# Notes for ACMICPC World Finals 2013

# ACMICPC World Finals 2013 参考资料

Chinese Edition 中文版

*Shanghai Jiao Tong University* : **Mithril**


***Coach*** **教　练**

Yong YU　　俞　勇

***Team member*** **队　员**

Jingbo SHANG　　商静波

Bin JIN　　金斌

Xiaoxu GUO　　郭晓旭

目录

**计算几何**

```
line point_make_line(point a, point b){ //=====两点求线
     line h; h.a=b.y-a.y; h.b=-(b.x-a.x); h.c=-a.x*b.y + a.y*b.x;
     return h;
}
//==========线段平移 D 的长度
line move_d(line a,const double d){
  return (line){a.a,a.b,a.c+d*sqrt(a.a*a.a+a.b*a.b)};
}
int PointInPolygon(point cp, point a[], int n){
      int i , k , d1 , d2 ,wn=0;  a[n]=a[0];
      rep(i,n){
          if ( PointOnSegment ( cp,a[i],a[i+1] ) ) return 2 ;
          k=cmp(area(a[i],a[i+1],cp));
          d1=cmp(a[i+0].y-cp.y); d2=cmp(a[i+1].y-cp.y);
          if (k>0 && d1<=0 && d2>0) wn++;
          if (k<0 && d2<=0 && d1>0) wn--;
}   return wn!=0;}
void CircleCenter(point  p0 , point p1 , point p2 , point &cp ){
     double a1=p1.x-p0.x , b1=p1.y-p0.y , c1=(sqr(a1)+sqr(b1)) / 2 ;
     double a2=p2.x-p0.x , b2=p2.y-p0.y , c2=(sqr(a2)+sqr(b2)) / 2 ;
     double d = a1*b2 - a2*b1 ;
     cp.x = p0.x + ( c1*b2 - c2*b1 ) / d ;
     cp.y = p0.y + ( a1*c2 - a2*c1 ) / d ;
}//  三角形内心
double Incenter(point A, point B, point C, point &cp ){
    double s , p , r , a , b , c ;
    a = dis(B, C) , b = dis(C, A) , c = dis(A, B) ; p = (a +b +c) / 2 ;
    s = sqrt ( p * ( p-a ) * ( p-b ) * ( p-c ) ) ; r = s / p ;
    cp.x = ( a*A.x + b*B. x + c*C.x ) / ( a + b + c ) ;
    cp.y = ( a*A.y + b*B. y + c*C.y ) / ( a + b + c ) ;
    return r ;
}// 三角形  垂心
void Orthocenter(point A, point B, point C, point &cp ){
    CircleCenter(A, B, C, cp );
    cp.x = A.x + B.x + C.x - 2 * cp.x ;cp.y = A.y + B.y + C.y - 2 * cp.y ;}
//   园外一点p0 ,半径为r， 直线ax+by+c=0 的交点
int CircleLine(point p0 , double r , double a , double b , double c , point
&cp1 , point &cp2 ) {
    double aa = a*a , bb = b*b , s = aa + bb ;
    double d = r*r*s - sqr ( a*p0.x+b*p0.y+c ) ;
    if (d+eps<0) return 0 ;  if (d<eps) d=0; else d=sqrt(d);
    double ab = a*b , bd = b*d , ad = a*d ;
    double xx = bb*p0.x - ab*p0.y - a*c ;
    double yy = aa*p0.y - ab*p0.x - b*c ;
    cp2.x = ( xx + bd ) / s ; cp2.y = ( yy - ad ) / s ;
    cp1.x = ( xx - bd ) / s ; cp1.y = ( yy + ad ) / s ;
    if( d>eps ) return 2 ; else return 1 ;
}// 两园交线|P - P1| = r1 and |P - P2| = r2 of the ax + by + c = 0 form
void CommonAxis (point p1 , double r1 , point p2 , double r2 , double &a , double
&b , double &c ){
    double sx = p2.x + p1.x , mx = p2.x - p1.x ;
    double sy = p2.y + p1.y , my = p2.y - p1.y ;
    a = 2*mx ; b = 2*my ; c = -sx*mx - sy*my - ( r1+r2 )*( r1-r2 ) ;
}//   两园交点  两个圆不能共圆心，请特判
int CircleCrossCircle( point p1 , double r1 , point p2 , double r2 , point &cp1 ,
point &cp2 ){
    double mx = p2.x - p1.x , sx = p2.x+p1.x , mx2 = mx*mx;
    double my = p2.y - p1.y , sy = p2.y+p1.y , my2 = my*my;
    double sq = mx2 + my2 , d = -( sq - sqr ( r1-r2 ) ) * ( sq - sqr ( r1+r2 ) ) ;
    if ( d+eps <0 ) return 0 ; if ( d<eps ) d=0 ;  else d = sqrt(d ) ;
    double x = mx* ( ( r1+r2 )*( r1-r2 ) + mx*sx ) + sx*my2 ;
    double y = my* ( ( r1+r2 )*( r1-r2 ) + my*sy ) + sy*mx2 ;
    double dx = mx*d , dy = my*d ; sq *= 2;
```

```
    cp1.x = ( x - dy ) / sq ; cp1.y = ( y + dx ) / sq ;
    cp2.x = ( x + dy ) / sq ; cp2.y = ( y - dx ) / sq ;
    if ( d>eps ) return 2 ; else return 1 ;
}//====两园面积交   dist = 是距离 ， dis是平方
double twoCircleAreaUnion(point a, point b , double r1, double r2){
    if (r1+r2<=(a-b).dist()) return 0;
    if (r1+(a-b).dist()<=r2) return pi*r1*r1;
    if (r2+(a-b).dist()<=r1) return pi*r2*r2;
    double c1,c2 , ans=0;
    c1=(r1*r1-r2*r2+(a-b).dis())/(a-b).dist()/r1/2.0;
    c2=(r2*r2-r1*r1+(a-b).dis())/(a-b).dist()/r2/2.0;
    double s1,s2;    s1=acos(c1); s2=acos(c2);
    ans+=s1*r1*r1-r1*r1*sin(s1)*cos(s1);
    ans+=s2*r2*r2-r2*r2*sin(s2)*cos(s2);
    return ans;
}//=====多边形和圆相交的面积用有向面积，划分成一个三角形和圆的面积的交
double area2(point pa, point pb) {
    if (pa.len() < pb.len()) swap(pa, pb);if (pb.len() < eps) return 0;
    double a, b, c, B, C, sinB, cosB, sinC, cosC, S, h, theta;
    a = pb.len();    b = pa.len();    c = (pb-pa).len();
    cosB=dot(pb,pb-pa)/a/c;    sinB=fabs(det(pb,pb-pa)/a/c);
    cosC=dot(pa, pb) / a / b;  sinC=fabs(det(pa,pb)/a/b);
    B=atan2(sinB , cosB);   C=atan2(sinC, cosC);
    if (a > r) { S = C/2*r*r;     h = a*b*sinC/c;
        if (h < r && B < PI/2) S -= (acos(h/r)*r*r - h*sqrt(r*r-h*h));
    }
    else if (b > r) {    theta = PI - B - asin(sinB/r*a);
        S = .5*a*r*sin(theta) + (C-theta)/2*r*r;   }
    else S = .5*sinC*a*b; return S; }// a, b, c, r   fixed
double area(const point &o) {
    double S = 0;    point oa = a-o, ob = b-o, oc = c-o;
    S += area2(oa, ob) * sign(det(oa, ob));
    S += area2(ob, oc) * sign(det(ob, oc));
    S += area2(oc, oa) * sign(det(oc, oa));   return abs(S);
}//======半平面交
void rebuild(point a, point b){//逆时针 ,ab左侧
    int i,t;double k1,k2;sol[m]=sol[0]; t=0;
    foru(i,1,m){ k1=area(a,b,sol[i]); k2=area(a,b,sol[i-1]);
     if (cmp(k1)*cmp(k2)<0){
         tmp[t].x=(sol[i].x*k2-sol[i-1].x*k1) / (k2-k1);
         tmp[t].y=(sol[i].y*k2-sol[i-1].y*k1) / (k2-k1); t++;
      } if (cmp(area(a,b,sol[i])) >=0){   tmp[t]=sol[i]; t++;}}
    m=t;  rep(i,m) sol[i]=tmp[i];
}//====nlogn半平面交
bool check(const Plane &u, const Plane &v, const Plane &w) {
   return intersect(u, v).in(w);
}
void build(vector <Plane> planes) {
   int head = 0, tail = 0;
   for (int i = 0; i < (int)planes.size(); ++ i) {
      while (tail - head > 1 && !check(queue[tail - 2], queue[tail - 1],
planes[i])) {
         tail --;
      }
      while (tail - head > 1 && !check(queue[head + 1], queue[head], planes[i]))
{
         head ++;
      }
      queue[tail ++] = planes[i];
   }
   while (tail - head > 2 && !check(queue[tail - 2], queue[tail - 1],
queue[head])) {
      tail --;
   }
```

```
    while (tail - head > 2 && !check(queue[head + 1], queue[head], queue[tail
- 1])) {
        head ++;
    }
}
```

