到根）

inline void splay(int x,int goal=0){

while(par[x]!=goal){

int y=par[x],z=par[y];

if(z!=goal){

if(chk(x)==chk(y))rotate(y);

else rotate(x);

}

rotate(x);

}

if(!goal)root=x;

}

//建立二叉（类线段树用splay维护）

int build(int l,int r,int \*arr){

if(l>r)return 0;

int mid=(l+r)>>1,cur=newNode(arr[mid]);

if(l==r)return cur;

if((ch[cur][0]=build(l,mid-1,arr)))par[ch[cur][0]]=cur;

if((ch[cur][1]=build(mid+1,r,arr)))par[ch[cur][1]]=cur;

pushup(cur);

return cur;

}

//查询第k个

inline int kth(int k){

int cur=root;

while(1){

pushdown(cur);

if(ch[cur][0]&&k<=size[ch[cur][0]])

cur=ch[cur][0];

else if(k>size[ch[cur][0]]+1){

k-=size[ch[cur][0]]+1;cur=ch[cur][1];

}

else return cur;

}

}

//在x位置后插入y

inline void insert(int x,int y){

int u=kth(x+1),v=kth(x+2);

splay(u);splay(v,u);

ch[v][0]=y;par[y]=v;

pushup(v);pushup(u);

}

//查询数组x开始y个元素之间的和

inline int qsum(int x,int y){

int u=kth(x),v=kth(x+y+1);

splay(u);splay(v,u);

return sum[ch[v][0]];

}

//最大子段和

inline int qgss(){

return gss[root];

}

//删除数组x开始y个元素

inline void remove(int x,int y){

int u=kth(x),v=kth(x+y+1);

splay(u);splay(v,u);

recycle(ch[v][0]);

ch[v][0]=0;

pushup(v);pushup(u);

}

//翻转整个区间相当于翻转每个区间的两部分

inline void reverse(int x,int y){

int u=kth(x),v=kth(x+y+1);

splay(u);splay(v,u);

int w=ch[v][0];

if(!upd[w]){//若非整体更新

rev[w]^=1;

swap(ch[w][0],ch[w][1]);

swap(la[w],ra[w]);

pushup(v);pushup(u);

}

}

//将[x,y]区间更新为z

inline void update(int x,int y,int z){

int u=kth(x),v=kth(x+y+1);

splay(u);splay(v,u);

int w=ch[v][0];

upd[w]=1;val[w]=z;sum[w]=size[w]\*z;

la[w]=ra[w]=max(0,sum[w]);

gss[w]=z<0?z:sum[w];

pushup(v);pushup(u);

}

int n,m;

int arr[MAX];

char buf[35];

int main(){

read(n);read(m);

for(int i=2;i<=n+1;i++)read(arr[i]);

gss[0]=val[0]=0xcfcfcfcf;

arr[1]=arr[n+=2]=0xcfcfcfcf;

build(1,n,arr);root=1;

while(m--){

scanf("%s",buf);

int x,y,z;

switch((buf[2]+buf[1])^\*buf){

case 'G'^('E'+'T')://查询区间和

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

printf("%d\n", qsum(x, y));

break;

case 'M'^('A'+'X')://查询最大子段和

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

break;

case 'R'^('E'+'V')://reverse区间

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

reverse(x, y);

break;

case 'M'^('A'+'K')://更新区间

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

update(x, y, z);

break;

case 'D'^('E'+'L')://删除区间

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

remove(x, y);

break;

case 'I'^('N'+'S')://插入新区间（对应于一个子树）

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

memset(arr, 0, sizeof arr);

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

scanf("%d", arr+i);

}

insert(x, build(1, y, arr));

break;

}

}

return 0;

}

**例2-牛客2018第3场C**

int size[MAX],val[MAX],ch[MAX][2],par[MAX];

int ncnt,root;

queue<int>q;

void recycle(int x){

if(ch[x][0])recycle(ch[x][0]);

if(ch[x][1])recycle(ch[x][1]);

q.push(x);

}

inline int newNode(int x){

int cur;

if(q.empty())cur=++ncnt;

else cur=q.front(),q.pop();

ch[cur][0]=ch[cur][1]=par[cur]=0;

val[cur]=x;

size[cur]=1;

return cur;

}

inline bool chk(int x){

return ch[par[x]][1]==x;

}

inline void pushup(int x){

int l=ch[x][0],r=ch[x][1];

size[x]=size[l]+size[r]+1;

}

inline void rotate(int x){

int y=par[x],z=par[y],k=chk(x),w=ch[x][k^1];

ch[y][k]=w;par[w]=y;

ch[z][chk(y)]=x;par[x]=z;

ch[x][k^1]=y;par[y]=x;

pushup(y);pushup(x);

}

inline void splay(int x,int goal=0){

while(par[x]!=goal){

int y=par[x],z=par[y];

if(z!=goal){

if(chk(x)==chk(y))rotate(y);

else rotate(x);

}

rotate(x);

}

if(!goal)root=x;

}

int build(int l,int r,int \*arr){

if(l>r)return 0;

int mid=(l+r)>>1,cur=newNode(arr[mid]);

if(l==r)return cur;

if((ch[cur][0]=build(l,mid-1,arr)))par[ch[cur][0]]=cur;

if((ch[cur][1]=build(mid+1,r,arr)))par[ch[cur][1]]=cur;

pushup(cur);

return cur;

}

inline int kth(int k){

int cur=root;

while(1){

if(ch[cur][0]&&k<=size[ch[cur][0]])

cur=ch[cur][0];

else if(k>size[ch[cur][0]]+1)

k-=size[ch[cur][0]]+1,cur=ch[cur][1];

else return cur;

}

}

inline void insert(int x,int y){

int u=kth(x+1),v=kth(x+2);

splay(u);splay(v,u);

ch[v][0]=y;par[y]=v;

pushup(v);pushup(u);

}

inline int remove(int x,int y){

int u=kth(x),v=kth(x+y+1);

splay(u);splay(v,u);

int re=ch[v][0];

// recycle(ch[v][0]);

ch[v][0]=0;

pushup(v);pushup(u);

return re;

}

void dfs(int now){

if(ch[now][0])dfs(ch[now][0]);

if(val[now]>=1)

printf("%d ",val[now]);

if(ch[now][1])dfs(ch[now][1]);

}

int n,m;

int a[MAX],b[MAX];

int main(){

read(n);read(m);

for(int i=2;i<=n+1;i++)a[i]=i-1;

a[1]=a[n+=2]=0xff3f3f3f;

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

while(m--){

int x,y;

read(x);read(y);

if(x==1)continue;

int z=remove(1,x-1);

insert(y,z);

}

dfs(root);

return 0;

}

**例3-维护区间和、最值及区间翻转-2019MW-camp-Day4-K**

int size[MAX],rev[MAX],lazy[MAX];

int ch[MAX][2],par[MAX];

ll minv[MAX],maxv[MAX],val[MAX],sum[MAX];

int ncnt,root;

inline int newNode(int x){

int cur=++ncnt;

ch[cur][0]=ch[cur][1]=par[cur]=0;

val[cur]=sum[cur]=minv[cur]=maxv[cur]=x;

lazy[cur]=0;

size[cur]=1;

rev[cur]=0;

return cur;

}

inline bool chk(int x){

return ch[par[x]][1]==x;

}

void pushup(int x){

int l=ch[x][0],r=ch[x][1];

size[x]=size[l]+size[r]+1;

sum[x]=sum[l]+sum[r]+val[x];

minv[x]=maxv[x]=val[x];

if(l)minv[x]=min(minv[x],minv[l]),maxv[x]=max(maxv[x],maxv[l]);

if(r)minv[x]=min(minv[x],minv[r]),maxv[x]=max(maxv[x],maxv[r]);

}

inline void rotate(int x){

int y=par[x],z=par[y],k=chk(x),w=ch[x][k^1];

ch[y][k]=w;par[w]=y;

ch[z][chk(y)]=x;par[x]=z;

ch[x][k^1]=y;par[y]=x;

pushup(y);pushup(x);

}

inline void pushdown(int x){

int l=ch[x][0],r=ch[x][1];

if(lazy[x]){

if(l){

lazy[l]+=lazy[x];

minv[l]+=lazy[x];maxv[l]+=lazy[x];

val[l]+=lazy[x];

sum[l]+=lazy[x]\*size[l];

}

if(r){

lazy[r]+=lazy[x];

minv[r]+=lazy[x];maxv[r]+=lazy[x];

val[r]+=lazy[x];

sum[r]+=lazy[x]\*size[r];

}

lazy[x]=0;

}

if(rev[x]){

rev[l]^=1;rev[r]^=1;rev[x]=0;

swap(ch[l][0],ch[l][1]);

swap(ch[r][0],ch[r][1]);

}

}

inline void splay(int x,int goal=0){

while(par[x]!=goal){

int y=par[x],z=par[y];

if(z!=goal){

if(chk(x)==chk(y))rotate(y);

else rotate(x);

}

rotate(x);

}

if(!goal)root=x;

}

int build(int l,int r,int \*arr){

if(l>r)return 0;

int mid=(l+r)>>1,cur=newNode(arr[mid]);

if(l==r)return cur;

if((ch[cur][0]=build(l,mid-1,arr)))par[ch[cur][0]]=cur;

if((ch[cur][1]=build(mid+1,r,arr)))par[ch[cur][1]]=cur;

pushup(cur);

return cur;

}

inline int kth(int k){

int cur=root;

while(1){

pushdown(cur);

if(ch[cur][0]&&k<=size[ch[cur][0]])cur=ch[cur][0];

else if(k>size[ch[cur][0]]+1){

k-=size[ch[cur][0]]+1;cur=ch[cur][1];

}

else return cur;

}

}

inline void query(int x,int y){

int u=kth(x),v=kth(y+2);

splay(u);splay(v,u);

int l=ch[v][0];

printf("%lld %lld %lld\n",sum[l],minv[l],maxv[l]);

}

inline void reverse(int x,int y){

int u=kth(x),v=kth(y+2);

splay(u);splay(v,u);

int w=ch[v][0];

rev[w]^=1;

swap(ch[w][0],ch[w][1]);

pushup(v);pushup(u);

}

inline void update(int x,int y,int c){//Çø¼ä¼Óc

int u=kth(x),v=kth(y+2);

splay(u);splay(v,u);

int w=ch[v][0];

minv[w]+=c;maxv[w]+=c;val[w]+=c;lazy[w]+=c;

sum[w]+=1LL\*size[w]\*c;

pushup(v);pushup(u);

}

int n,m;

int a[MAX];

char opt[10];

int main(){

read(n);

for(int i=2;i<=n+1;i++)read(a[i]);

minv[0]=minv[n+2]=INF;

maxv[0]=maxv[n+2]=-INF;n+=2;

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

read(m);

while(m--){

int l,r,v;

scanf("%s%d%d",opt,&l,&r);

if(opt[0]=='c'){

read(v);

update(l,r,v);

}

else if(opt[0]=='r')reverse(l,r);

else query(l,r);

}

return 0;

}

## link-cut tree

**模版-例3-BZOJ2631-动态维护树结构及路径加乘查询**

使用LCT求LCA

第一个点expose（即access）

第二个点expose（即access）

最后一次把虚边变成实边的位置

考虑第一次先将第一个点access 则根到其路径为一条实边

第二次access时，是想使第二个点到根的路径变成实边，与原本的实路径一定有交

int n,m;

int val[MAX],sum[MAX],size[MAX],mval[MAX],aval[MAX],fa[MAX],rev[MAX];

int ch[MAX][2],stk[MAX];

void cal(int k,int mulv,int addv){//用mulv\addv更新k结点

if(!k)return;

val[k]=(val[k]\*mulv+addv)%MOD;

sum[k]=(sum[k]\*mulv+addv\*size[k])%MOD;

aval[k]=(aval[k]\*mulv+addv)%MOD;//要下传的加的数的标记

mval[k]=mval[k]\*mulv%MOD;//要下传的乘标记

}

bool isroot(int k){//是否是所在splay的根

return ch[fa[k]][0]!=k&&ch[fa[k]][1]!=k;

}

void pushup(int k){//由子节点更新父节点的pushup

int l=ch[k][0],r=ch[k][1];

sum[k]=(sum[l]+sum[r]+val[k])%MOD;

size[k]=(size[l]+size[r]+1)%MOD;

}

void pushdown(int k){//父节点根据其标记向下更新

int l=ch[k][0],r=ch[k][1];

if(rev[k]){//整个区间需要反转

rev[k]^=1;rev[l]^=1;rev[r]^=1;

swap(ch[k][0],ch[k][1]);

}

int tem\_mul=mval[k],tem\_add=aval[k];

mval[k]=1;aval[k]=0;

if(tem\_mul!=1||tem\_add!=0){

cal(l,tem\_mul,tem\_add);

cal(r,tem\_mul,tem\_add);

}

}

void rotate(int x){//旋转x至其fa的位置

int y=fa[x],z=fa[y],l,r;

l=(ch[y][1]==x);r=l^1;

if(!isroot(y))ch[z][ch[z][1]==y]=x;

fa[x]=z;fa[y]=x;fa[ch[x][r]]=y;

ch[y][l]=ch[x][r];ch[x][r]=y;

pushup(y);pushup(x);

