

目录

杂物.....	3	线性基(套路).....	45
平板电视.....	4	图论.....	46
Dancing Links.....	5	二分图匹配.....	46
一点 DP 的.....	8	最短路.....	47
决策单调性优化.....	8	差分约束系统.....	48
斜率优化.....	9	01 分数规划.....	49
四边形不等式优化.....	9	最小生成树(曼哈顿在后面).....	50
数位 DP.....	10	强连通分量 tarjin.....	50
插头 DP.....	10	支配树.....	51
字符串的.....	12	网络流.....	52
KMP 最小表示法.....	12	最小费用流.....	54
字典树.....	12	上下界网络流.....	55
AC 自动机.....	13	上下界费用流.....	56
后缀数组.....	14	树分治.....	58
后缀自动机.....	15	部分树上 dp.....	59
后缀自动机+主席树合并.....	17	2-sat.....	59
马拉车.....	18	dfs 序.....	60
回文自动机.....	20	dfs 序_换根的讨论 233.....	60
二分 hash.....	21	树链剖分.....	62
一些 hashset hashmap.....	22	树链剖分求 LCA.....	63
后缀平衡树.....	22	离线 tarjin 求 LCA.....	63
数据结构.....	25	倍增.....	64
按秩合并并查集(+整体二分).....	25	最大团.....	64
二维树状数组.....	25	最小树形图.....	65
树状数组 不大于 k 的最大值.....	26	一般图最大匹配 带花树.....	65
BIT_差分.....	26	数学相关.....	67
二维线段树.....	26	逆元.....	67
扫描线 矩形周长并.....	28	博弈:NIM,SG.....	67
主席树.....	29	FFT、NTT、FWT、FMT.....	67
区间不重复数字个数和第 k 个是哪位.....	29	子集卷积(的分治套路).....	69
可持久化数组(主席树维护).....	30	高斯消元.....	70
树套树.....	31	矩阵树定理 拉格朗日插值.....	71
CDQ 分治(套线段树).....	32	Polya 定理 Burnside 引理.....	71
SPLAY.....	33	Miller_Rabin 素性测试+pollard_rho 因数分解.....	72
SPLAY 启发式合并.....	35	中国剩余定理(不一定互质).....	73
LCT.....	37	Prime-counting function.....	73
KD 树.....	39	欧拉降幂公式.....	74
莫队.....	41	其他的東西.....	75
树上莫队(套分块).....	41	自适应simpson 积分.....	75
回滚莫队套分块.....	42	(我全都不会的)积性函数.....	75
带修改莫队.....	43	超麻烦的类欧几里得.....	77
维护凸包.....	44	不会的计算几何.....	78
		求凸包(旋转卡壳,最近最远点对).....	78

板子 ???

```
#pragma comment(linker, "/STACK:102400000,102400000")
#include <cstdio>
#include <iostream>
#include <algorithm>
#include <vector>
#include <set>
#include <map>
#include <string>
#include <cstring>
#include <stack>
#include <queue>
#include <cmath>
#include <ctime>
#include <utility>
#include <cassert>
using namespace std;
#define REP(l,N) for (l=0;l<N;l++)
#define rREP(l,N) for (l=N-1;l>=0;l--)
#define rep(l,S,N) for (l=S;l<N;l++)
#define rrep(l,S,N) for (l=N-1;l>=S;l--)
#define FOR(l,S,N) for (l=S;l<=N;l++)
#define rFOR(l,S,N) for (l=N;l>=S;l--)
#define dbg(x) cout <<#x<<" = "<<x<<" ;  "
#define dbgln(x) cout <<#x<<" = "<<x<<endl
typedef unsigned long long ULL;
typedef long long LL;
const int INF=0x3f3f3f3f;
const LL INFF=0x3f3f3f3f3f3f3fll;
const LL M=1e9+7;
const LL maxn=1e6+7;
const double eps=0.00000001;
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<typename T>inline T powMM(T a, T b) {
    T ret=1;
    for (; b>=1ll,a=(LL)a*a%M)
        if (b&1) ret=(LL)ret*a%M;
    return ret;
}
```

头文件在上面~~~~~

杂物

首先是没啥用的两个板子

```

void msort(int le,int ri) { //逆序对
    if (le==ri) return;
    int mid=(le+ri)>>1,i=le,j=mid+1,k=i;
    msort(le,mid); msort(j,ri);
    while (i<=mid||j<=ri) {
        if (i==mid+1) {b[k++]=a[j++]; ans+=mid-i+1;}
        else if (j==ri+1) b[k++]=a[i++];
        else if (a[i]<=a[j]) b[k++]=a[i++];
        else {b[k++]=a[j++]; ans+=mid-i+1;}
    }
    for (i=le; i<=ri; i++) a[i]=b[i];
}

void fqsrt(int l,int r) { //O(n)第 k 大数
    int le=l,ri=r,m;
    m=a[le];
    while (le<ri) {
        while (le<ri&& a[ri]<=m) ri--;
        a[le]=a[ri];
        while (le<ri&& a[le]>=m) le++;
        a[ri]=a[le];
    }
    if (le==k) printf("%d\n",m);
    else if (le>k) fqsrt(l,le-1);
    else fqsrt(le+1,r);
}

```

普通输入挂

```

template<class T>
bool read_d(T &num) {
    char in; bool lsN=false;
    in=getchar();
    if (in==EOF) return false;
    while (in!= '-'&&(in<'0'||in>'9')) in=getchar();
    if (in=='-') {lsN=1; num=0;}
    else num=in-'0';
    while (in=getchar(),in>='0'&&in<='9')
        num=num*10+in-'0';
    if (lsN) num=-num;
    return 1;
}

template<class T>

```

```

bool read_f(T &num) {
    char in; bool lsN=false,lsD=false;
    T Dec=0.1;
    in=getchar();
    if (in==EOF) return false;
    while (in!='-'&&in!='.'&&(in<'0'||in>'9'))
        in=getchar();
    if (in=='-') {lsN=1; num=0;}
    else if (in=='.') {lsD=1; num=0;}
    else num=in-'0';
    if (!lsD) while (in=getchar(),in>='0'&&in<='9')
        num=num*10+in-'0';
    if (in=='.') while (in=getchar(),in>='0'&&in<='9')
        {num+=Dec*(in-'0'); Dec*=0.1;}
    if (lsN) num=-num;
    return 1;
}

```

fread 输入挂(namespace 的就是 fread==)

```

char buffer[36000000],*buf=buffer;
char write[7000000],*ed=write;
void read(int &x){
    for(x=0;*buf<48;++buf);
    while(*buf>=48)x=x*10+*buf-48,++buf;
}

void read(int &x){
    for(x=0;(*buf<'0'||*buf>'9')&&*buf!='-';++buf);
    int flag=0;if (*buf=='-') flag=1,buf++;
    while('0'<=*buf&&*buf<='9')
        x=x*10+*buf-48,++buf;
    if (flag) x=-x;
}

int pp[20];
void print(LL x){
    if (!x) *ed++='0';
    else {
        int now=0,i;
        while (x) pp[now++]=x%10,x/=10;
        while (now) *ed++=pp[--now]+48;
    }
    *ed++='\n';
}

fread(buffer,1,36000000,stdin);
fwrite(write,1,ed-write,stdout);

```

其他挂**扩栈**

```
#ifndef OPENSTACK
    int size = 256 << 20; // 256MB
    char *p = (char*)malloc(size) + size;
    #if (defined _WIN64) or (defined __unix)
        __asm__("movq %0, %%rsp\n" :: "r"(p));
    #else
        __asm__("movl %0, %%esp\n" :: "r"(p));
    #endif
#endif
注意最后加 exit(0);
```

玄学加速挂

```
#pragma comment(linker, "/stack:200000000")
#pragma GCC optimize("Ofast,no-stack-protector")
#pragma GCC
target("sse,sse2,sse3,ssse3,sse4,popcnt,abm,mmx,avx,tune=native")
```

然后加上并行计算(计组)

```
#pragma GCC optimize("Ofast,no-stack-protector")
#pragma GCC target("avx")
```

普通 C(n,n)DP

```
int i,j;
C[0][1]=C[1][1]=1;
FOR(j,2,Cmax) {
    C[0][j]=1;
    FOR(i,1,j) C[i][j]=(C[i-1][j-1]+C[i][j-1])%M;
}
```

平板电视**1、红黑树**

```
#include<cstdio>
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_cxx;
using namespace __gnu_pbds;
typedef
```

```
tree<int,null_type,less<int>,rb_tree_tag,tree_order_statistics_node_update> rbtree;
/*
定义一颗红黑树
int 关键字类型
null_type 无映射(低版本 g++ 为 null_mapped_type)
less<int> 从小到大排序
rb_tree_tag 红黑树 (splay_tree_tag)
tree_order_statistics_node_update 结点更新
插入 t.insert();
删除 t.erase();
Rank:t.order_of_key();
第 K 值:t.find_by_order();
前驱:t.lower_bound();
后继 t.upper_bound();
a.join(b) b 并入 a 前提是两棵树的 key 的取值范围不相交
a.split(v,b) key 小于等于 v 的元素属于 a, 其余的属于 b
T.lower_bound(x) >=x 的 min 的迭代器
T.upper_bound(x) >x 的 min 的迭代器
T.find_by_order(k) 有 k 个数比它小的数
*/
rbtree T;
rbtree::iterator it;
```

2、Rope

```
#include<ext/rope>
using namespace std;
using namespace __gnu_cxx;
/*
1) 运算符: rope 支持 operator += -= + - < ==
2) 输入输出: 可以用<<运算符由输入输出流读入或输出。
3) 长度/大小: 调用 length(), size()都可以哦
4) 插入/添加等:
append(const string&)
substr(start,length)
push_back(x); //在末尾添加 x
insert(pos,x); //在 pos 插入 x, 自然支持整个 char 数组的一次插入
erase(pos,x); //从 pos 开始删除 x 个
copy(pos,len,x); //从 pos 开始到 pos+len 为止用 x 代替
replace(pos,x); //从 pos 开始换成 x
substr(pos,x); //提取 pos 开始 x 个
at(x)/[x]; //访问第 x 个元素
```

板子 ???

```
*/
rope<int> V;

3、二项堆(这里是 dijkstra)
#include<iostream>
#include<cstdio>
#include<cstring>
#include<ext/pb_ds/priority_queue.hpp>
#define ll long long
#define pa pair<ll,int>
#define llinf 9000000000000000000LL
using namespace std;
using namespace __gnu_pbds;
typedef
__gnu_pbds::priority_queue<pa,greater<pa>,pairing_h
eap_tag > heap;
int n,m,cnt,last[1000005];
int T,rx,rc,rya,ryc,rp;
heap::point_iterator id[1000005];
int x,y,z;
ll dis[1000005];
struct data {int to,next,v;} e[1000005];
inline int read() {
    int x=0,f=1; char ch=getchar();
    while (ch<'0' || ch>'9') {if (ch=='-')f=-1;
ch=getchar();}
    while (ch>='0' && ch<='9') {x=x*10+ch-'0';
ch=getchar();}
    return x*f;
}
void insert(int u,int v,int w) {
    e[++cnt].to=v; e[cnt].next=last[u]; last[u]=cnt;
e[cnt].v=w;
}
void dijkstra() {
    heap q;
    for (int i=1; i<=n; i++)dis[i]=llinf;
    dis[1]=0; id[1]=q.push(make_pair(0,1));
    while (!q.empty()) {
        int now=q.top().second; q.pop();
        for (int i=last[now]; i; i=e[i].next)
            if (e[i].v+dis[now]<dis[e[i].to]) {
                dis[e[i].to]=e[i].v+dis[now];
                if (id[e[i].to]!=0)
```

```
q.modify(id[e[i].to],make_pair(dis[e[i].to],e[i].to));
                else
                    id[e[i].to]=q.push(make_pair(dis[e[i].to],e[i].to));
            }
        }
    }
}
int main() {
    n=read(); m=read();
    T=read(); rx=read(); rc=read(); rya=read();
    ryc=read(); rp=read();
    int a,b;
    for (int i=1; i<=T; i++) {
        x=((ll)x*rx+rc)%rp;
        y=((ll)y*rya+ryc)%rp;
        a=min(x%n+1,y%n+1);
        b=max(y%n+1,y%n+1);
        insert(a,b,100000000-100*a);
    }
    for (int i=1; i<=m-T; i++) {
        x=read(),y=read(),z=read();
        insert(x,y,z);
    }
    dijkstra();
    printf("%lld",dis[n]);
    return 0;
}
```

Dancing Links

1、不可重复

//数独

```
struct DLX{
    const static int maxn=1e5+7;
    const static int maxd=1e4+7;
    int n,m,size;
    int
    U[maxn],D[maxn],R[maxn],L[maxn],col[maxn],row[max
n];

    int H[maxd],S[maxd]; //S:cnt
    int ans[maxn];
    void init(int _n,int _m){
        n=_n;m=_m;int i;
        FOR(i,0,m) {
            S[i]=0;
            U[i]=D[i]=i;
```

```

        L[i]=i-1,R[i]=i+1;
    }R[m]=0;L[0]=m;
    size=m;
    FOR(i,0,n) H[i]=-1;
}
void link(int r,int c){
    S[col[++size]=c]++;row[size]=r;
    D[size]=D[c];U[D[c]]=size;
    D[c]=size;U[size]=c;
    if (H[r]<0) H[r]=L[size]=R[size]=size;
    else{
        R[size]=R[H[r]];
        L[R[H[r]]]=size;
        L[size]=H[r];
        R[H[r]]=size;
    }
}
void remove(int c){
    L[R[c]]=L[c];R[L[c]]=R[c];
    for (int i=D[c];i!=c;i=D[i])
        for (int j=R[i];j!=i;j=R[j])
            U[D[j]]=U[i],D[U[j]]=D[j],S[col[j]]--;
}
void resume(int c){
    for (int i=U[c];i!=c;i=U[i])
        for (int j=L[i];j!=i;j=L[j])
            U[D[j]]=D[U[j]]=j,S[col[j]]++;
    L[R[c]]=R[L[c]]=c;
}
char g[maxn];
bool dance(int pos){
    if (R[0]==0) {
        int i,j;
        REP(i,pos)
            g[(ans[i]-1)/16]=(ans[i]-1)%16+'A';
        REP(i,16)
            {REP(j,16) putchar(g[i*16+j]);puts("");}
        return 1;
    }
    int c=R[0];
    for (int i=R[0];i!=R[i])
        if (S[i]<S[c]) c=i;
    remove(c);
    for (int i=D[c];i!=c;i=D[i]){
        ans[pos]=row[i];

```

```

        for (int j=R[i];j!=i;j=R[j]) remove(col[j]);
        if (dance(pos+1)) return 1;
        for (int j=L[i];j!=i;j=L[j]) resume(col[j]);
    }resume(c);
    return 0;
}
}dlx;

char g[27][27];
int n,m;
void add(int x,int y,int k){
    int r=(x*16+y)*16+k;
    dlx.link(r,16*16*0+x*16+y+1);
    dlx.link(r,16*16*1+x*16+k);
    dlx.link(r,16*16*2+y*16+k);
    dlx.link(r,16*16*3+(x/4*4+y/4)*16+k);
}
int main(){
    int i,j,k;
    while (~scanf("%s",g[0])){
        rep(i,1,16) scanf("%s",g[i]);
        dlx.init(16*16*16,16*16*4);
        REP(i,16) REP(j,16) FOR(k,1,16)
            if (g[i][j]=='-'||g[i][j]=='A'-1+k)
                add(i,j,k);
        static int x=0;
        if (x) puts("");else x=1;
        dlx.dance(0);
    }
}

```

2、可重复

//暴力枚举,n 个覆盖 m; 注意一定要 init

```

struct DLX {
    const static int maxn=1e5+7;
    const static int maxd=1e4+7;
    int n,m,size;
    int U[maxn],D[maxn],R[maxn],L[maxn];
    int col[maxn],row[maxn];
    int H[maxd],S[maxd],S[cnt];
    int ans[maxn];
    void init(int _n,int _m) {
        n=_n; m=_m; int i;
        FOR(i,0,m) {
            S[i]=0;
            U[i]=D[i]=i;

```

```

        L[i]=i-1,R[i]=i+1;
    } R[m]=0; L[0]=m;
    size=m;
    FOR(i,0,n) H[i]=-1;
}
void link(int r,int c) {
    S[col[++size]=c]++; row[size]=r;
    D[size]=D[c]; U[D[c]]=size;
    D[c]=size; U[size]=c;
    if (H[r]<0) H[r]=L[size]=R[size]=size;
    else {
        R[size]=R[H[r]];
        L[R[H[r]]]=size;
        L[size]=H[r];
        R[H[r]]=size;
    }
}
void remove(int c) {
    for (int i=D[c]; i!=c; i=D[i])
        L[R[i]]=L[i],R[L[i]]=R[i];
}
void resume(int c) {
    for (int i=U[c]; i!=c; i=U[i])
        L[R[i]]=R[L[i]]=i;
}
bool v[maxd];
int f() {
    //估价函数,如果 max 的话其实可以直接 cnt[R[]]
    int ret=0;
    for (int c=R[0]; c; c=R[c]) v[c]=1;
    for (int c=R[0]; c; c=R[c]) if (v[c]) {
        ret++; v[c]=0;
        for (int i=D[c]; i!=c; i=D[i])
            for (int j=R[i]; j!=i; j=R[j])
                v[col[j]]=0;
    }
    return ret;
}
int cnt;
void dance(int pos) {
    if (pos+f())>=cnt) return;
    if (R[0]==0) {cnt=min(cnt,pos); return;}
    int c=R[0];

```

```

        for (int i=R[0]; i; i=R[i])
            if (S[i]<S[c]) c=i;
        for (int i=D[c]; i!=c; i=D[i]) {
            ans[pos]=row[i];
            remove(i);
            for (int j=R[i]; j!=i; j=R[j]) remove(j);
            dance(pos+1);
            for (int j=L[i]; j!=i; j=L[j]) resume(j);
            resume(i);
        }
    } dlx;
    int n,m;
    int check(int x,int y,int a,int b,double d) {
        return (x-a)*(x-a)+(y-b)*(y-b)<d*d;
    }
    int x1[maxn],x2[maxn],y1[maxn],y2[maxn];
    int main() {
        int T;
        scanf("%d",&T);
        while (T--) {
            int k,i;
            scanf("%d%d%d",&n,&m,&k);
            FOR(i,1,n) scanf("%d%d",&x1[i],&y1[i]);
            FOR(i,1,m) scanf("%d%d",&x2[i],&y2[i]);
            double l=0,r=1500;
            while (r-l>1e-7) {
                int i,j;
                double mid=(l+r)/2;
                dlx.init(m,n);
                FOR(i,1,n)
                    FOR(j,1,m)
                        if (check(x1[i],y1[i],x2[j],y2[j],mid))
                            dlx.link(j,i);
                dlx.cnt=k+1;
                dlx.dance(0);
                if (dlx.cnt>k) l=mid;
                else r=mid;
            } printf("%.6f\n",l);
        }
    }
}

```

一点 DP 的

决策单调性优化

```
//决策单调性优化可以处理所有斜率优化的题
//题意:sum{A[l]->A[k]},{1<=l<r<=n,k 是 l->r 的路径上最近的标记点}
//做法:DP; 注意有时 DP[0]甚至 DP[1]都要预处理的
//注意先写好 DP 方程
//注意 DP 方程上代表的意义!
//我的理解:从左往右来看,如果 l++,那么切的点只会向右移动,xl,xr 是指
//转折点可能出现的位置;
//CDQ 分治,传递下去了解可能存在的区间
//每次更新的是 mid 节点
//bfs,dfs 均可,时间均为 log(莫队不影响,莫队时间可证明 nlogn)
//CF868F 题意:切区间 k 段,每段数字出现个数 sigma{n(n-1)/2}最小的
//个数
LL L1[maxn],L2[maxn],R1[maxn],R2[maxn];//前缀和之和,小技巧
LL getL(int l,int r) { //一个求 l->r 的点到 l 的 sum 和
    return (L2[r]-L2[l])-L1[l]*(r-l);
}
LL getR(int l,int r) {
    return (R2[l]-R2[r])-R1[r]*(r-l);
}

LL pre[maxn],dp[maxn];
struct node {
    int l,r,xl,xr;
};
LL cnt,sum,sum_sum;
queue<node> Q;
void changel(LL val,int seg) {
    sum_sum+=sum*seg*2;
    sum_sum-=cnt*val*seg*2;
    cnt+=seg; sum+=val*seg;
}
void changer(LL val,int seg) {
    sum_sum-=sum*seg*2;
    sum_sum+=cnt*val*seg*2;
    cnt+=seg; sum+=val*seg;
}
int _l,_r;
LL A[maxn];
void changeto(int l,int r) {
```

```
while (_r<r) _r++,changer(A[_r],1);
while (_l>l) _l--,changer(A[_l],1);
while (_l<l) changer(A[_l],-1),_l++;
while (_r>r) changer(A[_r],-1),_r--;
}

void solve(int n) {
    int i;
    Q.push(node{1,n,0,n-1});
    while (Q.size()) {
        auto F=Q.front(); Q.pop();
        int l=F.l,r=F.r,L=F.xl,R=F.xr;//l,r,check_l,check_r
        int m=(l+r)/2,M=L;
        LL &now=dp[m];
        FOR(i,L,min(m-1,R)) {
            //这里 changeto 不会改变复杂度
            LL msum=(m-i)*getL(m,n);
            LL rsum=(n-m+1)*(getR(i+1,m)+i*(A[m]-A[i]));
            if (now>pre[i]-msum-rsum)
                now=pre[i]-msum-rsum,M=i;
        }
        if (l<m) Q.push(node{l,m-1,L,M});
        if (r>m) Q.push(node{m+1,r,M,R});
    }
}

//DP[i]:i_chosen; contains [i]->[i]; [i]->R(i+1->n)
//update:m [i-m]->[i], [i-m]->[m-n] [i-m]->[i-m]
int T;
int n,m,k;
int i,j;
int main() {
    while (~scanf("%d%d",&n,&k)) {
        FOR(i,1,n) scanf("%lld",&A[i]);
        A[0]=A[1]; A[n+1]=A[n];
        FOR(i,1,n) L1[i]=A[i]-A[i-1]+L1[i-1];
        FOR(i,1,n) L2[i]=L2[i-1]+L1[i];

        rFOR(i,1,n) R1[i]=A[i+1]-A[i]+R1[i+1];
        rFOR(i,1,n) R2[i]=R2[i+1]+R1[i];

        _l=1; _r=0; sum=sum_sum=cnt=0;
        changeto(1,n);
        FOR(i,0,n) dp[i]=sum_sum;
```



```
// FOR(i,1,n) printf("%lld ",dp[i]);puts(" <- start_DP");
    FOR(i,1,k) {
        int i;
        FOR(i,0,n) pre[i]=dp[i];
        solve(n);
//        FOR(m,1,n) FOR(i,0,m-1){
////            changeto(i+1,m);
////            cal:=-[m,n]->[i](differ)+[i+1-m](to m)
////            cal:=-[j+1,m]->[m,n](to m)
//            LL msum=(m-i)*getL(m,n);
//            LL rsum=(n-m+1)*(getR(i+1,m)+i*(A[m]-A[i]));
//            dp[m]=min(dp[m],pre[i]-msum-rsum);
//        }
//        FOR(i,1,n) printf("%lld ",dp[i]);puts(" <- DP");
    }
    LL ans=dp[0];
    FOR(i,1,n) ans=min(ans,dp[i]);
    printf("%lld\n",ans);
}
```

斜率优化

//HDU 3480//斜率优化

//题意:一堆数字,切成 k 份,每块的代价为(max-min)^2

//dp 方程:dp[i][j]=min{dp[k][j-1]+(a[i]-a[k+1])^2};

//dp 方程:

//dp[i][j]=min{dp[k][j-1]+a[k+1]^2-2*a[i]*a[k+1]}+a[i]^2

//k=(dp[k][j-1](pre)+a[k+1]^2)/(a[k+1]),常数 2*a[i]

//斜率优化本质是维护一个下凸壳

```
int n,m,i,j,k,t;
int a[maxn],pre[maxn],dp[maxn];
int head,tail;
int Q[maxn];//id
inline int getY(int id){
    return pre[id]+a[id+1]*a[id+1];
}
inline int getX(int id){
    return a[id+1];
}
int main(){
    int T,X=0;
    scanf("%d",&T);
    while (T--){
```

```
scanf("%d%d",&n,&m);
    FOR(i,1,n) scanf("%d",&a[i]);
    sort(a+1,a+1+n);
    int qi,qj,qk;
    FOR(i,1,n) dp[i]=(a[i]-a[1])*(a[i]-a[1]);
    FOR(j,2,m){
        FOR(i,1,n) pre[i]=dp[i];
        head=tail=0;
        dp[0]=0;Q[tail++]=0;
        FOR(i,1,n){
            while (head+1<tail){
                qi=Q[head],qj=Q[head+1];
                if (getY(qj)-getY(qi)<=2*a[i]*(getX(qj)-getX(qi)))
                    head++;
                else break;
            }qi=Q[head];
            dp[i]=pre[qi]+(a[i]-a[qi+1])*(a[i]-a[qi+1]);
            while (head+1<tail){
                qi=Q[tail-2],qj=Q[tail-1];qk=i;
                int y1=getY(qj)-getY(qi),x1=getX(qj)-getX(qi);
                int y2=getY(qk)-getY(qj),x2=getX(qk)-getX(qj);
                if (y2*x1<=y1*x2) tail--;//y2/x2>y1/x1
                else break;
            }Q[tail++]=i;
        }
    }
    printf("Case %d: %d\n",++X,dp[n]);
}
```

四边形不等式优化

//HDU 3516//四边形不等式优化

//题意:给定一个从左上往右下的图,只能往下往右连,求一个构造使得所有的边长度总和最小

//dp 方程:

//dp[i][j]=max{dp[i][k]+dp[k+1][j]+x[k+1]-x[i]+y[k]-y[j]};

//能用 : 满足:

//w[i][j]+w[i'][j']<=w[i][j']+w[i'][j];

//w[i'][j']<=w[i][j],那么决策区间包含

```
struct node{
    int x,y;
}a[maxn];
int n,m,i,j,k,t;
```

板子 ???

```
int dp[maxn][maxn],pos[maxn][maxn];
int main(){
    while (~scanf("%d",&n)){
        FOR(i,1,n) scanf("%d%d",&a[i].x,&a[i].y),pos[i][i]=i;
        FOR(i,1,n) FOR(j,i+1,n) dp[i][j]=INF;
        FOR(t,1,n-1){
            FOR(i,1,n-t){
                j=i+t;
                FOR(k,pos[i][j]-1,min(j-1,pos[j+1][j])){
                    int now=dp[i][k]+dp[k+1][j]+a[k+1].x-
a[i].x+a[k].y-a[j].y;
                    if (dp[i][j]>now){
                        dp[i][j]=now;
                        pos[i][j]=k;
                    }
                }
            }
        }
        printf("%d\n",dp[1][n]);
    }
}
```

数位 DP

//当板子了

//这道题是连续的差最大是 1

//需要注意时间空间限制,有时需要 hash

//注意取模时底下 calc 也要取 - -

LL f[27][17][2];

int value[27];

LL calc(int x,int prev,int not_0,int flag) {

if (x==0) return 1;

if (!flag&&f[x][prev][not_0]!=-1)

return f[x][prev][not_0];

LL ret=0; int i,maxi=9;

if (flag) maxi=min(maxi,value[x]);

FOR(i,0,maxi) {

// if (not_0||i)//这是与 lead_0 有关的写法

if (not_0&&abs(prev-i)<2) continue;

else ret+=calc(x-1,i,not_0||i,flag&&(i==maxi));

} if (!flag) f[x][prev][not_0]=ret;

return ret;

}LL calc(LL x) {

int length=0;

while (x) value[++length]=x%10,x/=10;

```
    return calc(length,0,0,1);
} LL calc(LL l,LL r) {
    return calc(r)-calc(l-1);
}
int n,m;
int i,j;
int T;
int main() {
    memset(f,0xff,sizeof(f));
    FOR(i,1,10000)
        if (calc(i,i)) printf("%d ",i);
    puts("");
    LL l,r;
    scanf("%lld%lld",&l,&r);
    printf("%lld\n",calc(l,r));
}
```

插头 DP

没什么可说的, 不会写

template<typename T1,typename T2> struct hashmap {

const static int seed=9999991;

const static int maxn=1e6+7;

struct node {

T1 key; T2 val; int next;

node() {};

node(T1 k,T2 v,int n):key(k),val(v),next(n) {};

} T[maxn]; //更好地空间局部性?(雾)

int head[seed],size;

void clear() {

memset(head,-1,sizeof(head));

size=0;

}

void insert(T1 pos,T2 val) {

int x=pos%seed;

T[size]=node(pos,val,head[x]);

head[x]=size++;

}

T2 &operator [](T1 x) {

for (int i=head[x%seed]; ~i; i=T[i].next)

if (T[i].key==x) return T[i].val;

insert(x,0);

return T[size-1].val;

}

};

