感受随机的美

——浅谈随机化思想在几何问题中的应用

广东省中山一中 顾研

摘要

近几年来，可以使用随机化来解决的几何题目越来越多。本文将着重介绍两种在信息学竞赛中常见的随机几何算法：随机增量法与模拟退火法，以及和传统方法的比较，说明了随机化思想的优势。

关键字

随机化 随机增量算法 模拟退火算法 调整

目录

正文

1 随机算法简介

随机算法是这样的一类算法：它在接受输入的同时，在算法中引入随机因素，即通过随机数选择算法的下一步。也就是说，一个随机算法在不同的运行中对于相同的输入可能有不同的结果，或执行时间有可能不同。

随机算法的特点：简单、快速、灵活和易于并行化，这些特点都会在本文中得到体现。随机算法可以理解为在时间、空间和精度上的一种的平衡。

常见的随机算法有四种：数值概率算法，蒙特卡罗（*Monte Carlo*）算法，拉斯维加斯（*Las Vegas*）算法和舍伍德（*Sherwood*）算法。

1.1 数值概率算法

数值概率算法常用于数值问题的求解。这类算法所得到的往往是近似解。而且近似解的精度随计算时间的增加不断提高。在许多情况下，要计算出问题的精确解是不可能或没有必要的，因此用数值概率算法可得到满意的解。

举个例子：计算π的近似值时，我们可以在单位圆的外接正方形内随机撒n个点，设有k个点落在单位圆内，可以得到$\pi \approx \frac{4k}{n}$。

数值概率算法不是本文的重点，这里仅作简单介绍。有兴趣的同学可自行研究。

1.2 拉斯维加斯（*Las Vegas*）算法

拉斯维加斯算法不会得到不正确的解，也就是说，一旦用拉斯维加斯算法找到一个解，那么这个解肯定是正确的。但是有时候用拉斯维加斯算法可能找不到解。算法所用的时间越

多，得到解的概率就越高。

1.3 蒙特卡罗（*Monte Carlo*）算法

又是一个以赌城命名的算法。通常所说的蒙特卡罗算法分为两类。

1．蒙特卡罗判定：蒙特卡罗算法总是能给出问题的解，但是偶尔也可能会产生非正确的解。求得正确解的概率依赖于算法所用的时间。蒙特卡罗判定的错误必须是“单边”的，即实际答案是YES（NO），算法给出的答案可能是NO（YES），但是实际答案是NO（YES），则算法给出的答案一定是NO（YES）。因此蒙特卡罗算法得到正确解的概率随着计算次数的增加而提高，即在时间和精度上的一种平衡。

最常见的蒙特卡罗判定是众所周知的*Miller-Rabin*素数测试字符串匹配的*Rabin-Karp*算法。

2．蒙特卡罗抽样：它的基本思想是，对于所求的问题，通过试验的方法，通过大样本来模拟，得到这个随机变量的期望值，并用它作为问题的解。它是以一个概率模型为基础，按照这个模型所描绘的过程，通过模拟实验的结果，作为问题的近似解的过程。

蒙特卡罗抽样的主要步骤可参考附录1。本文第三章的模拟退火算法就使用了蒙特卡罗抽样的思想。

1.4 舍伍德（*Sherwood*）算法

舍伍德算法总能求得问题的一个解，且所求得的解总是正确的。当一个确定性算法在最坏情况下的计算复杂性与其在平均情况下的计算复杂性有**较大差别**时，可以在这个确定算法中引入随机性将它改造成一个舍伍德算法，消除或减少问题的好坏实例之间的这种差别。舍伍德算法精髓不是避免算法的最坏情况的发生，而是设法消除这种最坏行为与特定实例之

间的关联性。舍伍德算法最广泛的一个应用就是快速排序的随机化实现。

本文第二章的随机增量算法就是舍伍德算法的一种应用。

除了随机算法，国家集训队2006年论文：唐文斌《浅谈“调整”思想在信息学竞赛中的应用》中很多带有随机化的调整思想也可以用在几何问题中。比如Ural1578的Mammoth Hunt：给出平面上的n个点，找一条n-1段的折线将点连接起来，并且相邻两段折线的夹角是锐角。我们可以先任意生成一条折线，并且按一定规则进行调整，直到满足要求。

2 随机增量算法

2.1 增量算法

增量法（*Incremental Algorithm*）的思想与第一数学归纳法类似，它的本质是将一个问题化为规模刚好小一层的子问题。解决子问题后加入当前的对象。写成递归式是：

$$T(n)=T(n-1)+g(n)$$

增量法形式简洁，可以应用于许多几何题目中。

2.2 随机增量算法的一个例子

增量法的优势在于它是线性递增的，代码实现通常比较简单，但是时间复杂度相对比较高[1]，尤其在 ACM/ICPC 竞赛中，一个极限数据很容易成为算法的瓶颈，导致 TLE。因此，增量算法往往结合舍伍德算法的思想（随机），成为“随机增量算法”，在一大类问题中大大改进了平均情况下的时间复杂度。

在增量算法中加入舍伍德算法（随机化），我们要做两件事：

[1] 构造极限数据很容易。

（1） 给出一个高效的随机洗牌算法。这个问题不复杂，以下代码就可以以线性的时间复杂度得到一个 1~n 的随机排列。（记录在数组 *O* 中。）

```
Algorithm Random_shuffle
for i ← 2 to n
    交换 O[i]，O[random(i)]
```

（2） 计算随机化后 $n!$ 种等概率的顺序的时间复杂度的数学期望的平均值。这个问题将在引例中讨论。

引例：最小外接圆（经典问题）[2]

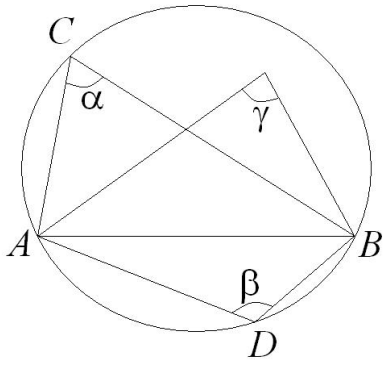
题目描述：

已知平面上 n 个点的坐标，求能够覆盖所有的点的最小圆。

分析：

本节主要讨论随机增量算法，仅在张角法前提下分析一些随机增量算法的性质。

2.2.1 张角法



首先，易知至少有两个点在圆上（除非总共只有一个点）。我们首先枚举这两个点（设为 *A*, *B*）。

初中的平面几何知识告诉我们对于同一段弧，圆周角小于圆内角（如 $\alpha < \gamma$），设 $\overline{AB}$ 上方最小的张角为 α，下方最小的张角为 β，则当 $\alpha + \beta \geq \pi$ 时，$\triangle ABC$ 或 $\triangle ABD$ 的外接圆可以覆盖所有的点。特别地，当 $\alpha + \beta = \pi$ 时 ABCD 四点共圆。

[2] 可在 zju1450 提交。

可以看到，该算法的时间复杂度为 $O(n^3)$，并且是一个确定性算法。

2.2.2 改进算法

我们用 $P=\{p_1,\cdots,p_n\}$ 表示题目中的点。考虑使用增量算法，令 $P_i=\{p_1,\cdots,p_i\}$，D_i 则代表相对于 P_i 的最小外接圆。