三维几何

```
//vlen(point3 P):length of vector; zero(double x):if fabs(x)<eps) return true;
double vlen(point3 p);
//平面法向量
point3 pvec(point3 s1,point3 s2,point3 s3){return det((s1-s2),(s2-s3));}
//check共线
int dots_inline(point3 p1,point3 p2,point3 p3){
    return vlen(det(p1-p2,p2-p3))<eps;}
//check共平面
int dots_onplane(point3 a,point3 b,point3 c,point3 d){
    return zero(dot(pvec(a,b,c),d-a));}
//check在线段上(end point inclusive)
int dot_online_in(point3 p,line3 l)
    int dot_online_in(point3 p,point3 l1,point3 l2){return
    zero(vlen(det(p-l1,p-l2)))&&(l1.x-p.x)*(l2.x-p.x)<eps&&(l1.y-p.y)*(l2.
    y-p.y)<eps&&(l1.z-p.z)*(l2.z-p.z)<eps;}
//check在线段上(end point exclusive)
int dot_online_ex(point3 p,line3 l)
int dot_online_ex(point3 p,point3 l1,point3 l2){ return
    dot_online_in(p,l1,l2)&&(!zero(p.x-l1.x)||!zero(p.y-l1.y)||!zero(p.z-
    l1.z))&&(!zero(p.x-l2.x)||!zero(p.y-l2.y)||!zero(p.z-l2.z));
}
//check一个点是否在三角形里(inclusive)
int dot_inplane_in(point3 p,plane3 s)
int dot_inplane_in(point3 p,point3 s1,point3 s2,point3 s3){
    return zero(vlen(det(s1-s2,s1-s3))-vlen(det(p-s1,p-s2))-
        vlen(det(p-s2,p-s3))-vlen(det(p-s3,p-s1)));
}
//check一个点是否在三角形里(exclusive)
int dot_inplane_ex(point3 p,plane3 s)
int dot_inplane_ex(point3 p,point3 s1,point3 s2,point3 s3){
    return dot_inplane_in(p,s1,s2,s3)&&vlen(det(p-s1,p-s2))>eps&&
        vlen(det(p-s2,p-s3))>eps&&vlen(det(p-s3,p-s1))>eps;
}
//check if two point and a segment in one plane have the same side
int same_side(point3 p1,point3 p2,point3 l1,point3 l2)
int same_side(point3 p1,point3 p2,line3 l){
    return dot(det(l.a-l.b,p1-l.b),det(l.a-l.b,p2-l.b))>eps;
}
//check if two point and a segment in one plane have the opposite side
int opposite_side(point3 p1,point3 p2,point3 l1,point3 l2)
int opposite_side(point3 p1,point3 p2,line3 l){
    return dot(det(l.a-l.b,p1-l.b), det(l.a-l.b,p2-l.b))<-eps;
}
//check if two point is on the same side of a plane
int same_side(point3 p1,point3 p2,point3 s1,point3 s2,point3 s3)
int same_side(point3 p1,point3 p2,plane3 s){
    return dot(pvec(s),p1-s.a)*dot(pvec(s),p2-s.a)>eps;
}
//check if two point is on the opposite side of a plane
int opposite_side(point3 p1,point3 p2,point3 s1,point3 s2,point3 s3)
int opposite_side(point3 p1,point3 p2,plane3 s){
    return dot(pvec(s),p1-s.a)*dot(pvec(s),p2-s.a)<-eps;
}
//check if two straight line is parallel
int parallel(point3 u1,point3 u2,point3 v1,point3 v2)
int parallel(line3 u,line3 v){    return vlen(det(u.a-u.b,v.a-v.b))<eps; }
//check if two plane is parallel
int parallel(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3)
```

```
int parallel(plane3 u,plane3 v){return vlen(det(pvec(u),pvec(v)))<eps;}
//check if a plane and a line is parallel
int parallel(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
int parallel(line3 l,plane3 s){ return zero(dot(l.a-l.b,pvec(s))); }
//check if two line is perpendicular
int perpendicular(point3 u1,point3 u2,point3 v1,point3 v2)
int perpendicular(line3 u,line3 v){return zero(dot(u.a-u.b,v.a-v.b)); }
//check if two plane is perpendicular
int perpendicular(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3)

int perpendicular(plane3 u,plane3 v){ return zero(dot(pvec(u),pvec(v))); }
//check if plane and line is perpendicular
int perpendicular(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
int perpendicular(line3 l,plane3 s){return vlen(det(l.a-l.b,pvec(s)))<eps;}
//check 两条线段是否有交点(end point inclusive)
int intersect_in(point3 u1,point3 u2,point3 v1,point3 v2)
int intersect_in(line3 u,line3 v){
    if (!dots_onplane(u.a,u.b,v.a,v.b)) return 0;
    if (!dots_inline(u.a,u.b,v.a)||!dots_inline(u.a,u.b,v.b))
        return !same_side(u.a,u.b,v)&&!same_side(v.a,v.b,u);
    return dot_online_in(u.a,v)||dot_online_in(u.b,v)||
            dot_online_in(v.a,u)||dot_online_in(v.b,u);
}
//check 两条线段是否有交点(end point exclusive)
int intersect_ex(point3 u1,point3 u2,point3 v1,point3 v2)
int intersect_ex(line3 u,line3 v){
    return dots_onplane(u.a,u.b,v.a,v.b)&&opposite_side(u.a,u.b,v)&&
            opposite_side(v.a,v.b,u);
}
//check线段和三角形是否有交点(end point and border inclusive)
int intersect_in(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
int intersect_in(line3 l,plane3 s){
    return !same_side(l.a,l.b,s)&&!same_side(s.a,s.b,l.a,l.b,s.c)&&
        !same_side(s.b,s.c,l.a,l.b,s.a)&&!same_side(s.c,s.a,l.a,l.b,s.b);
}
//check线段和三角形是否有交点(end point and border exclusive)
int intersect_ex(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
int intersect_ex(line3 l,plane3 s){
    return opposite_side(l.a,l.b,s)&&opposite_side(s.a,s.b,l.a,l.b,s.c)&&
    opposite_side(s.b,s.c,l.a,l.b,s.a)&&opposite_side(s.c,s.a,l.a,l.b,s.b)
;}
//calculate the intersection of two line
//Must you should ensure they are co-plane and not parallel
point3 intersection(point3 u1,point3 u2,point3 v1,point3 v2)
point3 intersection(line3 u,line3 v){
    point3 ret=u.a;
    double t=((u.a.x-v.a.x)*(v.a.y-v.b.y)-(u.a.y-v.a.y)*(v.a.x-v.b.x))
            /((u.a.x-u.b.x)*(v.a.y-v.b.y)-(u.a.y-u.b.y)*(v.a.x-v.b.x));
    ret+=(u.b-u.a)*t;    return ret;
}
//calculate the intersection of plane and line
point3 intersection(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
point3 intersection(line3 l,plane3 s){
    point3 ret=pvec(s);
double t=(ret.x*(s.a.x-l.a.x)+ret.y*(s.a.y-l.a.y)+ret.z*(s.a.z-l.a.z))/
        (ret.x*(l.b.x-l.a.x)+ret.y*(l.b.y-l.a.y)+ret.z*(l.b.z-l.a.z));
    ret=l.a + (l.b-l.a)*t;   return ret;
}
//calculate the intersection of two plane
bool intersection(plane3 pl1 , plane3 pl2 , line3 &li) {
    if (parallel(pl1,pl2)) return false;
    li.a=parallel(pl2.a,pl2.b, pl1) ? intersection(pl2.b,pl2.c,
pl1.a,pl1.b,pl1.c) : intersection(pl2.a,pl2.b, pl1.a,pl1.b,pl1.c);
    point3 fa; fa=det(pvec(pl1),pvec(pl2)); li.b=li.a+fa;  return true;
```

```
}
//distance from point to line
double ptoline(point3 p,point3 l1,point3 l2)
double ptoline(point3 p,line3 l){
    return vlen(det(p-l.a,l.b-l.a))/distance(l.a,l.b);}
//distance from point to plane
double ptoplane(point3 p,plane3 s){
    return fabs(dot(pvec(s),p-s.a))/vlen(pvec(s));}
double ptoplane(point3 p,point3 s1,point3 s2,point3 s3)
//distance between two line      当u,v平行时有问题
double linetoline(line3 u,line3 v){
    point3 n=det(u.a-u.b,v.a-v.b); return fabs(dot(u.a-v.a,n))/vlen(n);
}
double linetoline(point3 u1,point3 u2,point3 v1,point3 v2)
//cosine value of the angle formed by two lines
double angle_cos(line3 u,line3 v){
    return dot(u.a-u.b,v.a-v.b)/vlen(u.a-u.b)/vlen(v.a-v.b);
}
double angle_cos(point3 u1,point3 u2,point3 v1,point3 v2)
//cosine value of the angle formed by two planes
double angle_cos(plane3 u,plane3 v){
    return dot(pvec(u),pvec(v))/vlen(pvec(u))/vlen(pvec(v));}
double angle_cos(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3)
//cosine value of the angle formed by plane and line
double angle_sin(line3 l,plane3 s){
    return dot(l.a-l.b,pvec(s))/vlen(l.a-l.b)/vlen(pvec(s));}
double angle_sin(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
```

三维几何操作合并

```
const double pi = acos(-1.0); double a[4][4];
int dcmp(const double &a, const double &b = 0, const double & zero = 1e-6){
    if (a - b < -zero)   return -1;   return a - b > zero;}
void multi(const double a[4][4],const double b[4][4],double c[4][4]){
    for(int i=0;i<4;i++)for(int j=0;j<4;j++){
    c[i][j]=a[i][0]*b[0][j]; for(int k=1;k<4;k++) c[i][j]+=a[i][k]*b[k][j];
}}
void multi(double a[4][4],const double b[4][4]){
    static double c[4][4];   multi(a,b,c);memcpy(a,c,sizeof(a[0][0])*16);
}
void Macro(){
    double b[4][4]={1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1};
    memcpy(a,b,sizeof(a[0][0])*16);
}
void Translation(const Point_3 &s){
    double p[4][4]={1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, s.x, s.y, s.z, 1};
    multi(a,p);
}
void Scaling(const Point_3 &s){
    double p[4][4]={s.x, 0, 0, 0, 0, s.y, 0, 0, 0, 0, s.z, 0, 0, 0, 0, 1};
    multi(a,p);
}
void Rotate(const Point_3 &s, double r) {
    double l=s.Length(); double x=s.x/l,y=s.y/l,z=s.z/l;
    double SinA=sin(r),CosA=cos(r);
    double p[4][4]={CosA + (1 - CosA) * x * x, (1 - CosA) * x * y - SinA * z,
(1 - CosA) * x * z + SinA * y, 0,(1 - CosA) * y * x + SinA * z,
 CosA + (1 - CosA) * y * y, (1 - CosA) * y * z - SinA * x, 0,
(1 - CosA) * z * x - SinA * y, (1 - CosA) * z * y + SinA * x, CosA + (1 - CosA)
* z * z, 0, 0, 0, 0, 1};
    multi(a,p);
}
Point_3 opt(const Point_3&s){
    double x,y,z;
    return Point_3( s.x * a[0][0] + s.y * a[1][0] + s.z * a[2][0] + a[3][0],
            s.x * a[0][1] + s.y * a[1][1] + s.z * a[2][1] + a[3][1],
```

```
        s.x * a[0][2] + s.y * a[1][2] + s.z * a[2][2] + a[3][2]);
}
int main(){
    Macro();
    int n;for (scanf("%d", &n); n; n--) {
        char c;  Point_3 p;
        scanf("\n%c%lf%lf%lf", &c, &p.x, &p.y, &p.z);
        if (c == 'T') Translation(p); if (c == 'S') Scaling(p);
        if (c == 'R') {  double r;scanf("%lf\n", &r);
            Rotate(p, r);    //===========绕OP逆时针旋转r角度
    }}
    for (scanf("%d", &n); n; n--) {
        Point_3 p, p2;   scanf("%lf%lf%lf", &p.x, &p.y, &p.z);
        p2 = opt(p);     printf("%f %f %f\n",p2.x,p2.y,p2.z);
}}
```

三维旋转操作

```
//a点绕Ob向量,逆时针旋转弧度angle, sin(angle),cos(angle)先求出来,减少精度问题。
point e1,e2,e3; point Rotate( point a, point b, double angle ){
    b.std();//单位化，注意b不能为（0，0，0）
    e3=b; double lens=a*e3;//dot(a,e3)
    e1=a -  e3*lens; if (e1.len()>(1e-8)) e1.std(); else return a;
    e2=e1/e3; //det(e1,e3)
    double x1,y1,x,y;     y1=a*e1; x1=a*e2;
    x=x1*cos(angle) - y1*sin(angle);  y=x1*sin(angle) + y1*cos(angle);
    return e3*lens + e1*y + e2*x; }
```

