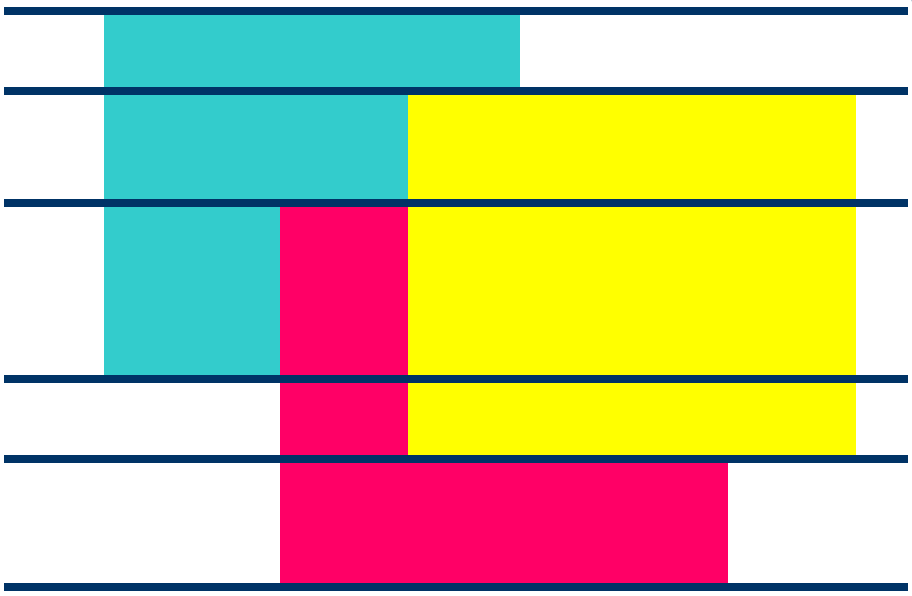
与圆有关的离散化方法

北京市清华附中

高逸涵

引言：一道经典的问题

- 平面上有N个矩形，求他们的面积并。
- 解法1：离散化
- 解法2：扫描线



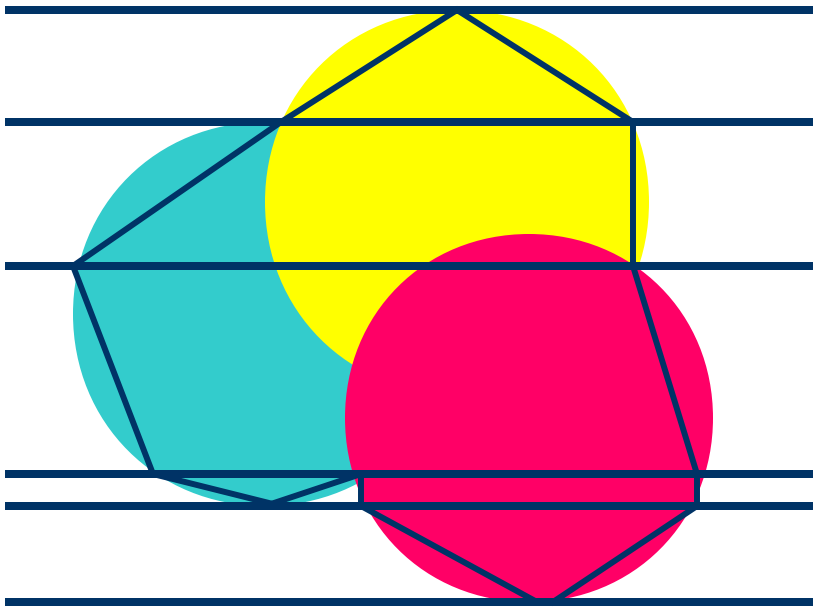
- 取每个矩形的上下边界，将平面分为若干部分，然后计算每一部分的面积。

引言：新的问题

- 由于矩形的边界为折线，如果我们用折线的转折点作为分界点，则折线被化简为许多线段。所以离散化法取得了成功。
- 如果需要统计的并不是矩形，而是某些曲线图形，怎么办?
- 对于曲线来说，根本没有所谓的转折点，传统的离散化法似乎对这一类问题失效了。

