与圆有关的离散化方法

清华附中 高逸涵

(gaoyihan@gmail.com)

【摘要】在计算几何问题中，离散化方法是一种较为通用的算法，在解决一些与矩形等直线型有关的题目时，能大大降低算法的时空复杂度。但当问题与圆相关时，直接离散化法有较大困难。本文讨论了离散化法在这类问题中的方法，然后通过几道例题说明如何利用离散化法解决与圆有关的计算几何问题。

【关键字】

计算几何 圆 离散化

【正文】

引言

对于绝大多数算法来说,连续的数据并不是一个合适的处理对象，,必须将其离散化才能处理。高效的离散化方法可以降低问题复杂度，因此如何实现离散化算法是一个有意义的课题。本文通过计算几何中与圆有关问题的分析，研究离散化方法的应用形式。

我们先来看一下离散化法是如何解决一道求矩形面积并的题目的：

平面上有 **3** 个矩形如图，分别为（**1**，**1**）-（**4**，**4**）、（**2**，**3**）-（**5**，**5**）、（**3**，**4**）-（**6**，**6**）。要求出三个矩形的面积并。

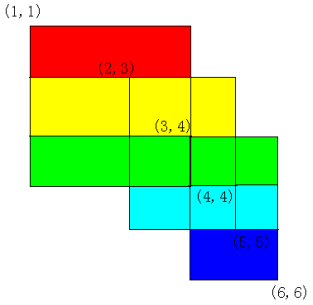

图 **1**

我们利用离散化将连续平面纵轴划分为 5 个区间：（1

—2）、（2—3）、（3—4）、（4—5）、（5—6）。可以看到，在同一区间内的属性一致，即横轴上被矩形所覆盖的区域是相同的，于是每一区域的面积可以通过乘法很快求出，然后再把所有区间的面积求和即可。由此可以看出离散化可以降低复杂性的优势。

方法

当计算的对象不是矩形而是圆时，例如求 3 个圆的合并面积，由于圆的边界是一条曲线，不存在固定的矩形区域，似乎不可能利用上述离散化方法来分成类似的各个区域。但是如下图，当我们把纵轴分成如下四个区间时，整个图形被我们分成了 6 部分，并且每一部分都由简单的弓形和梯形组成，构成复合属性的离散区域，同样可以使问题简化。这提示我们，可以把图形切成若干相对简单的块，使得每一部分的面积都很容易求出。这便是离散化在与圆有关的计算几何中最基础的应用。

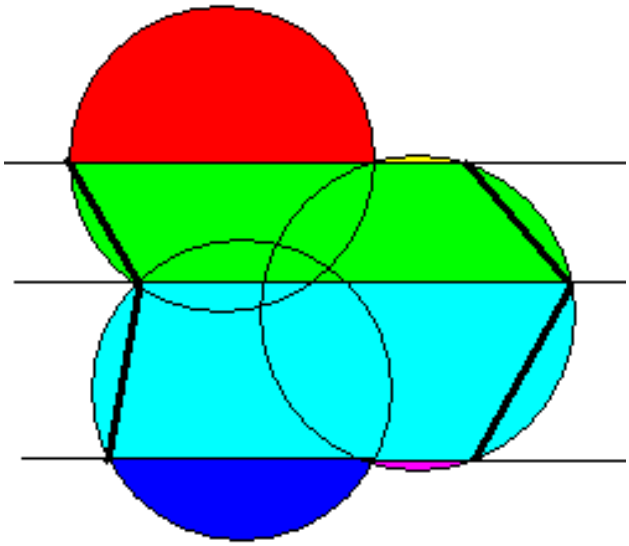

图 **2**

算法的一般步骤:

1. 根据问题将平面中的圆分割成若干区域，使每个区域具有一定简单的粗粒属性。一般可以用直线分割。
2. 根据属性确定区域内圆的具体算法，计算每个区域中结果。
3. 综合每个区域的结果，给出问题的最终答案。

应用实例及结果

这里我们通过两个例题具体分析来讨论算法的具体应用形式。

例一、**dolphin pool**[1]

题目大意：

平面中有 n 个圆，任意两圆不相切，没有一个圆的圆心在另一个圆内。求有多少闭合的区域没有被任何圆覆盖。

输入：

第一行包含一个数 $n(0<n\leq 20)$，表示圆的个数。

接下来 n 行分别输入三个整数圆心坐标 x、y 和半径 r。($1\leq x、y\leq 1000$，$1\leq r\leq 100$)

