FOI2022 算法夏令营上机练习（五）

试题讲评

Jul 20, 2022

Contents

1 综合成绩

2 修复白板

3 密室逃脱

4 真心话大冒险

章节目录

1 综合成绩
2 修复白板
3 密室逃脱
4 真心话大冒险

题目简述

有 $4n$ 个数从大到小排列，输出这些数依次加上 $A, B, C, D, A, B, C, D, \cdots$ 后从大到小排序的结果（只输出部分的数单纯只是为了减少输出花费的时间）。

算法一

数据特点：$A = B = C = D$。

算法一

数据特点：$A = B = C = D$。
可以发现所有的数修改之后相对位置是不变的，于是将每个数加上 A 后输出即可。

算法二

数据特点：$x_i \leq 100$。

算法二

数据特点：$x_i \leq 100$。
将每个数加上对应的数之后做桶排序，然后输出。

算法三

数据范围：$n \leq 10^5$。

算法三

数据范围：$n \leq 10^5$。
将每个数加上对应的数之后排序，然后输出。

算法三

数据范围：$n \leq 10^5$。
将每个数加上对应的数之后排序，然后输出。
时间复杂度 $\mathbb{O}(n \log n)$。

算法四

数据范围：$n \leq 10^6$。

算法四

数据范围：$n \leq 10^6$。

注意到输入的时候所有的数是有序的，$A = B = C = D$ 的部分分提示我们在分组并修改后组内的数的相对顺序也是不变的。

算法四

数据范围：$n \leq 10^6$。

注意到输入的时候所有的数是有序的，$A = B = C = D$ 的部分分提示我们在分组并修改后组内的数的相对顺序也是不变的。

于是考虑先对四个组的数分别修改，修改之后再两两归并排序得出最终的序列。

算法四

数据范围：$n \leq 10^6$。

注意到输入的时候所有的数是有序的，$A = B = C = D$ 的部分分提示我们在分组并修改后组内的数的相对顺序也是不变的。

于是考虑先对四个组的数分别修改，修改之后再两两归并排序得出最终的序列。

时间复杂度 $\mathbb{O}(n)$。

算法四

数据范围：$n \leq 10^6$。

注意到输入的时候所有的数是有序的，$A = B = C = D$ 的部分分提示我们在分组并修改后组内的数的相对顺序也是不变的。

于是考虑先对四个组的数分别修改，修改之后再两两归并排序得出最终的序列。

时间复杂度 $\mathbb{O}(n)$。

算法五：算法三 + 大力卡常。

章节目录

1 综合成绩
2 修复白板
3 密室逃脱
4 真心话大冒险

题目简述

有一个 $N\times M$ 的矩阵，每个位置上有一个数，求一个 $n\times m$ 或 $m\times n$ 的子矩阵中数字之和最大为多少（以下只考虑 $n\times m$ 的情况，$n\times m$ 的情况同理）。

算法一

数据范围：$N, M \leq 100$。

算法一

数据范围：$N, M \leq 100$。
枚举子矩阵的左上角位置，将子矩阵内的方格数值相加之后比较得出答案。

算法一

数据范围：$N, M \leq 100$。
枚举子矩阵的左上角位置，将子矩阵内的方格数值相加之后比较得出答案。
时间复杂度 $\mathbb{O}(MNmn)$。

算法二

数据特点：$N = 1$，即退化为一维的情况。

算法二

数据特点：$N=1$，即退化为一维的情况。

对于不带修改的快速求和问题，可以使用前缀和处理。记位置 $(1,i)$ 的数值为 a_i，令 $b_i=a_1+a_2+\cdots+a_i$。那么 a_l 到 a_r 的和就可以表示为 b_r-b_{l-1}。

算法二

数据特点：$N=1$，即退化为一维的情况。

对于不带修改的快速求和问题，可以使用前缀和处理。记位置 $(1,i)$ 的数值为 a_i，令 $b_i=a_1+a_2+\cdots+a_i$。那么 a_l 到 a_r 的和就可以表示为 b_r-b_{l-1}。

于是我们先预处理出这一行数的前缀和，然后枚举子序列的起点，用前缀和相减快速得出子序列中各个位置的数值之和然后比较得出答案。

算法二

数据特点：$N=1$，即退化为一维的情况。

对于不带修改的快速求和问题，可以使用前缀和处理。记位置 $(1,i)$ 的数值为 a_i，令 $b_i=a_1+a_2+\cdots+a_i$。那么 a_l 到 a_r 的和就可以表示为 b_r-b_{l-1}。

