第一题：

*/\**

*\* @Description:*

*\* @version:*

*\* @Author: LL Leung*

*\* @Date: 2019-12-01 21:03:02*

*\* @LastEditors: LL Leung*

*\* @LastEditTime: 2019-12-02 19:06:50*

*\*/*

static int overlapSegSeg2d(const float \*a, const float \*b, const float \*c, const float \*d){

    const float a1 = vcross2(a, b, d);

    const float a2 = vcross2(a, b, c);

    if (a1 \* a2 < 0.0f){

        float a3 = vcross2(c, d, a);

        float a4 = a3 + a2 - a1;

        if (a3 \* a4 < 0.0f)

            return 1;

    }

    return 0;

}

bool dtIntersectSegSeg2D(const float \*ap, const float \*aq, const float \*bp, const float \*bq, float &s, float &t){

    float u[3], v[3], w[3];

    dtVsub(u, aq, ap);

    dtVsub(v, bq, bp);

    dtVsub(w, ap, bp);

    float d = vperpXZ(u, v);

    if (fabsf(d) < 1e-6f)

        return false;

    s = vperpXZ(v, w) / d;

    t = vperpXZ(u, w) / d;

    return true;

}

*//凸多边形*

bool dtIntersectSegmentPoly2D(const float \*p0, const float \*p1, const float \*verts, int nverts, float &tmin, float &tmax,

                              int &segMin, int &segMax){

    static const float EPS = 0.00000001f;

    tmin = 0;

    tmax = 1;

    segMin = -1;

    segMax = -1;

    float dir[3];

    dtVsub(dir, p1, p0);

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

        float edge[3], diff[3];

        dtVsub(edge, &verts[i \* 3], &verts[j \* 3]);

        dtVsub(diff, p0, &verts[j \* 3]);

*//面积*

        const float n = dtVperp2D(edge, diff);

*// s和edge平行*

        const float d = dtVperp2D(dir, edge);

        if (fabsf(d) < EPS){

*//p0在edgej->i的左边*

            if (n < 0)

                return false;

            else

                continue;

        }

*// 面积之比 即 高之比*

        const float t = n / d;

*// n > 0 说明 i 在 dir 的 左边*

        if (d < 0){

*// 更新的是 tmin 应该取最大值，因为是靠近，所以交点肯定是越远离起点越好 越远离起点 t值越大*

            if (t > tmin){

                tmin = t;

                segMin = j;

                if (tmin > tmax)

                    return false;

            }

        }else{

            if (t < tmax){

                tmax = t;

                segMax = j;

                if (tmax < tmin)

                    return false;

            }

        }

    }

    return true;

}

第二题：

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

    int i, j, c = 0;

    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;

}

publicbool IsInside(PointLatLng p){

    int count = Points.Count;

    if (count < 3){

        return false;

    }

    bool result = false;

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

        var p1 = Points[i];

        var p2 = Points[j];

        if (p1.Lat < p.Lat && p2.Lat >= p.Lat || p2.Lat < p.Lat && p1.Lat >= p.Lat){

            if (p1.Lng + (p.Lat - p1.Lat) / (p2.Lat - p1.Lat) \* (p2.Lng - p1.Lng) < p.Lng){

                result = !result;

            }

        }

        j = i;

    }

    return result;

}

private static double SignedPolygonArea(List<PointLatLng> points){

    int pointsCount = points.Count;

    PointLatLng[] pts = new PointLatLng[pointsCount + 1];

    points.CopyTo(pts, 0);

    pts[pointsCount] = points[0];

    for (int i = 0; i < pointsCount + 1; ++i){

        pts[i].Lat = pts[i].Lat \* (System.Math.PI \* 6378137 / 180);

        pts[i].Lng = pts[i].Lng \* (System.Math.PI \* 6378137 / 180);

    }

    double area = 0;

    for (int i = 0; i < pointsCount; i++){

        area += (pts[i + 1].Lat - pts[i].Lat) \* (pts[i + 1].Lng + pts[i].Lng) / 2;

    }

    return area;

}

public static double GetPolygonArea(List<PointLatLng> points){

    return Math.Abs(SignedPolygonArea(points));

}