EarthCat 补充22-10-27

jls计算几何

```
1
2  #define mp make_pair
3  #define fi first
4  #define se second
5  #define pb push_back
6  typedef double db;
7  const db eps=1e-6;
8  const db pi=acos(-1);
9  int sign(db k){
10     if (k>eps) return 1; else if (k<-eps) return -1; return 0;
11 }
12 int cmp(db k1,db k2){return sign(k1-k2);}
13 int inmid(db k1,db k2,db k3){return sign(k1-k3)*sign(k2-k3)<=0;}// k3 在
   [k1,k2] 内
14 struct point{
15     db x,y;
16     point operator + (const point &k1) const{return (point)
   {k1.x+x,k1.y+y};}
17     point operator - (const point &k1) const{return (point){x-k1.x,y-
   k1.y};}
18     point operator * (db k1) const{return (point){x*k1,y*k1};}
19     point operator / (db k1) const{return (point){x/k1,y/k1};}
20     int operator == (const point &k1) const{return
   cmp(x,k1.x)==0&&cmp(y,k1.y)==0;}
21     // 逆时针旋转
22     point turn(db k1){return (point){x*cos(k1)-
   y*sin(k1),x*sin(k1)+y*cos(k1)};}
23     point turn90(){return (point){-y,x};}
24     bool operator < (const point k1) const{
25         int a=cmp(x,k1.x);
26         if (a==-1) return 1; else if (a==1) return 0; else return
   cmp(y,k1.y)==-1;
27     }
28     db abs(){return sqrt(x*x+y*y);}
29     db abs2(){return x*x+y*y;}
```

```
    db dis(point k1){return ((*this)-k1).abs();}
    point unit(){db w=abs(); return (point){x/w,y/w};}
    void scan(){double k1,k2; scanf("%lf%lf",&k1,&k2); x=k1; y=k2;}
    void print(){printf("%.11lf %.11lf\n",x,y);}
    db getw(){return atan2(y,x);}
    point getdel(){if (sign(x)==-1||(sign(x)==0&&sign(y)==-1)) return (*this)*(-1); else return (*this);}
    int getP() const{return sign(y)==1||(sign(y)==0&&sign(x)==-1);}
};
int inmid(point k1,point k2,point k3){return inmid(k1.x,k2.x,k3.x)&&inmid(k1.y,k2.y,k3.y);}
db cross(point k1,point k2){return k1.x*k2.y-k1.y*k2.x;}
db dot(point k1,point k2){return k1.x*k2.x+k1.y*k2.y;}
db rad(point k1,point k2){return atan2(cross(k1,k2),dot(k1,k2));}
// -pi -> pi
int compareangle (point k1,point k2){
    return k1.getP()<k2.getP()||(k1.getP()==k2.getP()&&sign(cross(k1,k2))>0);
}
point proj(point k1,point k2,point q){ // q 到直线 k1,k2 的投影
    point k=k2-k1; return k1+k*(dot(q-k1,k)/k.abs2());
}
point reflect(point k1,point k2,point q){return proj(k1,k2,q)*2-q;}
int clockwise(point k1,point k2,point k3){// k1 k2 k3 逆时针 1 顺时针 -1 否则 0
    return sign(cross(k2-k1,k3-k1));
}
int checkLL(point k1,point k2,point k3,point k4){// 求直线 (L) 线段 (S)k1,k2 和 k3,k4 的交点
    return cmp(cross(k3-k1,k4-k1),cross(k3-k2,k4-k2))!=0;
}
point getLL(point k1,point k2,point k3,point k4){
    db w1=cross(k1-k3,k4-k3),w2=cross(k4-k3,k2-k3); return (k1*w2+k2*w1)/(w1+w2);
}
int intersect(db l1,db r1,db l2,db r2){
    if (l1>r1) swap(l1,r1); if (l2>r2) swap(l2,r2); return cmp(r1,l2)!=-1&&cmp(r2,l1)!=-1;
}
int checkSS(point k1,point k2,point k3,point k4){
    return intersect(k1.x,k2.x,k3.x,k4.x)&&intersect(k1.y,k2.y,k3.y,k4.y)&&
    sign(cross(k3-k1,k4-k1))*sign(cross(k3-k2,k4-k2))<=0&&
    sign(cross(k1-k3,k2-k3))*sign(cross(k1-k4,k2-k4))<=0;
}
db disSP(point k1,point k2,point q){
    point k3=proj(k1,k2,q);
    if (inmid(k1,k2,k3)) return q.dis(k3); else return min(q.dis(k1),q.dis(k2));
}
db disSS(point k1,point k2,point k3,point k4){
    if (checkSS(k1,k2,k3,k4)) return 0;
    else return min(min(disSP(k1,k2,k3),disSP(k1,k2,k4)),min(disSP(k3,k4,k1),disSP(k3,k4,k2)));
}
int onS(point k1,point k2,point q){return inmid(k1,k2,q)&&sign(cross(k1-q,k2-k1))==0;}
struct circle{
```

```
    point o; db r;
    void scan(){o.scan(); scanf("%lf",&r);}
    int inside(point k){return cmp(r,o.dis(k));}
};
struct line{
    // p[0]->p[1]
    point p[2];
    line(point k1,point k2){p[0]=k1; p[1]=k2;}
    point& operator [] (int k){return p[k];}
    int include(point k){return sign(cross(p[1]-p[0],k-p[0]))>0;}
    point dir(){return p[1]-p[0];}
    line push(){ // 向外 ( 左手边 ) 平移 eps
        const db eps = 1e-6;
        point delta=(p[1]-p[0]).turn90().unit()*eps;
        return {p[0]-delta,p[1]-delta};
    }
};
point getLL(line k1,line k2){return getLL(k1[0],k1[1],k2[0],k2[1]);}
int parallel(line k1,line k2){return sign(cross(k1.dir(),k2.dir()))==0;}
int sameDir(line k1,line k2){return
parallel(k1,k2)&&sign(dot(k1.dir(),k2.dir()))==1;}
int operator < (line k1,line k2){
    if (sameDir(k1,k2)) return k2.include(k1[0]);
    return compareangle(k1.dir(),k2.dir());
}
int checkpos(line k1,line k2,line k3){return k3.include(getLL(k1,k2));}
vector<line> getHL(vector<line> &L){ // 求半平面交 , 半平面是逆时针方向 , 输出按
照逆时针
    sort(L.begin(),L.end()); deque<line> q;
    for (int i=0;i<(int)L.size();i++){
        if (i&&sameDir(L[i],L[i-1])) continue;
        while (q.size()>1&&!checkpos(q[q.size()-2],q[q.size()-1],L[i]))
q.pop_back();
        while (q.size()>1&&!checkpos(q[1],q[0],L[i])) q.pop_front();
        q.push_back(L[i]);
    }
    while (q.size()>2&&!checkpos(q[q.size()-2],q[q.size()-1],q[0]))
q.pop_back();
    while (q.size()>2&&!checkpos(q[1],q[0],q[q.size()-1])) q.pop_front();
    vector<line>ans; for (int i=0;i<q.size();i++) ans.push_back(q[i]);
    return ans;
}
db closepoint(vector<point>&A,int l,int r){ // 最近点对 , 先要按照 x 坐标排序
    if (r-l<=5){
        db ans=1e20;
        for (int i=l;i<=r;i++) for (int j=i+1;j<=r;j++)
ans=min(ans,A[i].dis(A[j]));
        return ans;
    }
    int mid=l+r>>1; db ans=min(closepoint(A,l,mid),closepoint(A,mid+1,r));
    vector<point>B; for (int i=l;i<=r;i++) if (abs(A[i].x-A[mid].x)<=ans)
B.push_back(A[i]);
    sort(B.begin(),B.end(),[](point k1,point k2){return k1.y<k2.y;});
    for (int i=0;i<B.size();i++) for (int j=i+1;j<B.size()&&B[j].y-
B[i].y<ans;j++) ans=min(ans,B[i].dis(B[j]));
    return ans;
}
int checkposCC(circle k1,circle k2){// 返回两个圆的公切线数量
```

```
    if (cmp(k1.r,k2.r)==-1) swap(k1,k2);
    db dis=k1.o.dis(k2.o);  int w1=cmp(dis,k1.r+k2.r),w2=cmp(dis,k1.r-k2.r);
    if (w1>0) return 4; else if (w1==0) return 3; else if (w2>0) return 2;
    else if (w2==0) return 1; else return 0;
}
vector<point> getCL(circle k1,point k2,point k3){ // 沿着 k2->k3 方向给出 , 相切给出两个
    point k=proj(k2,k3,k1.o); db d=k1.r*k1.r-(k-k1.o).abs2();
    if (sign(d)==-1) return {};
    point del=(k3-k2).unit()*sqrt(max((db)0.0,d)); return {k-del,k+del};
}
vector<point> getCC(circle k1,circle k2){// 沿圆 k1 逆时针给出 , 相切给出两个
    int pd=checkposCC(k1,k2); if (pd==0||pd==4) return {};
    db a=(k2.o-k1.o).abs2(),cosA=(k1.r*k1.r+a-k2.r*k2.r)/(2*k1.r*sqrt(max(a,(db)0.0)));
    db b=k1.r*cosA,c=sqrt(max((db)0.0,k1.r*k1.r-b*b));
    point k=(k2.o-k1.o).unit(),m=k1.o+k*b,del=k.turn90()*c;
    return {m-del,m+del};
}
vector<point> TangentCP(circle k1,point k2){// 沿圆 k1 逆时针给出
    db a=(k2-k1.o).abs(),b=k1.r*k1.r/a,c=sqrt(max((db)0.0,k1.r*k1.r-b*b));
    point k=(k2-k1.o).unit(),m=k1.o+k*b,del=k.turn90()*c;
    return {m-del,m+del};
}
vector<line> TangentoutCC(circle k1,circle k2){
    int pd=checkposCC(k1,k2); if (pd==0) return {};
    if (pd==1){point k=getCC(k1,k2)[0]; return {(line){k,k}};}
    if (cmp(k1.r,k2.r)==0){
        point del=(k2.o-k1.o).unit().turn90().getdel();
        return {(line){k1.o-del*k1.r,k2.o-del*k2.r},(line){k1.o+del*k1.r,k2.o+del*k2.r}};
    } else {
        point p=(k2.o*k1.r-k1.o*k2.r)/(k1.r-k2.r);
        vector<point>A=TangentCP(k1,p),B=TangentCP(k2,p);
        vector<line>ans; for (int i=0;i<A.size();i++) ans.push_back((line){A[i],B[i]});
        return ans;
    }
}
vector<line> TangentinCC(circle k1,circle k2){
    int pd=checkposCC(k1,k2); if (pd<=2) return {};
    if (pd==3){point k=getCC(k1,k2)[0]; return {(line){k,k}};}
    point p=(k2.o*k1.r+k1.o*k2.r)/(k1.r+k2.r);
    vector<point>A=TangentCP(k1,p),B=TangentCP(k2,p);
    vector<line>ans; for (int i=0;i<A.size();i++) ans.push_back((line){A[i],B[i]});
    return ans;
}
vector<line> TangentCC(circle k1,circle k2){
    int flag=0; if (k1.r<k2.r) swap(k1,k2),flag=1;
    vector<line>A=TangentoutCC(k1,k2),B=TangentinCC(k1,k2);