于是我们先预处理出这一行数的前缀和，然后枚举子序列的起点，用前缀和相减快速得出子序列中各个位置的数值之和然后比较得出答案。

时间复杂度 $\mathbb{O}(M)$。

算法三

数据特点：$n = 1$。

算法三

数据特点：$n = 1$。

虽然矩阵是二维的，但是求和操作都是一维的，所以对矩阵的每一行分别做前缀和后求解即可。

算法三

数据特点：$n = 1$。

虽然矩阵是二维的，但是求和操作都是一维的，所以对矩阵的每一行分别做前缀和后求解即可。

时间复杂度 $\mathbb{O}(MN)$。

算法四

数据特点：$N, M \leq 3\ 000$。

算法四

数据特点：$N, M \leq 3\ 000$。

考虑如何将前缀和扩展到二维的情况。记原矩阵为 $a_{i,j}$，前缀和后的结果 $b_{i,j}$ 表示以 $(1,1)$ 为左上角、(i,j) 为右下角的矩阵的数值之和。那么根据容斥原理：

$$b_{i,j} = b_{i,j-1} + b_{i-1,j} - b_{i-1,j-1} + a_{i,j}$$

算法四

数据特点：$N, M \leq 3\ 000$。

考虑如何将前缀和扩展到二维的情况。记原矩阵为 $a_{i,j}$，前缀和后的结果 $b_{i,j}$ 表示以 $(1,1)$ 为左上角、(i,j) 为右下角的矩阵的数值之和。那么根据容斥原理：

$$b_{i,j} = b_{i,j-1} + b_{i-1,j} - b_{i-1,j-1} + a_{i,j}$$

以 (x,y) 为左上角、(z,w) 为右下角的子矩阵的数值之和为：

$$b_{z,w} - b_{x-1,w} - b_{z,y-1} + b_{x-1,y-1}$$

算法四

数据特点：$N, M \leq 3\ 000$。

考虑如何将前缀和扩展到二维的情况。记原矩阵为 $a_{i,j}$，前缀和后的结果 $b_{i,j}$ 表示以 $(1,1)$ 为左上角、(i,j) 为右下角的矩阵的数值之和。那么根据容斥原理：

$$b_{i,j} = b_{i,j-1} + b_{i-1,j} - b_{i-1,j-1} + a_{i,j}$$

以 (x,y) 为左上角、(z,w) 为右下角的子矩阵的数值之和为：

$$b_{z,w} - b_{x-1,w} - b_{z,y-1} + b_{x-1,y-1}$$

使用这样的算法，在预处理之后可以做到 $\mathbb{O}(1)$ 的查询。于是直接枚举子矩阵的左上角，然后用二维前缀和求子矩阵的数值之和，最后比较得出答案。

算法四

数据特点：$N, M \leq 3\ 000$。

考虑如何将前缀和扩展到二维的情况。记原矩阵为 $a_{i,j}$，前缀和后的结果 $b_{i,j}$ 表示以 $(1,1)$ 为左上角、(i,j) 为右下角的矩阵的数值之和。那么根据容斥原理：

$$b_{i,j} = b_{i,j-1} + b_{i-1,j} - b_{i-1,j-1} + a_{i,j}$$

以 (x,y) 为左上角、(z,w) 为右下角的子矩阵的数值之和为：

$$b_{z,w} - b_{x-1,w} - b_{z,y-1} + b_{x-1,y-1}$$

使用这样的算法，在预处理之后可以做到 $\mathbb{O}(1)$ 的查询。于是直接枚举子矩阵的左上角，然后用二维前缀和求子矩阵的数值之和，最后比较得出答案。

时间复杂度 $\mathbb{O}(MN)$。

章节目录

1 综合成绩
2 修复白板
3 密室逃脱
4 真心话大冒险

题目简述

有一张有向图（**不保证连通**），每个点的出度均为 1，且第一次到达某一个点时可以获得一个正的收益，求从每个点出发的最大收益。

算法一

数据范围：$n \leq 1\ 000$。

算法一

数据范围：$n \leq 1\ 000$。

由于所有的点的收益都是正的，所以最优策略一定是将一条路径走到收益不会再增加为止。

算法一

数据范围：$n \leq 1\,000$。

由于所有的点的收益都是正的，所以最优策略一定是将一条路径走到收益不会再增加为止。

由于每个点的出度都是 1，我们并不需要选择走哪条边，只需要一直往下走到一个访问过的点停下（因为之后的点一定都不会有收益了）.

算法一

数据范围：$n \leq 1\,000$。

由于所有的点的收益都是正的，所以最优策略一定是将一条路径走到收益不会再增加为止。

由于每个点的出度都是 1，我们并不需要选择走哪条边，只需要一直往下走到一个访问过的点停下（因为之后的点一定都不会有收益了）.

