# NOI2016 模拟题 Solution

Chloe_fan 613

## Zbox loves ants

Made by Chloe_fan

首先所谓“当有两只蚂蚁相遇时，它们会各自调转方向，以原有的速度继续爬行”可以看做两只蚂蚁“交换了身躯”按照原有的方向爬行。那么现在等于说每只蚂蚁都有两个离开绳子的时间可以选择，而答案只可能会是每只蚂蚁离开时间其中的一个，即最多 2n 种。我们可以从小到大枚举答案，在当前答案下，有部分蚂蚁只能选择其中一个方向，有部分蚂蚁可以选择两个方向，有部分蚂蚁一个方向都不能选择。存在方案的前提是不存在蚂蚁两个方向都不能选择，而方案数就是 2^{能够选择两个方向的蚂蚁的数量}。只需要动态维护 3 种部分蚂蚁的数量即可。

# Zbox loves stack

Made by 613

## 算法 1

按照题意模拟

复杂度 O（nq）

可以通过测试点 1

## 算法 2

当没有弹栈操作的时候很明显可以用线段树算出每个时刻每个栈的元素个数 ,我们将这个线段树可持久化。对于每个询问二分之前判断是否存在这么多个元素

复杂度 O（qlog^2n+nlogn）

可以通过测试点 4

## 算法 3

按照题意模拟 ,并用一个线段树维护区间是否栈空保证在弹栈过程中只访问有元素的栈

复杂度 O（qlogn+n）

可以通过测试点 1,5

# 算法 4

只有单点弹栈的时候可以找到之前最后一个对这个点压栈的操作然后把这个操作拆成 2 个，然后做算法 2

复杂度 O（qlog^2n+nlogn）

可以通过测试点 4,6

# 算法 5

如果你写了能 A 的做法并且幸运地把删除标记写炸的话会得到一些奇怪的分数

不需要删除标记的测试点 4,6 以及不会在栈空的时候弹栈的测试点 8

# 奇怪的做法

可能有奇怪的块套树做法但是貌似出题人比较思博没有仔细想

复杂度大概是 O（qn^0.5logn）

如果有的话可以通过测试点 1,2,3

# 不靠谱的做法

对区间建线段树然后拿 rope 打标记，大家都知道 rope 是可持久化的平衡树维护字符串所以两个压栈标记可以直接拼起来弹栈只要 pop_back 一下就好了

复杂度 O（qlog^2n）常数大飞起来

可以通过测试点 1,2,3,5

# 靠谱的做法

rope 常数大就自己手写可持久化平衡树呀 QAQ

ps：rope 之所以常数飞起来主要是支持一系列灭绝人性的操作都是一个 log

这个题只需要 merge 和 pop_back 和 push_back 这 3 个操作，简单的可持久化 treap 就可以实现，代码不长

复杂度 O（qlog^2n）

分数取决于常数，std 采用了这个方法

## 我不想写那么恶心的东西怎么办

你发现出题人并没有强制在线这是为什么呢，当然是因为出题人思博（其实是想一个第二题还是不要出得太 jb）,既然大家都做过 ZJOI2016day2 的第一题你发现能用同样的做法搞，就是离线扫描线扫完用一个 splay 就可以直接维护辣

复杂度 O（nlogq）或 O（qlogq）取决于实现的方式

如果没有对区间离散化的话比较卡常数，否则基本 1s 之内出解

## 我要是没做过 ZJOI 而且不想写恶心的东西怎么办

这个。。。~\(≧▽≦)/~


你可以来裱出题人

# Zbox loves meizi

Made by 613

这个题目考察了选手的乱搞能力。。。。。

## 算法 1

交样例程序

你可以获得 0 分但不是这个题的最低分（雾

Q：为什么样例程序跑 sampletest 都是随便跑一定是出题人 sb

A：这个…三个样例都是完全随机的，就是说你随便找个点都有大概 1/4 的概率是解

## 算法 2

把样例程序改成严格的 n^2 次询问

可以得到 10 分

## 算法 3

随机撒一些点判断大小趋势然后退火或爬山或者一些面向数据的奇技淫巧

每个人的实现估计都完全不一样

如果没有严格的复杂度证明的话最多 30

## 算法 4

随机选一个点然后每次找最高的方向爬到没法爬就是一个答案

对不起还是 10 分

Q：怎么可能，一定是没有 srand 出题人 sb

A：你考虑下这种数据

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 50 | 1 | 70 | 71 | 72 | 19 | 92 | 93 | 94 | 37 |
| 51 | 2 | 69 | 18 | 73 | 20 | 91 | 36 | 95 | 38 |
| 52 | 3 | 68 | 17 | 74 | 21 | 90 | 35 | 96 | 39 |
| 53 | 4 | 67 | 16 | 75 | 22 | 89 | 34 | 97 | 40 |
| 54 | 5 | 66 | 15 | 76 | 23 | 88 | 33 | 98 | 41 |
| 55 | 6 | 65 | 14 | 77 | 24 | 87 | 32 | 99 | 42 |
| 56 | 7 | 64 | 13 | 78 | 25 | 86 | 31 | 100 | 43 |
| 57 | 8 | 63 | 12 | 79 | 26 | 85 | 30 | 101 | 44 |
| 58 | 9 | 62 | 11 | 80 | 27 | 84 | 29 | 102 | 45 |
| 59 | 60 | 61 | 10 | 81 | 82 | 83 | 28 | 103 | 104 |

