/\*\* 方法一  
\* 判断一个点是否在一个多边形范围内  
\* [@param](http://tieba.baidu.com/home/main?un=param&fr=pb" \t "_blank) point 点的坐标  
\* @param points 组成多边形的坐标点  
\* @return  
\*/  
public static boolean isPointInPolygon2(LatLng point, LatLng[] points) {  
boolean re = false;  
int nCross = 0;  
for (int i = 0; i < points.length; i++) {  
LatLng p1 = points[i];  
LatLng p2 = points[(i + 1) % points.length];  
if (p1.latitude == p2.latitude) {  
continue;  
}  
  
  
double min = (p1.latitude > p2.latitude) ? p2.latitude  
: p1.latitude;  
double max = (p1.latitude > p2.latitude) ? p1.latitude  
: p2.latitude;  
  
  
if (point.latitude < min) {  
continue;  
}  
if (point.latitude >= max) {  
continue;  
}  
// 求交点的x 坐标  
double x = (double) (point.latitude - p1.latitude)  
\* (double) (p2.longitude - p1.longitude)  
/ (double) (p2.latitude - p1.latitude) + p1.longitude;  
if (x > point.longitude) {  
nCross++; // 只统计单边交点  
}  
  
  
re = nCross % 2 == 1;  
}  
  
  
return re;  
}

/\*方法二 \*/

public static int isIntersect(ArrayList<GeoPoint> ps, GeoPoint p) {  
int num = 0;  
ps.add(ps.get(0));  
// 按顺序判断每条边是否与射线有交点  
for (int i = 0; i < ps.size() - 1; i++) {  
if (((ps.get(i).getLongitudeE6()/1e6 - p.getLongitudeE6()/1e6) \* (ps.get(i + 1).getLongitudeE6()/1e6 - p.getLongitudeE6()/1e6)) > 0) { } else {  
double a = (ps.get(i).getLongitudeE6()/1e6 - ps.get(i + 1).getLongitudeE6()/1e6)  
/ (ps.get(i).getLatitudeE6()/1e6 - ps.get(i + 1).getLatitudeE6()/1e6);  
double b = ps.get(i).getLongitudeE6()/1e6 - a \* ps.get(i).getLatitudeE6()/1e6;  
if (((p.getLongitudeE6()/1e6 - b) / a) > (p.getLatitudeE6()/1e6)) { } else  
num++;  
}  
}  
return num;  
}

/\* 方法三\*/

//    在范围内返回1，不在返回0

-(int)mutableBoundConrtolAction:(NSMutableArray \*)arrSome:(CLLocationCoordinate2D )myCoordinate4{

    int n=arrSome.count;

    float vertx[n];

    float verty[n];

    for (int i=0; i<arrSome.count; i++) {

//MyPoint类存储的是经度和纬度

        vertx[i]=((MyPoint \*)(arrSome[i])).x;

        verty[i]=((MyPoint \*)(arrSome[i])).y;

    }

    if (arrSome.count==0) {

        return 1;

    }

    BOOL i=pnpoly(arrSome.count, vertx, verty, myCoordinate4.latitude, myCoordinate4.longitude);

    if (i) {

        return 1;

    }else{

        return 0;

    }

    return 1;

}

//多边形由边界的坐标点所构成的数组组成，参数格式 该数组的count，  多边形边界点x坐标 的组成的数组，多边形边界点y坐标 的组成的数组，需要判断的点的x坐标，需要判断的点的y坐标

BOOL pnpoly (int nvert, float \*vertx, float \*verty, float testx, float testy) {

    int i, j;

    BOOL c=NO;

    for (i = 0, j = nvert-1; i < nvert; j = i++) {

        if ( ( (verty[i]>testy) != (verty[j]>testy) ) &&

            (testx < (vertx[j]-vertx[i]) \* (testy-verty[i]) / (verty[j]-verty[i]) + vertx[i]) )

            c = !c;

    }

    return c;

}