于是我们可以枚举从每一个点出发的情况，由于图上的每个点最多被经过一次，所以时间复杂度为 $\mathbb{O}(n^2)$。

算法一点五

下面给出两种类似但不符合条件的图：

算法一点五

下面给出两种类似但不符合条件的图：

1. n 个点构成一条链。

算法一点五

下面给出两种类似但不符合条件的图：
1. n 个点构成一条链。
2. n 个点构成一棵树。

算法一点五

下面给出两种类似但不符合条件的图：
1. n 个点构成一条链。
2. n 个点构成一棵树。
通过上面这两个例子我们可以发现，在算法一中，许多点的答案都被重复计算了。

算法一点五

下面给出两种类似但不符合条件的图：

1. n 个点构成一条链。
2. n 个点构成一棵树。

通过上面这两个例子我们可以发现，在算法一中，许多点的答案都被重复计算了。

对于上面这两种情况，可以通过逆拓扑序的更新顺序或者记忆化搜索等方法，让复杂度降为 $\mathbb{O}(n)$。

算法二

数据范围：$n \leq 2 \times 10^5$。

算法二

数据范围：$n \leq 2 \times 10^5$。

由于每个点只有一个出度，所以路线并不会出现“分叉”的情况。更准确地说，这张图是由一些环和一些指向环的树构成的。

算法二

数据范围：$n \leq 2 \times 10^5$。

由于每个点只有一个出度，所以路线并不会出现“分叉”的情况。更准确地说，这张图是由一些环和一些指向环的树构成的。

对于树的情况，我们可以用上一页的结论，逆着拓扑序更新。

算法二

数据范围：$n \leq 2 \times 10^5$。

由于每个点只有一个出度，所以路线并不会出现“分叉”的情况。更准确地说，这张图是由一些环和一些指向环的树构成的。

对于树的情况，我们可以用上一页的结论，逆着拓扑序更新。

对于环的情况，最优策略则是将整个环走完。

算法二

数据范围：$n \leq 2 \times 10^5$。

由于每个点只有一个出度，所以路线并不会出现“分叉”的情况。更准确地说，这张图是由一些环和一些指向环的树构成的。

对于树的情况，我们可以用上一页的结论，逆着拓扑序更新。

对于环的情况，最优策略则是将整个环走完。

于是我们可以先预处理出所有的环上的点的收益之和，然后再根据逆拓扑序更新环上的树即可。

算法二

数据范围：$n \leq 2 \times 10^5$。

由于每个点只有一个出度，所以路线并不会出现“分叉”的情况。更准确地说，这张图是由一些环和一些指向环的树构成的。

对于树的情况，我们可以用上一页的结论，逆着拓扑序更新。

对于环的情况，最优策略则是将整个环走完。

于是我们可以先预处理出所有的环上的点的收益之和，然后再根据逆拓扑序更新环上的树即可。

时间复杂度 $\mathbb{O}(n)$。

算法三

如果你学过 Tarjan 的话，这题也可以不需要使用“图由树和环构成”这一性质。对于每一个强连通分量，一定存在的一个最优策略是将整个强连通分量都走完。

算法三

如果你学过 Tarjan 的话，这题也可以不需要使用“图由树和环构成”这一性质。对于每一个强连通分量，一定存在的一个最优策略是将整个强连通分量都走完。

于是可以将每个强连通分量缩成一个点，这个点的收益就是原来图上所有属于这个强连通分量的点的收益之和。

算法三

如果你学过 Tarjan 的话，这题也可以不需要使用“图由树和环构成”这一性质。对于每一个强连通分量，一定存在的一个最优策略是将整个强连通分量都走完。

于是可以将每个强连通分量缩成一个点，这个点的收益就是原来图上所有属于这个强连通分量的点的收益之和。

在缩点之后，整张图就会变成一个有向无环图，进行拓扑序 DP 即可。

章节目录

1 综合成绩
2 修复白板
3 密室逃脱
4 真心话大冒险

题目简述

给定一棵 n 个点的树，边有非负边权且小于 2^{16}，有 m 条形如从 u 到 v 的路径上的边权异或和为 d 的信息。首先判断信息是否存在矛盾，若没有则输出树上各边边权的最值。

算法零

作为一个善良的出题人，这题我并没有卡掉直接输出某一字符串的骗分方法。

算法零

作为一个善良的出题人，这题我并没有卡掉直接输出某一字符串的骗分方法。
实测输出“N0”有 25 分，“Impossib1e”有 30 分。

算法零

作为一个善良的出题人，这题我并没有卡掉直接输出某一字符串的骗分方法。
实测输出“N0”有 25 分，“Impossib1e”有 30 分。
不过，如果你没有发现这两个字符串里面都藏有数字的话……

