A. Distance and Axis

n 不足 k 先补齐，否则看 $n+k$ 奇偶性。

```
#include<bits/stdc++.h>
#define ll long long
#define fi first
#define se second
#define mp make_pair
#define pb push_back
#define pii pair<int,int>
using namespace std;
const int N=5e4+10;
int a[N];
int main()
{
    int t;
    scanf("%d",&t);
    while(t--)
    {
        int n,k;
        scanf("%d%d",&n,&k);
        if(k>n)printf("%d\n",k-n);
        else if((n+k)&1)puts("1");
        else puts("0");

    }
    return 0;
}
```

B. Ternary Sequence

显然除了 $a1b2,a2b1$ 的组合外其他 c_i 为零，于是使得前者尽可能少，后者尽可能多。

```
#include<bits/stdc++.h>
#define ll long long
#define fi first
#define se second
#define mp make_pair
#define pb push_back
#define pii pair<int,int>
using namespace std;
const int N=5e4+10;
```

```
int a[N];
int main()
{
    int t;
    scanf("%d",&t);
    while(t--)
    {
        int a0,a1,a2,b0,b1,b2,res=0;
        scanf("%d%d%d%d%d%d",&a0,&a1,&a2,&b0,&b1,&b2);
        int t1=min(a2,b1);
        res+=t1*2,b1-=t1;
        res+=-2*max(0,a1-b0-b1);
        printf("%d\n",res);
    }
    return 0;
}
```

C. Mere Array

排序判断不在其位的数字是否能被最小值整除即可。

```
#include<bits/stdc++.h>
#define ll long long
#define fi first
#define se second
#define mp make_pair
#define pb push_back
#define pii pair<int,int>
using namespace std;
const int N=1e5+10;
int a[N],b[N];
int main()
{
    int t;
    scanf("%d",&t);
    while(t--)
    {
        int n,f=1;
        scanf("%d",&n);
        for(int i=1;i<=n;i++)scanf("%d",&a[i]),b[i]=a[i];
        sort(b+1,b+1+n);
        for(int i=1;i<=n;i++)
        if(a[i]!=b[i]&&a[i]%b[1]!=0){f=0;break;}
        if(f)puts("YES");else puts("NO");
    }
```

```
    return 0;
}
```

D. Maximum Distributed Tree

给出一棵树，要给每条边赋边权，使得所有边权的乘积为指定值（且 1 要最少），最大化 $\sum\limits_{i=1}^{n-1} \sum\limits_{j=i+1}^n f(i,j)$ 其中 $f(i,j)$ 为 i,j 简单路径长度。

每条边对答案的贡献次数是固定的（一侧的点数乘另一侧的点数），排序之后贪心地将更大的权值赋给使用次数更多的边（注意质因数多于边数时我们把前若干大的质因数乘起来做最大边权）。

```
#include<bits/stdc++.h>
#define ll long long
#define fi first
#define se second
#define mp make_pair
#define pb push_back
#define pii pair<int,int>
using namespace std;
const int N=1e5+10;
const ll mod=1e9+7;
int n,m,head[N],cnt,tot;
ll p[N],sz[N],tim[N];
struct Node{int nxt,to;}edges[N*2];
void addedge(int u,int v){edges[++cnt]=Node{head[u],v},head[u]=cnt;}
void dfs(int u,int f)
{
    sz[u]=1;
    for(int i=head[u];~i;i=edges[i].nxt)
    {
        int v=edges[i].to;
        if(v!=f)dfs(v,u),sz[u]+=sz[v],tim[++tot]=sz[v]*(n-sz[v]);
    }
}
int main()
{
    int t;
    scanf("%d",&t);
    while(t--)
    {
        memset(head,-1,sizeof(head)),cnt=0,tot=0;
        scanf("%d",&n);
        for(int i=1;i<n;i++)
        {
            int u,v;
            scanf("%d%d",&u,&v);
```

```
            addedge(u,v),addedge(v,u);
        }
        scanf("%d",&m);
        for(int i=1;i<=m;i++)scanf("%lld",&p[i]);
        sort(p+1,p+1+m,[&](ll x,ll y){return x>y;});
        if(m>n-1)
        {
            for(int i=2;i<=m-n+2;i++)p[1]=p[1]*p[i]%mod;
            for(int i=2;i<n;i++)p[i]=p[i+m-n+1];
        }
        else for(int i=m+1;i<=n-1;i++)p[i]=1;
        dfs(1,0);assert(tot==n-1);
        sort(tim+1,tim+1+tot,[&](ll x,ll y){return x>y;});
        ll res=0;
        for(int i=1;i<n;i++)res=(res+tim[i]%mod*p[i]%mod)%mod;
        printf("%lld\n",res);
    }
    return 0;
}
```

