

# 计算几何 2：旋转卡壳、三角形、圆

Ebola

Institute of Mathematics,  
Zhejiang University.

Jan, 2024

## ① 基础回顾

## ② 旋转卡壳

## ③ 圆

# ① 基础回顾

## ② 旋转卡壳

## ③ 圆

## P2116 城墙

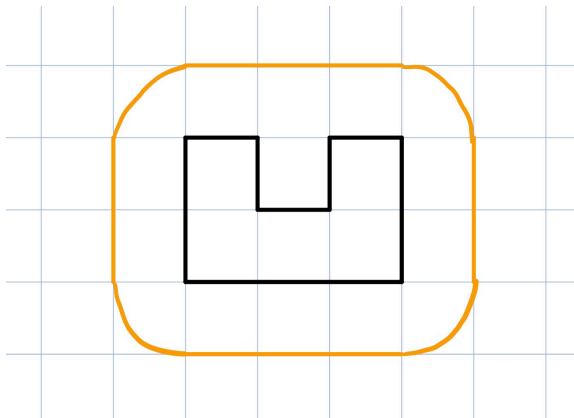
有一次，一个贪婪的国王命令他的骑士在他的城堡外修建一堵围墙，要求围墙离城堡的最近距离不能少于  $L$ 。

城堡是一个  $n$  边形，国王非常吝啬，不愿意多花建一米的围墙，多建的话他会杀掉负责修建的骑士。

请你帮助这个倒霉的骑士，帮他求出最少需要修建多长的围墙。

## P2116 城墙

答案是凸包周长  $+2\pi L$ .

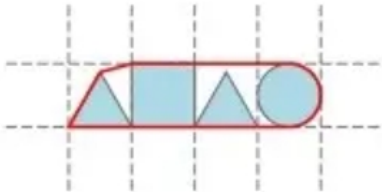


[CERC2016] 凸轮廓线

一些几何图形整齐地在一个网格图上从左往右排成一行。它们占据了连续的一段横行，每个位置恰好一个几何图形。每个图形是以下的三种之一：

- ① 一个恰好充满单个格子的正方形。
- ② 一个内切于单个格子的圆。
- ③ 一个底边与格子重合的等边三角形。

已知每个格子的边长都为 1，请求出这些几何图形的凸包的周长。



## [CERC2016] 凸轮廓线

没什么好说的，非常非常复杂的分类讨论，极有可能漏掉一些情况！

## [ECNA2017] Craters (圆的凸包)

给定平面上  $n$  ( $\leq 200$ ) 个圆的圆心坐标、半径，求能够围住所有圆的最短曲线的长度，精确到  $10^{-6}$ 。



## [ECNA2017] Craters (圆的凸包)

在每个圆上均匀地取  $T$  个顶点，求这些顶点构成的凸包周长。

$T$  需要充分大。实测取 2000 即可（比赛时尽量取大，反正 5000 也不会炸）

## [ECNA2017] Craters (圆的凸包)

在每个圆上均匀地取  $T$  个顶点，求这些顶点构成的凸包周长。

$T$  需要充分大。实测取 2000 即可（比赛时尽量取大，反正 5000 也不会炸）

一个有意思的结论： $T$  每增大一倍，求得的周长的误差（即与精确周长之差的绝对值）缩小到原来的四分之一。（线性插值的二阶收敛性质）

# 向量的旋转

如果一个向量  $\vec{v} = (x, y)$ , 现在将它逆时针旋转  $90^\circ$ , 得到什么?

# 向量的旋转

如果一个向量  $\vec{v} = (x, y)$ , 现在将它逆时针旋转  $90^\circ$ , 得到什么?

$$\vec{v}^\perp = (-y, x). \quad (1)$$

# 向量的旋转

如果一个向量  $\vec{v} = (x, y)$ , 现在将它逆时针旋转  $\theta$ , 得到什么?

# 向量的旋转

如果一个向量  $\vec{v} = (x, y)$ , 现在将它逆时针旋转  $\theta$ , 得到什么?