板子 ???

```

hashmap<int,LL> MP[2];
int T;
inline int getpos(int x,int k) {
    return (x>>(k+k))&3;
} inline int setpos(int x,int k,int v) {
    return (x&~(3<<(k+k)))(v<<(k+k));
} inline void remark(int k) {
    static int val[7];
    memset(val,0xff,sizeof(val));
}
char A[27][27];
int ex,ey;//012:#()
int main() {
    T=1;
    while (T--) {
        int n,m;
        int i,j,k;
        scanf("%d%d",&n,&m);
        FOR(i,1,n) scanf("%s",A[i]+1);
        FOR(i,1,n) FOR(j,1,m) if (A[i][j]=='.' ) ex=i,ey=j;
        int now=0,nxt=1;
        MP[now].clear(); MP[now].insert(0,1);
        FOR(i,1,n) {
            FOR(j,1,m) {
                MP[nxt].clear();
                for (int it=0; it<MP[now].size; it++) {
                    int k=MP[now].T[it].key; LL w=MP[now].T[it].val;
                    int L=getpos(k,j-1),U=getpos(k,j);
                    if (A[i][j]=='*') {//update0
                        if (!L&&!U) MP[nxt][k]+=w;
                    } else if (A[i][j]) {//update1
                        if (!L&&!U) {
                            int K=setpos(k,j-1,1);
                            K=setpos(K,j,2);
                            MP[nxt][K]+=w;
                        } else if ((!L)^(!U)) {
                            int K=setpos(k,j-1,U);
                            K=setpos(K,j,L);
                            MP[nxt][K]+=w;
                            MP[nxt][k]+=w;

```

```

} else if (L&&U) {
    int K=setpos(k,j-1,0);
    K=setpos(K,j,0);
    if (L!=U) {
        if (L==2||(!ex&&j==ey))
            MP[nxt][K]+=w;
    } else {
        if (L==1) {
            int cnt=1;
            for (int l=j+1; l<=m; l++) {
                int x=getpos(K,l);
                if (x==1) cnt++;
                if (x==2) cnt--;
                if (!cnt) {K=setpos(K,l,1); break;}
            } MP[nxt][K]+=w;
        } else if (L==2) {
            int cnt=-1;
            for (int l=j-2; l>=0; l--) {
                int x=getpos(K,l);
                if (x==1) cnt++;
                if (x==2) cnt--;
                if (!cnt) {K=setpos(K,l,2); break;}
            } MP[nxt][K]+=w;
        }
    }
}
}
}
} now^=1; nxt^=1;
} //shift
MP[nxt].clear();
for (int it=0; it<MP[now].size; it++) {
    int k=MP[now].T[it].key; LL w=MP[now].T[it].val;
    if (!getpos(k,m)) MP[nxt][k<<2]+=w;
}
now^=1; nxt^=1;
} static int x;
printf("%lld",MP[now][0]);
}
}

```

字符串的

KMP|最小表示法

```
//记得 border 是个等差数列
int fail[maxn];
int check(char a[],int n){
    fail[0]=fail[1]=0;
    int i,j;
    FOR(i,2,n){
        j=fail[i-1];
        while (j&& a[j+1]!=a[i]) j=fail[j];
        if (a[j+1]==a[i]) fail[i]=j+1;
        else fail[i]=0;
    }if (n%(n-fail[n])==0) return n/(n-fail[n]);
    return 1;
}
//最小表示暴力法
int getmin(char a[],int n){//1-start
    int i,j,l;
    FOR(i,1,n) a[i+n]=a[i];
    i=1,j=2;
    while (i<=n&&j<=n){
        REP(l,n) if (a[i+l]!=a[j+l]) break;
        if (l==n) break;
        if (a[i+l]>a[j+l]) swap(i,j);
        j=max(j+l+1,i+1);
    }return i;
}
int n,m;
int i,j,k;
char a[maxn],b[maxn];
int main(){
    while (~scanf("%s",a+1)){
        n=strlen(a+1);
        int now=getmin(a,n);
        printf("%d %d ",now,check(a+now-1,n));
        FOR(i,1,n) a[i]=-a[i];
        now=getmin(a,n);
        printf("%d %d\n",now,check(a+now-1,n));
    }
}
```

字典树

```
//dfs 格式
bool okay[maxn*20];
bool mark[maxn*20];
int nxt[maxn*20][2];
bool rev[21];
int cnt;
void insert(int x,int now,int dep){
    bool mark=x&(1<<dep);
    if (dep== -1) {okay[now]=1; return;}
    if (!nxt[now][mark]) nxt[now][mark]=++cnt;
    insert(x,nxt[now][mark],dep-1);
    okay[now]=okay[nxt[now][0]]&&okay[nxt[now][1]];
}
void reverse(int x,int dep){
    if (dep== -1) return;
    bool mark=x&(1<<dep);
    rev[dep]^=mark;
    reverse(x,dep-1);
}
int query(int x,int now,int dep){
    if (now==0||dep== -1) return x;
    if (rev[dep]^mark[now])
        swap(nxt[now][0],nxt[now][1]),mark[now]^=1;
    if (okay[nxt[now][0]])
        return query(x,(1<<dep),nxt[now][1],dep-1);
    return query(x,nxt[now][0],dep-1);
}

//求 i<j<k&&i^j<j^k 的三元组个数
//维护 val //left,right 各一个
int T,n;
LL ans;
int i,j;
int a[maxn*32];
int nxt[maxn*32][2];
LL num[maxn*32],last[maxn*32];
LL sum[maxn][32][2];//只有这位。。。
int cnt,now;
int main() {
```

```

scanf("%d",&T);
while (T--) {
    ans=0;
    cnt=0;
    scanf("%d",&n);
    FOR(i,1,n) scanf("%d",&a[i]);
    FOR(i,1,n) {
        rREP(j,32) {
            int mark=((a[i]&(1<<j))!=0);
            sum[i][j][0]=sum[i-1][j][0];
            sum[i][j][1]=sum[i-1][j][1];
            sum[i][j][mark]++;
        }
    }
    rFOR(i,1,n) {
        now=0;
        rREP(j,32) {
            int mark=((a[i]&(1<<j))!=0);
            if (nxt[now][mark^1])
                ans+=last[nxt[now][mark^1]]
                    -num[nxt[now][mark^1]]*sum[i][j][mark];
            if (!nxt[now][mark]) break;
            now=nxt[now][mark];
        }
        now=0;
        rREP(j,32) {
            int mark=((a[i]&(1<<j))!=0);
            if (!nxt[now][mark]) nxt[now][mark]=++cnt;
            now=nxt[now][mark];
            last[now]+=sum[i-1][j][mark^1]; //这点之前
            num[now]++;
        }
    }
    printf("%lld\n",ans);
    FOR(i,0,cnt) num[i]=last[i]=nxt[i][0]=nxt[i][1]=0;
    FOR(i,1,n) REP(j,32)
        sum[i][j][0]=sum[i][j][1]=0;
}
}

```

AC 自动机

//HDU2896,匹配多串

```
struct ACM {
```

```

int next[505*140][98],fail[505*140],tot;
vector<int> have[maxn];
void init() {tot=0; memset(next[0],0,sizeof(next[0]));}
void insert(char a[],int id) {
    int i,n=strlen(a),p=0;
    REP(i,n) {
        int c=a[i]-33;
        if (!next[p][c]) {
            next[p][c]=++tot;
            have[tot].clear();
            memset(next[tot],0,sizeof(next[tot]));
        }
        p=next[p][c];
    }
    have[p].push_back(id);
} int Q[maxn],front,end;
void buildAC() {
    front=end=0;
    Q[end++]=0;
    while (front!=end) {
        int x=Q[front++],c;
        REP(c,98) if (next[x][c]) {
            if (!x) fail[next[x][c]]=0;
            else {
                int p=fail[x];
                while (p&&!next[p][c]) p=fail[p];
                if (next[p][c])
                    fail[next[x][c]]=next[p][c];
                else fail[next[x][c]]=0;
            }
            Q[end++]=next[x][c];
        }
        for (int v:have[fail[x]])
            have[x].push_back(v);
    }
} void query(char a[],int ans[],int &tot) {
    tot=0;
    int p=0;
    int n=strlen(a),i;
    REP(i,n) {
        int c=a[i]-33;
        while (p&&!next[p][c]) p=fail[p];
        p=next[p][c];
        for (int v:have[p]) ans[tot++]=v;
    }
}

```

```

    }
} AC;
int T,i,j,n,m,total;
char s[maxn];
int ans[505],tot;
int main() {
    while (~scanf("%d",&n)) {
        total=0;
        AC.init();
        REP(i,n) {
            scanf("%s",s);
            AC.insert(s,i+1);
        }
        AC.buildAC();
        scanf("%d",&m);
        REP(i,m) {
            scanf("%s",s);
            AC.query(s,ans,tot);
            if (tot==0) continue;
            printf("web %d:",i+1);
            sort(ans,ans+tot);
            ans[tot]=0;
            REP(j,tot)
                if (ans[j]!=ans[j+1]) printf(" %d",ans[j]);
            puts("");
            total++;
        }
        printf("total: %d\n",total);
    }
}

```

后缀数组

HDU6138,前缀+公共子串

```

int wa[maxn],wb[maxn],wv[maxn],ws1[maxn];
int cmp(int *r,int a,int b,int l) {
    return r[a]==r[b]&&r[a+l]==r[b+l];
}
//sa->pos(后缀排名->pos)
void da(int *r,int *sa,int n,int m) {
    r[n+]=0;//使 rank 从 1 开始(sa[0]=n)
    int i,j,p,*x=wa,*y=wb,*t;
    REP(i,m) ws1[i]=0;//pre-cmp
    REP(i,n) ws1[x[i]=r[i]]++;//r->x
    rep(i,1,m) ws1[i]+=ws1[i-1];

```

```

    rREP(i,n) sa[--ws1[x[i]]]=i;//sort(计数排序)
    for (j=1,p=1; p<n; j<=1,m=p) { //j->2^x
        p=0; rep(i,n-j,n) y[p++]=i; //最后 j 个是不用加(显然)
        REP(i,n) if (sa[i]>=j) y[p++]=sa[i]-j;//后缀顺序
        REP(i,n) wv[i]=x[y[i]];//x+y->wv(由于后缀顺序)
        REP(i,m) ws1[i]=0;
        REP(i,n) ws1[wv[i]]++;
        rep(i,1,m) ws1[i]+=ws1[i-1];
        rREP(i,n) sa[--ws1[wv[i]]]=y[i]);//sort(计数排序)
        t=x,x=y,y=t;
        p=1; x[sa[0]]=0;
        rep(i,1,n) x[sa[i]]=cmp(y,sa[i-1],sa[i],j)?p-1:p++;
    }
}

int rnk[maxn],height[maxn];
void calheight(int *r,int *sa,int n) {
    int i,j,k=0;
    FOR(i,1,n) rnk[sa[i]]=i;
    REP(i,n) {
        if (k) k--;
        j=sa[rnk[i]-1];
        while (r[i+k]==r[j+k]) k++;
        height[rnk[i]]=k;
    }
}

int n,m;
int i,j,k;
char a[maxn];
int s[maxn],st[maxn];
int sa[maxn],id[maxn];
int val[maxn];
int tot,now,ans;
int main() {
    int T;
    scanf("%d",&T);
    while (T--) {
        scanf("%d",&n);
        tot=0;
        FOR(i,1,n) {
            scanf("%s",a);
            int len=strlen(a);
            st[tot]=len;
            REP(j,len) id[tot]=i,s[tot++]=a[j]-'a'+1;
            s[tot++]='z'-'a'+i+1;

```

```

}
s[tot]=0;
da(s,sa,tot,26+n+1);
calheight(s,sa,tot);
now=0;
FOR(i,1,tot) {
    val[i]=max(val[i],now);
    now=min(now,height[i+1]);
    if (st[sa[i]])
        now=max(now,height[i+1]),val[i]=INF;
}
now=0;
rFOR(i,1,tot) {
    val[i]=max(val[i],now);
    now=min(now,height[i]);
    if (st[sa[i]]) {
        now=max(now,height[i]);
        val[i]=max(val[i],st[sa[i]]);
    }
}
scanf("%d",&m);
REP(i,m) {
    int x,y,i;
    scanf("%d%d",&x,&y);
    now=0;
    ans=0;
    FOR(i,1,tot) {
        if (id[sa[i]]==x&&st[sa[i]])
            now=max(now,st[sa[i]]);
        if (id[sa[i]]==y)
            ans=max(ans,min(now,val[i]));
        now=min(now,height[i+1]);
        if (id[sa[i]]==x)
            now=max(now,height[i+1]);
    }
    now=0;
    rFOR(i,1,tot) {
        if (id[sa[i]]==x&&st[sa[i]])
            now=max(now,st[sa[i]]);
        if (id[sa[i]]==y)
            ans=max(ans,min(now,val[i]));
        now=min(now,height[i]);
        if (id[sa[i]]==x)
            now=max(now,height[i]);
    }
}

```

板子???

```

printf("%d\n",ans);
}
FOR(i,1,tot) val[i]=st[i]=0;
}
}

```

后缀自动机

// 1 题意:至少在 k 个子串中出现的子串数量

// 2 题意:sigma{循环后匹配 cnt}

// 这里的 len 不可以直接使用~ 原因是这里的 len 指的是原串

len

// fail 过后,len 是可以直接使用的~ (会 fail 到确定的节点上)

// 这个 fail 的含义是说后缀相同,向前拓展的 val(一个一个拓展

len 差项)

// sam 反向不为拓扑序!注意自己进行拓扑排序

// 更新时注意 len 的限制!(因为更新时可能根本没有考虑前缀

len)

// 注意 nq 在更新时更新时 val 和 q 是相等的,也就是说,维护值
时 nq 要完全和 q 一样

// sum{len[x]-len[fail[x]]}=不同串个数,每个串代表 fail->this
的 len

// 每个串的位置建议存的时候就保留下来~ 要不就有点麻烦了

// 复制出来的虚拟节点在计算次数时不参与计算~

// 也就是说计算相同串个数时,复制出来的只是个虚拟的节点

// query 时在末尾加个 0 可以去掉很多的判断!

// 加空字符时注意 len,这个 len 有两个作用:避免 topo 排错,减少
add 特判

// 加的不是 root,就是个空字符,dfs 的话只能 dfs 一个串!从后往前
递推可行

// 如果是在一颗树上建,那么直接计数排序按 len 排是错的!一定
注意!

// 注意看子串时的重复~

// 小技巧:由于每个节点对应的 len 是一定的,如果想要找 l->r 对
应串可以倍增来找到对应的串

// 用 fail 建后缀树时,压缩路径第一个位置为 pos[i]-len[fail[i]]

// 注意一件事:我这样做是并不能保证 len[fail[i]]!=len 的

// 只有 bfs trie 可以保证,这样来进行按 fail 排序建立后缀树

// dfs trie 的时间复杂度是 trie 叶结点深度和 =_!=证明..直接当
多个

// 只有 bfs 能稳定的保证复杂度,但是好像没人这样卡人

```

struct SAM{
    int next[maxn][26],fail[maxn],len[maxn];
    int cnt,last;
}

```

```

void init(){
    cnt=last=0;fail[0]=-1;len[0]=0;
    memset(next[0],0,sizeof(next[0]));
}
void add(int c){
    int np=++cnt,p=last;
    memset(next[np],0,sizeof(next[np]));
    len[np]=len[p]+1;
    for (;p!=-1&&!next[p][c];p=fail[p]) next[p][c]=np;
    if (p==-1) fail[np]=0;
    else {
        int q=next[p][c];
        if (len[p]+1==len[q]) fail[np]=q;
        else{
            int nq=++cnt;len[nq]=len[p]+1;
            memcpy(next[nq],next[q],sizeof(next[q]));
            fail[nq]=fail[q];
            fail[np]=fail[q]=nq;
            for (;p!=-1&&next[p][c]==q;p=fail[p])
                next[p][c]=nq;
        }
    }
    last=np;
}
// 1:trie 上建树,启发式合并 set
map<int,int> have[maxn];
int Next[maxn][26],Last[maxn],tot;
void add(char a[],int id){
    int n=strlen(a),i,p=0,last=0;
    REP(i,n) {
        int c=a[i]-'a';
        if (Next[p][c]) p=Next[p][c],last=Last[p];
        else add(c),Last[p=Next[p][c]]=++tot;
        have[last][id]++;
    }
}
void merge(map<int,int> &A,map<int,int> &B){
    if (A.size()<B.size()) swap(A,B);
    for (auto now:B) A[now.first]+=now.second;
    B.clear();//delete &B;
}
vector<int> edge[maxn];
LL Ans[maxn];
void DFS(int x,int k){
    for (int v:edge[x]){DFS(v,k);merge(have[x],have[v]);}
}

```

```

    if (have[x].size()>=k) for (auto v:have[x])
        Ans[v.first]+=(LL)v.second*(len[x]-len[fail[x]]);
}
void solve(int k){
    int i;
    FOR(i,0,cnt) edge[i].clear();
    FOR(i,1,cnt) edge[fail[i]].push_back(i);
    DFS(0,k);
}
// 2:在 query 前进行了 cnt[np]++和沿 fail 增加
set<int> A;int CNT[maxn];
LL query(char a[]){
    int i;LL ret=0;
    int n=strlen(a),p=0,l=0;A.clear();
    REP(i,n-1){
        int c=a[i%26]-'a';
        if (next[p][c]) l++,p=next[p][c];
        else {
            while (p!=-1&&!next[p][c]) p=fail[p];
            if (p==-1) p=l=0;
            else l=len[p]+1,p=next[p][c];
        }while (len[fail[p]]>=n) p=fail[p],l=len[p];
        if (l>=n){
            if (A.count(p)) continue;
            A.insert(p);
            ret+=CNT[p];
        }
        // if (l>=n) printf("i=%2d ret=id(%2d); l=%2d;
        +=%d\n",i,p,l,CNT[p]);
    }return ret;
}
void print(){
    int i;
    FOR(i,1,cnt) {
    }
}
char a[maxn];
void dfs(int x=0,int len=0){
    int i;
    for (auto v:have[x])
        printf("%2d(%2d) ",v.first,v.second);
    puts("");
    // printf("%-3d(fail:%-3d,len=%-
    2d):%s\n",x,fail[x],this->len[x],a);
}

```



```

REP(i,26){
    if (next[x][i]){
        a[len]=i+'a';
        dfs(next[x][i],len+1);
        a[len]=0;
    }
}
}
}
sam;
int n,m,T;
int i,j,k;
char a[maxn];
int main(){
    scanf("%d%d",&n,&k);
    sam.init();
    FOR(i,1,n){
        scanf("%s",a);
        sam.add(a,i);
    }sam.solve(k);
    // sam.dfs();sam.print();
    FOR(i,1,n) printf("%16d ",sam.Ans[i]);
}

```

后缀自动机+主席树合并

//查询某串部分在串 l->r 的最大出现次数及位置

//SAM(这个套路)

//做法:求出后缀树然后直接找到对应位置 merge

```

int nxt[maxn][27],pre[maxn],len[maxn];
int CNT,last;
void add(int c) {
    int np=++CNT,p=last;
    len[np]=len[p]+1;
    for (; p&&!nxt[p][c]; p=pre[p]) nxt[p][c]=np;
    if (!p) pre[np]=1;
    else {
        int q=nxt[p][c];
        if (len[p]+1==len[q]) pre[np]=q;
        else {
            int nq=++CNT; len[nq]=len[p]+1;
            memcpy(nxt[nq],nxt[q],sizeof(nxt[q]));
            pre[nq]=pre[q];
            pre[np]=pre[q]=nq;
            for (; p&& nxt[p][c]==q; p=pre[p])
nxt[p][c]=nq;

```

```

    }
    last=np;
}
//segtree
int cnt;
struct node {
    pair<int,int> val;//bigger
    int l,r;
} tree[maxn*25];
int root[maxn];
inline pair<int,int> add(pair<int,int> A,pair<int,int> B) {
    return make_pair(A.first+B.first,A.second);
}
inline pair<int,int> better(pair<int,int> A,pair<int,int> B) {
    if (A.first==B.first) return A.second<B.second?A:B;
    return A.first>B.first?A:B;
}
inline void insert(int &x,int val,int l,int r) {
    if (!x) x=++cnt;
    if (l==r) {
        tree[x].val.first++;
        tree[x].val.second=l;
        return;
    }
    int mid=(l+r)/2;
    if (val<=mid) insert(tree[x].l,val,l,mid);
    else insert(tree[x].r,val,mid+1,r);
    tree[x].val=tree[tree[x].l].val+tree[tree[x].r].val;
}
inline int Merge(int x,int y,int l,int r) {
    if (!x||!y) return x|y;
    int z=++cnt;
    if (l==r) {
        tree[z].val=add(tree[x].val,tree[y].val);
        return z;
    }
    int mid=(l+r)/2;
    tree[z].l=Merge(tree[x].l,tree[y].l,l,mid);
    tree[z].r=Merge(tree[x].r,tree[y].r,mid+1,r);
    tree[z].val=tree[tree[z].l].val+tree[tree[z].r].val;
    return z;
}

```

板子 ???

```
inline pair<int,int> query(int x,int l,int r,int L,int R) {
    if (!x) return make_pair(0,0);
    if (l<=L&&R<=r) return tree[x].val;
    int mid=(L+R)/2;
    pair<int,int> ret=make_pair(0,0);
    if (mid>=l)
        ret=better(ret,query(tree[x].l,l,r,L,mid));
    if (r>mid)
        ret=better(ret,query(tree[x].r,r,l,r,mid+1,R));
    return ret;
}

int father[21][maxn],pos[maxn];//倍增求 father
inline int getfather(int l,int r) {
    int L=(r-l+1),ret=pos[r],i;
    rFOR(i,0,20) if (len[father[i][ret]]>=L)
        ret=father[i][ret];
    return ret;
}

int n,m,q;
int i,j,k;
char s[maxn];
int S[maxn],K[maxn];
int main() {
    scanf("%s",s);
    last=++CNT;
    n=strlen(s);
    REP(i,n) add(s[i]-'a'),pos[i+1]=last;
    add(26);
    scanf("%d",&m);
    FOR(k,1,m) {
        scanf("%s",s);
        n=strlen(s);
        REP(i,n) add(s[i]-'a'),insert(root[last],k,1,m);
        add(26);
    }
    FOR(i,1,CNT) S[len[i]]++;
    FOR(i,1,CNT) S[i]+=S[i-1];
    FOR(i,1,CNT) K[S[len[i]]--]=i;
    rFOR(i,1,CNT) {
        if (pre[K[i]]) root[pre[K[i]]]=
            Merge(root[pre[K[i]]],root[K[i]],1,m);
    }
    FOR(i,1,CNT) father[0][i]=pre[i];
    FOR(j,1,20) FOR(i,1,CNT)
        father[j][i]=father[j-1][father[j-1][i]];//倍增
```

```
scanf("%d",&q);
while (q--) {
    int l,r,pl,pr;
    scanf("%d%d%d%d",&l,&r,&pl,&pr);
    int x=getfather(pl,pr);
    pair<int,int> ans=query(root[x],l,r,1,m);
    if (ans.first==0) printf("%d 0\n",l);
    else printf("%d %d\n",ans.second,ans.first);
}
}
```

马拉车

//p 是每个点为中心的延伸最长回文子串长度, -1 就是原串以这个点为中心的长度

//看到题先去想这种方法, 再说其他方法

```
int n,m;
char s[maxn],str[maxn];
int len1,len2,p[maxn],ans;
void init() {
    ans=0;
    int i;
    str[0]='+';
    str[1]='%';
    REP(i,len1+1) {
        str[i*2+2]=s[i];
        str[i*2+3]='%';
    }
    len2=len1*2+2;
}

// 主要是说已经对称匹配过的不再进行
void manacher() {
    int id=0,mx=0;
    int i;
    FOR(i,1,len2-1) {
        if (mx>i) p[i]=min(p[2*id-i],mx-i);
        else p[i]=1;
        while (str[i+p[i]]==str[i-p[i]]) p[i]++;
        if (p[i]+i>mx) {
            mx=p[i]+i;
            id=i;
        }
    }
}

int main() {
```

```

int i;
while (~scanf("%s",s)) {
    len1=strlen(s);
    init();
    manacher();
    REP(i,len2) ans=max(ans,p[i]);
    printf("%d\n",ans-1);
}
}
//滚动的最长回文子串(写了好久)
int a[maxn];
struct node{
    int left,right;
}tree[maxn*4*8];
int val[maxn*4*8],lazy[maxn*4*8];
void change(int x,int i){
    val[x]=max(val[x],i);
    lazy[x]=max(lazy[x],i);
}
void pushdown(int x){
    if (lazy[x]){
        change(x<<1,lazy[x]);
        change(x<<1|1,lazy[x]);
        lazy[x]=0;
    }
}
void build(int x,int l,int r){
    tree[x].left=l;tree[x].right=r;
    val[x]=lazy[x]=0;
    if (l==r) return;
    int mid=(l+r)/2;
    build(x<<1,l,mid);
    build(x<<1|1,mid+1,r);
}
void update(int x,int l,int r,LL val){
    int L=tree[x].left,R=tree[x].right;
    if (l<=L&&R<=r){
        change(x,val);
        return;
    }
    pushdown(x);
    int mid=(L+R)/2;
    if (mid>=l) update(x<<1,l,r,val);
    if (r>mid) update(x<<1|1,l,r,val);
}

```

板子 ???

```

int query(int x,int pos){
    int L=tree[x].left,R=tree[x].right;
    if (L==R) return val[x];
    pushdown(x);
    int mid=(L+R)/2;
    if (mid>=pos) return query(x<<1,pos);
    return query(x<<1|1,pos);
}
int n,m;
char s[maxn*2],str[maxn*4];
int len1,len2,p[maxn*8];
//p 是每个点为中心的延伸最长回文子串长度, -1 就是原串以这个点为
中心的长度
int i,j,k;
int del1[maxn*8],del2[maxn*8];
int ans[maxn*8];
int main(){
    scanf("%d",&n);
    scanf("%s",s);
    rep(i,n,n*2) s[i]=s[i-n];
    //init();
    int i;
    len1=strlen(s);
    str[0]='+';str[1]='%';
    REP(i,len1+1){
        str[i*2+2]=s[i];
        str[i*2+3]='%';
    }
    len2=len1*2+2;
    //manacher();
    int id=0,mx=0;
    FOR(i,1,len2-1){
        if (mx>i) p[i]=min(p[2*id-i],mx-i);
        else p[i]=1;
        while (str[i+p[i]]==str[i-p[i]]) p[i]++;
        if (p[i]+i>mx){
            mx=p[i]+i;
            id=i;
        }
    }
    REP(i,len2) p[i]--;//manacher
    //solve
    REP(i,len2) {
        if ((p[i]&1)==(n&1)) p[i]=min(p[i],n);
        else p[i]=min(p[i],n-1);
    }
}

```

```

}
build(1,1,len2*2);
REP(i,len2){
    del1[i-p[i]]=max(del1[i-p[i]],p[i]);
    if (i+p[i]-n*2>=0) del2[i+p[i]-n*2]=max(del2[i+p[i]-n*2],p[i]);
    if (i+p[i]-n*2<i-p[i]&&i-p[i]>0){
        update(1,max(0,i+p[i]-n*2)+1,max(0,i-p[i])+1,p[i]);
    }
}
mx=0;
REP(i,len2){
    if (str[i]!='&&str[i]!='+') mx-=2;
    mx=max(mx,del1[i]);
    ans[i]=max(ans[i],mx);
}
mx=0;
rREP(i,len2*2){
    if (str[i]!='&&str[i]!='+') mx-=2;
    mx=max(mx,del2[i]);
    ans[i]=max(ans[i],mx);
}
REP(i,len2) ans[i]=max(ans[i],query(1,i+1));
REP(i,n) printf("%d\n",max(ans[i*2+1],ans[i*2+2]));
}

```

回文自动机

//next 是将字符拼接到两端产生的字符串!

//一定注意这一点!

//也就是说,如果从上到下累积的话,可以很容易的将其与位置联系到一起!