引言：新的问题

- 事实上，如果我们对于划分后每一部分的性质要求不那么严格，划分还是可以继续下去的。



- 例如，我们可以将图形划分为若干梯形和弓形，然后对梯形和弓形的面积根据各边线初等地求出，从而发现整个图形由若干梯形和弓形构成。

圆的离散化算法

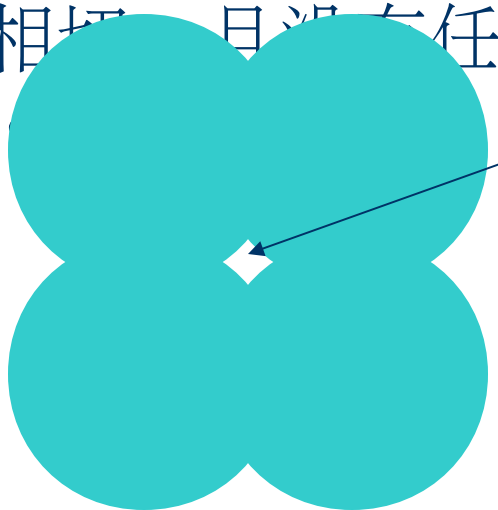
- 算法的一般步骤:
  - 根据问题将平面中的圆分割成若干区域，使每个区域具有一定简单的粗粒属性。一般可用直线分割。
  - 根据属性确定区域内圆的具体算法，计算每个区域中结果。
  - 综合每个区域的结果，给出问题的最终答案。
- 圆的离散化算法关键在于保证区域内数据在整体上易于处理。

算法应用实例

- 以上便是离散化法在圆的计算几何问题中最基本的应用。而这里将通过两道例题详细说明。
- [例 1] Dolphin Pool (Tehran 2000)
- [例 2] Empire strikes back (ural 1520)

Dolphin Pool (Tehran 2000)

- 给定 N(N<=20) 个圆的圆心坐标 (xi,yi) 和半径 Ri
- 求平面内在圆外的封闭区域的个数。
- 例如：如下四个圆构成一个圆外的封闭区域
- 没有任何两个圆相切，且没有任何一个圆的圆心在另一个圆内



初步分析

- 尽管圆的位置有一定的限制，但可能的情况还是很多的，我们希望有一个通用的算法来解决所有情况而不分类讨论，避免增加编程复杂度和错误几率。
- 由于输入输出都是整数，似乎题目对于精度的要求不高，并且坐标的范围较小 (x,y<=1000) 。于是可以使用一个基于 floodfill 的算法。