}

void splay(int x){//旋转x至其在的splay的根

int top=0;stk[++top]=x;

for(int i=x;!isroot(i);i=fa[i])

stk[++top]=fa[i];//将要更新的路径入栈

while(top)pushdown(stk[top--]);//一路向下pushdown更新

while(!isroot(x)){

int y=fa[x],z=fa[y];

if(!isroot(y)){

if(ch[y][0]==x^ch[z][0]==y)rotate(x);

else rotate(y);

}

rotate(x);

}

}

void access(int x){//将x到根的路径的边变为实边

for(int i=0;x;i=x,x=fa[x]){

splay(x);ch[x][1]=i;pushup(x);//强行修改了其右儿子 需要pushup

}

}

void makeroot(int x){//将x设置为新的真·树根

access(x);splay(x);//此时x只有左儿子无右儿子（因为x为该路径的深度最大点）

rev[x]^=1;//反转x到原本的真·树根的路径

}

void split(int x,int y){//将y设置成真·树根 并将x放到辅助树的根 方便处理

makeroot(y);access(x);splay(x);

}

void link(int x,int y){//连接x\y这条边 （未在函数中判断是否合法）

makeroot(x);fa[x]=y;

}

void cut(int x,int y){//切断x/y这条边 同样未判断操作是否合法

makeroot(x);access(y);splay(y);//此时ch[y][0]就应该是x

ch[y][0]=fa[x]=0;

}

int main(){

read(n);read(m);

for(int i=1;i<=n;i++)val[i]=sum[i]=mval[i]=size[i]=1;

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

read(u);read(v);link(u,v);

}

char opt[10];

while(m--){

scanf("%s",opt);

int u,v;read(u);read(v);

if(opt[0]=='+'){//u到v的路径加c

int c;read(c);

split(u,v);//此时u只有左儿子无右儿子

//且u的左子树即为要更新的路径

cal(u,1,c);

}

else if(opt[0]=='-'){//切断u v 加新边u' v'

cut(u,v);read(u);read(v);

link(u,v);

}

else if(opt[0]=='\*'){//u到v的路径乘c

int c;read(c);

split(u,v);cal(u,c,0);

}

else if(opt[0]=='/'){//求u导v的路径的权值和

split(u,v);printf("%d\n",sum[u]);

}

}

return 0;

}

**例1-BZOJ2049-动态删边加边查询连通性（保证连通只有是树）**

int n,m;

int par[MAX],ch[MAX][2];

int stk[MAX];//pushdown的栈

bool rev[MAX];

inline bool isroot(int x){//是否是该splay的根

return ch[par[x]][0]!=x&&ch[par[x]][1]!=x;

}

void pushdown(int k){

int l=ch[k][0],r=ch[k][1];

if(rev[k]){

rev[k]^=1;rev[l]^=1;rev[r]^=1;

swap(ch[k][0],ch[k][1]);

}

}

void rotate(int x){

int y=par[x],z=par[y],l,r;

if(ch[y][0]==x)l=0;

else l=1;

r=l^1;

if(!isroot(y)){

if(ch[z][0]==y)ch[z][0]=x;

else ch[z][1]=x;

}

par[x]=z;par[y]=x;par[ch[x][r]]=y;

ch[y][l]=ch[x][r];ch[x][r]=y;

}

void splay(int x){

int top=0;stk[++top]=x;

for(int i=x;!isroot(i);i=par[i])

stk[++top]=par[i];

for(int i=top;i;i--)pushdown(stk[i]);

while(!isroot(x)){

int y=par[x],z=par[y];

if(!isroot(y)){

if(ch[y][0]==x^ch[z][0]==y)

rotate(x);

else rotate(y);

}

rotate(x);

}

}

void access(int x){

int t=0;

while(x){

splay(x);

ch[x][1]=t;

t=x;x=par[x];

}

}

//将x设为根后 x到原本根的路径的点的深度发生了反转

//这里的rever函数实际上是重设真实树根

void rever(int x){

access(x);

splay(x);

assert(ch[x][1]==0);

rev[x]^=1;

}

void link(int x,int y){

rever(x);par[x]=y;splay(x);

}

void cut(int x,int y){

rever(x);access(y);splay(y);

ch[y][0]=par[x]=0;

}

int find(int x){

access(x);splay(x);

int y=x;while(ch[y][0])y=ch[y][0];

return y;

}

char opt[10];

int main(){

read(n);read(m);

while(m--){

scanf("%s",opt);

int u,v;read(u);read(v);

if(opt[0]=='C')link(u,v);

else if(opt[0]=='D')cut(u,v);

else{

if(find(u)==find(v))puts("Yes");

else puts("No");

}

}

return 0;

}

**例2-BZOJ2002-维护splay子树大小**

int n,m;

//par、fa分别是辅助树和实际树的父亲结点

int next[MAX],fa[MAX],ch[MAX][2],size[MAX],stk[MAX];

bool rev[MAX];

inline bool isroot(int k){

return ch[fa[k]][0]!=k&&ch[fa[k]][1]!=k;

}

inline void pushup(int k){

size[k]=size[ch[k][0]]+size[ch[k][1]]+1;

}

void pushdown(int k){

int l=ch[k][0],r=ch[k][1];

if(rev[k]){

rev[k]^=1;rev[l]^=1;rev[r]^=1;

swap(ch[k][0],ch[k][1]);

}

}

void rotate(int x){

int y=fa[x],z=fa[y],l,r;

if(ch[y][0]==x)l=0;else l=1;r=l^1;

if(!isroot(y)){

if(ch[z][0]==y)ch[z][0]=x;

else ch[z][1]=x;

}

fa[x]=z;fa[y]=x;fa[ch[x][r]]=y;

ch[y][l]=ch[x][r];ch[x][r]=y;

pushup(y);pushup(x);

}

void splay(int x){

int top=0;stk[++top]=x;

for(int i=x;!isroot(i);i=fa[i])stk[++top]=fa[i];

for(int i=top;i;i--)pushdown(stk[i]);

while(!isroot(x)){

int y=fa[x],z=fa[y];

if(!isroot(y)){

if(ch[y][0]==x^ch[z][0]==y)rotate(x);

else rotate(y);

}

rotate(x);

}

}

void access(int x){

int t=0;

while(x){

splay(x);ch[x][1]=t;

t=x;x=fa[x];

}

}

void rever(int x){

access(x);splay(x);rev[x]^=1;

}

void link(int x,int y){

rever(x);fa[x]=y;splay(x);

}

void cut(int x,int y){

rever(x);access(y);splay(y);ch[y][0]=fa[x]=0;

}

int main(){

read(n);

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

int x;read(x);

fa[i]=x+i;size[i]=1;

if(fa[i]>n+1)fa[i]=n+1;

next[i]=fa[i];

}

size[n+1]=1;

read(m);

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

int opt;read(opt);

if(opt==1){

rever(n+1);int u;read(u);++u;

access(u);splay(u);printf("%d\n",size[ch[u][0]]);

}

else{

int x,y;read(x);read(y);++x;

int tar=min(n+1,x+y);

cut(x,next[x]);link(x,tar);next[x]=tar;

}

}

return 0;

}

**例4-UOJ207-询问路径集合是否都经过某边**

int ID,n,m,father[MAXN],tag[MAXN],tr[MAXN][2],top,stack[MAXN],cnt;

int sum[MAXN],val[MAXN],S;

//sum表示子树内所有的点的异或和,val表示x加上虚儿子子树的异或和

struct node{ int x,y,z; }e[MAXN];

inline bool isroot(int x){

return (tr[father[x]][0]!=x) && (tr[father[x]][1]!=x);

}

inline void update(int x){

sum[x]=sum[tr[x][0]]^sum[tr[x][1]]^val[x];

}

inline int getint(){

int w=0,q=0; char c=getchar(); while((c<'0'||c>'9') && c!='-') c=getchar();

if(c=='-') q=1,c=getchar(); while (c>='0'&&c<='9') w=w\*10+c-'0',c=getchar(); return q?-w:w;

}

inline void pushdown(int x){

if(tag[x]==0) return ;

int l=tr[x][0],r=tr[x][1];

tag[l]^=1; tag[r]^=1; swap(tr[x][0],tr[x][1]);

tag[x]=0;

}

inline void rotate(int x){

int y=father[x],z=father[y];

int l,r; l=(tr[y][1]==x); r=l^1;

if(!isroot(y)) tr[z][(tr[z][1]==y)]=x;

father[x]=z; father[y]=x;

tr[y][l]=tr[x][r]; father[tr[x][r]]=y; tr[x][r]=y;

update(y); update(x);

}

inline void splay(int x){

top=0; stack[++top]=x;

for(int i=x;!isroot(i);i=father[i]) stack[++top]=father[i];

for(int i=top;i>=1;i--) pushdown(stack[i])/\*!!!不是top啊!!!\*/;

int y,z;

while(!isroot(x)) {

y=father[x]; z=father[y];

if(!isroot(y)) {

if((tr[z][0]==y) ^ (tr[y][0]==x)) rotate(x);

else rotate(y);

}

rotate(x);

}

}

inline void access(int x){

int last=0;

while(x){

splay(x);

val[x]^=sum[tr[x][1]];//把原来的右儿子加入虚儿子的统计中

//而对于sum数组即整个子树没有变化，所以无需考虑

val[x]^=sum[last];//把原来是虚儿子的去掉

tr[x][1]=last;//!!!

update(x);//!!!顺序

last=x; x=father[x];

}

}

inline void move\_to\_root(int x){

access(x);

splay(x);

tag[x]^=1;

}

inline void link(int x,int y){

move\_to\_root(x);

move\_to\_root(y);//!!!必须要把y也变成根，不然修改了y之后修改不到y的祖先，把y变成根节点就可以只修改自己了

father[x]=y; val[y]^=sum[x];//加入x的贡献

update(y);

}

inline void cut(int x,int y){

move\_to\_root(x);

access(y);

splay(y);assert(tr[y][0]==x);

tr[y][0]=father[x]=0;

update(y);//此处update的时候也可以消除x的贡献

}

inline void modify(int x,int vaL){

access(x); splay(x);

val[x]^=vaL; sum[x]^=vaL;//只要修改自己就好了

}

inline bool query(int x,int y){

move\_to\_root(x); access(y);

return val[y]==S?1:0;

}

inline void work(){

srand(time(NULL));

ID=getint(); n=getint(); m=getint(); int type,x,y;

for(int i=1;i<n;i++) x=getint(),y=getint(),link(x,y);

while(m--) {

type=getint();

if(type==1) {

x=getint(); y=getint(); cut(x,y);

x=getint(); y=getint(); link(x,y);

}

else if(type==2) {

x=getint(); y=getint();

e[++cnt].x=x;/\*!!!\*/ e[cnt].y=y; e[cnt].z=(rand()%10000+1)\*(rand()%100000+1);

modify(x,e[cnt].z); modify(y,e[cnt].z); S^=e[cnt].z;

}

else if(type==3) {

x=getint(); S^=e[x].z;

modify(e[x].x,e[x].z); modify(e[x].y,e[x].z);

}

else {

x=getint(); y=getint();

if(query(x,y)) puts("YES");

else puts("NO");

}

}

}

**例5-树重新连边动态求LCA-2019MWcamp-Day4-A**

int n,m;

int ch[MAX][2],fa[MAX],rev[MAX],stk[MAX];

bool isroot(int k){

return ch[fa[k]][0]!=k&&ch[fa[k]][1]!=k;

}

void pushdown(int k){

int l=ch[k][0],r=ch[k][1];

if(rev[k]){

rev[k]=0;rev[l]^=1;rev[r]^=1;

swap(ch[k][0],ch[k][1]);

}

}

bool chk(int k){

return ch[fa[k]][1]==k;

}

void rotate(int x){

int y=fa[x],z=fa[y],l,r;

l=chk(x);r=l^1;

if(!isroot(y))ch[z][chk(y)]=x;

fa[x]=z;fa[y]=x;fa[ch[x][r]]=y;

ch[y][l]=ch[x][r];ch[x][r]=y;

}

void splay(int x){

int top=0;stk[++top]=x;

for(int i=x;!isroot(i);i=fa[i])

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

while(top)pushdown(stk[top--]);

while(!isroot(x)){

int y=fa[x],z=fa[y];

if(!isroot(y)){

if(ch[y][0]==x^ch[z][0]==y)rotate(x);

else rotate(y);

}

rotate(x);

}

}

int access(int x){

int re=x;

for(int i=0;x;i=x,x=fa[x]){

splay(x);ch[x][1]=i;

re=x;

}

return re;

}

void makeroot(int x){

access(x);splay(x);

rev[x]^=1;

}

void link(int x,int y){//将x连接到y的下面

makeroot(x);

fa[x]=y;

}

void cut(int x,int y){

makeroot(x);access(y);splay(y);

ch[y][0]=fa[x]=0;

}

int lca(int x,int y){

makeroot(1);

access(x);

return access(y);