//注意 last 是可以在线的,但是如果加了个其他的可以从 fail 上爬的,

//在讨论外边也要向上爬,或者一次过后就保存下来下次接着使用

//对于 sans,diff,slink:

//sans 是把之前的 series_ans 保留下来

//diff 相同时,sans 一定会与上一个相同(由于对称的特殊性)

//所以只需改变 diff 改变时的 ans 即可

```

struct Ptree{
    int next[maxn][27]; //空间可优化
    int fail[maxn];
    // cnt:这个所代表的字符串个数(下到上所有),num:上到下的 length
    // int cnt[maxn],num[maxn];
    int len[maxn]; //长度
    int diff[maxn]; //length(this-fail)
}

```

```

int slink[maxn]; //diff 不同的 fail,共 log 个
// slink 用来算 sans,sabs 转移得到 ans //用来求的是分成串的个数
int S[maxn]; //字符
int last; //上一个字符节点
int n,tot; //n 表示字符位置
int newnode(int l){
    memset(next[tot],0,sizeof(next[tot]));
    // cnt[tot]=num[tot]=0;
    len[tot]=l; //不是 1...
    return tot++;
}
void init(){
    tot=0;last=n=0;
    newnode(0);newnode(-1);
    S[n]=-1; //减少特判
    fail[0]=1;
}
int getfail(int x){
    while(S[n-len[x]-1]!=S[n]) x=fail[x];
    return x;
}
void add(int c){
    c-='a';
    S[++n]=c;
    int cur=getfail(last);
    if (!next[cur][c]){
        int now=newnode(len[cur]+2);
        fail[now]=next[getfail(fail[cur])][c];
        next[cur][c]=now; //这里一定要在 fail 后边=_=
        diff[now]=len[now]-len[fail[now]];
        if (diff[now]==diff[fail[now]])
            slink[now]=slink[fail[now]];
        else slink[now]=fail[now];
        // num[now]=num[fail[now]]+1;
    }
    last=next[cur][c];
    // cnt[last]++;
}
// void count(){ //count 完 cnt 才对
//     int i;
//     rREP(i,tot) cnt[fail[i]]+=cnt[i];
// }
};
int n,m;
int i,j,k;

```

```

char a[maxn],b[maxn];
LL f[maxn],sans[maxn]; //g:sum; f:sum of sum
int main(){
    scanf("%s",a);
    n=strlen(a);
    if (n%2) return 0*puts(0);
    T.init();m=0;
    REP(i,n/2) b[++m]=a[i],b[++m]=a[n-i-1];
    f[0]=1;
    FOR(i,1,n){
        T.add(b[i]);
        for (int v=T.last;T.len[v]>0;v=T.slink[v]){
            sans[v]=f[i-(T.len[T.slink[v]]+T.diff[v]);
            if (T.diff[v]==T.diff[T.fail[v]])
                (sans[v]+=sans[T.fail[v]])%=M;
            if (!(i&1)) (f[i]+=sans[v])%=M; //f[x]
        }
    }printf("%16d\n",f[n]);
    // REP(i,T.tot) printf("%c",T.S[i]+'a');puts(" S");
    // REP(i,T.tot) printf("%2d ",i);puts(" i");
    // REP(i,T.tot) printf("%2d ",T.S[i]);puts(" S");
    // REP(i,T.tot) printf("%2d ",T.fail[i]);puts(" fail");
    // REP(i,T.tot) printf("%2d ",T.cnt[i]);puts(" cnt");
    // REP(i,T.tot) printf("%2d ",T.len[i]);puts(" len");
    // REP(i,T.tot) printf("%2d ",f[i]);puts(" f");
    // REP(i,T.tot) printf("%2d ",sans[i]);puts(" g");
}

```

二分 hash

// 题意:求上下拼接后的最长回文串长度(很坑)

```

struct hashset{
    const static int seed=1e7+7;
    const static int maxn=2e6+7;
    struct node{
        int x,y;int next;
        node(){};
        node(int _x,int _y,int n):x(_x),y(_y),next(n){};
    }T[maxn]; //更好地空间局部性?(雾)
    int head[seed],size;
    void clear(){
        memset(head,-1,sizeof(head));
        size=0;
    }
}

```

```

void insert(int x,int y){
    int& h=head[x%seed];
    for (int i=h;~i;i=T[i].next)
        if (T[i].x==x&&T[i].y==y) return;
    T[size]=node(x,y,h);h=size++;
}

bool count(int x,int y){
    for (int i=head[x%seed];~i;i=T[i].next)
        if (T[i].x==x&&T[i].y==y) return 1;
    return 0;
}

}have;

struct hash{
    int px[maxn],val[maxn],p;
    void setp(int P,int n=200000){
        int i;px[0]=1;p=P;
        FOR(i,1,n) px[i]=(LL)px[i-1]*p%M;
    }

    void set(char a[],int n){
        int i;val[0]=0;
        FOR(i,1,n) val[i]=((LL)val[i-1]*p+a[i-1])%M;
    }

    int get(int l,int r){
        l++;r++;
        int ret=val[r]-(LL)val[l-1]*px[r-l+1]%M;
        (ret<0)&&(ret+=M);return ret;
    }
}HA,RB;

void manacher(char A[],int p[],int len){
    int id=0,mx=0,i;
    rep(i,1,len){
        if (mx>i) p[i]=min(p[2*id-i],mx-i);
        else p[i]=1;
        while (A[i+p[i]]==A[i-p[i]]) p[i]++;
        if (p[i]+i>mx) mx=p[i]+i,id=i;
    }
}

int n,i;
int s[maxn];
char a[maxn],b[maxn],A[maxn*2],B[maxn*2];
int PA[maxn*2],PB[maxn*2]; //id
int len,ans;

int main(){
    scanf("%d",&n);
    scanf("%s%s",a,b+1);
}

```

```

a[n]='(';b[0]='';n++;
A[len]='+';B[len]='-';len++;
A[len]='%';B[len]='%';len++;
REP(i,n){
    A[len]=a[i];B[len]=b[i];len++;
    A[len]='%'; B[len]='%'; len++;
}A[len]='*';B[len]='/';len++;
n=len;
manacher(A,PA,len);
manacher(B,PB,len);
HA.setp(19);RB.setp(19);
HA.set(A,n);reverse(B,B+n);RB.set(B,n);
reverse(B,B+n);
rep(i,1,n){
    //min(i-1-PA[i]+1,n-1-i-PA[i]+1)+1
    //PA 和 PB 的判断相同 (只需一个最大即可)
    PA[i]=max(PA[i],PB[i]);
    int l=0,r=min(i-PA[i],n-1-i-PA[i])+1;//r: not
    while (l+1<r){
        int mid=(l+r)/2;
        int hash_A=HA.get(i-PA[i]-mid+1,i-PA[i]);
        int hash_B=RB.get(n-(i+PA[i]+mid),n-1-(i+PA[i]));
        if (hash_A==hash_B) l=mid;
        else r=mid;
    }ans=max(ans,PA[i]+l);
}printf("%d\n",ans-1);
}

```

一些 hashset/hashmap

```

template<typename T1,typename T2> struct hashmap{
    const static int seed=999991;
    const static int maxn=1e6+7;
    struct node{
        T1 key;T2 val;int next;
        node();
        node(T1 k,T2 v,int n):key(k),val(v),next(n){};
    }T[maxn];//更好地空间局部性?(雾)
    int head[seed],size;
    void clear(){
        memset(head,-1,sizeof(head));
        size=0;
    }
    void insert(T1 pos,T2 val){

```

```

        int x=pos%seed;
        T[size]=node(pos,val,head[x]);
        head[x]=size++;
    }
    T2 &operator [](T1 x){
        for (int i=head[x%seed];~i;i=T[i].next)
            if (T[i].key==x) return T[i].val;
        insert(x,0);
        return T[size-1].val;
    }
};
//用于字典树啥的空间优化
struct linknode{
    struct node{
        int key,val;int next;
        node();
        node(int k,int v,int n):key(k),val(v),next(n){};
    }T[maxn];//更好地空间局部性?(雾)
    int head[maxn],size;
    void clear(){
        memset(head,-1,sizeof(head));
        size=0;
    }
    int get(int x,int y){
        for (int i=head[x];~i;i=T[i].next)
            if (T[i].key==y) return T[i].val;
        return 0;
    }
    void insert(int pos,int key,int val){
        T[size]=node(key,val,head[pos]);
        head[pos]=size++;
    }
};

```

后缀平衡树

```

//替罪羊树...这道题卡 splay,treap
//题意：加字符，减字符，query 子串个数
//做法：建后缀自动机+LCT；right 集个数或者后缀顺序建平衡
树然后树上 query
//https://www.nowcoder.net/acm/contest/59/C
const double alpha=0.7;//奇了怪了..为啥 0.7-0.8 改改还 T 的
struct sat{
    struct node{

```

```

int son[2];
node(){son[0]=son[1]=0;}
}T[maxn];
ULL rank[maxn];int size[maxn],val[maxn];
int pre[maxn];char c[maxn];
int cnt,root,last;
inline bool cmp(int x,int y){
    return c[x]<c[y]||
        (c[x]==c[y]&&rank[pre[x]]<rank[pre[y]]);
}
int tot;
int f[maxn];
void getrank(int x){
    if (T[x].son[0]) getrank(T[x].son[0]);
    f[++tot]=x;//不能加 if (val[x]);原因我也不知道
    if (T[x].son[1]) getrank(T[x].son[1]);
}
void rerank(int &x,int l,int r,ULL L,ULL R){
    x=0;if (l>r) return;
    ULL mid=(L+R)/2;int m=(l+r)/2;
    x=f[m];rank[x]=mid;
    rerank(T[x].son[0],l,m-1,L,mid-1);
    rerank(T[x].son[1],m+1,r,mid+1,R);
    size[x]=val[x]+size[T[x].son[0]]+size[T[x].son[1]];
}
void rebuild(int &x,ULL l,ULL r){
    if (!x) return;
    tot=0;getrank(x);
    rerank(x,1,tot,l,r);
}
void ins(int &x,ULL l,ULL r){
    ULL mid=(l+r)/2;
    if (!x) {x=cnt;rank[x]=mid;return;}
    size[x]++;
    int p=cmp(x,cnt);
    int &son=T[x].son[p];
    if (p==0) ins(son,l,mid-1);
    else ins(son,mid+1,r);
    if (max(size[T[x].son[0]],size[T[x].son[1]])>=
        size[x]*alpha) rebuild(x,l,r);
}
void insert(char s[]){
    int len=strlen(s);
    REP(i,len){
        c[++cnt]=s[i];size[cnt]=1;val[cnt]=1;

```

```

        pre[cnt]=last;last=cnt;
        ins(root,1ull,1ull<<63);
    }
}
bool cmp(int k,char s[],int len){
    for (int i=0;i<len;i++,k=pre[k]){
        if (!k) return 1;
        if (s[i]!=c[k]) return c[k]<s[i];
    }return 0;
}
int query(char s[],int len){
    int ret=0;
    for (int now=root;now;){
        if (!cmp(now,s,len)) now=T[now].son[0];
        else {
            ret+=val[now]+size[T[now].son[0]];
            now=T[now].son[1];
        }
    }return ret;
}
int query(char s[]){
    int len=strlen(s);
    reverse(s,s+len);s[len]='Z'+1;s[len+1]=0;
    return query(s,len+1)-query(s,len);
}
void del(int k){
    int now,p;
    while (k--){
        for (now=root;now!=last;){
            size[now]--;
            p=rank[last]>rank[now];
            now=T[now].son[p];
        }val[last]=0;size[last]--;
        last=pre[last];
    }
}
}SAT;
int mask;
void decodeWithMask(char s[],int mask){
    int len=strlen(s);
    REP(j,len){
        mask=(mask*131+j)%len;
        char t=s[j];
        s[j]=s[mask];
        s[mask]=t;

```

板子 ???

```
    }  
}  
int n,m,i;  
char a[3000007],op[20];  
int main(){  
    scanf("%d",&n);  
    scanf("%s",a);  
    SAT.insert(a);  
    while (n--){  
        scanf("%s",op);  
        if (op[0]=='Q'){  
            scanf("%s",a);  
            decodeWithMask(a,mask);  
            int ans=SAT.query(a);  
            printf("%d\n",ans);mask^=ans;  
        }else if (op[0]=='D'){  
            scanf("%d",&i);  
            SAT.del(i);  
        }else{  
            scanf("%s",a);  
            decodeWithMask(a,mask);  
            SAT.insert(a);  
        }  
    }  
    return 0;  
}
```


数据结构

按秩合并并查集(+整体二分)

//求删去每个点后图是否存在奇环(主要是整体二分思想)

```
typedef pair<int,int> pii;
#define fi first
#define se second
#define mp make_pair
vector<pii> E[maxn<<2],have[maxn<<2],back[maxn<<2]; //防爆栈
int fa[maxn],val[maxn];
pii getfa(int x){
    int ret=x,color=val[ret];
    while (fa[ret]!=ret) ret=fa[ret],color^=val[ret];
    return mp(ret,color);
}
int sz[maxn];
int ans[maxn];
void solve(int X,int l,int r){
    bool flag=0;
    int i;
    for(pii e:have[X]){
        pii x=getfa(e.fi);
        pii y=getfa(e.se);
        if (x.fi==y.fi){
            if (x.se==y.se){
                flag=1;
                break;
            }
        }else{
            if (sz[x.fi]>sz[y.fi]) swap(x,y);
            back[X].push_back(mp(x.fi,x.se^y.se));
            fa[x.fi]=y.fi;
            sz[y.fi]+=sz[x.fi];
            val[x.fi]^=x.se^y.se;
        }
    }
}
if (flag){
    FOR(i,l,r) ans[i]=0;
}else if (l<r){
    int mid=(l+r)/2;
    for (pii e:E[X]){
```

```
        if ((l<=e.fi&&e.fi<=mid)||l<=e.se&&e.se<=mid))
            E[X<<1].push_back(e);
        else have[X<<1].push_back(e);
        if ((mid+1<=e.fi&&e.fi<=r)||mid+1<=e.se&&e.se<=r))
            E[X<<1|1].push_back(e);
        else have[X<<1|1].push_back(e);
    }
    solve(X<<1,l,mid);
    solve(X<<1|1,mid+1,r);
}
for (pii u:back[X]){
    sz[fa[u.fi]]-=sz[u.fi];
    fa[u.fi]=u.fi;
    val[u.fi]^=u.se;
}
vector<pii>().swap(E[X]);
vector<pii>().swap(have[X]);
vector<pii>().swap(back[X]);
}
int n,m;
int i;
int main()
{
    int T;
    scanf("%d",&T);
    while (T--){
        scanf("%d%d",&n,&m);
        FOR(i,1,n) fa[i]=i,sz[i]=1,ans[i]=1,val[i]=1;
        FOR(i,1,m){
            int u,v;
            scanf("%d%d",&u,&v);
            if (u>v) swap(u,v);
            E[1].push_back(make_pair(u,v));
        }
        solve(1,1,n);
        FOR(i,1,n) printf("%d",ans[i]);puts("");
    }
}
```

二维树状数组

//poj2155,修改区间 01,query 单点 01,差分来做

```
int n,m;
int c[maxn][maxn];
int lowbit(int x){return x&-x;}
```

```

void update(int _x,int _y){
    for (int x=_x;x<=n;x+=lowbit(x))
        for (int y=_y;y<=n;y+=lowbit(y)) c[x][y]^=1;
}
int sum(int _x,int _y){
    int ret=0;
    for (int x=_x;x<=n;x+=lowbit(x))
        for (int y=_y;y<=n;y+=lowbit(y)) ret^=c[x][y];
    return ret;
}
int T;
char s[10];
int i,j,k;
int x1,x2,y1,y2;
int main()
{
    scanf("%d",&T);
    while (T--){
        scanf("%d%d",&n,&m);
        FOR(i,1,n) FOR(j,1,n) c[i][j]=0;
        REP(i,m){
            scanf("%s",s);
            if (s[0]=='C'){
                scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
                update(x1,y1);update(x2+1,y2+1);
                update(x1,y2+1);update(x2+1,y1);
            }else{
                scanf("%d%d",&x1,&y1);
                printf("%d\n",sum(x1,y1));
            }
        }
        puts("");
    }
}

```

树状数组 不大于 k 的最大值

```

const int MAX=1000000;
inline int lowbit(int x){return x&-x;}
inline void insert(int x){
    for (;x<=MAX;x+=lowbit(x)) a[x]++;
}
inline int find(int 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;
}

```

BIT_差分

```

LL A[maxn],B[maxn]; //A*i+B
inline int lowbit(int x){return x&-x;}
void Add(int x,LL val,LL VAL){
    for (;x<=n;x+=lowbit(x)) (A[x]+=val)%=M,(B[x]+=VAL)%=M;
}
void add(int l,int r,LL val){
    Add(l,val,-((l-1)*val%M)+M);
    Add(r+1,M-val,r*val%M);
}
LL query(int x){
    LL ret=0;for (int i=x;x-=lowbit(x)) (ret+=A[x]*i+B[x])%=M;
    return ret;
}
LL query(int l,int r){
    return (query(r)-query(l-1)+M)%M;
}

```

二维线段树

//单点修改区间查询 min,max

```

struct node{
    int left,right;
}treeX[maxn*4],treeY[maxn*4];
int a[maxn*4][maxn*4];
int mx[maxn*4][maxn*4],mn[maxn*4][maxn*4];
void buildY(int x,int y,int yl,int yr){
    treeY[y].left=yl,treeY[y].right=yr;
    if (yl==yr){
        if (treeX[x].left==treeX[x].right)
            mx[x][y]=mn[x][y]=a[treeX[x].left][yl];
        else{
            mx[x][y]=max(mx[x<<1][y],mx[x<<1|1][y]);
            mn[x][y]=min(mn[x<<1][y],mn[x<<1|1][y]);
        }
    }
}

```

```

        return;
    }
    int mid=(yl+yr)/2;
    buildY(x,y<<1,yl,mid);
    buildY(x,y<<1|1,mid+1,yr);
    mx[x][y]=max(mx[x][y<<1],mx[x][y<<1|1]);
    mn[x][y]=min(mn[x][y<<1],mn[x][y<<1|1]);
}
void buildX(int x,int n,int xl,int xr){
    treeX[x].left=xl,treeX[x].right=xr;
    if (xl==xr){
        buildY(x,1,1,n);
        return;
    }
    int mid=(xl+xr)/2;
    buildX(x<<1,n,xl,mid);
    buildX(x<<1|1,n,mid+1,xr);
    buildY(x,1,1,n);
}

int querymaxY(int x,int y,int yl,int yr){
    int L=treeY[y].left,R=treeY[y].right;
    if (yl<=L&&R<=yr){
        return mx[x][y];
    }
    int mid=(L+R)/2,ret=0;
    if (mid>=yl) ret=max(ret,querymaxY(x,y<<1,yl,yr));
    if (yr>mid) ret=max(ret,querymaxY(x,y<<1|1,yl,yr));
    return ret;
}

int querymaxX(int x,int xl,int xr,int yl,int yr){
    int L=treeX[x].left,R=treeX[x].right;
    if (xl<=L&&R<=xr){
        return querymaxY(x,1,yl,yr);
    }
    int mid=(L+R)/2,ret=0;
    if (mid>=xl) ret=max(ret,querymaxX(x<<1,xl,xr,yl,yr));
    if (xr>mid) ret=max(ret,querymaxX(x<<1|1,xl,xr,yl,yr));
    return ret;
}

int queryminY(int x,int y,int yl,int yr){
    int L=treeY[y].left,R=treeY[y].right;
    if (yl<=L&&R<=yr){
        return mn[x][y];

```

```

    }
    int mid=(L+R)/2,ret=INF;
    if (mid>=yl) ret=min(ret,queryminY(x,y<<1,yl,yr));
    if (yr>mid) ret=min(ret,queryminY(x,y<<1|1,yl,yr));
    return ret;
}

int queryminX(int x,int xl,int xr,int yl,int yr){
    int L=treeX[x].left,R=treeX[x].right;
    if (xl<=L&&R<=xr){
        return queryminY(x,1,yl,yr);
    }
    int mid=(L+R)/2,ret=INF;
    if (mid>=xl) ret=min(ret,queryminX(x<<1,xl,xr,yl,yr));
    if (xr>mid) ret=min(ret,queryminX(x<<1|1,xl,xr,yl,yr));
    return ret;
}

void updateY(int x,int y,int posy,int val){
    int L=treeY[y].left,R=treeY[y].right;
    if (L==R){
        if (treeX[x].left==treeX[x].right)
            mx[x][y]=mn[x][y]=val;
        else{
            mx[x][y]=max(mx[x<<1][y],mx[x<<1|1][y]);
            mn[x][y]=min(mn[x<<1][y],mn[x<<1|1][y]);
        }
        return;
    }
    int mid=(L+R)/2;
    if (mid>=posy) updateY(x,y<<1,posy,val);
    else updateY(x,y<<1|1,posy,val);
    mx[x][y]=max(mx[x][y<<1],mx[x][y<<1|1]);
    mn[x][y]=min(mn[x][y<<1],mn[x][y<<1|1]);
}

void updateX(int x,int posx,int posy,int val){
    int L=treeX[x].left,R=treeX[x].right;
    if (L==R){
        updateY(x,1,posy,val);
        return;
    }
    int mid=(L+R)/2;
    if (mid>=posx) updateX(x<<1,posx,posy,val);
    else updateX(x<<1|1,posx,posy,val);
    updateY(x,1,posy,val);
}

```

```

int n,m,q;
int i,j;
int ans;
int main(){
    int T,x=0;
    scanf("%d",&T);
    while (T--){
        scanf("%d",&n);
        FOR(i,1,n)
            FOR(j,1,n) scanf("%d",&a[i][j]);
        buildX(1,n,1,n);
        scanf("%d",&q);
        printf("Case #%d:\n",++x);
        while (q--){
            int x,y,r;
            scanf("%d%d%d",&x,&y,&r);
            r/=2;
            int xl=max(1,x-r),xr=min(n,x+r);
            int yl=max(1,y-r),yr=min(n,y+r);
            int MX=querymaxX(1,xl,xr,yl,yr);
            int MN=queryminX(1,xl,xr,yl,yr);
            updateX(1,x,y,(MX+MN)/2);
            printf("%d\n",(MX+MN)/2);
        }
    }
}

```

扫描线 矩形周长并

```

int size;
int len[maxn*2];
int n,m;
int i,j,k;
struct Seg {
    struct node {
        int left,right;
        int len,num;
        bool cl,cr;//iff
        int lazy;
        void update(int x) {
            lazy+=x;
        }
    } tree[maxn*4];
    void pushup(int x) {
        if (tree[x].lazy) {

```

```

            tree[x].len=len[tree[x].right+1]-len[tree[x].left];
            tree[x].cl=tree[x].cr=1; tree[x].num=2;
        } else if (tree[x].left==tree[x].right) {
            tree[x].len=0;
            tree[x].cl=tree[x].cr=0; tree[x].num=0;
        } else {
            tree[x].len=tree[x<<1].len+tree[x<<1|1].len;
            tree[x].num=tree[x<<1].num+tree[x<<1|1].num;
            if (tree[x<<1].cr&&tree[x<<1|1].cl) tree[x].num-=2;
            tree[x].cl=tree[x<<1].cl;
            tree[x].cr=tree[x<<1|1].cr;
        }
    };
    void build(int x,int l,int r) {
        tree[x].left=l; tree[x].right=r;
        tree[x].len=tree[x].lazy=0;
        if (l==r) {
        } else {
            int mid=(l+r)/2;
            build(x<<1,l,mid);
            build(x<<1|1,mid+1,r);
            pushup(x);
        }
    }
    void update(int x,int l,int r,LL val) {
        int L=tree[x].left,R=tree[x].right;
        if (l<=L&&R<=r) {
            tree[x].update(val);
            pushup(x);
        } else {
            int mid=(L+R)/2;
            if (mid>=l) update(x<<1,l,r,val);
            if (r>mid) update(x<<1|1,l,r,val);
            pushup(x);
        }
    }
    int query(int x,int l,int r) { //num
        int L=tree[x].left,R=tree[x].right;
        if (l<=L&&R<=r) {
            return tree[x].len;
        } else {
            int mid=(L+R)/2;
            int ans;
            if (mid>=l) ans+=query(x<<1,l,r);
            if (r>mid) ans+=query(x<<1|1,l,r);

```

主席树

//静态区间第 k 大

vector<int> v;//学到的 hash 方法

int getid(int x){return lower_bound(v.begin(),v.end(),x)-v.begin()+1;}

int root[maxn],a[maxn],cnt;

struct Tnode{

int left,right,sum;

}T[maxn*40];

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)/2;

if (mid>=pos) update(l,mid,T[x].left,T[y].left,pos);

else update(mid+1,r,T[x].right,T[y].right,pos);

}

int query(int l,int r,int x,int y,int k){

if (l==r) return l;

int mid=(l+r)/2;

int sum=T[T[y].left].sum-T[T[x].left].sum;

if (sum>=k) return query(l,mid,T[x].left,T[y].left,k);

else return query(mid+1,r,T[x].right,T[y].right,k-sum);

}

int n,m;

int i,j,k,ii;

int main()

{

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

FOR(i,1,n) scanf("%d",&a[i]),v.push_back(a[i]);

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

FOR(i,1,n) update(1,n,root[i],root[i-1],getid(a[i]));

REP(ii,m){

scanf("%d%d%d",&i,&j,&k);

printf("%d\n",v[query(1,n,root[i-1],root[j],k)-1]);

}

return 0;

}

区间不重复数字个数和第 k 个是哪位

int cnt;

struct node{

int l,r,sum;

}T[maxn*40];

void update(int l,int r,int &x,int y,int pos,int v){

```

        pushup(x);
        return ans;
    }
}
} T;
struct point {
    int x1,x2,h;
    int n;
    bool operator <(const point &a)const {
        if (h!=a.h) return h<a.h;
        return n>a.n;
    }
} a[maxn];
map<int,int> Hash;
int x1,x2,y1,y2;
int ans;
int len1,len2,num;
int main() {
    while (~scanf("%d",&n)) {
        if (n==0) break;
        FOR(i,1,n) {
            scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
            len[i*2-1]=x1; len[i*2]=x2;
            a[i*2-1].x1=x1; a[i*2-1].x2=x2;
            a[i*2-1].n=1; a[i*2-1].h=y1;
            a[i*2].x1=x1; a[i*2].x2=x2;
            a[i*2].n=-1; a[i*2].h=y2;
        }
        sort(a+1,a+n*2+1);
        sort(len+1,len+n*2+1);
        Hash.clear();
        FOR(i,1,2*n) Hash[len[i]]=i;
        T.build(1,1,n*2);
        ans=0;
        FOR(i,1,2*n) {
            len1=T.tree[1].len; num=T.tree[1].num;
            T.update(1,Hash[a[i].x1],Hash[a[i].x2]-1,a[i].n);
            len2=T.tree[1].len;
            ans+=abs(len2-len1);
            ans+=num*(a[i].h-a[i-1].h);
        }
        printf("%d\n",ans);
    }
}

```

```

T[++cnt]=T[y],T[cnt].sum+=v,x=cnt;
if (l==r) return;
int mid=(l+r)/2;
if (mid>=pos) update(l,mid,T[x].l,T[y].l,pos,v);
else update(mid+1,r,T[x].r,T[y].r,pos,v);
}
int findsum(int l,int r,int x,int L,int R){
//每个点记录的都是这个点往后的相同数(前面把后面短路了)
if (L<=l&&r<=R) return T[x].sum;
int mid=(l+r)/2;
int sum=0;
if (mid>=L) sum+=findsum(l,mid,T[x].l,L,R);
if (R>mid) sum+=findsum(mid+1,r,T[x].r,L,R);
return sum;
}
int query(int l,int r,int x,int k){
if (l==r) return l;
int mid=(l+r)/2;
int sum=T[T[x].l].sum;
if (sum>=k) return query(l,mid,T[x].l,k);
else return query(mid+1,r,T[x].r,k-sum);
}
int n,m;
int i,j,k,pos;
int t,TT;
int ans[maxn],a[maxn];
int last[maxn],root[maxn];
int main()
{
scanf("%d",&TT);
FOR(t,1,TT){
scanf("%d%d",&n,&m);
FOR(i,1,n) scanf("%d",&a[i]);
FOR(i,1,n) last[a[i]]=0,root[i]=0;
cnt=0;
rFOR(i,1,n){
if (!last[a[i]]) update(1,n,root[i],root[i+1],i,1);
else {
update(1,n,root[i],root[i+1],last[a[i]],-1);
update(1,n,root[i],root[i],i,1);
}
last[a[i]]=i;
}
FOR(i,1,m){
scanf("%d%d",&j,&k);

```

```

j=(j+ans[i-1])%n+1;
k=(k+ans[i-1])%n+1;
if (j>k) swap(j,k);
pos=(findsum(1,n,root[j],j,k)+1)/2;
ans[i]=query(1,n,root[j],pos);
}
printf("Case #%d:",t);
FOR(i,1,m) printf(" %d",ans[i]);
puts("");
}
return 0;
}

```

可持久化数组(主席树维护)

```

struct Tnode{
int left,right,val;
}T[maxn*80];
int cnt=0;
void build(int &x,int l,int r){
if (!x) x=++cnt;
if (l==r) {T[x].val=l; return;}
int mid=(l+r)/2;
build(T[x].left,l,mid);
build(T[x].right,mid+1,r);
}
void update(int &x,int y,int pos,int val,int l,int r){
T[++cnt]=T[y];x=cnt;
if (l==r) {T[x].val=val; return;}
int mid=(l+r)/2;
if (mid>=pos) update(T[x].left,T[y].left,pos,val,l,mid);
else update(T[x].right,T[y].right,pos,val,mid+1,r);
}
int query(int x,int pos,int l,int r){
if (l==r) return T[x].val;
int mid=(l+r)/2;
if (mid>=pos) return query(T[x].left,pos,l,mid);
else return query(T[x].right,pos,mid+1,r);
}
int root[maxn];
int n,m;
int i,j,k,t;
int a,b,ans;
inline int getfather(int x){
int t=query(root[i],x,1,n);

```

```

if (t==x) return x;
int fa=getfather(t);
update(root[i],root[i],x,fa,1,n);
return fa;
}
int main()
{
    scanf("%d%d",&n,&m);
    build(root[0],1,n);
    FOR(i,1,m){
        scanf("%d",&k);
        root[i]=root[i-1];
        if (k==1){
            scanf("%d%d",&a,&b);
            a^=ans;b^=ans;
            int x=getfather(a),y=getfather(b);
            if (x==y) continue;
            update(root[i],root[i],x,y,1,n);
        }else if (k==2){
            scanf("%d",&t);
            t^=ans;
            root[i]=root[t];
        }else{
            scanf("%d%d",&a,&b);
            int x=getfather(a),y=getfather(b);
            a^=ans;b^=ans;
            if (x==y) puts("1"),ans=1;
            else puts("0"),ans=0;
        }
    }
    return 0;
}

```

树套树

// zoj2112 动态第 k 大(这个是类似 kuangbin 大佬的做法按点建树，我按权值多个 log...)

```

struct node{
    int l,r,cnt;
    node(){l=r=cnt=0;}
}T[2500010];
int cnt;
int SIZE;
inline int lowbit(int x){
    return x&(-x);
}