定理：设 $2<i\le n$，有：

（i） 若 $p_i\in D_{i-1}$，则 $D_i=D_{i-1}$；（这个非常显然。）

（ii） 若 $p_i\notin D_{i-1}$，则 p_i 必然落在 D_i 的边界上。

（ii）部分“看起来”就是正确的。但是证明部分却不是那么容易。当然这个证明不是本节的重点，有兴趣的同学可以参看附录 2。

因此我们很容易设计出基于定理的算法，每次插入 P_i 时判断其是否属于 D_i，不属于则枚举 p_i 和 $p_j(j<i)$，并用张角进行判断。

```
Algorithm MiniDisc
Random_shuffle
for i ← 3 to n
    if not (p[o[i]] in D)
*       for j ← 1 to i-1
*           以 o[i]，o[j]为圆周上的点进行求解
```

如果不使用随机化，复杂度依然是 $O(n^3)$ 的（图 2）。如果我们使用上文所描述的洗牌算法将 P 中元素的顺序打乱，复杂度如何呢？

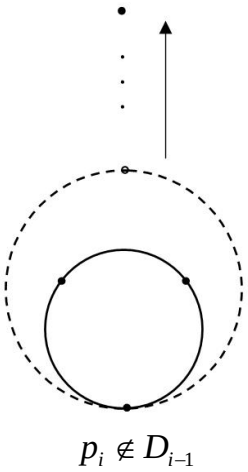


图 2

取随机变量 X_i，若 $p_i\notin D_{i-1}$，则 $X_i=1$，否则 $X_i=0$。算法在*处一共需要 $O(i^2)$ 的时间。因此，整个最小外接圆的算法复杂度就是

$$\sum_{i=1}^{n} O\left(i^2\right) \cdot X_i$$

为了界定上述和式的期望值，我们需要利用期望的线性率（*linearity of expectation*）：一组随机变量总和的期望值，等于这些随机变量各自期望值的总和，即便其中的随机变量不是相互独立的，也依然满足这个性质。因此，总的运行时间的期望值为

$$E\left[\sum_{i=1}^{n} O\left(i^2\right) \cdot X_i\right] = \sum_{i=1}^{n} O\left(i^2\right) \cdot E\left[X_i\right]$$

而 $E\left[X_i\right]$ 是多少呢？它等于 $p_i \notin D_{i-1}$ 的概率。但是这个问题比较棘手。

正难则反，我们从反方向看待这个问题。对于 D_i， p_i 在它的边界上的概率是多少呢？D_i 是由 P_i 中的 3 个点构成的，恰好是 p_i 的概率则为 $\frac{3}{i}$，即 $E\left[X_i\right] = \frac{3}{i}$。因此该算法的时间复杂度就为

$$\sum_{i=1}^{n} O\left(i^2\right) \cdot \frac{3}{i} = O\left(n^2\right)$$

最后再次声明一点，这里所指出的“期望”，是相对于洗牌算法所给出的随机的顺序而言的，并不是对各种输入数据的平均。确切地讲，这个期望的时间复杂度和输入顺序完全没有关系。

至此，我们通过随机增量算法将原问题 $O\left(n^3\right)$ 的时间复杂度成功地优化至 $O\left(n^2\right)$。

补充两点。一、此算法并不是该问题的最优算法，如果将算法的*处也作**定理**所指出的性质的优化，时间复杂度则可以进一步下降到 $O\left(n\right)$。这不是本节的重点，因此不再展开了，有兴趣的读者可以自行思考。

二、本题也可以使用本文第三节的模拟退火算法解决，并且效果很好。

2.3 随机增量算法的应用

昂贵的“饮品”[3]

题目描述：

你淘气的小妹妹喜欢将厨房里的水、咖啡、酒和糖掺在一起，并且让你帮他试尝！

你当然不愿意做，因此你决定帮她计算这种“饮品”的价格。你妹妹非常喜欢她的饮品，因此你得帮她计算出最大可能的价格。

水、咖啡、酒和糖的价格分别为 c_1, c_2, c_3, c_4，并且 $0 \le c_1 \le c_2 \le c_3$。如果你分别使用 a_1, a_2, a_3, a_4 数量的原料，饮品的价格就是 $a_1c_1+a_2c_2+a_3c_3+a_4c_4$。你妹妹告诉了你她先前配制的 n 种饮品的价格 p，每种饮品的 a_1、a_2、a_3，并且告诉你所有饮品 a_4c_4 的价格在区间[L, R]中。你的任务是推算出当前某种使用 a_1, a_2, a_3 的饮品的最大可能的价格。

数据规模：$n \le 100$，2000 组数据，2s 时限。

分析：

为了方便起见，我们用 x, y, z 代替 c_1, c_2, c_3。

题目实际上与 c_4 没有关系，对于第 i 种饮品，我们可以列出方程

$$p - R \le a_1 x + a_2 y + a_3 z \le p - L$$

同时题目要求 $0 \le c_1 \le c_2 \le c_3$，我们也可以表示成

$$0 \le x \le \infty$$
$$0 \le -x + y \le \infty$$
$$0 \le -y + z \le \infty$$

最后题目要求求出 $a_1 x + a_2 y + a_3 z$ 的最大值。全部约束整理一下就是：

$$\begin{aligned} &\text{maximize} && ax + by + cz \\ &\text{s.t.} && l_1 \le a_1 x + b_1 y + c_1 z \le r_1 \\ &&& l_2 \le a_2 x + b_2 y + c_2 z \le r_2 \end{aligned}$$

[3] 2007ACM/ICPC Regional Contest, Problem E, Expensive Drink, Beijing, Asia

……

$$l_{n+3} \le a_{n+3}x + b_{n+3}y + c_{n+3}z \le r_{n+3}$$

可以看出，这是一个有 2*n*+6 个线性约束的三维线性规划问题[4]。在赛场上遇到这样的问题，我们非得敲一个单纯形法、甚至内点法吗？答案是否定的。

2.3.1 蛮力算法

让我们回到起点，一起看看最基本的一些问题。

首先，问题的解会在哪里呢？显然可以分以下几种情况：

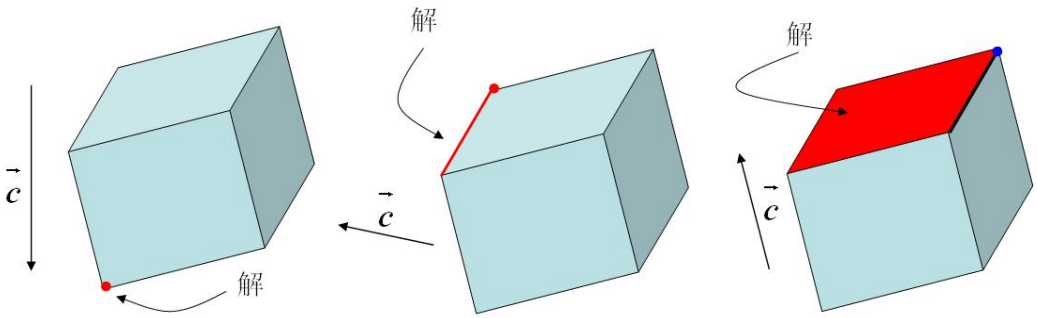
① 无解。

② 解在三个平面的交点上。（见图 3-a。）

③ 解在两个平面的公共棱上。这时我们不妨用棱与某个平面的交点来表示它。（见图 3-b。）

④ 解就是某个平面。我们可以任取它与某个平面的交点，则问题同样转化成③。（见图 3-c。）

⑤ 解无穷大。我们可以认为的加上一个非常大的框，使问题变到情况②③④。在输出答案的时候判断解是否在框上。



a) b) c)

