题解

计算

简要题意：给定三个数 a, b, c，判断 $a \times b$ 是否等于 c。

随便找一个数，然后对 a, b, c 分别取模，检查一下 a 取模后的结果与 b 取模后的结果的乘积是否和 c 同余就好了。

错误率的话，可以证明，若在 10^{18}内随机取一次数，错误率最多为一千亿分之一。用中国剩余定理证明。

代码实现非常简单，写个快读就好了。

模数要随便选，不然会被卡，虽然说原则上应该用随机数，但是实际上出题人没办法对着你的代码卡，所以随便找一个固定的数取模也是可以的，这题我卡掉了十六个常用模数。

固定模数不是本题解法，固定模数是错解，我肯定要卡错解，不能把错解放过去，随机模数才是正解，本来是打算把所有模数都卡掉的，但是洛谷数据上传大小限制，所以我只能卡了几个常用模数，之所以卡固定模数是让你想正解，证明随机模数的正确性，原则上我应该把所有模数都卡掉的。

所以，理论上是可以卡掉所有固定模数的，只是要很多测试点，实际上不行，所以我只能卡几个常用的。

卡模数的意思是让选手用随机模数，并不是为了恶心人，我已经说过了，随机模数正确性才是对的，不是随机模数就不是正解

所有模数全卡掉，随机一个仍然是对的，因为一个测试点卡掉的模数有限，我全部卡掉是通过增加测试点数量卡掉的。

换句话说，不是故意取构造这样的数据实际上改变了这个题目，而是题目本来就是这样的，是很多人一开始理解错了。

CRT证明我再写详细一点，卡的话就是要找两个同余的数，把你要卡的模数全部都得乘起来，最多只能乘 10^6 个数，因为再多数位就超了，所以最多卡掉 10^6 个数，而你的模数如果在 10^{18} 内随机，错误率就只有 $\frac{1}{10^{12}}$。

参考代码

```
#include<bits/stdc++.h>
#define int unsigned long long
using namespace std;
int t,k,x,y,z,mod;
inline int rd(){
    int s=0;char c=getchar();
    while(c<'0'||c>'9')c=getchar();
    while(c<='9'&&c>='0'){s=((s*10)+(c^48))%mod;c=getchar();}
    return s;
}
signed main(){
    mt19937 myr(time(0));
    mod=myr(); t=rd();
    while(t--){
        x=rd();y=rd();z=rd();
        if((1ll*x*y%mod)==z)
```

```
            printf("YES\n");
        else
            printf("NO\n");
    }
    return 0;
}
```

圆柱

记水的高度为 x，易得方程为 $x^3-3x+3V=0$，二分或公式法解方程即可。

以下是推导过程。

设水的高度为 x，于是水面下方圆柱的体积为 $Sh=1*x=x$，圆锥的体积为 $\frac{1}{3}Sh=\frac{1}{3}S_{small}x$

我们知道，面积是个二维量，高度是一维量，因此我们能直观地看出$S_{small}:S=(x:1)^2$

严格证明请用圆锥竖截面三角形上的相似以及 $S=\pi r^2$ 证明，读者自证不难。

于是，$S_{small}=x^2S=x^2$，有 $V=V_{柱}-V_{锥}=x-\frac{x^3}{3}$，即 $x^3-3x+3V=0$。

参考代码

```
#include<cstdio>
int main()
{
//  freopen("math.in","r",stdin);
//  freopen("math.out","w",stdout);
    int T;double V,l,r;
    scanf("%d",&T);
    while(T--)
    {
        scanf("%lf",&V);
        l=0,r=1;
        while(r-l>=1e-8)
        {
            double mid=(l+r)/2;
            if(mid-(mid*mid*mid)/3>V)r=mid;
            else l=mid;
        }
        printf("%lf\n",l);
    }
    return 0;
}
```