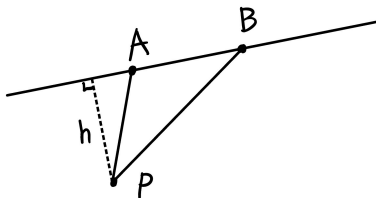
$$\text{Rotate}(\vec{v}, \theta) = (x \cos \theta - y \sin \theta, x \sin \theta + y \cos \theta). \quad (2)$$

# 点到直线的距离

给定  $P, A, B$  三点坐标, 求点  $P$  到直线  $AB$  的距离。(别用斜率)

# 点到直线的距离

给定  $P, A, B$  三点坐标, 求点  $P$  到直线  $AB$  的距离。(别用斜率)

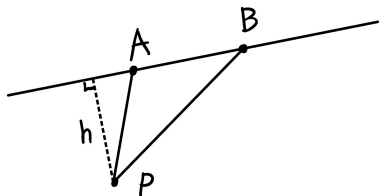


$$h = \frac{S_{\triangle PAB}}{|AB|} = \frac{|\vec{PA} \times \vec{PB}|}{|AB|}. \quad (3)$$



# 点到直线的距离

给定  $P, A, B$  三点坐标, 求点  $P$  到直线  $AB$  的距离。(别用斜率)



$$h = \frac{S_{\triangle PAB}}{|AB|} = \frac{|\overrightarrow{PA} \times \overrightarrow{PB}|}{|AB|}. \tag{3}$$

```
1 double DistanceToLine(Point P, Point A, Point B){
2     return fabs(Cross(A-P, B-P)) / Length(A-B);
3 }
```

## ① 基础回顾

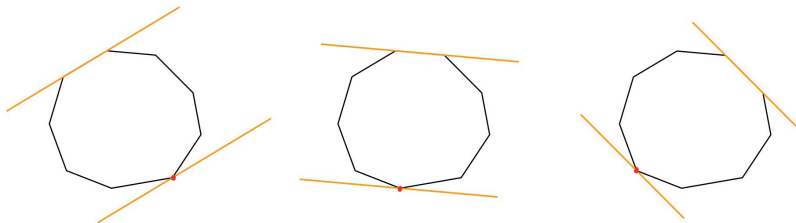
## ② 旋转卡壳

## ③ 圆

# 【模板】旋转卡壳

给定平面上  $n$  个点，求凸包直径（直径是指最远的两个顶点的距离）。

# 旋转卡壳



如图所示，枚举上凸壳的边  $P_iP_{i+1}$ ，找到距离它最远的点。显然，当我们顺时针枚举边时，对面的点也只会顺时针方向前进，所以复杂度是  $O(n)$  的。

# 旋转卡壳

```

1 double RotatingCalipers(Point *ch, int n){
2     if(n==2) return Length(ch[2] - ch[1]);
3     int cur=0;
4     double ans=0;
5     ch[n+1] = ch[1];
6     for(int i = 1; i <= n; i++){
7         while(DistanceToLine(ch[cur], ch[i], ch[i+1])
8             <= DistanceToLine(ch[cur%n+1], ch[i], ch[i+1])){
9             cur = cur % n + 1;
10        }
11        ans=max(ans, max(Length(ch[i] - ch[cur]),
12                        Length(ch[i+1] - poly[cur])));
13    }
14    return ans;
15 }
```

# 旋转卡壳

其实由于  $\triangle P_i P_{i+1} Q$  的底边  $P_i P_{i+1}$  是固定的, 最大化  $Q$  到直线  $l_{P_i P_{i+1}}$  的距离就等价于最大化  $\triangle P_i P_{i+1} Q$  的面积, 因此不要求点到直线的距离, 直接调用叉乘即可, 可以优化常数。

```

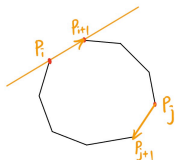
1 double RotatingCalipers(Point *ch, int n){
2     if(n==2) return Length(ch[2] - ch[1]);
3     int cur=0;
4     double ans=0;
5     ch[n+1] = ch[1];
6     for(int i = 1; i <= n; i++){
7         while(fabs(Cross(ch[i]-ch[cur], ch[i+1]-ch[cur]))
8             <= fabs(Cross(ch[i]-ch[cur%n+1], ch[i+1]-ch[cur%n+1]))) {
9             cur = cur % n + 1;
10        }
11        ans=max(ans, max(Length(ch[i] - ch[cur]),
12            Length(ch[i+1] - poly[cur])));
13    }
14    return ans;
15 }
```