}

int par[MAX];

int main(){

read(n);

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

read(par[i]);link(i,par[i]);

}

read(m);

while(m--){

char opt[10];

scanf("%s",opt);

int u,v;

read(u);read(v);

if(opt[0]=='C'){

cut(u,par[u]);

link(u,par[u]=v);

}

else{

printf("%d\n",lca(u,v));

}

}

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;

}

## 二分图

**例1-wannafly winter camp 2019 Day8 B**

**题意**：

给定一个n个点的一般图，m条边（n<=10000，m<=20000），其中每条边的权值为min(u,v)。一个一般图的匹配对应一个边集，求mex(匹配对应边集)的最大值。

int n,m,ecnt,cnt,ans,now;

bool used[MAX];

int head[MAX],match[MAX];

struct node{

int nxt,to;

node(){}

node(int \_nxt,int \_to):nxt(\_nxt),to(\_to){}

}edg[MAX<<1];

void add\_edge(int u,int v){

edg[++ecnt]=node(head[u],v);

head[u]=ecnt;

}

vector<int>G[MAX];//完整的原图

bool dfs(int v){//找增广路

used[v]=1;

for(int u=head[v];u;u=edg[u].nxt){

int to=edg[u].to;

if(to<=now)continue;

int w=match[to];

if(!w||(!used[w]&&dfs(w))){

match[v]=to;

match[to]=v;

return true;

}

}

return false;

}

int main(){

read(n);read(m);

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

read(u);read(v);

if(u>v)swap(u,v);

++u;++v;

G[u].eb(v);

}

for(int i=1;i<=n;i++){//枚举左侧的上界 左边点的个数

int who=match[i];

now=i;

head[i]=0;

if(who){

match[who]=0;match[i]=0;

--cnt;

memset(used,0,sizeof(used));

if(dfs(who))++cnt;

}

for(int u:G[i]){

add\_edge(i,u);add\_edge(u,i);

}

memset(used,0,sizeof(used));

if(dfs(i))++cnt;

if(cnt!=i){

ans=i-1;

break;

}

}

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

return 0;

}

## 三元环、四元环

**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;

}

**2019MW camp Day5 F**

**题意：给n个点m条边的图，求路径(x1,x2,x3,x4,x5)数：使得 x1 x2 x3 x4 x5互不相同。**

**做法：**

**Dp[i][j]表示从某个起点走i步后到j的方案数（相邻两步走的边不一样，即不能走一步之后往回走） 初始边界 dp[0][j]=1**

**转移 dp[i][j]=∑（与j相邻的所有x） dp[i-1][x] -dp[i-2][j]\* (max（0,i-2)?deg[j]-1：deg[j]） 其中deg[x]表示x的度数**

**（上述与0取max是因为可能有负数 i-2>0时 deg要减1 是因为对于已经走过边才到的点 就不能往回走了）**

vector<int>edg[MAX],hnxt[MAX];//原图 与rank比其大的点连的情况

int128 qua[MAX],tri[MAX],trisum[MAX];//每个点在四元环、三元环内的情况个数

bool vi[3][MAX];//走i步能否到j

int deg[MAX],rank[MAX],id[MAX];

int128 cnt[MAX];

int128 dp[MAX][5];//从i走j步的方案数（相邻两次方向不同）

set<int>tem;

int n,m;

bool cmp(int x,int y){

return deg[x]<deg[y];

}

inline void print(int128 x){

if(x==0){printf("0\n");return;}

int now=(int)(x%10);

x/=10;

if(x)print(x);

printf("%d",now);

}

int main(){

read(n);read(m);

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

read(u);read(v);

edg[u].eb(v);edg[v].eb(u);

++deg[u];++deg[v];

}

for(int i=1;i<=n;i++)dp[i][1]=deg[i],dp[i][0]=1;

for(int i=2;i<=4;i++){//步数

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

for(int u:edg[j])

dp[j][i]+=dp[u][i-1];

dp[j][i]-=dp[j][i-2]\*max(0,(i-2?deg[j]-1:deg[j]));//相邻两步不同

}

}

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

sort(id+1,id+1+n,cmp);//按deg由小到大排序

for(int i=1;i<=n;i++)rank[id[i]]=i;//deg大的rank大

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

for(int u:edg[i])

if(rank[i]<rank[u])hnxt[i].eb(u);//只保存deg比其大（rank高）的

}

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

for(int u:edg[i])vi[1][u]=1;

for(int u:edg[i]){

for(int v:hnxt[u]){

if(rank[v]>rank[i])

++cnt[v];

}

}

tem.clear();

for(int u:edg[i]){

for(int v:hnxt[u]){

if(rank[v]<=rank[i])continue;

qua[u]+=cnt[v]-1;

tem.insert(v);

if(vi[1][v])++tri[u],++tri[v],++tri[i];

}

}

for(int u:tem){

qua[i]+=cnt[u]\*(cnt[u]-1)/2;

qua[u]+=cnt[u]\*(cnt[u]-1)/2;

}

for(int v:tem)cnt[v]=0;

tem.clear();

for(int u:edg[i])vi[1][u]=0;

}

for(int i=1;i<=n;i++)tri[i]/=2;

for(int i=1;i<=n;i++)trisum[i]=-2\*tri[i];

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

for(int u:edg[i])

trisum[i]+=tri[u];

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

int128 ans=dp[i][4]-2\*(deg[i]-1)\*tri[i]-2\*trisum[i]-2\*qua[i];

print(ans);

printf("\n");

}

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]);

}

## 强连通分量

codeforces 1137C

题意：

n个城市，每个城市有一个博物馆，给定d天每个博物馆的是否开放（一周d天）。一个人第1天在第1个城市开始出发，每天走到一个相邻的城市，无限长的时间下，这个人最多能够看多少博物馆。

用(城市，周几）描述状态，建图进行连边。tarjan缩点，统计每个tarjan上的不同博物馆的个数，如果不同scc上不存在相同的博物馆，则就直接变成了一个DAG上dp的问题。可以证明，对一个城市x，如果(x,t)可以通过该图中的一些路径到达(x,t')则(x,t')也一定可以到达(x,t) 。那么就证明了不同SCC，如何一个是另一个能到达的，二者上一定不存在相同的博物馆。在实际tarjan时，只需要从(1,0) （第1个城市，模d下第0天）开始DFS即可。并且在DFS过程中，出栈统计强连通分量时，就可以直接来DP。（比较巧妙的trick）

int pre[MAX][52],lowlink[MAX][52],sccno[MAX][52],dfn,scc\_cnt;

pii S[MAX\*52];

int dp[MAX\*52];

bool ava[MAX][52],have[MAX];

int n,m,d,ecnt;

struct node{

int nxt,to;

}edg[MAX<<1];

int head[MAX];

int top;

vector<int>tm;

bool vi[MAX][55];

void dfs(int u,int day){

pre[u][day]=lowlink[u][day]=++dfn;

S[++top]=mp(u,day);assert(top<51\*MAX);

vi[u][day]=1;

int nxt\_day=(day+1)%d;

for(int i=head[u];i;i=edg[i].nxt){

int v=edg[i].to;

if(!pre[v][nxt\_day]){

dfs(v,nxt\_day);

lowlink[u][day]=min(lowlink[u][day],lowlink[v][nxt\_day]);

}

else if(vi[v][nxt\_day])lowlink[u][day]=min(lowlink[u][day],pre[v][nxt\_day]);

}

if(lowlink[u][day]==pre[u][day]){

++scc\_cnt;

int mor=0,nowans=0;

tm.clear();

for(;;){

int x=S[top].first,y=S[top].second;--top;

int nxt=(y+1)%d;

for(int i=head[x];i;i=edg[i].nxt){

int v=edg[i].to;

mor=max(mor,dp[sccno[v][nxt]]);

}

if(ava[x][y]){

nowans+=1-have[x];

if(!have[x])tm.eb(x);have[x]=1;

}

vi[x][y]=0;

sccno[x][y]=scc\_cnt;

if(x==u&&day==y)break;

}

for(int u:tm)have[u]=0;

tm.clear();

dp[scc\_cnt]=nowans+mor;

}

}

inline void add\_edge(int u,int v){

edg[++ecnt].nxt=head[u];

edg[ecnt].to=v;

head[u]=ecnt;

}

char a[55];

bool used[MAX];

int ans;

int main(){

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

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

scanf("%d%d",&u,&v);add\_edge(u,v);

}

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

scanf("%s",a);

for(int j=0;j<d;j++)ava[i][j]=a[j]-'0';

}

dfs(1,0);

printf("%d\n",dp[sccno[1][0]]);

return 0;

}

## 树的同构

**例1-BZOJ4337-求树的同构**

**题意**：给m个树，依次输出每个树在所有树中与其同构的树的编号的最小值。

（n<=50 m<=50）

做法：

对于一棵无根树，它的重心个数不超过2。

枚举每个重心，以重心为根求出这棵有根树的最小表示，然后取字典序最大的即可。

对于有根树的最小表示，可以看成括号序列，每次把子树的括号序列按字典序排序后依次串连起来即可。

int n,t,mx,ecnt;

int h[MAX],nxt[MAX],ts[MAX],siz[MAX],dp[MAX];

string qs[MAX],q[MAX],val[MAX];

/\*

加边

\*/

void add\_edge(int u,int v){

nxt[++ecnt]=h[u];ts[ecnt]=v;

h[u]=ecnt;

}

/\*

找树的重心

\*/

void findrt(int now,int fa){

siz[now]=1;dp[now]=0;

for(int i=h[now];i;i=nxt[i]){

int to=ts[i];

if(to!=fa){

findrt(to,now);

siz[now]+=siz[to];

if(siz[to]>dp[now])dp[now]=siz[to];

}

}

if(n-siz[now]>dp[now])dp[now]=n-siz[now];

if(dp[now]<mx)mx=dp[now];

}

/\*

从重心开始dfs得出整个树的括号表示 （即单个叶子节点用一对括号表示）

其中每个结点的一对括号里面各子树的顺序为由字典序小到大

\*/

void dfs(int now,int fa){

qs[now]="(";

for(int i=h[now];i;i=nxt[i]){

int to=ts[i];

if(to!=fa)dfs(to,now);

}

int lo=0;

for(int i=h[now];i;i=nxt[i]){

int to=ts[i];

if(to!=fa)q[lo++]=qs[to];

}

if(lo>1)sort(q,q+lo);

for(int i=0;i<lo;i++)qs[now]+=q[i];

qs[now]+=")";

}

/\*

读入并hash一棵树

\*/

string solve(){

string re="";

read(n);

mx=n;for(int i=1;i<=n;i++)h[i]=0;

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

int fa;read(fa);

if(fa)add\_edge(i,fa),add\_edge(fa,i);

}

findrt(1,0);

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

if(dp[i]==mx){

dfs(i,0);

if(qs[i]>re)re=qs[i];

}

}

return re;

}

int main(){

read(t);

for(int i=1;i<=t;i++)val[i]=solve();

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

{

int j,k;

for(j=k=i;j;j--)

if(val[j]==val[i])k=j;

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

}

return 0;

}

# 字符串

## KMP

**性质**：如果len%（len-next[len-1]）==0，则字符串中必存在最小循环节，且循环次数即为len/（len-next[len-1]）;

**证明**：在前len个字符组成的字符串，存在最小循环节k，那么next[len-1]=len-k;（为什么呐？因为next数组的定义就是最大前后缀相同的子串的长度，len的总长度减去最小循环节，比如有3个循环节，减去一个剩下两个，就是最大循环节）那么循环次数就是len/（len-next[len-1]）;因为len-next[len-1]=k;所以得出公式；

int f[MAX];

string x;

void getf(string x,int m)//m为串的长度 下标从1开始 最大为m

{

f[0]=f[1]=0;

for(int i=2,j=0;i<=m;i++)

{

while(j&&x[j+1]!=x[i])

j=f[j];

if(x[j+1]==x[i])

j++;

f[i]=j;

}

}

bool kmp(string x,string y)//在y中匹配x（即y比x长）

{

getf(x,x.size());

for(int i=1,j=0;i<=y.size();i++)

{

while(j&&x[j+1]!=y[i])

j=f[j];

if(x[j+1]==y[i])

j++;

if(j>=x.size())

return true;

}

return false;

}

## 最小表示法

**最小表示法**是求与某个字符串循环同构的所有字符串中，字典序最小的串是哪个。

设i、j是两个“怀疑是最小的位置”，比如说如果你比较到了jdrakioi的两个i，你目前还不知道从哪个i开始的字符串是最小的。

设k表示，从i往后数和从j往后数，有多少是相同的。

开始时先设i=0，j=1，k=0。

每次都对i+k、j+k进行一次比较。

比较完i+k和j+k，如果两个字符相等，那么显然k++。

如果不相等，那么哪边比较大，哪边就肯定不是最小的了，同时把k重置为0。

如果出现了i、j重合的情况，把j往后移动一位。

最后输出i、j较小的那个就好了。

本质上还是利用比较得到的大小关系的性质，来加速比较n个长度为n的字符串。使之达到O（n）的复杂度

int n,i,t,a[N<<1];

int minrep(){

int i=0,j=1,k=0,t;

while(i<n&&j<&&k<n){

if(t=a[(i+k)%n]-a[(j+k)%n]){

if(t>0)i+=k+1;

else j+=k+1;

if(i==j)++j;

k=0;