```

板子 ???

```

}
void Update(int &x,int y,int l,int r,int pos,int val){
    T[++cnt]=T[y];T[cnt].cnt+=val;x=cnt;
    if (l==r) return;
    int mid=(l+r)/2;
    if (mid>=pos) Update(T[x].l,T[y].l,l,mid,pos,val);
    else Update(T[x].r,T[y].r,mid+1,r,pos,val);
}
int n,m;
int root[maxn];
void update(int x,int pos,int val){
    while (x<=n){
        Update(root[x],root[x],1,SIZE,pos,val);
        x+=lowbit(x);
    }
}
int ROOT[maxn];
int useL[maxn],useR[maxn];//现在的 l/r
int Query(int l,int r,int L,int R,int pos,int pre_L,int pre_R){//颜色,pos L->R
    if (l==r) return l;
    int x;
    int mid=(l+r)/2,nowcnt=0;
    for(x=L-1;x-=lowbit(x)) nowcnt-=T[T[useL[x]].l].cnt;
    for(x=R;x-=lowbit(x)) nowcnt+=T[T[useR[x]].l].cnt;
    nowcnt+=T[T[pre_R].l].cnt-T[T[pre_L].l].cnt;
    if (nowcnt>=pos){
        for(x=L-1;x-=lowbit(x)) useL[x]=T[useL[x]].l;
        for(x=R;x-=lowbit(x)) useR[x]=T[useR[x]].l;
        return Query(l,mid,L,R,pos,T[pre_L].l,T[pre_R].l);
    }else{
        for(x=L-1;x-=lowbit(x)) useL[x]=T[useL[x]].r;
        for(x=R;x-=lowbit(x)) useR[x]=T[useR[x]].r;
        return Query(mid+1,r,L,R,pos-nowcnt,T[pre_L].r,T[pre_R].r);
    }
}
int query(int L,int R,int pos){
    int x;
    for(x=L-1;x-=lowbit(x)) useL[x]=root[x];
    for(x=R;x-=lowbit(x)) useR[x]=root[x];
    return Query(1,SIZE,L,R,pos,ROOT[L-1],ROOT[R]);
}
char K[maxn],Q[20];
int A[maxn][4];
int a[maxn];
vector<int> H;

```

```

inline int getid(int x){return lower_bound(H.begin(),H.end(),x)-H.begin()+1;}
void solve(){
    scanf("%d%d",&n,&m);
    int i;
    FOR(i,1,n) scanf("%d",&a[i]),H.push_back(a[i]);
    REP(i,m){
        scanf("%s",Q);
        K[i]=Q[0];
        if (K[i]=='Q') scanf("%d%d%d",&A[i][0],&A[i][1],&A[i][2]);
        if (K[i]=='C') scanf("%d%d",&A[i][0],&A[i][1]),H.push_back(A[i][1]);
    }
    sort(H.begin(),H.end());H.erase(unique(H.begin(),H.end()),H.end());
    SIZE=H.size();
    cnt=0;
    FOR(i,1,n) Update(ROOT[i],ROOT[i-1],1,SIZE,getid(a[i]),1);
    REP(i,m){
        if (K[i]=='Q') printf("%d\n",H[query(A[i][0],A[i][1],A[i][2])-1]);
        if (K[i]=='C'){
            update(A[i][0],getid(a[A[i][0]]),-1);
            a[A[i][0]]=A[i][1];
            update(A[i][0],getid(A[i][1]),1);
        }
    }
    FOR(i,1,n) root[i]=0;
    FOR(i,1,cnt) T[i]=node();
    vector<int>().swap(H);
}
int main(){
    T[0].cnt=T[0].l=T[0].r=0;
    int T_T;
    scanf("%d",&T_T);
    while (T_T--) solve();
}

```

CDQ 分治(套线段树)

```

// CF848C CDQ 分治 (区间数字出现的 r-l 之和)
//将所有操作计算成为 add 和 del,然后 solve(l,r),再去除影响
const LL MAX=10000007;
struct node{
    int l,r;
    LL sum;
}T[MAX];
int cnt;
void Update(int &x,int pos,int val,int l,int r){

```

```

    if (!x) x=++cnt;
    T[x].sum+=val;
    if (l==r) return;
    int mid=(l+r)/2;
    if (mid>=pos) Update(T[x].l,pos,val,l,mid);
    else Update(T[x].r,pos,val,mid+1,r);
}
LL Query(int x,int l,int r,int L,int R){
    if (!x||(l<=L&&R<=r)) return T[x].sum;
    int mid=(L+R)/2;
    LL ret=0;
    if (mid>=l) ret+=Query(T[x].l,l,r,L,mid);
    if (r>mid) ret+=Query(T[x].r,r,l,mid+1,R);
    return ret;
}
int n,m;
int root[maxn];
inline int lowbit(int x){
    return x&-x;
}
void update(int x,int pos,int val){
    for (;x<=n;x+=lowbit(x)) Update(root[x],pos,val,1,n);
}
LL query(int x,int l,int r){
    LL ret=0;
    for (;x<=n;x+=lowbit(x))
        ret+=Query(root[x],l,r,1,n);
    return ret;
}
int a[maxn];
set<int> S[maxn];
void ins(int pos,int val){//固定 R (L 用前缀和)
    S[val].insert(pos);
    set<int>::iterator it=S[val].lower_bound(pos),itt=it;itt++;
    int pre=0,suf=0;
    if (it!=S[val].begin()) it--,pre=*it;
    if (itt!=S[val].end()) suf=*itt;
    if (pre) update(pos,pre,pos-pre);
    if (suf) update(suf,pos,suf-pos);
    if (pre&&suf) update(suf,pre,pre-suf);
}
void del(int pos,int val){
    set<int>::iterator it=S[val].lower_bound(pos),itt=it;itt++;
    int pre=0,suf=0;
    if (it!=S[val].begin()) it--,pre=*it;

```



```

if (itt!=S[val].end()) suf=*itt;
if (pre) update(pos,pre,-(pos-pre));
if (suf) update(suf,pos,-(suf-pos));
if (pre&& suf) update(suf,pre,-(pre-suf));
S[val].erase(pos);
}
int i;
int main(){
    scanf("%d%d",&n,&m);
    FOR(i,1,n){
        scanf("%d",&a[i]);
        ins(i,a[i]);
    }
    REP(i,m){
        int k;
        scanf("%d",&k);
        if (k==1){
            int p,x;
            scanf("%d%d",&p,&x);
            del(p,a[p]);
            a[p]=x;
            ins(p,a[p]);
        }else if (k==2){
            int l,r;
            scanf("%d%d",&l,&r);
            printf("%l64d\n",query(r,l,r));
        }
    }
}

```

SPLAY

```

int a[maxn],cnt;
struct splay_tree{
    struct node{
        int val,min,add,size,son[2]; //add= lazy
        bool rev;
        void init(int _val){ //开始时 T[i].val==a[i-1](线性的);
            val=min=max=_val;size=1;
            if (_val==INF) max=-INF;
            add=rev=son[0]=son[1]=0;
        }
    }T[maxn*2]; //内存池
    int fa[maxn*2],root,tot;
    void pushup(int x){

```

```

T[x].min=T[x].max=T[x].val;T[x].size=1;
if (T[x].val==INF) T[x].max=-INF;
if (T[x].son[0]){
    T[x].min=min(T[x].min,T[T[x].son[0]].min);
    T[x].max=max(T[x].max,T[T[x].son[0]].max);
    T[x].size+=T[T[x].son[0]].size;
}
if (T[x].son[1]){
    T[x].min=min(T[x].min,T[T[x].son[1]].min);
    T[x].max=max(T[x].max,T[T[x].son[1]].max);
    T[x].size+=T[T[x].son[1]].size;
}
}
void pushdown(int x){
    if (x==0) return;
    if (T[x].add){
        if (T[x].son[0]){
            T[T[x].son[0]].val+=T[x].add;
            T[T[x].son[0]].min+=T[x].add;
            T[T[x].son[0]].max+=T[x].add;
            T[T[x].son[0]].add+=T[x].add;
        }
        if (T[x].son[1]){
            T[T[x].son[1]].val+=T[x].add;
            T[T[x].son[1]].min+=T[x].add;
            T[T[x].son[1]].max+=T[x].add;
            T[T[x].son[1]].add+=T[x].add;
        }
        T[x].add=0;
    }
    if (T[x].rev){
        if (T[x].son[0]) T[T[x].son[0]].rev^=1;
        if (T[x].son[1]) T[T[x].son[1]].rev^=1;
        swap(T[x].son[0],T[x].son[1]);
        T[x].rev=0;
    }
}
void rotate(int x,int kind){ //zig(1->) zag(0<-)都行
    int y=fa[x],z=fa[y];
    T[y].son[!kind]=T[x].son[kind],fa[T[x].son[kind]]=y;
    T[x].son[kind]=y,fa[y]=x;
    T[z].son[T[z].son[1]==y]=x,fa[x]=z;
    pushup(y);
}
void splay(int x,int goal){ //node x->goal's son

```

板子 ???

```
if (x==goal) return;
while (fa[x]!=goal){
    int y=fa[x],z=fa[y];
    pushdown(z),pushdown(y),pushdown(x);
    int rx=T[y].son[0]==x,ry=T[z].son[0]==y;
    if (z==goal) rotate(x,rx);
    else{
        if (rx==ry) rotate(y,ry);
        else rotate(x,rx);
        rotate(x,ry);
    }
}
pushup(x);
if (goal==0) root=x;
}

int select(int pos){//getnode
    int u=root;
    pushdown(u);
    while (T[u].son[0].size!=pos){//这里由于头节点有个-INF 所以不-1
        if (pos<T[u].son[0].size) u=T[u].son[0];
        else{
            pos-=T[u].son[0].size+1;
            u=T[u].son[1];
        }
        pushdown(u);
    }
    return u;
}

//下面是自己写的一点常用函数
void update(int l,int r,int val){
    int u=select(l-1),v=select(r+1);
    splay(u,0);
    splay(v,u);
    T[T[v].son[0]].min+=val;
    T[T[v].son[0]].max+=val;
    T[T[v].son[0]].val+=val;
    T[T[v].son[0]].add+=val;//lazy
}

void reverse(int l,int r){
    int u=select(l-1),v=select(r+1);
    splay(u,0);splay(v,u);
    T[T[v].son[0]].rev^=1;
}

void revolve(int l,int r,int x){//l~r->循环往后 x 位
```

```
int u=select(r-x),v=select(r+1);
splay(u,0);splay(v,u);
int tmp=T[v].son[0];T[v].son[0]=0;
pushup(v);pushup(u);
u=select(l-1),v=select(l);
splay(u,0);splay(v,u);
fa[tmp]=v;
T[v].son[0]=tmp;
pushup(v);pushup(u);
}

void cut(int l,int r,int x){//l~r->去掉的 x 位置后 //HDU3487
    int u=select(l-1),v=select(r+1);
    splay(u,0);splay(v,u);
    int tmp=T[v].son[0];
    T[v].son[0]=0;
    pushup(v);pushup(u);
    u=select(x),v=select(x+1);
    splay(u,0);splay(v,u);
    fa[tmp]=v;
    T[v].son[0]=tmp;
    pushup(v);pushup(u);
}

int query_min(int l,int r){
    int u=select(l-1),v=select(r+1);
    splay(u,0);
    splay(v,u);
    return T[T[v].son[0]].min;
}

void insert(int x,int val){
    int u=select(x),v=select(x+1);
    splay(u,0);
    splay(v,u);
    T[tot].init(val);
    fa[tot]=v;
    T[v].son[0]=tot++;
    pushup(v);pushup(u);
}

void erase(int x){
    int u=select(x-1),v=select(x+1);
    splay(u,0);
    splay(v,u);
    T[v].son[0]=0;
    pushup(v);pushup(u);
}

void exchange(int l1,int r1,int l2,int r2){//r1-l1+1?=r2-l2+1 OK
```

板子 ???

```
if (l1>l2){swap(l1,l2);swap(r1,r2);}
int u=select(l1-1),v=select(r1+1);
splay(u,0);splay(v,u);
int tmp=T[v].son[0];T[v].son[0]=0;
pushup(v);pushup(u);
l2-=T[tmp].size;r2-=T[tmp].size;
int _u=select(l2-1),_v=select(r2+1);
splay(_u,0);splay(_v,_u);
fa[tmp]=_v;
swap(T[_v].son[0],tmp);
pushup(_v);pushup(_u);
u=select(l1-1),v=select(l1);
splay(u,0);splay(v,u);
fa[tmp]=v;
T[v].son[0]=tmp;
pushup(v);pushup(u);
}
int dfs(int x,int k){//小于 k 的值个数,会被卡
    if (x==0) return 0;
    if (T[x].min!=INF&&T[x].min>=k) return 0;
    if (T[x].max!=-INF&&T[x].max<k) return T[x].size;
    int ret=T[x].val<k;
    if (T[x].son[0]) ret+=dfs(T[x].son[0],k);
    if (T[x].son[1]) ret+=dfs(T[x].son[1],k);
    return ret;
}
//小于 k 的值个数,会被卡 应该套主席树(但是太长, 两个 log)
int query(int l,int r,int k){
    int u=select(l-1),v=select(r+1);
    splay(u,0);splay(v,u);
    return dfs(T[v].son[0],k);
}
int build(int l,int r){
    if (l>r) return 0;
    if (l==r) return l;
    int mid=(l+r)/2;
    T[mid].son[0]=build(l,mid-1);
    T[mid].son[1]=build(mid+1,r);
    fa[T[mid].son[0]]=fa[T[mid].son[1]]=mid;
    pushup(mid);
    return mid;
}
void init(int n){
    tot=0;
    int i;//0 是虚的;
```

```
T[tot++].init(INF);//空的
T[tot++].init(INF);//前后两个-INF 节点
FOR(i,1,n) T[tot++].init(a[i]);
T[tot++].init(INF);
root=build(1,tot-1);
fa[root]=0;
fa[0]=0;T[0].son[1]=root;T[0].size=0;
}
void print(int now=-1){
    if (now==-1) now=root;
    pushdown(now);
    if (T[now].son[0]) print(T[now].son[0]);
    if (T[now].val!=-INF){
        if (cnt++) printf(" ");
        printf("%d",T[now].val);
    }
    if (T[now].son[1]) print(T[now].son[1]);
    pushup(now);
}
}T;
```

SPLAY 启发式合并

//HDU6133, 一棵树的合并

```
struct splaytree{
    struct node{
        LL val,sum;
        int son[2],size;
        void init(LL _val){
            val=sum=_val;size=1;
            son[0]=son[1]=0;
        }
    }T[maxn];//编号是对应的
    int fa[maxn];
    int root;
    inline void pushup(int x){
        T[x].sum=T[x].val;
        T[x].size=1;
        if (T[x].son[0]){
            T[x].sum+=T[T[x].son[0]].sum;
            T[x].size+=T[T[x].son[0]].size;
        }
        if (T[x].son[1]){
            T[x].sum+=T[T[x].son[1]].sum;
            T[x].size+=T[T[x].son[1]].size;
        }
    }
```

```

    }
}

void rotate(int x,int kind){
    int y=fa[x],z=fa[y];
    T[y].son[!kind]=T[x].son[kind],fa[T[x].son[kind]]=y;
    T[x].son[kind]=y,fa[y]=x;
    T[z].son[T[z].son[1]==y]=x,fa[x]=z;
    pushup(y);
}

void splay(int x,int goal){
    if (x==goal) return;
    while (fa[x]!=goal){
        int y=fa[x],z=fa[y];
        int rx=T[y].son[0]==x,ry=T[z].son[0]==y;
        if (z==goal) rotate(x,rx);
        else{
            if (rx==ry) rotate(y,ry);
            else rotate(x,rx);
            rotate(x,ry);
        }
    }
    pushup(x);
    if (goal==0) root=x;
}

LL insert(int x){//x 为原先位置
    int u=root,f=0;
    while (u){
        f=u;
        if (T[x].val<T[u].val) u=T[u].son[0];
        else u=T[u].son[1];
    }
    if (T[x].val<T[f].val) T[f].son[0]=x;
    else T[f].son[1]=x;
    fa[x]=f;
    splay(x,0);
    return T[T[x].son[0]].sum+T[x].val*(T[T[x].son[1]].size+1);
}

LL dfs(int x){
    int l=T[x].son[0],r=T[x].son[1];
    LL ret=0;
    T[x].init(T[x].val);
    if (l) ret+=dfs(l);
    ret+=insert(x);
    if (r) ret+=dfs(r);
    return ret;
}

```

```

    }

LL merge(int x,int y,LL tmp,LL ret){
    if (x==y) return tmp;
    splay(x,0);splay(y,0);
    if (T[x].size>T[y].size) swap(x,y),swap(tmp,ret);
    root=y;
    ret+=dfs(x);
    return ret;
}

int getkth(int x,int k){//未验证,抄的前面那个板子
    int u=root;
    while (T[T[u].son[0]].size!=k){
        if (k<T[T[u].son[0]].size) u=T[u].son[0];
        else{
            k-=T[T[u].son[0]].size+1;
            u=T[u].son[1];
        }
    }
    return T[x].val;
}

}T;
int n,m;
vector<int> edge[maxn];
LL ans[maxn];
int val[maxn];
void dfs(int x,int fa){
    ans[x]=val[x];
    for (int v:edge[x]){
        if (v==fa) continue;
        dfs(v,x);
        ans[x]=T.merge(x,v,ans[x],ans[v]);
    }
}

int i,j,k;
int main(){
    int TT;
    scanf("%d",&TT);
    while (TT--){
        scanf("%d",&n);
        FOR(i,1,n) scanf("%d",&val[i]);
        REP(i,n-1){
            int u,v;
            scanf("%d%d",&u,&v);
            edge[u].push_back(v);
            edge[v].push_back(u);
        }
    }
}

```

```

    }
    FOR(i,1,n) T.T[i].init(val[i]);
    dfs(1,0);
    FOR(i,1,n) printf("%lld ",ans[i]);
    puts("");
    FOR(i,1,n) T.fa[i]=0;
    FOR(i,1,n) ans[i]=0,vector<int>().swap(edge[i]);
}
}

```

LCT

//确认没写错, 加边减边, 改边权, 查第二大值

//修改边权:把边当成点,mark 一下,然后左右端点连边即可

```

struct LCT{
    struct node{
        int son[2],val,size;
        int max,add,cnt1;//max
        int ans,lazy,cnt2;//second
        bool rev;
        void init(int _val){
            son[0]=son[1]=rev=add=0;
            max=val=_val;
            size=1;
            cnt1=1;cnt2=0;
            ans=lazy=-INF;
        }
    }T[maxn];
    bool root[maxn];
    int fa[maxn];
    void Reverse(int x){
        T[x].rev^=1;
        swap(T[x].son[0],T[x].son[1]);
    }
    void Add(int x,int val){
        T[x].max+=val;
        T[x].add+=val;
        T[x].val+=val;
        if (T[x].ans!=-INF) T[x].ans+=val;;
        if (T[x].lazy!=-INF) T[x].lazy+=val;
    }
    void Change(int x,int val){//先 change
        T[x].max=val;
        T[x].add=0;
    }
}

```

```

T[x].val=val;
T[x].ans=-INF;
T[x].cnt2=-INF;
T[x].cnt1=T[x].size;
T[x].lazy=val;
}

void Update(int x,int val,int num){
    if (T[x].max==val) T[x].cnt1+=num;
    else if (T[x].max<val){
        T[x].ans=T[x].max;
        T[x].cnt2=T[x].cnt1;
        T[x].max=val;
        T[x].cnt1=num;
    }
    else if (T[x].ans==val) T[x].cnt2+=num;
    else if (T[x].ans<val){
        T[x].ans=val;
        T[x].cnt2=num;
    }
}

void pushup(int x){
    T[x].size=1;
    T[x].max=T[x].val;
    T[x].ans=T[x].lazy=-INF;
    T[x].cnt1=1;T[x].cnt2=0;
    if (T[x].son[0]){
        Update(x,T[T[x].son[0]].max,T[T[x].son[0]].cnt1);
        Update(x,T[T[x].son[0]].ans,T[T[x].son[0]].cnt2);
        T[x].size+=T[T[x].son[0]].size;
    }
    if (T[x].son[1]){
        Update(x,T[T[x].son[1]].max,T[T[x].son[1]].cnt1);
        Update(x,T[T[x].son[1]].ans,T[T[x].son[1]].cnt2);
        T[x].size+=T[T[x].son[1]].size;
    }
}

void pushdown(int x){
    if (T[x].rev){
        if (T[x].son[0]) Reverse(T[x].son[0]);
        if (T[x].son[1]) Reverse(T[x].son[1]);
        T[x].rev=0;
    }
    if (T[x].add){
        if (T[x].son[0]) Add(T[x].son[0],T[x].add);
        if (T[x].son[1]) Add(T[x].son[1],T[x].add);
    }
}

```

```

        T[x].add=0;
    }
    if (T[x].lazy!=-INF){
        if (T[x].son[0]) Change(T[x].son[0],T[x].lazy);
        if (T[x].son[1]) Change(T[x].son[1],T[x].lazy);
        T[x].lazy=-INF;
    }
}

void rotate(int x,int kind){
    int y=fa[x],z=fa[y];
    T[y].son[!kind]=T[x].son[kind],fa[T[x].son[kind]]=y;
    T[x].son[kind]=y,fa[y]=x;
    if (root[y]) {root[x]=true;root[y]=false;}
    else T[z].son[T[z].son[1]==y]=x;
    fa[x]=z;
    pushup(y);
}

void Prechange(int x){
    if (!root[x]) Prechange(fa[x]);
    pushdown(x);
}

void splay(int x){//to root
    Prechange(x);
    while (!root[x]){
        int y=fa[x],z=fa[y];
        int rx=T[y].son[0]==x,ry=T[z].son[0]==y;
        if (root[y]) rotate(x,rx);
        else{
            if (rx==ry) rotate(y,ry);
            else rotate(x,rx);
            rotate(x,ry);
        }
    }
    pushup(x);
}

int access(int x){//只有这条链上的是 mark 的
    int y=0;
    for (;x=fa[x]){
        splay(x);
        root[T[x].son[1]]=true;
        T[x].son[1]=y;
        root[y]=false;
        y=x;
        pushup(x);
    }
}

```

```

        return y;
    }

    bool judge(int u,int v){
        while (fa[u]) u=fa[u];
        while (fa[v]) v=fa[v];
        return u==v;
    }

    void makeroot(int x){
        access(x);
        splay(x);
        Reverse(x);
    }

    bool link(int u,int v){
        if (judge(u,v)) return 1;
        makeroot(u);
        fa[u]=v;
        return 0;
    }

    bool cut(int u,int v){
        makeroot(u);
        splay(v);
        fa[T[v].son[0]]=fa[v];
        fa[v]=0;
        root[T[v].son[0]]=true;
        T[v].son[0]=0;
        pushup(v);
        return 0;
    }

    bool add(int u,int v,int val){
        makeroot(u);
        access(v);
        splay(v);
        Add(v,val);
        return 0;
    }

    bool change(int u,int v,int val){
        makeroot(u);
        access(v);
        splay(v);
        Change(v,val);
        return 0;
    }

    pair<int,int> ask(int u,int v){
        makeroot(u);
        access(v);
    }
}

```

板子 ???

```
splay(v);
return make_pair(T[v].ans,T[v].cnt2);
}
}T;
vector<int> edge[maxn];
void dfs(int x,int fa){
    T.fa[x]=fa;
    for (int v:edge[x]) if (v!=fa) dfs(v,x);
}
int n,m,TT;
int i,j,k;
int u,v;
int main(){
    int x=0;
    scanf("%d",&TT);
    while (TT--){
        scanf("%d%d",&n,&m);
        FOR(i,1,n){
            int val;
            scanf("%d",&val);
            T.T[i].init(val);
        }
        FOR(i,1,n) T.root[i]=1;
        REP(i,n-1){
            scanf("%d%d",&u,&v);
            edge[u].push_back(v);
            edge[v].push_back(u);
        }
        dfs(1,0);
        printf("Case #%d:\n",++x);
        while(m--){
            scanf("%d",&k);
            int x,y;
            if (k==1){
                int x0,y0;
                scanf("%d%d%d%d",&x,&y,&x0,&y0);
                T.cut(x,y);
                T.link(x0,y0);
            }else if (k==2){
                int val;
                scanf("%d%d%d",&x,&y,&val);
                T.change(x,y,val);
            }else if (k==3){
                int val;
                scanf("%d%d%d",&x,&y,&val);
```

```
T.add(x,y,val);
            }else if (k==4){
                scanf("%d%d",&x,&y);
                pair<int,int> t=T.ask(x,y);
                if (t.first==-INF) puts("ALL SAME");
                else printf("%d %d\n",t.first,t.second);
            }
        }
        FOR(i,1,n) edge[i].clear();
    }
}
```

KD 树

//线段树套 KD 树

//KD 树,对于子树需要维护区间

//时间复杂度:nsqrt(n)

```
namespace KDT {
    const double alpha=0.75;
    const int DIM=2;
    struct point {
        int A[DIM],max[DIM],min[DIM];
        int l,r; int size;
        void init() {
            l=r=0; initval();
        }
        void initval() {
            int i; size=1;
            REP(i,DIM) min[i]=max[i]=A[i];
        }
    } T[maxn*30]; int TOT;
    int Cur;
    bool cmp(int x,int y) {
        return T[x].A[Cur]<T[y].A[Cur];
    }
    void update(int x) {
        int i; T[x].initval();
        int l=T[x].l,r=T[x].r;
        if (l) T[x].size+=T[l].size;
        if (r) T[x].size+=T[r].size;
        REP(i,DIM) {
            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]);
            }
```

板子 ???

```
        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]);
        }
    }
}

int id[maxn],tot;
void build(int &x,int l,int r,int cur) { //should have id
    x=0; if (l>r) return;
    int m=(l+r)/2; Cur=cur;
    nth_element(id+l,id+m,id+r+1,cmp);
    x=id[m];
    build(T[x].l,l,m-1,cur^1);
    build(T[x].r,m+1,r,cur^1);
    update(x);
}

void getid(int x) { //没有顺序=_=
    id[++tot]=x;
    if (T[x].l) getid(T[x].l);
    if (T[x].r) getid(T[x].r);
}

void rebuild(int &x,int cur) {
    tot=0; getid(x);
    build(x,1,tot,cur);
}

void insert(int &x,int now,int cur) {
    if (!x) {x=now; return;}
    Cur=cur;
    if (cmp(now,x)) insert(T[x].l,now,cur^1);
    else insert(T[x].r,now,cur^1);
    update(x);
    if (T[x].size*alpha<max(T[T[x].l].size,T[T[x].r].size))
        rebuild(x,cur);
}

void addnode(int &x,int px,int py) {
    TOT++; T[TOT].A[0]=px; T[TOT].A[1]=py;
    T[TOT].init(); insert(x,TOT,0);
}

int x0,y0,x1,y1;//check 两个=_=
int check(int x,int y) {
    return x0<=x&&x<=x1&&y0<=y&&y<=y1;
}

int ok(point &A) {
    return check(A.A[0],A.A[1]);
}
```

```
int allin(point &A) {
    return  x0<=A.min[0]&&A.max[0]<=x1&&
           y0<=A.min[1]&&A.max[1]<=y1;
}

int allout(point &A) {
    return  A.max[0]<x0||x1<A.min[0]||
           A.max[1]<y0||y1<A.min[1];
}

int query(int x) {
    if (!x) return 0;
    if (allin(T[x])) return T[x].size;
    if (allout(T[x])) return 0;
    int ret=0;
    if (ok(T[x])) ret++;
    if (T[x].size==1) return ret;
    ret+=query(T[x].l);
    ret+=query(T[x].r);
    return ret;
}

const int MAX=1e9+7;
struct Tnode {
    int l,r,KD_root;
    Tnode() {l=r=KD_root=0;}
} T[maxn*30]; int cnt;
void update(int &x,int px,int py,int pos,int L,int R) {
    if (!x) x=++cnt;
    KDT::addnode(T[x].KD_root,px,py);
    if (L==R) return;
    int mid=(L+R)/2;
    if (pos<=mid) update(T[x].l,px,py,pos,L,mid);
    else update(T[x].r,px,py,pos,mid+1,R);
}

int query(int x,int k,int L,int R) {
    if (!x) return 0;
    if (L==R) return L;
    int mid=(L+R)/2;
    if (T[x].r) {
        int rk=KDT::query(T[T[x].r].KD_root);
        if (rk<k) return query(T[x].l,k-rk,L,mid);
        return query(T[x].r,k,mid+1,R);
    } return query(T[x].l,k,L,mid);
}

char buffer[36000000],*buf=buffer;
void read(int &x) {
```



```

    for (x=0; *buf<48; ++buf);
    while (*buf>=48)x=x*10+*buf-48,++buf;
}
int n,q;
int i,j,k;
int root,lastans;
int main() {
    fread(buffer,1,36000000,stdin);
    read(n); read(q); KDT::TOT=0;
    FOR(i,1,q) {
        int op;
        read(op);
        if (op==1) {
            int x,y,v;
            read(x); read(y); read(v);
            x^=lastans; y^=lastans; v^=lastans;
            update(root,x,y,v,0,MAX);
        } else {
            int x1,y1,x2,y2,k;
            read(x1); read(y1); read(x2); read(y2); read(k);
            x1^=lastans; y1^=lastans;
            x2^=lastans; y2^=lastans;
            k^=lastans;
            KDT::x0=x1; KDT::y0=y1;
            KDT::x1=x2; KDT::y1=y2;
            lastans=query(root,k,0,MAX);
            if (!lastans) puts("NAIVE!ORZzyz.");
            else printf("%d\n",lastans);
        }
    }
}

```

莫队

```

struct node{int l,r,id;}Q[maxn]; //new direction
int pos[maxn];
LL ans[maxn],flag[maxn];
int a[maxn];
bool cmp(node a,node b){
    if (pos[a.l]==pos[b.l]) return a.r<b.r;
    return pos[a.l]<pos[b.l];
}
int n,m,k; int i,j;
LL Ans;