三维凸包随机增量

```
struct Point { double x, y, z;  Point() {x = y = z = 0;}
    Point(double x, double y, double z): x(x), y(y), z(z) {}
    bool operator <(const Point &p) const {x,y,z}
    bool operator ==(const Point &p) const {}
    Point cross(const Point &p) const {
        return Point(y * p.z - z * p.y, z * p.x - x * p.z, x * p.y - y * p.x);}
    double dot(const Point &p) const { return x * p.x + y * p.y + z * p.z; }
    double norm() {return dot(*this);}  double length() {return Sqrt(norm());}

};
int mark[1005][1005];Point info[1005];int n, cnt;
double mix(const Point &a, const Point &b, const Point &c) {
    return a.dot(b.cross(c));}
double area(int a, int b, int c) {
    return ((info[b] - info[a]).cross(info[c] - info[a])).length();}
double volume(int a, int b, int c, int d) {
    return mix(info[b] - info[a], info[c] - info[a], info[d] - info[a]);}
struct Face {    int a, b, c;   Face() {}
    Face(int a, int b, int c): a(a), b(b), c(c) {}
int &operator [](int k) {if (k == 0) return a; if (k == 1) return b; return
c; }};
vector <Face> face;
inline void insert(int a, int b, int c) { face.push_back(Face(a, b, c)); }
void add(int v) {
    vector <Face> tmp;    int a, b, c;  cnt ++;
    for (int i = 0; i < SIZE(face); i ++) {
        a = face[i][0]; b = face[i][1];  c = face[i][2];
        if (Sign(volume(v, a, b, c)) < 0)
mark[a][b]=mark[b][a]=mark[b][c] = mark[c][b] = mark[c][a] = mark[a][c] = cnt;
        else  tmp.push_back(face[i]);  }
    face = tmp;
    for (int i = 0; i < SIZE(tmp); i ++) {
        a = face[i][0];   b = face[i][1];   c = face[i][2];
        if (mark[a][b] == cnt) insert(b, a, v);
        if (mark[b][c] == cnt) insert(c, b, v);
        if (mark[c][a] == cnt) insert(a, c, v);
}}
int Find() {
```

```
    for (int i = 2; i < n; i ++) {
        Point ndir = (info[0] - info[i]).cross(info[1] - info[i]);
        if (ndir == Point()) continue;
        swap(info[i], info[2]);
        for (int j = i + 1; j < n; j ++)
            if (Sign(volume(0, 1, 2, j)) != 0) {
                swap(info[j], info[3]);  insert(0, 1, 2); insert(0, 2, 1);
                return 1;
} }    return 0;}
int main() {
    double ans, ret; int Case;
    for (scanf("%d", &Case); Case; Case --) {
        scanf("%d", &n);  for (int i = 0; i < n; i ++) info[i].read();
        sort(info, info + n);  n = unique(info, info + n) - info;
        face.clear(); random_shuffle(info, info + n);
        ans = ret = 0; if (Find()) {
            memset(mark, 0, sizeof(mark)); cnt = 0;
            for (int i = 3; i < n; i ++) add(i);
            int first = face[0][0];
            for (int i = 0; i < SIZE(face); i ++) {
               ret += area(face[i][0], face[i][1], face[i][2]);
               ans += fabs(volume(first, face[i][0], face[i][1], face[i][2]));

            }   ans /= 6;   ret /= 2;   }
        printf("%.3f %.3f\n", ret, ans);
    }  return 0; }
```

直线和凸包交点（返回最近和最远点）

```
double calc(point a, point b){
     double k=atan2(b.y-a.y , b.x-a.x);     if (k<0) k+=2*pi;return k;
}//= the convex must compare y, then x£¬a[0] is the lower-right point
//======= three is no 3 points in line. a[] is convex 0~n-1
void prepare(point a[] ,double w[],int &n) {
    int i;   rep(i,n) a[i+n]=a[i];a[2*n]=a[0];
    rep(i,n) {   w[i]=calc(a[i],a[i+1]);w[i+n]=w[i];}
}
int find(double k,int n , double w[]){
    if (k<=w[0] || k>w[n-1]) return 0;    int l,r,mid; l=0; r=n-1;
    while (l<=r) {   mid=(l+r)/2;if (w[mid]>=k) r=mid-1; else l=mid+1;
    }return r+1;
}
int dic(const point &a, const point &b , int l ,int r , point c[]) {
    int s;   if (area(a,b,c[l])<0) s=-1; else s=1; int mid;
    while (l<=r) {
        mid=(l+r)/2; if (area(a,b,c[mid])*s <= 0) r=mid-1; else l=mid+1;
    }return r+1;
}
point get(const point &a, const point &b, point s1, point s2) {
    double k1,k2; point tmp; k1=area(a,b,s1); k2=area(a,b,s2);
    if (cmp(k1)==0) return s1; if (cmp(k2)==0) return s2;
    tmp=(s1*k2 ¨C s2*k1) / (k2-k1);  return tmp;
}
bool line_cross_convex(point a, point b ,point c[] , int n, point &cp1, point
&cp2 , double w[]) {
    int i,j;
    i=find(calc(a,b),n,w);
    j=find(calc(b,a),n,w);
    double k1,k2;
    k1=area(a,b,c[i]); k2=area(a,b,c[j]);
    if (cmp(k1)*cmp(k2)>0) return false; //no cross
    if (cmp(k1)==0 || cmp(k2)==0) {   //cross a point or a line in the convex
        if (cmp(k1)==0) {
            if (cmp(area(a,b,c[i+1]))==0) {cp1=c[i]; cp2=c[i+1];}
            else cp1=cp2=c[i];  return true;
        }
```

```
        if (cmp(k2)==0) {
            if (cmp(area(a,b,c[j+1]))==0) {cp1=c[j];cp2=c[j+1];}
            else cp1=cp2=c[j];
        }return true;
    }
    if (i>j) swap(i,j); int x,y;  x=dic(a,b,i,j,c); y=dic(a,b,j,i+n,c);
    cp1=get(a,b,c[x-1],c[x]); cp2=get(a,b,c[y-1],c[y]);
    return true;}
```

**KM**

```
const int maxn=200;const int oo=0x7fffffff;
int w[maxn][maxn],x[maxn],y[maxn],px[maxn],py[maxn],sy[maxn],slack[maxn];
int par[maxn];int n;int pa[200][2],pb[200][2],n0,m0,na,nb;char s[200][200];
void adjust(int v){  sy[v]=py[v]; if (px[sy[v]]!=-2) adjust(px[sy[v]]);}
bool find(int v){for (int i=0;i<n;i++)
        if (py[i]==-1){
            if (slack[i]>x[v]+y[i]-w[v][i]){
                slack[i]=x[v]+y[i]-w[v][i];   par[i]=v;}
            if (x[v]+y[i]==w[v][i]){
                py[i]=v; if (sy[i]==-1){adjust(i);    return 1;}
                if (px[sy[i]]!=-1) continue;  px[sy[i]]=i;
                if (find(sy[i])) return 1;
    }}return 0;}
int km(){int i,j,m;
    for (i=0;i<n;i++) sy[i]=-1,y[i]=0;
    for (i=0;i<n;i++) {x[i]=0; for (j=0;j<n;j++) x[i]=max(x[i],w[i][j]);}
    bool flag;
    for (i=0;i<n;i++){
        for (j=0;j<n;j++) px[j]=py[j]=-1,slack[j]=oo;
        px[i]=-2;    if (find(i)) continue; flag=false;
        for (;!flag;){
            m=oo; for (j=0;j<n;j++) if (py[j]==-1) m=min(m,slack[j]);
            for (j=0;j<n;j++){
                if (px[j]!=-1) x[j]-=m;
                if (py[j]!=-1) y[j]+=m;
                else slack[j]-=m;}
            for (j=0;j<n;j++){
                if (py[j]==-1&&!slack[j]){
                    py[j]=par[j];
                    if (sy[j]==-1){  adjust(j);   flag=true;   break;}
                    px[sy[j]]=j;  if (find(sy[j])){flag=true;break;}
            }}}}
    int ans=0; for (i=0;i<n;i++) ans+=w[sy[i]][i];return ans;}
```

**费用流**

```
const int inf = 1000000000;
int s, t, node, totalCost;
vector<int> head, dist, vtx, next, c, cost;
vector<bool> vis;
void resize(vector<T> &a, int size, T init) //设大小、初始值
void init(int source, int target, int nodeCount) //初始化，记得清空
void add(int a,int b,int cc,int cst) //双向加边
void spfa() {
    resize(vis, node, false); resize(dist, node, -inf);
    queue<int> q; q.push(t); vis[t]=true; dist[t]=0;
    while (q.size()) {
        int u = q.front();    q.pop();
        vis[u] = false;
        for (int p = head[u]; p != -1; p = next[p]) {
            if (c[p ^ 1] && dist[u] + cost[p ^ 1] > dist[vtx[p]]) {
                dist[vtx[p]] = dist[u] + cost[p ^ 1];
                if (!vis[vtx[p]]) {
                    vis[vtx[p]] = true;    q.push(vtx[p]);
                    if (dist[q.back()] < dist[q.front()]) {
                        swap(q.front(), q.back());
```

```
}}}}}} //补齐上一页的括号
int dfs(int u, int limit) {
    if (u == t) {
        totalCost += limit * dist[s];
        return limit;
    }
    int current = 0;
    vis[u] = true;
    for (int p = head[u]; p != -1; p = next[p]) {
        if (c[p] && !vis[vtx[p]] && dist[vtx[p]] + cost[p] == dist[u])
{
            int delta = dfs(vtx[p], min(limit - current, c[p]));
            c[p] -= delta; c[p ^ 1] += delta;
            current += delta;
            if (current == limit) {
                break;
            }
        }
    }
    return current;
}
inline bool adjust() {
    int maxi = -inf;
    for (int i = 0; i < node; ++ i) {
        if (vis[i]) {
            for (int p = head[i]; p != -1; p = next[p]) {
                if (c[p] && !vis[vtx[p]]) {
                    assert(dist[vtx[p]] + cost[p] != dist[i]);
                    maxi = max(maxi, dist[vtx[p]] + cost[p] - dist[i]);
                }
            }
        }
    }
    if (maxi == -inf) {
        return false;
    }
    for (int i = 0; i < node; ++ i) {
        if (vis[i]) {
            dist[i] += maxi;
        }
    }
    return true;
}
int maxCostFlow() {
    spfa();
    totalCost = 0;
    do{
        do{
            resize(vis, node, false);
        }while (dfs(s, inf));
    }while (adjust());
    return totalCost;
}
```