异或月饼

> 给定一个序列，多次在线求一个区间的最大异或字串。

知识点：可持久化；0-1 trie；分块 。

先只想一个区间怎么做。

不妨设 s_i 表示前缀异或和，所以我们只要用 0-1 trie 维护 $\{s_1,s_2,\dots,s_n\}$，查询 s_{l-1} 异或的最大值即可。

对于多组数据，考虑可持久化，单次查询复杂度 $\mathcal{O}(n\log v)$。

考虑继续优化，我们对这个东西分块，记录 $f_{i,j}$ 表示第 i 块到位置 j 的最大异或字串的值。

查询的时候，先枚举所有整块中 $f_{i,r}$ 的值。再对左边不完整的块用 0-1 trie 查询。

复杂度 $\mathcal{O}(n\sqrt{n}\log v)$ 。可以通过本题。

```
#include<bits/stdc++.h>
#define N 30010
using namespace std;
int n,m,q,sum[N],tr[N*32*15][2],bel[N],qn,f[110][12010],qlen,rt[N];
int tot[N*32*15];
int nw[N*32*15];
long long pos(long long x,long long pos){
    return ((x&(1<<pos))>0);
}
int insert(int pre,int num) {
    int RT=++m,al=m;
//  cout<<m<<endl;
    for(int i=30; i>=0; i--) {
        tr[al][0]=tr[pre][0];
        tr[al][1]=tr[pre][1];
        tot[al]=tot[pre]+1;
        tr[al][pos(num,i)]=++m;
        pre=tr[pre][pos(num,i)];
        al=tr[al][pos(num,i)];
    }
    tot[al]=tot[pre]+1;
//  cout<<tot[al]<<endl;
    return RT;
}
int calc(int l,int r,int num) {
    int res=0;
    int ar=rt[r],al=rt[l-1];
    for(int i=30; i>=0; i--) {
        if(tot[tr[ar][!pos(num,i)]]>tot[tr[al][!pos(num,i)]]) {
            ar=tr[ar][!pos(num,i)];
            al=tr[al][!pos(num,i)];
            res=res|(1ll<<i);
        } else {
            ar=tr[ar][pos(num,i)];
            al=tr[al][pos(num,i)];
        }
    }
    return res;
}
long long work(int l,int r) {
    int ans=0;
    for(int i=bel[l]+1; i<=bel[r]; i++) ans=max(ans,f[i][r]);
    for(int i=l; i<=min(bel[l]*qlen,r); i++) ans=max(ans,calc(i,r,sum[i-1]));
    return ans;
}

int main()
{
    cin>>n>>q;
    qlen=sqrt(n);
```

```
    for(int i=1; i<=n; i++) {
        int num;
        cin>>num;
        sum[i]=sum[i-1]^num;
//      cout<<sum[i]<<endl;
        bel[i]=(i-1)/qlen+1;
        rt[i]=insert(rt[i-1],sum[i]);
    }
    qn=bel[n];
    for(int i=1; i<=qn; i++) {
        int l=(i-1)*qlen+1;
        for(int j=l; j<=n; j++)
            f[i][j]=max(f[i][j-1],calc(l-1,j-1,sum[j]));
    }
    long long lst=0;
    while(q--) {
        long long l,r;
        cin>>l>>r;
        l=(l+lst)%n+1,r=(r+lst)%n+1;
        if(l>r) swap(l,r);
        lst=work(l,r);
        cout<<lst<<endl;
    }
}
```

按摩

首先考虑静态时，是否存在一个以黑点作为顶点的所有边均平行于网格的多边形如何判断，注意到，条件其实就是存在一个边平行于网格的环，那么我们将每个点看做连接行 i 与列 j 的边，当存在环时满足条件。

然后考虑动态的，我们只要将修改操作丢到线段树上，线段树分治一下，用可撤销的并查集来判环，就可以了。

```
#include<bits/stdc++.h>
# define N (500005)
# define mid (l+r>>1)
using namespace std;
inline int rd(){
    char c=getchar();int sum=0,f=1;
    while(c<'0'||c>'9'){if(c=='-')f=-1;c=getchar();}
    while(c<='9'&&c>='0'){sum=(sum<<3)+(sum<<1)+(c^48);c=getchar();}
    return sum*f;
}
int u[N],v[N],n,m,k,f[N],d[N];
struct node{
    int l,r;
    vector<pair<int,int> >v;
}t[N];
map<pair<int,int>,int>mp;
stack< pair<int,int> >cl;
int find(int x){
    while(x^f[x])x=f[x];
    return x;
}
```

```
void build(int p,int l,int r){
    t[p].l=l;t[p].r=r;
    if(l==r)return;
    build(p<<1,l,mid);build(p<<1|1,mid+1,r);
    return;
}
void update(int p,int l,int r,pair<int,int> x){
    if(t[p].l>r||t[p].r<l)return;
    if(t[p].l>=l&&t[p].r<=r){
        t[p].v.push_back(x);
        return;
    }
    update(p<<1,l,r,x);
    update(p<<1|1,l,r,x);
    return;
}
void merge(int x,int y){
    x=find(x),y=find(y);
    if(d[x]>d[y])swap(x,y);
    cl.push(make_pair(x,d[x]==d[y]));
    f[x]=y;d[y]+=(d[x]==d[y]);
}
void dfs(int p,int l,int r){
    bool flag=1;
    int st=cl.size(),x,y;
    for(int i=0;i<t[p].v.size();++i){
        pair<int,int> s=t[p].v[i];
        x=find(s.first);y=find(s.second);
        if(x==y){
            flag=0;
            for(int j=l;j<=r;++j)
                printf("Yes\n");
            break;
        }
        merge(s.first,s.second);
    }
    if(flag){
        if(l==r)printf("No\n");
        else {
            dfs(p<<1,l,mid);
            dfs(p<<1|1,mid+1,r);
        }
    }
    while(cl.size()>st){
        x=cl.top().first;
        y=cl.top().second;
        f[x]=x;d[x]-=y;
        cl.pop();
    }
}
signed main(){
    int x,y;
    n=rd();m=rd();
    for(int i=1;i<=2*n;++i)f[i]=i;
    for(int i=1;i<=m;++i){
        x=rd();y=rd()+n;
```

```
        mp[make_pair(x,y)]=1;
    }
    k=rd();
    build(1,1,k);
    for(int i=1;i<=k;++i){
        x=rd();y=rd()+n;
        if(mp[make_pair(x,y)]){
            update(1,mp[make_pair(x,y)],i-1,make_pair(x,y));
            mp.erase(make_pair(x,y));
        }else mp[make_pair(x,y)]=i;
    }
    for(auto i=mp.begin();i!=mp.end();++i)
        update(1,i->second,k,i->first);
    dfs(1,1,k);
    return 0;
}
```