世界线 解题报告

安徽师范大学附属中学 罗哲正

1 试题来源

UOJ Round 10 B.世界线

链接：http://uoj.ac/contest/22/problem/153。

2 试题大意

为了回到正确的过去，picks 博士观测了 n 个过去和 n个未来。经过 picks 博士的研究，他发现存在一个 1 到 n 的排列 A，使得对于每一个 i 都存在一条世界线连接着第 i 个过去和第 A_i 个未来。现在 picks 博士想要通过实验得到排列 A。

2.1 任务描述

因为有关世界线的理论非常复杂，所以 picks 博士的实验同样非常繁琐——他甚至用了不止一轮实验才得到了答案。但是我们可以把实验过程简化得到一道这样的题目：

这是一道交互题，在交互库中生成了一个长度为 n 的置换 A，你需要编写一个函数 `query_permutation` 来得到这个置换。

`query_permutation(n,ans)`

- n：置换 A 的长度，保证 $n \geq 1$。
- ans：一个 int 数组，你需要把你得到的排列 A 的第 i 项存到 $ans[i]$ 中作为结果，其中 $1 \leq i \leq n$，并返回 1。如果你发现无论如何都无法唯一确定排列 A，那么就返回 0。

你可以四个函数 `new_round`、`next_step`、`addedge`、`query` 来帮助你确定这个排列。

new_round() 调用这个函数后，将开始新的一轮实验，新的实验默认阶段为 1。

next_step() 调用这个函数后，实验将进入下一个阶段。

addedge(u,v) 这个函数只能在每一轮实验的第一个阶段使用，表示在第 u 个点和第 v 个点之间连一条边。如果 u 或者 v 不在范围 $[1,n]$ 之内，这次操作将会被忽略。

query(u,v) 将返回 $u+n$ 和 $v+n$ 的连通性，如果联通则返回 1，否则返回 0。如果 u 或者 v 不在范围 $[1,n]$ 之内，将会返回 0。

当你调用函数new_round的时候，将开始一轮新的实验。这时交互库中会生成一个 $2n$ 个点的无向图，初始状态下有 n 条边，第 i 条边连接了点 i 和点 A_i+n。每一轮实验可以分成两个阶段：

1. 这个阶段在调用new_round后自动进入，**你只能在每一轮实验的这一个阶段内调用函数 addedge**。每当你调用一次函数 addedge(u,v)，交互库将会在图中的第 u 个点和第 v 个点之间连上一条无向边。如果在这个阶段内调用了函数next_step，那么将会进入第二个阶段。**这个阶段内不允许调用函数 query 和 new_round**。

2. **你只能在每一轮实验的这一个阶段内调用函数 query**。每当你调用一次函数 query(u,v)，交互库将会返回图中第 $u+n$ 个点和第 $v+n$ 个点之间的连通性。如果在这个阶段内调用了函数 new_round，将会重新开始一轮新的实验。**这个阶段不允许调用函数 addedge 和 next_step**。

如果你已经得到了答案，那么你可以在任意一轮实验的任意一个阶段返回答案。

picks 博士的巴拉拉能量是有限的，因此为了节约能源，他规定实验最多进行两轮，即你最多只能调用两次函数 new_round（**注意：程序开始必须调用一次new_round**）。

同时，picks博士发现调用函数 query 也是会消耗巴拉拉能量的，因此他想让你尽可能地减少函数 query 的调用次数。

2.2 限制与约定

共 10 个测试点，每个测试点 10 分。

对于每一个测试点，你得程序必须满足如下条件，才能获得 10 分，否则得

0 分：

1. 对于每一组数据，函数调用都是合法的。

2. 对于每一组数据，`query` 函数和 `addedge` 函数的调用次数都不会超过 2×10^6 次。

测试点编号	n的规模
1	$n = 4$
2	$n = 5$
3	$n = 990$
4	$n = 1000$
5	$n = 4950$
6	$n = 5000$
7	$n = 5050$
8	$n \leq 10000$
9	
10	

对于所有数据，保证有 $n \geq 1, T \leq 10$，以上的 10 个测试点都满足 $T = 10$。

时间限制：1s

空间限制：256MB

2.3 额外的抱枕

如果这题场上有人 AC，那么我们会给所有在比赛时通过了这题的选手中，后三个点调用 `query` 函数总数最少的人赠送萌萌哒 UOJ 抱枕一个!