输出：

输出一个整数，表示闭合区域的个数。

初步分析：

虽然题目中已经对圆的位置有了一定的约束，但实际上可能的情况还是很多的，我们希望能找到一种简单而通用的算法。

由于数据都是整数，我们的初步想法是这道题的精度要求不高，因此可以采用一种基于 floodfill 的算法：

在平面内每隔一定距离取一个点形成点阵，删除所有在圆内的点，然后在剩下的点中 floodfill 求连通块，最后减 1 就是答案。算法复杂度为 $O(k^2n)$，k 为 1000 长度中取点的个数。

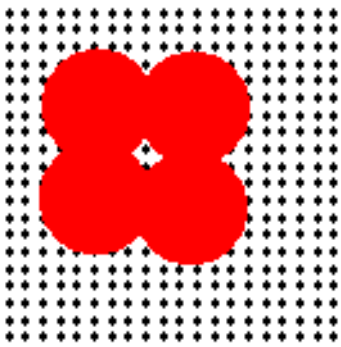

图 **3**

我们知道，这个算法的正确性或精度取决于 k 的大小，而对于这道题目的时限来说，这个算法远远不能达到要求，因此我们需要改进算法。

第一步离散化：

考虑点集的每一横行，我们发现每一个圆在其上覆盖的点一定是连续的。

[1] Peking University Online Judge 1688(acm.pku.edu.cn/JudgeOnline)

图 **4**

于是我们想到在每一横行中不必枚举每一个点，转而计算每一个圆在其上的覆盖区间，这一步可以用数学算法在常数时间内算出。

$$\text{Left} = \text{circle.x} - \sqrt{\text{circle.radius}^2 - (\text{circle.y} - \text{line.height})^2}$$

$$\text{Right} = \text{circle.x} + \sqrt{\text{circle.radius}^2 - (\text{circle.y} - \text{line.height})^2}$$

然后合并相交的区间。在预处理时，可以将圆按横坐标排好序，这样在合并区间时会变得更简单。

最后根据被覆盖的区间求出未被圆覆盖的区间。

仿照方法 1，我们将所有的区间记录下来，并在相邻两行有公共部分的区间连一条边，然后求连通块数，最后减 1 即为答案。

由于区间个数太多，无论 DFS 还是 BFS 都消耗了太多内存，因此我们可以利用图的特殊性，创造一种效率较高的顺序扫描求连通块数法：

1. 初始化第一横行（仅有一个区间），这一区间从属于 1 个集合。
2. 给下一横行的每一区间初始化一个新的集合仅包含他。
3. 如果两行中某两个区间有交集，合并他们所从属的集合。
4. 如果某一集合中的所有区间在下一行都没有继承者，答案加 1。
5. 重复 2~4，直到扫描至最后一行。

可以看到，这个算法的复杂度为 *O(nk)*，在时限内可以得到较高的精度。

但是，由于题目中存在某些精度要求非常高的数据，并且此算法本身的精度不够，最终这个算法得到了 Wrong Answer。所以，我们仍需要进一步改进。

第二步离散化：

仔细观察程序，我们发现，似乎有很多无用的工作。如下例：

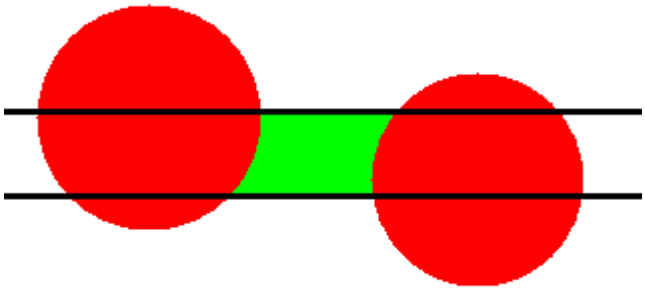

图 **5**

在绿色区域中，我们直观的感觉到，这肯定是同一片区域，因为在其中根本就没有其他的圆，而我们的程序却在其中辛苦的一步步跟踪，直到最后确认最下一行确实是最上一行的后继。

于是我们有了另一种想法——省略多余的操作。我们发现，区间之间的继承关系在大多数时候都没有改变，只在仅有的少数几个时刻才会发生改变：

- 一个圆新出现时，将一个区间分成两半。
- 两圆相交，一个区间逐渐减小并被吞噬。
- 两圆相交，一个新的区间生成。
- 一个圆的最下端，两个区间合并为一个区间。

