POJ Challenge Round5 Solutions

绍兴一中

2014 年 1 月 26 日

Contents

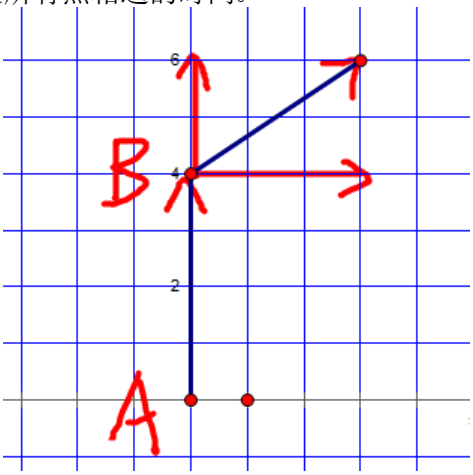
1 magic

1.1 题目大意

正 N 边形所有顶点朝顺时针方向下一个点以相同的固定速度运动（移动方向会随着下一个点的位置变化而变化），求该正 N 边形的所有顶点都达到中心所需要的时间。

1.2 标准算法

点在移动的过程中始终保持着原来的形状，即正 N 边形。因为它们都是以顺时针的点作为基准，而且移动速度完全一样，所以整个图形是个类似辐射对称的形状。这样显然所有点同时到达中心，而且到达中心的时间就是所有点相遇的时间。



如图，我们分析其中相邻的两个点。将 A 与 B 的移动方向关于 AB 线段正交分解， AB 相互靠近的速度可以分为两个部分，一部分为 A 靠近的速度，即为 A 原本的速度，另一部分为 B 远离的速度，即 $B \times \cos(\theta)$ 的速度（ θ 指 A 与 B 速度的夹角）。而 B 分解出来的另一个关于线段 AB 垂直的速度，对于我们要考虑的问题没有影响，仅仅影响了偏转角度，所以没有分析价值，直接忽略。由于题目已给出边数，速度间的夹角可以求出。问题就转化成了一个简单的追及问题。

【时间复杂度】$O(1)$。

2 BG Card

2.1 题目大意

给出一手由 $1 \sim n(n \leq 13)$ 组成的牌，编号为 i 的牌有 $a_i \leq 8$ 张，问最少能几手将牌出完。只能出顺子，并且顺子的长度应该属于给定的集合。

2.2 关键字

动态规划，费用流，ZKW

2.3 算法一

动态规划。

假设我们从左到右按序取顺子，且按照顺子的右端点排序取，设 $f(i, mask)$ 为现在要取以 i 为结尾的顺子，$1 \sim i-1$ 剩余的张数的状态为 j 的方案数。

容易发现，在取以 i 为结尾的顺子前，$1 \sim i-1$ 剩余的张数必然为单调不降。

由于一副扑克牌的每种牌最多只出现了 8 次，所以有用的状态数为不超过 $n\binom{n+7}{8} \leq 1637610$。

时间复杂度 $O\left(n\binom{n+7}{8}\right)$。

2.4 算法二

费用流。

预处理 $cost(l)$ 表示长度为 l 的顺子最少能够用几副顺子拼合而成，就可以直接转成类似 [NOI2008]志愿者招募 的问题。建图如下：

如果 $a_i - a_{i-1} > 0$，则连源到 i 流量为 $a_i - a_{i-1}$ 费用为 0 的边，否则连 i 到汇的流量为 $a_{i-1} - a_i$ 费用为 0 的边；i 到 j 连流量为 ∞ 费用为 $cost(j-i)$ 的边。

如果使用 ZKW 费用流，能在 2 秒内跑出 $n = 1000, a_i \leq 1000$ 的数据。

3 Billiards

3.1 题目大意

给出 $n(\leq 22)$ 个台球，其中一个有初速度，问发生第 $m(\leq 50)$ 次碰撞时，所有球的坐标。

3.2 关键字

事件点，模拟，物理，动量

3.3 物理基础

这种程度的物理应该轻松虐吧。

首先，两个球碰撞的时候，弹力方向是两球心的连线，设其 x 轴，将速度分解为水平方向分速度和垂直方向分速度。

设两球初始速度分别为 $\boldsymbol{v_1}, \boldsymbol{v_2}$，碰撞后速度为 $\boldsymbol{v'_1}, \boldsymbol{v'_2}$，那么因为垂直方向上没有受力，所以有：