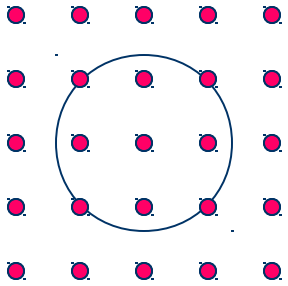
基于 **Floodfill** 的算法概述

- 在原图中建立点阵
- 标记所有在圆内的点为已访问。
- 使用深度优先遍历（DFS）求连通块数
- 连通块数减 1 即为答案

在原图中，建立点阵，

时间复杂度高。

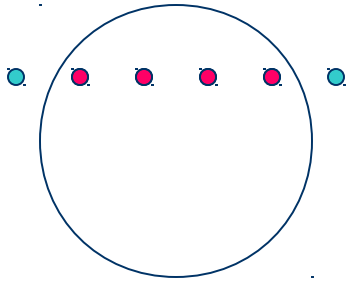
而对于这道题目的时限来说，这个算法远远不能达到要求，因此我们需要改进算法。



在原图中，建立点阵，这个算法的效率和正确性都由点阵的密集程度决定。

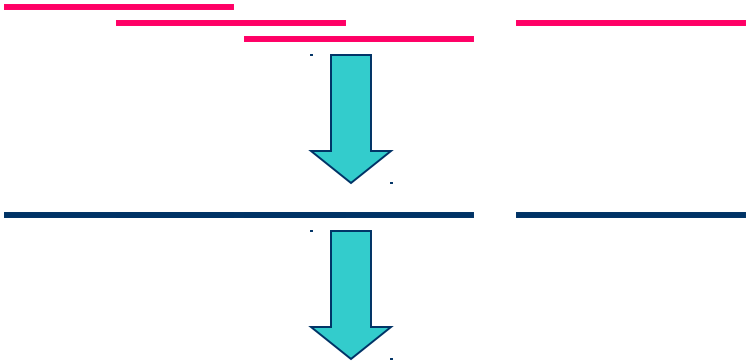
第一次离散化

- 考虑点阵中的每一横行，我们发现圆在每一横行上覆盖的一定是一个连续区间。
- 因此，可以仅记录每一个圆在当前横线上覆盖的区间而不记录每个点具体的被覆盖情况。
- 考虑平面内的一条平行于 x 轴的直线。可以很容易的计算出每一个圆在其上覆盖的区间。



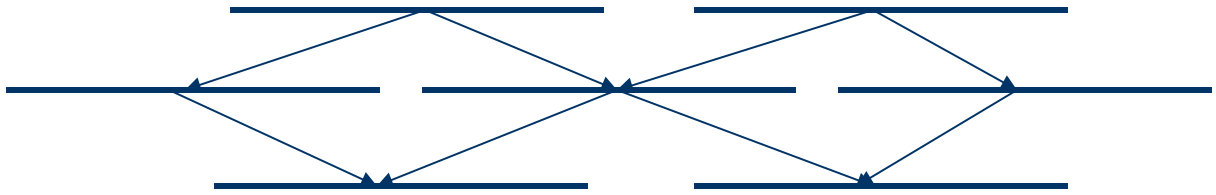
第一次离散化——区间合并

- 接下来，将所有的被覆盖区间合并。这一问题需要应用到基本数据结构栈，可以在线性时间复杂度内解决。
- 区间合并完成后，可以得到所有未被覆盖的区间。



第一步离散化——图论模型

- 建图，每个未覆盖区间为一个顶点，相邻两条横线上的区间如果相交，则在它们之间连一条边。然后判断在建成的图中有多少个连通块。
- 判断图中的连通块数量可以使用按层次遍历的方式减少空间需求。