图 3

[4] 关于线性规划问题及其解法，可参看 07 年集训队论文：李宇骞《浅谈信息学竞赛中的线性规划——简洁高效的单纯形法实现与应用》。

也就是说只要问题有解（设解集为 *ans*），则必然与三个平面交点的集合（设为 *P*）的交非空，即 $ans \cap P \neq \phi$。因此我们可以设计出一个最基础的算法：枚举三个平面求出交点，并对其余约束条件逐一判断。（如果没有交点或交点退化则可忽略。）

```
Algorithm drink_simple
  for i ← 1 to tot
  for j ← i+1 to tot
  for k ← j+1 to tot
    if 第 i，j，k 个平面存在唯一交点 int
      then flag ← true
           for l ← 1 to tot do
             if int 不满足第 l 个约束
```

很显然，这个算法的时间复杂度为 $O(n^4)$，需要优化。

2.3.2 切割线段算法

上述算法之所以没有做到高效，是因为没有用到问题的特殊之处。众所周知，线性规划问题的可行区域（核）是凸的。在本题里，核是一个凸多面体。

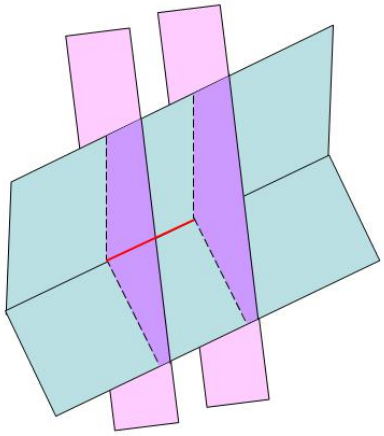


图 4

这个性质怎么利用呢？如果我们不枚举三个平面而是枚举两个，这时我们得到了一条直线。而当我们用剩余的每对[5]约束和直线求交，得到的一定是一条线段（不相交的不计）。我们将所有的约束与直线求交，最终的结果非空，该线段（可能退化成点）即可满足所有不等式。怎么更新答案呢？我们有：

定理 1 只有线段的两个端点有可能成为最优解。

证明 设线段的两个端点分别在 p_0 和 p_1，我们用 $\{P(\lambda) | 0 \le \lambda \le 1\}$ 表示线段上的点，其

[5] 形如 $l \le ax + by + cz \le r$ 的两个约束。

中 $P_0 = p_0$、$P_1 = p_1$、$\overline{P(\lambda)P_0} = \lambda\overline{P_0P_1}$。因为 $\{P(\lambda)\}$ 是空间中的一条直线，求导后，*x*、*y*、*z* 对于 λ 的变化量都是常数。设 $C(\lambda)$ 为 $P(\lambda)$ 的目标函数的值，则 $C'(\lambda)$ 也为常数。不妨设 $C'(\lambda) = k$。

若 *k*=0，则线段上每个点的解相同。

若 *k*>0，则 $C(1)$ 是线段上的最优解。

若 *k*<0，则 $C(0)$ 是线段上的最优解。

定理 2 不会有某三个平面的交点被遗漏。

证明 枚举两个平面，并用其余约束切割公共棱，设线段的两个端点分别在 p_0 和 p_1，设该三个平面的交点为 *p*，分三种情况：

① *p* 为 p_0 或 p_1 中的一个。已枚举到。

② *p* 在 $\overline{p_0p_1}$ 上。由**定理 1** 知必然不会比 p_0 和 p_1 的解中较大的优。

③ *p* 不在 $\overline{p_0p_1}$ 上。则必然有某个（些）约束条件不能满足，解不合法。

至此，我们已经证明了算法的正确性。下面给出伪代码。

```
Algorithm cutting
for i ← 1 to tot
for j ← 1 to tot
   求出平面 i，j 的公共棱
      for k ← 1 to n+3
         用第 k 对约束切割直线，求并
```

下面讲一下实现的问题，也就是切割的具体实现。因为空间中的直线情况比较多、比较复杂，因此我们可以使用参数方程进行统一表示。

$$\begin{cases} x = x_0 + x_1 \cdot t \\ y = y_0 + y_1 \cdot t \\ z = z_0 + z_1 \cdot t \end{cases}$$

这样，我们对直线的切割就转化成为对于参数值求交的过程。

最后是求解参数方程的过程。首先我们假设枚举的两个平面不平行，我们任意消去 *x*、*y*、*z* 中的一个，得到一个二元（一元）一次方程。取任意一个自由元的方程的系数，经过两次回代即可求出直线的参数方程。

该算法的时间复杂度为 $O(n^3)$，还需要优化。

2.3.3 随机增量算法

2.3.2 的算法不能让我们满意的原因是：将一条条直线割裂开来进行讨论，并没有利用到与之前已经计算的结果之间的关系。如果我们使用增量算法，假设已经计算了满足前 *n* 个线性约束的核为 C_n，最优解为 v_n，加入第 *n*+1 个半空间 h_{n+1} 的时候，存在两种情况：

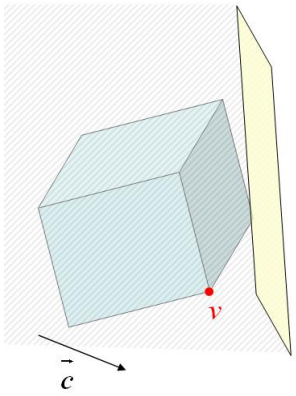


图 5

① $v_n \in h_{n+1}$，则 $v_{n+1} = v_n$。（图 5。）

② $v_n \notin h_{n+1}$，则如果 $h_{n+1} \cap C_n = \phi$，则至此无解；否则 v_{n+1} 必然是第 *n*+1 个平面上的点。

以上两种情况用反证法均易证。

通过以上结论，我们把 2.3.2 的线段切割算法又通过增量算法实现了。

```
Algorithm Incremental
*
设 v 为无穷大
for i ← 3 to tot
   if v 不满足第 i 个约束
   then  设 v 为无穷大
         for j ← 1 to i-1
            求出平面 i，j 的公共棱
            for k ← 1 to i
```

现在算法的时间复杂度是多少呢？我们不难得到图 6 的数据，此时的复杂度依旧为 $O(n^3)$。不过造成复杂度高的原因已经从算法本身转移到了输入数据的顺序上了。此时我们不难联想到 2.2 中的随机洗牌，如果我们在算法*的位置加入随机的过程，打乱枚举半空间的顺序，必然能使复杂度大大降低。

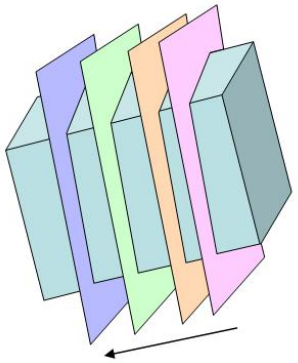


图 6

我们还是取随机变量 X_i，若 $v_i \notin h_{i-1}$，则 $X_i = 1$，否则 $X_i = 0$。算法复杂度为

$$\sum_{i=1}^{n} O(i^2) \cdot X_i$$

对于 h_i，$v_i \notin h_{i-1}$的概率是多少呢？h_i 是由 3 个线性约束的平面的交点构成的，恰好是 h_i 的概率自然是 $\frac{3}{i}$，即 $E[X_i] = \frac{3}{i}$。因此该算法的时间复杂度就为

$$\sum_{i=1}^{n} O(i^2) \cdot \frac{3}{i} = O(n^2)$$

至此，本问题得到完美的解决。随机的使用，使得一个原本普通的方法在性能上有了质的飞跃，让我们领略到随机的美。

我们还欣喜地发现，在三维的线性规划中，本算法的运算速度竟大大优于单纯形法。

要补充的一点是：目前世界上先进的研究成果可以将 *n* 维线性规划转化为 *n*-1 维。解决 1 维线性规划的时间复杂度为$O(n)$，所以这题理论上存在$O(n)$复杂度的方法。但是该算法有两点弊病：