$$\boldsymbol{v_{1y}} = \boldsymbol{v'_{1y}} \tag{1}$$

$$\boldsymbol{v_{2y}} = \boldsymbol{v'_{2y}} \tag{2}$$

再根据动量守恒和能量守恒得到：

$$m\boldsymbol{v_1} + m\boldsymbol{v_2} = m\boldsymbol{v'_1} + m\boldsymbol{v'_2} \tag{3}$$

$$\frac{1}{2}mv_1^2 + \frac{1}{2}mv_2^2 = \frac{1}{2}mv_1'^2 + \frac{1}{2}mv_2'^2 \tag{4}$$

又因为:

$$\boldsymbol{v} = \boldsymbol{v_x} + \boldsymbol{v_y} \tag{5}$$

$$v^2 = v_x^2 + v_y^2 \tag{6}$$

联立可得

$$\boldsymbol{v_{1x}} + \boldsymbol{v_{2x}} = \boldsymbol{v'_{1x}} + \boldsymbol{v'_{2x}} \tag{7}$$

$$v_{1x}^2 + v_{2x}^2 = v_{1x}'^2 + v_{2x}'^2 \tag{8}$$

解得 $\boldsymbol{v'_{1x}} = \boldsymbol{v_{1x}}, \boldsymbol{v'_{2x}} = \boldsymbol{v_{2x}}$ 或 $\boldsymbol{v'_{1x}} = \boldsymbol{v_{2x}}, \boldsymbol{v'_{2x}} = \boldsymbol{v_{1x}}$。取后者即可。

其实就一句话，沿力的方向分解，转成一维上的碰撞问题，由于质量相等，所以直接交换力方向上的分速度即可。

3.4 标准算法

枚举两个球，忽视另外球，用二次方程求出他们的碰撞时间。这个应该不成问题，就不赘述了。同理找出每个球撞到边界上的时间。

在这些时间里求一个最早的，然后推到该时间点，改变相关的球的速度，具体的计算方法在上面已经讲了。

此题对精度要求比较高，如果使用 `double` 可能会出错。

复杂度：$O(n^2m)$。

4 history

4.1 题目大意

支持两种操作，在无向图中加边和询问某一过去时刻两点是否连通，强制在线。

4.2 关键字

动态树，[函数式]并查集，按秩合并，启发式合并

4.3 伪标准算法

由于只有加边和询问连通性，所以只需维护森林，可以开边表（或用只记父亲的方法，两树连接时其中一棵树换根，翻转一段）。在询问时求两点之间时间最迟的边即可。

用数据结构优化刚才的暴力，需要支持维护森林和求两点间的最大值，直接套用动态树 $(Link-Cut-Tree)$ 即可。数据范围较大，可能需要优化常数。

【时间复杂度】$O(n\log n)$。

4.4 标准算法

如果只要求询问当时的情况，那么并查集足矣，但如果要查询之前的情况呢？一种暴力的思想是全部存下来，但空间就是 $O(mn)$ 的了。

但是，把每次修改前的结果存下来是不是太浪费了？很多信息是可以共用的。

由于路径压缩修改的信息较多，我们采用另一种并查集的优化：按秩合并。

这种优化即在并查集的根节点上再多记录一个值表示这个森林中的最大深度，由于并查集询问和深度有关，所以在合并时要尽可能减小森林的深度，把深度小的合并到深度大的即可。

那么这样并查集的合并只修改了一个节点的父亲，而且这个节点的父亲一旦确定就不会再修改，所以可以直接记录下这个节点父亲确定的时间。

而在询问时，若询问时间大于该父亲确定的时间，则说明此时这条边还没有连上，当前节点就是当时的根。

这样就得到了一个可以处理历史询问的并查集了。

但是再仔细想想这个并查集，觉得似乎这已经不像并查集了，它维护的就是森林，但是这里的树的形态却和原来的树不同，但这没有关系。所以，忽略树的形态后，我们可以按我们想要的方式合并两棵树，而不一定要按给出的边合并。同时，按秩合并就像启发式合并，有效地使树高降低到了 $\log n$ 的级别。也就是说，从最开始的暴力算法出发，用启发式合并来维护森林，也可以到达这里。