无向图最小割

```
#define typec int // type of res (or long long)
const typec inf = 0x3f3f3f3f; // max of res
const typec maxw = 1000; // maximum edge weight, g[i][j]=g[j][i]
typec g[V][V], w[V];  int a[V], v[V], na[V];
typec mincut(int n){
    int i, j, pv, zj;     typec best = maxw * n * n;
    for (i = 0; i < n; i++) v[i] = i; // vertex: 0 ~ n-1
    while (n > 1) {
        for (a[v[0]] = 1, i = 1; i < n; i++) {
```

```
            a[v[i]] = 0; na[i - 1] = i; w[i] = g[v[0]][v[i]];}
        for (pv = v[0], i = 1; i < n; i++ ) {
            for (zj = -1, j = 1; j < n; j++ )
                if (!a[v[j]] && (zj < 0 || w[j] > w[zj]))  zj = j;
            a[v[zj]] = 1;
            if (i == n - 1) {
                if (best > w[zj]) best = w[zj];
                for (i = 0; i < n; i++)
                    g[v[i]][pv] = g[pv][v[i]]+=g[v[zj]][v[i]];
                v[zj] = v[--n]; break;
            }   pv = v[zj];
            for (j = 1; j < n; j++) if(!a[v[j]]) w[j] += g[v[zj]][v[j]];
    }} return best;}
```

一般图最大匹配_片段

```
const int maxn=310;
vector<int> link[maxn];
int n; int match[maxn]; int Queue[maxn], head, tail; int pred[maxn],
base[maxn];
bool InQueue[maxn], InBlossom[maxn]; bool use[maxn];  //===这个点是否有用
int start, finish; int newbase;
void push(int u) {   Queue[tail++] = u; InQueue[u] = true; }
int pop() {  return Queue[head++];}
int FindCommonAncestor(int u, int v) {
    bool InPath[maxn];    for (int i = 0; i < n; i++) InPath[i] = 0;
    while(true) {
        u = base[u]; InPath[u] = true;
        if(u == start) break; u = pred[match[u]];}
    while(true) {v = base[v]; if(InPath[v]) break; v = pred[match[v]];  }
    return v;}
void ResetTrace(int u) {
    int v;
    while(base[u] != newbase) {
        v = match[u]; InBlossom[base[u]] = InBlossom[base[v]] = true;
        u = pred[v]; if(base[u] != newbase) pred[u] = v;}}
void BlossomContract(int u, int v) {
    newbase = FindCommonAncestor(u, v);
    for (int i = 0; i < n; i++) InBlossom[i] = 0;
    ResetTrace(u); ResetTrace(v);
    if(base[u] != newbase) pred[u]=v;if(base[v] != newbase) pred[v]=u;
    for(int i = 0; i < n; ++i)
     if(InBlossom[base[i]]) {base[i]=newbase;  if(!InQueue[i]) push(i);}}
bool FindAugmentingPath(int u) {
    bool found = false;
    for(int i = 0; i < n; ++i) pred[i] = -1, base[i] = i;
    for (int i = 0; i < n; i++) InQueue[i] = 0;
    start = u; finish = -1; head = tail = 0;  push(start);
    while(head < tail) {
        int u = pop();
        for(int i = link[u].size() - 1; i >= 0; i--) {
            int v = link[u][i];
            if(use[u] && use[v] && base[u] != base[v] && match[u] != v)
            if(v == start || (match[v] >= 0 && pred[match[v]] >= 0))
                BlossomContract(u, v);
            else if(pred[v] == -1) {pred[v] = u;
                if(match[v] >= 0) push(match[v]);
                else {finish = v;    return true;}
    }}} return found;}
void AugmentPath() {
    int u, v, w; u = finish;
while(u >= 0) {  v = pred[u];w = match[v];match[v] = u;match[u] = v;u = w;}}
void FindMaxMatching() {
    for(int i = 0; i < n; ++i) match[i] = -1;
    for(int i = 0; i < n; ++i)
      if(match[i] == -1 && use[i])if(FindAugmentingPath(i)) AugmentPath();}
```

```
int main() {
    foru(i,0,n) link[i].clear();  memset(use,1,sizeof(use));
//========编号从0~n-1 ， link[i] push_back所有i号点连向的点。  双向边
    FindMaxMatching();   k=0;rep(i,n) if (match[i]>=0) k++;
    printf("%d\n",k/2);  return 0;
}
```

<u>有向图最小生成树</u>

```
const int maxn=1100;  int n,m , g[maxn][maxn] , used[maxn] , pass[maxn] ;
int eg[maxn] , more , queue[maxn];
void combine (int id , int &sum ) {
    int tot = 0 , from , i , j , k ;
    for ( ; id!=0 && !pass[ id ] ; id=eg[id] ) {
        queue[tot++]=id ; pass[id]=1; }
    for ( from=0; from<tot && queue[from]!=id ; from++);
    if  ( from==tot ) return ;
    more = 1 ;
    for ( i=from ; i<tot ; i++) {
        sum+=g[eg[queue[i]]][queue[i]] ;
        if ( i!=from ) {
            used[queue[i]]=1;
            for ( j = 1 ; j <= n ; j++) if ( !used[j] )
                if ( g[queue[i]][j]<g[id][j] ) g[id][j]=g[queue[i]][j] ;}}
    for ( i=1; i<=n ; i++) if ( !used[i] && i!=id ) {
        for ( j=from ; j<tot ; j++){ k=queue[j];
            if ( g[i][id]>g[i][k]-g[eg[k]][k] )
g[i][id]=g[i][k]-g[eg[k]][k]; }}}
int mdst( int root ) { // return the total length of MDST
    int i , j , k , sum = 0 ;
    memset ( used , 0 , sizeof ( used ) ) ;
    for ( more =1; more ; ) {
        more = 0 ;   memset (eg,0,sizeof(eg)) ;
        for ( i=1 ; i <= n ; i ++) if ( !used[i] && i!=root ) {
            for ( j=1 , k=0 ; j <= n ; j ++) if ( !used[j] && i!=j )
                if ( k==0 || g[j][i] < g[k][i] ) k=j ;
            eg[i] = k ;
        } memset(pass,0,sizeof(pass));
        for ( i=1;i<=n;i++) if (!used[i] && !pass[i] && i!=
    root )combine(i,sum);
    }
    for ( i =1; i<=n ; i ++) if ( !used[i] && i!= root ) sum+=g[eg[i]][i];
    return sum ; }
int main(){
  int i,j,k,test,cases; cases=0; scanf("%d%d",&n,&m);
  foru(i,1,n) foru(j,1,n) g[i][j]=1000001;
  foru(i,1,m) {scanf("%d%d",&j,&k);j++;k++;scanf("%d",&g[j][k]);}
  k=mdst(1); if (k>1000000) printf("Possums!\n");  //===no
  else printf("%d\n",k);   return 0;}
```

<u>**Hopcroft**</u>

```
#define maxn 50005   #define maxm 150005
inline int Maxmatch(){
    memset(mk,-1,sizeof(mk));memset(cx,-1,sizeof(cx));
    memset(cy,-1,sizeof(cy));
    for (int p=1,fl=1,h,tail;fl;++p){
        fl=0;   h=tail=0;
        for (int i=0;i<n;++i) if (cx[i]==-1)
            q[++tail]=i,pre[i]=-1,src[i]=i;
        for (h=1;h<=tail;++h){
            int u=q[h];    if (cx[src[u]]!=-1) continue;
            for (int pp=head[u],v=vtx[pp];pp;pp=next[pp],v=vtx[pp])
            if (mk[v]!=p)  {   mk[v]=p;   q[++tail]=cy[v];
                if (cy[v]>=0)  {
                    pre[cy[v]]=u; src[cy[v]]=src[u];continue;
                }  int d,e,t;
```

```
                for
(--tail,fl=1,d=u,e=v;d!=-1;t=cx[d],cx[d]=e,cy[e]=d,e=t,d=pre[d]);
                break;      }   }   }
int res=0;  for (int i=0;i<n;++i)  res+=(cx[i]!=-1);return res;}
```

割点缩块 /*考虑割点的无向图缩块*/

```
const int maxn = 100000+5; const int maxm = 200000+5;
int e[maxm],prev[maxm],info[maxn],dfn[maxn],low[maxn],stack[maxn];
vector<int> Block[maxn]; int cntB,cnt,top,tote;
void insertE( int x,int y ){
    ++tote; e[tote]=y; prev[tote]=info[x]; info[x]=tote;}
void Min( int &x,int y ){if(y < x) x = y;}
void Dfs( int x,int father ){
    dfn[x] = low[x] = ++cnt; stack[++top] = x;
    for(int t=info[x];t;t=prev[t])
        if(dfn[e[t]] == 0)    {
            int tmp = top;   Dfs(e[t],x); Min(low[x],low[e[t]]);
            if(low[e[t]] >= dfn[x]){
                Block[++cntB].clear();
                for(int k=tmp+1;k<=top;++k)
Block[cntB].push_back(stack[k]);
                Block[cntB].push_back(x); top=tmp; }
        }else if(e[t]!=father)    Min(low[x],dfn[e[t]]);}
int main(){
    int n,m; scanf("%d%d",&n,&m); memset(info,0,sizeof(info)); tote=0;
    for(int i=0;i<m;++i){
        int x,y; scanf("%d%d",&x,&y); insertE(x,y); insertE(y,x);}
    memset(dfn,0,sizeof(dfn)); cnt=top=cntB=0;
    for(int i=1;i<=n;++i) if(dfn[i] == 0) Dfs(i,-1);
    printf("%d\n",cntB);
    for(int i=1;i<=cntB;++i){
      for(int j=0;j<Block[i].size();++j) printf("%d
",Block[i][j]);puts("");
    }return 0;}
```

**Manacher**

```
void manacher(char text[], int n, int palindrome[]) {
    palindrome[0] = 1;
    for (int i = 1, j = 0, i < (n << 1) - 1; ++ i) {
        int p = i >> 1;
        int q = i - p;
        int r = (j + 1 >> 1) + palindrome[j] - 1;
        palindrome[i] = r < q? 0: min(r - q + 1, palindrome[(j << 1)
- i]);
        while (0 <= p - palindrome[i] && q + palindrome[i] < n
                && text[p - palindrome[i]] == text[q + palindrome[i]])
{
            palindrome[i] ++;
        }
        if (q + palindrome[i] - 1 > r) {
            j = i;
        }
    }
}
```

回文串//=o(n) 统计出，r(i) 表示 (i-r[i]+1 , i)==(i+r[i], i+1)

```
void calc_radius(char s[]){
    for (int i=0,j=0,k=0;i<len;){
        while (i - j >= 0  && i+j+1 < len && s[i-j] == s[i+j+1]) j++;
        radius[i] = j;   k = 1;
        while (k <= radius[i] && radius[i-k] < radius[i] - k)  {
            radius[i+k] = min(radius[i-k],radius[i] - k);  k++;
        }j = max(j - k,0);   i += k;
}}
```