① 时间复杂度中隐藏的常数巨大。本题中在时间上的优势微小。（*n* 仅 100。）

② 编程复杂度过大。其实$O(n)$的算法并不难想：每次加入一个半空间后，如果先前的解不成立需要更新，此时就是要将目标向量在平面上的投影作为新的目标向量，将其他半空间转换成半平面做一次二维线性规划[6]。几次空间和平面间的转换与旋转，将该算法仅仅保留在理论上。

我们使用随机思想是希望事半功倍、化繁为简，本算法显然有悖于我们的初衷。而且无论在信息学还是 ACM 赛场上比赛的时间都是有限的，因此本算法并不值得推广。

[6] 二维线性规划的线性算法可参考《算法艺术与信息学竞赛》P417。

3 模拟退火（*Simulated Annealing*）算法

在很多信息学竞赛选手看来，很多时候几何题目就是代码量大的代名词，即使对于一些经典问题，庞大的代码量也使很多人望而却步。模拟退火算法思维及编写简单、灵活，可以在一类最远、最近或第 *k* 近距离问题中发挥威力。

3.1 模拟退火算法介绍

3.1.1 模拟退火算法的原理

模拟退火算法是一种元启发式（*Meta-Heuristics*）算法，来源于固体退火原理，将固体加热至充分高的温度，再让其徐徐冷却。加热时，固体内部粒子随温升变为无序状，内能增大，而徐徐冷却时粒子渐趋有序，在每个温度都达到平衡态，最后在常温时达到基态，内能减为最小。根据 *Metropolis* 准则，粒子在温度 *T* 时趋于平衡的概率为 $e^{-\frac{\Delta E}{kT}}$，其中 *E* 为温度 *T* 时的内能，Δ*E* 为其改变量，*k* 为 *Boltzmann* 常数。

3.1.2 模拟退火算法的模型

① 初始化：初始温度 *T*(充分大)，初始解状态 *S*(算法迭代的起点)， 每次迭代次数 *L*

② *for k*=1 *to L* 做③至⑥

③ 产生新解 *S'*

④ 计算增量Δ*t'*=*C*(*S'*)-*C*(*S*)，其中 *C*(*S*)为评价函数

⑤ 若Δ*t'*<0 则接受 *S'*作为新的当前解，否则以概率 $e^{-\frac{\Delta t}{T}}$ 接受 *S'*作为新的当前解

⑥ 如果满足终止条件则输出当前解作为最优解，结束程序

⑦ *T* 逐渐减少，然后转②

3.2 例一：Run Away[7]

题目描述：

平面上有一个矩形，在矩形内有一些陷阱。求得矩形内一个点，该点离与它最近的已知陷阱最远（点的个数≤1000）。精度要求：$\varepsilon = 10^{-1}$。

分析：

本题在 2007 年集训队论文：王栋《浅析平面 Voronoi 图的构造及应用》中出现过。文中给出了一个$O(n\log n)$的算法。下文的一些分析也摘自该文。

首先解可能出现在两种位置：

①过三陷阱的圆的圆心　　②两个陷阱的中垂线与矩形的边的交点

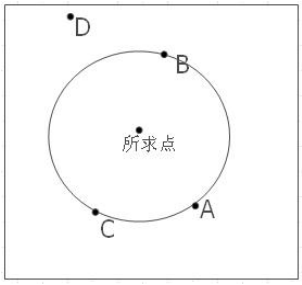


图 7

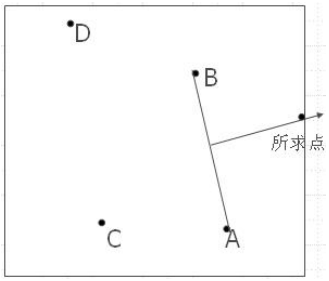


图 8

简单的枚举法的时间复杂度为$O(n^4)$，不能满足题目要求。

在本题中，所有可行解对应 Voronoi 边的交点，或 Voronoi 边与边界的交点，当然还有矩形的 4 个顶点。因此一个直接的想法就是求出 Voronoi 图解。然而，Voronoi 图$O(n\log n)$的经典构造方法 Fortune 算法或者使用 Delaunay 三角剖分，动辄 300 行的代码量，让人裹足不前。即使是$O(n^2\log n)$的算法也需要用到分治法的半平面交。

[7] Central Europe 1999, pku1379

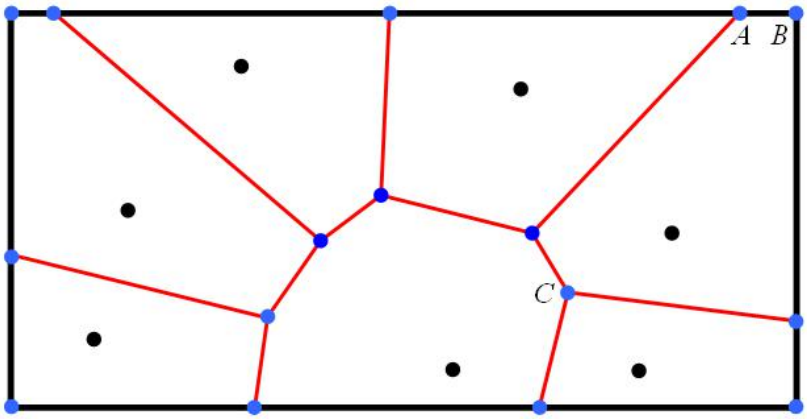


图 9

如果我们从另一个角度分析：这题的精度要求不高，一个很朴素的想法是枚举平面中的一些网格点并进行判断。当然这样的点太多了，我们必须选一些比较有“希望”的点，同时还不能忽略任何平面上的任何一点。

我们使用类比的方法引入模拟退火算法，初始解状态 *S* 可以在矩形内随便选取，初始温度 *T* 对应于每次调整的距离Δ，产生新解的方式是在目前解为圆心、半径为Δ的圆周上任取一点，评价函数取距离最近的陷阱的距离，终止条件为Δ足够小。

由此可得本问题的模拟退火算法：由初始解 *S* 和控制参数初值Δ开始，对当前解重复“产生新解→计算目标函数差→接受或舍弃”的迭代，并逐步衰减Δ值，算法终止时可以得到一组解，这是基于蒙特卡罗迭代求解法的一种启发式随机搜索过程。退火过程由冷却进度表（*Cooling Schedule*）控制，包括控制参数的初值Δ及其衰减因子Δ’、每个值Δ时的迭代次数 *L*。

模拟退火算法还有一个特点：具有并行性。因此我们可以将初始解 *S* 改成初始解集，对于每个候选解都进行迭代，答案取最终解集的最优解。

```
Algorithm SA
① 在矩形中取 p 个点作为初始解
② 取一个足够大的步长 delta
while delta>epsilon
  for 每个候选解
    for i ← 1 to L
      随机一个模长为 delta，方向随机的向量加到点上作为新解
```

由于我们必须保证能覆盖矩形上的每个位置，因此在①中确定 p 后（我们按网格状放置），②中的 delta 可取 $\frac{\max\{矩形的边长\}}{\sqrt{n}}$。设算法的迭代次数为 t，则算法的复杂度为 $O(p \cdot L \cdot n \cdot t)$。