# 旋转卡壳

还可以进一步优化常数，把两次叉乘简化成一次，

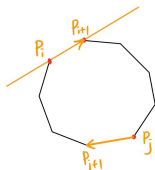
# 旋转卡壳

还可以进一步优化常数，把两次叉乘简化成一次，只需要注意到：



$$\overrightarrow{P_i P_{i+1}} \times \overrightarrow{P_j P_{j+1}} < 0$$

$$\Rightarrow j++$$



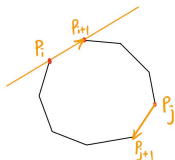
$$\overrightarrow{P_i P_{i+1}} \times \overrightarrow{P_j P_{j+1}} > 0$$

$$\Rightarrow \text{停止}$$



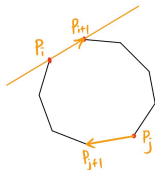
# 旋转卡壳

还可以进一步优化常数，把两次叉乘简化成一次，只需要注意到：



$$\overrightarrow{P_i P_{i+1}} \times \overrightarrow{P_j P_{j+1}} < 0$$

$\Rightarrow j++$



$$\overrightarrow{P_i P_{i+1}} \times \overrightarrow{P_j P_{j+1}} > 0$$

$\Rightarrow$  停止

1

```
while( Cross(ch[i+1]-ch[i], ch[cur%n+1]-ch[cur]) < 0 )
```

## [HNOI2007] 最小矩形覆盖

给定一些点的坐标，求能够覆盖所有点的最小面积的矩形，输出所求矩形的面积和四个顶点坐标。

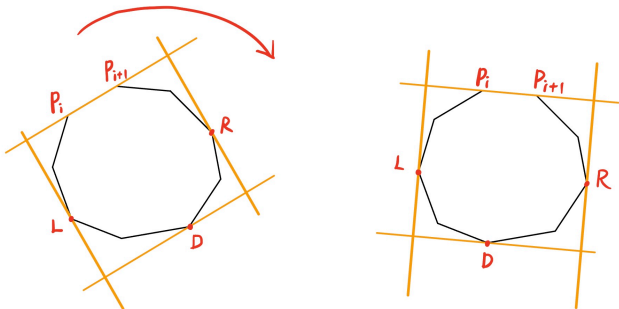
# [HNOI2007] 最小矩形覆盖

覆盖所有点等价于覆盖凸包的所有顶点，所以肯定先求凸包。

# [HNOI2007] 最小矩形覆盖

覆盖所有点等价于覆盖凸包的所有顶点，所以肯定先求凸包。

接下来都会想到旋转卡壳，像这样：

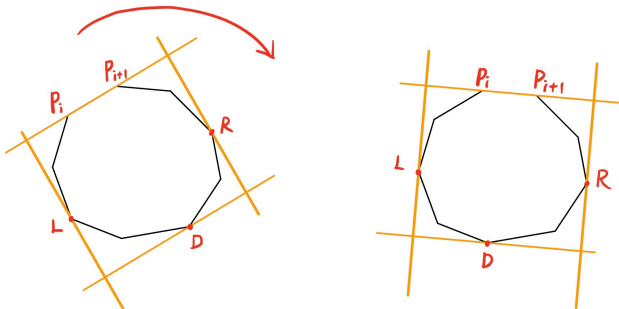


然后按顺序枚举边  $P_iP_{i+1}$ ，并维护  $L, R, D$  三个点。但是有一个问题。

# [HNOI2007] 最小矩形覆盖

覆盖所有点等价于覆盖凸包的所有顶点，所以肯定先求凸包。

接下来都会想到旋转卡壳，像这样：



然后按顺序枚举边  $P_i P_{i+1}$ ，并维护  $L, R, D$  三个点。但是有一个问题。

为什么最小矩形一定有一条边和凸包的某条边重合？难道最小矩形不能每条边都只和凸包的一个顶点相切吗？

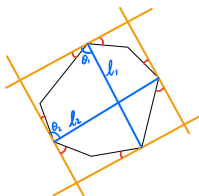
# [HNOI2007] 最小矩形覆盖

**结论：**最小矩形一定有一条边与凸包重合。

# [HNOI2007] 最小矩形覆盖

**结论：**最小矩形一定有一条边与凸包重合。

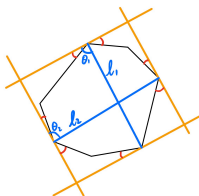
**证明：**若不然，即最小矩形的每条边都只和凸包的一个顶点相切。



# [HNOI2007] 最小矩形覆盖

**结论：**最小矩形一定有一条边与凸包重合。

**证明：**若不然，即最小矩形的每条边都只和凸包的一个顶点相切。



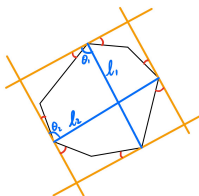
此时面积是：  $S = l_1 \sin \theta_1 \cdot l_2 \sin \theta_2$ .