}

else k++;

}

return i<j?i:j;

}

int main(){

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

for(t=minrep(),i=0;i<ni;++)printf(i<n-1?"%d ":"%d",a[i+t]);

}

## AC自动机

**例1-HDU2825-AC自动机上DP**

**题意：**

给 m 个单词构成的集合，统计所有长度为 n 的串中，包含至少 k 个单词的方案数。

**做法：**

dp[i][j][k]表示在AC自动机上走了i步，当前在j点，已包含串的状态为k的方案数。

const int SIGMA\_SIZE=26;

const int MAXNODE=110;

struct AhoCorasickAutomata

{

int ch[MAXNODE][SIGMA\_SIZE];//Trie树

int f[MAXNODE];//fail函数

int val[MAXNODE];//每个字符串的结尾结点都有非0的val

int num;//trie树编号（亦为包含根结点的当前size）

//初始化

void init()

{

num=1;

memset(ch[0],-1,sizeof(ch[0]));

}

//返回字符对应编号

int idx(char c)

{

return c-'a';

}

//插入权值为v的字符串

void insert(char \*s,int v)

{

int u=0,n=strlen(s);

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

{

int c=idx(s[i]);

if(ch[u][c]==-1)

{

memset(ch[num],-1,sizeof(ch[num]));

val[num]=0;

ch[u][c]=num++;

}

u=ch[u][c];

}

val[u]|=v;

}

//计算fail函数

void getFail()

{

queue <int> que;

f[0]=0;

for(int c=0;c<SIGMA\_SIZE;c++)

{

int u=ch[0][c];

if(u!=-1)

{

f[u]=0;que.push(u);

}

else ch[0][c]=0;

}

while(!que.empty())

{

int r=que.front();que.pop();

if(val[f[r]])

val[r]|=val[f[r]];

for(int c=0;c<SIGMA\_SIZE;c++)

{

int u=ch[r][c];

if(u==-1){ch[r][c]=ch[f[r]][c];continue;}

que.push(u);

int v=f[r];

// while(v&&ch[v][c]==-1)//类似kmp的过程

// v=f[v];

f[u]=ch[v][c];

}

}

}

}AC;

int n,m,k;

char tem[20];

ll dp[26][105][1024];

ll num[1024],an;

int main()

{

num[0]=0;

for(int i=1;i<1024;i++)num[i]=num[i>>1]+(i&1);

while(scanf("%d%d%d",&n,&m,&k)&&n)

{

int tot=(1<<m);

AC.init();

an=0;

for(int i=1;i<=m;i++){scanf("%s",tem);AC.insert(tem,1<<(i-1));}

AC.getFail();

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

for(int j=0;j<=AC.num;j++)

for(int s=0;s<=tot;s++)dp[i][j][s]=0;

dp[0][0][0]=1;

for(int i=0;i<n;i++)//步数

for(int j=0;j<AC.num;j++)//当前位置

for(int s=0;s<tot;s++)//当前状态

{

if(!dp[i][j][s])continue;//无当前状态

for(int k=0;k<26;k++)

{

dp[i+1][AC.ch[j][k]][s|AC.val[AC.ch[j][k]]]+=dp[i][j][s];

dp[i+1][AC.ch[j][k]][s|AC.val[AC.ch[j][k]]]%=MOD;

}

}

for(int s=0;s<tot;s++)

{

if(num[s]<k)continue;

for(int i=0;i<AC.num;i++)

an=(an+dp[n][i][s])%MOD;

}

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

}

return 0;

}

**例2-ZOJ3228-AC自动机上DP**

/\*

ZOJ 3228

题目大意：首先给你一下母串，长度不超过10^5，然后有 N（10^5） 次查询，

每次查询有两种命令，0或者1，然后加一个子串，询问母串里面有多少个子串，

0表示可以重复，1表示不可以重复。

\*/

int cnt[MAX][2];

int las[MAX];

const int SIGMA\_SIZE=26;

const int MAXNODE=610000;

struct AhoCorasickAutomata

{

int ch[MAXNODE][SIGMA\_SIZE];//Trie树

int f[MAXNODE];//fail函数

int val[MAXNODE];//每个字符串的结尾结点都有非0的val

int last[MAXNODE];//输出链表的下一个单词结点

int dep[MAXNODE];

int num;//trie树编号（亦为包含根结点的当前size）

//初始化

void init()

{

num=1;

memset(ch[0],-1,sizeof(ch[0]));

}

//返回字符对应编号

int idx(char c)

{

return c-'a';

}

//插入权值为v的字符串

int insert(char \*s,int& v)

{

int u=0,n=strlen(s);

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

{

int c=idx(s[i]);

if(ch[u][c]==-1)

{

memset(ch[num],-1,sizeof(ch[num]));

val[num]=0;

dep[num]=dep[u]+1;

ch[u][c]=num++;

}

u=ch[u][c];

}

if(!val[u])

val[u]=++v;

return val[u];

}

//递归打印以结点j结尾的所有字符串

void print(int j,int pos)

{

if(j)

{

cnt[val[j]][0]++;

if(las[val[j]]==-1||las[val[j]]+dep[j]<=pos)

{

las[val[j]]=pos;

cnt[val[j]][1]++;

}

print(last[j],pos);

}

}

//计算fail函数

void getFail()

{

queue <int> que;

f[0]=0;

for(int c=0;c<SIGMA\_SIZE;c++)

{

int u=ch[0][c];

if(u!=-1)

{

f[u]=0;que.push(u);last[u]=0;

}

else

ch[0][c]=0;

}

while(!que.empty())

{

int r=que.front();que.pop();

// if(val[f[r]]&&)val[r]

for(int c=0;c<SIGMA\_SIZE;c++)

{

int u=ch[r][c];

// if(u==-1)continue;

if(u==-1){ch[r][c]=ch[f[r]][c];continue;}

que.push(u);

int v=f[r];

while(v&&!ch[v][c])//类似kmp的过程

v=f[v];

f[u]=ch[v][c];

last[u]=val[f[u]]?f[u]:last[f[u]];

}

}

}

//在T中中模板

int find(char \*T)

{

int n=strlen(T);

int u=0;//当前结点编号,初始为根结点

for(int i=0;i<n;i++)//文本串当前指针

{

int c=idx(T[i]);

// while(u&&!ch[u][c])u=f[u];//顺着失配指针走，直到可以匹配或回到根节点

u=ch[u][c];

if(val[u])

print(u,i);

else if(last[u])//明白了 如果当前恰为某串末尾 则直接print从其开始 不然从当前后缀作为某串的位置开始print

print(last[u],i);

}

}

}AC;

char a[MAX];

int n,num;

int kind[MAX],id[MAX],Case;

char tem[30];

int main()

{

while(~scanf("%s",a))

{

AC.init();

num=0;

scanf("%d",&n);

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

{

scanf("%d",&kind[i]);

scanf("%s",tem);

id[i]=AC.insert(tem,num);

}

AC.getFail();

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

las[i]=-1;

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

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

AC.find(a);

printf("Case %d\n",++Case);

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

printf("%d\n",cnt[id[i]][kind[i]]);

printf("\n");

}

}

**例3-BZOJ3881**

const int SIGMA\_SIZ=26;

const int MAXNODE=2e6+5;

int pos[MAX];

struct AhoCorasickAutomata{

int ch[MAXNODE][SIGMA\_SIZ];

int f[MAXNODE];

int num=0;

void init(){

num=0;

memset(ch[0],-1,sizeof(ch[0]));

}

void insert(char \*s,int i){

int n=strlen(s),u=0;

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

int to=s[i]-'a';

if(ch[u][to]==-1){

ch[u][to]=++num;

memset(ch[num],-1,sizeof(ch[num]));

}

u=ch[u][to];

}

pos[i]=u;

}

void getFail(){

queue<int>que;

for(int i=0;i<SIGMA\_SIZ;i++){

if(ch[0][i]==-1)ch[0][i]=0;

else que.push(ch[0][i]),f[ch[0][i]]=0;

}

while(!que.empty()){

int r=que.front();que.pop();

for(int c=0;c<SIGMA\_SIZ;c++){

int u=ch[r][c];

if(u==-1){

ch[r][c]=ch[f[r]][c];continue;

}

que.push(u);

f[u]=ch[f[r]][c];

}

}

}

}ac;

int ecnt,tcnt;

int h[MAX],nxt[MAX],to[MAX],fa[MAX][20],LOG[MAX],st[MAX],en[MAX],dep[MAX];

int va[MAX];

void add\_edge(int u,int v){

to[++ecnt]=v;nxt[ecnt]=h[u];h[u]=ecnt;

}

void dfs(int x){

st[x]=++tcnt;

for(int i=1;i<=LOG[dep[x]];i++)fa[x][i]=fa[fa[x][i-1]][i-1];

for(int i=h[x];i;i=nxt[i]){

int ts=to[i];

fa[ts][0]=x;dep[ts]=dep[x]+1;dfs(ts);

}

en[x]=tcnt;

}

inline int lca(int u,int v){

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

int dis=dep[u]-dep[v];

for(int i=LOG[dis];i>=0;i--)

if(dis&(1<<i))u=fa[u][i];

if(u==v)return u;

for(int i=LOG[dep[u]];i>=0;i--)

if(dep[u]>=(1<<i)&&fa[u][i]!=fa[v][i])u=fa[u][i],v=fa[v][i];

return fa[u][0];

}

int n,lim,m,opt;

char a[MAX];

inline void add(int x,int val){

for(int i=x;i<=lim;i+=i&(-i))

{

va[i]+=val;

}

}

inline int query(int x){

int re=0;

for(int i=x;i;i-=i&(-i))re+=va[i];

return re;

}

bool cmp(int x,int y){

return st[x]<st[y];

}

int tem[MAX],cnt;

int main(){

read(n);

ac.init();

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

scanf("%s",a);

ac.insert(a,i);

}

ac.getFail();

for(int i=1;i<=ac.num;i++){

add\_edge(ac.f[i],i);LOG[i]=LOG[i>>1]+1;

}

read(m);

dfs(0);

lim=tcnt;

while(m--){

read(opt);

if(opt==1){

scanf("%s",a);int lo=0;

int len=strlen(a);cnt=0;

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

lo=ac.ch[lo][a[i]-'a'];

tem[cnt++]=lo;

}

sort(tem,tem+cnt,cmp);

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

int who=tem[i];

add(st[who],1);

}

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

int u=tem[i],v=tem[i+1];

int LCA=lca(u,v);

add(st[LCA],-1);

}

}

else{

int id;read(id);

int lo=pos[id];

printf("%d\n",query(en[lo])-query(st[lo]-1));

}

}

return 0;

}

## 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;

}

## 后缀数组

**例1-BZOJ5073**

const int MAXN=2e5+5;

int t1[MAXN],t2[MAXN],c[MAXN];

bool cmp(int \*r,int a,int b,int l){

return r[a]==r[b]&&r[a+l]==r[b+l];

}

void da(int str[],int sa[],int rank[],int height[],int n,int m){

str[n++]=0;

int i,j,p,\*x=t1,\*y=t2;

for(i=0;i<m;i++)c[i]=0;

for(i=0;i<n;i++)c[x[i]=str[i]]++;

for(i=1;i<m;i++)c[i]+=c[i-1];

for(i=n-1;i>=0;i--)sa[--c[x[i]]]=i;

for(j=1;j<=n;j<<=1){

p=0;

for(i=n-j;i<n;i++)y[p++]=i;

for(i=0;i<n;i++)if(sa[i]>=j)y[p++]=sa[i]-j;

for(i=0;i<m;i++)c[i]=0;

for(i=0;i<n;i++)c[x[y[i]]]++;

for(i=1;i<m;i++)c[i]+=c[i-1];

for(i=n-1;i>=0;i--)sa[--c[x[y[i]]]]=y[i];

swap(x,y);

p=1;x[sa[0]]=0;

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

x[sa[i]]=cmp(y,sa[i-1],sa[i],j)?p-1:p++;

if(p>=n)break;

m=p;

}

int k=0;n--;

for(i=0;i<=n;i++)rank[sa[i]]=i;

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

if(k)k--;

j=sa[rank[i]-1];while(str[i+k]==str[j+k])k++;

height[rank[i]]=k;

}

}

int sa[MAXN],st[MAXN];

int rank[MAXN],height[MAXN];

int RMQ[MAXN];

int mm[MAXN];

int best[20][MAXN];

void initRMQ(int n){

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

mm[0]=-1;

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

mm[i]=((i&(i-1))==0)?mm[i-1]+1:mm[i-1];

for(int i=1;i<=n;i++)best[0][i]=i;

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

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

int a=best[i-1][j];

int b=best[i-1][j+(1<<i-1)];

if(RMQ[a]<RMQ[b])best[i][j]=a;

else best[i][j]=b;