这样，我们仅需记录所有的点事件，并模拟区间的生长消亡，然后求解。注意到，我们没有必要记录区间的确切长度。

但由于点事件的定性有四种，其中中间两种较难分辨，程序很容易出错。事实上，我们甚至没有必要记录点事件的性质，对于每一个点事件，取它之前的极短一段时间和之后的极短一段时间，然后利用方法 2 中的顺序扫描法，下一个点事件的前驱直接继承这个点事件的后继。

至此，由于采用了离散化法，本题已被完美解决。时间复杂度为 $O(n^3)$。

例二、**Empire strikes back**[1]

题目大意：

Saddam 的国度为半径为 R 的圆，圆心在(0,0)，其中有 N 个化学武器工厂（坐标分别为(Xi, Yi)），George II 将在每个工厂中投掷炸弹（炸弹将炸平以工厂为圆心一定半径内的一切）。但由于不清楚 Saddam 躲藏在哪里，George II 决定使炸弹的杀伤范围将整个国土都覆

[1] Ural State University Online Judge 1520(acm.timus.ru)

盖，问题是炸弹的最小杀伤半径为多少。

输入：

第一行包含两个数 N($1 \le N \le 300$),R($1 \le R \le 1000$) 分别表示炸药数和半径。

接下来 n 行每行包含一个炸药的描述。输入两个数(Xi, Yi) ($\mathbf{X_i}^2 + \mathbf{Y_i}^2 \le \mathbf{R}^2$)

输出：

最小的半径，保留 5 位小数

初步想法：

题目仅仅求一个数，所以我们考虑使用二分法求得答案。

二分法步骤如下：

1. 初始化 $\text{Left} = 0, \text{Right} = 2 \times \text{R}$ 。
2. 求出中间数 $\text{Mid} = (\text{Left} + \text{Right}) \div 2$ 。
3. 判断以 Mid 为半径是否能覆盖整个圆，如果能则 $\text{Left} = \text{Mid}$，否则 $\text{Right} = \text{Mid}$ 。
4. 当 $\text{Right} - \text{Left} < 10^{-6}$ 时退出，否则重复 2~3。

二分法将一个最优性问题转化为一个可行性问题，现在问题的重点是判断当半径为确定值时，大圆是否被完全覆盖。

算法分析：

初看起来，这题和上一题非常相似。

但如果沿用上一例的算法，时间上完全承受不了。

细细分析，才发现两题之间有些许不同，上一题要求出区域个数，而本题仅需判定是否完全覆盖，由计数问题转化为判定性问题，而我们仍在用原先的思路解题，必然无法得到高效的算法。因此我们需要重新设计算法

考虑最原始的思路：枚举大圆内每个点，判定其是否被小圆覆盖。如果每一个大圆内的点都被小圆覆盖，则大圆一定被小圆完全覆盖。

但是，由于题目对精度的要求非常高，因此本算法必然不能正确求得答案，我们想到可以利用离散化方法优化此算法。

第一步离散化：

沿用上例的思路，我们将一个横行看成一个整体。对于每一横行求得每个小圆覆盖的区间，然后合并所有相交的区间，如果最后仅剩一个区间，且这个区间包含了大圆在其上覆

盖的区间，则可以判定这个大圆在这条横线已被完全覆盖。如果对于所有横线大圆都被完全覆盖，则我们可以近似的认为大圆已被小圆完全覆盖。

由于题目仅仅需要我们判定是否完全覆盖，因此取特殊点是一个不错的想法。考虑每一横行中关键点的左右两端极其接近的点判断其是否被其他圆覆盖。

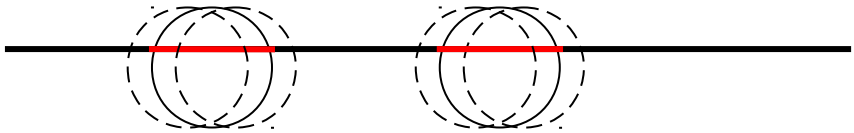

图 **6**

可以看到，如果仅看我们所取的所有特殊点，那么这些特殊点所组成的图形恰好为每个圆向左偏移一点和向右偏移一点的图形之和。并且，其中在原先的圆内的一部分可以不考虑。又因为偏移量非常小，因此我们可以将所有特殊点视为在偏移前的圆的边界上。

第二步离散化

应用集中处理的原则，我们同时处理同一圆上的所有点。我们看到，每个圆在另一个圆的边界上覆盖的区间一定是连续的。因此，问题转化为很多个小区间是否完全覆盖整个区间。这和例 1 中第一步离散化后得到的问题十分类似，因此可以使用类似的算法解决。