## [HNOI2007] 最小矩形覆盖

**结论：**最小矩形一定有一条边与凸包重合。

**证明：**若不然，即最小矩形的每条边都只和凸包的一个顶点相切。



此时面积是： $S = l_1 \sin \theta_1 \cdot l_2 \sin \theta_2$ .

将矩形某条边旋转很小很小的  $\theta$  角度（使标红的那些角仍然大于 0），另外三条边卡上去，新矩形的面积为：

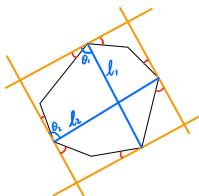
$$S' = l_1 \sin(\theta_1 + \theta) \cdot l_2 \sin(\theta_2 + \theta) \quad (4)$$

$$= -\frac{1}{2} l_1 l_2 (\cos(\theta_1 + \theta_2 - 2\theta) - \cos(\theta_1 - \theta_2)). \quad (5)$$

## [HNOI2007] 最小矩形覆盖

**结论：**最小矩形一定有一条边与凸包重合。

**证明：**若不然，即最小矩形的每条边都只和凸包的一个顶点相切。



此时面积是： $S = l_1 \sin \theta_1 \cdot l_2 \sin \theta_2$ .

将矩形某条边旋转很小很小的  $\theta$  角度（使标红的那些角仍然大于 0），另外三条边卡上去，新矩形的面积为：

$$S' = l_1 \sin(\theta_1 + \theta) \cdot l_2 \sin(\theta_2 + \theta) \quad (4)$$

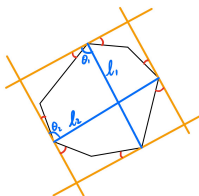
$$= -\frac{1}{2} l_1 l_2 (\cos(\theta_1 + \theta_2 - 2\theta) - \cos(\theta_1 - \theta_2)). \quad (5)$$

按照假设，必须当  $\theta = 0$  时  $S'$  取得最小值。也就是说当  $\theta = 0$  时  $\cos(\theta_1 + \theta_2 - 2\theta)$  取得最大值。

## [HNOI2007] 最小矩形覆盖

**结论：**最小矩形一定有一条边与凸包重合。

**证明：**若不然，即最小矩形的每条边都只和凸包的一个顶点相切。



此时面积是： $S = l_1 \sin \theta_1 \cdot l_2 \sin \theta_2$ .

将矩形某条边旋转很小很小的  $\theta$  角度（使标红的那些角仍然大于 0），另外三条边卡上去，新矩形的面积为：

$$S' = l_1 \sin(\theta_1 + \theta) \cdot l_2 \sin(\theta_2 + \theta) \quad (4)$$

$$= -\frac{1}{2} l_1 l_2 (\cos(\theta_1 + \theta_2 - 2\theta) - \cos(\theta_1 - \theta_2)). \quad (5)$$

按照假设，必须当  $\theta = 0$  时  $S'$  取得最小值。也就是说当  $\theta = 0$  时  $\cos(\theta_1 + \theta_2 - 2\theta)$  取得最大值。根据  $\cos$  的性质我们必须有  $\theta_1 + \theta_2 = 2k\pi$ ，这显然是不可能的。证毕！