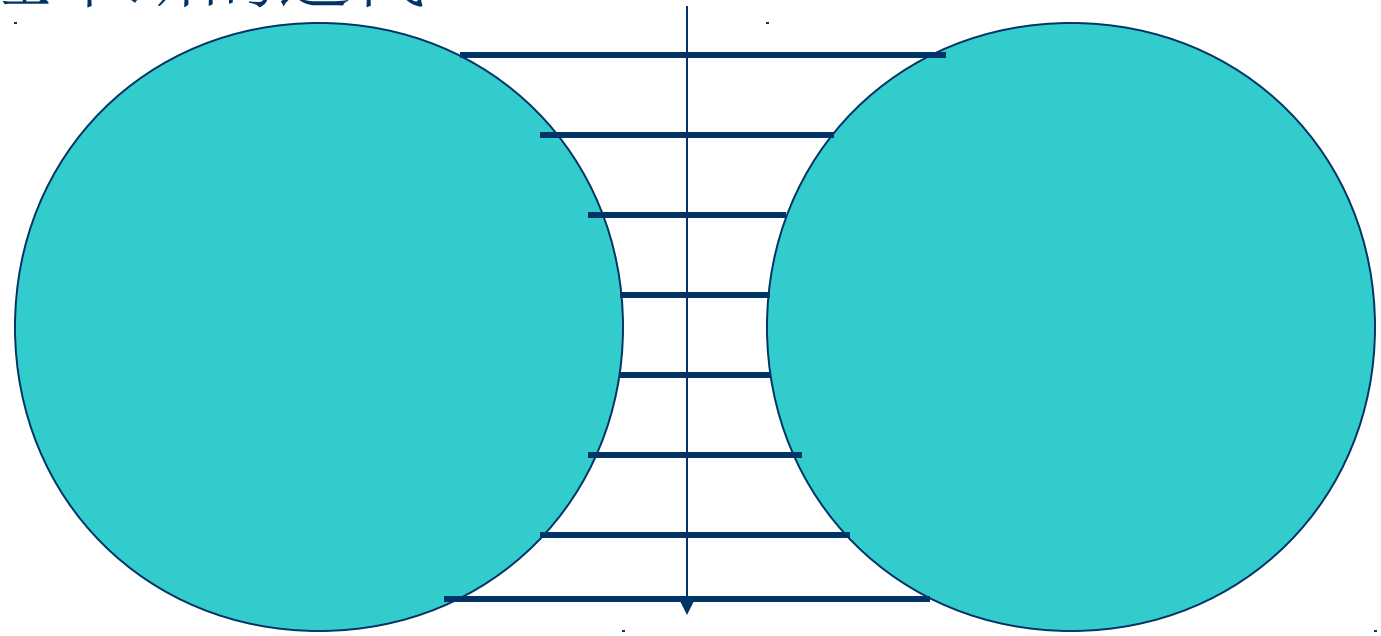


仍存在的问题

- 经过初步的离散化，上述算法的正确性和速度都有很大提高。但由于 ACM 竞赛要求程序必须通过全部数据，而该算法对于某些极限数据的精度仍不能令人满意，因此还有进一步改进算法的必要。