    for (line k:B) A.push_back(k);
    if (flag) for (line &k:A) swap(k[0],k[1]);
    return A;
}
db getarea(circle k1,point k2,point k3){
    // 圆 k1 与三角形 k2 k3 k1.o 的有向面积交
```

```
    point k=k1.o; k1.o=k1.o-k; k2=k2-k; k3=k3-k;
    int pd1=k1.inside(k2),pd2=k1.inside(k3);
    vector<point>A=getCL(k1,k2,k3);
    if (pd1>=0){
        if (pd2>=0) return cross(k2,k3)/2;
        return k1.r*k1.r*rad(A[1],k3)/2+cross(k2,A[1])/2;
    } else if (pd2>=0){
        return k1.r*k1.r*rad(k2,A[0])/2+cross(A[0],k3)/2;
    }else {
        int pd=cmp(k1.r,disSP(k2,k3,k1.o));
        if (pd<=0) return k1.r*k1.r*rad(k2,k3)/2;
        return cross(A[0],A[1])/2+k1.r*k1.r*(rad(k2,A[0])+rad(A[1],k3))/2;
    }
}
circle getcircle(point k1,point k2,point k3){
    db a1=k2.x-k1.x,b1=k2.y-k1.y,c1=(a1*a1+b1*b1)/2;
    db a2=k3.x-k1.x,b2=k3.y-k1.y,c2=(a2*a2+b2*b2)/2;
    db d=a1*b2-a2*b1;
    point o=(point){k1.x+(c1*b2-c2*b1)/d,k1.y+(a1*c2-a2*c1)/d};
    return (circle){o,k1.dis(o)};
}
circle getScircle(vector<point> A){
    random_shuffle(A.begin(),A.end());
    circle ans=(circle){A[0],0};
    for (int i=1;i<A.size();i++)
        if (ans.inside(A[i])==-1){
            ans=(circle){A[i],0};
            for (int j=0;j<i;j++)
                if (ans.inside(A[j])==-1){
                    ans.o=(A[i]+A[j])/2; ans.r=ans.o.dis(A[i]);
                    for (int k=0;k<j;k++)
                        if (ans.inside(A[k])==-1)
                            ans=getcircle(A[i],A[j],A[k]);
                }
        }
    return ans;
}
db area(vector<point> A){ // 多边形用 vector<point> 表示 , 逆时针
    db ans=0;
    for (int i=0;i<A.size();i++) ans+=cross(A[i],A[(i+1)%A.size()]);
    return ans/2;
}
int checkconvex(vector<point>A){
    int n=A.size(); A.push_back(A[0]); A.push_back(A[1]);
    for (int i=0;i<n;i++) if (sign(cross(A[i+1]-A[i],A[i+2]-A[i]))==-1)
return 0;
    return 1;
}
int contain(vector<point>A,point q){ // 2 内部 1 边界 0 外部
    int pd=0; A.push_back(A[0]);
    for (int i=1;i<A.size();i++){
        point u=A[i-1],v=A[i];
        if (onS(u,v,q)) return 1; if (cmp(u.y,v.y)>0) swap(u,v);
        if (cmp(u.y,q.y)>=0||cmp(v.y,q.y)<0) continue;
        if (sign(cross(u-v,q-v))<0) pd^=1;
    }
    return pd<<1;
}
```

```cpp
vector<point> ConvexHull(vector<point>A,int flag=1){ // flag=0 不严格 flag=1 严格
    int n=A.size(); vector<point>ans(n*2);
    sort(A.begin(),A.end()); int now=-1;
    for (int i=0;i<A.size();i++){
        while (now>0&&sign(cross(ans[now]-ans[now-1],A[i]-ans[now-1]))<flag) now--;
        ans[++now]=A[i];
    } int pre=now;
    for (int i=n-2;i>=0;i--){
        while (now>pre&&sign(cross(ans[now]-ans[now-1],A[i]-ans[now-1]))<flag) now--;
        ans[++now]=A[i];
    } ans.resize(now); return ans;
}
db convexDiameter(vector<point>A){
    int now=0,n=A.size(); db ans=0;
    for (int i=0;i<A.size();i++){
        now=max(now,i);
        while (1){
            db k1=A[i].dis(A[now%n]),k2=A[i].dis(A[(now+1)%n]);
            ans=max(ans,max(k1,k2)); if (k2>k1) now++; else break;
        }
    }
    return ans;
}
int rotating_calipers()  //卡壳
{
    int i , q=1;
    int ans = 0;
    stack[top]=0;
    for(i = 0 ; i < top ; i++)
    {
        while( xmult( p[stack[i+1]] , p[stack[q+1]] , p[stack[i]] ) >
            xmult( p[stack[i+1]] , p[stack[q]] , p[stack[i]] ) )
            q = (q+1)%(top);
        ans = max(ans , max( dis(p[stack[i]] , p[stack[q]]) ,
            dis(p[stack[i+1]] , p[stack[q+1]])));
    }
    return ans;
}
vector<point> convexcut(vector<point>A,point k1,point k2){
    // 保留 k1,k2,p 逆时针的所有点
    int n=A.size(); A.push_back(A[0]); vector<point>ans;
    for (int i=0;i<n;i++){
        int w1=clockwise(k1,k2,A[i]),w2=clockwise(k1,k2,A[i+1]);
        if (w1>=0) ans.push_back(A[i]);
        if (w1*w2<0) ans.push_back(getLL(k1,k2,A[i],A[i+1]));
    }
    return ans;
}
int checkPoS(vector<point>A,point k1,point k2){
    // 多边形 A 和直线 ( 线段 )k1->k2 严格相交 , 注释部分为线段
    struct ins{
        point m,u,v;
        int operator < (const ins& k) const {return m<k.m;}
    }; vector<ins>B;
    //if (contain(A,k1)==2||contain(A,k2)==2) return 1;
```

```
    vector<point>poly=A; A.push_back(A[0]);
    for (int i=1;i<A.size();i++) if (checkLL(A[i-1],A[i],k1,k2)){
        point m=getLL(A[i-1],A[i],k1,k2);
        if (inmid(A[i-1],A[i],m)/*&&inmid(k1,k2,m)*/) B.push_back((ins)
{m,A[i-1],A[i]});
    }
    if (B.size()==0) return 0; sort(B.begin(),B.end());
    int now=1; while (now<B.size()&&B[now].m==B[0].m) now++;
    if (now==B.size()) return 0;
    int flag=contain(poly,(B[0].m+B[now].m)/2);
    if (flag==2) return 1;
    point d=B[now].m-B[0].m;
    for (int i=now;i<B.size();i++){
        if (!(B[i].m==B[i-1].m)&&flag==2) return 1;
        int tag=sign(cross(B[i].v-B[i].u,B[i].m+d-B[i].u));
        if (B[i].m==B[i].u||B[i].m==B[i].v) flag+=tag; else flag+=tag*2;
    }
    //return 0;
    return flag==2;
}
int checkinp(point r,point l,point m){
    if (compareangle(l,r)){return compareangle(l,m)&&compareangle(m,r);}
    return compareangle(l,m)||compareangle(m,r);
}
int checkPosFast(vector<point>A,point k1,point k2){ // 快速检查线段是否和多边形
严格相交
    if (contain(A,k1)==2||contain(A,k2)==2) return 1; if (k1==k2) return 0;
    A.push_back(A[0]); A.push_back(A[1]);
    for (int i=1;i+1<A.size();i++)
        if (checkLL(A[i-1],A[i],k1,k2)){
            point now=getLL(A[i-1],A[i],k1,k2);
            if (inmid(A[i-1],A[i],now)==0||inmid(k1,k2,now)==0) continue;
            if (now==A[i]){
                if (A[i]==k2) continue;
                point pre=A[i-1],ne=A[i+1];
                if (checkinp(pre-now,ne-now,k2-now)) return 1;
            } else if (now==k1){
                if (k1==A[i-1]||k1==A[i]) continue;
                if (checkinp(A[i-1]-k1,A[i]-k1,k2-k1)) return 1;
            } else if (now==k2||now==A[i-1]) continue;
            else return 1;
        }
    return 0;
}
// 拆分凸包成上下凸壳 凸包尽量都随机旋转一个角度来避免出现相同横坐标
// 尽量特判只有一个点的情况 凸包逆时针
void getUDP(vector<point>A,vector<point>&U,vector<point>&D){
    db l=1e100,r=-1e100;
    for (int i=0;i<A.size();i++) l=min(l,A[i].x),r=max(r,A[i].x);
    int wherel,wherer;
    for (int i=0;i<A.size();i++) if (cmp(A[i].x,l)==0) wherel=i;
    for (int i=A.size();i;i--) if (cmp(A[i-1].x,r)==0) wherer=i-1;
    U.clear(); D.clear(); int now=wherel;
    while (1){D.push_back(A[now]); if (now==wherer) break; now++; if
(now>=A.size()) now=0;}
    now=wherel;
    while (1){U.push_back(A[now]); if (now==wherer) break; now--; if
(now<0) now=A.size()-1;}
```

```
}
// 需要保证凸包点数大于等于 3,2 内部 ,1 边界 ,0 外部
int containCoP(const vector<point>&U,const vector<point>&D,point k){
    db lx=U[0].x,rx=U[U.size()-1].x;
    if (k==U[0]||k==U[U.size()-1]) return 1;
    if (cmp(k.x,lx)==-1||cmp(k.x,rx)==1) return 0;
    int where1=lower_bound(U.begin(),U.end(),(point){k.x,-1e100})-
U.begin();
    int where2=lower_bound(D.begin(),D.end(),(point){k.x,-1e100})-
D.begin();
    int w1=clockwise(U[where1-1],U[where1],k),w2=clockwise(D[where2-
1],D[where2],k);
    if (w1==1||w2==-1) return 0; else if (w1==0||w2==0) return 1; return 2;
}
// d 是方向 , 输出上方切点和下方切点
pair<point,point> getTangentCow(const vector<point> &U,const vector<point>
&D,point d){
    if (sign(d.x)<0||(sign(d.x)==0&&sign(d.y)<0)) d=d*(-1);
    point whereU,whereD;
    if (sign(d.x)==0) return mp(U[0],U[U.size()-1]);
    int l=0,r=U.size()-1,ans=0;
    while (l<r){int mid=l+r>>1; if (sign(cross(U[mid+1]-U[mid],d))<=0)
l=mid+1,ans=mid+1; else r=mid;}
    whereU=U[ans]; l=0,r=D.size()-1,ans=0;
    while (l<r){int mid=l+r>>1; if (sign(cross(D[mid+1]-D[mid],d))>=0)
l=mid+1,ans=mid+1; else r=mid;}
    whereD=D[ans]; return mp(whereU,whereD);
}
// 先检查 contain, 逆时针给出
pair<point,point> getTangentCoP(const vector<point>&U,const
vector<point>&D,point k){
    db lx=U[0].x,rx=U[U.size()-1].x;
    if (k.x<lx){
        int l=0,r=U.size()-1,ans=U.size()-1;