```

板子 ???

```

int L=1,R=0;
void add(int x){
    Ans+=flag[a[x]^k];
    flag[a[x]]++; }
void del(int x){
    flag[a[x]]--;
    Ans-=flag[a[x]^k]; }
int main(){
    scanf("%d%d%d",&n,&m,&k);
    int sz=sqrt(n);
    FOR(i,1,n){
        scanf("%d",&a[i]);
        a[i]^=a[i-1];
        pos[i]=i/sz;
    }
    FOR(i,1,m){
        scanf("%d%d",&Q[i].l,&Q[i].r);
        Q[i].id=i;
    }
    sort(Q+1,Q+1+m,cmp);
    flag[0]=1;
    FOR(i,1,m){
        while (L<Q[i].l){del(L-1);L++;}
        while (L>Q[i].l){L--;add(L-1);}
        while (R<Q[i].r){R++;add(R);}
        while (R>Q[i].r){del(R);R--;}
        ans[Q[i].id]=Ans;
    }
    FOR(i,1,m) printf("%lld\n",ans[i]);
}

```

树上莫队(套分块)

//<http://codeforces.com/gym/100962/attachments>

//题意是求路径上最小没出现数字

//主要思路是分类,每个点进出各算一次可以消除影响

```

const int SIZE=500;
vector<pair<int,int> > edge[maxn];
int cl[maxn],cr[maxn],val[maxn],dfn[maxn<<1];
int tot;
int dfs(int x,int fa) {
    cl[x]=++tot; dfn[tot]=x;
    for (auto now:edge[x]) if (now.first!=fa) {
        dfs(now.first,x);
    }
}

```

```

        val[now.first]=now.second;
    } cr[x]=++tot; dfn[tot]=x;
}
int block[maxn<<1];
struct node {
    int l,r,id;
} Q[maxn];
int cmp(node a,node b) {
    if (block[a.l]==block[b.l]) return a.r<b.r;
    return block[a.l]<block[b.l];
}
bool vis[maxn];
int cnt[maxn],cur[maxn];//block,now
void change(int x) {
    x=dfn[x]; vis[x]^=1;
    if (vis[x]) {
        if (!cur[val[x]]) cnt[block[val[x]]]++;
        cur[val[x]]++;
    } else {
        cur[val[x]]--;
        if (!cur[val[x]]) cnt[block[val[x]]]--;
    }
}
int ans[maxn];
int L,R;
int main() {
    int n,q;
    int i;
    scanf("%d%d",&n,&q);
    FOR(i,0,n*2+1) block[i]=i/SIZE;
    REP(i,n-1) {
        int u,v,len;
        scanf("%d%d%d",&u,&v,&len); len=min(len,n+1);
        edge[u].push_back(make_pair(v,len));
        edge[v].push_back(make_pair(u,len));
    }
    val[1]=n+1; dfs(1,0);
    REP(i,q) {
        int a,b;
        scanf("%d%d",&a,&b);
        if (cl[a]>cl[b]) swap(a,b);
        if (cr[a]>cr[b]) Q[i].l=cl[a]+1,Q[i].r=cl[b];
        else Q[i].l=cr[a],Q[i].r=cl[b];
        Q[i].id=i;
    }
}

```

```

sort(Q,Q+q,cmp);
L=1; R=0;
REP(i,q) {
    while (L<Q[i].l) {change(L); L++;}
    while (R>Q[i].r) {change(R); R--;}
    while (L>Q[i].l) {L--; change(L);}
    while (R<Q[i].r) {R++; change(R);}
    int now=0;
    while (cnt[now]==SIZE) now++;
    now*=SIZE;
    while (cur[now]) now++;
    ans[Q[i].id]=now;
}
REP(i,q) printf("%d\n",ans[i]);
}

```

回滚莫队套分块

//北京区域赛

//分块_状态直接记录转移,比滚动要慢

//回滚分块(然而我没回滚,记录了一下)

//queries 按照左端点排序(有边的要按照我这种方式来排,否则菊花图会卡死)

//按右端点往右走,走到头即可

```

int SIZE;
struct node {
    int u,v,id,o;
    node() {}
    node(int _u,int _v,int _id=0):u(_u),v(_v),id(_id) {}
} to[maxn],re[maxn],queries[maxn];
int BID[maxn],L[maxn];
bool cmpu(node A,node B) {
    if (A.u!=B.u) return A.u<B.u;
    if (A.v!=B.v) return A.v>B.v;//为了避免漏掉
    return A.id>B.id;
} bool cmpv(node A,node B) {
    if (A.v!=B.v) return A.v<B.v;
    if (A.u!=B.u) return A.u<B.u;
    return A.id<B.id;
} bool cmpQ(node A,node B) {
    if (A.o!=B.o) return A.o<B.o;
    if (A.v!=B.v) return A.v<B.v;
    if (A.u!=B.u) return A.u<B.u;
    return A.id<B.id;
}

```

板子 ???

```
}
int fa[maxn],size[maxn];
LL Ans[maxn];
inline int getfa(int x) {
    if (fa[x]==x) return x;
    return fa[x]=getfa(fa[x]);
}
int FA[maxn],SZ[maxn],PID[maxn];
inline int getFA(int x) {
    if (FA[x]==x) return x;
    return FA[x]=getFA(FA[x]);
}
inline void update(int u,int pid) {
    if (PID[u]!=pid) {
        int f=getfa(u);
        if (PID[f]!=pid) {
            FA[f]=f;
            PID[f]=pid;
            SZ[f]=size[f];
        } PID[u]=pid; FA[u]=f;
    }
} int tot=0;
LL now;
int main() {
    int T;
    scanf("%d",&T);
    while (T--) {
        int n,m,q,i,j,k;
        scanf("%d%d%d",&n,&m,&q);
        FOR(i,0,(m+1)/SIZE) L[i]=0;
        FOR(i,1,m+1) { BID[i]=i/SIZE; if (!L[i/SIZE]) L[i/SIZE]=i; }
        if (q==0) SIZE=m; else SIZE=m/sqrt(q);
        if (!SIZE) SIZE++;
        FOR(i,1,m) {
            int u,v;
            scanf("%d%d",&u,&v);
            if (u>v) swap(u,v);
            to[i]=node(u,v);
            re[i]=node(u,v);
        } sort(to+1,to+m+1,cmpv);
        sort(re+1,re+m+1,cmpu);
        FOR(i,1,m) {
            to[i].o=BID[lower_bound(re+1,re+1+m,to[i].cmpu)-re];
            re[i].o=BID[i];
        }
```

```
}
FOR(i,1,q) {
    int u,v;
    scanf("%d%d",&u,&v);
    if (u>v) swap(u,v);
    queries[i]=node(u,v,i);
}
queries[i].o=BID[lower_bound(re+1,re+1+m,queries[i].cmpu)-re];
} sort(queries+1,queries+q+1,cmpQ);
FOR(i,1,q) {
    if (i==1||queries[i].o!=queries[i-1].o) { //initialize
        FOR(j,1,n) fa[j]=j,size[j]=1;
        j=1; now=0;
    }
    for (; j<=m&&to[j].v<=queries[i].v; j++) {
        if (to[j].o>queries[i].o) { //sorted by l
            node &e=to[j];
            int x=getfa(e.u),y=getfa(e.v);
            if (x==y) continue; fa[x]=y;
            now+=(LL)size[x]*size[y];
            size[y]+=size[x];
        }
    }
    LL ans=now; tot++;
    for (k=L[queries[i].o]; BID[k]==queries[i].o; k++) {
        if
        (queries[i].u<=re[k].u&&re[k].v<=queries[i].v) {
            node &e=re[k];
            update(e.u,tot); update(e.v,tot);
            int x=getFA(e.u),y=getFA(e.v);
            if (x==y) continue; FA[x]=y;
            ans+=(LL)SZ[x]*SZ[y];
            SZ[y]+=SZ[x];
        }
    }
    Ans[queries[i].id]=ans;
}
FOR(i,1,q) printf("%lld\n",Ans[i]);
}
```

带修改莫队

//change 常数大时 size 可以增大

```

//sort 时先 block,改变顺序可以降低常数
//n^2/3,注意常数
//注意 change 时间时排的顺序
const int SIZE=2500;
struct queries{
    int l,r,t;//pre
    queries(){};
    queries(int _l,int _r,int _t):l(_l),r(_r),t(_t){};
}Q[maxn],S[maxn];
int n,m,q;
int i,j,k;
int a[maxn];
int BLOCK[maxn];
bool cmp(queries &A,queries &B){
    if (BLOCK[A.l]!=BLOCK[B.l]) return BLOCK[A.l]<BLOCK[B.l];
    if (BLOCK[A.r]!=BLOCK[B.r]) return BLOCK[A.r]<BLOCK[B.r];
    return (A.t<B.t)^((BLOCK[A.l]^BLOCK[A.r])&1);
}vector<int> V;
inline int getid(int x){return lower_bound(V.begin(),V.end(),x)-
V.begin()+1;}
int L,R,T;
int num[maxn],cnt[maxn];
inline void add(int pos){
    int &T=num[a[pos]];
    cnt[T]--;T++;cnt[T]++;
}inline void del(int pos){
    int &T=num[a[pos]];
    cnt[T]--;T--;cnt[T]++;
}inline void change(int pos,int val){
    if (L<=pos&&pos<=R){del(pos),a[pos]=val,add(pos);}
    else a[pos]=val;
}
int ans[maxn];
int main(){
    scanf("%d%d",&n,&q);
    FOR(i,1,n) scanf("%d",&a[i]),V.push_back(a[i]);
    FOR(i,1,q){
        int op,l,r;
        scanf("%d%d%d",&op,&l,&r);
        if (op==1){
            Q[i]=queries(l,r,i);
        }if (op==2) {
            S[i]=queries(l,r,a[l]);a[l]=r;
            V.push_back(a[l]);
        }
    }
}

```

```

}sort(V.begin(),V.end());
V.erase(unique(V.begin(),V.end()),V.end());
FOR(i,1,n) a[i]=getid(a[i]);
FOR(i,1,q) if (S[i].t) S[i].r=getid(S[i].r),S[i].t=getid(S[i].t);
FOR(i,1,max(n,q)) BLOCK[i]=i/SIZE;
sort(Q+1,Q+q+1,cmp);
L=1;R=0;T=q;cnt[0]=INF;
FOR(i,1,q) if (Q[i].t){
    while (T<Q[i].t){T++;if (S[T].t) change(S[T].l,S[T].r);}
    while (T>Q[i].t){if (S[T].t) change(S[T].l,S[T].t);T--;}
    while (L<Q[i].l){del(L);L++;}
    while (R>Q[i].r){del(R);R--;}
    while (L>Q[i].l){L--;add(L);}
    while (R<Q[i].r){R++;add(R);}
    int now=0;
    while (cnt[now]) now++;
    ans[Q[i].t]=now;
}FOR(i,1,q) if (ans[i]) printf("%d\n",ans[i]);
}

```

维护凸包

/*这是抄的维护上半凸壳*/

```

bool Q;
struct Line {
    mutable LL a,b,k;
    bool operator<(const Line &o)const {
        return Q?k<o.k:a<o.a;
    }
};
struct convexHull:public multiset<Line> {
    LL div(LL a,LL b) {
        return a/b-((a^b)<0&&a%b);
    }
    bool getK(iterator x,iterator y) {
        if (y==end()) {x->k=INFF; return 0;}
        if (x->a==y->a) x->k=x->b>y->b?INFF:-INFF;
        else x->k=div(y->b-x->b,x->a-y->a);
        return x->k>=y->k;
    }
    void insPos(LL a,LL b) {
        auto z=insert({a,b,0}); auto y=z++,x=y;
        while (getK(y,z)) z=erase(z);
        if (y!=begin()&&getK(--x,y)) getK(x,erase(y));
    }
}

```

```

        while ((y=x)!=begin())&&(--x)->k>=y->k)
            getK(x,erase(y));
    }
    LL query(LL x) {
        assert(size());
        Q=1; auto now=lower_bound({0,0,x}); Q=0;
        return now->a*x+now->b;
    }
};
int n;
int i,j,k;
LL a[maxn],b[maxn];
LL ans[maxn];
convexHull A[maxn];
vector<int> edge[maxn];
void merge(int &x,int y) {
    if (A[x].size()<A[y].size()) swap(x,y);
    for (auto now:A[y]) A[x].insPos(now.a,now.b);
}
int dfs(int x,int fa) {
    int ret=x;
    for (auto u:edge[x]) if (u!=fa)
        merge(ret,dfs(u,x));
    if (A[ret].size()) ans[x]=-A[ret].query(a[x]);
    else ans[x]=0;
    A[ret].insPos(-b[x],-ans[x]);
    return ret;
}
int main() {
    scanf("%d",&n);
    FOR(i,1,n) scanf("%l64d",&a[i]);
    FOR(i,1,n) scanf("%l64d",&b[i]);
    REP(i,n-1) {
        int u,v;
        scanf("%d%d",&u,&v);
        edge[u].push_back(v);
        edge[v].push_back(u);
    } dfs(1,0);
    FOR(i,1,n) printf("%l64d ",ans[i]);
}

```

线性基(套路)

```

struct L_B {

```

板子 ???

```

LL A[63];bool have_0;
void clear(){memset(A,0,sizeof(A));have_0=0;}
LL XORMIN(LL x){
    int i;
    rREP(i,63) if ((A[i]^x)<x) x^=A[i];
    return x;
}
LL XORMAX(LL x){
    int i;
    rREP(i,63) if ((A[i]^x)>x) x^=A[i];
    return x;
}
void insert(LL x){
    int i;
    if (!have_0&&!XORMIN(x)) have_0=1;
    rREP(i,63) if ((x>>i)&1){
        if (!A[i]) A[i]=x;x^=A[i];
    }
}
void rebuild(){
    int i,j;
    rREP(i,63) rREP(j,i) if ((A[i]>>j)&1) A[i]^=A[j];
}
LL querykth(LL k){
    LL ret=0;int i;k-=have_0;
    REP(i,63) if (A[i]) {if(k&1) ret^=A[i];k>>=1;}
    if(k) return -1;
    return ret;
}
}A;

```

图论

二分图匹配

//最小不相交路径覆盖 \Leftrightarrow 节点数-拆点以后二分图最大匹配

//最小相交路径覆盖 \Leftrightarrow 所有能走到的节点连边，然后节点数-拆点以后

匹配

```
int n,m,i,j,k,t;
vector<int>edge[N];
int used[N];
int matching[N];
/*注意数组的标号，必须满足二分图的条件
bool dfs(int u){
    int v,i;
    REP(i,edge[u].size()){
        v=edge[u][i];
        if (!used[v]){
            used[v]=1;
            if (matching[v]==-1||dfs(matching[v])){
                matching[v]=u;
                matching[u]=v;
                return 1;
            }
        }
    }return 0;
}
int DFS(){
    int ans=0;
    memset(matching,-1,sizeof(matching));
    int u;
    FOR(u,1,n){
        if (matching[u]==-1){
            memset(used,0,sizeof(used));
            if (dfs(u)) ans++;
        }
    }return ans;
}*/
/*注意数组的标号，必须满足二分图的条件
queue<int> Q;
int prev[N];//两格
int check[N];//matchright
int BFS(){
    int ans=0;
    memset(matching,-1,sizeof(matching));
```

```
memset(check,-1,sizeof(check));
FOR(i,1,n){
    if (matching[i]==-1){
        while (!Q.empty()) Q.pop();
        Q.push(i);
        prev[i]=-1;
        bool flag=false;
        while (!Q.empty()&&!flag){
            int u=Q.front();Q.pop();
            for (j=0;!flag&&j<edge[u].size();j++){
                int v=edge[u][j];
                if (check[v]!=i){
                    check[v]=i;
                    Q.push(matching[v]);
                    if (matching[v]!=-1) prev[matching[v]]=u;
                }else{
                    flag=1;
                    int d=u,e=v;
                    while (d!=-1){
                        int t=matching[d];
                        matching[d]=e;
                        matching[e]=d;
                        d=prev[d];
                        e=t;
                    }
                }
            }
        }
    }
}
if (matching[i]!=-1) ans++;
}
}return ans;
}*/
int main(){
    int T;
    scanf("%d",&T);
    while (T--){
        scanf("%d%d",&n,&m);
        FOR(i,1,n){
            scanf("%d",&k);
            edge[i].clear();
            REP(j,k) scanf("%d",&t),edge[i].push_back(t+n);
        }
        if (BFS()==n) puts("YES");
```

```

        else puts("NO");
    }
}

```

最短路

Dijkstra (n^2) :

```

LL n,m,x;
LL a[N+2][N+2];
LL b[N+2];
bool vis[N+2];
LL i,j,k;
LL A,B,T;
int main()
{
    scanf("%lld%lld%lld",&n,&m,&x);
    FOR(i,n)
        FOR(j,n) a[i][j]=INF;
    FOR(i,m){
        scanf("%lld%lld%lld",&A,&B,&T);
        a[A][B]=T;
    }
    FOR(i,n) {b[i]=INF;vis[i]=0;}
    b[0]=INF;
    b[x]=0;
    int pos;
    FOR(i,n){
        pos=0;
        FOR(j,n) if (!vis[j]&&b[j]<b[pos]) pos=j;
        vis[pos]=1;
        FOR(j,n) if (!vis[j]&&b[pos]+a[pos][j]<b[j]) b[j]=b[pos]+a[pos][j];
    }
    FOR(i,n) printf("%lld ",b[i]);
}

```

Dijkstra (堆优化) :

```

struct node{
    int n,d;
    node(){}
    node(int a,int b):n(a),d(b){}
    bool operator<(const node&a)const{
        if (d==a.d) return n<a.n;
        return d>a.d;//注意 !!!
    }
};
vector<node> edge[maxn];//注意这里 priority_queue 是大根堆

```

板子 ???

```

int dis[maxn],n,m;
void dijkstra(int s){//DIJKSTRA+HEAP
    int i;
    FOR(i,1,n) dis[i]=INF;
    dis[s]=0;
    priority_queue<node> Q;
    Q.push(node(s,dis[s]));
    while (!Q.empty()){
        node x=Q.top();Q.pop();
        REP(i,edge[x.n].size()){
            node y=edge[x.n][i];
            if (dis[y.n]>x.d+y.d){
                dis[y.n]=x.d+y.d;
                Q.push(node(y.n,dis[y.n]));
            }
        }
    }
}

```

SPFA BFS

```

vector<node> edge[maxn];
int dis[maxn],n,m;
bool vis[maxn];
int sumnum[maxn];//judge negative ring
bool spfa(int s){
    int i;
    FOR(i,1,n) dis[i]=INF;
    FOR(i,1,n) vis[i]=0;
    FOR(i,1,n) sumnum[i]=0;//judge negative ring
    dis[s]=0;
    deque<int> Q;//slf need
    Q.push_back(s);
    // int sum=0;//lll
    while (!Q.empty()){
        int u=Q.front();Q.pop_front();
        // if (!Q.empty()&&sum/Q.size()<dis[u])
        Q.push_back(u);//lll
        // else {vis[u]=0; sum-=dis[u];};//lll
        vis[u]=0;//not lll
        REP(i,edge[u].size()){
            node v=edge[u][i];
            if (dis[u]+v.d<dis[v.n]){
                dis[v.n]=dis[u]+v.d;
                if (!vis[v.n]){
                    vis[v.n]=1;
                    if (Q.empty()||dis[Q.front()]<dis[v.n])

```

```

Q.push_back(v.n); //slf
else Q.push_front(v.n); //slf
Q.push_back(v.n); //not slf
// sumnum[v.n]++; //judge negative ring
// if (sumnum[v.n]>=n) return 1; //judge
negative ring
// sum+=dis[v.n]; //lll
}
}
}
// return 0; //judge negative ring
}
SPFA DFS(只用于判负环)
vector<node> edge[maxn];
int dis[maxn], n, m;
bool vis[maxn];
bool spfa(int u){
    int i;
    vis[u]=1;
    REP(i, edge[u].size()){
        node v=edge[u][i];
        if (dis[u]+v.d<dis[v.n]){
            dis[v.n]=dis[u]+v.d;
            if (vis[v.n]) return 1;
        }
        else {
            dis[v.n]=dis[u]+v.d;
            if (spfa(v.n)) return 1;
        }
    }
}
vis[u]=0;
return 0; //judge negative ring
}
int s, t;
int u, v, len;
int main(){
    int i, j, k;
    while (~scanf("%d%d", &n, &m)){
        FOR(i, 1, n) edge[i].clear();
        REP(i, m){
            scanf("%d%d%d", &u, &v, &len);
            edge[u].push_back(node(v, len));
            edge[v].push_back(node(u, len));
        }
    }
}

```

```

dij(1);
FOR(i, 2, n) printf("%d ", dis[i]==INF?-1:dis[i]);
puts("");
}
return 0;
}

```

差分约束系统

//主要在于建图

//连边 $u \rightarrow v, len \Leftrightarrow val(v) - val(u) \leq len$

//其他的都要化成这种形式 $int\ n, m;$

//最好 spfa!(可能负环)

int i, j;

struct node{

int n, d, next;

node(){}

node(int a, int b): n(a), d(b){}

bool operator<(const node &a) const{

if (d==a.d) return n<a.n;

return d>a.d;

}

} edge[150007];

int cnt=0;

int head[maxn];

void addedge(int u, int v, int len){

edge[cnt].n=v;

edge[cnt].d=len;

edge[cnt].next=head[u];

head[u]=cnt++;

};

int dis[maxn];

void dij(int s){

int i;

FOR(i, 1, n) dis[i]=INF;

dis[s]=0;

priority_queue<node> Q;

Q.push(node(s, dis[s]));

while (!Q.empty()){

node x=Q.top(); Q.pop();

for(i=head[x.n]; i!=-1; i=edge[i].next){

node &y=edge[i];

if (dis[y.n]>x.d+y.d){

dis[y.n]=x.d+y.d;

Q.push(node(y.n, dis[y.n]));


```

    }
}
}
int u,v,len;
int main(){
    while (~scanf("%d%d\n",&n,&m)){
        memset(head,0xff,sizeof(head));
        cnt=0;
        REP(i,m){
            scanf("%d%d%d",&u,&v,&len);
            //val(v)-val(u)<=len
            addedge(u,v,len);
        }
        dij(1);
        printf("%d\n",dis[n]);
    }
}

```

01 分数规划

//2017-harbin-K

//选出 k 个区间，使得这 k 个区间全覆盖，而且 sigmaA/sigmaB 最小

//俩 log dp TLE

//做法：建最短路，01 分数规划玄学过题

```

struct node{
    int n;
    double d;
    node(){}
    node(int _n,double _d):n(_n),d(_d){};
    bool operator<(const node&A)const{
        if (d==A.d) return n<A.n;
        return d>A.d;
    }
};
struct node_e{
    int n,A,B;
    double d;
    node_e(int _n,int _A,int _B,double _d):n(_n),A(_A),B(_B),d(_d){}
};
vector<node_e> edge[maxn];
int dis[maxn];
int preA[maxn],preB[maxn];
void dij(int s,int n){

```

```

    int i;
    FOR(i,1,n) dis[i]=INF;
    dis[s]=0;
    priority_queue<node> Q;
    Q.push(node(s,dis[s]));
    while (Q.size()){
        node x=Q.top();Q.pop();
        for (auto &y:edge[x.n]){
            if (dis[y.n]>x.d+y.d){
                dis[y.n]=x.d+y.d;
                Q.push(node(y.n,dis[y.n]));
                preA[y.n]=preA[x.n]+y.A;
                preB[y.n]=preB[x.n]+y.B;
            }
        }
    }
}
int n,t;
int S[maxn],T[maxn],A[maxn],B[maxn];
double check(double x){
    int i;double allA=0,allB=0;
    FOR(i,1,t+1)
        edge[i].clear();
    FOR(i,1,n){
        if (A[i]-B[i]*x<=0){
            allA+=A[i];allB+=B[i];
            edge[S[i]].emplace_back(node_e(T[i]+1,0,0,0));
        }else
            edge[S[i]].emplace_back(node_e(T[i]+1,A[i],B[i],A[i]-B[i]*x));
        }
    FOR(i,1,t)
        edge[i+1].emplace_back(node_e(i,0,0,0));
    dij(1,t+1);
    allA+=preA[t+1];allB+=preB[t+1];
    return allA/allB;
}
int main(){
    int i,j,m,x,T;
    scanf("%d",&T);
    while (_T--){
        scanf("%d",&n,&t);
        FOR(i,1,n)
            scanf("%d%d%d%d",&S[i],&T[i],&A[i],&B[i]);
        double ans=100;
        while (1){

```

```

double now=check(ans);
if (abs(now-ans)<0.001) break;
ans=now;
}
printf("%.3lf\n",ans);
}
return 0;
}

```

最小生成树(曼哈顿在后面)

//最小曼哈顿距离生成树在后面

//按照 45 度 4 个方向排序，最近的两个点连边即可

//最大曼哈顿距离生成树是维护最远的点的距离（四个方向的）

//Kruskal(有道分治题用的 Boruvka，和这个思想也类似)

//注意理解并查集的内涵，每次找最短的路也可以通过其他方式来找到

强连通分量 tarjin

```

struct Edge {
    int to,next;
    Edge(int _to=0,int _next=-1):to(_to),next(_next) {};
} edge[maxn*2];
int head[maxn],etot;
inline void addedge(int u,int v) {
    edge[++etot]=Edge(v,head[u]);
    head[u]=etot;
}
//lowlink 是说,遇到的 min
//无向图:
//u 割点:low[v]>=dfn[u];(表示能到的点都在之后)
//u-v 割边(桥):low[v]>dfn[u];(要在 u-v 处得到)
//块:low[u]==dfn[u];(最终从 stack 取出 x)
//dfs 时注意 fa 和重边处理
//无向图不用 vis 这个东西=_,vis 是为了避免横叉边
vector<int> nodes[maxn];
int cnt;
int dfn[maxn],low[maxn],tot;
bool vis[maxn]; //instack
int S[maxn],top;
int id[maxn];
void tarjan(int x,int fa) {
    low[x]=dfn[x]=++tot;
    S[++top]=x;

```

```

vis[x]=1;
for(int i=head[x]; ~i; i=edge[i].next) {
    int v=edge[i].to;
    if(v==fa) continue;
    if(!dfn[v]) {
        tarjan(v,x);
        low[x]=min(low[x],low[v]);
    } else if(vis[v])
        low[x]=min(low[x],dfn[v]);
}
if(low[x]==dfn[x]) {
    cnt++;
    while(1) {
        int now=S[top--];
        vis[now]=0;
        id[now]=cnt;
        nodes[cnt].push_back(now);
        if(now==x) break;
    }
}
}

int n,m;
int D[maxn],U[maxn],V[maxn];
set<pair<int,int> > H;
int ans,Ans;
int main() {
    int i;
    while(~scanf("%d%d",&n,&m)) {
        FOR(i,1,n) head[i]=-1,dfn[i]=0;
        FOR(i,1,cnt) D[i]=0;
        etot=tot=cnt=0;
        H.clear();
        FOR(i,1,m) {
            int u,v;
            scanf("%d%d",&u,&v);
            if(u>v) swap(u,v);
            if(H.count(make_pair(u,v))) continue;
            H.insert(make_pair(u,v));
            addedge(u,v);
            addedge(v,u);
            U[i]=u;
            V[i]=v;
        }
        Ans=0;
        tarjan(1,0);

```

```

// FOR(i,1,n) if (!dfn[i]) tarjan(i),Ans++;
FOR(i,1,m) if(id[U[i]]!=id[V[i]]) D[id[U[i]]]++,D[id[V[i]]]++;
FOR(i,1,tot) if(D[i]==1) Ans++;
printf("%d\n", (Ans+1)/2);
}
}

```

支配树

```

//lowlink 是说,遇到的 min
//无向图:
//u 割点:low[v]>=dfn[u];(表示能到的点都在之后)
//u-v 割边(桥):low[v]>dfn[u];(要在 u-v 处得到)
//块:low[u]==dfn[u];(最终从 stack 取出 x)
//dfs 时注意 fa 和重边处理
//有向图:
//DAG 上的割边:u-v:cnt[u]*cnt[v]==cnt[t](mod?)
//DAG 上的割边是固定的,也就是说求出来以后最短路是一样长的
//有环割边:将边变成点,然后跑支配树即可
//支配树:(注意,由于可能有到达不了的节点,初始化时注意答案更新)
//必经点(semi=mindep{通过非树枝边 fa})定理:(semi[x]=id[temp]),
//temp=min(temp,dfn[pre]),dfn[x]>dfn[pre](树枝边|前向边)
//temp=min(temp,dfn[semi[ancestor_pre(fa)]])
//dfn[x]<dfn[pre](横叉边|后向边)
//必经点(idom)定理:y=id[min{dfn[z]}],z:semi_path 上的点
//idom[x]=semi[x],semi[x]==semi[y]
//idom[x]=idom[y],semi[x]!=semi[y]
struct Edge {
    int to,next;
    Edge(int _to=0,int _next=-1):to(_to),next(_next) {};
} edge[maxn*4];
int head[maxn],pre[maxn],dom[maxn],etot; //edges
inline void addedge(int head[],int u,int v) {
    edge[++etot]=Edge(v,head[u]);
    head[u]=etot;
}
int dfn[maxn],tot,par[maxn]; //dfs-tree
int Fa[maxn],best[maxn]; //disjoint-set
int semi[maxn],id[maxn],idom[maxn]; //dom-tree
inline int getfa(int x) {
    if(Fa[x]==x) return x;
    int F=getfa(Fa[x]);
    if(dfn[semi[best[x]]]>dfn[semi[best[Fa[x]]]])
        best[x]=best[Fa[x]];
    return Fa[x]=F;
}