# [HNOI2007] 最小矩形覆盖

现在我们已经证明了结论，可以放心地卡壳了。

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

```
double RotatingCalipers(Point *ch, int n){
    int D = 0, L = -1, R = -1;
    double ans = 0;
    ch[n+1] = ch[1];
    for(int i = 1; i <= n; i++){
        while(DistanceToLine(ch[D], ch[i], ch[i+1])
            <= DistanceToLine(ch[D%n+1], ch[i], ch[i+1])){
            D = D % n + 1;
        }
        if(L== -1) L = i + 2;
        if(R== -1) R = D % n + 1;
        Point D2 = D + Rotate(ch[i+1]-ch[i], pi/2);
        while(DistanceToLine(ch[R], ch[D], D2)
            <= DistanceToLine(ch[R%n+1], ch[D], D2)){
            R = R % n + 1;
        }
        while(...) L = L % n + 1; // 自己写
        ans = max(ans, ...); // 自己写
    }
    return ans;
}
```

## [SCOI2007] 最大土地面积

给定一些点的坐标，选其中四个点，使它们构成的四边形面积最大。

不超过 2000 个点。（其实  $10^5$  也能做）

# [SCOI2007] 最大土地面积

旋转卡壳枚举对角线  $P_uP_v$ ，然后维护  $a, b$  分别为  $l_{P_uP_v}$  两侧距离该直线最远的点。下面是确定  $u, v$  后维护  $a, b$  的代码。

1

2

3

4

5

6

7

8

9

10

11

12

13

14

```

void chkmx(int u, int v, int &a, int &b) {
    while ((a + 1) % n != v &&
           distance_to_line(p[a+1], p[u], p[v]) >
           distance_to_line(p[a], p[u], p[v]))
        a = (a + 1) % n;
    while ((b + 1) % n != u &&
           distance_to_line(p[b+1], p[u], p[v]) >
           distance_to_line(p[b], p[u], p[v]))
        b = (b + 1) % n;
    ans = max(ans,
              length(p[u] - p[v]) *
              (distance_to_line(p[b], p[u], p[v])
               + distance_to_line(p[a], p[u], p[v])));
}

```

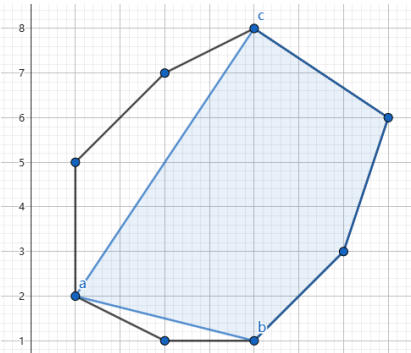
[SDCPC2023] Computational Geometry

给定一个有  $n$  个顶点的凸多边形  $P$ ，您需要选择  $P$  的三个顶点，按逆时针顺序记为  $a$ ， $b$  和  $c$ 。要求在  $b$  沿逆时针方向到  $c$  之间恰有  $k$  条边（也就是说， $a$  不是这  $k$  条边的端点）。

考虑用线段  $ab$  和  $ac$  将  $P$  割开。将由线段  $ab$ ， $ac$ ，以及  $b$  和  $c$  之间的  $k$  条边围成的  $(k + 2)$  边形记作  $Q$ 。