使用本算法，p 取 10，L 取 30。我们发现可以轻松地通过全部测试数据。那么这个算法的正确性如何呢？

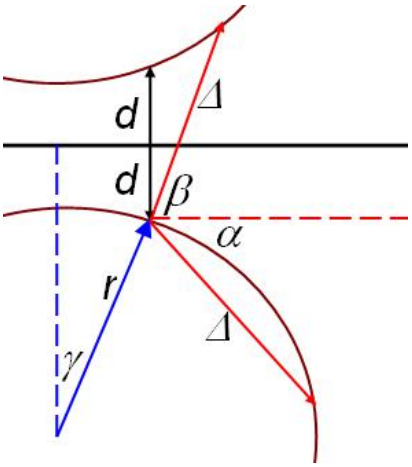
感性认识：

最优解之所以能成为最优解，因为它附近（如点 A，B）的陷阱非常稀少且距离很远，因此有点在它周围（所在的 Delaunay 三角剖分区域内）的概率是很大的。而且此时的距离比较大，我们对方向进行多次尝试，因此调整出去的概率也很小。

理性认识：

前面说明了候选解的存在性，我们同时还必须保证调整过程候选的解都要向更优的位置进发。当然，一开始的大范围调整主要是向比较优的解逼近，为了覆盖更多的点。我们着重分析一下调整范围较小（$\Delta < 2r$）时候的调整情况。

如图 10 所示，假设点到中垂线（Voronoi 边）的距离为 d，到下方陷阱距离为 r，与垂线夹角 γ，目前调整长度为 Δ，一次随机调整成功的概率为 P，则 $P \geq \frac{\alpha+\beta}{2\pi}$。



（有可能向左调整）。

$$\alpha = \pi - \left(\frac{\pi}{2} - \gamma\right) - \arccos\left(\frac{\Delta}{2r}\right)$$

$$= \frac{\pi}{2} + \gamma - \arccos\left(\frac{\Delta}{2r}\right)$$

图 10

$\beta \geq \alpha$（当 $d = 0$ 时 $\beta = \alpha$）

$$\Rightarrow P \geq \frac{1}{2} + \frac{\gamma}{\pi} - \frac{\arccos\left(\frac{\Delta}{2r}\right)}{\pi}$$

若我们的 L 取 30，则经过 L 次随机后随机调整成功的概率为 P_t。

我们的 Δ 每次乘以 0.8，因此 50%的调整成功率就足够了。

令 $P_t \geq 50\%$，求得 $P \geq 2.7\%$，即 $\frac{\Delta}{r} \geq \cos(0.473\pi + \gamma)$。这里若 $0.473\pi + \gamma \geq 1$，则调整可以满足。

当 $\gamma = 0$ 时，Δ 取到最大值 $0.085r$，因为此时两者垂直，因此 Δ 对于答案的影响很小。另外我们还必须清楚：这里我们把 γ、d 都当成 0 计算，而事实上却不会是 0，同时这两者对于调整的概率的影响还是很大的。

但是如果题目的精度要求非常高，怎么办呢？算法是死的，人是活的，既然很难随机到向量和 Voronoi 边平行，我们可以直接枚举平行于 Voronoi 边的向量，虽然在时间上付出一点代价，但是在调整成功的概率和解的精度将大大提高。当然对于普通的题目（本节三道例题），普通的随机调整就可以了。当然你可以怀疑本题的数据比较弱，精度要求比较低，但是本算法在下一题中的表现，可以打消你的顾虑。

3.3 例二：Empire strikes back[8]

题目描述：

Saddam 的国度是半径为 R 的圆，圆心在(0,0)。其中有 N 个化学武器工厂，坐标分别

[8] Ural State University Online Judge 1520

为（X_i, Y_i），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盖，问题是炸弹的最小杀伤半径为多少。

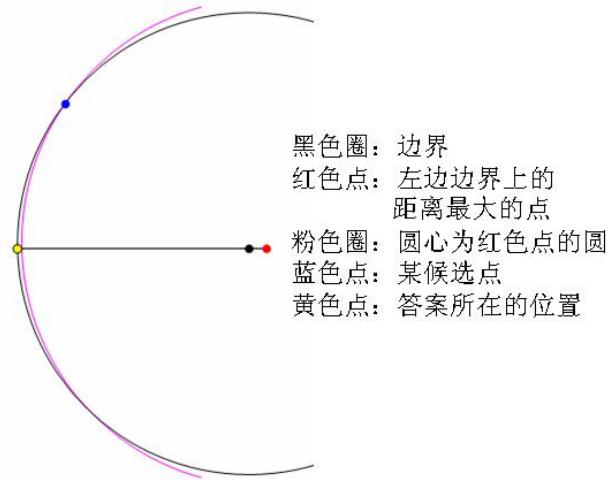
数据范围：1≤n≤300，1≤R≤1000。精度要求：保留 5 位小数。

分析：

这道题目是 Ural 的经典难题之一，曾出现在 2007 年国家集训队论文：高逸涵《与圆有关的离散化方法》中，文中给出了一个复杂度为$O\left(n^2 \log n \log R\right)$的优秀算法，但是本题的时间限制比较苛刻，有超时的可能。笔者目前知道方戈同学在本算法上进行了改进，使复杂度减少一维，并通过测试。有兴趣的同学可以和他进行交流。

仔细分析，可知本题只是将 3.2 例一中的边框从矩形变成了圆，其他并无太大区别。因此我们还是可以使用 Voronoi 图解。因为这题的数据范围稍小，因此$O\left(n^3\right)$的算法如果实现得很好，也是可以 Accepted 的。

既然和例一类似，我们是否可以直接套用同样的随机调整呢？两题唯一的区别就在于边界的情况。但是恰恰就在这里，单纯使用模拟退火算法有一点问题。



如图 11 所示，候选解一旦比较接近边界，无论调整的长度如何，调整成功的概率都非常低。而在这里一点的误差都是直接加在答案上。（不同于例一中两者垂直，基本没有影响。）因而候选点很有可能一旦接近边界就动弹不得。

好在这个问题一点都不难处理，答案在边界上无非两种情况：

① 在工厂所处直径与圆的交点。

② 某条 Voronoi 边（两点中垂线）与圆的交点。（图 12。）

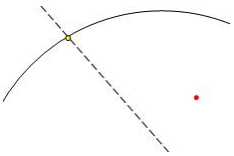


图 11

图 12

两者分别是线性数量级和平方数量级的。加之$O(n)$的判断，可以在$O(n^3)$的时间复杂度内完成，而且常数比较小，不会使用很多时间，压缩主算法的时限。

加上特判，模拟退火算法就可以通过所有的官方测试数据。即使本题精度要求非常高，对于人工构造的一些极限数据也可以出解。

在 ACM 比赛中使用模拟退火算法风险还是比较大的，因为一旦有一个数据的分析不到位，很容易就导致整道题目的全军覆没，而且在参数（冷却进度参数）的把握上也要下一番功夫。但是在单组测试数据为主的 OI 比赛中，尤其一些根据解的质量评分的题目，正是模拟退火算法表现的舞台。

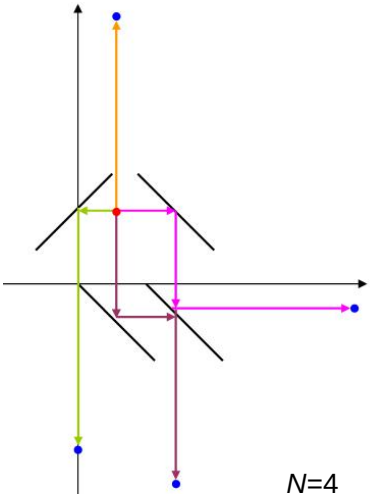
3.4 例三：激光坦克[9]

题目描述：

在平面上有 *N* 个坦克，抽象为点；*M* 个镜子，抽象成线段。激光发射器可以向任何方向发射激光。激光碰到镜子后发生反射，经过不超过 *k* 次反射的激光碰到坦克就能将其摧毁。

