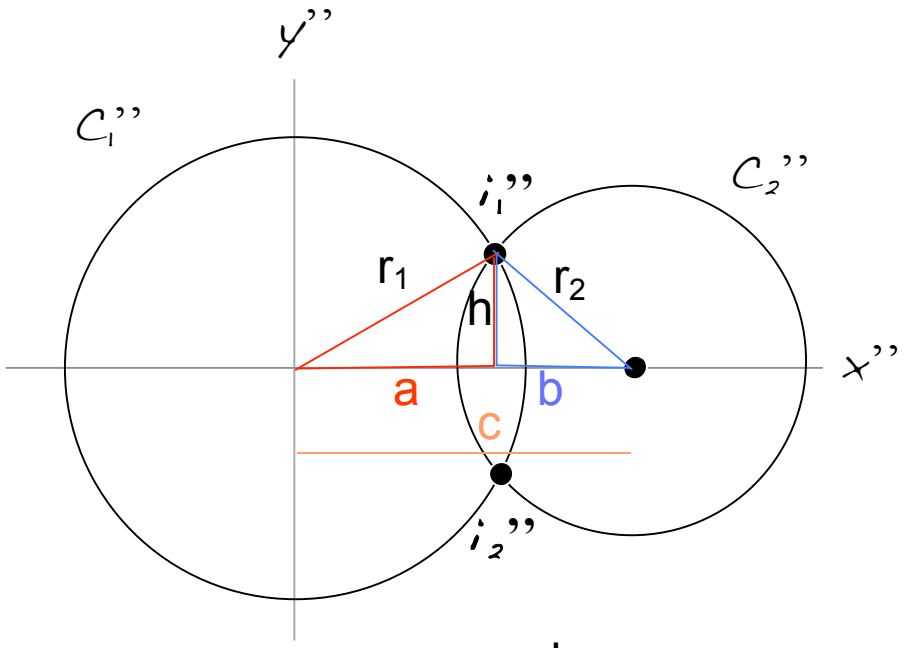
Subproblem: How many intersection

# numIntPts	Draw examples	Decision logic
∞		
1		
0		
2		

Subproblem: Locations of intersection

Easier-to-solve special case:



$c = a + b$

$a^2 + h^2 = r_1^2$

$b^2 + h^2 = r_2^2$

$a^2 - b^2 = r_1^2 - r_2^2$

$(a + b)(a - b) = r_1^2 - r_2^2$

$c\,(a - b) = r_1^2 - r_2^2$

$a - b = (r_1^2 - r_2^2) / c$

$a + b = c$

$2a = c + (r_1^2 - r_2^2) / c$

$2a = c^2/c + (r_1^2 - r_2^2) / c$

$a = (c^2 + r_1^2 - r_2^2) / 2c$

$c = C_2''.c.x$

$a = (c^2 + r_1^2 - r_2^2) / 2c$

$h = \text{sqrt}\ (r_1^2 - a^2)$

$i_1'' = [\ a \quad h\ ]$

$i_2'' = [\ a \quad -h\ ]$

```
//******************** Number of Intersection Points Of 2 Circles ****************************
     // a HELPER function for intersectionPtsOf(C1, C2)
      private static int numIntersectionPts(Circle C1, Circle C2) {
           if (identical(C1, C2)) return 3;
           if (Vec2.distance(C1.c, C2.c) == C1.r + C2.r) return 1;
           if (Vec2.distance(C1.c, C2.c) > C1.r + C2.r) return 0;
           if (Vec2.distance(C1.c, C2.c) == Math.abs(C1.r - C2.r)) return 1;
           if (Vec2.distance(C1.c, C2.c) < Math.abs(C1.r - C2.r)) return 0;
           return 2; // default if not 0, 1 or 3
          }

     //******************** intersection Points Of 2 Circles ****************************
     public static int intersectionPtsOf (Circle C1, Circle C2, /*returns*/ Vec2 i1, Vec2 i2) {
          int numPts = numIntersectionPts(C1, C2);
          if (numPts == 0 || numPts == 3) return numPts; // no intersection points can be computed

          // transform problem into simpler special case
          Circle C2_p = Circle.translate(C1.c, C2); // shift to C1-local coordinates
          DirVec newXaxis = new DirVec(C2_p.c);
          Circle C2_pp = Circle.rotate(newXaxis, C2_p); // rotate: centers C2_pp on positive x-axis

          double c = C2_pp.c.x;
          double a = ( c*c  + C1.r*C1.r - C2.r*C2.r ) / (2*c);
          double h = Math.sqrt(C1.r*C1.r - a*a);
          Vec2 i1_pp = new Vec2(a,h);
          Vec2 i2_pp = new Vec2(a,-h);

          // back-transform using Unrotate then Untranslate
          i1.copyFrom(Vec2.untranslate(C1.c, Vec2.unrotate(newXaxis,i1_pp)));
          i2.copyFrom(Vec2.untranslate(C1.c, Vec2.unrotate(newXaxis,i2_pp)));
          return numPts;
          }
```