IOI2020中国国家集训队作业
解题报告

绍兴一中 周雨扬

Contents

1 Sasha Circle

【试题来源】

Codeforces 549 E

【试题大意】

给定n个红点m个蓝点，判断是否存在一个圆将红点和蓝点划分成两部分

数据范围：$n, m \leq 10^4$,坐标绝对值$\leq 10^4$,坐标为整点。

【算法介绍】

若$n = 1$或$m = 1$则一定有解，此处略去。现在我们假定$n, m \geq 2$，同时我们假定A是在圆内的点集，B是在圆外的点集。这里我们讨论时默认A点集在圆周上合法因为我们可以通过给圆的半径加上一个非常小的eps 使得其合法。

性质1:若存在解，总存在一组解满足圆上有不少于2个A中的节点。

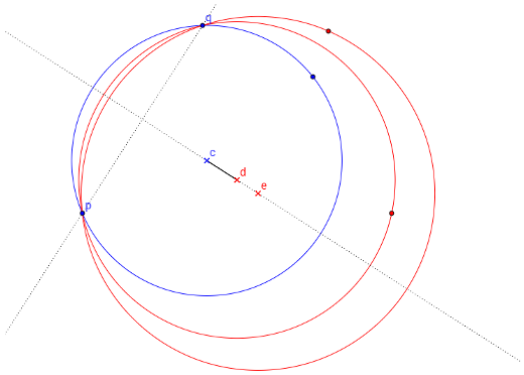
证明1:若圆上没有A中的节点，则可以通过减小圆的半径使得圆上至少有1个节点。

对于圆上恰有1个节点的情况我们可以找到那个节点，设其为P,则我们可以通过在PO上调整圆心O'的坐标使得圆O'上恰有P 和另外一个节点。不难发现圆O'内切于圆O,所以其合法。

根据该性质，我们可以枚举圆上的两个节点QR，显然圆心在QR的中垂线l上。其余点会给圆心坐标带来一个限制，每条限制的合法范围形如$x \geq v$或$x \leq v$。做一次区间并即可判断是否合法。

时间复杂度为$O((n^2 + m^2)(n + m))$。

这里举个例子：



有一个非常显然的优化:我们只枚举点集A的凸包上的点对。由于整点凸包上点个数不超过$O(C^{\frac{2}{3}})$,其中C为整点坐标范围的最大值，因此可以将时间复杂度优化为$O(min(n^2+m^2, C^{\frac{4}{3}})(n+m))$。

但是实际上我们并不需要枚举那么多的节点。

考虑设原平面转化成平面$z=0$，并构造抛物面$\alpha: z=x^2+y^2$。

考虑计算一个不与$z=0$垂直的平面$\beta: ax+by+z=c$截抛物面α得到的截面在$z=0$上的投影。

列方程得$x^2+ax+y^2+by+z=c$，即$x^2+ax+y^2+by=c$。

两边同加$\frac{a^2+b^2}{4}$，得到$(x+\frac{a}{2})^2+(y+\frac{b}{2})^2=c+\frac{a^2+b^2}{4}$，显然为圆方程。

也就是说，平面$\beta: ax+by+z=c$截抛物面α得到的截面在$z=0$上的投影是圆。这里我们不考虑圆所在的坐标以及半径问题，我们只需要知道我们可以根据圆方程构造出满足条件的平面。

那么考虑将$z=0$ 上的点$(x,y,0)$ 映射到(x,y,x^2+y^2) ，这是显然一一对应的

同时，我们发现，投影在圆内的点$P(x,y,x^2+y^2)$在平面β下侧，投影在圆外的点$Q(x,y,x^2+y^2)$在平面β 上侧，投影在圆上的点$R(x,y,x^2+y^2)$在平面β 上。证明略。

于是现在问题转化为求一个平面，将抛物面上两点集分开，使得一点集在平面上方，另一点集在平面下方或平面上，询问是否有解。

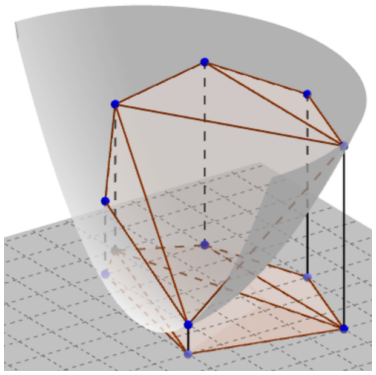
性质2:如果有解，则一定存在一种方案包含下方点集的上凸壳的一条边。这里边的定义为上凸壳中两个面的交集。

证明2:考虑上凸壳一个面所在平面的性质。显然所有点均不在这个面的上方。设确定这个面的不共线三点分别为P,Q,R,则$\triangle PQR$的外接圆包含所有A中的点。反之，若圆O包含所有A 中的点，其一定为一种合法的划分方案。

同理，在上凸壳中的边P,Q必定存在一个包含所有$|A|$中的点的圆满足P,Q在圆上。反之，若一个包含A中所有点的圆满足有不少于两个A中的圆上的点，设为P,Q,则PQ 一定为上凸壳中的一条边。

根据性质1，可得一定存在一条包含上凸壳的边的解。

举个例子：



因此，我们只需要去枚举上凸壳的边的解即可。

如果我们忽略三点共线的情况，则上凸壳在平面$z=0$上的投影可以看成是一个对凸包的三角剖分。这里因为凸包内的点显然不优，因此三角剖分的点只会经过凸包上的点。因此三角剖分的边数就降至凸包点数级别。