你的任务是架设一台激光发射器，在满足摧毁所有敌方坦克的前提下，最长的一条激光攻击路径长度最短。

本题是提交答案式题目，根据你所给出的解的优劣评分。



N=4
M=4
k=2

图 13

[9] CTSC2007

分析：

乍一看题目，感觉完全无从下手，我们脑海中那些常用的几何算法都和题目没有关系。

这题是提交答案式题目，没有思路时不妨先观察一下数据的特点，看看是否能得到启发。

Testcase	*k*	*N*	*M*
6	0	200000	0
2	1	2	50
3	1	2	1000
7	1	10000	3
5	2	3	30
8	2	3	70
1	3	4	4
4	3	2	35
9	3	18	30
10	5	25	100

我们把测试数据以*k*为关键字排序，得到了不少有用的信息：

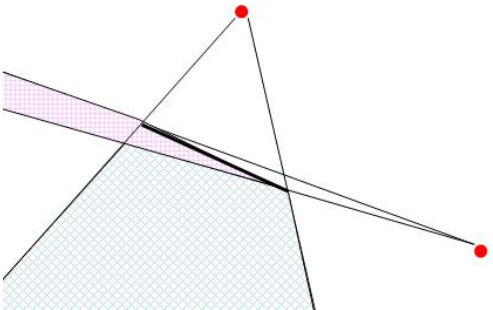
① *k*很小，除数据10外都有 $k \le 3$。

② 除数据1、2规模较小外，对于相同的*k*，*N*+*M*的值比较接近，但不同的*k*之间则相差若干个数量级。

很显然，测试数据暗示我们采用分段处理的程序，对于不同的*k*，进行不同处理。同时本题的计分函数 $C_1 + \left\lceil \frac{(Best - C_2 \cdot Ans) \cdot (10 - C_1)}{(1 - C_2) \cdot Ans} \right\rceil$ 保证只要出解就有C_1的底分，并大致按照解与官方解答的比例向上取整，可见公式还是比较宽松的，因此我们要力保出解。

⑴ k=0：

这时因为没有镜子的存在（数据6），我们可以使用最远点Voronoi图求解。但是数据规模过于庞大，因此应该不会有人在比赛中写一个基于分治法的 $O(n \log n)$ 构造最远点Voronoi图解的



方法去获得10分。本题和先前的题目相比，只是最近距离改成了最远距离，但这丝毫不影响模拟退火算法的使用。

加入镜子，相当于把平面中的一些区域标志为不可到达（图14），并不影响图的几何性质，因此我们可以一并处理。

```
Algorithm Check-No-Reflection(point p, int i)
for j ← 1 to m do
    if 线段 p-tank[i]与镜子 j 相交
```

```
Algorithm Count(point p)
dist ← 0
for i ← 1 to n do
    dist ← max(dist, Check-No-Reflection(p, i))
```

图 14

我们还是使用3.1中的算法框架，加入上面的判断的过程就可以了，算法复杂度为 $O(p \cdot L \cdot n \cdot t \cdot (1+m))$。但是第6、7组数据中的*N*非常大，因此，模拟退火算法的另一个优势——灵活，便体现出来了。因为解集的大小和每轮迭代的次数是我们自己设定的，也就是说算法的运行时间是可以控制的。我们可以先适当牺牲解的质量，找到最优解的大体位置，然后再在最优解附近再次求解即可。（当然如果不放心的话多迭代几次也没关系。）

Testcase	K	得分
6	0	10
2	1	10
3	1	0
7	1	1
5	2	10
8	2	6
1	3	9
4	3	10
9	3	0
10	5	0

总计		56

红色表示得到的解小于等于官方解答。（许多官方解答非最优解）

我们试着实现一下，其中输入输出等代码约为60行，再加上100行左右的模拟退火的过程（包括一些基本的几何模块），最理想的情况下可以得到56分，是一个性价比很高的算法。（参加过的同学可能都有体会，CTSC的每一分得起来都是那么艰难……）当然如果大家仍有时间、有能力，还可以进行进一步的挖掘。

⑵ k=1：

此时我们要处理一次的反射。不过有个很好性质，就是激光发射器确定后，**不经过反射的路径必然比发生反射的路径短**。因此我们可以在情况⑴中略加修改，并加入一个过程。

```
Algorithm Check-One-Reflection(point p, int i)
dist ← ∞
for f ← 1 to m do
  if 存在 p 经过镜子 f 的反射到 tank[i]的路径
    inter ← 路径与镜子 f 的交点
    ok ← true
    for j ← 1 to m do
      if 线段 p-inter，inter-tank[i]与镜子 j 相交
        then ok ← false; break
```

```
Algorithm Count(point p)
dist ← 0
for i ← 1 to n do
  temp ← Check-No-Reflection(p, i)
  if temp=∞
    then dist ← max(dist, Check-One-Reflection(p, i))
    else dist ← max(dist, temp)
```

加上这两段代码，以及一些反射、求线段交点等过程总计约80行后，算法的时间复杂

度为$O\left(p\cdot L\cdot n\cdot t\cdot m^2\right)$（这里不存在$m$=0），所幸我们依旧可以通过参数来控制程序的运行时间。我们看一下运行的实际效果。

Testcase	K	得分
6	0	10
2	1	10
3	1	10
7	1	10
5	2	10
8	2	10
1	3	10
4	3	10
9	3	0
10	5	0
总计		80

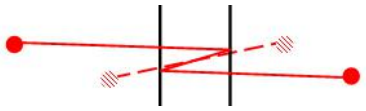
k≤1的时候程序当然可以得到最优解，最后两组数据仍然无法出解。然而令我们感到兴奋的是第1、4、5、8组数据中有的结果并不逊于官方解答，有的虽然不及官方解答，但是通过算分公式向上取整后也能得到满分。当然在真正的比赛中，心情肯定比较紧张，不一定能将所有数据的当前的最优解跑出来(参数设置的问题，多次随机取最优值等，笔者在CTSC中实现这个算法的得分是63分)， 但是240行左右的代码，60~80的得分应该能让多数人满意，毕竟有半数的集训队队员在这次比赛中的成绩不足25分。

从这张成绩表中可以看出，再进行k≥2的分析，意义并不大。但是作为论文，本着科学严谨的思想，笔者还是对于k=2的情况进行了编码实现。

⑶ k=2：

k=2的时候情况就复杂多了，这体现在两个方面：

a. 两次反射的求解和判断。我们不仅要求出反射的坐标，还要判断相交顺序的先后，稍不留心就会出现图15中的荒谬情况。同时三条线段的 图 15 两条麻



烦不少。

b. 没有情况(2)中优美的性质，因为反射一次不一定优于反射两次，因此我们不得不对Count过程进行修改，先将每种情况的距离统统计算出来再取最小值。同时在输出的部分也要做较大的改动。

所幸的是加入诸多代码同时对程序进行了很多改动后，程序能对第9组数据出解了。不过在真正的比赛中为了区区10分对程序大动干戈是很不划算的。

(4) k≥3：

我们只剩下第10组数据了，图16是使用VB生成出来的示意图。可以看到，并没有什么明显的位置可供我们放置激光发射器。同时我们的算法复杂度是$O\left(p\cdot L\cdot n\cdot t\cdot m^{k+1}\right)$，这组数据中m达到100之大，即便想做，也恐怕运行不出结果。(这组数据很有可能是反向出的，先找到路径，再添加镜子。)