```

```

}
void dfs(int x) {
    dfn[x]=++tot;
    id[tot]=x;
    for(int i=head[x]; ~i; i=edge[i].next) {
        int v=edge[i].to;
        if(!dfn[v]) par[v]=x,dfs(v);
    }
}
void tarjan(int n) {
    int i;
    FOR(i,1,n) dom[i]=-1;
    FOR(i,1,n) best[i]=semi[i]=Fa[i]=i;
    rFOR(i,2,tot) {
        int x=id[i];
        for(int j=pre[x]; ~j; j=edge[j].next) {
            int v=edge[j].to;
            if(!dfn[v]) continue; //could not reach
            getfa(v); //pre_dfn: not changed
            if(dfn[semi[best[v]]]<dfn[semi[x]])
                semi[x]=semi[best[v]];
        }
        addedge(dom,semi[x],x);
        Fa[x]=par[x];
        x=id[i-1];
        for(int j=dom[x]; ~j; j=edge[j].next) { //path
            int v=edge[j].to;
            getfa(v); //id[min{dfn[z]}];
            if(semi[best[v]]==x) idom[v]=x;
            else idom[v]=best[v];
        }
    }
    FOR(i,2,tot) {
        int x=id[i];
        if(idom[x]!=semi[x]) idom[x]=idom[idom[x]];
    }
}
LL n,m;
LL CNT[maxn];
LL solve() {
    LL ret=(LL)tot*(tot-1)/2;
    int i;
    rFOR(i,2,tot) {
        int x=id[i];
        CNT[x]++;
    }
}

```

板子???

```
if(idom[x]==1) ret-=CNT[x]*(CNT[x]-1)/2;
else CNT[idom[x]]+=CNT[x];
}
return ret;
}
int main() {
    int i;
    scanf("%d%d",&n,&m);
    FOR(i,1,n) head[i]=pre[i]=-1;
    FOR(i,1,n) dfn[i]=id[i]=idom[i]=0;etot=tot=0;
    FOR(i,1,m) {
        int u,v;
        scanf("%d%d",&u,&v);
        addedge(head,u,v);
        addedge(pre,v,u);
    }
    dfs(1);
    tarjan(n);
    // FOR(i,1,n) printf("%2d ",par[i]);puts("");
    // FOR(i,1,n) printf("%2d ",id[i]);puts("");
    // FOR(i,1,n) printf("%2d ",idom[i]);puts("");
    printf("%lld\n",solve());
}
```

网络流

最大权闭合图

题意:给定一个有向图,每个点有权值,求最大权闭合图(与没选的没边相连),使得 $\sigma(val)$ 最大

做法: $S \rightarrow +node(val); -node \rightarrow T(-val)$;原边 $\rightarrow INF$,与 S 相连的最小割即为所求

原因:简单割 \Rightarrow 切的全是和 S, T 相连的边

假设最终与 S 相连的点正的 $x1$,负的 $y1$; T 的正的 $x2$,负的 $y2$, ($x2=S$ 切, $y1=T$ 切)

最小割 $C=S$ 切的正的 $+T$ 切的负的 $=x2+y1$ (即反过来)

要求的 $val=x1-y1$

$C+val=x1+x2=$ 定值, $val=x1+x2-C$

C 最小,即最大流

最大密度子图

这个是转化成权闭合图的做法:

二分答案

将边看成点

$S \rightarrow$ 边, 1

边 \rightarrow 连着的两点, 1

每个点 $\rightarrow T, val$

求完即可

因为 边 $-k \times \text{点} \geq 0$, 二分出这个即可得到答案

做法二:

$s \rightarrow$ 顶点, 权值 m

顶点之间连边, 权值 1

顶点 $\rightarrow T, m+2 \times \text{ans}-d[i]$ (度数)

满流就 OK

//DINIC+当前弧优化

```
struct node{
    int to,cap,next;
    node(int t=0,int c=0,int n=0):to(t),cap(c),next(n){}
}edge[maxn*50];
int head[maxn];
int tot;
void addedge(int from,int to,int cap){
    edge[tot].to=to;
    edge[tot].next=head[from];
    edge[tot].cap=cap;
    head[from]=tot++;
    edge[tot].to=from;
    edge[tot].next=head[to];
    edge[tot].cap=0;
    head[to]=tot++;
}
queue<int> Q;
bool vis[maxn];
int d[maxn];
int cur[maxn]; //当前弧优化
bool bfs(int s,int t){
    memset(vis,0,sizeof(vis));
    while (Q.size()) Q.pop();
    Q.push(s);
    d[s]=0;vis[s]=1;
    int i;
    while (!Q.empty()){
        int x=Q.front();Q.pop();
        for(i=head[x];i!=-1;i=edge[i].next){
            if (!vis[edge[i].to]&&edge[i].cap){
                vis[edge[i].to]=1;
                d[edge[i].to]=d[x]+1;
                Q.push(edge[i].to);
            }
        }
    }
}
```

```

    }
    return vis[t];
}

int dfs(int x,int t,int flow){
    if (x==t||flow==0) return flow;
    int i,ret=0,f;
    for (i=cur[x];i!=-1;i=edge[i].next){
        if (d[x]+1==d[edge[i].to]&&
            ((f=dfs(edge[i].to,t,min(flow,edge[i].cap)))>0)){
            edge[i].cap-=f;
            edge[i^1].cap+=f;
            ret+=f;
            flow-=f;
            cur[x]=i;
            if (flow==0) break;
        }
    }
    return ret;
}

int n,m,i;
int u,v,len,ans;
int s,t;
int main(){
    while (~scanf("%d%d",&n,&m)){
        memset(head,-1,sizeof(head));
        ans=0;tot=0;
        s=n+1;t=n+2;
        FOR(i,1,n){
            int a,b;
            scanf("%d%d",&a,&b);
            addedge(s,i,a);
            addedge(i,t,b);
        }
        FOR(i,1,m){
            scanf("%d%d%d",&u,&v,&len);
            addedge(u,v,len);
            addedge(v,u,len);
        }
        while (bfs(s,t)){
            int f;
            memcpy(cur,head,sizeof(head));
            while (f=dfs(s,t,INF)) ans+=f;
        }
        printf("%d\n",ans);
    }
}

```

```

    }
    //ISAP
    struct node{
        int to,cap,next;
        node(int t=0,int c=0,int n=0):to(t),cap(c),next(n){}
    }edge[maxn*50];
    int head[maxn];
    int tot;
    void addedge(int from,int to,int cap,int rcap=0){
        edge[tot].to=to;edge[tot].next=head[from];
        edge[tot].cap=cap;head[from]=tot++;
        edge[tot].to=from;edge[tot].next=head[to];
        edge[tot].cap=rcap;head[to]=tot++;
    }
    queue<int> Q;
    int gap[maxn],dep[maxn],cur[maxn];
    void bfs(int s,int t){
        memset(dep,0xff,sizeof(dep));
        memset(gap,0,sizeof(gap));
        gap[0]=1;
        dep[t]=0;
        Q.push(t);
        while (Q.size()){
            int u=Q.front();Q.pop();
            for(int i=head[u];i!=-1;i=edge[i].next){
                int v=edge[i].to;
                if (dep[v]!=-1) continue;
                Q.push(v);
                dep[v]=dep[u]+1;
                gap[dep[v]]++;
            }
        }
    }
    int S[maxn];
    int sap(int s,int t,int n){
        bfs(s,t);
        memcpy(cur,head,sizeof(head));
        int top=0,u=s,ret=0;
        while (dep[s]<n){
            if (u==t){
                int MIN=INF,inser,i;
                REP(i,top) if (MIN>edge[S[i]].cap)
                    MIN=edge[S[i]].cap,inser=i;
                REP(i,top) edge[S[i]].cap-=MIN,
                    edge[S[i]^1].cap+=MIN;
            }
        }
    }
}

```

```

        ret+=MIN;
        top=insert;
        u=edge[S[top]^1].to;
        continue;
    }
    bool flag=0;
    int v;
    for (int i=cur[u];i!=-1;i=edge[i].next){
        v=edge[i].to;
        if (edge[i].cap&&dep[v]+1==dep[u]){
            flag=1;
            cur[u]=i;
            break;
        }
    }
    if (flag){
        S[top++]=cur[u];
        u=v;
        continue;
    }
    int MIN=n;
    for (int i=head[u];i!=-1;i=edge[i].next){
        v=edge[i].to;
        if (edge[i].cap&&dep[v]<MIN)
            MIN=min(MIN,dep[v]),cur[u]=i;
    }
    gap[dep[u]]--;
    if (!gap[dep[u]]) return ret;
    dep[u]=MIN+1;
    gap[dep[u]]++;
    if (u!=s) u=edge[S[--top]^1].to;
}
return ret;
}
int n,m,s,t;
int main(){
    while (~scanf("%d%d",&n,&m)){
        memset(head,-1,sizeof(head));
        tot=0;
        s=n+1;t=n+2;
        int i;
        FOR(i,1,n){
            int a,b;
            scanf("%d%d",&a,&b);
            addedge(s,i,a);

```

```

            addedge(i,t,b);
        }
        FOR(i,1,m){
            int u,v,len;
            scanf("%d%d%d",&u,&v,&len);
            addedge(u,v,len,len);
        }
        printf("%d\n",sap(s,t,n+2));
    }
}

```

最小费用流

//拆点后可以 S 向入连边,出向 T 连边,然后入和出就可以保持动态平衡

//注意观察特殊性质

```

struct node {
    LL to,cap,cost,rev;
    node(int t=0,int c=0,int n=0,int r=0):
        to(t),cap(c),cost(n),rev(r) {}
};
vector<node> edge[maxn];
void addedge(int from,int to,LL cap,LL cost) {
    edge[from].push_back(node(to,cap,cost,edge[to].size()));
    edge[to].push_back(node(from,0,-cost,edge[from].size()-1));
}
int n,m,V;
LL dis[maxn];
bool mark[maxn];
int pre_v[maxn],pre_e[maxn];
deque<int> Q;
pair<LL,LL> mincostflow(int s,int t,LL f) {
    LL ret=0,d;
    int i,v;
    while (f) {
        memset(dis,0x3f,sizeof(dis));
        memset(mark,0,sizeof(mark));
        while (Q.size()) Q.pop_front();
        dis[s]=0;
        Q.push_back(s);
        while (Q.size()) {
            v=Q.front();
            mark[v]=0;
            Q.pop_front();
            REP(i,edge[v].size()) {

```

板子 ???

```
node &e=edge[v][i];
if (e.cap>0&&dis[e.to]>dis[v]+e.cost) {
    dis[e.to]=dis[v]+e.cost;
    pre_v[e.to]=v;
    pre_e[e.to]=i;
    if (!mark[e.to]) {
        if (Q.empty()||dis[Q.front()]<dis[e.to])
            Q.push_back(e.to);
        else Q.push_front(e.to);
        mark[e.to]=1;
    }
}
}
}
if (dis[t]==INFF) break;
d=f;
for (v=t; v!=s; v=pre_v[v])
    d=min(d,edge[pre_v[v]][pre_e[v]].cap);
f-=d;
ret+=d*dis[t];
for (v=t; v!=s; v=pre_v[v]) {
    node &e=edge[pre_v[v]][pre_e[v]];
    e.cap-=d;
    edge[v][e.rev].cap+=d;
}
if (d==0) break;
}
return make_pair(INFF-f,ret);
}
int i,j,k;
int main() {
    scanf("%d%d",&n,&m);
    FOR(i,1,m) {
        LL u,v,c,w;
        scanf("%lld%lld%lld%lld",&u,&v,&c,&w);
        addedge(u,v,c,w);
    }
    V=n;
    pair<LL,LL> ans=mincostflow(1,n,INFF);
    printf("%lld %lld",ans.first,ans.second);
}
```

上下界网络流

//可二分 $t \rightarrow s$ 边的下/上界,即可达到最大最小流
//最大流: $t \rightarrow s$ 连边, $ss \rightarrow tt$ 流, $s \rightarrow t$ 正向最大流,会流掉反向建的
边的流量
//最小流: $ss \rightarrow tt$ 流, $t \rightarrow s$ 连边, $ss \rightarrow tt$ 流

```
int n,m,q;
int i,j,k;
int ss,tt;
struct node {
    int to,cap,next;
    node(int _to=0,int _cap=0,int _next=-1):
        to(_to),cap(_cap),next(_next) {}
} edge[maxn*3];
int tot;
int head[307];
int addedge(int from,int to,int cap) {
    edge[tot]=node(to,cap,head[from]);
    head[from]=tot++;
    edge[tot]=node(from,0,head[to]);
    head[to]=tot++;
    return tot-1;//反的边 cap=正的 flow
}
bool vis[307];
int d[307];
queue<int> Q;
bool bfs(int s,int t) {
    memset(vis,0,sizeof(vis));
    while (Q.size()) Q.pop();
    Q.push(s);
    d[s]=0; vis[s]=1;
    int i;
    while (Q.size()) {
        int x=Q.front(); Q.pop();
        for (i=head[x]; i!=-1; i=edge[i].next) {
            if (!vis[edge[i].to]&&edge[i].cap) {
                vis[edge[i].to]=1;
                d[edge[i].to]=d[x]+1;
                Q.push(edge[i].to);
            }
        }
    }
    return vis[t];
}
int cur[307]; //当前弧优化
int dfs(int x,int t,int flow) { //dinic
    if (x==t||flow==0) return flow;
```

板子 ???

```
int i,ret=0,f;
for (i=cur[x]; i!=-1; i=edge[i].next) {
    if (d[x]+1==d[edge[i].to]&&
        (f=dfs(edge[i].to,t,min(flow,edge[i].cap)))>0) {
        edge[i].cap-=f;
        edge[i^1].cap+=f;
        ret+=f;
        flow-=f;
        cur[x]=i;
        if (flow==0) break;
    }
}
return ret;
}

int in[307],out[307];
int add(int u,int v,int low,int high) {
    int ret=addedge(u,v,high-low);
    out[u]+=low; in[v]+=low;
    return ret;
}

int sum,flow,E[maxn],ans[maxn]; //E 为对应的边位置
int solve() {
    memset(head,0xff,sizeof(head));
    memset(in,0,sizeof(in));
    memset(out,0,sizeof(out));
    scanf("%d%d",&n,&m);
    flow=0; sum=0; tot=0;
    FOR(i,1,m) {
        int u,v,low,high;
        scanf("%d%d%d%d",&u,&v,&low,&high);
        ans[i]=low;
        E[i]=add(u,v,low,high); //E[i]很有用
    }
    ss=n+1; tt=n+2;
    FOR(i,1,n) {
        sum+=max(in[i]-out[i],0);
        if (in[i]>out[i]) addedge(ss,i,in[i]-out[i]);
        if (in[i]<out[i]) addedge(i,tt,out[i]-in[i]);
    }
    while (bfs(ss,tt)) {
        int f;
        memcpy(cur,head,sizeof(head));
        while (f=dfs(ss,tt,INF)) flow+=f;
    }
    if (flow!=sum) return 0*puts("NO");
```

```
else {
    puts("YES");
    FOR(i,1,m) {
        ans[i]+=edge[E[i]].cap;
        printf("%d\n",ans[i]);
    }
}

int main() {
    int T;
    scanf("%d",&T);
    while (T--) {
        solve();
    }
}
```

上下界费用流

// Hihocoder 1424, 限制很多的一道题, 只是留板子

```
struct node{
    LL to,cap,cost,rev;
    node(int t=0,int c=0,int n=0,int
r=0):to(t),cap(c),cost(n),rev(r){}
};
vector<node> edge[maxn];
void addedge(int from,int to,LL cap,LL cost){
    edge[from].push_back(node(to,cap,cost,edge[to].size()));
    edge[to].push_back(node(from,0,-cost,edge[from].size()-1));
}
LL dis[maxn];
bool mark[maxn];
int pre_v[maxn],pre_e[maxn];
deque<int> Q;
pair<int,int> mincostflow(int s,int t,int f){
    int ret=0,d;
    int i,v;
    while (f){
        memset(dis,0x3f,sizeof(dis));
        memset(mark,0,sizeof(mark));
        while (Q.size()) Q.pop_front();
        dis[s]=0;Q.push_back(s);
        while (Q.size()){
            v=Q.front();mark[v]=0;Q.pop_front();
            REP(i,edge[v].size()){
                node &e=edge[v][i];
```



```

        if (e.cap>0&&dis[e.to]>dis[v]+e.cost){
            dis[e.to]=dis[v]+e.cost;
            pre_v[e.to]=v;
            pre_e[e.to]=i;
            if (!mark[e.to]){
                if (Q.empty()||dis[Q.front()]<dis[e.to])
Q.push_back(e.to);

                else Q.push_front(e.to);
                mark[e.to]=1;
            }
        }
    }
}
if (dis[t]==INF) break;
d=f;
for (v=t;v!=s;v=pre_v[v])
    d=min(d,edge[pre_v[v]][pre_e[v]].cap);
f-=d;
ret+=d*dis[t];
for (v=t;v!=s;v=pre_v[v]){
    node &e=edge[pre_v[v]][pre_e[v]];
    e.cap-=d;
    edge[v][e.rev].cap+=d;
}
if (d==0) break;
}
return make_pair(INF-f,ret);
}

int n,m;
int i,j;
int VAL[57][57];
int addrow[57][57];
int addcol[57][57];
int row[57],col[57];
int in[maxn],out[maxn];
int u,v;
int s,t,S,T;
int tot;
int sum;
void add(int u,int v,int low,int high,int cost){
    addedge(u,v,high-low,cost);
    out[u]+=low;in[v]+=low;
}
void solve(int n){

```

```

tot=0;
FOR(i,1,n) row[i]=++tot;
FOR(i,1,n) col[i]=++tot;
s=++tot;t=++tot;
S=++tot;T=++tot;
FOR(i,1,n)
    FOR(j,1,n) scanf("%d",&VAL[i][j]);
FOR(i,1,n){
    int cnt=0;
    FOR(j,1,n) cnt+=VAL[i][j];
    add(s,row[i],cnt,cnt,0);
    cnt=0;
    FOR(j,1,n) cnt+=VAL[j][i];
    add(s,col[i],cnt,cnt,0);
}
FOR(i,1,n){
    int l,r;
    scanf("%d%d",&l,&r);
    add(row[i],t,l,r,0);
}
FOR(i,1,n){
    int l,r;
    scanf("%d%d",&l,&r);
    add(col[i],t,l,r,0);
}
FOR(i,1,n)
    FOR(j,1,n) addrow[i][j]=addcol[i][j]=0;
REP(i,n*n/2){
    int x0,y0,x1,y1;
    scanf("%d%d%d%d",&x0,&y0,&x1,&y1);
    if (VAL[x0][y0]==VAL[x1][y1]) continue;
    if (VAL[x0][y0]==1){
        if (y0==y1) addrow[x0][x1]++;
        else addcol[y0][y1]++;
    }else if (VAL[x1][y1]==1){
        if (y0==y1) addrow[x1][x0]++;
        else addcol[y1][y0]++;
    }
}
FOR(i,1,n){
    FOR(j,1,n){
        if (addrow[i][j]) add(row[i],row[j],0,addrow[i][j],1);
        if (addcol[i][j]) add(col[i],col[j],0,addcol[i][j],1);
    }
}
}

```

```

sum=0;
add(t,s,0,INF,0);
FOR(i,1,tot){
    sum+=max(in[i]-out[i],0);
    if (in[i]>out[i]) addedge(S,i,in[i]-out[i],0);
    if (in[i]<out[i]) addedge(i,T,out[i]-in[i],0);
}
pair<int,int> now=mincostflow(S,T,INF);
if (now.first!=sum) puts("-1");
else printf("%d\n",now.second);
FOR(i,1,tot) edge[i].clear();
FOR(i,1,tot) in[i]=out[i]=0;
}
int main()
{
    while (~scanf("%d",&n)) solve(n);
}

```

树分治

//乘积立方数个数，如果是 sum 直接枚举其实就好

//树分支正反各 dfs 一次可以正常求出经过一点的 cnt

```

LL K;
LL MUL[37];
LL getSum(LL x,LL y){
    LL ret=0,i;
    REP(i,K) ret=ret+(x/MUL[i]%3+y/MUL[i]%3)%3*MUL[i];
    return ret;
}
LL getDiv(LL x){
    LL ret=0,i;
    REP(i,K) ret=ret+(3-x/MUL[i]%3)%3*MUL[i];
    return ret;
}
LL color[maxn];
vector<int> edge[maxn];
LL ans;
int size[maxn];
bool mark[maxn];
int minweight,root;
void dfs1(int x,int fa,int n){
    int weight=0;
    size[x]=1;
    for (int v:edge[x]){
        if (v==fa||mark[v]) continue;

```

板子 ???

```

        dfs1(v,x,n);
        size[x]+=size[v];
        weight=max(weight,size[v]);
    }
    weight=max(weight,n-size[x]);
    if (weight<minweight) {root=x;minweight=weight;}
}
map<LL,int> now;
map<LL,int> MP;
void dfs2(int x,int fa,LL num){
    now[getSum(color[x],num)]++;
    for (int v:edge[x]){
        if (v==fa||mark[v]) continue;
        dfs2(v,x,getSum(num,color[x]));
    }
}
void calc(int x){
    MP.clear();
    MP[color[x]]++;
    for (int u:edge[x]){
        if (mark[u]) continue;
        now.clear();
        dfs2(u,0,0);
        for(pair<LL,int> P:now) ans+=MP[getDiv(P.first)]*P.second;
        for(pair<LL,int> P:now) MP[getSum(color[x],P.first)]+=P.second;
    }
    MP.clear();
}
void dfs3(int x){
    mark[x]=1;
    calc(x);
    for (int v:edge[x]){
        if (mark[v]) continue;
        minweight=size[v];
        dfs1(v,0,size[v]);
        dfs3(root);
    }
}
int n,m;
LL C[maxn];
LL P;
int main(){
    int i,j;
    MUL[0]=1;
    FOR(i,1,33) MUL[i]=MUL[i-1]*3;

```

板子 ???

```
while (~scanf("%d",&n)){
    ans=0;
    scanf("%d",&K);
    REP(i,K) scanf("%lld",&C[i]);
    FOR(i,1,n){
        scanf("%lld",&P);
        REP(j,K){
            int t=0;
            while (P%C[j]==0){
                P/=C[j];
                t++;
            }
            if (t==3) t=0;
            color[i]+=MUL[j]*t;
        }
        if (color[i]==0) ans++;
    }
    REP(i,n-1){
        int u,v;
        scanf("%d%d",&u,&v);
        edge[u].push_back(v);
        edge[v].push_back(u);
    }
    minweight=n;
    dfs1(1,0,n);
    dfs3(root);
    printf("%lld\n",ans);
    FOR(i,1,n) mark[i]=0;
    FOR(i,1,n) color[i]=0;
    FOR(i,1,n) vector<int>().swap(edge[i]);
}
}
```

部分树上 dp

从求含某条边的最小生成树截下来的代码(当然前面 sort 了)合并(要记得 merge 咋写),先 sort 然后从小到大讨论

```
inline int Union(int u,int v,int len){
    int ret=0;
    while (u!=v&&(fa[u]!=u||fa[v]!=v)){
        if (fa[u]==u||fa[v]!=v&&sz[u]>sz[v]) {ret=max(ret,val[v]);v=fa[v];}
        else {ret=max(ret,val[u]);u=fa[u];}
    }
    if (u==v) return ret;
    if (sz[u]>sz[v]) swap(u,v);
```

```
fa[u]=v;val[u]=len;
sz[v]+=sz[u];ans=ans+len;
return len;
}
树上距离除 k 向上取整
LL count[maxn][6];
vector<int> edge[maxn];
LL num[maxn],cnt[maxn]; // 端点,满足条件的次数
int k;
LL ans;
void dfs(int u,int from){
    int i,j,c1,c2;
    count[u][0]=1;
    cnt[u]=1;
    REP(i,edge[u].size()){
        int v=edge[u][i];
        if (from==v) continue;
        dfs(v,u);
        REP(c1,k){
            REP(c2,k){
                ans+=count[u][c1]*count[v][c2];
                if (c1+c2+1>k) ans+=count[u][c1]*count[v][c2];
            }
            ans+=cnt[u]*num[v]+num[u]*cnt[v];
            num[u]+=num[v]+count[v][k-1];
            cnt[u]+=cnt[v];
            REP(c1,k) count[u][c1]+=count[v][(c1-1+k)%k];
        }
    }
}
```

2-sat

//重点是维护拆点后各种限制之间的关系,这个是个二分以后 2-sat 的

```
struct T_SAT {
    struct enode {
        int to,next;
        enode(int _to=0,int _next=-1):to(_to),next(_next) {};
    } edge[maxn*maxn*2];
    int head[maxn*2],etot;
    void addedge(int u,int v) {
        edge[++etot]=enode(v,head[u]);
        head[u]=etot;
    }
    int dfn[maxn*2],low[maxn*2],belong[maxn*2];
```

```

bool vis[maxn*2];
int tot,cnt;
int S[maxn*2],top;
void dfs(int x) {
    dfn[x]=low[x]=++tot;
    S[++top]=x;
    vis[x]=1;
    for (int i=head[x]; ~i; i=edge[i].next) {
        int v=edge[i].to;
        if (!dfn[v]) {
            dfs(v);
            low[x]=min(low[x],low[v]);
        } else if (vis[v])
            low[x]=min(low[x],dfn[v]);
    }
    if (dfn[x]==low[x]) {
        cnt++;
        while (1) {
            int now=S[top--];
            vis[now]=0;
            belong[now]=cnt;
            if (now==x) break;
        }
    }
}
void init(int n) {
    int i;
    REP(i,2*n) head[i]=-1;
    etot=0;
}
bool solve(int n) {
    int i;
    tot=cnt=0;
    REP(i,2*n) dfn[i]=vis[i]=0;
    REP(i,2*n) if (!dfn[i]) dfs(i);
    REP(i,n) if (belong[i]==belong[i+n]) return 0;
    return 1;
}
} two_sat;
int n,m;
int i,j;
int a1,a2,c1,c2;
int main() {
    while (~scanf("%d%d",&n,&m)) {
        two_sat.init(n);

```

```

REP(i,m) {
    scanf("%d%d%d%d",&a1,&a2,&c1,&c2);
    if (c1==1&&c2==1) {
        two_sat.addedge(a1+n,a2);
        two_sat.addedge(a2+n,a1);
    } else if (c1==0&&c2==1) {
        two_sat.addedge(a1,a2);
        two_sat.addedge(a2+n,a1+n);
    } else if (c1==1&&c2==0) {
        two_sat.addedge(a1+n,a2+n);
        two_sat.addedge(a2,a1);
    } else if (c1==0&&c2==0) {
        two_sat.addedge(a1,a2+n);
        two_sat.addedge(a2,a1+n);
    }
}
if (two_sat.solve(n)) puts("YES");
else puts("NO");
}
}

```

dfs 序

//常用方法：时间戳、莫队、拆开操作

```

void dfs(int u,int from){
    int v,i;
    in[u]=++tot;
    REP(i,edge[u].size()){
        v=edge[u][i];
        if (v==from) continue;
        dfs(v,u);
    }
    out[u]=tot;
}

```

dfs 序_换根的讨论 233

//<http://codeforces.com/contest/916/problem/E>

//改根,子树加,查,令人窒息的讨论

```

LL sum[maxn<<2],lazy[maxn<<2];
void update(int x,int l,int r,LL val,int L,int R) {
    if (l>r) return;
    if (l<=L&&R<=r) {lazy[x]+=val; sum[x]+=(R-L+1)*val; return;}
    int mid=(L+R)/2;
    if (lazy[x]) {

```

```

    lazy[x<<1]+=lazy[x];
    lazy[x<<1|1]+=lazy[x];
    sum[x<<1]+=(mid-L+1)*lazy[x];
    sum[x<<1|1]+=(R-mid)*lazy[x];
    lazy[x]=0;
}
if (l<=mid) update(x<<1,l,r,val,L,mid);
if (mid<r) update(x<<1|1,l,r,val,mid+1,R);
sum[x]=sum[x<<1]+sum[x<<1|1];
}
LL query(int x,int l,int r,int L,int R) {
    LL ret=0;
    if (l>r) return 0;
    if (l<=L&&R<=r) return sum[x];
    int mid=(L+R)/2;
    if (lazy[x]) {
        lazy[x<<1]+=lazy[x];
        lazy[x<<1|1]+=lazy[x];
        sum[x<<1]+=(mid-L+1)*lazy[x];
        sum[x<<1|1]+=(R-mid)*lazy[x];
        lazy[x]=0;
    }
    if (l<=mid) ret+=query(x<<1,l,r,L,mid);
    if (mid<r) ret+=query(x<<1|1,l,r,mid+1,R);
    sum[x]=sum[x<<1]+sum[x<<1|1];
    return ret;
}
vector<int> edge[maxn];
int fa[maxn][27];
int in[maxn],out[maxn],tot,dep[maxn];
void dfs(int x,int f,int d) {
    int i;
    fa[x][0]=f;
    in[x]=++tot;
    dep[x]=d;
    rep(i,1,20) fa[x][i]=fa[fa[x][i-1]][i-1];
    for (int v:edge[x]) if (v!=f) dfs(v,x,d+1);
    out[x]=tot;
}
int lca(int x,int y) {
    int i;
    if (dep[x]<dep[y]) swap(x,y);
    rREP(i,20) if (dep[x]-dep[y]>=1<<i) x=fa[x][i];
    if (x==y) return x;
    rREP(i,20) if (fa[x][i]!=fa[y][i]) x=fa[x][i],y=fa[y][i];