}

}

int askRMQ(int a,int b){

int t;t=mm[b-a+1];

b-=(1<<t)-1;

a=best[t][a];b=best[t][b];

return RMQ[a]<RMQ[b]?a:b;

}

int lcp(int a,int b){

a=rank[a];b=rank[b];

if(a>b)swap(a,b);

return height[askRMQ(a+1,b)];

}

char str[MAXN],str2[MAXN];

int n,m,k;

int len;

int dp[MAXN][102];

int main(){

int T;

read(T);

while(T--){

read(n);read(m);read(k);

scanf("%s",str);scanf("%s",str2);

len=0;

for(int i=0;i<n;i++)st[len++]=str[i]-'a'+1;

st[len++]=27;

for(int i=0;i<m;i++)st[len++]=str2[i]-'a'+1;

da(st,sa,rank,height,len,28);

for(int i=1;i<=len;i++)RMQ[i]=height[i];

memset(dp,0,sizeof(dp));

initRMQ(len);

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

for(int t,j=0;j<=k;j++)

dp[i+1][j]=max(dp[i+1][j],dp[i][j]),t=lcp(i,dp[i][j]+n+1),dp[i+t][j+1]=max(dp[i+t][j+1],dp[i][j]+t);

int i;

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

if(dp[n][i]==m)break;

puts(i<=k?"YES":"NO");

}

return 0;

}

**例2-2019 bytedance MW camp Day3 I**

**题意**：询问k使得lcp(si,sj)=k且lcp(si,sj)字典序最小的情况下(i,j)最小

**做法**：求出height后，按字典序由小到大插入height，使用单调栈维护。

char s[MAX];

int a[MAX];

int rank[MAXN],height[MAXN],sa[MAXN];

int stk[MAX],id[MAX],top;

pii ans[MAX];

int qs;

bool used[MAX];

int n;

void ad(int len,int lo){

int lt=n+1;//small val

while(stk[top]>=len){

if(lt!=n+1&&!used[stk[top]])

ans[stk[top]]=min(ans[stk[top]],mp(min(lt,id[top]),max(lt,id[top])));

if(stk[top]!=len)used[stk[top]]=1;

lt=min(lt,id[top]);--top;

}

if(lt!=n+1&&lo!=n+1&&!used[len])

ans[len]=min(ans[len],mp(min(lt,lo),max(lt,lo)));

stk[++top]=len,id[top]=lt;

stk[++top]=n+1,id[top]=lo;

}

int main(){

scanf("%s",s);

n=strlen(s);

for(int i=0;i<n;i++)a[i]=s[i]-'a'+1;

da(a,sa,rank,height,n,26+1);

stk[0]=-1;

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

ans[i]=mp(n+1,n+1);

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

int lo=sa[i]+1,len=height[i];

ad(len,lo);ad(n-lo+1,lo);

}

ad(0,n+1);

read(qs);

while(qs--){

int val;read(val);

if(ans[val].fi==n+1)puts("-1 -1");

else printf("%d %d\n",ans[val].first,ans[val].second);

}

return 0;

}

## 序列自动机

**例1-wannafly-wintercamp-Day3-J**

**//序列自动机: 假设y之后第一次出现位置为nxt[x][y]**

**//f[x]=f[x]+f[nxt[x][y]]; 所以只与head是几有关**

**//trans[head][last(nextvalue)]**

**//保存的是第一个head处的值**

struct matrix{

int trans[30][30];

bool have[30];

matrix(){memset(trans,0,sizeof(trans));memset(have,0,sizeof(have));}

};

matrix mul(matrix A,matrix B){

matrix ret;

for(int i=0;i<=26;i++){

if(A.have[i]){

for(int j=0;j<=26;j++){

if(B.have[j])

for(int k=0;k<=26;k++)

addi(ret.trans[i][k],(int)mul(A.trans[i][j],B.trans[j][k]));

else addi(ret.trans[i][j],A.trans[i][j]);

}

}

else for(int j=0;j<=26;j++)addi(ret.trans[i][j],B.trans[i][j]);

}

for(int i=0;i<=26;i++)ret.have[i]=A.have[i]|B.have[i];

return ret;

}

matrix getMat(int c){

c-='a';

matrix re;re.have[c]=1;

for(int i=0;i<=26;i++)re.trans[c][i]=1;

return re;

}

int n;

char a[2005];

int main(){

read(n);

scanf("%s",a);

matrix ans=getMat(a[n-1]);

int re=0;

for(int i=n-2;i>=0;i--)ans=mul(ans,mul(getMat(a[i]),ans));

for(int i=0;i<26;i++)addi(re,ans.trans[i][26]);

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

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-SDOI2016-征途**

给n个数，将其分成m段，使所有段长度平方和最小。

int n,m,now,pre;

ll p[MAX];

int dp[2][MAX];

ll getY(int id){

return dp[pre][id]+p[id]\*p[id];

}

struct convex\_hull{

typedef pll point;

#define x first

#define y second

point p[MAX];

int head,tail;///站着上头下尾

convex\_hull():head(0),tail(0){}

inline void init(){

head=tail=0;

}

inline ll cross(point &p,point &q,point &r){

return (q.x-p.x)\*(r.y-p.y)-(q.y-p.y)\*(r.x-p.x);

}

inline void insert(ll x,ll y){

point q(x,y);

while(head>tail+1&&cross(p[head-1],p[head],q)<=0)--head;

p[++head]=q;

}

///在查询点需要的斜率单调时 得到用谁来转移当前的dp值

inline void update(ll x){

while(head>tail+1&&(p[tail+2].y-p[tail+1].y)<=2\*x\*(p[tail+2].x-p[tail+1].x) )++tail;

}

#undef x

#undef y

}hull;

int main(){

read(n);read(m);

pre=1;

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

read(p[i]);p[i]+=p[i-1];

}

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

dp[now][i]=p[i]\*p[i];

for(int j=2;j<=m;j++){

swap(now,pre);

memset(dp[now],0x3f,sizeof(dp[now]));

hull.init();

ll tem=getY(j-1);

hull.insert(p[j-1],tem);

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

hull.update(p[i]);

ll px=hull.p[hull.tail+1].fi,dpval=hull.p[hull.tail+1].se-px\*px;

dp[now][i]=dpval+(p[i]-px)\*(p[i]-px);

hull.insert(p[i],getY(i));

}

}

printf("%lld\n",m\*dp[now][n]-p[n]\*p[n]);

return 0;

}

**斜率优化+CDQ分治-例2-POJ1180**

用pre[i]表示w的前缀和，容易推出 dp[i]=min{ dp[j] +(h[i]-h[j])^2 +(pre[i-1]-pre[j]) }

但是这里h[i]不单调递增，于是需要使用CDQ分治来分治的做，采用归并排序nlog即可。

int n;

inline ll sqr(ll x){return x\*x;}

struct node{

ll h,w,pre,dp,id,y;

bool operator <(const node & z)const{

if(h!=z.h)

return h<z.h;

return id<z.id;

}

ll getY(){

return dp-pre+h\*h;

}

}a[MAX],tmp[MAX];

struct convex\_hull{

typedef pll point;

#define x first

#define y second

point p[MAX];

int head,tail;

convex\_hull():head(0),tail(0){}

inline void init(){

head=tail=0;

}

inline ll cross(point &p,point &q,point &r){

return (q.x-p.x)\*(r.y-p.y)-(q.y-p.y)\*(r.x-p.x);

}

inline void insert(ll x,ll y){

point q(x,y);

while(head>tail+1&&cross(p[head-1],p[head],q)<=0)--head;

p[++head]=q;

}

inline void update(ll x){

while(head>tail+1&&(p[tail+2].y-p[tail+1].y)<=2\*x\*(p[tail+2].x-p[tail+1].x ) )

++tail;

}

#undef x

#undef y

}hull;

void CDQ(int l,int r){

if(l==r){

a[l].y=a[l].getY();

return ;

}

int mid=(l+r)/2;

int stl=l,str=mid+1;

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

if(a[i].id<=mid)tmp[stl++]=a[i];

else tmp[str++]=a[i];

}

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

CDQ(l,mid);

hull.init();

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

hull.insert(a[i].h,a[i].y);

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

hull.update(a[i].h);

ll X=hull.p[hull.tail+1].fi,Y=hull.p[hull.tail+1].se;

a[i].y=Y-2\*a[i].h\*(X-a[i].h)-a[i].w;

a[i].dp=min(a[i].dp,a[i].y+a[i].pre-sqr(a[i].h));

}

CDQ(mid+1,r);

stl=l,str=mid+1;

int lo=l;

while(stl<=mid&&str<=r){

if(a[stl].h==a[str].h){

if(a[stl].y<a[str].y)tmp[lo++]=a[stl++];

else tmp[lo++]=a[str++];

}

else{

if(a[stl].h<a[str].h)

tmp[lo++]=a[stl++];

else tmp[lo++]=a[str++];

}

}

while(stl<=mid)tmp[lo++]=a[stl++];

while(str<=r)tmp[lo++]=a[str++];

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

a[i]=tmp[i];

}

int main(){

read(n);

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

read(a[i].h);

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

read(a[i].w);a[i].pre=a[i-1].pre+a[i].w;a[i].id=i;

a[i].dp=INFF/2LL;

}

a[1].dp=0;

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

CDQ(1,n);

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

if(a[i].id==n)printf("%lld\n",a[i].dp);

return 0;

}

## 树形DP

**例0-**[**2018牛客暑期多校第6场G**](https://www.nowcoder.com/acm/contest/144/G)

题意：给定一棵树。n个点重构为一个完全图，任意两点之间的边定义为树上两点之间的长度。求这个新图中任意两点最大流之和。

做法：最大流显然转化为最小割。即求S到T中边的和的最小值。由于总和一定，也就是求S中的和与T中的和两者之和的最大值。显然除了s和t外，其余点都在同一个集合中时其和最大。那么我们就是要min｛s到其他点距离之和，t到其他点距离之和｝。

这个可以通过两次dfs进行树上DP来处理。求出所有的之后排序，从小到大，第i小的贡献为n-I (其在n-i个点对中作为答案进行贡献）。总和会超过long long,使用两个long long模拟一下大数即可。

树上DP。先DP每个点到其子树所有点的距离之和。再dp每个点到其非子树中的点距离之和。二者求和即可。

const ll lim=1e15;

int t,n,ecnt;

struct node{

ll x0,x1;

void init(){

x0=x1=0;

}

void add(ll z){

x0+=z/lim;

x1+=z%lim;

if(x1>=lim)x0+=x1/lim,x1%=lim;

}

void print(){

if(x0)printf("%lld%015lld\n",x0,x1);

else printf("%lld\n",x1);

}

}ans;

struct nod{

int to,nxt;

ll val;

nod(){};

nod(int \_to,int \_nxt,ll \_val):to(\_to),nxt(\_nxt),val(\_val){};

}edg[MAX<<1];

int h[MAX],siz[MAX];

ll dp[MAX],fp[MAX];

void add\_edge(int u,int v,ll w){

edg[++ecnt]=nod(v,h[u],w);

h[u]=ecnt;

}

void dfs1(int now,int fa){

siz[now]=1;dp[now]=0;

for(int i=h[now];i;i=edg[i].nxt){

int to=edg[i].to;if(to==fa)continue;

ll ju=edg[i].val;

dfs1(to,now);

dp[now]+=dp[to]+ju\*siz[to];

siz[now]+=siz[to];

}

}

void dfs2(int now,int fa){

for(int i=h[now];i;i=edg[i].nxt){

int to=edg[i].to;

if(to==fa)continue;

ll ju=edg[i].val;

fp[to]=ju\*(n-2LL\*siz[to])+fp[now]+dp[now]-dp[to];

dfs2(to,now);

}

dp[now]+=fp[now];

}

int main()

{

read(t);

for(int Case=1;Case<=t;Case++){

read(n);

memset(h,0,sizeof(h));

ecnt=0;

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

ll w;

read(u);read(v);read(w);

add\_edge(u,v,w);add\_edge(v,u,w);

}

dfs1(1,0);dfs2(1,0);

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

ans.init();

printf("Case #%d: ",Case);

for(int i=1;i<n;i++)ans.add(dp[i]\*(n-i));

ans.print();

}

}

**例2- 2019 wannafly winter camp Day3 E**

**题意：**

一棵n个点的树，有n种有根树。进行m次操作，每次操作给定一个值x，加入一个以x为根的最初树的有根树。将加入前的树复制n份，将其根连到以x为根的有根树的各个节点上。对于m次操作，以及最初状态（最初以1为根），输出最大独立集的大小。n<=1e5

**做法：**

首先，对于固定的一棵树，既可以简单的树形DP，也可以有如下贪心策略：

选择所有叶子节点作为独立集中的元素，抹去其父节点，直到所有点均被选过或被抹去了，此时选的点数目即为最大独立集的大小。

注意到，题目给定的操作，实际上只与加点前的树的根节点在上述贪心策略中是否被选中有关。

如果加点前的树的根节点被选中了，则其父节点都应该被抹去，即新加入的某个以x为根的n个点的树的所有点均被抹去了，且新形成的树的根节点没有被选。即此时新的ans’=ans\*n