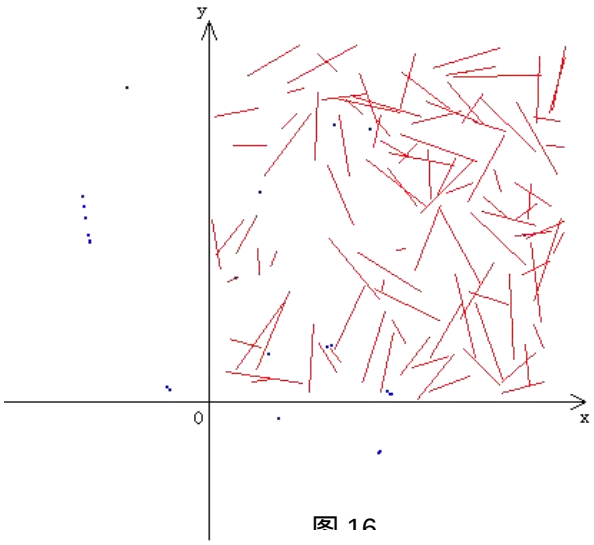


图 16

小结：

我们把三个表格进行汇总。

Testcase	*k*	程序 1	程序 2	程序 3
6	0	10	10	10
2	1	10	10	10
3	1	0	10	10
7	1	1	10	10
5	2	10	10	10

8	2	6	10	10
1	3	9	10	10
4	3	10	10	10
9	3	0	0	10
10	5	0	0	0
总得分		56	80	90
代码长度	60	160	240	300

可以看出，前两个程序的性价比比较高，尤其是程序1，思维难度和代码量都不是太大，得分还算可观。如果时间充裕，或其他题目没有思路，程序2也可以考虑，而程序3相对就不推荐了。

3.5 小结

这种模拟退火算法可以推广到一些最大、最小、第 *k* 小距离的问题甚至距离和的最大、最小的问题（比如 zju1901, A Star not a Tree?），尤其是在一些函数值随位置连续变化的问题里。比如说，大家不一定会最远点 Voronoi 图的构造算法，*k* 阶 Voronoi 图的构造算法，但是模拟退火算法都可以在一定程度上解决这些问题。同时，模拟退火算法有着随机算法共有的优点：简单（主过程很短小，实现也很简单）、快速、灵活和易于并行化。

经常做网上题库的同学一定知道 Ural 这个网站的测试数据是相当严谨的，因此从侧面反映了解的质量还是比较高的。当然笔者提出这个算法，肯定还有分析涵盖不到位的地方，对于算法的确定性以及应用范围等问题，欢迎大家与我讨论。

4 总结

近几年，随着信息学的发展，几何题目不再是经典问题的代码比赛了，各种各样灵活多变的问题层出不穷。随机算法以其简单、快速、灵活和易于并行化的特点，容易在时间、空间和精度三者之间实现平衡。希望本文的思想能为大家打开一扇窗，在遇到几何问题的时

候多一种思路。当然，随机化思想的灵活运用，是在对于经典问题的熟练掌握的前提下的，因为创新永远都是建立在扎实的基础之上的。

感谢

感谢刘汝佳教练的指导。

感谢和俞华程同学在多道题目中的讨论。

感谢在中山一中信息学竞赛组多年的训练和讨论。

感谢 Internet 的提供的资源。

参考文献

1.《算法引论——一种创造性方法》 [美]*Udi Manber* 著

2.《算法设计技巧与分析》 [沙特]*M.H.Alsuwaiyel* 著

3.《算法艺术与信息学竞赛》 刘汝佳、黄亮著

4．Computational Geometry Algorithms and Applications: *M. de Berg* 等 著

5．国家集训队 2006 年论文：唐文斌《浅谈“调整”思想在信息学竞赛中的应用》

6．国家集训队 2007 年论文：王栋《浅析平面 Voronoi 图的构造及应用》

7．国家集训队 2007 年论文：高逸涵《与圆有关的离散化方法》

8．国家集训队 2007 年论文：李宇骞《浅谈信息学竞赛中的线性规划——简洁高效的单纯形法实现与应用》

附录

附录 1　蒙特卡罗抽样的步骤

蒙特卡罗抽样主要分为三个主要步骤：

1． 构造或描述概率过程。

2． 实现从已知概率分布抽样。

3． 建立各种估计量。

构造或描述概率过程：

对于本身就具有随机性质的问题，主要是正确描述和模拟这个概率过程，对于本来不是随机性质的确定性问题，比如计算定积分，就必须事先构造一个人为的概率过程，使它的某些参量正好是所要求问题的解。即要将不具有随机性质的问题转化为随机性质的问题。

实现从已知概率分布抽样：

构造了概率模型以后，由于各种概率模型都可以看作是由各种各样的概率分布构成的，因此产生已知概率分布的随机变量（或随机向量），就成为实现蒙特卡罗方法模拟实验的基本手段，这也是蒙特卡罗方法被称为随机抽样的原因。随机数就是具有这种均匀分布的随机变量。随机数序列就是具有这种分布的总体的一个简单子样，也就是一个具有这种分布的相互独立的随机变数序列。产生随机数的问题，就是从这个分布的抽样问题。在计算机上，可以用物理方法产生随机数，但代价昂贵，不能重复，使用不便。另一种方法是用数学递推公式产生。这样产生的序列，与真正的随机数序列不同，所以称为伪随机数，或伪随机数序列。不过，经过多种统计检验表明，它与真正的随机数，或随机数序列具有相近的性质，因此可把它作为真正的随机数来使用。由已知分布随机抽样有各种方法，都是借助于随机序列来实现的，也就是说，都是以产生随机数为前提的。由此可见，随机数是我们实现蒙特卡罗模拟

的基本工具。

建立各种估计量：

一般说来，构造了概率模型并能从中抽样后，即实现模拟实验后，我们就要确定一个随机变量，作为所要求的问题的解，我们称它为无偏估计。建立各种估计量，相当于对模拟实验的结果进行考察和登记，从中得到问题的解。

附录 2　2.2.2 定理的证明

设 P 为平面上的一个点集；设 R 为另一个（允许为空的）点集，而且 $P \cap R = \phi$；设 $p \in P$，则下列命题成立：

① 如果存在某个圆覆盖了 P，而且其边界穿过 R 中的所有点，那么这样的圆中必然存在唯一的最小者。我们将其记作 $md(P,R)$。

② 如果 $P \in md(P \setminus \{p\}, R)$，那么 $md(P,R) = md(P \setminus \{p\}, R)$。

③ 如果 $p \notin md(P \setminus \{p\}, R)$，那么 $md(P,R) = md(P \setminus \{p\}, R \cup \{p\})$

证明 ① 如图 17 所示，假设存在半径相同的两个不同的覆盖圆 D_0 和 D_1，其圆心分别在 x_0 和 x_1。显然 P 中的所有的点必然落在交集 $D_0 \cap D_1$ 中。我们将按照下面的方法，构造出一系列连续的圆 $\{D(\lambda) | 0 \le \lambda \le 1\}$。取 D_0 和 D_1 的边界的一个交点 z，$D(\lambda)$ 的圆心是点 $x(\lambda) = (1-\lambda)x_0 + \lambda x_1$，其半径为 $r(\lambda) = d(x(\lambda), z)$，对于任何满足 0≤λ≤1 的λ，都有 $D_0 \cap D_1 \subset D(\lambda)$，因此作为一个特例，