        while (l<r){int mid=l+r>>1; if (clockwise(k,U[mid],U[mid+1])==1)
l=mid+1; else ans=mid,r=mid;}
        point w1=U[ans]; l=0,r=D.size()-1,ans=D.size()-1;
        while (l<r){int mid=l+r>>1; if (clockwise(k,D[mid],D[mid+1])==-1)
l=mid+1; else ans=mid,r=mid;}
        point w2=D[ans]; return mp(w1,w2);
    } else if (k.x>rx){
        int l=1,r=U.size(),ans=0;
        while (l<r){int mid=l+r>>1; if (clockwise(k,U[mid],U[mid-1])==-1)
r=mid; else ans=mid,l=mid+1;}
        point w1=U[ans]; l=1,r=D.size(),ans=0;
        while (l<r){int mid=l+r>>1; if (clockwise(k,D[mid],D[mid-1])==1)
r=mid; else ans=mid,l=mid+1;}
        point w2=D[ans]; return mp(w2,w1);
    } else {
        int where1=lower_bound(U.begin(),U.end(),(point){k.x,-1e100})-
U.begin();
        int where2=lower_bound(D.begin(),D.end(),(point){k.x,-1e100})-
D.begin();
        if ((k.x==lx&&k.y>U[0].y)||(where1&&clockwise(U[where1-
1],U[where1],k)==1)){
            int l=1,r=where1+1,ans=0;
            while (l<r){int mid=l+r>>1; if (clockwise(k,U[mid],U[mid-
1])==1) ans=mid,l=mid+1; else r=mid;}
```

```
            point w1=U[ans]; l=where1,r=U.size()-1,ans=U.size()-1;
            while (l<r){int mid=l+r>>1; if (clockwise(k,U[mid],U[mid+1])==1) l=mid+1; else ans=mid,r=mid;}
            point w2=U[ans]; return mp(w2,w1);
        } else {
            int l=1,r=where2+1,ans=0;
            while (l<r){int mid=l+r>>1; if (clockwise(k,D[mid],D[mid-1])==-1) ans=mid,l=mid+1; else r=mid;}
            point w1=D[ans]; l=where2,r=D.size()-1,ans=D.size()-1;
            while (l<r){int mid=l+r>>1; if (clockwise(k,D[mid],D[mid+1])==-1) l=mid+1; else ans=mid,r=mid;}
            point w2=D[ans]; return mp(w1,w2);
        }
    }
}
struct P3{
    db x,y,z;
    P3 operator + (P3 k1){return (P3){x+k1.x,y+k1.y,z+k1.z};}
    P3 operator - (P3 k1){return (P3){x-k1.x,y-k1.y,z-k1.z};}
    P3 operator * (db k1){return (P3){x*k1,y*k1,z*k1};}
    P3 operator / (db k1){return (P3){x/k1,y/k1,z/k1};}
    db abs2(){return x*x+y*y+z*z;}
    db abs(){return sqrt(x*x+y*y+z*z);}
    P3 unit(){return (*this)/abs();}
    int operator < (const P3 k1) const{
        if (cmp(x,k1.x)!=0) return x<k1.x;
        if (cmp(y,k1.y)!=0) return y<k1.y;
        return cmp(z,k1.z)==-1;
    }
    int operator == (const P3 k1){
        return cmp(x,k1.x)==0&&cmp(y,k1.y)==0&&cmp(z,k1.z)==0;
    }
    void scan(){
        double k1,k2,k3; scanf("%lf%lf%lf",&k1,&k2,&k3);
        x=k1; y=k2; z=k3;
    }
};
P3 cross(P3 k1,P3 k2){return (P3){k1.y*k2.z-k1.z*k2.y,k1.z*k2.x-k1.x*k2.z,k1.x*k2.y-k1.y*k2.x};}
db dot(P3 k1,P3 k2){return k1.x*k2.x+k1.y*k2.y+k1.z*k2.z;}
//p=(3,4,5),l=(13,19,21),theta=85 ans=(2.83,4.62,1.77)
P3 turn3D(db k1,P3 l,P3 p){
    l=l.unit(); P3 ans; db c=cos(k1),s=sin(k1);
    ans.x=p.x*(l.x*l.x*(1-c)+c)+p.y*(l.x*l.y*(1-c)-l.z*s)+p.z*(l.x*l.z*(1-c)+l.y*s);
    ans.y=p.x*(l.x*l.y*(1-c)+l.z*s)+p.y*(l.y*l.y*(1-c)+c)+p.z*(l.y*l.z*(1-c)-l.x*s);
    ans.z=p.x*(l.x*l.z*(1-c)-l.y*s)+p.y*(l.y*l.z*(1-c)+l.x*s)+p.z*(l.z*l.z*(1-c)+c);
    return ans;
}
typedef vector<P3> VP;
typedef vector<VP> VVP;
db Acos(db x){return acos(max(-(db)1,min(x,(db)1)));}
// 球面距离 , 圆心原点 , 半径 1
db Odist(P3 a,P3 b){db r=Acos(dot(a,b)); return r;}
db r; P3 rnd;
vector<db> solve(db a,db b,db c){
```

```
    db r=sqrt(a*a+b*b),th=atan2(b,a);
    if (cmp(c,-r)==-1) return {0};
    else if (cmp(r,c)<=0) return {1};
    else {
        db tr=pi-Acos(c/r); return {th+pi-tr,th+pi+tr};
    }
}
vector<db> jiao(P3 a,P3 b){
    // dot(rd+x*cos(t)+y*sin(t),b) >= cos(r)
    if (cmp(Odist(a,b),2*r)>0) return {0};
    P3 rd=a*cos(r),z=a.unit(),y=cross(z,rnd).unit(),x=cross(y,z).unit();
    vector<db> ret = solve(-(dot(x,b)*sin(r)),-(dot(y,b)*sin(r)),-(cos(r)-
dot(rd,b)));
    return ret;
}
db norm(db x,db l=0,db r=2*pi){ // change x into [l,r)
    while (cmp(x,l)==-1) x+=(r-l); while (cmp(x,r)>=0) x-=(r-l);
    return x;
}
db disLP(P3 k1,P3 k2,P3 q){
    return (cross(k2-k1,q-k1)).abs()/(k2-k1).abs();
}
db disLL(P3 k1,P3 k2,P3 k3,P3 k4){
    P3 dir=cross(k2-k1,k4-k3); if (sign(dir.abs())==0) return
disLP(k1,k2,k3);
    return fabs(dot(dir.unit(),k1-k2));
}
VP getFL(P3 p,P3 dir,P3 k1,P3 k2){
    db a=dot(k2-p,dir),b=dot(k1-p,dir),d=a-b;
    if (sign(fabs(d))==0) return {};
    return {(k1*a-k2*b)/d};
}
VP getFF(P3 p1,P3 dir1,P3 p2,P3 dir2){// 返回一条线
    P3 e=cross(dir1,dir2),v=cross(dir1,e);
    db d=dot(dir2,v); if (sign(abs(d))==0) return {};
    P3 q=p1+v*dot(dir2,p2-p1)/d; return {q,q+e};
}
// 3D Covex Hull Template
db getV(P3 k1,P3 k2,P3 k3,P3 k4){ // get the Volume
    return dot(cross(k2-k1,k3-k1),k4-k1);
}
db rand_db(){return 1.0*rand()/RAND_MAX;}
VP convexHull2D(VP A,P3 dir){
    P3 x={(db)rand(),(db)rand(),(db)rand()}; x=x.unit();
    x=cross(x,dir).unit(); P3 y=cross(x,dir).unit();
    P3 vec=dir.unit()*dot(A[0],dir);
    vector<point>B;
    for (int i=0;i<A.size();i++) B.push_back((point)
{dot(A[i],x),dot(A[i],y)});
    B=ConvexHull(B); A.clear();
    for (int i=0;i<B.size();i++) A.push_back(x*B[i].x+y*B[i].y+vec);
    return A;
}
namespace CH3{
    VVP ret; set<pair<int,int> >e;
    int n; VP p,q;
    void wrap(int a,int b){
        if (e.find({a,b})==e.end()){
```

```
            int c=-1;
            for (int i=0;i<n;i++) if (i!=a&&i!=b){
                if (c==-1||sign(getV(q[c],q[a],q[b],q[i]))>0) c=i;
            }
            if (c!=-1){
                ret.push_back({p[a],p[b],p[c]});
                e.insert({a,b}); e.insert({b,c}); e.insert({c,a});
                wrap(c,b); wrap(a,c);
            }
        }
    }
    VVP ConvexHull3D(VP _p){
        p=q=_p; n=p.size();
        ret.clear(); e.clear();
        for (auto &i:q) i=i+(P3){rand_db()*1e-4,rand_db()*1e-4,rand_db()*1e-4};
        for (int i=1;i<n;i++) if (q[i].x<q[0].x) swap(p[0],p[i]),swap(q[0],q[i]);
        for (int i=2;i<n;i++) if ((q[i].x-q[0].x)*(q[1].y-q[0].y)>(q[i].y-q[0].y)*(q[1].x-q[0].x)) swap(q[1],q[i]),swap(p[1],p[i]);
        wrap(0,1);
        return ret;
    }
}
VVP reduceCH(VVP A){
    VVP ret; map<P3,VP> M;
    for (VP nowF:A){
        P3 dir=cross(nowF[1]-nowF[0],nowF[2]-nowF[0]).unit();
        for (P3 k1:nowF) M[dir].pb(k1);
    }
    for (pair<P3,VP> nowF:M) ret.pb(convexHull2D(nowF.se,nowF.fi));
    return ret;
}
//  把一个面变成 ( 点 , 法向量 ) 的形式
pair<P3,P3> getF(VP F){
    return mp(F[0],cross(F[1]-F[0],F[2]-F[0]).unit());
}
// 3D Cut 保留 dot(dir,x-p)>=0 的部分
VVP ConvexCut3D(VVP A,P3 p,P3 dir){
    VVP ret; VP sec;
    for (VP nowF: A){
        int n=nowF.size(); VP ans; int dif=0;
        for (int i=0;i<n;i++){
            int d1=sign(dot(dir,nowF[i]-p));
            int d2=sign(dot(dir,nowF[(i+1)%n]-p));
            if (d1>=0) ans.pb(nowF[i]);
            if (d1*d2<0){
                P3 q=getFL(p,dir,nowF[i],nowF[(i+1)%n])[0];
                ans.push_back(q); sec.push_back(q);
            }
            if (d1==0) sec.push_back(nowF[i]); else dif=1;
            dif|=(sign(dot(dir,cross(nowF[(i+1)%n]-nowF[i],nowF[(i+1)%n]-nowF[i])))==-1);
        }
        if (ans.size()>0&&dif) ret.push_back(ans);
    }
    if (sec.size()>0) ret.push_back(convexHull2D(sec,dir));
    return ret;
```

```
}
db vol(VVP A){
    if (A.size()==0) return 0; P3 p=A[0][0]; db ans=0;
    for (VP nowF:A)
        for (int i=2;i<nowF.size();i++)
            ans+=abs(getV(p,nowF[0],nowF[i-1],nowF[i]));
    return ans/6;
}
VVP init(db INF) {
    VVP pss(6,VP(4));
    pss[0][0] = pss[1][0] = pss[2][0] = {-INF, -INF, -INF};
    pss[0][3] = pss[1][1] = pss[5][2] = {-INF, -INF, INF};
    pss[0][1] = pss[2][3] = pss[4][2] = {-INF, INF, -INF};
    pss[0][2] = pss[5][3] = pss[4][1] = {-INF, INF, INF};
    pss[1][3] = pss[2][1] = pss[3][2] = {INF, -INF, -INF};
    pss[1][2] = pss[5][1] = pss[3][3] = {INF, -INF, INF};
    pss[2][2] = pss[4][3] = pss[3][1] = {INF, INF, -INF};
    pss[5][0] = pss[4][0] = pss[3][0] = {INF, INF, INF};
    return pss;
}
```

