类型1作业。

Sasha Circle (CF549E)

题目大意

平面上有两个点集，点集大小分别为n, m。求一个圆将两个点集隔开。

数据范围

$n, m \leq 10^4$。

所有的点都在整点处，且坐标的绝对值不超过$C(C \leq 10^4)$

引理

- 1.整点集的凸包大小上限为$O(C^{\frac{2}{3}})$。
- 2.抛物面$z = x^2 + y^2$与平面$ax + by + cz = 1$的交为一个椭圆。这个椭圆在平面$z = 0$上的投影是一个圆。(联立两个方程不难发现这个结论。)

记号

- 令$convex(A)$为点集A的凸包。
- 令$|A|$为集合A的大小(所含元素的个数)。

方法一

假定点集A被划分到了圆内，点集B在圆外。一个非常直观的想法是：只保留A的凸包上的点即可。这么做一定是对的，正确性非常显然。利用这一步操作，$|A|$被缩减到了$|convex(A)| = O(C^{\frac{2}{3}})$。

但是对于B就不一样了。B中的每一个点似乎都有可能限制最终圆的大小和位置。所以，我们可能需要保留B中所有的点。

为了让划分圆包含所有A中的点，不包含B中的点，我们可以想到的一个充要条件是：圆心到任意一个A中的点的距离小于到任意一个B中的点的距离。根据这个条件，我们枚举$a \in A, b \in B$，则圆心必然在a, b中垂线划分出的包含a的半平面内。

我们总共会得到$O(|convex(A)||B|)$个半平面，再通过半平面交判定合法性。总时间复杂度为$O(|convex(A)||B| \log(|convex(A)||B|))$。

这个时间复杂度有点大，不能通过本题。

方法二

本题还有一个性质，我们一定可以把最终的划分圆尽量缩小，使得它经过至少两个A中的点。

如果枚举A中的每一个点，以这个点为反演中心做圆的反演，那么问题就转化成了：求平面上一条直线，使直线的两侧各只有一种颜色的点。

这种做法同样只能做到$O(|convex(A)||B| \log(|convex(A)||B|))$的复杂度，不能通过本题。

方法三

我们可以枚举最终的划分圆经过的两个点，并判定是否存在一个合法的圆心，使得这两个点和这个圆心确定的圆满足题意。

实际上，**最终的划分圆可能经过的A中的点对的个数是$O(|convex(A)|)$的**。为什么呢？

做一个二维到三维的变换：$f(x,y) \to (x,y,x^2+y^2)$。这些点全部都会落在一个抛物面$P: z=x^2+y^2$上。而任何一个划分圆都可以被表示为某一个平面$F: ax+by+cz=1$和抛物面P的交在平面$z=0$上的投影，而且圆内的点即为F下方的点，圆外的点即为F上方的点。

一个合法的划分圆必然将两种点完全分开。因而，我们可以求出变换后的A_3, B_3的凸包(用下标表示维度)，并判定是否存在一个平面同时与$convex(A_3), convex(B_3)$相切。如果不存在，即意为A_3, B_3的凸包相交因而不存在合法的划分圆，反之意为存在一个合法的划分圆。

我们注意到，我们只需要求B_3的上凸壳和A_3的下凸壳。而B_3的上凸壳上的点一定是B_2的凸包上的点，而A_3的下凸壳上的点就是整个A_2。

为了找到一个合法的划分平面，强制让划分平面与$convex(A_3)$的上凸壳的棱相切即可。由于$convex(A_3)$的上凸壳的棱数只有$O(|convex(A)|)$，**因此最终的划分圆可能经过的A中的点对的个数是$O(|convex(A)|)$的**。

枚举$convex(A_3)$的每一条棱，以及$convex(B_3)$的每一个点，并判断它们确定的平面是否是合法的划分平面即可。这样做略微有些困难，因而我们可以将这个问题再转化成原模型下的问题去解决：

枚举$convex(A_3)$的每一条棱对应的A_2中的点对，用以确定划分圆圆心所在的直线，再枚举A_2, B_2中的每一个点，用来确定圆心所在直线上的区间。如果最后这些区间有交，即判定为有解。

问题是如何求出$convex(A_3)$呢？我们注意到，A_3的上凸壳的每一个面对应的平面都满足：该平面对应的划分圆包含所有A_2中的点。因此，我们A_3的上凸壳又对应了$convex(A_2)$的一个三角剖分，这个三角剖分满足：任意一个三角形的外接圆都包含A_2的每一个点。

