比赛链接

补题情况

题目	蒋贤蒙	王赵安	王智彪
A	0	0	0
C	0	0	0
E	0	0	0
F	0	0	2
G	2	0	2
I	0	0	0
J	2	0	0

题解

F. Finding Points

题意

给定一个凸包，点按照逆时针给出，然后求凸包内一点，想要这个点与这个凸多边形相邻点组成的 n 个角的最小值最大，求这个最大值 $(4≤n≤100)$

题解

赛场上两分钟出思路，然后看通过率…感觉是不是有坑就没敢写…赛后听说改数据了…血亏！

显然要二分（废话）。

然后对于每一组相邻的点，这个点和这两个点组成的角大于某个角，则这个点一定在这两个点组成的大弓形内，根据圆周角求 n 个圆看面积交即可，然后比赛的时候猜到会有点跑到凸包外面的情况，但是我不会写圆的交再交凸包，这也是我怂了的原因之一，谁知道改数据嘛！

所以就是个板子题…

```
int N;
Point ppx[110];
int main() {
    scanf("%d",&C.n);
    N=C.n;
    for(int i=0;i<C.n;i++) {
        ppx[i].input();
    }
    ppx[C.n]=ppx[0];
    double l=eps,r=2.0*acos(-1)/N;
    while(r-l>1e-15) {
```

```
        double mid=(l+r)/2;
        for (int i=0;i<C.n;i++) {
            C.c[i].r=(ppx[i].distance(ppx[i+1])/2/sin(mid));
            double dtmp=C.c[i].r*cos(mid);
            Point ptmp=(ppx[i]+ppx[i+1])/2;
            Point pptmp=(ppx[i]-ptmp);
            pptmp=pptmp.rotright();
            pptmp=pptmp.trunc(dtmp);
            ptmp=ptmp+pptmp;
            C.c[i].p=ptmp;
        }
        C.getarea();
        if(C.ans[C.n]>1e-20)l=mid;
        else r=mid;
    }
    printf("%.20lf\n",l*180/pi);
    return 0;
}
```

G. Greater Integer, Better LCM

题意

给一个大数的质因数分解形式，设为 c 并给出 a 和 b 求最小的 $x+y$ 使得 $lcm(a+x,b+y)=c$ $(1≤n≤18)$ 代表质因子个数，保证因子幂次和相加不超过 18 $a,b,c≤10^{32}$

题解

类似于数位 dp 我们对每一个质因子进行枚举幂次，并且状压，位为 1 代表这一个因子的幂次取满，不取满为 0 $dp[con]$ 代表这个压缩状态下的相对于 b 的最小代价，也就是 b 最少要补多少才能到这个状态。

于是我们跑出初始每个状态下的最小代价，但是还需要处理 a 我们把每一个末状态放进 vec 里，然后看哪些比 a 大，代价是存的 v 值减 a 剩下至少需要满足 $(1<<n)-1-con$ 的状态，于是我们需要找一个无后效性的求状态数组的方法，就是让 dp 数组变成至少满足这个状态的最小代价。再处理一下就好了

```
#include <bits/stdc++.h>
using namespace std;
#define ll __int128

inline void scan(ll &X) {
    X = 0;
    int w=0;
    char ch=0;
```

```
    while(!isdigit(ch)) {
        w|=ch=='-';
        ch=getchar();
    }
    while(isdigit(ch)) X=(X<<3)+(X<<1)+(ch^48),ch=getchar();
    if (w) X = -X;
}
void print(ll x) {
    if (!x) return ;
    if (x < 0) putchar('-'),x = -x;
    print(x / 10);
    putchar(x % 10 + '0');
}

ll n,p[110],q[110],a,b;
ll dp[270000];
vector<pair<ll,int> > d;

void dfs(ll pos,ll value,int con) {
    if(pos==n) {
        d.push_back(make_pair(value,con));
        if(value>=b) dp[con]=value-b;
        return;
    }
    for(int i=0;i<=q[pos];i++) {
        dfs(pos+1,value,(i==q[pos])?(con|(1<<pos)):con);
        value=value*p[pos];
    }
}

int main() {
    memset(dp,0x3f3f3f3f,sizeof(dp));
    scan(n);
    for(int i=0;i<n;i++) scan(p[i]),scan(q[i]);
    scan(a);
    scan(b);
    dfs(0,1,0);
    for(int i=0;i<n;i++) {
        for(int j=0;j<(1<<n);j++) {
            if(!(j&(1<<i))) dp[j]=min(dp[j],dp[j+(1<<i)]);
        }
    }
    ll ans=1e36;
    for (int i=0;i<d.size();i++) {
        pair<ll,int> x=d[i];
        if (x.first>=a) ans=min(ans,x.first-a+dp[(1<<n)-1-x.second]);
    }
    if(ans) print(ans);
    else puts("0");
    return 0;
```

```
}
```