第二次离散化

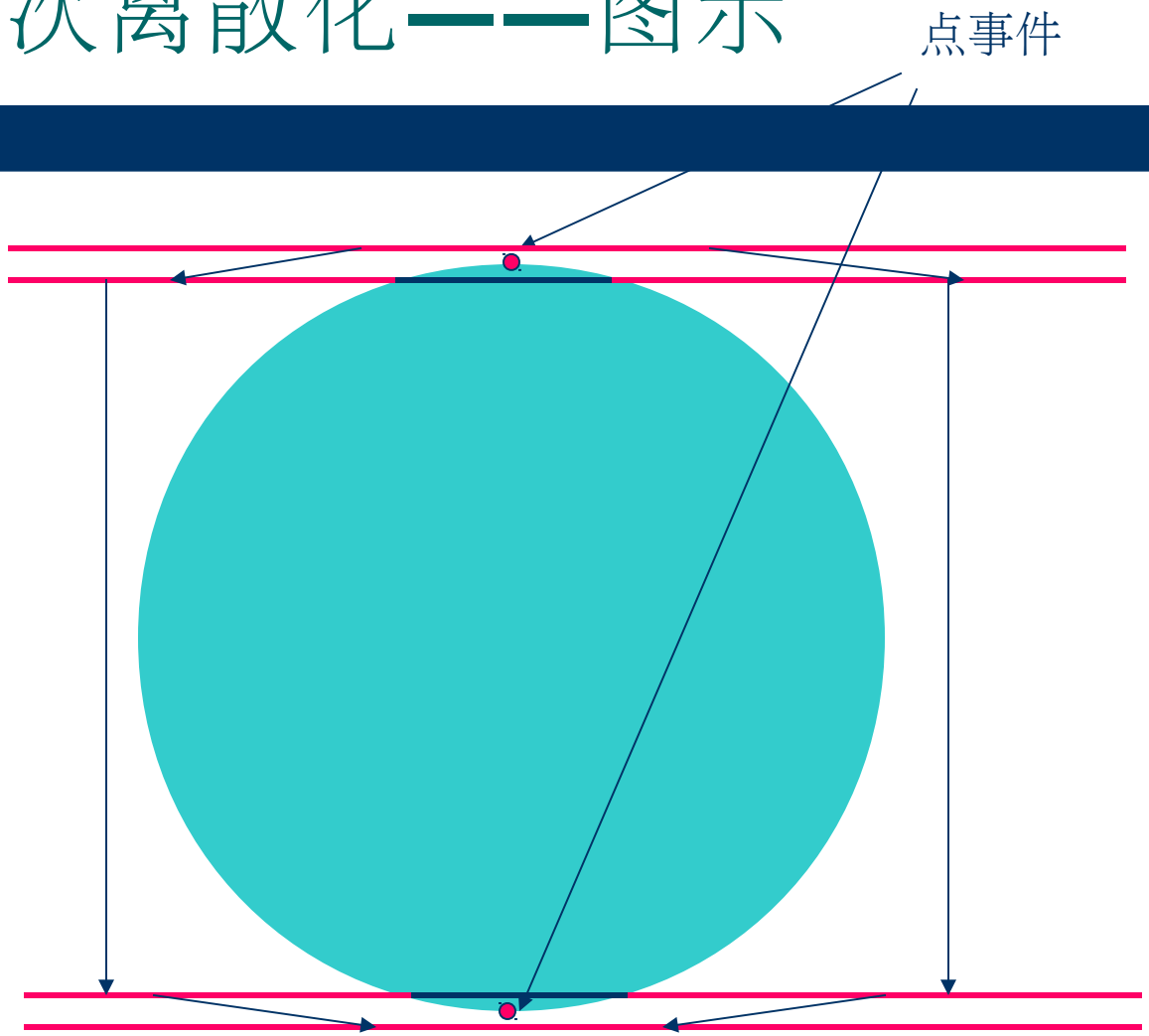
- 我们观察程序的运行，发现相邻两次循环之间，我们的程序在不断的迭代。



第二次离散化

- 事实上，区间之间的继承关系在大多数时候并没有发生改变，仅在少数时刻才会有所改动：
  - 一个圆新出现时，将一个区间分为两半。
  - 一个圆最下端，两个区间合并为一个。
  - 两圆相交，一个区间逐渐减小直至消失。
  - 两圆相交，一个新的区间生成。
- 这样，我们可以求出所有的点事件，并模拟区间的生长消亡，从而求出最终答案。
- 但是，如果完全按照上述想法编程实现，编程复杂度非常高。事实上，我们可以考虑一种懒惰的实现方式。

第二次离散化——图示



小结

- 至此，本题已被完美解决，时间复杂度为$O(N^3)$。
- 回顾我们得到算法的步骤：
  - 根据题目描述得到一种简单的算法。
  - 通过离散化不断将其时间复杂度降低，并将精度提高。
- 可以看到，使用了离散化法后，我们轻易地得到了一个简单而优秀的算法。

例 2 Empire strikes back(ural 1520)

- 平面中有 1 个大圆和 N(N<=300) 个小圆，大圆圆心为 (0,0)，小圆圆心都在大圆内。所有小圆半径相同。
- 求使所有小圆能覆盖整个大圆的小圆最小半径。