KM

```
#include<bits/stdc++.h>

using namespace std;

const int inf = 0x3f3f3f3f;
const int maxN = 505;

namespace KM {
    int mp[maxN][maxN], link_x[maxN], link_y[maxN], N;
    bool visx[maxN], visy[maxN];
    int que[maxN << 1], top, fail, pre[maxN];
    int hx[maxN], hy[maxN], slk[maxN];

    inline int check(int i) {
        visx[i] = true;
        if (link_x[i]) {
            que[fail++] = link_x[i];
            return visy[link_x[i]] = true;
        }
        while (i) {
            link_x[i] = pre[i];
            swap(i, link_y[pre[i]]);
        }
        return 0;
    }

    void bfs(int S) {
        for (int i = 1; i <= N; i++) {
            slk[i] = inf;
            visx[i] = visy[i] = false;
```

```
        }
        top = 0;
        fail = 1;
        que[0] = S;
        visy[S] = true;
        while (true) {
            int d;
            while (top < fail) {
                for (int i = 1, j = que[top++]; i <= N; i++) {
                    if (!visx[i] && slk[i] >= (d = hx[i] + hy[j] - mp[i]
[j])) {
                        pre[i] = j;
                        if (d) slk[i] = d;
                        else if (!check(i)) return;
                    }
                }
            }
            d = inf;
            for (int i = 1; i <= N; i++) {
                if (!visx[i] && d > slk[i]) d = slk[i];
            }
            for (int i = 1; i <= N; i++) {
                if (visx[i]) hx[i] += d;
                else slk[i] -= d;
                if (visy[i]) hy[i] -= d;
            }
            for (int i = 1; i <= N; i++) {
                if (!visx[i] && !slk[i] && !check(i)) return;
            }
        }
    }

    void prework() {
        for (int i = 1; i <= N; i++) {
            link_x[i] = link_y[i] = 0;
            visy[i] = false;
        }
        for (int i = 1; i <= N; i++) {
            hx[i] = 0;
            for (int j = 1; j <= N; j++) {
                if (hx[i] < mp[i][j]) hx[i] = mp[i][j];
            }
        }
    }

    void init(int n) {
        N = n;
        top = fail = 0;
        for (int i = 1; i <= N; i++) {
            link_x[i] = link_y[i] = visx[i] = visy[i] = pre[i] = hx[i] =
hy[i] = slk[i] = 0;
            for (int j = 1; j <= N; j++) {
                mp[i][j] = 0;
            }
        }
    }
}