被吊打的标算

考虑枚举 $S=\{p_1,p_2\cdots p_n\}$ 的所有子集。

对每个子集 $T=\{a_1,a_2\cdots a_i\}$，强制令 x 取 $\{p_1,p_2\cdots p_n\}-T$ 的每个素因子的最高次幂，强制令 y 取 T 的每个素因子的最高次幂。

这样，就消除了 $\text{lcm}(x,y)=c$ 的限制，接下来分别考虑 $x\ge a,y\ge b$ 的限制。

不难发现 x 在 $a_1,a_2\cdots a_i$ 的幂次都是自由的，设 $x=t\cfrac{c}{a_1^{q_1}a_2^{q_2}\cdots a_i^{q_i}}$，因此需要找到最小的 $t\mid a_1^{q_1}a_2^{q_2}\cdots a_i^{q_i}$，且 $x\ge a$。

一种暴力解法为直接枚举 $a_1^{q_1}a_2^{q_2}\cdots a_i^{q_i}$ 的所有因子，最坏情况下 c 有 n 个因子，每个因子幂次均为 1。

此时时间复杂度等价于子集枚举套子集枚举的时间复杂度，可以认为是

$$ \sum_{i=0}^n {n\choose i}2^i=(1+2)^n $$

考虑一个优化，将 $a_1^{q_1}a_2^{q_2}\cdots a_i^{q_i}$ 平均分成两个序列，每个序列枚举因子，对一个序列的因子进行排序，然后另一个序列进行二分查找。

这样里层子集枚举的复杂度优化为 $O\left(n{\sqrt 2}^n\right)$，总时间复杂度为

$$ \sum_{i=0}^n {n\choose i}i{\sqrt 2}^i=(1+\sqrt 2)^{n+1} $$

```
const int MAXN=18;
int n,p[MAXN],q[MAXN];
LL A[1<<MAXN],B[1<<MAXN];
LL vec1[1<<MAXN],vec2[1<<MAXN];
vector<pair<int,int> > d;
int cal(int l,int r,LL *vec){
    int n=0;
    vec[n++]=1;
    _for(i,l,r){
        int last=n;
        LL x=1;
        _for(j,0,d[i].second){
            x*=d[i].first;
            _for(k,0,last)
            vec[n++]=vec[k]*x;
        }
    }
    return n;
}
void solve(LL v,LL *ans){
    _for(i,0,1<<n){
```

```
        LL base=1;
        d.clear();
        _for(j,0,n){
            if(i&(1<<j))
            d.push_back(make_pair(p[j],q[j]));
            else{
                _for(k,0,q[j])
                base*=p[j];
            }
        }
        int n1=cal(0,d.size()/2,vec1);
        int n2=cal(d.size()/2,d.size(),vec2);
        sort(vec1,vec1+n1);
        ans[i]=1e32;
        _for(j,0,n2){
            vec2[j]*=base;
            int pos=lower_bound(vec1,vec1+n1,(v+vec2[j]-1)/vec2[j])-vec1;
            if(pos!=n1)
            ans[i]=min(ans[i],vec1[pos]*vec2[j]);
        }
    }
}
int main(){
    n=read_int();
    _for(i,0,n){
        p[i]=read_int();
        q[i]=read_int();
    }
    LL a=read_LL(),b=read_LL();
    solve(a,A);
    solve(b,B);
    LL ans=1e32;
    int S=(1<<n)-1;
    _for(i,0,1<<n)
    ans=min(ans,A[i]+B[S^i]-a-b);
    enter(ans);
    return 0;
}
```