【时间复杂度】$O(n \log n)$。

4.5 比赛中出现的其他做法

启发式合并倍增数组，查询路径上最大值。

【时间复杂度】$O(n \log^2 n)$。数据造得不够强，没把$O(n \log^2 n)$卡掉。

启发式合并，记录每个节点不同时间的根。

【时间复杂度】$O(n \log n)$。

5 the nth Power of P

5.1 题目大意

给出一个置换 Q 和一个整数 $n(\leq 1000)$，问有多少置换 P 满足 $P^n = Q$。用 Q 中长度为 $l(\leq 10000)$ 的环有 $a_l(\leq 100)$ 个来描述 Q。

5.2 关键字

置换群，动态规划

5.3 一些思路

可以先观察 n 较小的情况：

如果 $n = 2$，那么一个长度为 L 的环的 n 次方只有两种情况：L 是奇数，那么它的 n 次方仍然为一个长度为 L 的环；否则将是两个长度为 $\frac{L}{2}$ 的环。

如果 $n = 4$，那么有三种情况：如果 $L \equiv 0 \pmod 4$，则变为 4 个长度为 $\frac{L}{4}$ 的环；如果 $L \equiv 2 \pmod 4$，则变为 2 个长度为 $\frac{L}{2}$ 的环；否则仍然为一个长度为 L 的环。

也就是说，一个长度为 L 的环，它的 n 次方为 (L, n)[1] 个长度为 $\frac{L}{(L,n)}$。这个证明非常容易，读者可以尝试一下。

上面我们研究了 $P \to Q$，我们反过来思考就能得到 $Q \to P$：对于 Q 中的一个长度为 l 的环，枚举 $d|n$，如果 $(l \cdot d, n) = d$，亦即 $\left(l, \frac{n}{d}\right) = 1$，那么 d 个 l 的环能由 P 中长度为 $l \cdot d$ 的环的 n 次方变来。

5.4 算法一

至此，我们可以得出一个算法：

首先枚举 l，并且得到所有合法的 d，令 $m = a_l$，现在只需要求将 m 个长为 l 的环并起来的方案数即可。

[1] (a, b) 表示 a 和 b 的最大公约数

接着设计动态规划方程，设 $f(m)$ 表示 m 个长为 l 的环并起来的方案数，那么:

$$f(m) = \sum_{d|n,\left(l,\frac{n}{d}\right)=1} f(m-d) \cdot \frac{(m-1)!}{(m-d)!} \cdot l^{d-1}$$

这个方程的意义是：首先枚举第 m 个环是由 d 个环并在一起的，然后在前 $m-1$ 个环中取出 $d-1$ 个来依次插进第 m 个环中，每次插入有 l 种方案。

对于这个动态规划的转移次数，我们可以这样计算：

$$\begin{aligned}
& m \sum_{i=1}^{l} \sum_{d|n} e\left(\left(i, \frac{n}{d}\right)\right) \\
= & m \sum_{i=1}^{l} \sum_{d|n} e\left((i,d)\right) \\
= & m \sum_{i=1}^{l} \sum_{d|n} \sum_{d'|(i,d)} \mu(d') \\
= & m \sum_{d|n} \sum_{d'|d} \left\lfloor \frac{l}{d'} \right\rfloor \mu(d') \\
\leq & m \sum_{d|n} \left\lceil \frac{l}{d} \right\rceil \sum_{d'|d} \frac{d}{d'} \mu(d') \\
= & m \sum_{d|n} \left\lceil \frac{l}{d} \right\rceil \phi(d)
\end{aligned}$$

所以时间复杂度为 $O(m \cdot l\sqrt{n})$[2]。

5.5 算法二

仔细观察算法一之后发现，我们枚举 l 那一步似乎太过于浪费了。因为如果 $(l,n) = (l',n)$ 相同，那么能转移的 d 的集合也相同。这一点非常好证明。