求  $Q$  可能的最大面积。

注意， $ab$  和  $ac$  可以与  $P$  的边重合。



## [SDCPC2023] Computational Geometry

枚举  $b$ , 然后  $c$  就确定了, 接下来要让  $a$  离  $l_{bc}$  的距离最远, 其实也就是旋转卡壳。



① 基础回顾

② 旋转卡壳

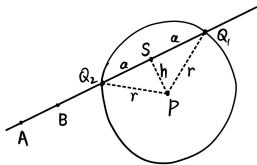
③ 圆

# 圆与直线的交点

给定直线上两个点  $A, B$  的坐标、圆心  $P$  的坐标、圆的半径  $r$ ，求直线  $l_{AB}$  与圆的交点。

# 圆与直线的交点

给定直线上两个点  $A, B$  的坐标、圆心  $P$  的坐标、圆的半径  $r$ ，求直线  $l_{AB}$  与圆的交点。



先求出  $P$  到  $l_{AB}$  的距离  $h$ ，然后计算  $a = \sqrt{r^2 - h^2}$ ，然后求出：

$$S = P + h \frac{\overrightarrow{AB}^\perp}{|\overrightarrow{AB}|} \tag{6}$$

$$Q_1 = S + a \frac{\overrightarrow{AB}}{|\overrightarrow{AB}|} \tag{7}$$

$$Q_2 = S - a \frac{\overrightarrow{AB}}{|\overrightarrow{AB}|} \tag{8}$$

## [BZOJ2178] 圆的面积并

给定  $n$  ( $n \leq 1000$ ) 个圆，求它们覆盖区域的面积。

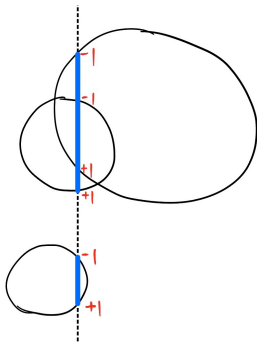
## [BZOJ2178] 圆的面积并

基本思路是辛普森积分求  $\int_L^R f(t) dt$ , 其中  $f(t)$  表示直线  $x = t$  与覆盖区域相交部分的长度。

# [BZOJ2178] 圆的面积并

基本思路是辛普森积分求  $\int_L^R f(t) dt$ , 其中  $f(t)$  表示直线  $x = t$  与覆盖区域相交部分的长度。

对于一条直线  $x = t$ , 我们可以枚举所有圆, 求出与该直线的所有交点, 并从小到大排序, 然后依次枚举; 每碰到一个下交点, 就加 1, 碰到上交点就减 1; 非零部分的总长度就是相交部分的长度。



# 圆交

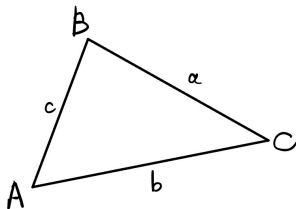
给定两个圆的圆心坐标  $P_1, P_2$ ，以及它们的半径  $r_1, r_2$ ，判断它们是否有交点（或切点），如果有，求之。

# 正余弦定理

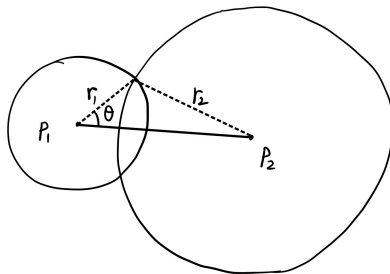
在求圆交之前，我们先来看两个三角形里的基本定理。



在求圆交之前，我们先来看两个三角形里的基本定理。

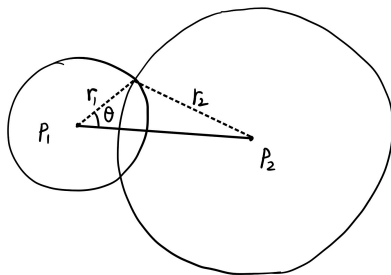

$$\frac{\sin A}{a} = \frac{\sin B}{b} = \frac{\sin C}{c}. \quad (9)$$
$$\cos A = \frac{b^2 + c^2 - a^2}{2bc}, \quad \cos B = \frac{a^2 + c^2 - b^2}{2ac}, \quad \cos C = \frac{a^2 + b^2 - c^2}{2ab}. \quad (10)$$

## 圆交



先用余弦定理求出  $\theta$ ,

## 圆交



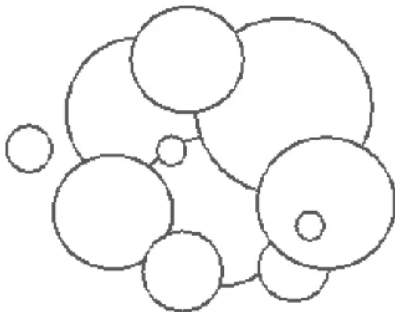
先用余弦定理求出  $\theta$ ，然后求出第一个交点：

$$P_1 + r_1 \frac{\text{Rotate}(\overrightarrow{P_1 P_2}, \theta)}{|P_1 P_2|}. \quad (11)$$

