点是否完全在矩形的左侧、右侧、上侧或下侧，如果都不是，就一定在其中。如果矩形的边不平行于坐标轴呢？也非常简单，就是高中数学的知识，通过坐标变换把矩阵转成平行的情况，在旋转时所有的点跟着转动就可以。旋转完成后，再用上面的方式进行判断。具体请参看如下代码中的 isInside 方法。

```
public boolean isInside(double x1, double y1, double x2, double y2,
                double x3, double y3, double x4, double y4, double x, double y)
{
        if (y1 == y2) {
                return isInside(x1, y1, x4, y4, x, y);
        }
        double l = Math.abs(y4 - y3);
        double k = Math.abs(x4 - x3);
        double s = Math.sqrt(k * k + l * l);
        double sin = l / s;
        double cos = k / s;
        double x1R = cos * x1 + sin * y1;
        double y1R = -x1 * sin + y1 * cos;
        double x4R = cos * x4 + sin * y4;
        double y4R = -x4 * sin + y4 * cos;
        double xR = cos * x + sin * y;
        double yR = -x * sin + y * cos;
        return isInside(x1R, y1R, x4R, y4R, xR, yR);
}
```

判断一个点是否在三角形内部

【题目】

在二维坐标系中，所有的值都是 double 类型，那么一个三角形可以由 3 个点来代表，给定 3 个点代表的三角形，再给定一个点(x,y)，判断(x,y)是否在三角形中。

【难度】

尉 ★★☆☆

【解答】

本书提供两种解法，第一种解法是从面积的角度来解决这道题，第二种解法是从向量的角度来解决。解法一在逻辑上没有问题，但是没有解法二好，下面会给出详细的解释。

先来介绍解法一，如果点 O 在三角形 ABC 内部，如图 9-1 所示，那么，有面积 ABC=

面积 *ABO*+面积 *BCO*+面积 *CAO*。如果点 *O* 在三角形 *ABC* 外部，如图 9-2 所示，那么，有面积 *ABC*<面积 *ABO*+面积 *BCO*+面积 *CAO*。既然得知了这样一种评判标准，实现代码就变得很简单了。首先实现求两个点(*x*1,*y*1)和(*x*2,*y*2)之间距离的函数，具体请参看如下代码中的 getSideLength 方法。

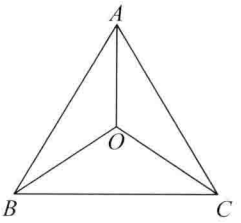

图 9-1

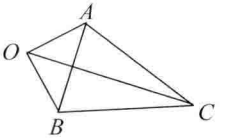

图 9-2

```
public double getSideLength(double x1, double y1, double x2, double y2) {
        double a = Math.abs(x1 - x2);
        double b = Math.abs(y1 - y2);
        return Math.sqrt(a * a + b * b);
}
```

有了如上函数后，就可以求出一条边的边长。下面根据边长来求三角形的面积，用海伦公式来求解三角形面积是非常合适的，具体请参看如下代码中的 getArea 方法。

```
public double getArea(double x1, double y1, double x2, double y2,
                double x3, double y3) {
        double side1Len = getSideLength(x1, y1, x2, y2);
        double side2Len = getSideLength(x1, y1, x3, y3);
        double side3Len = getSideLength(x2, y2, x3, y3);
        double p = (side1Len + side2Len + side3Len) / 2;
        return Math.sqrt((p - side1Len) * (p - side2Len) * (p - side3Len) * p);
}
```

最后就可以根据我们的标准来求解，具体请参看如下代码中的 isInside1 方法。

```
public boolean isInside1(double x1, double y1, double x2, double y2,
                double x3, double y3, double x, double y) {
        double area1 = getArea(x1, y1, x2, y2, x, y);
        double area2 = getArea(x1, y1, x3, y3, x, y);
        double area3 = getArea(x2, y2, x3, y3, x, y);
        double allArea = getArea(x1, y1, x2, y2, x3, y3);
        return area1 + area2 + area3 <= allArea;
}
```

