crisis解题报告

福州一中　卓亮

April 30, 2012

Contents

1 题意简述

有n个士兵，第i个士兵的初始位置是(x_i, y_i)。有q个操作，操作分为3类：命令，撤销和重做，询问。

命令有三种:

- Move i j a b （第i个士兵到第j个士兵向东移动a米，向北移动b米）；
- Patrol i j a （第i个士兵到第j个士兵绕着敌军大本营逆时针巡逻a弧度）；
- Lurk i j （第i个士兵到第j个士兵潜伏到敌军大本营）。

撤销和重做:

- Cancel a（撤销最后a次要被执行的（移动，巡逻或潜伏）命令）；
- Redo a（重做最后a个被撤销的命令）。

询问:

- Ask i（询问第i个士兵应处的位置）。

2 命题思路

范浩强介绍了新的数据结构，本人尝试就此出一道题。

3 考查点

本题是一个数据结构题。需要选手掌握矩阵的相关知识，基本的线段树。完整的解决还需要选手懂得可持久化数据结构的思想。

4 问题分析

我们可以看到，如果我们对位于(x, y)的士兵进行了移动操作，那么它新的位置(x', y')就满足$x' = x + a, y' = y + b$。如果我们对它进行了旋转操作，那么$x' = x\cos a - y\sin a, y' = x\sin a + y\cos a$。如果进行了潜伏，那么$x' = 0, y' = 0$。

考虑到这3个命令都是变换，我们可以尝试用矩阵写出。即

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$$

由此，我们看出，一开始给的位置并没有在变换中起作用。由于矩阵乘法具有结合律，我们可以用3×3的矩阵维护信息。

我们来看如果没有撤销和重做，应当如何解决。首先，我们的命令，相当于把一个区间全部乘一个矩阵。然而，我们的询问只问一个点的状态。因而，我们可以考虑采用线段树解决。我们对每个线段维护，这个线段全部乘了什么矩阵。一开始全部设为单位矩阵。在插入时，如果完整覆盖，那么，要插入的矩阵乘到线段上。否则，我们就把首先要把这个线段的矩阵下传到两个儿子上。然后把这个线段的矩阵设为单位矩阵，接着分别在至多两个儿子上处理插入操作。询问时，可以自底向上，依次乘经过的线段上的矩阵。

如果没有潜伏，也就是说所有矩阵是可逆的。我们可以构建一棵“版本”树。如果某一时刻a，每个点的状态和另一个时刻b 完全相同，那么它们同属一个点。如果某一个时刻a的状态，经过一次命令，得到另一个时刻b的状态，那么a就向b连一条边。

容易看出，一次命令相当于走一个儿子，撤销相当于找若干次父亲，而重做相当于沿着找父亲的路径退回去。而询问就可以挂在当前停留位置上。这样，我们就能够建立起这一棵“版本”树了。

接着我们对这棵版本树进行深度优先搜索。只有两种可能的情况。入栈，即乘对应的矩阵。出栈，即乘对应矩阵的逆。如果碰到挂有询问的节点时，我们就处理对应的询问。

这样我们就解决了没有潜伏的情况了，期望得分60分。

有潜伏的情况，由于不存在逆，我们势必要记录下每一个历史版本，也就是“版本”树的每个节点的状态。这就是本题的考点，可持久化。

可持久化数据结构(Persistent data structure)，可参考[1]，总是记录下它之前的版本，当它被修改的时候。本题需要对线段树进行可持久化。最朴素的方法是，每一次复制一下整棵线段树。这个方法不太优秀。

我们注意到，修改操作最多只会涉及$O(\log n)$个线段树节点。这就意味着，大部分

的节点，是和之前一模一样的。因而，如果一个节点的左子树和之前一模一样，我们不是把整个左子树复制一遍，而是用一个指针，指向过去的左子树，代表这个节点的左子树与被指的部分完全一样。如果右子树一样，也同理操作。这样，虽然一个线段树中的节点，可能被很多点指向，也就是说有很多父亲，但是一个节点仍然只有两个儿子。而我们的线段树操作可以自顶向下实现，亦即，不需要找父亲操作。因而这个方法是完全可行的。注意到每次操作最多新建$O(\log n)$的节点，因而这个算法的空间复杂度是$O(n + q \log n)$的。这个算法的期望得分是100分。