## 给算法 4 打个补丁

随机 1 个点不行，我随机一堆点选最大的这个随机爬山总可以了吧

实践证明，效果非常好

我们来讲道理

假设你随机了 T 个位置

在你最倒霉的情况下数据有一条 O（n^2 长度的单调递增路径）

那么最大的一个位置离路径的终点的距离的期望是 O（n^2/T）

然后你每花 O（1）的代价走一步

总的询问次数是 O（T+n^2/T）当 T=O（n）时询问次数为 O（n）

然后就莫名其妙得到了 50~70 分

# 为什么这个算法无法 AC

来分析一下常数因子

随机选 T 个点询问用了 T 的询问次数

最坏情况每爬 1 步要询问 3 次

所以总的询问次数是 3n^2/T+T

当 T=sqrt(3)*n 的时候取到最小值 2*sqrt(3)*n

Q：这不是很靠谱吗

A：对不起是捆绑测试，这个做法的询问次数是期望的但在多组数据下是用最坏情况算分所以得分在 50~70 之间

Q：出题人 sb

A：如果随机化技巧足够高超的话还是有希望 AC 的，有验题人面向数据用该算法通过该题

# 一个询问次数严格的做法

考虑分治，每次从中间从上到下切一刀，询问这一列上的点的权值

找到权值最大的一个数询问左边和右边的权值

分 3 种情况讨论

1. 左右都比中间小->这个点是一个答案
2. 左边的数小于右边的->递归到右半个

3. 左边的数不小于右边的->递归到左半个

一个显然的结论是这个算法的询问次数是 O（nlogn）

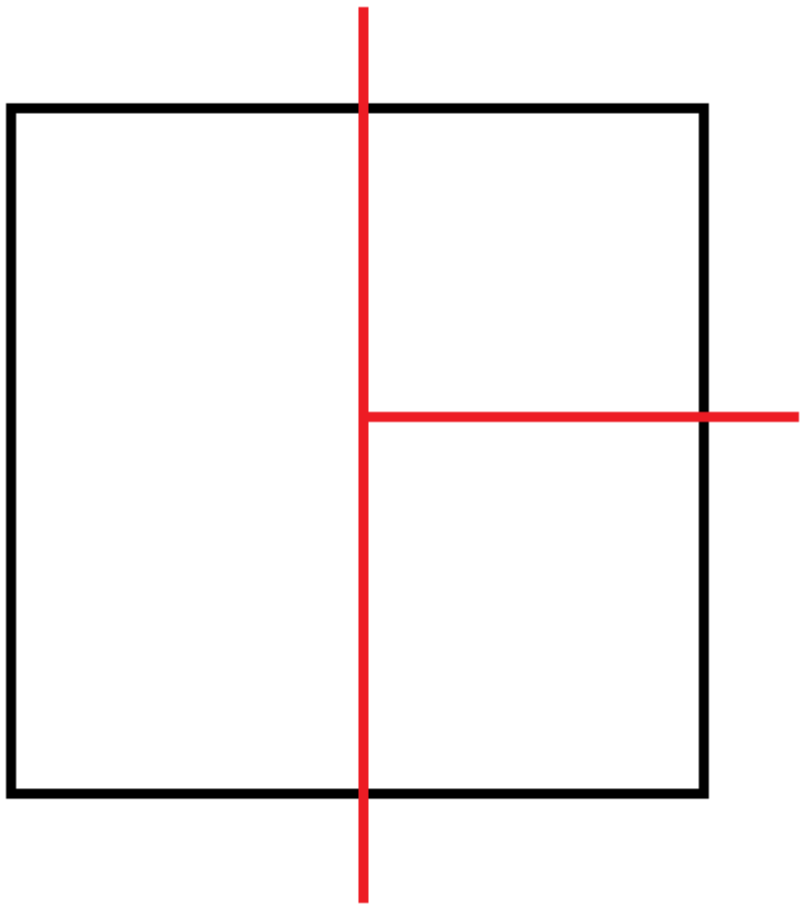
来证明一下正确性

我们把这个矩阵抽象成一个有向图，如果 i 和 j 相邻而且 i 的权值比 j 小则 i 向 j 连边，然后假设左边的数大于这个“权值最大的一个数”那么这个点能到达的任何答案都在左边，右边也是同理，所以该算法是正确的

可以得到 70 分

# 考虑如何把 log 去掉

你考虑这么切



假设能成功判断递归到哪个子矩阵，那么询问次数是 3n+logn 的，可以 AC。

考虑如何判断

我们先算出 mxed 表示当前矩阵的边界最大值，然后考虑从中间切一刀，令 mx 为中间切的一刀上的最大值

如果 mx<mxed 则递归到 mxed 的位置所在的一边

不然的话判断 mx 是否合法，如果不合法询问 mx 左边的数 l 和右边的数 r，如果 l 大就向 l 一边递归否则向 r 一边递归

正确性的证明类似 nlogn 的算法

可以 AC