如果加点前的树的根节点没有被选中，则相当于没有任何影响的处理此时以x为根的n个点的树。

此时新的ans'=ans\*n+ n个点的树的最大独立集的大小，新的树的根节点是否被选只与最初n个点状态下以x为根的有根树的贪心策略下最大独立集中x有没有被选有关。

于是，我们只需要预先树形DP出整棵树的最大独立集大小、以每个点分别为根的有根树贪心策略的最大独立集中根节点是否有被选即可。通过两次dfs即可。

使用的变量如下：

int dp[MAX];//以i为根的子树取最大独立集 i选(1)or 不选(0)

int tp[MAX];//整棵树除了以i为根的子树的部分 i的父节点是否选

int cnt[MAX];//以i为根的子树中的各个儿子结点的子树的根选的个数

int rt[MAX];//以i为根的整棵树最大独立集中i是否选 1表示选了

int n,m;

vector<int>edg[MAX];

int dp[MAX];//以i为根的子树取最大独立集 i选(1)or 不选(0)

int tp[MAX];//整棵树除了以i为根的子树的部分 i的父节点是否选

int cnt[MAX];//以i为根的子树中的各个儿子结点的子树的根选的个数

int rt[MAX];//以i为根的整棵树最大独立集中i是否选 1表示选了

int stans;//初始的树的最大独立集大小

void dfs\_dp(int now,int fa){

dp[now]=1;

for(int u:edg[now]){//计算dp

if(u==fa)continue;

dfs\_dp(u,now);

if(dp[u])//有选的

dp[now]=0,++cnt[now];

}

stans+=dp[now];

}

void dfs\_tp(int now,int fa){

cnt[now]+=tp[now];

rt[now]=!cnt[now];

for(int u:edg[now]){

if(u==fa)continue;

cnt[now]-=dp[u];

tp[u]=!cnt[now];

dfs\_tp(u,now);

cnt[now]+=dp[u];

}

}

int ans;

int main(){

read(n);read(m);

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

read(u);read(v);

edg[u].eb(v);edg[v].eb(u);

}

dfs\_dp(1,0);

dfs\_tp(1,0);

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

bool rtst=rt[1];//根节点的状态

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

read(u);

if(!rtst)//根节点未选

muli(ans,n),addi(ans,stans),rtst=rt[u];

else muli(ans,n),rtst=0;//根节点已被选

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

}

return 0;

}

**例3- ZJOI2016 小星星**

**题意：**

给一棵n个点的树和一个n个点m条边的无向图。   
问有多少种方案把树上的节点一一对应到一个无向图上的节点，   
且树上两点间有边时，他们在无向图上对应的两个点之间也要有边。   
n≤17

**做法**：原有的m条边实际相当于树上边的限制关系。用dp[i][j]表示，树上的第i个点为根的子树，i对应原图中的第j个点，的合法分配方式数。这样很容易得到一个 n^3的dp。但是这个dp无法满足：一个对应只出现一次。解决这一问题的方法，显然就是容斥。

容斥只需枚举**实际可用的原图中的点的集合**。这是2^n种集合。于是总复杂度 O(2^n\*n^3)考虑到远远不满，因此这样就已经可以做了。

int n,m;

bool ava[20][20];

char e[20][20],ecnt[20];

char a[20],acnt;

ll dp[20][20];

ll ans,sum;

void dfs(int now,int fa){

for(char i=0;i<acnt;i++)dp[now][a[i]]=1;

for(int i=0;i<ecnt[now];i++){

int to=e[now][i];

if(to==fa)continue;

dfs(to,now);

for(int j=0;j<acnt;j++){//当前节点颜色

ll tot=0;

for(int k=0;k<acnt;k++)tot+=dp[to][a[k]]\*ava[a[j]][a[k]];

dp[now][a[j]]\*=tot;

}

}

}

int main(){

read(n);read(m);

for(char u,v,i=1;i<=m;i++){

read(u);read(v);--u;--v;

ava[u][v]=ava[v][u]=1;

}

for(char u,v,i=1;i<n;i++){

read(u);read(v);--u;--v;

e[u][ecnt[u]++]=v;

e[v][ecnt[v]++]=u;

}

int tot=1<<n;

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

acnt=0;

int now=i;

int ge=n;

for(char j=0;j<n;j++)

if((now>>j)&1)

a[acnt++]=j,--ge;

sum=0;

dfs(0,-1);

for(char i=0;i<acnt;i++)sum+=dp[0][a[i]];

if(ge&1)ans-=sum;

else ans+=sum;

}

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

return 0;

}

## 概率DP

**1、2019 wannafly winter camp Day4-H**

**题意**：6个人大富翁，按顺序依次掷骰子，给定每个人掷出1-6的概率，第一个到某个位置的人买到该地。求500轮后每个人地数的期望。（地为长度为n的环，所有人都从1出发，第二次到1才能买1）

**做法**：

难点1：dp什么？这里选择的dp是用dp[i][j][k]表示第i个人在前j轮，没到过第k个位置的概率。其转移即为dp[i][j][k]=sigma(dp[i][j-1][k-l]) // l: 1——6。并且dp[i][j-1][k]-dp[i][j][k]即为第i个人在第j轮第一次到达k的概率。

难点2：上述dp时，尽管题目说了出发时在1并不能买地，但转移上述时，1是在最开始就到过的。用[0,n-1]的i表示实际到过某地i+1的概率(最开始都到过了0，即对应原图中的1）。用下标为n表示第一次能买1这块地的概率。

再枚举一下每个人每个轮第一次到某个位置的概率，计算最终答案即可。

double dp[7][505][505];//第i个人在第j轮之前没到过第k个位置的概率

double p[7][7];

int n;

double ans[7];

int main(){

read(n);

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

for(int j=1;j<=6;j++)scanf("%lf",&p[i][j]);

for(int i=1;i<=6;i++){//人

for(int j=1;j<=n;j++)dp[i][0][j]=1;//第0轮没到过任何一个点

dp[i][0][0]=0;//第0轮到过0

for(int j=1;j<=500;j++){//轮数

dp[i][j][0]=0;//全都到过0

for(int k=1;k<=n;k++)//位置

for(int t=1;t<=6;t++)//步数

dp[i][j][k]+=p[i][t]\*dp[i][j-1][(k-t+n\*6)%n];//n可能很小 t最大为6 所以6\*n

}

}

for(int i=1;i<=6;i++)//人

for(int j=1;j<=500;j++)//轮

for(int k=1;k<=n;k++){//位置

double pos=1;

for(int t=1;t<i;t++)pos\*=dp[t][j][k];

for(int t=i+1;t<=6;t++)pos\*=dp[t][j-1][k];

pos\*=dp[i][j-1][k]-dp[i][j][k];

ans[i]+=pos;

}

for(int i=1;i<=6;i++)printf("%.3f\n",ans[i]);

return 0;

}

## 数位DP

**1、小于等于n的不出现49的数字的个数**

ll dp[30][2];//第几位 上一位是不是4

int dig[30];//最大范围的每一位数 下标从1开始 下标为1表示个位的数

ll DP(int wei,bool si,bool lim)//第几位 上一位是否为4 是否仍在边界限制（e.g. 范围568 到此时前两位为56 即有限制）

{

if(!wei)return 1;//此为到0位不需要特别判断的直接返回1（个数）

if(!lim&&dp[wei][si]!=-1)return dp[wei][si];//之前计算过直接返回

int da=9;//该位最大的数值

ll re=0;

if(lim)da=dig[wei];//若有边界限制 上界取为边界

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

{

if(si&&i==9)continue;//不能出现49

re+=DP(wei-1,i==4,lim&&(i==da));

}

if(!lim)dp[wei][si]=re;//若无边界限制 可以更新该位的值 （若有限制的话 此值随数的变化而变 无需记忆）

return re;

}

ll cal(ll x)

{

int wei=0;

while(x)//求出每一位的数码

{

dig[++wei]=x%10;x/=10;

}

return DP(wei,0,1);

}

int t;

ll n;

int main()

{

memset(dp,-1,sizeof(dp));//dp数组初始化为-1

scanf("%d",&t);

while(t--)

{

cin>>n;

cout<<(n-cal(n)+1LL)<<"\n";

}

return 0;

}

**2-小于等于n的二进制下各位为0/1的数的个数**

inline ll cal\_1(ll x){//小于等于x的奇数个数

return (x+1)/2;

}

void get\_mcnt(ll val){//对于val计算mcnt

for(int i=0;i<=31;i++){

ll ge=(val>>i);//数位dp无lim的部分(包含0)

if(ge==0)break;

ll num=cal\_1(ge-1);//无lim部分该位置为1的个数

addi(mcnt[i][1],(num<<i)%MOD);

addi(mcnt[i][0],((ge-num)<<i)%MOD);

ll limcnt=(val-(ge<<i)+1)%MOD;//有lim的数的个数

if(val&(1<<i))addi(mcnt[i][1],limcnt);

else addi(mcnt[i][0],limcnt);

sub(mcnt[i][0],1);

}

}

## 插头DP

**例1-HDU1693**

**题意**：在n\*m的矩阵中，有些格子有树，没有树的格子不能到达，找一条或多条回路，吃完所有的树，求有多少种方法。

**做法**：dp[i][j][k]表示到第i行第j列的格子后插头状况（轮廓线）为k的方案数。

最开始纠结的是初始化问题。其实为了代码方便，可以初始化 f[0][m][0]=1，这样在处理换行的时候就会自动把这个顺延到第一行。换行处理就左移一下，因为前一行的最后一个（右插头）和当前行的第一个（右插头）必定是空的。

**做法1-简短版**

int t,n,m;

int a[15][15];

ll dp[15][15][(1<<14)|1];

int main(){

read(t);

for(int Case=1;Case<=t;Case++){

memset(dp,0,sizeof(dp));

read(n);read(m);

int tot=1<<(m+1);

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

for(int j=1;j<=m;j++)read(a[i][j]);

dp[0][m][0]=1;

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

///m个格子，轮廓线共m+1部分

///新的一行，原本编号i的轮廓变为i+1，新的编号为0的轮廓线均为0

for(int k=0;k<(1<<m);k++)

dp[i][0][k<<1]=dp[i-1][m][k];

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

for(int k=0;k<tot;k++){///到该格子时的轮廓线

bool lef=((k>>(j-1))&1),up=((k>>j)&1);

int x=((1<<(j-1))|(1<<j));

if(!a[i][j]){//不能放

if(!lef&&!up)//不能有插头

dp[i][j][k]+=dp[i][j-1][k];

}

else if(lef^up){//恰有一个插头

dp[i][j][k]+=dp[i][j-1][k];

dp[i][j][k^x]+=dp[i][j-1][k];

}

else//没有或同时有两个插头

dp[i][j][k^x]+=dp[i][j-1][k];

}

}

}

printf("Case %d: There are %lld ways to eat the trees.\n",Case,dp[n][m][0]);

}

}

**做法2-使用之前冗长模板版**

const int HASH=10007;

const int STATE=1000010;

const int MAXD=15;

int N,M;

int code[MAXD],maze[MAXD][MAXD];

struct HASHMAP

{

int head[HASH],next[STATE],state[STATE],size;//链表形式 head通过取模进行了一种索引 提高查询效率

long long f[STATE];//到某状态的个数

void init()

{

size=0;memset(head,-1,sizeof(head));

}

void push(int st,long long ans)//达到第i种状态的方式加ans个

{

int i,h=st%HASH;//h为哈希过后的编码 作为索引

for(i=head[h];i!=-1;i=next[i])

if(st==state[i])//若成功找到

{

f[i]+=ans;

return;

}

//未找到，需自己创建

f[size]=ans;//达到该状态的方法数即为ans

state[size]=st;//其具体编码

next[size]=head[h];

head[h]=size++;

}

}hm[2];//滚动数组，节省空间

/\*

对st进行解码，将结果保存在code数组中，其长度为m

\*/

void decode(int \*code,int m,int st)

{

int i;

for(i=m;i>=0;i--)

{

code[i]=st&1;

st>>=1;

}

}

/\*

将当前的长度为m的code数组进行编码

\*/

int encode(int \*code,int m)

{

int i,st=0;

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

{

st<<=1;st|=code[i];

}

return st;

}

/\*

可作为通用的读入函数

注意需要在右侧、下策边界设一圈代表不可“到达”的数

\*/

void init()

{

int i,j;

scanf("%d%d",&N,&M);

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

for(j=1;j<=M;j++)

scanf("%d",&maze[i][j]);

for(int i=1;i<=N;i++)maze[i][M+1]=0;

for(int i=1;i<=M;i++)maze[N+1][i]=0;

}

/\*

换行的时候移位

\*/

void shift(int \*code,int m)

{

int i;

for(i=m;i>0;i--)code[i]=code[i-1];

code[0]=0;

}

/\*

处理可以放的格子

i行，j列，当前滚动数组的状态

\*/

void dpblank(int i,int j,int cur)

{

int k,left,up;

for(k=0;k<hm[cur].size;k++)