5 数据生成方法

本题的数据是随机生成。在前期，基本只出命令操作。在后期，出撤销和重做的机会大大增加。

6 选手得分统计

本题对国家队候选队员进行了测试，得分统计如表(1)所示。

分数	人数
60	2
100	3

Table 1: 得分统计

本题的平均分大约为38.2分。

7 后记

很不好意思，本人在命制此题时，没有仔细地计算至多会用到多少内存，只是随便估了一下。测试后，少数同学抱怨此题的内存限制过紧。经过认真计算，使用本算法是不会超过内存限制的。不过内存的确较为紧张。有同学因对此处理不慎，丢掉了满分，本人在此表示歉意。

来自哈尔滨第三中学的王钦石同学，对本算法进行了改进。并非采用矩阵来描述变换，而是用复数来描述变换。我们把平面中的点(x, y)，对应于复数$x + \mathrm{i}y$。可以对每个线段树节点记两个复数a, b，表示这个区间乘a加b。即一个复数z变换为$z' = az + b$。由于复数的运算有结合律和分配率，因而是可维护的。本题中，三种命令都可以对应一个复数运算：移动对应于加一个复数；巡逻对应于乘一个复数；而潜伏对应于乘0。上述改进基于了本题中特定的变换。虽然时间复杂度和空间复杂度不变，但是隐藏在大O符号内的系数变小了很多。

References

[1] *Persistent data structure*[EB/OL]. [2012-3-29]. http://en.wikipedia.org/wiki/Persistent_data_structure.

Appendices

A crisis完整题面

Time Limit: 2s, Memory Limit: 256MB

某地爆发了经济危机，反对派武装发动造反。你任某军军长，负责消灭该地的反对派武装力量。由于你英勇神武，战无不克，很快，反对派武装就被基本消灭。残部退守到位于$(0,0)$附近的大本营。你决定包围它，然后择日发起总攻。你将对士兵下达命令。例如第1旅和第2旅向西行军500公里，第2旅和第3旅绕着大本营巡逻，吸引敌军注意。当然，有时候你发现你的一些部署不大妥当，连忙撤销之前下达的一些命令。部署后，作为一名一丝不苟的少将，你一定要抽查一些士兵的位置是否恰好达到了预定的位置。当然，你首先要算出被抽查士兵应该所处的位置才行。

Task: crisis

Input: stdin

第一行是一个数n，表示你的麾下士兵的总数。接下来n行，每行两个数，表示每个士兵在开始行动前所处的位置。接下来一个数q，表示操作个数。再q行，每行是一个操作。一共有6种可能的操作。操作分为3类：命令，撤销和重做，询问。

命令有三种：

- Move i j a b （第i个士兵到第j个士兵向东移动a米，向北移动b米）；
- Patrol i j a （第i个士兵到第j个士兵绕着敌军大本营逆时针巡逻a弧度）；
- Lurk i j （第i个士兵到第j个士兵潜伏到敌军大本营）。

撤销和重做：

- Cancel a（撤销最后a次要被执行的（移动，巡逻或潜伏）命令）；
- Redo a（重做最后a个被撤销的命令）。

询问：

- Ask i（询问第i个士兵应处的位置）。

Output: stdout

对每个询问输出一行，每行两个数，代表被询问的士兵应处的位置。你的输出与参考答案的误差少于0.0001即视为正确。

Sample Input	Sample Output
10 1 1 1 2 1 3 1 4 1 5 1 6 1 7 1 8 1 9 1 10 15 Lurk 1 5 Move 1 5 100 100 Move 1 1 100 100 Move 1 1 10 0 Move 1 1 0 10 Cancel 1 Cancel 1 Patrol 6 10 3.14159265358979323 Patrol 6 8 1.57079632679489661 Move 2 7 -100 0 Cancel 2 Ask 1 Ask 10 Redo 1 Ask 5	200 200 -1 -10 100 100

Hints

你早已摸清了反对派武装的脾气，知道他们不敢贸然行动，因此即使你的士兵很靠近敌军大本营，在你未主动攻击之前，敌人也不会攻击你的士兵。

另外一个军的人数一般为15000-20000人。

不会有超过50000次操作。

另外，考虑到潜伏是十分危险的行动，大概有60%的数据不含潜伏。