3 算法介绍

3.1 算法1

前两档数据$n = 4$和$n = 5$是可以手玩的。

经过人类智慧，可以发现$n = 4$时只需要第一次$< 1, 2 >$第二次$< 2, 3 >$，那么四个点就都区别开了。

$n = 5$ 时可以连边$< 1, 2 >, < 3, 4 >$和$< 2, 3 >, < 4, 5 >$。首先判断出1和5，然后就可以顺着推出$2, 3, 4$了。

期望得分20分。

3.2 算法2

我们考虑推广$n = 4$和$n = 5$的情况，我们第一次连边$< 2k - 1, 2k >$，第二次连边$< 2k, 2k + 1 >$。对于奇数，首先能区分出1和n。然后从A_1开始。对于奇数i，在第一次建的图里找到与A_i相连的点，即为A_{i+1}，对于偶数，在第二次建的图里找到与之相连的点即可。

于是问题就变成了求匹配，由于只能询问连通性，所以我只有枚举每条边求匹配的办法，询问复杂度是$O\left(n^2\right)$的。

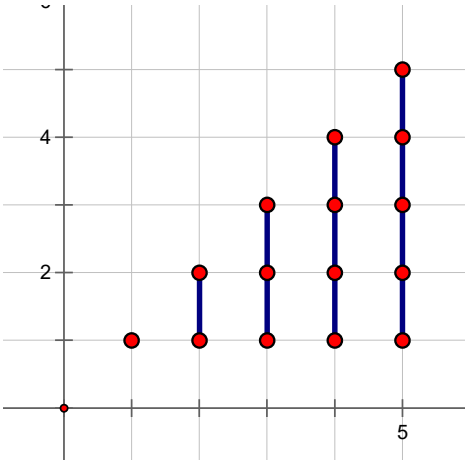
期望得分40分。

3.3 算法3

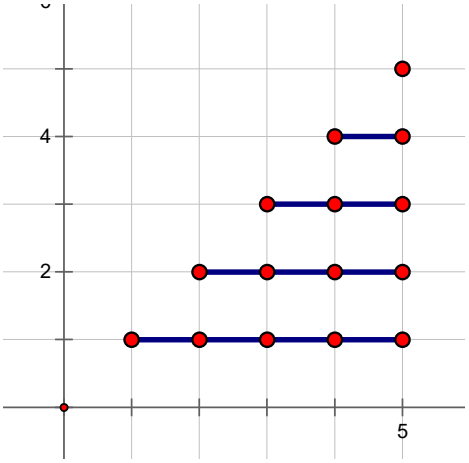
可以发现，一个连通块内的点是不可区分的，那么当连通块大小超过2时，有效的信息只有块的大小，我们不妨考虑使用块的大小来区分点。

我们把点分组，第i组有i个点，一个点分配一个坐标(i, j)表示它是第i组第j个点（$j \le i$）。

对于第一轮，我们连纵向边：



对于第二轮，我们连横向边。

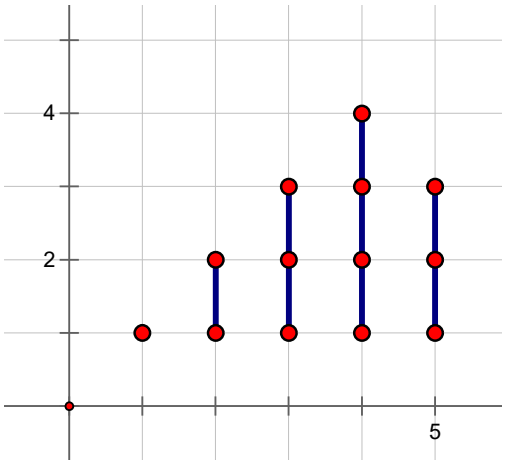


统计第一轮中所在块的大小可以确定i，统计第二轮则可以确定j，于是一个点的坐标就被确定了。

那么该如何统计一轮中每个点所在块大小呢，换句话说就是要把连通块给分出来。

考虑按某种顺序添加点，对于每个点，枚举之前的所有连通块，如果当前点与某个连通块连通就把当前点加入那个连通块，这样的复杂度是$O\left(n\sqrt{n}\right)$的。如果采用随机的顺序加入点，则可以获得约0.4的常数，这是由于比较大的块容易较早的被找到。

