图书馆

题意简述

给定长度为 n 的数列 $a_1, a_2, \ldots, a_n$ 和长度为 m 的数列 $b_1, b_2, \ldots, b_m$，求有多少对 (i, j) 满足 $1 \le i \le n$，$1 \le j \le m$ 且 $\operatorname{lcm}(a_i, b_j)$ 是完全平方数。

值域范围为 10^6。

题解

考虑直接对每个完全平方数，求出 lcm 为它的方案数。

对于完全平方数 a^2，我们考虑枚举所有 x, y，满足 $\operatorname{lcm}(x, y) = a^2$。

若 $a^2 = \prod\limits_k p_k^{\alpha_k}$，则对于每个 k，x, y 中至少有一个包含 $p_k^{\alpha_k}$。我们 2^k 枚举哪些在 x 中，则剩下的在 y 中。

对于没有强制取满的质因子 p_i，它的个数可以在 $0 \sim \alpha_i$ 中任取，为避免重复计算，可以令 y 中的这些 p_i 的个数最多取到 $\alpha_i - 1$。使用搜索找出所有的 x 和 y。

把读入的数列开桶存，然后对搜出的每对 x, y 可以直接计算答案。

这样看上去不太靠谱，考虑加一些优化。

由于值域只有 10^6，所以 x, y 都不应超过 10^6，在超过的时候直接剪枝掉即可。

若一开始就有 $x > 10^6$ 或 $y > 10^6$，则直接不进行搜索。

然后就可以通过了，实测最慢的点只需要 0.3 秒。

事实上由于时限比较宽松，加一些其他的优化也是能够通过的。

决战圣诞树

题意简述

给定一棵 n 个点的无根树，树上每个结点维护一个多项式，初始为 1。

要求支持以下几个操作：

1. 给定 v, a，将结点 v 的多项式乘上 $(1 + x^a)$。
2. 给定 v, a, b，将结点 v 的多项式乘上 $(1 + x^a + x^b)$。
3. 给定 v，将树上与 v 相邻的结点的多项式都乘上结点 v 的多项式。
4. 给定 u, v，保证 u, v 有连边，将以 u 为根时，v 子树内所有结点的多项式乘上结点 u 的多项式。
5. 给定 u, v, s，保证 u, v 有连边，断开这条边，分别求分成的两棵子树中结点的多项式乘积在模 $(x^K - 1)$ 意义下的 s 次项系数。答案对素数 P 取模。

保证 $P \bmod K = 1$。

题解

可以发现，这里的两个多项式相乘是一个循环卷积。每次暴力做的话时间复杂度是 $O(K^2)$ 的。

由循环卷积容易想到 FFT。我们也尝试对要维护的多项式进行 FFT，这样就只需要维护点值信息，两个多项式的乘法只需要 $O(K)$。

我们需要找到模 P 意义下的 K 次单位根，只需求出 P 的原根 g，然后 $g'=g^{\frac{P-1}{K}}$ 就是满足条件的数。

由于单点修改的操作的系数很少，所以我们暴力带入 $1,g',g'^2,\cdots,g'^{K-1}$ 求值即可。原理同 FFT，此处不进行阐述。

考虑操作 3 和 4，如果只有 3，我们可以对树进行 BFS 序标号，用线段树维护点值乘积，则每次操作相当于对 $O(1)$ 个区间做区间乘法。同理，只有操作 4，我们对树进行 DFS 序标号，也可以用线段树维护。

现在这两个操作混合在一起，我们也可以考虑结合 BFS 序和 DFS 序。

我们尝试如下标号方式：将当前根结点标号，然后将该结点的儿子按顺序标号，然后递归每个儿子，重复上述操作进行标号。可以发现，这个方式保证了一个结点的儿子的标号连续，且该结点的**后代**的标号也是连续的。这样我们就可以同时支持上述两种操作。

对于操作 5，我们先在线段树上求出的点值，然后代入 g'^{-s} 求值即可。

算法的时间复杂度为 $O((n+q)K\log n+P)$，标程最慢的点只用了 2 秒左右，9 秒的时限应该不存在卡常的问题。

注意这里的线段树需要支持区间乘，区间求乘积，因此对于长度为 L 的区间里的每个数乘上 x，这个区间的乘积需要乘上 x^L，如果每次都使用快速幂计算，则单次修改的复杂度将为 $O(K\log n\log P)$。由于我们已经求出了 P 的原根，因此可以利用它来 $O(1)$ 求幂（类似 $\ln$ 和 $\exp$）。