虽然解法一的逻辑是正确的，但 double 类型的值在计算时会出现一定程度的偏差。所

以经常会发生明明 *O* 点在三角形内，但是面积却对不准的情况出现，最后导致判断出错。所以解法一并不推荐。

解法二使用了和解法一完全不同的标准，而且几乎不会受精度损耗的影响。如果点 *O* 在三角形 *ABC* 内部，除面积上的关系外，还有其他关系存在，如图 9-3 所示。

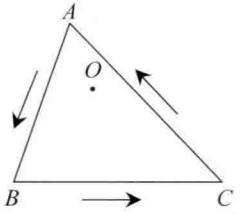

图 9-3

如果点 *O* 在三角形 *ABC* 中，那么如果从三角形的一点出发，逆时针走过所有边的过程中，点 *O* 始终都在走过边的左侧。比如，图 9-3 中，*O* 都在 *AB*、*BC* 和 *CA* 的左侧。如果点 *O* 在三角形 *ABC* 外部，则不满足这个关系。

新的标准有了，接下来解决一个棘手的问题。我们知道作为参数传入的三个点的坐标代表一个三角形，可是这三个点依次的顺序不一定是逆时针的。比如，如果参数的顺序为 *A* 坐标、*B* 坐标和 *C* 坐标，那就没问题，因为这是逆时针的。但如果参数的顺序为 *C* 坐标、*B* 坐标和 *A* 坐标，就有问题，因为这是顺时针的。作为程序的实现者，要求用户按你规定的顺序传入三角形的三个点坐标，这明显是不合适的。所以需要自己来解决这个问题。假设得到的坐标依次为点 1、点 2、点 3。顺序可能是顺时针，也可能是逆时针，如图 9-4 所示。

如果点 2 在 1->3 边的右边，此时按照点 1、点 2 和点 3 的顺序没有问题，这个顺序本来就是逆时针的。但如果如图 9-5 所示，如果点 2 在 1->3 边的左边，那么按照点 1、点 2 和点 3 的顺序就有问题，因为这个顺序是顺时针的，所以应该按照点 1、点 3 和点 2 的顺序。

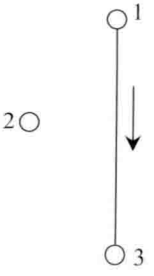

图 9-4

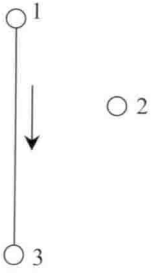

图 9-5

如何判断一个点在一条有向边的左边还是右边？这个利用几何上向量积（叉积）的求解公式即可。如果有向边 1->2 叉乘有向边 1->3 的结果为正，说明 2 在有向边 1->3 的左边，比如图 9-4；如果有向边 1->2 叉乘有向边 1->3 的结果为负，说明 2 在有向边 1->3 的右边，比如图 9-5。

具体过程请参看如下代码中的 crossProduct 方法，该方法描述了向量($x1$,$y1$)叉乘向量($x2$,$y2$)，两个向量的开始点都是原点。

```
public double crossProduct(double x1, double y1, double x2, double y2) {
        return x1 * y2 - x2 * y1;
}
```

至此，我们已经解释了解法二的所有细节，全部过程请参看如下代码中的 isInside2 方法。

```
public boolean isInside2(double x1, double y1, double x2, double y2,
                double x3, double y3, double x, double y) {
        // 如果三角形的点不是逆时针输入，改变一下顺序
        if (crossProduct(x3 - x1, y3 - y1, x2 - x1, y2 - y1) >= 0) {
                double tmpx = x2;
                double tmpy = y2;
                x2 = x3;
                y2 = y3;
                x3 = tmpx;
                y3 = tmpy;
        }
        if (crossProduct(x2 - x1, y2 - y1, x - x1, y - y1) < 0) {
                return false;
        }
        if (crossProduct(x3 - x2, y3 - y2, x - x2, y - y2) < 0) {
                return false;
        }
        if (crossProduct(x1 - x3, y1 - y3, x - x3, y - y3) < 0) {
                return false;
        }
        return true;
}
```

折纸问题

【题目】

请把一段纸条竖着放在桌子上，然后从纸条的下边向上方对折 1 次，压出折痕后展开。此时折痕是凹下去的，即折痕突起的方向指向纸条的背面。如果从纸条的下边向上方连续