这个三角剖分可以用分治来求解。给定一个凸包，任选一条边(a,b)，在凸包上找到一个c，使得(a,b,c)构成的三角形包含整个凸包，然后递归处理$a \cdots c$和$c \cdots b$中的点即可。该算法的正确性不难证明。

凭借这个算法，我们就可以在$O(|convex(A_2)|^2)$的时间内求出$convex(A_3)$的上凸壳，然后花费$O(|convex(A_2)|(|convex(A_2)|+|B|))$的时间求解原问题。

选择理由和做题心得

这一道计算几何题，从一个非常简单的模型出发，但是为了发觉这个模型的性质，却需要将问题放到更高的维度去分析，实在是巧妙。

并且，我在做题的过程中独立找到了两种方法，并尝试用自己的方法解决这道题。虽然由于复杂度不够优秀而没能通过，但是解决问题的过程让我收益匪浅。这个过程我认识到了精度误差在实数计算中带来的影响，还让我明白了，如果没有发掘出模型本身足够的性质，往往没法写出简洁而高效的算法。

前两种算法并没有发觉到问题全部的性质，因此复杂度不够优秀。

第三种算法通过将问题转化为三维空间中的另一个问题，而发现了新的性质，从而能用更加优秀的复杂度解决。

Intergalaxy Trips

题意

给定一张有向完全图，每一条边有一个出现概率p，意味着对于任何一天，这条边都有p的概率在这一天可以通行。通过这一条边恰好需要花费一天的时间。

你想要从1号节点到n号节点去，每一天你可以选择一条可以通行的边前进，或者停留在当前节点等待一天。

问最优策略下，从1号节点到n号节点的期望时间花费。

数据范围

$n \leq 1000$

分析

这道题的模型一开始让我有点摸不着头脑。它类似于普通的最短路问题，但是多了"概率"这一限制。

按照以往的套路，定义$f[i]$为i号节点到达n号节点的期望时间。但是以什么顺序求f呢？

仿照$Bellman-Ford$算法，我们先给出每一个点的$f_e[x]$(预计f)。一个点的$f_e[x]$是通过所有其他已经确定了的$f_r[x]$(真实f)计算得来的。

每次选择$f_e[x]$最小的点，将$f_r[x]$设为$f_e[x]$。并且用这个点去更新其他点的$f_e[x]$。如此重复，直到1号节点的$f_r[x]$被确定。

这样做的正确性很好理解。$f_e[x]$最小的点一定不可能通过走到其他的未知$f_r[x]$的点，来获得比它的$f_e[x]$更小的$f_r[x]$的，因此它的$f_r[x]$可以被完全确定为$f_e[x]$。

剩下的问题，就是如何通过已知点的f求未知点的$f_e[x]$。

对于任何一种边的存在情况，一个点最好的选择就是，走向可以走向的$f_r[x]$最小的节点，或者停下不动。根据$Bellman-Ford$最短路的性质，每次被确定的$f_r[x]$一定是依次递增的，这给我们接下来的工作带来了便利。

每确定一个点x的$f_e[x]$，我们考虑另一个还没有被确定的点y。如果y连向所有$f_r[v] < f_r[x]$的点v的边全部不存在，y才会有可能选择走向x（如果x实在不够优秀，y会选择停留一天）。因此：

$$f_e[y] = \min\{\frac{\sum_{i=1}^{k}(f_r[x_i]+1)p[y \to x_i]\prod_{j=1}^{i-1}(1-p[y \to x_j])}{\prod_{j=1}^{k}(1-p[y \to x_j])}\}(1 \leq k \leq \text{num of x}, f_r[x_i] \text{is in ascending order})$$

如上的公式似乎有点难以计算，实际上，由于我们会按照递增顺序确定f_r，我们每次新增一个x_i后，只需要判断最优的k是否需要增加即可。实现成代码非常简单。

选择理由和做题心得

这道题本身就十分妙，我在见到这道题的时候感觉眼前一亮，而在解决问题之后，又是眼前一亮。

在解决这道题之前，我对$Bellman-Ford$算法的理解并不是很深刻。而在解决这道题的过程中，我一步步通过对题目性质的分析，找到了一个类似于$Bellman-Ford$的算法，而回头一看，发现这个算法就是基础图论算法$Bellman-Ford$。这是一个奇妙的过程，经过这个过程，我对这个图论算法的理解也更加深刻了。