E. Divide Square

二维平面上有一个大正方形，给出若干线段（线段必有一端在正方形边框上，且同一行不会有两条线段），问正方形被分成多少块。

每有一条横跨正方形的线段块数加一，每有一个交点块数加一。交点这个可以线段树搞一搞，又有点类似扫描线，呃一维排序加入一维查询什么的。（不是一个很优美的写法）

```
#include<bits/stdc++.h>
#define ll long long
#define fi first
#define se second
#define mp make_pair
#define pb push_back
#define pii pair<int,int>
using namespace std;
const int N=1e5+10;
const int M=1e6;
int n,m,num[N*3],tot,lz[N*12],maxn[N*12];
ll res=1;
int read()
{
    int x=0,f=1;char c=getchar();
    while(c<'0'||c>'9'){if(c=='-')f=-1;c=getchar();}
    while(c>='0'&&c<='9'){x=x*10+c-'0';c=getchar();}
    return x*f;
```

```
}
struct Node1{int y,l,r;}a[N];
struct Node2{int x,l,r;};
vector<Node2>b,c;
int id(int x){return lower_bound(num+1,num+1+tot,x)-num;}
void pushdown(int idx)
{
    int t=lz[idx];
    lz[idx]=0,lz[idx<<1]+=t,lz[idx<<1|1]+=t;
    maxn[idx<<1]+=t,maxn[idx<<1|1]+=t;
}
void add(int idx,int l,int r,int a,int b)
{
    if(l>=a&&r<=b){lz[idx]++,maxn[idx]++;return;}
    pushdown(idx);
    int mid=(l+r)>>1;
    if(b<=mid)add(idx<<1,l,mid,a,b);
    else if(a>mid)add(idx<<1|1,mid+1,r,a,b);
    else add(idx<<1,l,mid,a,b),add(idx<<1|1,mid+1,r,a,b);
    maxn[idx]=max(maxn[idx<<1],maxn[idx<<1|1]);
}
int query(int idx,int l,int r,int x)
{
    if(l==r)return maxn[idx];
    pushdown(idx);
    int mid=(l+r)>>1;
    if(x<=mid)return query(idx<<1,l,mid,x);
    else return query(idx<<1|1,mid+1,r,x);
}
int main()
{
    n=read(),m=read();
    for(int i=1;i<=n;i++)
    {
        a[i].y=read(),a[i].l=read(),a[i].r=read();
        num[++tot]=a[i].l,num[++tot]=a[i].r;
        res+=(a[i].l==0&&a[i].r==M);
    }
    sort(a+1,a+1+n,[&](Node1 o1,Node1 o2){return o1.y<o2.y;});
    for(int i=1;i<=m;i++)
    {
        int x=read(),l=read(),r=read();
        if(l==0)b.pb(Node2{x,l,r});else c.pb(Node2{x,l,r});
        num[++tot]=x,res+=(l==0&&r==M);
    }
    sort(num+1,num+1+tot);
    tot=unique(num+1,num+1+tot)-num-1;
    sort(b.begin(),b.end(),[&](Node2 o1,Node2 o2){return o1.r<o2.r;});
    for(int i=0,j=1;i<b.size();i++)
    {
        while(j<=n&&a[j].y<=b[i].r)add(1,1,tot,id(a[j].l),id(a[j].r)),j++;
```

```
        int t=query(1,1,tot,id(b[i].x));
        res+=t;
    }
    memset(lz,0,sizeof(lz));
    memset(maxn,0,sizeof(maxn));
    sort(c.begin(),c.end(),[&](Node2 o1,Node2 o2){return o1.l>o2.l;});
    for(int i=0,j=n;i<c.size();i++)
    {
        while(j>0&&a[j].y>=c[i].l)add(1,1,tot,id(a[j].l),id(a[j].r)),j--;
        int t=query(1,1,tot,id(c[i].x));
        res+=t;
    }
    printf("%lld\n",res);
    return 0;
}
```

F. Reverse and Swap

应该是有一个 $O(nq)$ 的异或一个值的神秘做法但我懂得还有点模糊。线段树这个还蛮好理解的，交换和翻转操作其实都可以归为对线段树的一些层交换左右子树，记录每层是否需要交换，之后就单点修改区间查询，根据是否交换选择进入左还是右子树即可。

```
#include<bits/stdc++.h>
#define ll long long
#define fi first
#define se second
#define mp make_pair
#define pb push_back
#define pii pair<int,int>
using namespace std;
const int N=(1<<18)+10;
int read()
{
    int x=0,f=1;char c=getchar();
    while(c<'0'||c>'9'){if(c=='-')f=-1;c=getchar();}
    while(c>='0'&&c<='9'){x=x*10+c-'0';c=getchar();}
    return x*f;
}
ll a[N],rev[20],sum[N<<2];
void build(int idx,int l,int r)
{
    if(l==r){sum[idx]=a[l];return;}
    int mid=(l+r)>>1;
    build(idx<<1,l,mid),build(idx<<1|1,mid+1,r);
    sum[idx]=sum[idx<<1]+sum[idx<<1|1];
```

```
}
void modify(int idx,int k,int l,int r,int pos,int x)
{
    if(l==r){sum[idx]=x;return;}
    int mid=(l+r)>>1;
    if(!rev[k])
    {
        if(pos>mid)modify(idx<<1|1,k-1,mid+1,r,pos,x);
        else modify(idx<<1,k-1,l,mid,pos,x);
    }
    else
    {
        if(pos>mid)modify(idx<<1,k-1,mid+1,r,pos,x);
        else modify(idx<<1|1,k-1,l,mid,pos,x);
    }
    sum[idx]=sum[idx<<1]+sum[idx<<1|1];
}
ll query(int idx,int k,int l,int r,int a,int b)
{
    if(l>b||r<a)return 0;
    if(a<=l&&b>=r)return sum[idx];
    int mid=(l+r)>>1;
    if(!rev[k])return
query(idx<<1,k-1,l,mid,a,b)+query(idx<<1|1,k-1,mid+1,r,a,b);
    else return
query(idx<<1,k-1,mid+1,r,a,b)+query(idx<<1|1,k-1,l,mid,a,b);
}
int main()
{
    int n=read(),q=read();
    for(int i=1;i<=(1<<n);i++)a[i]=read();
    build(1,1,(1<<n));
    for(int i=1;i<=q;i++)
    {
        int op=read(),x,k,l,r;
        if(op==1)x=read(),k=read(),modify(1,n,1,(1<<n),x,k);
        else if(op==2){k=read();for(int i=1;i<=k;i++)rev[i]^=1;}
        else if(op==3)k=read(),rev[k+1]^=1;
        else l=read(),r=read(),printf("%lld\n",query(1,n,1,(1<<n),l,r));
    }
    return 0;
}
```