{

decode(code,M,hm[cur].state[k]);

left=code[j-1];//左侧是否有连

up=code[j];//上侧是否有连

if(left&&up)//11 -> 00

{

code[j-1]=code[j]=0;

if(j==M)shift(code,M);

hm[cur^1].push(encode(code,M),hm[cur].f[k]);

}

else if(left||up)//01 或 10

{

if(maze[i][j+1])//可以向右侧连 并且并不涉及换行 无需shift

{

code[j-1]=0;code[j]=1;

hm[cur^1].push(encode(code,M),hm[cur].f[k]);

}

if(maze[i+1][j])

{

code[j-1]=1;code[j]=0;

if(j==M)shift(code,M);

hm[cur^1].push(encode(code,M),hm[cur].f[k]);

}

}

else//00 -> 11 必须两个都可以连

{

if(maze[i][j+1]&&maze[i+1][j])

{

code[j]=code[j-1]=1;

hm[cur^1].push(encode(code,M),hm[cur].f[k]);

}

}

}

}

/\*

不能放的格子

i行，j列，当前滚动数组的状态

\*/

void dpblock(int i,int j,int cur)

{

int k;

for(k=0;k<hm[cur].size;k++)

{

decode(code,M,hm[cur].state[k]);

code[j-1]=code[j]=0;

if(j==M)shift(code,M);

hm[cur^1].push(encode(code,M),hm[cur].f[k]);

}

}

void solve()

{

int i,j,cur=0;

long long ans=0;

hm[cur].init();

hm[cur].push(0,1);//dp前初始化 仅状态0的个数为1（即当下）

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

for(j=1;j<=M;j++)

{

hm[cur^1].init();//滚动数组的初始化

if(maze[i][j])dpblank(i,j,cur);

else dpblock(i,j,cur);

cur^=1;

}

for(i=0;i<hm[cur].size;i++)

ans+=hm[cur].f[i];

printf("There are %I64d ways to eat the trees.\n",ans);

}

int main()

{

int T;

int iCase=0;

scanf("%d",&T);

while(T--)

{

iCase++;

printf("Case %d: ",iCase);

init();

solve();

}

return 0;

}

**例2-URAL1519**

**题意：**

给你一个m \* n的棋盘，有的格子是障碍，问共有多少条回路使得经过每个非障碍格子恰好一次．m, n ≤ 12。

**做法1-摘自网上博客**

#define ll long long

#define hash ddf

using namespace std;

int n,m,x[15][15],cur,pre,ex,ey;

int st[2][300010];ll ans[2][300010],re;

int tot[2],bit[20],state[300010],st\_tot,hash=300000;

struct edge{

int to,next;

}a[300010];

void insert(int sta,ll val){

// cout<<"insert "<<sta<<ends<<val<<endl;

int p=sta%hash,i;

for(i=state[p];i;i=a[i].next){

if(st[cur][a[i].to]==sta){

ans[cur][a[i].to]+=val;return;

}

}

tot[cur]++;

a[++st\_tot].to=tot[cur];

a[st\_tot].next=state[p];

state[p]=st\_tot;st[cur][tot[cur]]=sta;ans[cur][tot[cur]]=val;

}

int main(){

int i,j,k,l,now,down,right;ll val;char s[20];

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

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

scanf("%s",s);

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

if(s[j]=='.')

x[i][j+1]=1,ex=i,ey=j+1;

}

for(i=1;i<15;i++) bit[i]=i<<1;

cur=0;tot[cur]=1;ans[cur][1]=1;st[cur][1]=0;

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

for(j=1;j<=tot[cur];j++) st[cur][j]<<=2;

for(j=1;j<=m;j++){

// cout<<"begin "<<i<<ends<<j<<endl;

st\_tot=0;memset(state,0,sizeof(state));

pre=cur;cur^=1;tot[cur]=0;

for(k=1;k<=tot[pre];k++){

now=st[pre][k];val=ans[pre][k];

down=(now>>bit[j-1])%4;right=(now>>bit[j])%4;

// cout<<" from "<<now<<ends<<val<<ends<<down<<ends<<right<<endl;

if(!x[i][j]){

if(!down&&!right){

insert(now,val);continue;

}

}

else if(!down&&!right){

if(x[i][j+1]&&x[i+1][j])

insert(now+(1<<bit[j-1])+((1<<bit[j])<<1),val);

}

else if(!down&&right){

if(x[i][j+1]) insert(now,val);

if(x[i+1][j])

insert(now-right\*(1<<bit[j])+right\*(1<<bit[j-1]),val);

}

else if(down&&!right){

if(x[i+1][j]) insert(now,val);

if(x[i][j+1])

insert(now+down\*(1<<bit[j])-down\*(1<<bit[j-1]),val);

}

else if(down==1&&right==1){

int cnt=1;

for(l=j+1;l<=m;l++){

if((now>>bit[l])%4==1) cnt++;

if((now>>bit[l])%4==2) cnt--;

if(!cnt){

insert(now-(1<<bit[l])-(1<<bit[j])-(1<<bit[j-1]),val);

break;

}

}

}

else if(down==2&&right==2){

int cnt=1;

for(l=j-2;l>=0;l--){

if((now>>bit[l])%4==2) cnt++;

if((now>>bit[l])%4==1) cnt--;

if(!cnt){

insert(now+(1<<bit[l])-((1<<bit[j])<<1)-((1<<bit[j-1])<<1),val);

break;

}

}

}

else if(down==2&&right==1){

insert(now-((1<<bit[j-1])<<1)-(1<<bit[j]),val);

}

else if(down==1&&right==2){

if(i==ex&&j==ey) re+=val;

}

}

}

}

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

}

**做法2-冗长模版版**

const int HASH=30007;

const int STATE=1000010;

const int MAXD=15;

int N,M,ex,ey;//ex\ey非必需

int code[MAXD],maze[MAXD][MAXD];

int ch[MAXD];

struct HASHMAP

{

int head[HASH],next[STATE],size;//链表形式 head通过取模进行了一种索引 提高查询效率

long long f[STATE],state[STATE];//到某状态的个数

void init()

{

size=0;memset(head,-1,sizeof(head));

}

void push(long long st,long long ans)//达到第i种状态的方式加ans个

{

int i,h=st%HASH;//h为哈希过后的编码 作为索引

for(i=head[h];i!=-1;i=next[i])

if(st==state[i])//若成功找到

{

f[i]+=ans;

return;

}

//未找到，需自己创建

f[size]=ans;//达到该状态的方法数即为ans

state[size]=st;//其具体编码

next[size]=head[h];

head[h]=size++;

}

}hm[2];//滚动数组，节省空间

/\*

对st进行解码，将结果保存在code数组中，其长度为m

\*/

void decode(int \*code,int m,long long st)

{

int i;

for(i=m;i>=0;i--)

{

code[i]=st&7;

st>>=3;

}

}

/\*

将当前的长度为m的code数组进行编码

\*/

long long encode(int \*code,int m)

{

int i;

int cnt=1;

memset(ch,-1,sizeof(ch));

ch[0]=0;

ll st=0;

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

{

if(ch[code[i]]==-1)ch[code[i]]=cnt++;

code[i]=ch[code[i]];

st<<=3;

st|=code[i];

}

return st;

}

/\*

可作为通用的读入函数

注意需要在右侧、下策边界设一圈代表不可“到达”的数

\*/

char lin[20];

/\*

读入数据，将状态图记录在maze中

\*/

void init()

{

ex=ey=0;

memset(maze,0,sizeof(maze));

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

{

scanf("%s",lin);

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

if(lin[j]=='.'){maze[i][j+1]=1;ex=i;ey=j+1;}

}

}

/\*

换行的时候移位

不想改变code数组 可以采用 encode(code,j==M?M-1:M);的方式

\*/

void shift(int \*code,int m)

{

int i;

for(i=m;i>0;i--)code[i]=code[i-1];

code[0]=0;

}

/\*

处理可以放的格子

i行，j列，当前滚动数组的状态

\*/

void dpblank(int i,int j,int cur)

{

int k,left,up;

for(k=0;k<hm[cur].size;k++)

{

decode(code,M,hm[cur].state[k]);

left=code[j-1];//左侧是否有连 及其连通编号

up=code[j];//上侧是否有连 及其连通编号

if(left&&up)//11 -> 00

{

if(left==up)//左、上已经连通

{

if(i==ex&&j==ey)

{

code[j-1]=code[j]=0;

if(j==M)shift(code,M);

hm[cur^1].push(encode(code,M),hm[cur].f[k]);

}

}

else//左、上并未连通 则合并

{

code[j-1]=code[j]=0;

for(int s=0;s<=M;s++)

if(code[s]==up)code[s]=left;

if(j==M)shift(code,M);

hm[cur^1].push(encode(code,M),hm[cur].f[k]);

}

}

else if(left||up)//01 或 10

{

int t=left?left:up;

if(maze[i][j+1])//可以向右侧连 并且并不涉及换行 无需shift

{

code[j-1]=0;code[j]=t;

hm[cur^1].push(encode(code,M),hm[cur].f[k]);

}

if(maze[i+1][j])

{

code[j-1]=t;code[j]=0;

if(j==M)shift(code,M);

hm[cur^1].push(encode(code,M),hm[cur].f[k]);

}

}

else//00 -> 11 必须两个都可以连

{

if(maze[i][j+1]&&maze[i+1][j])

{

code[j]=code[j-1]=13;

hm[cur^1].push(encode(code,M),hm[cur].f[k]);

}

}

}

}

/\*

不能放的格子

i行，j列，当前滚动数组的状态

\*/

void dpblock(int i,int j,int cur)

{

int k;

for(k=0;k<hm[cur].size;k++)

{

decode(code,M,hm[cur].state[k]);

code[j-1]=code[j]=0;

if(j==M)shift(code,M);

hm[cur^1].push(encode(code,M),hm[cur].f[k]);

}

}

void solve()

{

int i,j,cur=0;

long long ans=0;

hm[cur].init();

hm[cur].push(0,1);//dp前初始化 仅状态0的个数为1（即当下）

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

for(j=1;j<=M;j++)

{

hm[cur^1].init();//滚动数组的初始化

if(maze[i][j])dpblank(i,j,cur);

else dpblock(i,j,cur);

cur^=1;

}

for(i=0;i<hm[cur].size;i++)

ans+=hm[cur].f[i];

printf("%I64d\n",ans);

}

int main()

{

while(~scanf("%d%d",&N,&M))

{

init();

if(!ex)

printf("0\n");

else solve();

}

return 0;

}

## nlogn最长上升子序列

const int MAXN=500010;

int a[MAXN],b[MAXN];

//用二分查找的方法找到一个位置，使得num>b[i-1] 并且num<b[i],并用num代替b[i]

int Search(int num,int low,int high)

{

int mid;

while(low<=high)

{

mid=(low+high)/2;

if(num>=b[mid]) low=mid+1;

else

high=mid-1;

}

return low;

}

int DP(int n)

{

int i,len,pos; b[1]=a[1];

len=1;

for(i=2;i<=n;i++)

{

if(a[i]>=b[len])//如果a[i]比b[]数组中最大还大直接插入到后面即可

{

len=len+1; b[len]=a[i];

}

else//用二分的方法在b[]数组中找出第一个比a[i]大的位置并且让a[i]替代这个位置

{

pos=Search(a[i],1,len);

b[pos]=a[i];

}

}

return len;

}

# 网络流

## 上下界网络流

**例1-SGU194-无源汇有上下界可行流**

namespace Maxflow{

typedef int type;

const int INF=0x3f3f3f3f;

const int maxn=4e4;

struct node{

int to;type cap;int nxt;int id;

node(int \_to=0,type \_cap=0,int \_nxt=0,int \_id=0):to(\_to),cap(\_cap),nxt(\_nxt),id(\_id){}

}edg[maxn\*50];

int head[maxn],tot;

void addedge(int from,int to,type cap,int \_id=0,type rcap=0){

edg[tot]=node(to,cap,head[from],\_id);

head[from]=tot++;

edg[tot]=node(from,rcap,head[to]);head[to]=tot++;

}

int dep[maxn],cur[maxn];

bool bfs(int s,int t,int n){

static int Q[maxn],ST,ED;

memset(dep+1,0,sizeof(int)\*n);

ST=0;ED=-1;

Q[++ED]=s;dep[s]=1;

while(ST<=ED){

int u=Q[ST++];

for(int i=head[u];~i;i=edg[i].nxt){

int to=edg[i].to;

if(!dep[to]&&edg[i].cap){

Q[++ED]=to;dep[to]=dep[u]+1;

}

}

}

return dep[t]!=0;

}

type dfs(int x,const int &t,type flow=INF){

if(x==t||flow==0)return flow;

type ret=0;

for(int i=cur[x];~i;i=edg[i].nxt){

if(dep[x]+1==dep[edg[i].to]&&edg[i].cap){

type f=dfs(edg[i].to,t,min(flow,edg[i].cap));

edg[i].cap-=f;edg[i^1].cap+=f;

ret+=f;flow-=f;cur[x]=i;

if(flow==0)break;