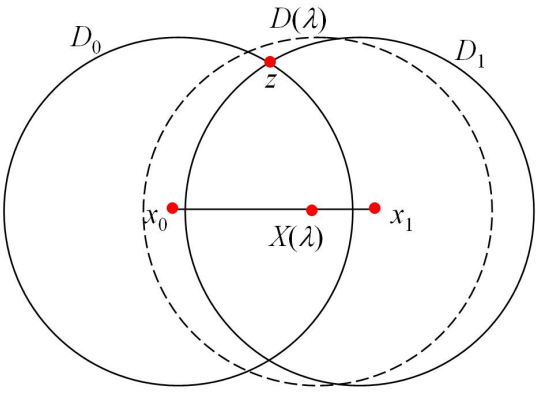


图 17

λ=0.5 时也成立。于是鉴于 D_0 和 D_1 都分别覆盖了 P 中的所有点，故 $D\left(\frac{1}{2}\right)$ 也必然如此。此外 $\partial D\left(\frac{1}{2}\right)$ 也必然经过 ∂D_0 和 ∂D_1 的每个交点。由于 $R \subset \partial D_0 \cap \partial D_1$，故有 $R \subset \partial D\left(\frac{1}{2}\right)$。也就是说，$D\left(\frac{1}{2}\right)$ 不仅是 P 的一个覆盖圆，而且其边界同样会穿过 R 中的所有点。然而，就半径而言，$D\left(\frac{1}{2}\right)$ 要严格小于 D_0 和 D_1。因此，一旦出现了半径相等的两个圆，而且它们的边界都各自穿过 R 中的所有点，就说明肯定存在另一个半径更小的覆盖圆，而且它们的边界都各自穿过 R 中的所有点。于是，最小覆盖圆 $md(P,R)$ 是唯一的。

② 令 $D = md(P \setminus \{p\}, R)$，若 $p \notin D$，则 D 必然包含 P，而且其边界穿过 R 中的所有点，不可能有任何更小的圆可以覆盖 P，而且其边界穿过 R 中的所有点。否则，这样的一个圆必然是 $P \setminus \{p\}$ 的一个包围圆，同时其边界穿过 R 中的所有点，这与 D 的定义矛盾。因此可以得到 $D = md(P,R)$。

③ 令 $D_0 = md(P \setminus \{p\}, R)$，取 $D_1 = md(P,R)$。再次考虑前面所定义的一系列 $D(\lambda)$。注意到，$D(0) = D_0$，$D(1) = D_1$。实际上，由这一系列的圆，定义了从 D_0 到 D_1 的一个连续变形的过程。根据假设，有 $p \in D_1$。因此，由其连续性，必然存在某个 $0 < \lambda^* \le 1$，使得 p 正好落在 $D(\lambda^*)$ 的边界上。与①的证明同理，我们可以得到 $P \subset D(\lambda^*)$ 和 $R \subset \partial D(\lambda^*)$，对于任何 0<$\lambda$<1，$D(\lambda)$ 的半径必然会严格地小于 D_1 的半径。然而根据其定义，D_1 是 P 的最小覆盖圆，因此我们只有一种选择——$\lambda^* = 1$。也就是说，D_1 的边界必然穿过 p。

附录 3　论文原题

昂贵的“饮品”题目描述

There are some water, milk and wine in your kitchen. Your naughty little sister made some strange drinks by mixing them together, and then adds some sugar! She wants to know whether they taste good, but she doesn't want to try them herself. She needs your help.

Your sister knows that you don't want to drink them either (anyone wants to?), so she gives you a chance to escape: if you can guess the price of a special drink, she gives you freedom. Warning: she loves her special drink so much that you should never under-estimate its cost! That is, you're to find the most expensive possible price of it.

The price of each drink equals to its cost. If the amounts of water, milk, wine and sugar used in the drink are a_1, a_2, a_3 and a_4 respectively, and the unit costs of water, milk, wine and sugar are c_1, c_2, c_3 and c_4 respectively, then the drink costs $a_1c_1+a_2c_2+a_3c_3+a_4c_4$. To give you some hope to win, she told you the costs of exactly n ordinary drinks. Furthermore, she promised that the total cost of sugar a_4c_4 is always a ***real number*** in the interval [L, R], in any drink.

Sadly, you don't know the exact price of anything (you're a programmer, not a housewife!), but you know that water is the cheapest; wine is the most expensive, i.e., $0 \le c_1 \le c_2 \le c_3$. Then the best thing you can do is to assume ***units costs can be any real numbers satisfying this inequality*****.**

Write a program to find the highest possible price of the special drink.

Input

The input contains several test cases. The first line of each test case contains three positive integers n, L, R ($1 \le n \le 100$, $0 \le L \le R \le 100$). The next n lines each contain four non-negative integer a_1, a_2, a_3, p ($0 \le a_1, a_2, a_3 \le 100$, $0 \le p \le 10000$), the amount of water, milk and wine, and the price. The last line of the case contains three integers a_1, a_2, a_3 ($0 \le a_1, a_2, a_3 \le 100$), the drink to be estimated. The last test case is followed by a single zero, which should not be processed.

Output

For each test case, print the case number and the highest possible price to four decimal places. If the input is selfcontradictory, output "Inconsistent data". If the price can be arbitrarily large, output "Too expensive!".

Sample Input

```
1 3 5
1 2 3 10
2 4 6
1 2 4
1 1 1 1
1 1 1
1 3 8
0 1 0 17
0 0 1
3 1 2
2 1 3 14
1 5 1 15
7 3 2 21
4 1 6
2 0 2
45 31 53 4087
30 16 1 1251
11 51 34
```

0
Output for the Sample Input
Case 1: 19.0000
Case 2: Inconsistent data
Case 3: Too expensive!
Case 4: 26.2338
Case 5: 3440.3088

Run Away 题目描述

One of the traps we will encounter in the Pyramid is located in the Large Room. A lot of small holes are drilled into the floor. They look completely harmless at the first sight. But when activated, they start to throw out very hot java, uh ... pardon, lava. Unfortunately, all known paths to the Center Room (where the Sarcophagus is) contain a trigger that activates the trap. The ACM were not able to avoid that. But they have carefully monitored the positions of all the holes. So it is important to find the place in the Large Room that has the maximal distance from all the holes. This place is the safest in the entire room and the archaeologist has to hide there.

Input
The input consists of T test cases. The number of them (T) is given on the first line of the input file. Each test case begins with a line containing three integers X, Y, M separated by space. The numbers satisfy conditions: 1 <= X,Y <=10000, 1 <= M <= 1000. The numbers X and Yindicate the dimensions of the Large Room which has a rectangular shape. The number M stands for the number of holes. Then exactly M lines follow, each containing two integer numbers Ui and Vi (0 <= Ui <= X, 0 <= Vi <= Y) indicating the coordinates of one hole. There may be several holes at the same position.

Output
Print exactly one line for each test case. The line should contain the sentence "The safest point is (P, Q)." where P and Qare the coordinates of the point in the room that has the maximum distance from the nearest hole, rounded to the nearest number with exactly one digit after the decimal point (0.05 rounds up to 0.1).

Sample Input
3
1000 50 1
10 10
100 100 4
10 10
10 90
90 10
90 90
3000 3000 4
1200 85

63 2500
2700 2650
2990 100

Sample Output
The safest point is (1000.0, 50.0).
The safest point is (50.0, 50.0).
The safest point is (1433.0, 1669.8).

Empire Strikes Back 题目描述

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 (Xi, Yi).

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 ($1 \le N \le 300$) and R ($1 \le R \le 1000$). Each of the next N lines contains the integer numbers Xi and Yi ($Xi2 + Yi2 \le R2$) 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

Sample Output
-2 0 2.94725152


激光坦克题目描述 tank.pdf

附录 3　参考程序


昂贵的“饮品” drink.pas


Run Away run_away.cpp

Empire strikes back 与 **Run Away** 相近，有兴趣的同学可自行编写，有问题欢迎联系我。

激光坦克由于是提交答案题目，参数需要自行设置，针对不同的数据程序也需要修改或多次进行迭代，所以就不提供代码了。有什么疑问同样可以联系我。

联系方式

QQ：283330025

E-mail：henryy321@gmail.com