但是这个算法只能处理$n=\frac{k(k+1)}{2}$的情况，如果不满足这个条件例如$n=13$：



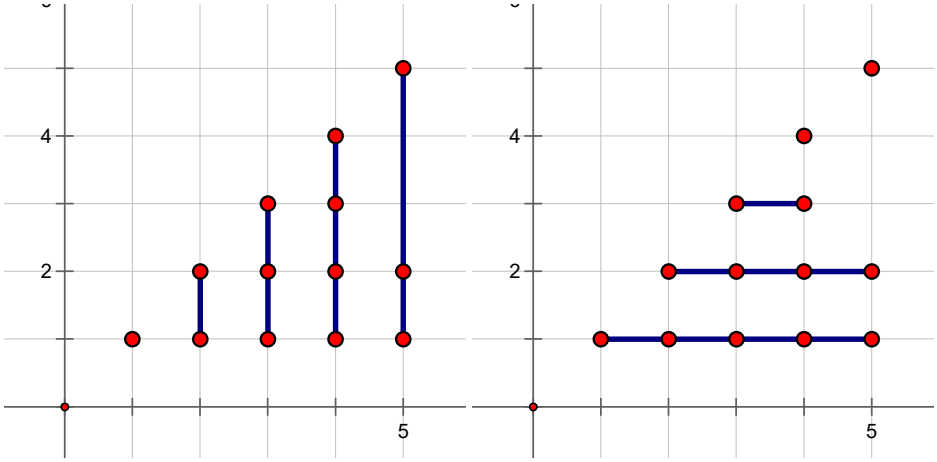
那么在第一轮连边的时候，就会出现两个大小为3的块，它们是无法区分的。

观察数据，哇！我们可以得到30分（其实那些鬼畜的n是我建议放的）。

3.4 算法4

我们来考虑更通用的办法，假设$n=\frac{k(k+1)}{2}+p$，其中$p<k$。

我们考虑移动最后一个点，将第n个点$(k+1,p)$移动至$(k+1,k+1)$，然后还是按照通常的方法构图：

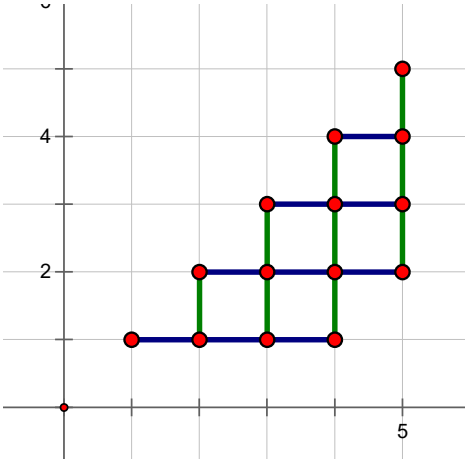


我们会发现n号点在第一轮中所在块大小为p，第二轮中为1，第二轮中所在块大小为1的只有两个点，另一个是(k,k)，而由于$p<k$所以这是可以区分出来的。那么最后一列就可以通过在第一轮中与n号点连通而变得容易区分，其余的部分直接使用算法3即可。

3.5 算法5

算法4已经接近完美了，但是还遗留了一个小问题——$p=k$的情况，这种情况下如果直接套用算法4，点$(k+1,k+1)$和点(k,k)将会变得不可区分，于是需要特判。

这种情况的特判方法很多，我来讲讲我的做法，设$n=\frac{k(k+1)}{2}-1$，那么我们去掉$(k,1)$，例如$n=14$时如图所示：



这种情况下，我们通过查找在第1轮和第2轮中作为单独的块出现的点就可以方便的区分出$(1,1)$和(k,k)，于是第1行和第k列都被区分出来了，剩下的就跟算法3一模一样了。

可以发现这个方法的询问数是很小的，写的好的话似乎可以拿抱枕的。

4 总结

这是一道真正意义上的交互题（我认为仅仅是使用交互库作为输入输出工具并不算是真正意义上交互题，交互题应该允许程序向交互库传递信息）。其思路比较巧妙，主要考察了分块的思想与构造思维，并且要求选手对细节和特殊情况有敏锐的观察和处理能力，是一道有区分度的思维题。