Longest Increasing Subsequence

题意

让我们考虑一个正整数序列，但其中有一些位置空缺。

我们有一些可以用来填补空缺的位置的数字。每一个给出的数字最多只能使用一次。

你的任务是确定一种填补空缺的方式，使填补完空缺后的序列的最长上升子序列的长度最大。

数据范围

序列长度$n \le 10^5$，空缺位置个数k不超过1000，可以填的数字个数m不超过10^5。

方法一

如果我们任意使用给出的数字，而不考虑重复，我们构造出的最长上升子序列仍然只会使用每一个数字至多一次。因此不能重复使用数字的限制完全可以去掉。

如果只需要求最长上升子序列的长度，我们仿照最常见的Dp做法：我们维护$f[i][j]$表示前i个数字构成的长度为j的最长上升子序列的最小结尾数字。转移分为两种：

- 当第i位是数字x，我们在f中找到最小的满足$f[i-1][j] < x$的j，令$f[i][j+1] = x$，而其余的$f[i][j]$的值继承$f[i-1][j]$。
- 当第i位可以任意填，我们将所有可以用来填补空缺的数字排序，按照单调性去修改$f[i][j]$，修改的规则类似第一种转移。

这样使我们可以在$O(n \log n + (n+m)k)$的时间内求解。

但是需要输出方案。这就有点难办了。

我们可以将这道题转化成这样一个模型：在网格图上找到一条权值最大的折线。其中每一个点的横坐标代表它在序列中的位置，纵坐标代表它的值。

如果我们要在Dp的过程中记录方案，将会消耗大量的空间。但是，实际上有一种时间复杂度基本不变，但是只需要消耗$O(n)$空间的算法。

- $F(x_1, y_1, x_2, y_2)$：求解(x_1, y_1)到(x_2, y_2)的最优折线。
- 按照之前的Dp方式进行求解，但是记录这条折线上$x = (x_1 + x_2)/2$的点的纵坐标y_{mid}。
- 分别求解$F(x_1, y_1, (x_1 + x_2)/2, y_{mid})$和$F((x_1 + x_2)/2, y_{mid}, x_2, y_2)$。

求解一次$F(x_1, y_1, x_2, y_2)$的复杂度为：

$$\begin{aligned} T(x_1, y_1, x_2, y_2) = & T(x_1, y_1, (x_1 + x_2)/2, y_{mid}) \\ & + T((x_1 + x_2)/2, y_{mid}, x_2, y_2) \\ & + O((x_2 - x_1) \log (x_2 - x_1) + sum\_empty(x_1, x_2)(y_2 - y_1 + x_2 - x_1)) \end{aligned}$$

通过计算，$T(1, 0, n, n) = O(n \log^2 n + (n+m)k)$。

而空间复杂度为$O(n+m)$。

方法二

这也是官方题解给出的做法。

假设$len[i]$为到i结尾的最长上升子序列的最长长度(i一定是一个固定的数)。

对于每一个固定的数，以它结尾的最长上升子序列有如下两种情况：

- 它的上一个数字是固定的数。
- 它的上一个数字是一个空缺。

对于第一种情况，我们记录$pre[i]$表示i的上一个数字的位置，而对于第二种情况，则令$pre[i] = -1$。

为了输出答案，我们在序列的最前面添加$-\infty$，末尾添加$+\infty$，在求解完len和pre后，从后往前输出答案。

- 如果当前位置$pre[i] \neq -1$则上一个数就是$pre[i]$。
- 如果当前位置$pre[i] = -1$，则向前枚举一个位置j，使得$len[j]$加上i, j之间所有空位可以构成的最长上升子序列等于$len[i]$，那么i的上一个固定的数就是j。

按照这个方式进行dp，并输出答案，复杂度是$O(n\log n + (n+m)k)$的。

选择理由和做题心得

在解决这道题的过程中，我自行脑补出了一种可以通过这道题的算法(方法一)，而该算法甚至可以推广到其他的问题的计算上，比如背包问题。这让我感觉收获非常大。

虽然说，题解算法的复杂度更加优秀，但是我的这种算法的可推广性更强，也不是没有优势(嘻嘻)。