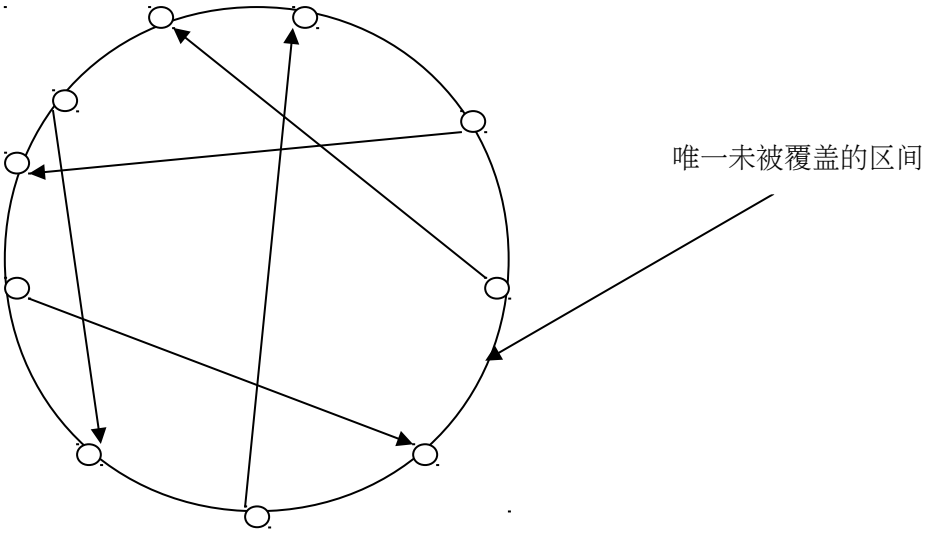

图 **7**

复杂度分析：

二分的复杂度 O(logR)；

对于区间的排序 O(NlogN)；

区间合并 O(N)；

枚举 N 个圆 O(N)；

算法总复杂度：

$O(N^2 \log N \log R)$

至此，问题已被完美解决。

一些小的优化：

事实上，如果在排序时，按照区间的中点的极角进行排序，则区间的序和半径长度无关，可以在预处理时完成，复杂度可降为 $O(N^2 (\log N+\log R))$。

由于每一个圆被覆盖所需要的最小半径都是相互独立的，所以我们可以认为它是一个常数，而我们所需要的是求出这些常数中的最大值。所以可以在枚举每个圆的时候，先判断一下它在当前最小半径下是否能被完全覆盖，如果能覆盖则不再二分。杨弋同学和袁昕颢同学证明了这种算法的算法复杂度期望值为 $O(N^2*\log N +N*\log N*\log R)$，证明如下：

考虑一种算法：

1. 随机选一个小圆
2. 二分测出完全覆盖它的所需的最小半径
3. 然后和其他所有的小圆比较
4. 把不能完全覆盖的小圆留下来
5. 递归

可以发现，这个算法和上文所述算法本质上是相同的。而这个算法[1]很容易证明其复杂度为 $O(N+\log N*\log R)$，所以原算法的复杂度为 $O(N^2*\log N+N*\log N*\log R)$。另外，可以通过随机化避免最坏情形发生。

【总结】

离散化法在计算几何中是一种比较普遍适用的算法。当问题与圆有关时，离散化法能利用区域的离散化将问题转化为相对简单的“图形”计算几何问题。离散化法的本质在于化“零”为“整”，化“曲”为“直”，实现简化算法复杂度的目的。当然，由于这一过程不可避免地破坏了曲线的性质，因此离散化法作为一种较为朴素的处理方法，往往不如其他直接利用曲线性质的算法高效。但其普遍适用的特性和简洁性使得它能够在计算几何算法中占有一席之地。本文虽然分析的是圆的计算几何问题，但很容易推广到其他曲线的计算几何问题。