```

```cpp
int main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    int n, m;
    cin >> n >> m;
    KM::init(max(n, m));
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            cin >> KM::mp[i][j];
        }
    }
    KM::prework();
    int ans = 0;
    for (int i = 1; i <= KM::N; i++) KM::bfs(i);
    for (int i = 1; i <= KM::N; i++) ans += KM::mp[i][KM::link_x[i]];

}
```

PN筛+2022HDU多校5-1002

相关证明可以看：https://oi-wiki.org/math/number-theory/powerful-number/

PN筛

积性函数：对于所有互质的a和b，总有$f(ab)=f(a)f(b)$，则称f为积性函数。

PN(power number)：对于正整数n，记n的质因数分解为$\prod p_i^{c_i}$。 是 PN 当且仅当 $\forall 1<=i<=m, c_i>1$。 （1也是PN）

性质1：所有 PN 都可以表示成a^2b^3的形式，因为大于等于2的数总能分解成$2x+3y$的形式。

性质2： n以内的 PN 至多有$O(\sqrt{n})$个，可以通过dfs枚举下一个质数的次数在$O(\sqrt{n})$时间内找到这些PN。

已知f为积性函数，求$F(n)=\sum_{i=1}^{n} f(i)$

通过PN筛求解的一般过程：

1.构造积性函数g满足$g(p)=f(p)$，且$G(n)=\sum_{i=1}^{n} g(i)$能够快速求出（或者快速预处理$2\sqrt{n}$个有效值）

2.构造（其实不用真的构造）h满足$g*h=f$，且$h(p^c)$能够快速求出，由$g*h=f$可得h也是积性函数

3.其实h满足$h(1)=1, h(p)=0, f(p^c)=\sum_{i=0}^{c} g(p^i)h(p^{c-i})$即可，也可以移项递推
$h(p^c)=f(p^c)-\sum_{i=1}^{c} g(p^i)h(p^{c-i})$

3.5.如果你在程序里递推$h(p^c)$复杂度会是$O(\sqrt{n}log(n))$的，但是实测跑不满，耗时较少。

4.dfs求出小于等于n的PN的同时计算$h(PN)$和答案，$F(n)=\sum_{i=1}^{n}[i是PN]h(i)G(\lfloor\frac{n}{i}\rfloor)$

复杂度与计算h和G的复杂度有关（有时需要预处理），两者均为$O(1)$时，总复杂度为$O(\sqrt{n})$

2022HDU多校5-1002

$f(p^c)=\frac{p^c}{c}$，求$\frac{1}{n}\sum_{i=1}^{n}f(i)$,$n<=1e12$

构造积性函数$g(p^c)=p^c$,即$g(x)=x, G(x)=\frac{x(x+1)}{2}$，并求出$h$

这里先递推找规律：

c	$f(p^c)=\frac{p^c}{c}$	$g(p^c)=p^c$	$h(p^c)$
0	1	1	1
1	p	p	0
2	$\frac{p^2}{2}$	p^2	$-\frac{p^2}{2}$
3	$\frac{p^3}{3}$	p^3	$-\frac{p^3}{6}$
4	$\frac{p^4}{4}$	p^4	$-\frac{p^4}{12}$
5	$\frac{p^5}{5}$	p^5	$-\frac{p^5}{20}$

容易发现$c>=2$时$h(p^c)=-\frac{p^c}{c(c-1)}$

也可以直接推式子：

$f(p^c)=\sum_{i=0}^{c}g(p^i)h(p^{c-i})$

$\frac{p^c}{c}=\sum_{i=0}^{c}p^ih(p^{c-i})$

$\frac{1}{c}=\sum_{i=0}^{c}\frac{h(p^{c-i})}{p^{c-i}}=\sum_{i=0}^{c}\frac{h(p^i)}{p^i}$