**dc3**

```
//DC3 待排序的字符串放在r 数组中，从r[0]到r[n-1]，长度为n，且最大值小于m。
```

```
//约定除r[n-1]外所有的r[i]都大于0, r[n-1]=0。
//函数结束后，结果放在sa 数组中，从sa[0]到sa[n-1]。
#define maxn 10000
#define F(x) ((x)/3+((x)%3==1?0:tb))
#define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
int wa[maxn],wb[maxn],wv[maxn],wss[maxn]; //必须这么大
int s[maxn*3],sa[maxn*3];
int c0(int *r,int a,int b){return
    r[a]==r[b]&&r[a+1]==r[b+1]&&r[a+2]==r[b+2];}
int c12(int k,int *r,int a,int b){
    if(k==2) return r[a]<r[b]||r[a]==r[b]&&c12(1,r,a+1,b+1);
    else return r[a]<r[b]||r[a]==r[b]&&wv[a+1]<wv[b+1];
}
void sort(int *r,int *a,int *b,int n,int m){
    int i;  for(i=0;i<n;i++) wv[i]=r[a[i]];
    for(i=0;i<m;i++) wss[i]=0; for(i=0;i<n;i++) wss[wv[i]]++;
    for(i=1;i<m;i++) wss[i]+=wss[i-1];
    for(i=n-1;i>=0;i--) b[--wss[wv[i]]]=a[i];
}
void dc3(int *r,int *sa,int n,int m){
    int i,j,*rn=r+n,*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;
    r[n]=r[n+1]=0;
    for(i=0;i<n;i++) if(i%3!=0) wa[tbc++]=i;
    sort(r+2,wa,wb,tbc,m);   sort(r+1,wb,wa,tbc,m);
    sort(r,wa,wb,tbc,m);
    for(p=1,rn[F(wb[0])]=0,i=1;i<tbc;i++)
        rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;
    if (p<tbc) dc3(rn,san,tbc,p);
    else for (i=0;i<tbc;i++) san[rn[i]]=i;
    for (i=0;i<tbc;i++)  if(san[i]<tb) wb[ta++]=san[i]*3;
    if(n%3==1) wb[ta++]=n-1;
    sort(r,wb,wa,ta,m);
    for(i=0;i<tbc;i++)   wv[wb[i]=G(san[i])]=i;
    for(i=0,j=0,p=0;i<ta && j<tbc;p++)
        sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];
    for(;i<ta;p++) sa[p]=wa[i++]; for(;j<tbc;p++) sa[p]=wb[j++];}
int main(){
    int n,m=0;   scanf("%d",&n);
    for (int i=0;i<n;i++) scanf("%d",&s[i]),s[i]++,m=max(s[i]+1,m);
    printf("%d\n",m);    s[n++]=0;    dc3(s,sa,n,m);
    for (int i=0;i<n;i++) printf("%d ",sa[i]);printf("\n");
}
```

**<u>最大团搜索算法</u>**

Int g[][]为图的邻接矩阵。 MC(V)表示点集V的最大团

令Si={vi, vi+1, ..., vn}, mc[i]表示MC(Si). 倒着算mc[i]，那么显然MC(V)=mc[1]

此外有mc[i]=mc[i+1] **or** mc[i]=mc[i+1]+1

```
void init(){
    int i, j;for (i=1; i<=n; ++i) for (j=1; j<=n; ++j) scanf("%d", &g[i][j]);
}
void dfs(int size){
    int i, j, k;
    if (len[size]==0) {  if (size>ans) {  ans=size; found=true;} return;}
    for (k=0; k<len[size] && !found; ++k) {
        if (size+len[size]-k<=ans) break;
        i=list[size][k]; if (size+mc[i]<=ans) break;
        for (j=k+1, len[size+1]=0; j<len[size]; ++j)
        if (g[i][list[size][j]])
list[size+1][len[size+1]++]=list[size][j];
        dfs(size+1);}}
void work(){
    int i, j;   mc[n]=ans=1;
    for (i=n-1; i; --i) {found=false; len[1]=0;
        for (j=i+1; j<=n; ++j) if (g[i][j]) list[1][len[1]++]=j;
        dfs(1);  mc[i]=ans;}}
```

**极大团的计数**

Bool g[][] 为图的邻接矩阵，图点的标号由1至n。

```
void dfs(int size){
    int i, j, k, t, cnt, best = 0;    bool bb;
    if (ne[size]==ce[size]){if (ce[size]==0) ++ans;return;}
    for (t=0, i=1; i<=ne[size]; ++i) {
        for (cnt=0, j=ne[size]+1; j<=ce[size]; ++j)
        if (!g[list[size][i]][list[size][j]]) ++cnt;
        if (t==0 || cnt<best) t=i, best=cnt;
    }
    if (t && best<=0) return;
    for (k=ne[size]+1; k<=ce[size]; ++k) {
        if (t>0){
            for (i=k; i<=ce[size]; ++i)
                if (!g[list[size][t]][list[size][i]]) break;
            swap(list[size][k], list[size][i]);
        }
        i=list[size][k]; ne[size+1]=ce[size+1]=0;
        for (j=1; j<k; ++j)if (g[i][list[size][j]])
            list[size+1][++ne[size+1]]=list[size][j];
        for (ce[size+1]=ne[size+1], j=k+1; j<=ce[size]; ++j)
        if (g[i][list[size][j]]) list[size+1][++ce[size+1]]=list[size][j];
        dfs(size+1); ++ne[size]; --best;
        for (j=k+1, cnt=0; j<=ce[size]; ++j) if (!g[i][list[size][j]]) ++cnt;
        if (t==0 || cnt<best) t=k, best=cnt;
        if (t && best<=0) break;
}}
void work(){
    int i;   ne[0]=0; ce[0]=0; for (i=1; i<=n; ++i) list[0][++ce[0]]=i;
    ans=0;   dfs(0);}
```

**FFT**

```
void FFT(Complex P[], int n, int oper)
{
    for (int i = 1, j = 0; i < n - 1; i++) {
        for (int s = n; j ^= s >>= 1, ~j & s;);
        if (i < j) {
            swap(P[i], P[j]);
        }
    }
    Complex unit_p0;
    for (int d = 0; (1 << d) < n; d++) {
        int m = 1 << d, m2 = m * 2;
        double p0 = pi / m * oper;
        sincos(p0, &unit_p0.y, &unit_p0.x);
        for (int i = 0; i < n; i += m2) {
            Complex unit = 1;
            for (int j = 0; j < m; j++) {
                Complex &P1 = P[i + j + m], &P2 = P[i + j];
                Complex t = unit * P1;
                P1 = P2 - t;
                P2 = P2 + t;
                unit = unit * unit_p0;
            }
        }
    }
}
```

**Simpson**

```
double simpson(const T&f,double a,double b,int n){
    const double h=(b-a)/n; double ans=f(a)+f(b);
    for(int i=1;i<n;i+=2)ans+=4*f(a+i*h);
    for(int i=2;i<n;i+=2)ans+=2*f(a+i*h);
    return ans*h/3;
}printf("%lf\n",simpson(test,0,1,(int)1e6)
```

长方体表面两点最短距离

```
int r;
void turn(int i, int j, int x, int y, int z,int x0, int y0, int L, int W, int
H) {
    if (z==0) {  int R = x*x+y*y;if (R<r) r=R;}
    else{
        if(i>=0 && i< 2)turn(i+1, j, x0+L+z, y, x0+L-x, x0+L, y0, H, W, L);
        if(j>=0 && j< 2)turn(i, j+1, x, y0+W+z, y0+W-y, x0, y0+W, L, H, W);
        if(i<=0 && i>-2)turn(i-1, j, x0-z, y, x-x0, x0-H, y0, H, W, L);
        if(j<=0 && j>-2)turn(i, j-1, x, y0-z, y-y0, x0, y0-H, L, H, W);
}}
int main(){
    int L, H, W, x1, y1, z1, x2, y2, z2;
    cin >> L >> W >> H >> x1 >> y1 >> z1 >> x2 >> y2 >> z2;
    if (z1!=0 && z1!=H)  if (y1==0 || y1==W)
         swap(y1,z1), std::swap(y2,z2), std::swap(W,H);
    else swap(x1,z1), std::swap(x2,z2), std::swap(L,H);
    if (z1==H) z1=0, z2=H-z2;
    r=0x3fffffff; turn(0,0,x2-x1,y2-y1,z2,-x1,-y1,L,W,H);
    cout<<r<<endl;   return 0;
}
```

字符串的最小表示

```
A[1..n]; A[n+1..n+n]=A[1..n]; i:=1;   j:=2;   k:=0;   t:=0;
while (j<=n) {     k=0;  while (a[i+k]=a[j+k]) k++;
     if (a[i+k]>a[j+k]) i=i+k+1; else  j=j+k+1;
     if (i==j) j++;  if (i>j) swap(i,j);
} printf("%d\n",i);
```

二次剩余

```
/*a*x^2+b*x+c==0 (mod P) 求 0..P-1 的根 */
int pDiv2,P,a,b,c,Pb,d;
inline int calc(int x,int Time){
    if (!Time) return 1;  int tmp=calc(x,Time/2);
    tmp=(long long)tmp*tmp%P;
    if (Time&1) tmp=(long long)tmp*x%P;    return tmp;
}
inline int rev(int x){ if (!x) return 0;   return calc(x,P-2);}
inline void Compute(){
    while (1) { b=rand()%(P-2)+2;  if (calc(b,pDiv2)+1==P) return; }
}
int main(){
    srand(time(0)^312314);  int T;
    for (scanf("%d",&T);T;--T) {
        scanf("%d%d%d%d",&a,&b,&c,&P);
        if (P==2)  {
            int cnt=0; for (int i=0;i<2;++i) if ((a*i*i+b*i+c)%P==0) ++cnt;
            printf("%d",cnt);
            for (int i=0;i<2;++i) if ((a*i*i+b*i+c)%P==0) printf(" %d",i);
            puts("");
        }else  {
            int delta=(long long)b*rev(a)*rev(2)%P;
            a=(long long)c*rev(a)%P-sqr( (long long)delta )%P;
            a%=P;a+=P;a%=P;  a=P-a;a%=P;  pDiv2=P/2;
            if (calc(a,pDiv2)+1==P) puts("0");
            else {
                int t=0,h=pDiv2;    while (!(h%2)) ++t,h/=2;
                int root=calc(a,h/2);
                if (t>0) {  Compute();  Pb=calc(b,h); }
                for (int i=1;i<=t;++i) {
                    d=(long long)root*root*a%P;
                    for (int j=1;j<=t-i;++j)  d=(long long)d*d%P;
                    if (d+1==P)  root=(long long)root*Pb%P;
                    Pb=(long long)Pb*Pb%P;
                }
                root=(long long)a*root%P;
```

```
            int root1=P-root; root-=delta;
            root%=P; if (root<0) root+=P;
            root1-=delta; root1%=P;  if (root1<0) root1+=P;
            if (root>root1) { t=root;root=root1;root1=t;  }
            if (root==root1) printf("1 %d\n",root);
            else printf("2 %d %d\n",root,root1);
    }}}return 0;  }
```