【感谢】

感谢安徽师大附中的杨弋同学和长沙市长郡中学的袁昕颢对例二算法复杂度的证明。

[1] 事实上，这个算法就是快速选择算法的一个特殊形式

感谢所有帮助过我的人。

【参考文献】

刘汝佳，黄亮《算法艺术与信息学竞赛》 清华大学出版社

【附件】

例一《Dolphin Pool》一题的源程序 
Dolphin Pool.pas

求两圆交点（基于余弦定理，有一定误差）：

```
int get_inter( int x1, int y1, int x2, int y2, double rad1, double rad2, double
&ret1, double &ret2 )
{
    double b = sqrt( (double)( sqr( y2 - y1 ) + sqr( x2 - x1 ) ) );
    if ( b > rad1 + rad2 || b < rad1 - rad2 )
    {
        ret1 = 0;
        ret2 = 0;
        return 0;
    }
    else if ( b < rad2 - rad1 )
    {
        ret1 = 0;
        ret2 = 2 * pi;
        return 0;
    }
  //以上为判断特殊情况
    double mid = atan2( y2 - y1, x2 - x1 );
    double a = rad1, c = rad2;
    double t = acos( ( sqr(a) + sqr(b) - sqr(c) ) / ( 2 * a * b ) );
  //所取三角形的三顶点分别是两圆圆心和其中一交点
    ret1 = mid - t;
    ret2 = mid + t;
  //返回值为交点在第一个圆上的极角
    return 0;
}
```

【附录】

[例一原题]

Dolphin Pool

Time Limit:1000MS Memory Limit:10000K

Description

In a newly constructed dolphin pool in the Kish island in Persian Gulf, one of the fun games is as follows: the game director throws several plastic rings in the pool such that center of no ring lies inside any other ring, and no two rings are tangent. The dolphins are trained to jump out on the director's whistle through the closed areas that are completely outside the rings, one dolphin from one such area. The dolphins jump out if and only if the number of closed areas exactly equals to the number of dolphins.

You are to write a program to given the following input/output description, finds the number of closed areas between rings to help the dolphins decide to jump out or not.

Input

The first line includes the number of test cases (at most 20). Each test case data has an integer N (1 <= N <= 20), the number of plastic rings, in its first line. Following the first line there are N lines, each containing three integers, the first and second being the x and y coordinates of the circle of the ring, and the third is its radius. Coordinates are positive integers less than 1000 and the radius is in the range 1...100.

Output

For each test case, there must be one line in the output including the number of closed areas in that test case.

Sample Input

```
2
4
100 100 20
100 135 20
135 100 20
135 135 20
1
10 10 40
```

Sample Output

1
0

Source

Tehran 2000

[例二原题]

Empire strikes back

Time Limit: 1.0 second
Memory Limit: 16 MB

Background

Many years have passed since the good and wise emperor George II the Great began to rule the cultural and civilized Empire. Oh, the world he created is so mighty and beautiful! Under his rule majestic cities of marble and steel are being turned into the sky, and huge fields are being scattering with seeds. The children play, the old men laugh, while the workpeople and the peasants forge the common weal...

But once George got to know, that fearful danger threatened the mankind. Malicious and cruel dictator Saddam III the Terrible, who ruled much less cultural and civilized Republic, intends to create the newest chemical weapon and seize the power over the planet.

Problem

According to a secret service report, Saddam constructed **N** chemical weapon factories within the Republic frontier, which is a circle of radius **R** with its center at the point (0, 0). Each factory is skillfully disguised as a hospital, a school or an old people's home and located at the point with cartesian coordinates ($\mathbf{X_i}$, $\mathbf{Y_i}$).

Saddam's vile intentions did not please George at all. So he decided to destroy all the factories by bombing. All bombs should have the

same effective casualty radius and be dropped precisely onto the corresponding factory.

Each bomb transforms any object within its effective casualty radius into a scorching gas cloud. This very fact prompted George to a funny thought, that it would be great to kill two birds with one stone and transform Saddam himself into such cloud.
Unfortunately, the secret service failed to define exact whereabouts of the villain. That is why George wants to calculate the effective casualty radius of the bombs so that, being dropped precisely onto the factories, they would destroy Saddam regardless of his location within the Republic. By the way, it needs a lot of very expensive polonium-210 to create a high-power bomb, so the effective casualty radius should be minimal.

Input

The first line contains the integer numbers **N** ($1 \le \mathbf{N} \le 300$) and **R** ($1 \le \mathbf{R} \le 1000$). Each of the next **N** lines contains the integer numbers $\mathbf{X_i}$ and $\mathbf{Y_i}$ ($\mathbf{X_i}^2 + \mathbf{Y_i}^2 \le \mathbf{R}^2$) for the corresponding factory.

Output

You should output the desired effective casualty radius. The radius should be printed with at least five digits after decimal point.

Sample Input

```
4 4
0 2
0 -2
2 0
-2 0
```

Sample Output

```
2.94725152
```