}

}

if(!ret)dep[x]=0;

return ret;

}

type maxflow(int s,int t,int n){

type ret=0;

while(bfs(s,t,n)){

type f;

memcpy(cur+1,head+1,n\*sizeof(int));

while((f=dfs(s,t))>0)ret+=f;

}

return ret;

}

void init(int n){

memset(head+1,-1,sizeof(int)\*n);tot=0;

}

}

int d[MAX],ans[MAX];

int n,m;

int main(){

read(n);read(m);

Maxflow::init(n+2);

for(int u,v,l,r,i=1;i<=m;i++){

read(u);read(v);read(l);read(r);

d[u]-=l;d[v]+=l;

Maxflow::addedge(u,v,r-l,i);

ans[i]=l;

}

int S=n+1,T=n+2;

int sum\_in=0,sum\_out=0;

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

if(d[i]>0){

sum\_out+=d[i];

Maxflow::addedge(S,i,d[i]);

}

else if(d[i]<0){

sum\_in-=d[i];

Maxflow::addedge(i,T,-d[i]);

}

}

if(sum\_in!=sum\_out||Maxflow::maxflow(S,T,n+2)!=sum\_in)

puts("NO");

else{

puts("YES");

for(int i=0;i<Maxflow::tot;i++)

if(Maxflow::edg[i].id)

ans[Maxflow::edg[i].id]+=Maxflow::edg[i^1].cap;

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

printf("%d\n",ans[i]);

}

return 0;

}

**例2-POJ2396-有源汇有上下界可行流**

int ans[404][24],a[404],low[404][24],up[404][24],deg[404];

int n,m,t,k;

inline void op(int x,int y,char opt,int z){

if(opt=='>')

low[x][y]=std::max(low[x][y],z+1);

else if (opt=='<')

up[x][y]=std::min(up[x][y],z-1);

else{

low[x][y]=max(low[x][y],z);

up[x][y]=min(up[x][y],z);

}

}

char opt;

int main(){

read(t);

while(t--){

read(n);read(m);

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

for(int j=1;j<=m;j++)low[i][j]=0,up[i][j]=INF;

for(int i=1;i<=n+m+2;i++)deg[i]=0;

for(int i=1;i<=n+m;i++)read(a[i]);

read(k);

while(k--){

int x,y,z;

read(x);read(y);

while(opt=getchar(),opt!='='&&opt!='<'&&opt!='>');

read(z);

if(!x&&!y)

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

for(int j=1;j<=m;j++)op(i,j,opt,z);

else if(!x)

for(int i=1;i<=n;i++)op(i,y,opt,z);

else if(!y)

for(int i=1;i<=m;i++)op(x,i,opt,z);

else op(x,y,opt,z);

}

bool an=1;

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

for(int j=1;j<=m&&an;j++)

if(low[i][j]>up[i][j])an=0;

if(!an){

puts("IMPOSSIBLE");

if(t)puts("");

continue;

}

int s=n+m+1,t=n+m+2;

int ss=t+1,tt=ss+1;

Maxflow::init(tt);

Maxflow::add\_edge(t,s,INF);

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

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

int l=low[i][j],r=up[i][j];

ans[i][j]=l;

Maxflow::add\_edge(i,j+n,r-l,i\*400+j);

deg[j+n]+=l;deg[i]-=l;

}

for(int i=1;i<=n;i++)deg[i]+=a[i],deg[s]-=a[i];

for(int i=n+1;i<=n+m;i++)deg[i]-=a[i],deg[t]+=a[i];

int sum\_in=0,sum\_out=0;

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

if(deg[i]>0){

sum\_in+=deg[i];Maxflow::add\_edge(ss,i,deg[i]);

}

else if(deg[i]<0){

sum\_out-=deg[i];Maxflow::add\_edge(i,tt,-deg[i]);

}

}

if(sum\_in!=sum\_out||Maxflow::maxflow(ss,tt,tt)!=sum\_in)

puts("IMPOSSIBLE");

else{

for(int i=0;i<Maxflow::tot;i+=2){

int id=Maxflow::edg[i].id;

if(id){

int x=id/400,y=id%400;

ans[x][y]+=Maxflow::edg[i^1].cap;

}

}

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

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

printf("%d%c",ans[i][j],j==m?'\n':' ');

}

if(t)puts("");

}

}

**例3-LOJ117-有源汇有上下界最小流（有局限性）**

int n,m,s,t;

ll deg[MAX];

int main(){

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

int S=n+1,T=n+2;

Maxflow::init(n+2);

memset(deg,0,sizeof(deg));

for(int u,v,z,i=1;i<=m;i++){

ll c;

read(u);read(v);read(z);read(c);

Maxflow::add\_edge(u,v,c-z);

deg[u]-=z;deg[v]+=z;

}

ll sum\_in=0,sum\_out=0;

int now=Maxflow::tot;

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

if(deg[i]>0)sum\_in+=deg[i],Maxflow::add\_edge(S,i,deg[i]);

else if(deg[i]<0)sum\_out-=deg[i],Maxflow::add\_edge(i,T,-deg[i]);

}

Maxflow::add\_edge(t,s,INFF);

if(sum\_in!=sum\_out||Maxflow::maxflow(S,T,n+2)!=sum\_in)

printf("please go home to sleep\n");

else{

ll ans=Maxflow::edg[Maxflow::tot-1].cap;

for(int i=now;i<Maxflow::tot;i++)

Maxflow::edg[i].cap=0;

ans-=Maxflow::maxflow(t,s,n+2);

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

}

}

}

**例4-SGU176-有源汇最小流并输出各边流量**

int n,m;

int deg[MAX];

int an[MAX],pre[MAX],now[MAX];

int main(){

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

Maxflow::init(n+2);

memset(an,0,sizeof(an));

memset(deg,0,sizeof(deg));

for(int u,v,z,c,i=1;i<=m;i++){

read(u);read(v);read(z);read(c);

if(!c)Maxflow::add\_edge(u,v,z,i);

else{

deg[u]-=z;deg[v]+=z;

an[i]=z;

}

}

int ans=0;

int sum\_in=0,sum\_out=0;

int s=n+1,t=n+2;

int now=Maxflow::tot;

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

if(deg[i]>0)sum\_in+=deg[i],Maxflow::add\_edge(s,i,deg[i]);

else if(deg[i]<0)sum\_out-=deg[i],Maxflow::add\_edge(i,t,-deg[i]);

}

//

int tmp=Maxflow::maxflow(s,t,n+2);

Maxflow::add\_edge(n,1,INF);

if(sum\_in!=sum\_out||tmp+Maxflow::maxflow(s,t,n+2)!=sum\_in)

printf("Impossible\n");

else{

ans=Maxflow::edg[Maxflow::tot-1].cap;

for(int i=0;i<Maxflow::tot;i++)

if(Maxflow::edg[i].id)

an[Maxflow::edg[i].id]=Maxflow::edg[i^1].cap;

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

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

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

}

printf("\n");

}

}

}

**例4-SGU176-有源汇最小流并输出各边流量-方法2**

int n,m;

int deg[MAX];

int an[MAX],pre[MAX],now[MAX];

int main(){

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

Maxflow::init(n+2);

memset(an,0,sizeof(an));

memset(deg,0,sizeof(deg));

for(int u,v,z,c,i=1;i<=m;i++){

read(u);read(v);read(z);read(c);

if(!c)Maxflow::add\_edge(u,v,z,i);

else{

deg[u]-=z;deg[v]+=z;

an[i]=z;

}

}

int ans;

ans=-Maxflow::maxflow(n,1,n+2);

int sum\_in=0,sum\_out=0;

int s=n+1,t=n+2;

int now=Maxflow::tot;

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

if(deg[i]>0)sum\_in+=deg[i],Maxflow::add\_edge(s,i,deg[i]);

else if(deg[i]<0)sum\_out-=deg[i],Maxflow::add\_edge(i,t,-deg[i]);

}

Maxflow::add\_edge(n,1,INF);

if(sum\_in!=sum\_out||Maxflow::maxflow(s,t,n+2)!=sum\_in)

printf("Impossible\n");

else{

ans+=Maxflow::edg[Maxflow::tot-1].cap;

for(int i=now;i<Maxflow::tot;i++)

Maxflow::edg[i].cap=0;

ans-=Maxflow::maxflow(n,1,n+2);

if(ans<0){

Maxflow::add\_edge(s,1,-ans);

Maxflow::maxflow(s,n,n+2);

ans=0;

}

for(int i=0;i<Maxflow::tot;i++)

if(Maxflow::edg[i].id)

an[Maxflow::edg[i].id]=Maxflow::edg[i^1].cap;

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

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

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

}

printf("\n");

}

}

}

## 最小割

**例1-判断最小割是否唯一-bzoj3258**

int t,n,m;

int ecnt1,ecnt2,eid,dcnt;

int head1[MAX],head2[MAX],a[MAX];

int lev[MAX],cur[MAX];

int dfn[MAX],low[MAX],scc,bel[MAX];

ll dis[2][MAX];

struct node{

int to,nxt,val;

node(){}

node(int \_to,int \_nxt,int \_val):to(\_to),nxt(\_nxt),val(\_val){}

}edg1[MAX],edg2[MAX];

inline void add\_edge1(int u,int v,int w){

edg1[++ecnt1]=node(v,head1[u],w);

head1[u]=ecnt1;

}

inline void add\_edge2(int u,int v,int w){

edg2[++ecnt2]=node(v,head2[u],w);

head2[u]=ecnt2;

edg2[++ecnt2]=node(u,head2[v],0);

head2[v]=ecnt2;

}

void dij(int st){

priority\_queue<pli,vector<pli>,greater<pli> >que;

int id=(st==1?0:1);

memset(dis[id],0x3f,sizeof(dis[id]));

que.push(mp(0,st));dis[id][st]=0;

while(!que.empty()){

int who=que.top().se;

ll d=que.top().fi;que.pop();

if(dis[id][who]<d)continue;

for(int i=head1[who];i;i=edg1[i].nxt){

int to=edg1[i].to,val=edg1[i].val;

if(dis[id][to]>dis[id][who]+val){

dis[id][to]=dis[id][who]+val;

que.push(mp(dis[id][to],to));

}

}

}

}

inline bool bfs(){

queue<int>que;

memset(lev,0,sizeof(lev));

lev[1]=1;que.push(1);

while(!que.empty()){

int x=que.front();que.pop();

for(int i=head2[x];i;i=edg2[i].nxt){

int to=edg2[i].to;

if(lev[to]||!edg2[i].val)continue;

lev[to]=lev[x]+1;

if(to==n)return 1;

que.push(to);

}

}

return 0;

}

inline int dinic(int x,int low){

if(x==n)return low;

int tmp=low;

for(int &i=cur[x];i;i=edg2[i].nxt){

int to=edg2[i].to;

if(lev[to]!=lev[x]+1||!edg2[i].val)continue;

int res=dinic(to,min(tmp,edg2[i].val));

if(!res)lev[to]=0;

else tmp-=res,edg2[i].val-=res,edg2[i^1].val+=res;

if(!tmp)return low;

}

return low-tmp;

}

stack<int>stk;

bool inq[MAX];

void tarjan(int x){

dfn[x]=low[x]=++dcnt;

stk.push(x);inq[x]=1;

for(int i=head2[x];i;i=edg2[i].nxt){

if(!edg2[i].val)continue;

int to=edg2[i].to;

if(!dfn[to])tarjan(to),low[x]=min(low[x],low[to]);

else if(inq[to])low[x]=min(low[x],dfn[to]);

}

if(low[x]==dfn[x]){

++scc;

while(1){

int y=stk.top();stk.pop();

inq[y]=0;bel[y]=scc;

if(x==y)break;

}

}

}

int main(){

read(t);

while(t--){

read(n);read(m);

memset(head1,0,sizeof(head1));

memset(head2,0,sizeof(head2));

ecnt1=0;ecnt2=1;

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

a[n]=INFF;

while(m--){

int u,v,w;

read(u);read(v);read(w);

add\_edge1(u,v,w);add\_edge1(v,u,w);

}

dij(1);dij(n);eid=n;

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

for(int j=head1[i];j;j=edg1[j].nxt){

int to=edg1[j].to,val=edg1[j].val;

if(dis[0][i]+val+dis[1][to]==dis[0][n]){

++eid;

add\_edge2(i,eid,a[i]);

add\_edge2(eid,to,a[to]);

}

}

ll ans=0;

while(bfs()){memcpy(cur,head2,sizeof(head2));ans+=dinic(1,INF); }

memset(dfn,0,sizeof(dfn));

memset(inq,0,sizeof(inq));

dcnt=scc=0;

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

if(!dfn[i])tarjan(i);

bool flag=0;

for(int i=2;i<=ecnt2&&!flag;i+=2){

if(edg2[i].val)continue;//没有满流

int x=edg2[i^1].to,y=edg2[i].to;

if(bel[x]==bel[y])continue;

if(bel[x]!=bel[1]||bel[y]!=bel[n])flag=1;

}

if(flag)printf("No %lld\n",ans);

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

}

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());

}