**Pell 方程求解**

```
//求x^2-ny^2=1的最小正整数根,n不是完全平方数
p[1]=1;p[0]=0; q[1]=0;q[0]=1; a[2]=(int)(floor(sqrt(n)+1e-7));
g[1]=0;h[1]=1;
for (int i=2;i;++i) {
    g[i]=-g[i-1]+a[i]*h[i-1];     h[i]=(n-sqr(g[i]))/h[i-1];
    a[i+1]=(g[i]+a[2])/h[i];      p[i]=a[i]*p[i-1]+p[i-2];
    q[i]=a[i]*q[i-1]+q[i-2];      检查p[i],q[i]是否为解，如果是，则退出
}
```

**莫比乌斯函数以及 gcd=1 的对数**

```
inline void prepare()//计算莫比乌斯函数，及其前缀和sum，复杂度O(nlogn)
inline void calc(int a,int b) {
    for (int i=1,j,p,q;i<=a;i=j+1) {
        p=a/i;q=b/i; j=b/q; if (a<p*j) j=a/p;
        ans+=(long long)(sum[j]-sum[i-1])*p*q;
    }  }//求1..a和1..b中有多少对的gcd=1，复杂度O(sqrt(a+b))
```

**Exact Cover**

```
class ExactCover{
private:
    vector<int> u,d,l,r,C,R,head,tail;
    int head0,tail0,seed;
    void cover(int x){
        int i=x,j;
        r[l[x]]=r[x];  l[r[x]]=l[x];
        while((i=d[i])!=x){
            j=i;
            while((j=l[j])!=i){
                u[d[j]]=u[j];  d[u[j]]=d[j];  R[C[j]]--;
            }
        }
    }
    void uncover(int x){
        int i=x,j;
        while((i=u[i])!=x){
            j=i;
            while((j=r[j])!=i){
                u[d[j]]=j;  d[u[j]]=j;  R[C[j]]++;
            }
        }
        r[l[x]]=x;  l[r[x]]=x;
    }
public:
    vector<int> ans;
    void resize(int n){
        u.resize(1,0);  d.resize(1,0);  l.resize(1,0); r.resize(1,0);
        C.resize(1,-1);  R.resize(1,-1);
        head.resize(n,-1);  tail.resize(n,-1);
        ans.resize(n,0);  head0=tail0=0;
    }
    void add(vector<int> a,bool must=true){
        u.push_back(u.size()+a.size());
        if(must){
            l.push_back(tail0);    r.push_back(head0);
            tail0=l[r[d.size()]]=r[l[d.size()]]=d.size();
        }else{
```

```
            l.push_back(l.size()); r.push_back(r.size());
        }
        C.push_back(C.size()); R.push_back(a.size());
        int n=u.size(),m=a.size(),i,j;
        for(i=0;i<m;i++){
            j=a[i];
            if(head[j]==-1){
                l.push_back(n+i);  r.push_back(n+i);
                head[j]=n+i;  tail[j]=n+i;
            }else{
                l.push_back(tail[j]); r.push_back(head[j]);
                tail[j]=r[l[n+i]]=l[r[n+i]]=n+i;
            }
            u.push_back(n+i-1);    d.push_back(n+i);
            C.push_back(C.back()); R.push_back(j);
        }
        d.push_back(n-1);
    }
    void select(int a){
        ans[a]=1; a=head[a];
        if(a==-1) return;
        int x=a;
        while((x=r[x])!=a) cover(C[x]);
        cover(C[a]);
    }
    bool search(){
        if(r[0]==0)    return true;
        int x,i,j,min=0x7fffffff;
        i=0;
        while((i=r[i])!=0)
            if(R[i]<min||!(++seed&3)&&R[i]==min)
                min=R[x=i];
        cover(i=x);
        while((i=d[i])!=x){
            j=i;
            while((j=r[j])!=i) cover(C[j]);
            ans[R[i]]=1;
            if(search()) return true;
            ans[R[i]]=0;
            while((j=l[j])!=i) uncover(C[j]);
        }
        uncover(x);
        return false;
    }
};
```

**Link-Cut-Tree**

```
void rotate(int x) {
    int t = type[x];
    int y = parent[x];
    int z = children[x][1 ^ t];
    type[x] = type[y];
    parent[x] = parent[y];
    if (type[x] != 2) {
        children[parent[x]][type[x]] = x;
    }
    type[y] = 1 ^ t;
    parent[y] = x;
    children[x][1 ^ t] = y;
    if (z != 0) {
        type[z] = t;
        parent[z] = y;
```

```
    }
    children[y][t] = z;
    update(y);
}
void splay(int x) {
    vector <int> stack(1, x);
    for (int i = x; type[i] != 2; i = parent[i]) {
        stack.push_back(parent[i]);
    }
    while (!stack.empty()) {
        push(stack.back());
        stack.pop_back();
    }
    while (type[x] != 2) {
        int y = parent[x];
        if (type[x] == type[y]) {
            rotate(y);
        } else {
            rotate(x);
        }
        if (type[x] == 2) {
            break;
        }
        rotate(x);
    }
    update(x);
}
void access(int x) {
    int z = 0;
    while (x != 0) {
        splay(x);
        type[children[x][1]] = 2;
        children[x][1] = z;
        type[z] = 1;
        update(x);
        z = x;
        x = parent[x];
    }
}
```

后缀自动机

```
struct State {
    static vector <State*> states;
    int id, length;
    State *parent;
    State* go[C];
    State(int length) : id((int)states.size()), length(length),
parent(NULL) {
        memset(go, NULL, sizeof(go));
        states.push_back(this);
    }
    State* extend(State* start, int token) {
        State *p = this;
        State *np = new State(length + 1);
        while (p && !p->go[token]) {
            p->go[token] = np;
            p = p->parent;
        }
        if (!p) {
            np->parent = start;
        } else {
            State *q = p->go[token];
            if (p->length + 1 == q->length) {
```

```
                np->parent = q;
            } else {
                State *nq = new State(p->length + 1);
                memcpy(nq->go, q->go, sizeof(q->go));
                nq->parent = q->parent;
                np->parent = q->parent = nq;
                while (p && p->go[token] == q) {
                    p->go[token] = nq;
                    p = p->parent;
                }
            }
        }
        return np;
    }
};
```

差分序列

```
F(n) = c0 * C(n, 0) + c1 * C(n, 1) + ... + cp * C(n, p)
S(n) = F(0) + F(1) + ... + F(n)
    = c0 * C(n + 1, 1) + c1 * (n + 1, 2) + ... + cp * C(n + 1, p + 1)
```

求某年某月某日是星期几

```
int whatday(int d, int m, int y) { //day month year
    int ans;    if (m == 1 || m == 2) { m += 12; y --; }
    if ((y < 1752) || (y == 1752 && m < 9)||(y == 1752 && m == 9 && d < 3))
            ans = (d + 2*m + 3*(m+1)/5 + y + y/4 +5) % 7;
    else ans = (d + 2*m + 3*(m+1)/5 + y + y/4 - y/100 + y/400)%7;
    return ans;
}
```

弦图的完美消除序列

从n到1的顺序依次给点标号（标号为i的点出现在完美消除序列的第i个）

设lable[i]表示第i个点与多少个已标号的点相邻，每次选择label[i]最大的未标号的点进行标号。

任取一个已标号的与当前新标号的点相邻的点，如果与其他的已标号的且与当前点相邻的点之间没有边，则无解。

弦图里的团数等于色数，色数（从后往前）和最大独立集（从前往后）都可以按完美消除序列的顺序贪心。

双人零和矩阵游戏（公式）

N*N的方阵A，选行的玩家的最优策略是p，选列的是q，则

q = A逆 * e / (e转置 * A逆 * e)

p转置 = e转置 * A逆 / (e转置 * A逆 * e)    e是全为1的列向量

当A不可逆时，每个元素加上一个值就可以了。

当矩阵是m行，n列的时候：

P[1]+P[2]+……+P[m]=1; P[i]>=0

V<=sigma(P[i]*Matrix[i][j])

最大化V

质数测试

```
bool primeTest(LL n, LL b) {
    LL m = n - 1;
    LL counter = 0;
    while ((m & 1) == 0) {
        m >>= 1;
        counter ++;
    }
    LL ret = powMod(b, m, n);
    if (ret == 1 || ret == n - 1) {
        return true;
    }
    counter --;
    while (counter >= 0) {
        ret = multiplyMod(ret, ret, n);
        if (ret == n - 1) {
            return true;
```

```
        }
        counter --;
    }
    return false;
}
const int BASIC[12] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};
bool isPrime(LL n) {
    if (n < 2) return false;
    if (n < 4) return true;
    if (n == 3215031751LL) return false;
    for (int i = 0; i < 12 && BASIC[i] < n; ++ i)
        if (!primeTest(n, BASIC[i])) return false;
    return true;
}
```

**Pollard-Rho**

```
LL pollardRho(LL n, LL seed) {
    LL x, y;
    x = y = rand() % (n - 1) + 1;
    LL head = 1, tail = 2;
    while (true) {
        x = multiplyMod(x, x, n);
        x = addMod(x, seed, n);
        if (x == y) return n;
        LL d = gcd(abs(x - y), n);
        if (1 < d && d < n) return d;
        head ++;
        if (head == tail) {
            y = x;
            tail <<= 1;
        }
    }
}
vector <LL> divisors;
void factorize(LL n) {
    if (n > 1) {
        if (isPrime(n)) {
            divisors.push_back(n);
        } else {
            LL d = n;
            while (d >= n) {
                d = pollardRho(n, rand() % (n - 1) + 1);
            }
            factorize(n / d); factorize(d);
        }
    }
}
```

直线下有多少个格点

求$\sum_{k=0..n-1}\left\lfloor\frac{a+bk}{m}\right\rfloor, a, b > 0$

```
LL count(LL n, LL a, LL b, LL m) {
    if (b==0) return n * (a / m);
    if (a>=m) return n * (a / m) + count(n, a % m, b, m);
    if (b>=m) return (n - 1) * n / 2 * (b / m) + count(n, a, b % m, m);
    return count((a + b * n) / m, (a + b * n) % m, m, b);
}
```

综合

设正整数 n 的质因数分解为 n = ∏pi^ai,则 x^2+y^2=n 有整数解的充要条件是 n 中不存在形如 pi≡3(mod 4) &(and) 指数 ai 为奇数的质因数 pi

Pick 定理：简单多边形，不自交。(严格在多边形内部的整点数*2 +在边上的整点数– 2)/2 =面积

定理 1：最小覆盖数 = 最大匹配数　定理 2：最大独立集 S 与 最小覆盖集 T 互补。

算法：

1. 做最大匹配，没有匹配的空闲点∈S 2. 如果 u∈S 那么 u 的临点必然属于 T
3. 如果一对匹配的点中有一个属于 T 那么另外一个属于 S
4. 还不能确定的，把左子图的放入 S，右子图放入 T 算法结束

有上下界网络流，可行流增广的流量不是实际流量。若要求实际流量应该强算一遍源点出去的流量。
求最小下届网络流： 方法一：加 t-s 的无穷大流，求可行流，然后把边反向后（减去下届网络流），在残留网络中从汇到源做最大流。
方法二：在求可行流的时候，不加从汇到源的无穷大边，得到最大流 X， 加上从汇到源无穷大边后，再求最大流得到 Y。那么 Y 即是答案最小下届网络流。

gcd(2^(a)-1,2^(b)-1)=(2^gcd(a,b))-1. Fibonacci 数 gcd(Fn，Fm)=Fgcd(n,m)

Fibonacci 质数（和前面所有的 Fibonacci 数互质）（大多已经是质数了，可能有 BUG 吧，不确定）
定理：如果 a 是 b 的倍数，那么 Fa 是 Fb 的倍数。
二次剩余：p 为奇素数，若(a,p)=1，a 为 p 的二次剩余必要充分条件为 a^((p-1)/2) mod p=1.
(否则为 p-1)

p 为奇素数，x^b = a(mod p),x 为 p 的 b 次剩余的必要充分条件为 若 x^（(p-1)/ (p-1 和 b 的最大公约数)） mod p=1.