抽象画

Point 1,2

点比较少。考虑搜索加剪枝。

需要判断两线段是否相交。~~这是计算几何基础，这里就不再赘述了~~

Code

```
#include<cstdio>
#include<cmath>
typedef long long LL;
struct point{
    int x,y;
    inline point operator-(const point&rhs)const{return(point){x-rhs.x,y-
rhs.y};}
}e[23];
inline int cross(const point&a,const point&b){
    LL ans=a.x*(LL)b.y-a.y*(LL)b.x;
    return ans>0?1:-1;
}
inline double length(const point&a,const point&b){
    return sqrt((LL)(a.x-b.x)*(a.x-b.x)+(LL)(a.y-b.y)*(a.y-b.y));
}
int n,mx[23],my[23],vis[23],ansx[23],ansy[23];
double mxx=0;
void dfs(int nw,int ok,double len){
    if(ok==n){
        if(len>mxx){
            mxx=len;
            for(int i=1;i<=n;++i)ansx[i]=mx[i],ansy[i]=my[i];
        }
        return;
```

```
    }
    if(vis[nw])dfs(nw+1,ok,len);else{
        for(int i=nw+1;i<=n*2;++i)
        if(!vis[i]){
            bool can=1;
            const point a=e[nw]-e[i];
            for(int j=1;j<=ok&&can;++j){
                int fh=cross(a,e[mx[j]]-e[i])+cross(a,e[my[j]]-e[i]),tj=fh==0;
                const point b=e[mx[j]]-e[my[j]];
                fh=cross(b,e[nw]-e[my[j]])+cross(b,e[i]-e[my[j]]);
                if(tj&&fh==0)can=0;
            }
            if(can){
                mx[ok+1]=nw,my[ok+1]=i;
                vis[nw]=vis[i]=1;
                dfs(nw+1,ok+1,len+length(e[nw],e[i]));
                vis[nw]=vis[i]=0;
            }
        }
    }
}
int to[2333];
int main(){
    scanf("%d",&n);
    for(int i=1;i<=2*n;++i)scanf("%d%d",&e[i].x,&e[i].y);
    dfs(1,0,0);
    for(int i=1;i<=n;++i)
    to[ansx[i]]=ansy[i],to[ansy[i]]=ansx[i];
    for(int i=1;i<=2*n;++i)
    printf("%d\n",to[i]);
    return 0;
}
```

Point 3,4

看上去纵坐标相近的横坐标都离很远。

按纵坐标为第一关键字，横坐标为第二关键字排序,然后相邻两个连边。

这个方法也可以跑其他点，这种方法总共能拿 62 分。

Code

```
#include<cstdio>
#include<algorithm>
struct point{
    int x,y,id;
    inline bool operator<(const point&rhs)const{return y!=rhs.y?
y<rhs.y:x<rhs.x;}
}e[2333333];
int n,to[233333];
int main(){
    scanf("%d",&n);
    for(int i=1;i<=2*n;++i)
    scanf("%d%d",&e[i].x,&e[e[i].id=i].y);
    std::sort(e+1,e+n+n+1);
    for(int i=1;i<=2*n;++i)
    if(i%2==1)to[e[i].id]=e[i+1].id;else to[e[i].id]=e[i-1].id;
```

```
    for(int i=1;i<=2*n;++i)printf("%d\n",to[i]);
    return 0;
}
```

Point 5~8

5 和 6 是个圆，7 和 8 是个近似凸包的东西，答案和凸包的方法算出来一样。

直接对其求一下凸包，允许斜率有一定误差就好了。

Code

```
#include<bits/stdc++.h>
using namespace std;
struct point{
    int x,y,id;
    inline bool operator<(const point&rhs)const{return x!=rhs.x?
x<rhs.x:y<rhs.y;}
}a[6666];
inline bool cmp(const point&a,const point&b){
    return a.x!=b.x?a.x>b.x:a.y<b.y;
}
inline bool cmp2(const point&a,const point&b){
    return a.x!=b.x?a.x>b.x:a.y>b.y;
}
int n,to[6666];
double mx=0;
vector<pair<int,int> >v;
vector<point>s;
inline double length(const point&a,const point&b){
    return sqrt(1ll*(a.x-b.x)*(a.x-b.x)+1ll*(a.y-b.y)*(a.y-b.y));
}
inline double slope(const point&a,const point&b){
    //if(b.x-a.x==0)fputs("nan",stderr);
    if(b.x-a.x==0)return 1e17;
    return(b.y-a.y)*1./(b.x-a.x);
}
int sta[6666],top=1;
void init(){
    sort(a+1,a+2*n+1);
    sta[1]=1;
    for(int i=2;i<=2*n;++i){
        if(top==1)sta[++top]=i;else{
            while(top>1&&slope(a[sta[top-
1]],a[sta[top]])-.5>=slope(a[sta[top]],a[i]))--top;
            sta[++top]=i;
        }
    }
    for(int i=1;i<=top;++i)s.push_back(a[sta[i]]),a[sta[i]].x=-2147483647;
    sort(a+1,a+2*n+1,cmp);
    for(int i=1;i<=2*n;++i)
    if(a[i].x!=-2147483647)s.push_back(a[i]);else break;
    for(int i=1;i<=2*n;++i)
    a[i]=s[i-1];
}
int main(){
    scanf("%d",&n);
    for(int i=1;i<=2*n;++i)
```

```
    scanf("%d%d",&a[a[i].id=i].x,&a[i].y);
    init();
    for(int i=1;i<=2*n;++i){
        v.clear();
        double now=0;
        int l=1,r=n;
        while(l<r){
            now+=length(a[l],a[r]);
            v.push_back(make_pair(a[l].id,a[r].id));
            ++l,--r;
        }
        l=n+1,r=2*n;
        while(l<r){
            now+=length(a[l],a[r]);
            v.push_back(make_pair(a[l].id,a[r].id));
            ++l,--r;
        }
        if(now>mx){
            mx=now;
            for(int i=0;i<v.size();++i)
            to[v[i].first]=v[i].second,
            to[v[i].second]=v[i].first;
        }
        a[2*n+1]=a[1];
        for(int i=1;i<=2*n;++i)
        a[i]=a[i+1];
        /*if(a[1].id==1){
            for(int i=1;i<=2*n;++i)
            fprintf(stderr,"%d\n",a[i].id);
        }*/
    }
    for(int i=1;i<=2*n;++i)printf("%d\n",to[i]);
    fprintf(stderr,"%.6f\n",mx);
    return 0;
}
```