$c>=2$时：

$\frac{1}{c}-\frac{1}{c-1}=\sum_{i=0}^{c}\frac{h(p^i)}{p^i}-\sum_{i=0}^{c-1}\frac{h(p^i)}{p^i}$

$\frac{h(p^c)}{p^c}=\frac{1}{c}-\frac{1}{c-1}=-\frac{1}{c(c-1)}$

$h(p^c)=-\frac{p^c}{c(c-1)}$

预处理逆元（或者记忆化h函数后）可以$O(1)$求出。

dfs出n以内的所有PN，在dfs过程中，对于每个PN——x,将$h(x)G(\lfloor\frac{n}{x}\rfloor)$累加到答案上即可。

实际代码时要注意的点：

1.质数需要处理到$\sqrt{n}$，可以多处理一点例如处理到$100+\sqrt{maxn}$

2.h函数只会用到PN数处的值，预处理/记忆化时，只需要存$c>=2$的$h(p^c)$，$h(p_1^{c_1}p_2^{c_2}p_3^{c_3})$之类的可以直接在dfs中通过积性函数性质$O(1)$运算。

3.满足$c>=2$且$p^c<=n$的p^c都是PN，所以其数量是不超过$O(\sqrt{n})$的。

4.$h(p^c)$可以按质数的下标（即第几个质数） 和c存来省空间。

5.对于同一个n，G的有效取值会有$2\sqrt{n}$个，即$G(1)$、$G(2)\dots G(\sqrt{n})$和$G(\frac{n}{1})$、$G(\frac{n}{2})\dots G(\frac{n}{\sqrt{n}})$，有时需要预处理。

6.这题部分乘法要先转__int128再乘再取模。

PN筛

```

///2022杭电多校5-1002
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<int,int> pii;

const int inf=0x3f3f3f3f,N=1e7+9;
const ll mod=4179340454199820289;

const int PMAX=N,PN=N;//PN开到n以内P的最大数量可以省空间
int prime[PN],pcnt;//prime[0]=1,prime[1]=2
bool notp[PMAX];//motp[1]=1,notp[2]=0,notp[4]=1
void Prime(){
    pcnt=0;
    for(int i=2;i<PMAX;i++){
        if(!notp[i])prime[++pcnt]=i;
        for(int j=1;j<=pcnt&&i*prime[j]<PMAX;j++){
            notp[i*prime[j]]=1;
            if(i%prime[j]==0)break;
        }
    }
    notp[1]=1;
}

ll qpow(ll a,ll b){
    ll ans=1;
    while(b){
        if(b&1)ans=(__int128)ans*a%mod;
        a=(__int128)a*a%mod;
        b>>=1;
    }
    return ans;
}

struct PNS{//修改G，f，g即可使用，不同的n只需要初始化ans和n。
    ll ans,n;
    ll G(ll x){
        return (__int128)x*(x+1)/2%mod;
    }
    //f和g均只在h函数中使用，如果可以直接公式求h函数则不用定义这两个函数
    ll f(int pid,ll c){
        return (__int128)qpow(prime[pid],c)*qpow(c,mod-2)%mod;
    }
    //g函数不用记忆化，实际调用次数很少
    ll g(int pid,ll c){
        return qpow(prime[pid],c);
    }

    vector<ll>vh[PN];//这里要确保c>2时调用h(pid,c)前调用过h(pid,c-1)
    ll h(int pid,ll c){
        if(c==0)return 1;
        if(c==1)return 0;
```

```
        if(c-2>=(ll)vh[pid].size()){//n=1e12、1e13、1e14时会进80070、230567、670121次，跑不满根号n，所以递推的耗时也是不高的。
            //vh[pid].push_back((ll)((-(__int128)qpow(prime[pid],c)*qpow(c*(c-1),mod-2)%mod+mod)%mod));
            //递推h函数，需要配合f函数和g函数一起使用
            ll ans=f(pid,c);
            for(ll i=1;i<=c;i++){
                ans=((ans-(__int128)g(pid,i)*h(pid,c-i))%mod+mod)%mod;
            }
            vh[pid].push_back(ans);
        }
        return vh[pid][c-2];
    }
    void dfs(ll prod,ll hprod,int pid){
        ans=(ans+(__int128)hprod*G(n/prod))%mod;
        for(int i=pid;i<=pcnt;i++){
            if(prod>n/prime[i]/prime[i])break;
            for(ll c=2,x=prod*prime[i]*prime[i];x<=n;c++,x*=prime[i]){
                dfs(x,(__int128)hprod*h(i,c)%mod,i+1);
                if(x>n/prime[i])break;
            }
        }
    }
}pns;

int main(){
    #ifdef ONLINE_JUDGE
        //std::ios::sync_with_stdio(false);
    #else
        freopen("1002.in","r",stdin);
        //freopen("out.txt","w",stdout);
    #endif
    Prime();
    int t;
    scanf("%d",&t);
    while(t--){
        pns.ans=0;
        scanf("%lld",&pns.n);
        pns.dfs(1,1,1);
        //printf("%lld\n",pns.ans);
        printf("%lld\n",(ll)((__int128)pns.ans*qpow(pns.n,mod-2)%mod));
    }
    return 0;
}
```

PN筛_zyx

```
#include <bits/stdc++.h>

using namespace std;
#define de(x) cout << #x << " = " << x << endl
#define dd(x) cout << #x << " = " << x << " "
typedef long long ll;
```

```cpp
const __int128 N = 1e6 + 10;
const __int128 M = 41;
const __int128 mod = 417934045419982028911;
const __int128 inv2 = 208967022709991014511;

__int128 isp[N], pri[N], pcnt;
vector<__int128> h[N];

void getPrime() {
    fill(isp + 2, isp + N, 1);
    for (__int128 i = 2; i < N; i++) {
        if (isp[i]) {
            pri[pcnt++] = i;
        }
        for (__int128 j = 0; j < pcnt && i * pri[j] < N; j++) {
            isp[i * pri[j]] = 0;
            if (i % pri[j] == 0) break;
        }
    }
}

__int128 qpow(__int128 x, __int128 y) {
    __int128 ans = 1;
    while (y) {
        if (y & 1) ans = ans * x % mod;
        x = x * x % mod;
        y >>= 1;
    }
    return ans;
}

__int128 g(__int128 x) {
    return x;
}

__int128 G(__int128 x) {
    return x * (x + 1) % mod * inv2 % mod;
}

__int128 f(__int128 x, __int128 c) {
    return x * qpow(c, mod - 2) % mod;
}

__int128 ans;
ll n;

void dfs(__int128 deep, __int128 hpn, __int128 pn, bool flag) {
    if (flag) {
        ans = (ans + hpn * G(n / pn)) % mod;
    }
    if (deep >= pcnt) return;
    if (pri[deep] * pri[deep] * pn > n) return;
    dfs(deep + 1, hpn, pn, false);
    for (__int128 i = 2, pi = pri[deep] * pri[deep] % mod; pn * pi <= n;
i++, pi = pi * pri[deep] % mod) {
        dfs(deep + 1, hpn * h[deep][i] % mod, pn * pi, true);
    }
}
```

```cpp

signed main() {
    ios::sync_with_stdio(0);
    getPrime();
    for (__int128 pid = 0; pid < pcnt; pid++) {
        h[pid].push_back(1);
        h[pid].push_back(0);
        __int128 invp = qpow(pri[pid], mod - 2);
        for (__int128 c = 2, pc = pri[pid] * pri[pid]; c < M && pc <= 1e12; c++, pc = pc * pri[pid]) {
            h[pid].push_back(f(pri[pid], c));//唯一f使用，传入参数类型自定义
            __int128 pci = qpow(pri[pid], c);
            for (__int128 i = 1, pi = pri[pid]; i <= c; i++, pi = pi * pri[pid] % mod) {
                pci = pci * invp % mod;
                h[pid][c] = (h[pid][c] - g(pi) * h[pid][c - i] % mod + mod) % mod;
            }
        }
    }

    ll t;
    cin >> t;
    while (t--) {
        cin >> n;
        ans = G(n);
        dfs(0, 1, 1, false);
        cout << (ll) ans << endl;
    }
}


```

卡特兰

‘卡特兰数1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012,...