最小二乘法。对于方程组 AX=b，构造方程组$A' \times A \times x = A' \times b$，则 x 一定有解。
混合图欧拉路算法。S 向出度过多的点连边，权值为过多的出度的一半。入度过多的点向 T 连边，权值为过多的入度的一半。若双向边(a,b)的初始方向是 a ← b，则 a 到 b 连边权值为 1.找到一条路就把路上所有边反向。

**java scl**

```java
public class main{
    public static StringTokenizer st; public static DataInputStream in;
    public static PrintStream out;
    public static BigInteger getsqrt(BigInteger n){
        if (n.compareTo(BigInteger.ZERO)<=0) return n;
        BigInteger x,xx,txx;     xx=x=BigInteger.ZERO;
        for (int t=n.bitLength()/2;t>=0;t--){
        txx=xx.add(x.shiftLeft(t+1)).add(BigInteger.ONE.shiftLeft(t+t));
        if (txx.compareTo(n)<=0){
                x=x.add(BigInteger.ONE.shiftLeft(t)); xx=txx;
            }}return x;
    }
    public static void main(String args[]) throws Exception{
        in=new DataInputStream(System.in);
        out=new PrintStream(new BufferedOutputStream(System.out));
        st=new StringTokenizer(in.readLine()); out.close();
    }
}//BigInteger
a.modPow(b,c);//a^b mod c;    a.isProbablePrime(int certainty);
a.nextProbablePrime();        a.shiftLeft(int);
bitCount()       bitLength()       clearBit(int i)
setBit(int i)    flipBit(int i)    testBit(int i)
//BigDecimal
static int ROUND_CEILING,ROUND_DOWN,ROUND_FLOOR,
ROUND_HALF_DOWN,ROUND_HALF_EVEN,ROUND_HALF_UP,ROUND_UP;
a.stripTrailingZeros();
//Vector
a.add((index),elem); a.remove(index); a.set(index,elem);
//Queue
a.add(elem);     a.peek();//front     a.poll();//pop
cout.setf(ios::fixed,ios::floatfield);
cout.precision(3); cout<<double(u)<<endl;
```

**基本形 公式**

**椭圆：**

椭圆$\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$，其中离心率$e=\frac{c}{a}$，$c=\sqrt{a^2-b^2}$;焦点参数$p=\frac{b^2}{a}$

椭圆上(x,y)点处的曲率半径为 $R = a^2b^2\left(\frac{x^2}{a^4}+\frac{y^2}{b^4}\right)^{\frac{3}{2}} = \frac{(r_1r_2)^{\frac{3}{2}}}{ab}$，其中$r_1$和$r_2$分别为(x,y)与两焦点$F_1$和$F_2$的距离。设点 A 和点 M 的坐标分别为(a,0)和(x,y)，则 AM 的弧长为

$$L_{AM} = a\int_0^{\arccos\frac{x}{a}}\sqrt{1-e^2\cos^2 t}\,dt = a\int_{\arccos\frac{x}{a}}^{\frac{\pi}{2}}\sqrt{1-e^2\sin^2 t}\,dt$$

椭圆的周长为 $L = 4a\int_0^{\frac{\pi}{2}}\sqrt{1-e^2\sin^2 t}\,dt = 4aE(e,\frac{\pi}{2})$ ，其中

$$E\left(e,\frac{\pi}{2}\right) = \frac{\pi}{2}[1-\left(\frac{1}{2}\right)^2 e^2 - \left(\frac{1*3}{2*4}\right)^2\frac{e^4}{3} - \left(\frac{1*3*5}{2*4*6}\right)^2\frac{e^6}{5} - \cdots$$

设椭圆上点 M(x,y),N(x,-y),x,y>0,A(a,0),原点 O(0,0)。

扇形 OAM 的面积 $S_{OAM} = \frac{1}{2}ab\arccos\frac{x}{a}$ 弓形 MAN 的面积$S_{MAN} = ab\arccos\frac{x}{a} - xy$

方程，5 个点确定一个圆锥曲线。

θ 为(x,y)点关于椭圆中心的极角，r 为(x,y)到椭圆中心的距离，椭圆极坐标方程:

$$x = r\cos\theta, y = r\sin\theta，其中r^2 = \frac{b^2a^2}{b^2\cos^2\theta + a^2\sin^2\theta}$$

**抛物线**

标准方程 $y^2 = 2px$ 曲率半径 $R = ((p+2x)\text{^}(\mathbf{3/2}))/\boldsymbol{sqrt(p)}$

弧长：设 M(x,y)是抛物线上一点，则 $L_{OM} = \frac{p}{2}[\sqrt{\frac{2x}{p}\left(1+\frac{2x}{p}\right)} + ln(\sqrt{\frac{2x}{p}} + \sqrt{1+\frac{2x}{p}})]$

弓形面积：设M，D是抛物线上两点，且分居一、四象限。作一条平行于MD且与抛物线相切的直线L。若M到L的距离为h。则有 $S_{MOD} = \frac{2}{3}MD\cdot h$

**重心**

半径为 r、圆心角为θ的扇形的重心与圆心的距离为(4rsin (θ/2))/3θ

半径为 r、圆心角为θ的圆弧的重心与圆心的距离为(4rsin^3 (θ/2))/(3(θ − sinθ))

椭圆上半部分的重心与圆心的距离为 (4/3π) b

抛物线中弓形 MOD 的重心满足 CQ = (2/5) PQ，P 是直线 L 与抛物线的切点，Q 在 MD 上且 PQ 平行 x 轴。C 是重心。

**内心** r = 三角形面积/(p = 1/2(a + b + c)) I = (aA + bB + cC)/(a + b + c)

**三重积公式** $a\times(b\times c) = b(a\cdot c) - c(a\cdot b)$

**额外的公式**

*四边形*: D1,D2 为对角线,M 对角线中点连线,A 为对角线夹角

1.a^2+b^2+c^2+d^2=D1^2+D2^2+4M^2 2. S=D1D2sin(A)/2

(以下对圆的内接四边形)

3. ac+bd=D1D2 4.S=sqrt((P-a)(P-b)(P-c)(P-d)),P 为半周长

*正 n 边形*:R 为外接圆半径,r 为内切圆半径

1. 中心角 A=2PI/n 2. 内角 C=(n-2)PI/n
3. 边长 a=2sqrt(R^2-r^2)=2Rsin(A/2)=2rtan(A/2)
4. 面积 S=nar/2=nr^2tan(A/2)=nR^2sin(A)/2=na^2/(4tan(A/2))

*圆*: 1. 弧长 l=rA 2. 弦长 a=2sqrt(2hr-h^2)=2rsin(A/2)

3. 弓形高 h=r-sqrt(r^2-a^2/4)=r(1-cos(A/2))=atan(A/4)/2
4.扇形面积 S1=rl/2=r^2A/2
5.弓形面积 S2=(rl-a(r-h))/2=r^2(A-sin(A))/2

*棱柱*: 1. 体积 V=Ah,A 为底面积,h 为高

2. 侧面积 S=lp,l 为棱长,p 为直截面周长 3. 全面积 T=S+2A

*棱锥*: 1.体积 V=Ah/3,A 为底面积,h 为高 (以下对正棱锥)

2. 侧面积 S=lp/2,l 为斜高,p 为底面周长 3. 全面积 T=S+A

*棱台*:1. 体积 V=(A1+A2+sqrt(A1A2))h/3,A1.A2 为上下底面积,h 为高

(以下为正棱台)

2. 侧面积 S=(p1+p2)l/2,p1.p2 为上下底面周长,l 为斜高
3. 全面积 T=S+A1+A2

**算法**

**树的计数**

**有根树的计数**

令 $S_{n,j} = \sum_{1\le i\le n/j} a_{n+1-ij} = S_{n-j,j} + a_{n+1-j}$

于是，n+1 个结点的有根树的总数为 $a_{n+1} = \frac{\sum_{1\le j\le n} ja_jS_{n,j}}{n}$

附：$a_1 = 1, a_2 = 1, a_3 = 2, a_4 = 4, a_5 = 9, a_6 = 20, a_9 = 286, a_{11} = 1842$

**无根树的计数**

当 n 是奇数时，则有 $a_n - \sum_{1\le i\le n/2} a_i a_{n-i}$ 种不同的无根树。

当 n 是偶数时，则有这么多种不同的无根树。

$$a_n - \sum_{1\le i\le \frac{n}{2}} a_i a_{n-i} + \frac{1}{2} a_{n/2}(a_{n/2}+1)$$

## 代数

**Burnside引理** $\text{ans} = \frac{(\sum \text{每种置换下的不变的元素个数})}{\text{置换群中置换的个数}}$

**三次方程求根公式** $x^3 + px + q = 0$

$$x_j = \omega^j \sqrt[3]{-\frac{q}{2} + \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}} + \omega^{2j} \sqrt[3]{-\frac{q}{2} - \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}}$$

其中 j=0, 1, 2, $\omega = (-1 + i\sqrt{3})/2$

当求解$ax^3 + bx^2 + cx + d = 0$时， 令 $x = y - b/3a$ 再求解y，即转化成$x^3 + px + q = 0$的形式

**组合公式**

$\sum_{k=1}^{n}(2k-1)^2 = \frac{n(4n^2-1)}{3}$　　$\sum_{k=1}^{n} k^3 = \left(\frac{n(n+1)}{2}\right)^2$

$\sum_{k=1}^{n}(2k-1)^3 = n^2(2n^2-1)$　　$\sum_{k=1}^{n} k^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$

$\sum_{k=1}^{n} k^5 = \frac{n^2(n+1)^2(2n^2+2n-1)}{12}$　　$\sum_{k=1}^{n} k(k+1) = \frac{n(n+1)(n+2)}{3}$

$\sum_{k=1}^{n} k(k+1)(k+2) = \frac{n(n+1)(n+2)(n+3)}{4}$

$\sum_{k=1}^{n} k(k+1)(k+2)(k+3) = \frac{n(n+1)(n+2)(n+3)(n+4)}{5}$

错排：$D_n = n!\,(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \cdots + \frac{(-1)^n}{n!}) = (n-1)(D_{n-2} - D_{n-1})$