初步分析

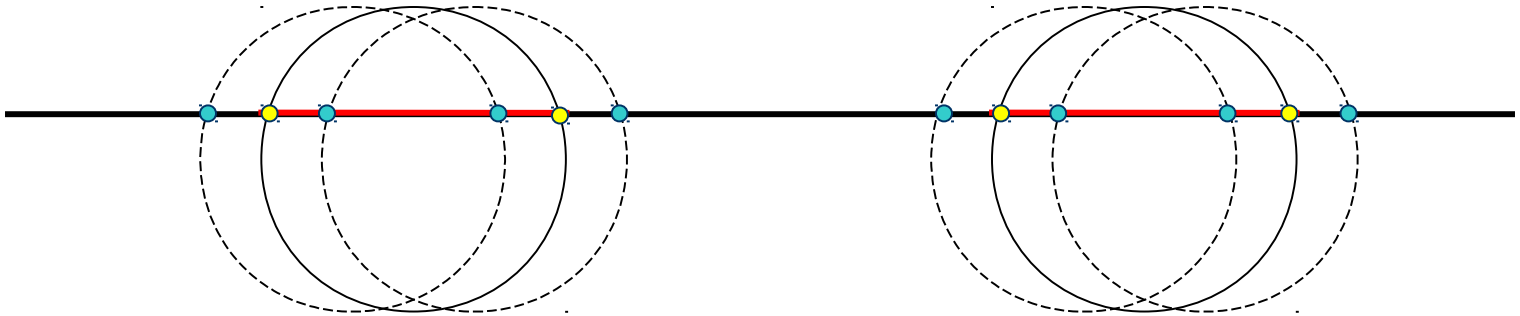
- 由于题目仅需求出一个实数，因此我们考虑使用二分法求得答案。
- 在已知小圆半径的情况下，我们所需要做的工作仅仅是判断大圆是否被小圆完全覆盖。

试探性离散化

- 如果使用和例 1 相同的离散化策略，得到的时间复杂度为 $O(N^3*k)$，k 为二分的次数，这个时间复杂度太高了！
- 之所以会出现离散化失败的情况，是因为我们没有透彻的分析问题。
- 由于已经二分了答案，所以问题已被转化为一个判定性问题，而我们仍使用原先解决计数性问题的思路解题，必然无法设计出高效的算法。

离散化算法改进

- 当然题目仅需我们判断线段是否被完全覆盖，因此很容易想到一种特殊的整体观察所有横线，考虑横线们所组成的图形(线段的左右端点)，并向左向右移动两端距离到新椭圆的和作为特殊点。
- 如果所有的特殊点都被覆盖，则整条横线一定被完全覆盖。



第二次离散化

- 注意到，每一个圆在向左向右偏移后，得到的新的圆一部分在原先的圆内，可以不做考虑。而向左向右偏移后得到的图形之和与原先的圆非常相近，可以认为是相同的。
- 于是问题转化为，其他 N-1 个小圆在这个小圆上覆盖的边界是否完全覆盖了大圆在这个小圆上覆盖的边界。
- 这和例 1 中第一步离散化时得到的问题非常相似，可以使用类似的方法解决。

时间复杂度分析

- 算法总复杂度 $O(N^2 \log N * k)$
  - 二分的复杂度 $O(k)$
  - 区间排序的复杂度 $O(N \log N)$
  - 区间合并的复杂度 $O(N)$
  - 枚举 N 个圆 $O(N)$
- 事实上，如果采用一些其他的小优化，可以将算法的复杂度进一步降低为 $O(N^2 \log N + N \log N k)$

小结

- 回顾我们得到最终算法的步骤：
  - 一开始，我们分析问题，得到了一种简单的算法。
  - 然后，我们尝试利用离散化法对其进行优化，但传统的离散化得到的时间复杂度十分不理想。
  - 于是我们进一步分析问题的特性，并通过转化问题的方式得到了一个新的离散化法，最终完美解决了此题。
- 由此可见，离散化法并不仅仅是把图形按照横纵坐标划分为若干区域。对于同一个问题，不同的离散化法得到的算法效率相差甚远。

总结及讨论

- 离散化法作为一种较为朴素的处理方法，可以解决多数与圆有关的计算几何问题。
- 成功的关键是要把握区域划分的精细程度：
  - 过细区域划分，单元内的计算简单，但整合复杂度增加。
  - 区域划分过粗，会使单元内的计算无法实现。
- 虽然这里分析的是圆，但很容易推广到其他曲线的计算几何问题。