```

```

    return fa[x][0];
}
int getnthfa(int x,int k) {
    int i;
    rREP(i,20) if ((k>>i)&1) x=fa[x][i];
    return x;
}
int root;
int n,m;
int a[maxn];
int main() {
    int i,j;
    scanf("%d%d",&n,&m);
    FOR(i,1,n) scanf("%d",&a[i]);
    FOR(i,1,n-1) {
        int u,v;
        scanf("%d%d",&u,&v);
        edge[u].push_back(v);
        edge[v].push_back(u);
    }
    dfs(1,0,0);
    FOR(i,1,n) update(1,in[i],in[i],a[i],1,n);
    root=1;
    while (m--) {
        int op,u,v,x;
        scanf("%d",&op);
        if (op==1) {
            scanf("%d",&root);
        } else if (op==2) {
            scanf("%d%d%d",&u,&v,&x);
            int f=lca(u,v)^lca(v,root)^lca(u,root);
            if (f==root) update(1,1,n,x,1,n);
            else if (lca(f,root)==f) {
                int t=getnthfa(root,dep[root]-dep[f]-1);
                update(1,1,in[t]-1,x,1,n);
                update(1,out[t]+1,n,x,1,n);
            } else update(1,in[f],out[f],x,1,n);
        } else if (op==3) {
            int x;
            LL ans;
            scanf("%d",&x);
            if (x==root) ans=query(1,1,n,1,n);
            else if ((in[x]<=in[root]&&in[root]<=out[x]) {
                int t=getnthfa(root,dep[root]-dep[x]-1);
                ans=query(1,1,in[t]-1,1,n)+query(1,out[t]+1,n,1,n);
            }

```

```

    } else ans=query(1,in[x],out[x],1,n);
    printf("%lld\n",ans);
}
}
}

```

树链剖分

难题(区间合并)

```

int tot;
struct node{
    int lval,rval,lldown,lup,rldown,rup,upmx,downmx;
    node():upmx(0),downmx(0){};
}tree[maxn<<2];
int a[maxn];
node merge(node L,node R){
    if (L.upmx==0) return R;
    if (R.upmx==0) return L;
    node ret;
    ret.upmx=max(L.upmx,R.upmx);
    ret.downmx=max(L.downmx,R.downmx);
    ret.lval=L.lval;
    ret.lup=L.lup;
    ret.lldown=L.lldown;
    ret.rval=R.rval;
    ret.rup=R.rup;
    ret.rldown=R.rldown;
    if (L.rval<R.lval){
        ret.upmx=max(ret.upmx,L.rup+R.lup);
        if (L.downmx==1) ret.lup=L.lup+R.lup;
        if (R.downmx==1) ret.rup=L.rup+R.rup;
    }
    if (L.rval>R.lval){
        ret.downmx=max(ret.downmx,L.rldown+R.lldown);
        if (L.upmx==1) ret.lldown=L.lldown+R.lldown;
        if (R.upmx==1) ret.rldown=L.rldown+R.rldown;
    }
    return ret;
}
void build(int x,int l,int r){
    if (l==r){
        tree[x].lval=tree[x].rval=a[l];
        tree[x].lup=tree[x].lldown=tree[x].rup=tree[x].rldown=tree[x].upmx=tree[x].downmx=1;
    }
    return;
}

```

```

}
int mid=(l+r)/2;
build(x<<1,l,mid);
build(x<<1|1,mid+1,r);
tree[x]=merge(tree[x<<1],tree[x<<1|1]);
}
node query(int x,int l,int r,int L,int R){
    node ret;
    if (l<=L&&R<=r) return tree[x];
    int mid=(L+R)/2;
    if (mid>=l&&r>mid) return
        merge(query(x<<1,l,r,L,mid),query(x<<1|1,l,r,mid+1,R));
    if (mid>=l) return query(x<<1,l,r,L,mid);
    return query(x<<1|1,l,r,mid+1,R);
}
int n,i,j,q;
int u,v;
vector<int> edge[maxn];
int fa[maxn],son[maxn],top[maxn],dep[maxn],id[maxn],sz[maxn];
int b[maxn];
void dfs1(int u,int depth){
    int v,i,mx=-1;
    son[u]=0;sz[u]=1;dep[u]=depth;
    REP(i,edge[u].size()){
        v=edge[u][i];
        dfs1(v,depth+1);
        sz[u]+=sz[v];
        if (sz[v]>mx) mx=sz[v],son[u]=v;
    }
}
void dfs2(int u,int x){
    int v,i;
    top[u]=x;id[u]=++tot;
    if (son[u]) dfs2(son[u],x);
    REP(i,edge[u].size()){
        v=edge[u][i];
        if (v==fa[u]||v==son[u]) continue;
        dfs2(v,v);
    }
}
int Query(int x,int y){//这里需要注意方向
    node up,down;
    int ret,mark1=0,mark2=0;
    while (top[x]!=top[y]){
        if (dep[top[x]]>dep[top[y]]){

```

```

        up=merge(query(1,id[top[x]],id[x],1,tot),up);
        x=fa[top[x]];
        mark1=1;
    }else {
        down=merge(query(1,id[top[y]],id[y],1,tot),down);
        y=fa[top[y]];
        mark2=1;
    }
}
if (dep[x]>dep[y]) up=merge(query(1,id[y],id[x],1,tot),up),mark1=1;
else down=merge(query(1,id[x],id[y],1,tot),down),mark2=1;
ret=max(up.downmx,down.upmx);
if (mark1&&mark2&&up.lval<down.lval)
    ret=max(ret,up.ldown+down.lup);
return ret;
}
int T,t;
int main(){
    scanf("%d",&T);
    FOR (t,1,T){
        scanf("%d",&n);
        FOR(i,1,n) edge[i].clear();tot=0;
        FOR(i,1,n) scanf("%d",&b[i]);
        FOR(i,2,n){scanf("%d",&fa[i]); edge[fa[i]].push_back(i);}
        dfs1(1,1);
        dfs2(1,1);
        FOR(i,1,n) a[id[i]]=b[i];
        build(1,1,tot);
        scanf("%d",&q);
        printf("Case #%d:\n",t);
        while (q--){
            scanf("%d%d",&u,&v);
            printf("%d\n",Query(u,v));
        }
        if (t!=T) puts("");
    }
}

```

树链剖分求 LCA

```

vector<int> edge[maxn];
int sz[maxn],fa[maxn],son[maxn],top[maxn],dep[maxn],id[maxn];
int tot=0;
void dfs1(int u,int depth){
    int v,i,mx=-1;

```

```

        sz[u]=1;dep[u]=depth;son[u]=0;
        for(int v:edge[u]){
            dfs1(v,depth+1);
            sz[u]+=sz[v];
            if (sz[v]>mx) mx=sz[v],son[u]=v;
        }
    }
void dfs2(int u,int x){
    int v,i;
    top[u]=x;id[u]=++tot;
    if (son[u]) dfs2(son[u],x);
    for (int v:edge[u]){
        if (v==son[u]) continue;
        dfs2(v,v);
    }
}
int query(int x,int y){
    while (top[x]!=top[y]){
        if (dep[top[x]]<dep[top[y]]) swap(x,y);
        x=fa[top[x]];
    }
    if (dep[x]>dep[y]) swap(x,y);
    return x;
}
int len(int x,int y){
    return dep[x]+dep[y]-dep[query(x,y)]*2+1;//point
}

```

离线 tarjin 求 LCA

```

vector<int> edge[maxn];
int fa1[maxn],fa2[maxn];
inline int getfa(int *fa,int x){
    if (fa[x]==x) return x;
    return fa[x]=getfa(fa,fa[x]);
}
int n,m,q;
int i,k;
int u,v;
int ans[maxn];
vector<pair<int,int> > Q[maxn];
void dfs(int x){
    int i;
    for (int v:edge[x]){
        dfs(v);

```

```

        fa2[v]=x;
    }
    REP(i,Q[x].size())
        if (fa2[Q[x][i].first]!=Q[x][i].first)
            ans[Q[x][i].second]=getfa(fa2,Q[x][i].first);
}
void solve(){
    REP(i,q){
        scanf("%d%d%d",&k,&u,&v);
        if (k==1){
            if (getfa(fa1,u)!=getfa(fa1,v)) ans[i]=-1;
            else{
                if (u==v) ans[i]=u;
                else{
                    Q[u].push_back(make_pair(v,i));
                    Q[v].push_back(make_pair(u,i));
                }
            }
        }else{
            edge[u].push_back(v);
            fa1[v]=u;
            ans[i]=0;
        }
    }
    FOR(i,1,n) if (fa1[i]==i) dfs(i);
    REP(i,q) if (ans[i]) printf("%d\n",ans[i]);
}

```

倍增

//记住 ladder 时,只在 top 处 resize!!

```

void dfs(int x,int depth){
    dep[x]=depth;
    for (int v:edge[x]) dfs(v,depth+1);
}
int lca(int x,int y){
    int i;
    if (dep[x]<dep[y]) swap(x,y);
    rREP(i,20) if (dep[x]-dep[y]>=1<<i) x=fa[x][i];
    if (x==y) return x;
    rREP(i,20) if (fa[x][i]!=fa[y][i]) x=fa[x][i],y=fa[y][i];
    return fa[x][0];
}
int dis(int x,int y){
    return dep[x]+dep[y]-2*dep[lca(x,y)];
}

```

```

}
INIT:
FOR(i,2,n) rep(j,1,20) fa[i][j]=fa[fa[i][j-1]][j-1];

```

最大团

```

int n;
int ans;
int edge[maxn][maxn],cnt[maxn],vis[maxn];//vis:元素
bool dfs(int u,int pos){
    int i,j;
    FOR(i,u+1,n){
        if (cnt[i]+pos<=ans) return 0;
        if (edge[u][i]){
            REP(j,pos) if (!edge[i][vis[j]]) break;
            if (j==pos){
                vis[pos]=i;
                if (dfs(i,pos+1)) return 1;
            }
        }
    }
    if (pos>ans){
        ans=pos;
        return 1;
    }
    return 0;
}
int maxclique(){
    int i;
    ans=-1;
    rFOR(i,1,n){
        vis[0]=i;
        dfs(i,1);
        cnt[i]=ans;
    }
    return ans;
}
int main(){
    int k;
    int i,j;
    scanf("%d%d",&n,&k);
    FOR(i,1,n)FOR(j,1,n) scanf("%d",&edge[i][j]);
    maxclique();
    printf("%.16lf",0.5*k*k*(ans-1)/ans);
}

```


最小树形图

//不定根:新加一个节点,向所有点加一条 INF 的边,最后减一下即可

//主要思路:缩点

//输出路径思路:缩完点记录边,然后新建边记录等价关系

```
struct node{
    int u,v,val,id;//id->usedID
}edge[maxn];
int pre[maxn],len[maxn],vis[maxn],id[maxn];
struct used{
    int pre,id;//original
}U[maxn*20]; //edges
int UID[maxn],used[maxn*20];
int OK[maxn];
int solve(int root,int n,int m){
    int ret=0,i,tot=m,em=m;
    REP(i,m) edge[i].id=U[i].id=0;
    while (1){
        FOR(i,1,n) len[i]=INF,vis[i]=0,id[i]=0;
        REP(i,m)
            if (edge[i].u!=edge[i].v&&edge[i].val<len[edge[i].v]){
                pre[edge[i].v]=edge[i].u;
                len[edge[i].v]=edge[i].val;
                UID[edge[i].v]=edge[i].id;
            }
        FOR(i,1,n) if (i!=root&&len[i]==INF) return -1;
        int cnt=0;len[root]=0;
        FOR(i,1,n){
            if (i!=root) used[UID[i]]++;
            ret+=len[i];int v;
            for(v=i;vis[v]!=i&&!id[v]&&v!=root;v=pre[v])
                vis[v]=i;

            if (v!=root&&!id[v]){
                cnt++;id[v]=cnt;
                for (int u=pre[v];u!=v;u=pre[u]) id[u]=cnt;
            }
        }if (!cnt) break;
        FOR(i,1,n) if (!id[i]) id[i]=++cnt;
        REP(i,m){
            int v=edge[i].v;
            edge[i].u=id[edge[i].u];edge[i].v=id[edge[i].v];
            if (edge[i].u==edge[i].v) edge[i--]=edge[--m];
            else {U[tot].id=edge[i].id;U[tot].pre=UID[v];
                edge[i].id=tot++;edge[i].val-=len[v];}
        }
    }
}
```

```
}n=cnt;root=id[root];
}
rrep(i,em,tot) if (used[i]){
    used[U[i].id]++;
    used[U[i].pre]--;
}
return ret;
}
int main(){
    freopen("input.txt","r",stdin);
    freopen("output.txt","w",stdout);
    int n,m,root;
    int i,j,k;
    scanf("%d%d",&n,&m);
    REP(i,m) scanf("%d%d%d",&edge[i].u,&edge[i].v,&edge[i].val);
    REP(i,m) OK[i]=edge[i].val;
    int ans=solve(1,n,m);
    printf("%d\n",ans);
    if (ans!=-1){
        REP(i,m) if (OK[i]&&used[i]) printf("%d ",i+1),ans--;
        if (ans) printf("\n%d\n",ans);
    }
}
```

一般图最大匹配 带花树

```
//缩奇环
int n,m;
vector<int> edge[maxn];
bool inQueue[maxn];
int belong[maxn];
int getbelong(int x) {
    if (belong[x]==x) return x;
    return belong[x]=getbelong(belong[x]);
}
int match[maxn],nxt[maxn],mark[maxn],vis[maxn];
int cnt;
queue<int> Q;
int used[maxn];
int lca(int u,int v) {
    cnt++;
    while (1) {
        u=getbelong(u);
        if (vis[u]==cnt) return u;
        vis[u]=cnt;
    }
}
```

板子 ???

```

        u=nxt[match[u]];
        if (v) swap(u,v);
    }
}
void merge(int u,int p) {
    while (u!=p) {
        int mu=match[u],v=nxt[mu];
        if (getbelong(v)!=p) nxt[v]=mu;
        if (mark[mu]==2) mark[mu]=1,Q.push(mu);
        if (mark[v]==2) mark[v]=1,Q.push(v);
        int x,y;
        x=getbelong(u),y=getbelong(mu);
        if (x!=y) belong[x]=y;
        x=getbelong(mu),y=getbelong(v);
        if (x!=y) belong[x]=y;
        u=v;
    }
}
void solve(int s) { //增广
    int i;
    FOR(i,1,n) belong[i]=i,mark[i]=nxt[i]=0;
    while (Q.size()) Q.pop();
    Q.push(s);
    while (Q.size()) {
        if (match[s]) return;
        int u=Q.front();
        Q.pop();
        for (int v:edge[u]) {
            if (match[u]==v) continue;
            if (getbelong(u)==getbelong(v)) continue;
            if (mark[v]==2) continue; //T 型点
            if (mark[v]==1) { //S 型点,缩点
                int p=lca(u,v);
                if (getbelong(u)!=p) nxt[u]=v;
                if (getbelong(v)!=p) nxt[v]=u;
                merge(u,p);
                merge(v,p);
            } else if (!match[v]) { //增广
                nxt[v]=u;
                for (int x=v; x;) {
                    int y=nxt[x],xx=match[y];
                    match[x]=y;
                    match[y]=x;
                    x=xx;
                }
            }
        }
    }
}

```

```

        break;
    } else {
        nxt[v]=u;
        mark[match[v]]=1;
        Q.push(match[v]);
        mark[v]=2;
    }
}
}
bool E[maxn][maxn];
int ans;
int main() {
    scanf("%d%d",&n,&m);
    int i;
    while (m--) {
        int u,v;
        scanf("%d%d",&u,&v);
        if (u!=v&&!E[u][v]) {
            edge[u].push_back(v);
            edge[v].push_back(u);
            E[u][v]=E[v][u]=1;
        }
    }
    memset(match,0,sizeof(match));
    FOR(i,1,n) if (!match[i]) solve(i);
    FOR(i,1,n) if (match[i]) ans++;
    ans/=2;
    printf("%d\n",ans);
    FOR(i,1,n) printf("%d ",match[i]);
}

```

数学相关

逆元

```
LL inv[1000002]; //inverse
LL fac[1000002]; //Factorial
// 求出的是  $ax+by=1$  的解(a,b 正负不限,而且挺小的);
// d(gcd)=1 时存在逆元;(d!=1)&&(num%d)时,num*a/d 可
认为逆元
// (x+p)%p 为逆元
// DP:C[i][j]=(C[i-1][j-1]+C[i][j-1])%M
void exgcd(LL a,LL b,LL &d,LL &x,LL &y){
    if (!b) {d=a;x=1;y=0;}
    else {exgcd(b,a%b,d,y,x);y-=a/b*x;}
}
// 前面那个线性求逆元的 log 版 2333
int getinv(int n){
    if (n==1) return 1;
    return (M-M/n)*(getinv(M%n))%M;
}
LL C(int n,int m){
    return fac[n]*inv[m]%M*inv[n-m]%M;
}
LL lucas(LL n,LL m){//注意 MOD 不能太大=_=! Mlogn
    return m==0?1:1ll*C(n%M,m%M)*lucas(n/M,m/M)%M;
}
int main(){
    int i;
    fac[0]=1;
    FOR(i,1,1000000) fac[i]=i*fac[i-1]%M;
    inv[0]=inv[1]=1;
    FOR(i,2,1000000) inv[i]=(M-M/i)*inv[M%i]%M;
    FOR(i,1,1000000) inv[i]=inv[i]*inv[i-1]%M;// inv(n!)
    printf("%l64d",C(10,3));
}
```

博弈：NIM,SG

选择的最多次数,main 中为异或!=0

```
int sg[maxm+2]; //打表~~~
/*这个是状态和剩余个数有关的
map<int,int> Hash;
int SG(int mask){
    if (Hash.count(mask)) return Hash[mask];
```

```
set<int> mex;
for (int i=0;i<maxm;++i){
    if (!(mask>>i)&1) continue;//continue
    int tp=mask;
    for (int j=i;j<maxm;j+=i+1)//change
        if ((mask>>j)&1) tp^=1<<j;
    mex.insert(SG(tp)); //dfs
}
int ret=0;
for (;mex.count(ret);++ret);
return Hash[mask]=ret;
}*/
```

/*这个是状态和剩余个数无关的

```
map<LL,int> Hash[62];
int SG(int x,LL mask){
    // printf("%d %d\n",x,mask);
    if (Hash[x].count(mask)) return Hash[x][mask];
    set<int> mex;
    for (int i=1;i<=x;++i){
        if ((mask>>(i-1))&1) continue;//continue
        int tp=mask;
        tp^=1<<(i-1);//change
        mex.insert(SG(x-i,tp)); //dfs
    }
    int ret=0;
    for (;mex.count(ret);++ret);
    return Hash[x][mask]=ret;
}*/
int main(){
    sg[0]=0;
}
```

FFT、NTT、FWT、FMT

FFT：

```
struct complex{
    double a,b;
    complex(double _a=.0,double _b=.0):a(_a),b(_b){}
    complex operator+(const complex x)const
    {return complex(a+x.a,b+x.b);}
    complex operator-(const complex x)const
    {return complex(a-x.a,b-x.b);}
    complex operator*(const complex x)const
```

板子 ???

```
{return complex(a*x.a-b*x.b,a*x.b+b*x.a);}
};
void fft(complex *A,int len,int inv){//抄的板子
    int i,j,k;
    for (i=1,j=len/2;i<len-1;i++){
        if (i<j) swap(A[i],A[j]);
        k=len/2;
        while(j>=k){
            j-=k;
            k/=2;
        }if (j<k) j+=k;
    }
    for(i=2;i<=len;i<=1){
        complex wn(cos(-inv*2*pi/i),sin(-inv*2*pi/i));
        for (j=0;j<len;j+=i){
            complex w(1.0,0.0);
            for (k=j;k<(j+i/2);k++){
                complex a=A[k],b=w*A[k+i/2];
                A[k]=a+b;
                A[k+i/2]=a-b;
                w=w*wn;
            }
        }
    }
    if (inv== -1) REP(i,len) A[i].a/=len;
}
complex x1[maxn],x2[maxn];
char a[maxn],b[maxn];
int ans[maxn];
int main(){
    int T;
    int i,j,k;
    scanf("%d",&T);
    while (T--){
        scanf("%s%s",a,b);
        bool mark=0;;
        int len1=strlen(a),len2=strlen(b),len=1;
        if (a[0]=='-')
        {REP(i,len1) a[i]=a[i+1];len1--;mark^=1;}
        if (b[0]=='-')
        {REP(i,len2) b[i]=b[i+1];len2--;mark^=1;}
        while(len<=len1+len2+1) len<=1;
        REP(i,len1) x1[i]=complex(a[len1-i-1]-'0',0);
        rep(i,len1,len) x1[i]=complex(0,0);
        REP(i,len2) x2[i]=complex(b[len2-i-1]-'0',0);
```

```
        rep(i,len2,len) x2[i]=complex(0,0);
        fft(x1,len,1);fft(x2,len,1);
        REP(i,len) x1[i]=x1[i]*x2[i];
        fft(x1,len,-1);
        REP(i,len) ans[i]=x1[i].a+0.5;
        REP(i,len) ans[i+1]+=ans[i]/10,ans[i]%10;
        while (ans[len-1]<=0&&len-1>0) len--;
        if (mark) putchar('-');
        rREP(i,len) putchar(ans[i]+'0');
        puts("");
    }
}
```

NTT :

```
const LL MOD=2281701377;
const LL g=3;
LL mul ( LL x , LL y ) {
    return x*y%MOD;
    // return ( x * y - ( long long ) ( x / ( long double ) MOD * y +
1e-3 ) * MOD + MOD ) % MOD ;
}
LL poww(LL a,LL b){
    LL ret=1;
    for (;b>=>=1ll,a=mul(a,a))
        if (b&1) ret=mul(ret,a);
    return ret;
}
void ntt(LL *A,int len,int inv){//抄的板子
    int i,j,k;
    for (i=1,j=len/2;i<len-1;i++){
        if (i<j) swap(A[i],A[j]);
        k=len/2;
        while(j>=k){
            j-=k;
            k/=2;
        }if (j<k) j+=k;
    }
    for(i=2;i<=len;i<=1){
        LL wn=poww(g,(MOD-1)/i);
        if (inv== -1) wn=poww(wn,MOD-2);
        for (j=0;j<len;j+=i){
            LL w=1;
            for (k=j;k<(j+i/2);k++){
                LL a=A[k],b=mul(w,A[k+i/2]);
```

```

        A[k]=(a+b)%MOD;
        A[k+i/2]=(a-b+MOD)%MOD;
        w=mul(w,wn);
    }
}
}
if (inv== -1){
    LL vn=poww(len,MOD-2);
    REP(i,len) A[i]=mul(A[i],vn);
}
}

FWT :

void fwt(LL *A,int len,int inv)//对拍对了
{
    int i,j,k;
    int div=powMM(2ll,M-2);
    for(i=2;i<=len;i<=1){
        for(j=0;j<len;j+=i){
            for(k=j;k<j+i/2;k++){
                if (inv==1){
                    LL a=A[k],b=A[k+i/2];
                    A[k]=(a+b)%M;
                    A[k+i/2]=(a-b+M)%M;
                    //xor:a[k]=x+y,a[k+i/2]=(x-y+mod)%mod;
                    //and:a[k]=x+y;
                    //or:a[k+i/2]=x+y;
                }else{
                    LL a=A[k],b=A[k+i/2];
                    A[k]=(a+b)*div%M;
                    A[k+i/2]=(a-b+M)%M*div%M;
                    //xor:a[k]=(x+y)/2,a[k+i/2]=(x-y)/2;
                    //and:a[k]=x-y;
                    //or:a[k+i/2]=y-x;
                }
            }
        }
    }
}
}

```

子集卷积(的分治套路)

<http://acm.hdu.edu.cn/showproblem.php?pid=6057>
 //很容易卡 $T \dots 3^{18}$ 也许能过
 //这个比 $2^{n \log^2 n}$ ($n=19$) 的慢了快 5 倍

板子 ???

```

//这种思路这种题都能用
int T;
int n;
ULL A[1<<19|7],B[1<<19|7];
ULL C[1<<22|7];
ULL ans,mul;
inline void solve(ULL *A,ULL *B,ULL *C,int len){
    int i;
    if (len==2)
        {C[1]=A[0]*B[1];C[0]=2*A[1]*B[1]+A[0]*B[0];return;}//这样要快
    // if (len==1) {C[0]=1ll*A[0]*B[0]%M;return;}
    ULL *D=C+len;
    len>>=1;
    solve(A,B,D,len);//这里 A 和 B 可能是要算的,这种情况下这就是正解
    solve(A,B+len,D+len,len);
    solve(A+len,B+len,D+len+len,len);
    REP(i,len){
        C[i+len]=D[i+len];
        (C[i]=D[i+len+len]*2+D[i])>INFF&&(C[i]=M);
    }
}
int main()
{
    int i;
    scanf("%d",&n);
    REP(i,(1<n)) read(A[i]);
    REP(i,(1<n)) read(B[i]);
    solve(A,B,C,1<n);
    mul=1;
    REP(i,(1<n)) {
        // printf("%d ",C[i]);
        C[i]=M;
        ans+=C[i]*mul;
        if(ans>INFF) ans%=M;
        mul=1526*mul%M;
    }ans%=M;
    printf("%llu\n",ans);
    return 0;
}

//真*子集卷积 by TLS
const int maxn = 1 << 19 | 1, mod = 998244353, seed =
1526;
int n, all, bit[maxn], a[maxn], b[maxn], ans;
inline void mod_inc(int &x, int y) {

```

```

(x += y) >= mod && (x -= mod);
}
int main() {
    while(scanf("%d", &n) == 1) {
        all = (1 << n) - 1;
        for(int i = 0; i <= all; ++i)
            scanf("%d", a + i);
        for(int i = 0; i <= all; ++i)
            scanf("%d", b + i);
        bit[0] = 1;
        for(int i = 1; i <= all; ++i) {
            bit[i] = bit[i >> 1] << (i & 1);
            a[i] = (LL)a[i] * bit[i] % mod;
        }
        ans = 0;
        for(int i = all; i >= 0; --i) {
            int msk = all ^ i, tim = 0;
            ULL cnt = 0;
            for(int j = msk; j; j = (j - 1) & msk) {
                cnt += (ULL)a[j] * b[i | j];
                (++tim) == 18 && (tim = 0, cnt %=
mod);
            }
            cnt += (ULL)a[0] * b[i];
            cnt %= mod;
            ans = ((LL)seed * ans + cnt) % mod;
        }
        printf("%d\n", ans);
    }
    return 0;
}

```

高斯消元

//求行列式的值

//%m,m 为质数的积

//从 0 开始

```

template<typename T>inline T poww(T a,T b,T M) {
    T ret=1;
    for (; b;b>>=1ll,a=1ll*a*a%M)
        if (b&1) ret=1ll*ret*a%M;
    return ret;
}
LL gauss(LL A[107][107],int n,LL M) {
    LL ret=1; int i,j,k;

```

```

REP(i,n) {
    int id=i;
    if (!A[i][i]) rep(j,i+1,n) if (A[j][i]) id=j;
    if (!A[id][i]) continue;
    if (id!=i) {rep(j,i,n) swap(A[i][j],A[id][j]); ret*=-1;}
    A[i][i]%=M; (A[i][i]<0) &&(A[i][i]+=M);
    LL rev=poww(A[i][i],M-2,M);
    rep(k,i+1,n)
        rrep(j,i,n)(A[k][j]-=(LL)A[k][i]*rev%M*A[i][j])%=M;
    } REP(i,n)(ret*=A[i][i])%=M;
    (ret<0) &&(ret+=M);
    return ret;
}
LL A[107][107],B[107][107];
void exgcd(LL a,LL b,LL &d,LL &x,LL &y) {
    if (!b) {d=a; x=1; y=0;}
    else {exgcd(b,a%b,d,y,x); y-=a/b*x;}
}
vector<LL> P;
vector<LL> Ans;
LL ans;
LL chinese_remainder(vector<LL> &m,vector<LL> &r) {
    int i; LL M=m[0],R=r[0];
    rep(i,1,P.size()) {
        LL x,y,d;
        exgcd(M,m[i],d,x,y);
        if ((r[i]-R)%d) return -1;
        x=(r[i]-R)/d*x%(m[i]/d);
        R+=x*M; M=M/d*m[i];
        R%=M; (R<0) &&(R+=M);
    } return R;
}
int n,m;
int i,j,k;
int main() {
    while (~scanf("%d%d",&n,&m)) {
        P.clear(); Ans.clear();
        REP(i,n)
            REP(j,n) scanf("%lld",&A[i][j]);
        for (i=2; i*i<=m; i++) if (m%i==0) {
            P.push_back(i);
            while (m%i==0) m/=i;
        } if (m!=1) P.push_back(m);
        for (int v:P) {
            REP(i,n) REP(j,n) B[i][j]=A[i][j];

```

板子 ???

```
        Ans.push_back((LL)guass(B,n,v));
    }
    ans=chinese_remainder(P,Ans);
    printf("%lld\n",ans);
}
}
//emmmm kuangbin 模板好像是错的
//这里是求正数的类似解,可能会不够精确
bool guass(long double A[107][107],long double X[107],int n,int m) {
    int i,j,k;
    REP(i,n) {
        int id=i;
        rep(j,i+1,m) if (abs(A[j][i])>abs(A[id][i])) id=j;
        if (abs(A[id][i])<eps) continue;
        if (id!=i)
            {rep(j,i,n) swap(A[j][i],A[id][i]); swap(X[i],X[id]);}
        REP(k,m) if (k!=i) {
            X[k]-=A[k][i]/A[i][i]*X[i];
            rrep(j,i,n) A[k][j]-=A[k][i]/A[i][i]*A[i][j];
        }
    }
    REP(i,n) if (abs(A[i][i])<eps&&abs(X[i])>eps) return 0;
    rep(i,n,m) if (abs(X[i])>eps) return 0;
    REP(i,n) if (abs(A[i][i])<eps||abs(X[i])<eps) X[i]=0;
    else X[i]/=A[i][i];
    return 1;
}
```