Point 9,10

这两个点没有什么特殊性质，考虑一些随机的做法。

由于要保证线段不相交，所以模拟退火等算法的效果并不好。我们需要保证每次都能出一组合法解，这样效果会比较好。

一个靠谱的做法是：每次将所有点绕原点同时旋转一个随机角度，然后按坐标排序，将前 n 个点染红，后 n 个点染蓝。

每次求上凸壳，取上凸壳上连接红点和蓝点的线段（有且只有 1 条）。然后剩下的点继续如此操作。容易说明这样得到的线段不会相交。

然后让电脑跑一会就好了。本机 65 秒能够出解。

Code

```
#include<cstdio>
#include<cmath>
#include<cstdlib>
#include<ctime>
```

```
#include<algorithm>
typedef long double ld;
const ld pi=acosl(-1);
struct point{
    ld x,y,len2,len;
    int id;
    bool flg;
    inline bool operator<(const point&rhs)const{
        return x!=rhs.x?x<rhs.x:y<rhs.y;
    }
}a[4444];
int n,to[4444];
inline ld slope(const point&a,const point&b){
    if(fabsl(b.x-a.x)<1e-7)return 1e18;
    return(b.y-a.y)*1./(b.x-a.x);
}
void rotate(point&a,ld t){
    ld s=sinl(t),c=cosl(t);
    ld x=a.x*c-a.y*s,y=a.x*s+a.y*c;
    a.x=x,a.y=y;
}
ld ans=0,lm;
int sta[4444],top;
int main(){
    freopen("paint9.in","r",stdin);
    freopen("paint9.out","w",stdout);
    FILE* fe=fopen("paint9.ans","r");
    fscanf(fe,"%Lf",&lm);
    fclose(fe);
    scanf("%d",&n);
    for(int i=1;i<=2*n;++i){
        scanf("%Lf%Lf",&a[i].x,&a[i].y);
        a[i].id=i;
        a[i].len2=a[i].x*a[i].x-a[i].y*a[i].y;
        a[i].len=sqrtl(a[i].len2);
    }
    srand(time(0));
    while(lm-ans>1){
        ans=0;
        const ld xz=(rand()<<15|rand())%18000000*1./36000000*pi;
        for(int i=1;i<=2*n;++i)rotate(a[i],xz),a[i].flg=0;
        std::sort(a+1,a+2*n+1);
        int ok=0;
        while(ok!=n){
            top=0;
            for(int i=1;i<=2*n;++i)
            if(!a[i].flg){
                if(top<2)sta[++top]=i;else{
                    while(top>1&&slope(a[sta[top-1]],a[sta[top]])
<slope(a[sta[top]],a[i]))--top;
                    sta[++top]=i;
                }
            }
            for(int i=1;i<top;++i)
            if(sta[i]<=n&&sta[i+1]>n){
                a[sta[i]].flg=a[sta[i+1]].flg=1;
                int x=sta[i],y=sta[i+1];
```

```
                ans+=sqrtl((a[x].x-a[y].x)*(a[x].x-a[y].x)+(a[x].y-a[y].y)*
(a[x].y-a[y].y));
                to[a[x].id]=a[y].id,to[a[y].id]=a[x].id;
                ++ok;
            }
        }
    }
    for(int i=1;i<=2*n;++i)
    printf("%d\n",to[i]);
    fprintf(stderr,"%.6Lf\n",ans);
    return 0;
}
```

不知道有没有更高明的做法。