[2]这里将 n 的约数个数 $d(n)$ 视为 $O(\sqrt{n})$

所以 l 与 l' 的转移方程中唯一不同的就是 l^{d-1} 这部分。于是，我们可以将转移中的 l 看成一个未知数，$f(m)$ 就变成一个关于 l 的一元 m 次多项式。

于是，我们对于每个 $d|n$，做一遍 $l=d$ 的动态规划，得到一系列关于 l 的多项式。对于一个 $(l,n)=d$ 的 l，我们直接将 l 代入 $f(a_l)$ 就能得到答案了。

这样复杂度降为 $O(n\cdot m^2+l\cdot m)$。

6 architect

6.1 题目大意

支持动态加点、询问一个内角均为 45° 的倍数的多边形内的点数。其中点在网格中心，多边形在网格边界上。

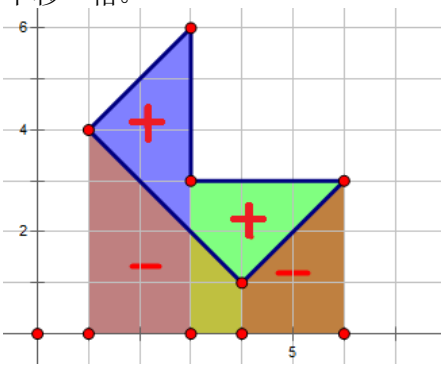
6.2 关键字

多边形分解，扫描线，按时间分治

6.3 标准算法

由于多边形较复杂，考虑拆分。把计算多边形面积时每条边变为原点与该边构成的三角形的方法运用到此题上，发现该方法多出了一些与坐标轴既不平行，也不呈 45° 的边，破坏了题目的条件。

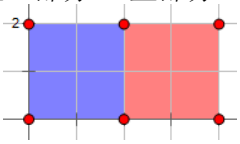
则考虑另一种拆分，除竖直边外每条边都向 x 轴投影，该边与投影构成的 45° 直角梯形和矩形。所有 45° 直角梯形和矩形内点数通过加减（沿顺时针遍历则向右的边 $+$，向左的边 $-$）即可得到原多边形内的点数，这样仍保留了题目的条件（由于题目中的点都在网格中心，所以不用考虑多边形顶点在边上的划分，若点在网格边界上，则可以把多边形的边界扩大 0.5 后再计算）。要注意由于点在多边形边界上也算，所以 $-$ 的 45° 直角梯形需要向下移一格。



多边形拆分后题目变为求 45° 直角梯形和矩形内的点数。

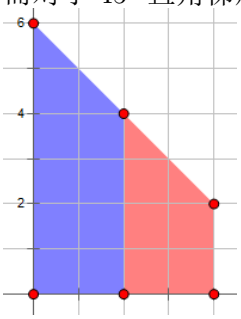
先不考虑动态加点询问，假设所有点已经加完，然后再进行询问。

对于询问某个一边在 x 轴上的矩形内的点数，可以进行转换，即用（红 + 蓝）部分 − 蓝部分。两部分都是求询问点的左下方的点数。



然后使用扫描线算法，即把加点和询问分别排序按 x 坐标(y 坐标为第二关键字)排序，扫描询问，把在询问点左侧的点都加入数据结构中（比如树状数组），然后根据 y 坐标在数据结构中询问区间和（前缀和）。

而对于 45° 直角梯形也可以用类似方法：



此时 y 关键字变为了 $x + y$ 。

最后由于题目给出的是一边加点一边询问，而点对询问的贡献又是独立的，所以可以使用按时间分治（cdq分治），即所有事件开始按时间排序，然后找到中点，将序列划分为两部分，分别递归左右部分，然后计算左侧点对右侧询问的贡献，此时由于左侧的时间都小于右侧，就变成了刚才先加点再询问的情况，使用刚才所述的方法即可。刚才点和询问的排序可以使用归并来完成，以减小常数。

设 n 为总点数，则复杂度 $T(n) = 2T\left(\frac{n}{2}\right) + O(n \log n)$ ，由主定理的 Case 2 可得 $T(n) = O(n \log^2 n)$ 。

此外也可以用树套树来支持在线支持二维平面内的加点和查询，复杂度也为 $O(n \log^2 n)$ ，但常数较大。

【时间复杂度】$O(n \log^2 n)$。