$C_n = \frac{1}{n+1}C_{2n}^n = C_{2n}^n - C_{2n}^{n-1}$

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

~~C{n}=\frac{4n-2}{n+1}C{n-1}(C_0=1)~~

$C_{n+1} = \sum_{i=0}^{n} C_i C_{n-i} (C_0 = 1)$

超级卡特兰数1, 1, 3, 11, 45, 197, 903, 4279, 20793, 103049,...（从第0项开始）

$F_n * (n+1) = (6 * n - 3) * F_{n-1} - (n-2) * F_{n-2}$

大施罗德数(OEIS A006318)1, 2, 6, 22, 90, 394, 1806, 8558, 41586, 206098,...

超级卡特兰数的两倍（除第一项）

回文自动机

```cpp

#include <bits/stdc++.h>

using namespace std;
const int maxn = 300000 + 5;

namespace pam {
    int sz, tot, last;
    int cnt[maxn], ch[maxn][26], len[maxn], fail[maxn];
    char s[maxn];

    int node(int l) {  // 建立一个新节点，长度为 l
        sz++;
        memset(ch[sz], 0, sizeof(ch[sz]));
        len[sz] = l;
        fail[sz] = cnt[sz] = 0;
        return sz;
    }

    void clear() {  // 初始化
        sz = -1;
        last = 0;
        s[tot = 0] = '$';
        node(0);
        node(-1);
        fail[0] = 1;
    }

    int getfail(int x) {  // 找后缀回文
        while (s[tot - len[x] - 1] != s[tot]) x = fail[x];
        return x;
    }

    void insert(char c) {  // 建树
        s[++tot] = c;
        int now = getfail(last);
        if (!ch[now][c - 'a']) {
            int x = node(len[now] + 2);
            fail[x] = ch[getfail(fail[now])][c - 'a'];
            ch[now][c - 'a'] = x;
        }
        last = ch[now][c - 'a'];
        cnt[last]++;
    }

    long long solve() {
        long long ans = 0;
        for (int i = sz; i >= 0; i--) {
            cnt[fail[i]] += cnt[i];
        }
        for (int i = 1; i <= sz; i++) {  // 更新答案
            ans = max(ans, 1ll * len[i] * cnt[i]);
        }
        return ans;
    }
}  // namespace pam

```

```
char s[maxn];

int main() {
    pam::clear();
    scanf("%s", s + 1);
    for (int i = 1; s[i]; i++) {
        pam::insert(s[i]);
    }
    printf("%lld\n", pam::solve());
    return 0;
}


```

多项式全家桶

```


//#pragma GCC optimize(2)
#include <bits/stdc++.h>

using namespace std;
typedef long long ll;

const ll N = 3000007;
const ll p = 998244353, gg = 3, ig = 332738118, img = 86583718;
const ll mod = 998244353;

ll qpow(ll a, ll b) {
    ll res = 1;
    while (b) {
        if (b & 1) res = 1ll * res * a % mod;
        a = 1ll * a * a % mod;
        b >>= 1;
    }
    return res;
}

namespace Poly {
#define mul(x, y) (1ll * x * y >= mod ? 1ll * x * y % mod : 1ll * x * y)
#define minus(x, y) (1ll * x - y < 0 ? 1ll * x - y + mod : 1ll * x - y)
#define plus(x, y) (1ll * x + y >= mod ? 1ll * x + y - mod : 1ll * x + y)
#define ck(x) (x >= mod ? x - mod : x)//取模运算太慢了

    typedef vector<ll> poly;
    const ll G = 3;//根据具体的模数而定，原根可不一定不一样！！！
    //一般模数的原根为 2 3 5 7 10 6
    const ll inv_G = qpow(G, mod - 2);
    ll RR[N], inv[N];

    void init() {
        inv[0] = inv[1] = 1;
        for (ll i = 2; i < N; ++i)
            inv[i] = 1ll * inv[mod % i] * (mod - mod / i) % mod;
    }

    ll NTT_init(ll n) {//快速数论变换预处理
```

```
        ll limit = 1, L = 0;
        while (limit <= n) limit <<= 1, L++;
        for (ll i = 0; i < limit; ++i)
            RR[i] = (RR[i >> 1] >> 1) | ((i & 1) << (L - 1));
        return limit;
    }

//  省空间用
    ll deer[2][N];

    void NTT(poly &A, ll type, ll limit) {//快速数论变换
        A.resize(limit);
        for (ll i = 0; i < limit; ++i)
            if (i < RR[i])
                swap(A[i], A[RR[i]]);
        for (ll mid = 2, j = 1; mid <= limit; mid <<= 1, ++j) {
            ll len = mid >> 1;

//          省空间用
            ll buf1 = qpow(G, (mod - 1) / (1 << j));
            ll buf0 = qpow(inv_G, (mod - 1) / (1 << j));
            deer[0][0] = deer[1][0] = 1;
            for (ll i = 1; i < (1 << j); ++i) {
                deer[0][i] = 1ll * deer[0][i - 1] * buf0 % mod;//逆
                deer[1][i] = 1ll * deer[1][i - 1] * buf1 % mod;
            }

            for (ll pos = 0; pos < limit; pos += mid) {
//                ll *wn = deer[type][j];
//              省空间用
                ll *wn = deer[type];
                for (ll i = pos; i < pos + len; ++i, ++wn) {
                    ll tmp = 1ll * (*wn) * A[i + len] % mod;
                    A[i + len] = ck(A[i] - tmp + mod);
                    A[i] = ck(A[i] + tmp);
                }
            }
        }
        if (type == 0) {
            for (ll i = 0; i < limit; ++i)
                A[i] = 1ll * A[i] * inv[limit] % mod;
        }
    }

    poly poly_mul(poly A, poly B) {//多项式乘法
        ll deg = A.size() + B.size() - 1;
        ll limit = NTT_init(deg);
        poly C(limit);
        NTT(A, 1, limit);
        NTT(B, 1, limit);
        for (ll i = 0; i < limit; ++i)
            C[i] = 1ll * A[i] * B[i] % mod;
        NTT(C, 0, limit);
        C.resize(deg);
        return C;
    }

    poly poly_inv(poly &f, ll deg) {//多项式求逆
```

```
        if (deg == 1)
            return poly(1, qpow(f[0], mod - 2));

        poly A(f.begin(), f.begin() + deg);
        poly B = poly_inv(f, (deg + 1) >> 1);
        ll limit = NTT_init(deg << 1);
        NTT(A, 1, limit), NTT(B, 1, limit);
        for (ll i = 0; i < limit; ++i)
            A[i] = B[i] * (2 - 1ll * A[i] * B[i] % mod + mod) % mod;
        NTT(A, 0, limit);
        A.resize(deg);
        return A;
    }

    poly poly_dev(poly f) {//多项式求导
        ll n = f.size();
        for (ll i = 1; i < n; ++i) f[i - 1] = 1ll * f[i] * i % mod;
        return f.resize(n - 1), f;//f[0] = 0，这里直接扔了,从1开始
    }

    poly poly_idev(poly f) {//多项式求积分
        ll n = f.size();
        for (ll i = n - 1; i; --i) f[i] = 1ll * f[i - 1] * inv[i] % mod;
        return f[0] = 0, f;
    }

    poly poly_ln(poly f, ll deg) {//多项式求对数
        poly A = poly_idev(poly_mul(poly_dev(f), poly_inv(f, deg)));
        return A.resize(deg), A;
    }

    poly poly_exp(poly &f, ll deg) {//多项式求指数
        if (deg == 1)
            return poly(1, 1);

        poly B = poly_exp(f, (deg + 1) >> 1);
        B.resize(deg);
        poly lnB = poly_ln(B, deg);
        for (ll i = 0; i < deg; ++i)
            lnB[i] = ck(f[i] - lnB[i] + mod);

        ll limit = NTT_init(deg << 1);//n -> n^2
        NTT(B, 1, limit), NTT(lnB, 1, limit);
        for (ll i = 0; i < limit; ++i)
            B[i] = 1ll * B[i] * (1 + lnB[i]) % mod;
        NTT(B, 0, limit);
        B.resize(deg);
        return B;
    }

    poly poly_sqrt(poly &f, ll deg) {//多项式开方
        if (deg == 1) return poly(1, 1);
        poly A(f.begin(), f.begin() + deg);
        poly B = poly_sqrt(f, (deg + 1) >> 1);
        poly IB = poly_inv(B, deg);
        ll limit = NTT_init(deg << 1);
        NTT(A, 1, limit), NTT(IB, 1, limit);
        for (ll i = 0; i < limit; ++i)
```

```
            A[i] = 1ll * A[i] * IB[i] % mod;
        NTT(A, 0, limit);
        for (ll i = 0; i < deg; ++i)
            A[i] = 1ll * (A[i] + B[i]) * inv[2] % mod;
        A.resize(deg);
        return A;
    }

    poly poly_pow(poly f, ll k) {//多项式快速幂
        if (f.size() == 1) {
            f[0] = qpow(f[0], k);
            return f;
        }
        f = poly_ln(f, f.size());
        for (auto &x: f) x = 1ll * x * k % mod;
        return poly_exp(f, f.size());
    }

    poly poly_cos(poly f, ll deg) {//多项式三角函数（cos）
        poly A(f.begin(), f.begin() + deg);
        poly B(deg), C(deg);
        for (ll i = 0; i < deg; ++i)
            A[i] = 1ll * A[i] * img % mod;

        B = poly_exp(A, deg);
        C = poly_inv(B, deg);
        ll inv2 = qpow(2, mod - 2);
        for (ll i = 0; i < deg; ++i)
            A[i] = 1ll * (1ll * B[i] + C[i]) % mod * inv2 % mod;
        return A;
    }

    poly poly_sin(poly f, ll deg) {//多项式三角函数（sin）
        poly A(f.begin(), f.begin() + deg);
        poly B(deg), C(deg);
        for (ll i = 0; i < deg; ++i)
            A[i] = 1ll * A[i] * img % mod;

        B = poly_exp(A, deg);
        C = poly_inv(B, deg);
        ll inv2i = qpow(img << 1, mod - 2);
        for (ll i = 0; i < deg; ++i)
            A[i] = 1ll * (1ll * B[i] - C[i] + mod) % mod * inv2i % mod;
        return A;
    }

    poly poly_arcsin(poly f, ll deg) {
        poly A(f.size()), B(f.size()), C(f.size());
        A = poly_dev(f);
        B = poly_mul(f, f);
        for (ll i = 0; i < deg; ++i)
            B[i] = minus(mod, B[i]);
        B[0] = plus(B[0], 1);
        C = poly_sqrt(B, deg);
        C = poly_inv(C, deg);
        C = poly_mul(A, C);
        C = poly_idev(C);
        return C;
```

```
    }

    poly poly_arctan(poly f, ll deg) {
        poly A(f.size()), B(f.size()), C(f.size());
        A = poly_dev(f);
        B = poly_mul(f, f);
        B[0] = plus(B[0], 1);
        C = poly_inv(B, deg);
        C = poly_mul(A, C);
        C = poly_idev(C);
        return C;
    }
}

using namespace Poly;

signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    ll n, k;
    cin >> n >> k;
    Poly::init();
    vector<ll> v(n);
    for (ll i = 0; i < n; i++) cin >> v[i];
    auto res = Poly::poly_pow(v, k);
    for (auto x: res) cout << x << ' ';
    cout << endl;
}