第二个交点类似。

## [UVA10969] Sweet Dream

依次输入  $n$  个圆的圆心坐标、半径，后输入的圆堆叠在先输入的圆上面，问最后可见部分的弧长总和。 $T$  组数据。 $(T, n \leq 100)$



## [UVA10969] Sweet Dream

对于第  $i$  个圆，求出它和其它所有圆的交点，并将这些交点按逆时针排序。

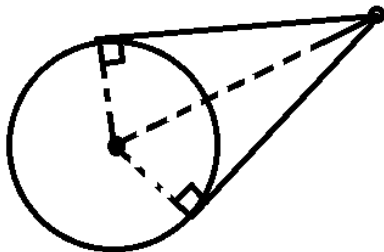
## [UVA10969] Sweet Dream

对于第  $i$  个圆，求出它和其它所有圆的交点，并将这些交点按逆时针排序。

这些交点将圆分成若干个圆弧，每个圆弧要么整个露出来，要么整个被挡掉。所以只要判断每个圆弧的中点是否位于某个圆  $j$  ( $j > i$ ) 内部即可。最后将露出来的圆弧长度累加。

# 过定点作圆的切线

过圆外一点作圆的切线，求两个切点。



# 过定点作圆的切线

直接用反正弦函数计算夹角，再由勾股定理计算切线段长度即可。

```

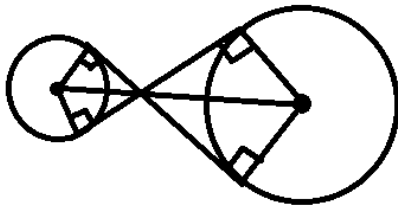
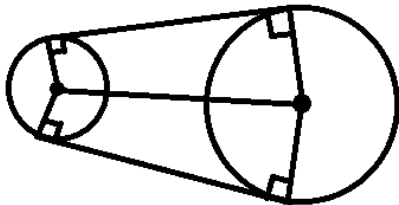
1 void circleTangent(Point P, double r, Point Q, Point &A, Point &B) {
2     Vector v = Q - P;
3     double c = Length(v);
4     double b = sqrt(c*c - r*r);
5     double theta = asin(r/c);
6     A = Rotate(v, theta) / c * b;
7     B = Rotate(v, -theta) / c * b;
8 }

```



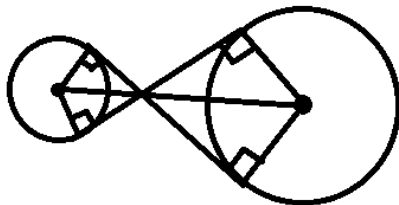
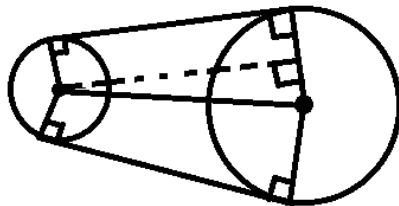
# 求两圆的公切线

给定两个圆，求外公切线（上）和内公切线（下）。



# 求两圆的公切线

注意，需要对圆的位置分类讨论切线是否存在。这里我们假设内外切线都存在，且都有两条，作辅助线：



# 三点定圆

给出三角形的三个顶点  $(x_i, y_i)$  ( $i = 1, 2, 3$ ), 求外接圆 (用解析几何)。

# 三点定圆

设圆心为  $(x, y)$ , 半径为  $r$ , 列方程:

$$(x - x_i)^2 + (y - y_i)^2 = r^2, \quad i = 1, 2, 3. \quad (12)$$

方程  $i$  ( $i = 2, 3$ ) 减去方程 1, 得到:

$$2(x_i - x_1)x + 2(y_i - y_1)y = x_i^2 - x_1^2 + y_i^2 - y_1^2, \quad i = 2, 3. \quad (13)$$

解线性方程即可得到  $x, y$ , 再代入方程 1 求出  $r$ .

*Thank You*