## 三角公式

$\sin(\alpha \pm \beta) = \sin\alpha \cos\beta \pm \cos\alpha \sin\beta$　　$\cos(\alpha \pm \beta) = \cos\alpha \cos\beta \mp \sin\alpha \sin\beta$

$\tan(\alpha \pm \beta) = \frac{\tan(\alpha) \pm \tan(\beta)}{1 \mp \tan(\alpha)\tan(\beta)}$　　$\tan(\alpha) \pm \tan(\beta) = \frac{\sin(\alpha \pm \beta)}{\cos(\alpha)\cos(\beta)}$

$\sin(\alpha) + \sin(\beta) = 2\sin\frac{(\alpha+\beta)}{2}\cos\frac{(\alpha-\beta)}{2}$　　$\sin(\alpha) - \sin(\beta) = 2\cos\frac{(\alpha+\beta)}{2}\sin\frac{(\alpha-\beta)}{2}$

$\cos(\alpha) + \cos(\beta) = 2\cos\frac{(\alpha+\beta)}{2}\cos\frac{(\alpha-\beta)}{2}$　　$\cos(\alpha) - \cos(\beta) = -2\sin\frac{(\alpha+\beta)}{2}\sin\frac{(\alpha-\beta)}{2}$

$$\sin(n\alpha) = n\cos^{n-1}\alpha\,\sin\alpha - \binom{n}{3}\cos^{n-3}\alpha\,\sin^3\alpha + \binom{n}{5}\cos^{n-5}\alpha\,\sin^5\alpha - \cdots$$

$$\cos(n\alpha) = \cos^n\alpha - \binom{n}{2}\cos^{n-2}\alpha\,\sin^2\alpha + \binom{n}{4}\cos^{n-4}\alpha\,\sin^4\alpha - \cdots$$

## 积分表

| | | |
|---|---|---|
| $(\arcsin x)' = \frac{1}{\sqrt{1-x^2}}$ | $(\arccos x)' = -\frac{1}{\sqrt{1-x^2}}$ | $(\arctan x)' = \frac{1}{1+x^2}$ |
| $a^x \to a^x/\ln a$ | $\sin x \to -\cos x$ | $\cos x \to \sin x$ |
| $\tan x \to -\ln\cos x$ | $\sec x \to \ln\tan(x/2 + \pi/4)$ | $\tan^2 x \to \tan x - x$ |
| $\csc x \to \ln\tan\frac{x}{2}$ | $\sin^2 x \to \frac{x}{2} - \frac{1}{2}\sin x\cos x$ | $\cos^2 x \to \frac{x}{2} + \frac{1}{2}\sin x\cos x$ |
| $\sec^2 x \to \tan x$ | $\frac{1}{\sqrt{a^2-x^2}} \to \arcsin\left(\frac{x}{a}\right)$ | $\csc^2 x \to -\cot x$ |

| | |
|---|---|
| $\frac{1}{a^2-x^2}(\lvert x\rvert < \lvert a\rvert) \to \frac{1}{2a}\ln\frac{(a+x)}{a-x}$ | $\frac{1}{x^2-a^2}(\lvert x\rvert > \lvert a\rvert) \to \frac{1}{2a}\ln\frac{(x-a)}{x+a}$ |
| $\sqrt{a^2-x^2} \to \frac{x}{2}\sqrt{a^2-x^2} + \frac{a^2}{2}\arcsin\frac{x}{a}$ | $\frac{1}{\sqrt{x^2+a^2}} \to \ln\left(x + \sqrt{a^2+x^2}\right)$ |
| $\sqrt{a^2+x^2} \to \frac{x}{2}\sqrt{a^2+x^2} + \frac{a^2}{2}\ln\left(x + \sqrt{a^2+x^2}\right)$ | $\frac{1}{\sqrt{x^2-a^2}} \to \ln\left(x + \sqrt{x^2-a^2}\right)$ |
| $\sqrt{x^2-a^2} \to \frac{x}{2}\sqrt{x^2-a^2} - \frac{a^2}{2}\ln\left(x + \sqrt{x^2-a^2}\right)$ | $\frac{1}{x\sqrt{a^2-x^2}} \to -\frac{1}{a}\ln\frac{a+\sqrt{a^2-x^2}}{x}$ |
| $\frac{1}{x\sqrt{x^2-a^2}} \to \frac{1}{a}\arccos\frac{a}{x}$ | $\frac{1}{x\sqrt{a^2+x^2}} \to -\frac{1}{a}\ln\frac{a+\sqrt{a^2+x^2}}{x}$ |

| | |
|---|---|
| $\frac{1}{\sqrt{2ax-x^2}} \to \arccos(1-\frac{x}{a})$ | $\frac{x}{ax+b} \to \frac{x}{a}-\frac{b}{a^2}\ln(ax+b)$ |
| $\sqrt{2ax-x^2} \to \frac{x-a}{2}\sqrt{2ax-x^2}+\frac{a^2}{2}\arcsin(\frac{x}{a}-1)$ | |
| $\frac{1}{x\sqrt{ax+b}}(b<0) \to \frac{2}{\sqrt{-b}}\arctan\sqrt{\frac{ax+b}{-b}}$ | $x\sqrt{ax+b} \to \frac{2(3ax-2b)}{15a^2}(ax+b)^{\frac{3}{2}}$ |
| $\frac{1}{x\sqrt{ax+b}}(b>0) \to \frac{1}{\sqrt{-b}}ln\frac{\sqrt{ax+b}-\sqrt{b}}{\sqrt{ax+b}+\sqrt{b}}$ | $\frac{x}{\sqrt{ax+b}} \to \frac{2(ax-2b)}{3a^2}\sqrt{ax+b}$ |
| $\frac{1}{x^2\sqrt{ax+b}} \to -\frac{\sqrt{ax+b}}{bx}-\frac{a}{2b}\int\frac{dx}{x\sqrt{ax+b}}$ | $\frac{\sqrt{ax+b}}{x} \to 2\sqrt{ax+b}+\mathrm{b}\int\frac{dx}{x\sqrt{ax+b}}$ |
| $\frac{1}{\sqrt{(ax+b)^n}}(n>2) \to \frac{-2}{a(n-2)}\cdot\frac{1}{\sqrt{(ax+b)^{n-2}}}$ | |
| $\frac{1}{ax^2+c}(a>0,c>0) \to \frac{1}{\sqrt{ac}}\arctan(x\sqrt{\frac{a}{c}})$ | $\frac{x}{ax^2+c} \to \frac{1}{2a}\ln(ax^2+c)$ |
| $\frac{1}{ax^2+c}(a+,c-) \to \frac{1}{2\sqrt{-ac}}ln\frac{x\sqrt{a}-\sqrt{-c}}{x\sqrt{a}+\sqrt{-c}}$ | $\frac{1}{x(ax^2+c)} \to \frac{1}{2c}\ln\frac{x^2}{ax^2+c}$ |
| $\frac{1}{ax^2+c}(a-,c+) \to \frac{1}{2\sqrt{-ac}}ln\frac{\sqrt{c}+x\sqrt{-a}}{\sqrt{c}-x\sqrt{-a}}$ | $x\sqrt{ax^2+c} \to \frac{1}{3a}\sqrt{(ax^2+c)^3}$ |
| $\frac{1}{(ax^2+c)^n}(n>1) \to \frac{x}{2c(n-1)(ax^2+c)^{n-1}}+\frac{2n-3}{2c(n-1)}\int\frac{dx}{(ax^2+c)^{n-1}}$ | |
| $\frac{x^n}{ax^2+c}(n\neq1) \to \frac{x^{n-1}}{a(n-1)}-\frac{c}{a}\int\frac{x^{n-2}}{ax^2+c}dx$ | $\frac{1}{x^2(ax^2+c)} \to \frac{-1}{cx}-\frac{a}{c}\int\frac{dx}{ax^2+c}$ |
| $\frac{1}{x^2(ax^2+c)^n}(n\geq2) \to \frac{1}{c}\int\frac{dx}{x^2(ax^2+c)^{n-1}}-\frac{a}{c}\int\frac{dx}{(ax^2+c)^n}$ | |
| $\sqrt{ax^2+c}(a>0) \to \frac{x}{2}\sqrt{ax^2+c}+\frac{c}{2\sqrt{a}}\ln\left(x\sqrt{a}+\sqrt{ax^2+c}\right)$ | |
| $\sqrt{ax^2+c}(a<0) \to \frac{x}{2}\sqrt{ax^2+c}+\frac{c}{2\sqrt{-a}}\arcsin\left(x\sqrt{\frac{-a}{c}}\right)$ | $\frac{1}{\sqrt{ax^2+c}}(a<0) \to \frac{1}{\sqrt{-a}}\arcsin\left(x\sqrt{-\frac{a}{c}}\right)$ |
| $\frac{1}{\sqrt{ax^2+c}}(a>0) \to \frac{1}{\sqrt{a}}\ln\left(x\sqrt{a}+\sqrt{ax^2+c}\right)$ | |

| | | |
|---|---|---|
| $\sin^2 ax \to \frac{x}{2}-\frac{1}{4a}\sin 2ax$ | $\cos^2 ax \to \frac{x}{2}+\frac{1}{4a}\sin 2ax$ | $\frac{1}{\sin ax} \to \frac{1}{a}\ln\tan\frac{ax}{2}$ |
| $\frac{1}{\cos^2 ax} \to \frac{1}{a}\tan ax$ | $\frac{1}{\cos ax} \to \frac{1}{a}\ln\tan\left(\frac{\pi}{4}+\frac{ax}{2}\right)$ | $\ln(ax) \to x\ln(ax)-x$ |
| $\sin^3 ax \to \frac{-1}{a}\cos ax+\frac{1}{3a}\cos^3 ax$ | $\cos^3 ax \to \frac{1}{a}\sin ax-\frac{1}{3a}\sin^3 ax$ | |
| $\frac{1}{\sin^2 ax} \to -\frac{1}{a}\cot ax$ | $x\ln(ax) \to \frac{x^2}{2}\ln(ax)-\frac{x^2}{4}$ | $\cos ax \to \frac{1}{a}\sin ax$ |
| $x^2e^{ax} \to \frac{e^{ax}}{a^3}(a^2x^2-2ax+2)$ | $(\ln(ax))^2 \to x(\ln(ax))^2-2x\ln(ax)+2x$ | |
| $x^2\ln(ax) \to \frac{x^3}{3}\ln(ax)-\frac{x^3}{9}$ | $x^n\ln(ax) \to \frac{x^{n+1}}{n+1}\ln(ax)-\frac{x^{n+1}}{(n+1)^2}$ | |
| $\sin(\ln ax) \to \frac{x}{2}[\sin(\ln ax)-\cos(\ln ax)]$ | $\cos(\ln ax) \to \frac{x}{2}[\sin(\ln ax)+\cos(\ln ax)]$ | |