所以只需要找到一个这样的剖分，需要枚举的边数就降到凸包的点数级别了

考虑求出一个这样的剖分，考虑分治，在凸包上按逆（顺）时针讨论，$solve(l,r)$ 表示将点l 到点r 进行三角剖分。

将(l,r) 这条边作为底边，找到一个点$k,(l<k<r)$ ，使得这三个点构成三角形的外接圆包含凸包中的所有的点，可以证明在不考虑四点共圆的情况下,恰有一

个三角形的外接圆包含凸包中所有的点。此时直接递归做$solve(l,k), solve(k,r)$ 就好了。

之后可以直接套用之前的单次$O(n+m)$的check做法。可以通过本题。

因为边数是$O(C^{\frac{2}{3}})$级别的，总复杂度为$O(n\log n + m\log m + C^{\frac{2}{3}}(n+m))$。

2 Choosing Ads

【试题来源】

Codeforces 674 G

【试题大意】

给定一个长度为n的数组a，Q次操作，操作有两种:

1 l r id:将$a_l, a_{l+1}, \cdots, a_r$修改为id

2 l r:找到$a_l, a_{l+1}, \cdots, a_r$中出现不少于$p\%$的元素。

数据范围：$n, Q, a_i, id \leq 150000, 20 \leq p \leq 100$.

【算法介绍】

我们首先假设$p = 51$，这是一个非常常见的问题，我们可以采用消去算法来解决。具体的，我们尽量贪心的消去不同的元素，一直消去到仅剩于不超过一种元素，设其为j。显然如果$2cnt_j > r - l + 1$,无论任何情况剩余的元素均为j。这也是唯一一种可能出现超过51%的元素，直接输出即可。

我们尝试扩展消去算法，现在我们需要维护出现至少$p\%$的数字。我们仍然尝试构造一个大小为$\lfloor \frac{100}{p} \rfloor$的候选集合存储有可能出现超过$p\%$的数字。每次插入数字的时候，若该数字已经存在于集合中，给对应数字出现次数+1，否则若集合大小小于$\lfloor \frac{100}{p} \rfloor$，将该数加入集合，否则若存在出现次数为0的数字，用该数覆盖出现次数为0的数字，否则将集合内所有数字的出现次数−1,记为一次消去。

现在我们需要证明所有出现超过$p\%$的数字一定会出现在候选集合中。考虑一次消去的代价。由于一次消去至少需要$\lfloor \frac{100}{p} \rfloor + 1$ 次数字插入操作。由于插入操作仅有$r - l + 1$次，所以消去次数严格小于$\lfloor \frac{p(r-l+1)}{100} \rfloor$，这显然不会超过出现次数，即$\lceil \frac{p(r-l+1)}{100} \rceil$。这说明无论插入顺序，该元素一定会出现在最终的候选集合中。

因为结果和插入顺序无关，所以我们可以使用线段树维护每一段区间的候选集合。候选集合的合并可以直接依次枚举一个候选集合中的元素插入另一个候选集合中，插入方式基本同修改后的消去算法。

设$k = \lfloor \frac{100}{p} \rfloor$，则时间复杂度为$O(n \log nk^2)$。

如果使用Set来维护候选集合可以优化至$O(n \log nk \log k)$。

3 Distance Sums

【试题来源】

Atcoder Regular Contest 103 F

【试题大意】

给定不存在相同元素的长度为n的数组D。

询问是否存在一棵树满足$\Sigma_{j=1}^{j<=n} dis(i,j) = D_i$,其中$dis(i,j)$表示$(i,j)$的简单路径上的边数。

数据范围：$n \leq 10^5, D_i \leq 10^{12}$.

【算法介绍】

设$Sub_{x,y}$表示以x为根的情况下子树y的节点个数。

性质1：若x,y在原树上相邻，则$D_y = D_x + n - 2Sub_{x,y}$。

性质2：对于所有度数不为1的节点x，总存在一颗与其相邻的节点y满足$2Sub_{x,y} < n$。

我们找到节点i满足D_i为所有值中最大的，显然节点i为原树的一个叶子。若不为叶子，则根据性质1，2总能找到一个与其相邻的节点k满足$D_k > D_i$。设与其相邻的节点为j。则根据性质1可以推导出$D_j = D_i - (n-2)$，因此即可推导出唯一的j。此时我们即可将该叶子从树上剥去。

更一般的，我们将节点重标号之后使得其满足$D_1 < D_2 < \cdots < D_n$。我们可以假设节点1为新树的根。设在以1为根的情况下节点i的父节点为fa_i,则有$D_{fa_i} = D_i - n + 2Sub_{fa_i,i}$。显然在确定$Sub_{fa_i,i}$后$fa_i$是唯一的.

性质3:$\forall i \geq 2, 2Sub_{fa_i,i} < n$。

证明:首先考虑和1直接相连的节点，设其为j。根据$D_1 < D_j$，可以得到$2Sub_{fa_j,j} < n$。对于那些不和1直接相连的节点j,设其父节点为k，则有$Sub_{k,j} < Sub_{fa_k,k}$。归纳可得$2Sub_{fa_j,j} < n$

因此我们可以按照$n, n-1, \cdots, 2$的顺序依次确定fa_i。同时我们可以在确定父亲后更新父亲的子树大小。此时我们已经构造出来唯一一颗可能合法的树，直接两次dfs检验答案是否合法即可。

时间复杂度$O(n \log n)$