矩阵树定理|拉格朗日插值

// 题意:求生成树中含 k 条给定树边的生成树个数
// 做法:为给定边加不同权值,然后矩阵树定理
// 矩阵树定理:生成树数量=|基尔霍夫矩阵 C=D-A|;
// D 为度数矩阵,A 为边矩阵
// 然后拉格朗日插值求出系数即可
LL guass(LL A[107][107],int n,LL M) {

```
    LL ret=1; int i,j,k;
    REP(i,n) {
        int id=i;
        if (!A[i][i]) rep(j,i+1,n) if (A[j][i]) id=j;
        if (!A[id][i]) continue;
        if (id!=i) {rep(j,i,n) swap(A[j][i],A[id][i]); ret*=-1;}
        A[i][i]=M; (A[i][i]<0) &&(A[i][i]+=M);
        LL rev=poww(A[i][i],M-2,M);
        rep(k,i+1,n) rrep(j,i,n)
```

```
        (A[k][j]-=(LL)A[k][i]*rev%M*A[i][j])%=M;
    } REP(i,n)(ret*=A[i][i])%=M;
    (ret<0) &&(ret+=M);
    return ret;
}
int n,m;
int i,j,k;
int a[107][107]; LL A[107][107];
LL val[107],v_v[107];
LL f[107],g[107],ans[107];
int main() {
    scanf("%d",&n);
    FOR(i,1,n-1) {
        int u,v;
        scanf("%d%d",&u,&v); u--; v--;
        a[u][v]=a[v][u]=1;
    } REP(i,n) v_v[i]=i;
    REP(k,n) {
        REP(i,n) REP(j,n) A[i][j]=0;
        REP(i,n) REP(j,n) if (i!=j) {
            if (a[i][j]) A[i][j]=M-v_v[k],A[i][i]+=v_v[k];
            else A[i][j]=M-1,A[i][i]++;
        } val[k]=guass(A,n-1,M);
    }
    g[0]=1; REP(i,n) rFOR(j,0,i)(g[j+1]+=g[j])%=M,(g[j]*=(M-v_v[i]))%=M;
    REP(k,n) {
        LL rev=1;
        rFOR(i,0,n) f[i]=(g[i+1]+f[i+1]*v_v[k]%M+M)%M;
        REP(j,n) if (j!=k)(rev*=(v_v[k]-v_v[j]))%=M;
        (rev<0) &&(rev+=M); rev=powMM(rev,M-2);
        rev=(rev*val[k])%M;
        FOR(i,0,n)(ans[i]+=((LL)f[i]*rev%M)%M)=M;
    } FOR(i,0,n-1) printf("%lld ",ans[i]);
}
```

Polya 定理| Burnside 引理

//HDU3923; 颜色 m, 个数 n, 翻转或者置换当成一种
//ans=1/|G|*sigma{pow(k(color),m(not move point 不动点数))}
//注意特殊形式
//Burnside 引理:等价类个数 l=sum{ci(ai)},ci 是置换下的不动点数
//这个 pow 是可以变化成其他形式的
//注意,polya 定理相当于手动算了一下 Burnside 引理中不动点的
个数!

```
int n,m;
```

```

bool mark[maxn];
int phi[maxn];
int p[maxn], tot;
int main() {
    int i, j;
    phi[1] = 1;
    FOR(i, 2, 1000000) {
        if (!mark[i]) p[tot++] = i, phi[i] = i - 1;
        REP(j, tot) {
            if (i * p[j] > 1000000) break;
            //感觉上不会爆, 因为是从小往筛的
            mark[i * p[j]] = 1;
            if (i % p[j] == 0) { phi[i * p[j]] = phi[i] * p[j]; break; }
            else phi[i * p[j]] = phi[i] * (p[j] - 1);
        }
    }
    int t, T;
    scanf("%d", &T);
    FOR(t, 1, T) {
        scanf("%d%d", &m, &n);
        LL all = 0, cnt = 0;
        // FOR(i, 1, n) {
        //     (all += powMM((LL)m, gcd(n, i))) %= M;
        //     (all < 0) && (all += M);
        // } cnt = n;
        //置换
        FOR(i, 1, n) if (n % i == 0) {
            (all += (LL)powMM(m, i) * phi[n / i]) %= M;
            (all < 0) && (all += M);
        }
        cnt = n;
        //翻转
        if (n & 1) {
            (all += (LL)n * powMM(m, (n + 1) / 2)) %= M;
            cnt += n;
        } else {
            (all += (LL)n / 2 * powMM(m, n / 2)) %= M;
            (all += (LL)n / 2 * powMM(m, n / 2 + 1)) %= M;
            cnt += n;
        }
        // printf("%lld %lld\n", cnt, all);
        all = all * powMM(cnt, M - 2) % M;
        printf("Case # %d: %lld\n", t, all);
    }
}

```

Miller_Rabin 素性测试 + pollard_rho 因数分解

poj1181

```

/*miller_rabin*/
const int times = 8; // random_check; 8-12 is OK
LL mul(LL a, LL b, LL M) {
    LL ret = 0;
    for (; b >= 1; (a += a) >= M && (a -= M))
        if (b & 1) (ret += a) >= M && (ret -= M);
    return ret;
}
LL poww(LL a, LL b, LL M) {
    LL ret = 1;
    for (; b >= 1; a = mul(a, a, M))
        if (b & 1) ret = mul(ret, a, M);
    return ret;
}
bool check(LL a, LL n, LL x, LL t) {
    LL ret = poww(a, x, n);
    LL last = ret;
    for (ret = mul(ret, ret, n); t--; last = ret, ret = mul(ret, ret, n))
        if (ret == 1 && last != 1 && last != n - 1) return true;
    if (ret != 1) return true;
    return false;
}
bool miller_rabin(LL n) {
    if (n < 2) return false;
    if (!(n & 1)) return (n == 2);
    LL x = n - 1, t = 0;
    while (!(x & 1)) x >>= 1, t++;
    int i;
    REP(i, times)
        if (check(rand() % (n - 1) + 1, n, x, t)) return false;
    return true;
}
/*pollard_rho*/
LL pollard_rho(LL x, LL c) {
    LL x0 = rand() % (x - 1) + 1;
    LL y = x0; c %= x;
    for (LL i = 2, k = 2; i++) {
        ((x0 = mul(x0, x0, x) + c) >= x) && (x0 -= x);
        LL d = gcd(y - x0 + x, x);
        if (d != 1 && d != x) return d;
    }
}

```



```

        if (y==x0) return x;
        if (i==k) y=x0,k+=k;
    }
}
LL factor[107]; int tot;
void findfac(LL n,int k) {
    if (n==1) return;
    if (miller_rabin(n)) {factor[tot++]=n; return;}
    LL p=n;
    int c=k;
    while (p>=n) p=pollard_rho(p,c--);
    findfac(p,k);
    findfac(n/p,k);
}
int main() {
    int T;
    srand(time(0));
    scanf("%d",&T);
    while (T--) {
        LL n; int i;
        scanf("%l64d",&n);
        if (miller_rabin(n)) puts("Prime");
        else {
            tot=0;
            findfac(n,107);
            LL ans=factor[0];
            REP(i,tot) ans=min(ans,factor[i]);
            printf("%l64d\n",ans);
        }
    }
}

```

中国剩余定理(不一定互质)

```

void exgcd(LL a,LL b,LL &d,LL &x,LL &y){
    if (!b) {d=a;x=1;y=0;}
    else {exgcd(b,a%b,d,y,x);y-=a/b*x;}
}
int n,m;
int i,j,k;
vector<LL> P,O;
int ans;
LL chinese_remainder(vector<LL> &m,vector<LL> &r){
    int i;LL M=m[0],R=r[0];
    rep(i,1,P.size()){

```

```

        LL x,y,d;
        exgcd(M,m[i],d,x,y);
        if ((r[i]-R)%d) return -1;
        x=(r[i]-R)/d*x%(m[i]/d);
        R+=x*M;M=M/d*m[i];
        R%=M;(R<0)&&(R+=M);
    }return R;
}
int main(){
    while (~scanf("%d",&n)){
        P.clear();O.clear();
        REP(i,n){
            LL k;
            scanf("%lld",&k);P.push_back(k);
            scanf("%lld",&k);O.push_back(k);
        }printf("%lld\n",chinese_remainder(P,O));
    }
}

```

Prime-counting function

//这道题题意:小于 n 有多少个数字有 4 个因子

//(两个质数积,一个质数三次方)

//注意容斥减去多算的

//http://codeforces.com/blog/entry/44466?#comment-290036/

//考虑 $S(v,m):2 \dots v$,质因子全都 $\geq m$;那么考虑容斥:

//容斥掉的至少有一个 p,而且没有小于 p 的因子

//很明显的, $p=\min(p,\sqrt{v})$;

// $S(v,p)=S(v,p-1)-(S(v/p,p-1)-S(p-1,p-1))$; (DP)

//那么反过来算即可; $\pi(n)=S(n,n)$;

// $H[i]:\pi(n/i);L[i]:\pi(i)$

//计算过程中, $L[i]$ 表示 $S(i,p)$,最终 $S(i,i)$

//简单的这样 DP,时间复杂度 $O(n^{3/4})$,如果预处理 $n^{2/3}$ 则最终 $n^{2/3}$

//在后方,如果要容斥,FOR 是很不方便的,感觉还是最好直接搞复杂度有保障

```
LL H[maxn],L[maxn];
```

```
void calc(LL n) {
```

```
    LL p,k,m;
```

```
    for (m=1; m*m<=n; ++m) H[m]=n/m-1;
```

```
    FOR(p,1,m) L[p]=p-1;
```

```
    FOR(p,2,m) { //在这里,如果前方限制了 P 的最大值,是
```

```
min(P,m)
```

```
    if (L[p]==L[p-1]) continue; //not_prime
```

```

FOR(k,1,min(m-1,n/p/p)) {
    if (p*k<m) H[k]-=H[p*k]-L[p-1];
    else H[k]-=L[n/p/k]-L[p-1];
}
rFOR(k,p*p,m) L[k]-=L[k/p]-L[p-1];
}
}
LL n,ans,i;
int main() {
    scanf("%l64d",&n);
    calc(n);
    LL m=sqrt(n-1);
    while (m*m<=n) m++;
    m--;
    FOR(i,2,m) if (L[i]!=L[i-1]) ans+=H[i]-L[i];
    m=cbrt(n-1);
    while (m*m*m<=n) m++;
    m--;
    ans+=L[m];
    printf("%l64d\n",ans);
}

```

$N^{2/3}$ 的方法:

//pcf::get_cnt ; pcf::Lehmer(x)

```

namespace pcf {
#define clr(ar) memset(ar, 0, sizeof(ar))
#define chkbit(ar, i) (((ar[(i) >> 6]) & (1 << (((i) >> 1) & 31))))
#define setbit(ar, i) (((ar[(i) >> 6]) |= (1 << (((i) >> 1) & 31))))
#define isprime(x) (( (x) && ((x)&1) && (!chkbit(ar, (x)))) || ((x) ==
2))

const int MAXN=100;
const int MAXM=100010;
const int MAXP=666666;
const int MAX=10000010;
long long dp[MAXN][MAXM];
unsigned int ar[(MAX >> 6) + 5] = {0};
int len = 0, primes[MAXP], counter[MAX];
void Sieve() { //nloglogn...这里压 64 位减了点空间
    setbit(ar, 0), setbit(ar, 1);
    for (int i = 3; (i * i) < MAX; i++, i++) {
        if (!chkbit(ar, i)) {
            int k = i << 1;
            for (int j = (i * i); j < MAX; j += k) setbit(ar, j);
        }
    }
}

```

```

for (int i = 1; i < MAX; i++) {
    counter[i] = counter[i - 1];
    if (isprime(i)) primes[len++] = i, counter[i]++;
}
}
void init() {
    Sieve();
    for (int n = 0; n < MAXN; n++) {
        for (int m = 0; m < MAXM; m++) {
            if (!n) dp[n][m] = m;
            else dp[n][m] = dp[n - 1][m] - dp[n - 1][m /
primes[n - 1]];
        }
    }
}
long long phi(long long m, int n) {
    if (n == 0) return m;
    if (primes[n - 1] >= m) return 1;
    if (m < MAXM && n < MAXN) return dp[n][m];
    return phi(m, n - 1) - phi(m / primes[n - 1], n - 1);
}
long long Lehmer(long long m) { //这里只是加速
    if (m < MAX) return counter[m];
    long long w, res = 0;
    int i, a, s, c, x, y;
    s = sqrt(0.9 + m), y = c = cbrt(0.9 + m);
    a = counter[y], res = phi(m, a) + a - 1;
    for (i = a; primes[i] <= s; i++) res = res - Lehmer(m /
primes[i]) + Lehmer(primes[i]) - 1;
    return res;
}
}

```

欧拉降幂公式

// $n^x \pmod m = m^{(\phi(m) + x\phi(m))\%m}$ ($x > m$)

//这个题让求 $\text{pow}(l, \text{pow}(l+1, \dots, \text{pow}(r)))$

```

inline int mod(LL a, int b) {
    if (a < b) return a;
    return a % b + b;
}
inline int poww(int a, int b, int M) {
    int ret = 1;
    for (; b >= 1; a = mod(1ll * a * a, M))
        if (b & 1) ret = mod(1ll * ret * a, M);
}

```

```

    return ret;
}
typedef pair<int,int> pii;
int P[maxn];
int phi(int x){
    int k=x;
    for (int i=2;i*i<=k;i++) if (k%i==0){
        x=x/i*(i-1);
        while (k%i==0) k/=i;
    }if (k!=1) x=x/k*(k-1);
    return x;
}
int a[maxn];
int tot;
int solve(int l,int r,int pos){
    if (l==r||pos==tot) return mod(a[l],P[pos]);
    return poww(a[l],solve(l+1,r,pos+1),P[pos]);
}
int n,m,q;
int i,j,k;
int main(){
    scanf("%d%d",&n,&m);
    FOR(i,1,n) scanf("%d",&a[i]);
    P[1]=m;
    for (tot=1;P[tot]!=1;tot++) P[tot+1]=phi(P[tot]);
    // FOR(i,1,tot) printf("%d ",P[i]);puts("");
    scanf("%d",&q);
    FOR(i,1,q){
        int l,r;int ans=1;
        scanf("%d%d",&l,&r);
        printf("%d\n",solve(l,r,1)%m);
    }
}

```

其他的東西

BSGS : $a^x = b \pmod p$

做法 : 假设 $m=\sqrt{p}+1$; $x=i*m-j(0<i<j)$

枚举 i 和 j , 我们得到了一个 \sqrt{p} 的做法

自适应simpson 积分

```

double simpson(double a,double b) {
    double c = a + (b-a)/2;
    return (F(a) + 4*F(c) + F(b))*(b-a)/6;
}

```

```

}
double asr(double a,double b,double eps,double A) {
    double c = a + (b-a)/2;
    double L = simpson(a,c), R = simpson(c,b);
    if (fabs(L + R - A) <= 15*eps)
        return L + R + (L + R - A)/15.0;
    return asr(a,c,eps/2,L) + asr(c,b,eps/2,R);
}
double asr(double a,double b,double eps) {
    return asr(a,b,eps,simpson(a,b));
}

```

(我全都不会的)积性函数

1、欧拉函数

```

bool mark[maxn];
int phi[maxn];
int p[maxn],tot;
const int MAX=1e6;
void init(int x){
    tot=0;int i,j;phi[1]=1;
    FOR(i,2,MAX){
        if (!mark[i]) p[tot++]=i,phi[i]=i-1;
        REP(j,tot){
            //感觉上不会爆,因为是从小往筛的
            if (i*p[j]>MAX) break;
            mark[i*p[j]]=1;
            if (i%p[j]==0) {phi[i*p[j]]=phi[i]*p[j];break;}
            else phi[i*p[j]]=phi[i]*(p[j]-1);
        }
    }
}

```

2、莫比乌斯函数

```

bool mark[maxn];
int mu[maxn];
int p[maxn],tot;
const int MAX=1e6;
void init(int x){
    tot=0;int i,j;
    FOR(i,2,MAX){
        if (!mark[i]) p[tot++]=i,mu[i]=1;
        REP(j,tot){
            //感觉上不会爆,因为是从小往筛的
            if (i*p[j]>MAX) break;
            mark[i*p[j]]=1;

```

板子???

```

        if (i%p[j]==0) break;
        else mu[i*p[j]]=-mu[i];
    }
}

```

3、用法和性质(不会 qwq)

首先是简单性质：

$n = \sum_{d|n} \phi(d)$ 将 ϕ 看作容斥系数
 $\phi(n) = \sum_{d|n} \mu(d) \cdot \frac{n}{d}$ 将 i/n 化为最简分数
 $1 \cdots n$ 的与 n 互质数和 $n \cdot \phi(n)/2$
 然后，经过推导可能将某些式子化成简单形式就能做了 qwq 完全不会，智商不够没办法……
 懒得写了放个图

2. 关于莫比乌斯函数和欧拉函数有两个经典的公式

1. $\sum_{d|n} \mu(d) = [n=1]$ ，将 $\mu(d)$ 看作是容斥的系数即可证明。

2. $n = \sum_{d|n} \phi(d)$ ，将 $\frac{i}{n} (1 \leq i \leq n)$ 化为最简分数统计个数即可证明。

3. 若 $f(n)$ 为积性函数，则对于正整数 $n = \prod_{i=1}^t p_i^{k_i}$ 有 $f(n) = \prod_{i=1}^t f(p_i^{k_i})$ ；若 $f(n)$ 为完全积性函数，则对于正整数 $n = \prod_{i=1}^t p_i^{k_i}$ 有 $f(n) = \prod_{i=1}^t f(p_i)^{k_i}$ 。

狄利克雷卷积与莫比乌斯反演

1. 数论函数 f 和 g 狄利克雷卷积定义为 $(f * g)(n) = \sum_{d|n} f(d) \cdot g(\frac{n}{d})$ ，狄利克雷卷积满足交换律、结合律，对加法满足分配律，存在单位元函数 $e(n) = [n=1]$ 使得 $f * e = f = e * f$ ，若 f 和 g 为积性函数则 $f * g$ 也为积性函数。

2. 狄利克雷卷积的一个常用技巧是对于积性函数 f 与恒等函数 I 的卷积的处理，例如 $n = \prod_{i=1}^t p_i^{k_i}$ ， $g(n) = \sum_{d|n} f(d)$ ，则有 $g(n) = \prod_{i=1}^t \sum_{j=0}^{k_i} f(p_i^j)$ 。

3. 莫比乌斯反演也是对于 $g(n) = \sum_{d|n} f(d)$ 的讨论，但是不要求 f 是积性函数，适用于已知 $g(n)$ 求 $f(n)$ 的情况，由于 $I * \mu = e$ ，则 $g * \mu = f * I * \mu = f * e = f$ ，即 $f(n) = \sum_{d|n} g(d) \cdot \mu(\frac{n}{d})$ ，类似地有 $g(n) = \sum_{n|d} f(d) \Rightarrow f(n) = \sum_{n|d} g(d) \cdot \mu(\frac{d}{n})$ ，二项式反演也是类似的技巧。有一个例子可以看出欧拉函数和莫比乌斯函数之间的关系，由于 $\sum_{d|n} \phi(d) = id(n)$ ，所以 $\phi(n) = \sum_{d|n} \mu(d) \cdot \frac{n}{d}$ ，也即 $\frac{\phi(n)}{n} = \sum_{d|n} \frac{\mu(d)}{d}$ 。

黑科技大概就是说把 ϕ ， μ 或者其他的东西提到前面然后换元来做，根本不会 qwq

首先看一个简单的例子，求前 n 个正整数的约数之和，即 $\sum_{i=1}^n \sigma(i)$ ，其中 $n \leq 10^{12}$ 。

显然不能直接做了，但是我们可以推导一番：

$$\sum_{i=1}^n \sigma(i) = \sum_{i=1}^n \sum_{j=1}^n [j|i] \cdot j = \sum_{i=1}^n i \cdot \sum_{j=1}^n [i|j] = \sum_{i=1}^n i \cdot \left\lfloor \frac{n}{i} \right\rfloor$$

当 $i \leq \sqrt{n}$ 时， $\left\lfloor \frac{n}{i} \right\rfloor$ 显然只有 $O(\sqrt{n})$ 个取值；当 $i > \sqrt{n}$ 时， $\left\lfloor \frac{n}{i} \right\rfloor < \sqrt{n}$ 显然也只有 $O(\sqrt{n})$ 个取值；对于固定的 $\left\lfloor \frac{n}{i} \right\rfloor$ ， i 的取值是一段连续的区间，这段区间是 $\left[\left\lfloor \frac{n}{\left\lfloor \frac{n}{i} \right\rfloor + 1} \right\rfloor + 1, \left\lfloor \frac{n}{\left\lfloor \frac{n}{i} \right\rfloor} \right\rfloor \right]$ ，因此可以 $O(\sqrt{n})$ 计算所求。

同样地，求前 n 个正整数的约数个数之和也可以这样计算，留给读者练习。

另外需要说明的是， $\sum_{i=1}^n \left\lfloor \frac{n}{i} \right\rfloor \cdot i = \sum_{i=1}^n \frac{\left\lfloor \frac{n}{i} \right\rfloor \cdot (\left\lfloor \frac{n}{i} \right\rfloor + 1)}{2}$ ，这也是一种常见的表示形式。

超麻烦的类欧几里得

$$\begin{aligned}
 f(a, b, c, n) &= \sum_{i=0}^n \lfloor \frac{ai+b}{c} \rfloor \\
 g(a, b, c, n) &= \sum_{i=0}^n i \lfloor \frac{ai+b}{c} \rfloor \\
 h(a, b, c, n) &= \sum_{i=0}^n \lfloor \frac{ai+b}{c} \rfloor^2 \\
 m &= \lfloor \frac{an+b}{c} \rfloor
 \end{aligned}$$

$$\begin{aligned}
 f(a, b, c, n) &= \sum_{i=0}^n \sum_{j=0}^{m-1} [(ai+b)/c \geq j+1] \\
 f(a, b, c, n) &= \sum_{i=0}^n \sum_{j=0}^{m-1} [ai \geq cj + c - b] \\
 f(a, b, c, n) &= \sum_{i=0}^n \sum_{j=0}^{m-1} [ai > cj + c - b - 1] \\
 f(a, b, c, n) &= \sum_{i=0}^n \sum_{j=0}^{m-1} [i > (cj + c - b - 1)/a] \\
 f(a, b, c, n) &= \sum_{j=0}^m (n - (cj + c - b - 1)/a) \\
 f(a, b, c, n) &= nm - f(c, c - b - 1, a, m - 1)
 \end{aligned}$$

$a \geq c$ 或 $b \geq c$ 的时候，容易得到

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

注意这里0到n的二次方和公式就是 $n * (n+1) * (2n+1)/6$

接下来我们来推a和b均小于c的情况。

首先先和f一样把下取整那玩意变换一下

$$\begin{aligned}
 g(a, b, c, n) &= \sum_{i=0}^n i \sum_{j=1}^m [(ai+b)/c \geq j] \\
 g(a, b, c, n) &= \sum_{i=0}^n i \sum_{j=0}^{m-1} [i > (cj + c - b - 1)/a] \\
 g(a, b, c, n) &= 1/2 * \sum_{j=0}^{m-1} (n+1 + (cj + c - b - 1)/a) * (n - (cj + c - b - 1)/a)
 \end{aligned}$$

这里看的有点难受，f的时候转化的其实是0次方和，而g是1次方和，所以这里用了个求和公式。

$$g(a, b, c, n) = 1/2 * \sum_{j=0}^{m-1} n(n+1) - (cj + c - b - 1)/a - [(cj + c - b - 1)/a]^2$$

拆出来了

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

$a \geq c$ 或 $b \geq c$ 的时候，h也是很麻烦的.....

三项式的平方，拆出来大概是这样

$$\begin{aligned}
 h(a, b, c, n) &= (a/c)^2 * n(n+1)(2n+1)/6 + (b/c)^2 * (n+1) + (a/c) * (b/c) * n(n+1) \\
 &+ h(a\%c, b\%c, c, n) + 2 * (a/c) * g(a\%c, b\%c, c, n) + 2 * (b/c) * f(a\%c, b\%c, c, n)
 \end{aligned}$$

接下来a和b均小于c的情况，我们要转化一下思路了。

如何获得一个 n^2 ?

$$n^2 = 2 * \frac{n(n+1)}{2} - n = 2 \sum_{i=0}^n i - n$$

有了思路我们来推h

$$h(a, b, c, n) = \sum_{i=0}^n (2 * \sum_{j=1}^{(ai+b)/c} j - (ai+b)/c)$$

可以想到交换主体。

$$\begin{aligned}
 h(a, b, c, n) &= 2 * \sum_{j=0}^{m-1} (j+1) * \sum_{i=0}^n [(ai+b)/c \geq j+1] - f(a, b, c, n) \\
 h(a, b, c, n) &= 2 * \sum_{j=0}^{m-1} (j+1) * \sum_{i=0}^n [i > (cj + c - b - 1)/a] - f(a, b, c, n) \\
 h(a, b, c, n) &= 2 * \sum_{j=0}^{m-1} (j+1) * (n - (cj + c - b - 1)/a) - f(a, b, c, n) \\
 h(a, b, c, n) &= nm(m+1) - 2g(c, c - b - 1, a, m - 1) - 2f(c, c - b - 1, a, m - 1) \\
 &- f(a, b, c, n)
 \end{aligned}$$

不会的计算几何

切比雪夫距离转曼哈顿距离：

切比雪夫距离： $\max(|x1-x2|,|y1-y2|)$;

曼哈顿距离： $|x1-x2|+|y1-y2|$

转化方式：旋转 45 度然后/2

$(x,y) \rightarrow ((x+y)/2, (x-y)/2)$

曼哈顿距离最小生成树：

按照 45 度 4 个方向排序，最近的两个点连边即可

swap 方向代码：

```
int a[MAXN], b[MAXN];
tot = 0;
for (int dir = 0; dir < 4; dir++) {
//4 种坐标变换
    if (dir == 1 || dir == 3) {
        for (int i = 0; i < n; i++) swap(p[i].x, p[i].y);
    } else if (dir == 2) {
        for (int i = 0; i < n; i++) p[i].x = -p[i].x;
    }
    sort(p, p+n, cmp);
    for (int i = 0; i < n; i++)
        a[i] = b[i] = p[i].y - p[i].x;
    sort(b, b+n);
    int m = unique(b, b+n) - b;
    for (int i = 1; i <= m; i++) bit[i].init();
    for (int i = n-1; i >= 0; i--) {
        int pos = lower_bound(b, b+m, a[i]) - b + 1;
        int ans = ask(pos, m);
        if (ans != -1)
            addedge(p[i].id, p[ans].id, dist(p[i], p[ans]));
        update(pos, p[i].x+p[i].y, i);
    }
}
```

求凸包(旋转卡壳,最近最远点对)

```
inline int sgn(double x) {
    if (abs(x)<eps) return 0;
    if (x<0) return -1;
    return 1;
}
struct point {
```

```
LL x, y;
bool operator <(const point &a) const {
    if (y<a.y) return 1;
    if (y>a.y) return 0;
    return x<a.x;
}
} p[maxn], P[maxn], p1[maxn];
inline LL X(point A, point B, point C) {
    return (B.x-A.x)*(C.y-A.y)-(B.y-A.y)*(C.x-A.x);
}
inline LL len(point A, point B) {
    return (A.x-B.x)*(A.x-B.x)+(A.y-B.y)*(A.y-B.y);
}
bool cmp(point A, point B) {
    LL cp=X(p[0], A, B);
    if (cp>0) return 1;
    if (cp<0) return 0;
    return len(p[0], A)<len(p[0], B);
// return sgn(len(p[0], A)-len(p[0], B))<=0;
}
int n;
LL getMAX() { //求完凸包旋转卡壳
    int tot, i, j, m;
    LL ans=0;
    if (n==1) {
        tot=0;
        P[0]=p[0];
    } else if (n==2) {
        tot=1;
        P[0]=p[0];
        P[1]=p[1];
    } else {
        REP(i, n) if (p[i]<p[0]) swap(p[0], p[i]);
        sort(p+1, p+n, cmp);
        P[0]=p[0];
        P[1]=p[1];
        tot=1;
        rep(i, 2, n) {
            while (tot&&X(P[tot-1], P[tot], p[i])<=0) tot--;
            P[++tot]=p[i];
        }
    }
}
```


板子 ???

```
} m=tot;
FOR(i,0,tot) P[++m]=P[i];
j=0; ans=0;
FOR(i,0,m) {
    while (j<m&&len(P[i],P[j])<len(P[i],P[j+1])) j++;
    ans=max(ans,len(P[i],P[j]));
} return ans;
}

inline int cmpx(point a,point b) {return a.x<b.x;}
inline int cmpy(point a,point b) {return a.y<b.y;}
LL getMIN(int l,int r) { //分治求最近点对,nsqrtn
    LL ans=0;
    int i,j;
    if (l==r) return INFF;
    if (l+1==r) return len(p[l],p[r]);
    int mid=(l+r)>>1;
    ans=min(getMIN(l,mid),getMIN(mid+1,r));
    int cn=0;
    FOR(i,l,r) if (p[i].x-p[mid].x<ans) p1[cn++]=p[i];
    sort(p1,p1+cn,cmpy);
    REP(i,cn) {
        rep(j,i+1,cn) {
            if (p1[j].y-p1[i].y>=ans) break;
            ans=min(ans,len(p1[i],p1[j]));
        }
    }
    return ans;
}

int i,j,k;
LL ans;
int main() { //0->tot 是凸包上的点
    while (~scanf("%d",&n)) {
        REP(i,n) scanf("%lld%lld",&p[i].x,&p[i].y);
        sort(p,p+n,cmpx);
        printf("%lld %lld\n",getMIN(0,n-1),getMAX());
    }
}
```