```

大质数表

https://www.cnblogs.com/ljxtt/p/13514346.html

1e17	1e18
100000000000000003	1000000000000000003
100000000000000013	1000000000000000009
100000000000000019	1000000000000000031
100000000000000021	1000000000000000079
100000000000000049	1000000000000000177
100000000000000081	1000000000000000183
100000000000000099	1000000000000000201
100000000000000141	1000000000000000283
100000000000000181	1000000000000000381
100000000000000337	1000000000000000387
100000000000000339	1000000000000000507
100000000000000369	1000000000000000523
100000000000000379	1000000000000000583
100000000000000423	1000000000000000603
100000000000000519	1000000000000000619
100000000000000543	1000000000000000621
100000000000000589	1000000000000000799
100000000000000591	1000000000000000841
100000000000000609	1000000000000000861
100000000000000669	1000000000000000877
100000000000000691	1000000000000000913
100000000000000781	1000000000000000931
100000000000000787	1000000000000000997

正多面体

4 6 8 12 20

点双连通分量+缩点建图

```
1
2 #include <bits/stdc++.h>
3
```

```cpp
#define ll long long
using namespace std;
const int N = 10010;
const int M = 10010 * 4;
int head[N];
int ver[M];
int Next[M];
int tot, n, m;

void add(int x, int y) {
    ver[++tot] = y;
    Next[tot] = head[x];
    head[x] = tot;
}

int root;
vector<int> dcc[N];
int stackk[N];
int dfn[N], low[N];
int num = 0;//ʰ���
int top;//stackk
int cnt = 0;//��ᶜ����Ŀ
bool cut[N];//����ж
void tarjan(int x) {
    dfn[x] = low[x] = ++num;
    stackk[++top] = x;
    if (x == root && head[x] == 0) {
        dcc[++cnt].push_back(x);//cnt��ᶜ����
        return;
    }
    int flag = 0;
    for (int i = head[x]; i; i = Next[i]) {
        int y = ver[i];
        if (!dfn[y]) {
            tarjan(y);
            low[x] = min(low[x], low[y]);
            if (low[y] >= dfn[x]) {
                flag++;
                if (x != root || flag > 1)cut[x] = true;
                cnt++;
                int z;
                do//������Ѽ����xh�□
                    {
                        z = stackk[top--];
                        dcc[cnt].push_back(z);
                    } while (z != y);
                dcc[cnt].push_back(x);
            }
        } else low[x] = min(low[x], dfn[y]);
    }
}

int tot2 = 1;
int new_id[N];

int hc[N];
int vc[M];
```

```
int nc[M];

void add_c(int x, int y) {
    vc[++tot2] = y;
    nc[tot2] = hc[x];
    hc[x] = tot2;
}

int main() {
    while (cin >> n >> m) {
        tot = 1;//???????^???????ʸ??'??ɟ?
        for (int i = 1; i <= m; ++i) {
            int x, y;
            cin >> x >> y;
            if (x == y)continue;
            add(x, y);
            add(y, x);
        }
        for (int i = 1; i <= n; ++i) {
            if (!dfn[i])root = i, tarjan(i);
        }
        /*for(int i=1;i<=n;++i)
            if(cut[i])printf("%d ",i);*/
        //??????????ʰ??V-DCC
        //????????ÿ?????ᶜ???eĵ?
        for (int i = 1; i <= cnt; ++i) {
            for (int j = 0; j < dcc[i].size(); ++j)cout << i << " " << dcc[i][j] << endl;
        }

        //????
        tot2 = 1;
        int num2 = cnt;
        for (int i = 1; i <= n; ++i) {
            if (cut[i])new_id[i] = ++num2;//??????????±??,?ℚ??ÿ?????濫?????Ih?????ᶜ??
        }
        for (int i = 1; i <= cnt; ++i) {
            for (int j = 0; j < dcc[i].size(); ++j) {
                int x = dcc[i][j];
                if (cut[x])//h????ᶜ??????????ᶜh????□ᶜ???????ǰ???Ш??ᶜ??????????;
                {
                    add_c(i, new_id[x]);
                    add_c(new_id[x], i);
                } else new_id[x] = i;//□□□□□□□□□□□h????ᶜ??
            }
        }

        //??????????c?и???°????¿Ы?є???ט???^???ŵ?'?ã?i??2??'??ÿ?µ?2??<tot2????<=??
        for (int i = 2; i < tot2; i += 2)
            cout << vc[i ^ 1] << "   " << vc[i] << endl;


    }
    return 0;
```

```
}

/*
 * tot2Ï��������2��’����
 * num2Ï������̃j���
 * ��ㅏ���徨������ㅜ��ㅜ뇌N��C
*/


```