几个经典技巧

1. 异或运算满足一个性质：$a \hat{} b \hat{} b = a$，即将一个数异或上另一个数两次等于什么都没做。

几个经典技巧

1. 异或运算满足一个性质：$a\hat{}b\hat{}b = a$，即将一个数异或上另一个数两次等于什么都没做。

2. 若树上某一个点到根的路径权值之和为 s_i，那么路径 (u, v) 的上各边的边权之和等于 $s_u + s_v - 2s_{LCA(u,v)}$；

几个经典技巧

1. 异或运算满足一个性质：$a\hat{}b\hat{}b=a$，即将一个数异或上另一个数两次等于什么都没做。

2. 若树上某一个点到根的路径权值之和为 s_i，那么路径 (u,v) 的上各边的边权之和等于 $s_u+s_v-2s_{LCA(u,v)}$；若树上某一个点到根的路径权值的异或和 $val(root,i)=x_i$，那么路径 (u,v) 的上各边的边权的异或和为 $x_u\hat{}x_v\hat{}x_{LCA(u,v)}\hat{}x_{LCA(u,v)}=x_u\hat{}x_v$，和 $LCA(u,v)$ 无关！

几个经典技巧

1. 异或运算满足一个性质：$a \hat{} b \hat{} b = a$，即将一个数异或上另一个数两次等于什么都没做。

2. 若树上某一个点到根的路径权值之和为 s_i，那么路径 (u, v) 的上各边的边权之和等于 $s_u + s_v - 2s_{LCA(u,v)}$；若树上某一个点到根的路径权值的异或和 $val(root, i) = x_i$，那么路径 (u, v) 的上各边的边权的异或和为 $x_u \hat{} x_v \hat{} x_{LCA(u,v)} \hat{} x_{LCA(u,v)} = x_u \hat{} x_v$，和 $LCA(u, v)$ 无关！用这样的方法可以将每条路径的异或和仅与两端点联系起来而弱化 LCA 在求解过程中的作用，同时避免了相对繁琐的 LCA 求解过程。

几个经典技巧

1. 异或运算满足一个性质：$a \hat{} b \hat{} b = a$，即将一个数异或上另一个数两次等于什么都没做。

2. 若树上某一个点到根的路径权值之和为 s_i，那么路径 (u, v) 的上各边的边权之和等于 $s_u + s_v - 2s_{LCA(u,v)}$；若树上某一个点到根的路径权值的异或和 $val(root, i) = x_i$，那么路径 (u, v) 的上各边的边权的异或和为 $x_u \hat{} x_v \hat{} x_{LCA(u,v)} \hat{} x_{LCA(u,v)} = x_u \hat{} x_v$，和 $LCA(u, v)$ 无关！用这样的方法可以将每条路径的异或和仅与两端点联系起来而弱化 LCA 在求解过程中的作用，同时避免了相对繁琐的 LCA 求解过程。

3. 对于路径 (u, v) 和树上的任意一点 t，$val(u, v) = x_u \hat{} x_v = x_u \hat{} x_t \hat{} x_t \hat{} x_v = val(x, t) \hat{} val(t, v)$，所以求一条路径的异或和也可以借助一个中转点求解。

几个经典技巧

1. 异或运算满足一个性质：$a \hat{} b \hat{} b = a$，即将一个数异或上另一个数两次等于什么都没做。

2. 若树上某一个点到根的路径权值之和为 s_i，那么路径 (u, v) 的上各边的边权之和等于 $s_u + s_v - 2s_{LCA(u,v)}$；若树上某一个点到根的路径权值的异或和 $val(root, i) = x_i$，那么路径 (u, v) 的上各边的边权的异或和为 $x_u \hat{} x_v \hat{} x_{LCA(u,v)} \hat{} x_{LCA(u,v)} = x_u \hat{} x_v$，和 $LCA(u, v)$ 无关！用这样的方法可以将每条路径的异或和仅与两端点联系起来而弱化 LCA 在求解过程中的作用，同时避免了相对繁琐的 LCA 求解过程。

3. 对于路径 (u, v) 和树上的任意一点 t，$val(u, v) = x_u \hat{} x_v = x_u \hat{} x_t \hat{} x_t \hat{} x_v = val(x, t) \hat{} val(t, v)$，所以求一条路径的异或和也可以借助一个中转点求解。

4. 对于很多与二进制或位运算相关的问题，由于不涉及进位的问题，我们可以将数位分开，对每一位分别处理。

算法一

考虑使用并查集求解，令 f_u 为点 u 在**并查集中**的父节点，val_u 为点 u 到其**并查集中**的父节点的路径边权异或和。为了简化问题，我们直接考虑路径压缩后的情形。

算法一

考虑使用并查集求解，令 f_u 为点 u 在**并查集中**的父节点，val_u 为点 u 到其**并查集中**的父节点的路径边权异或和。为了简化问题，我们直接考虑路径压缩后的情形。

路径压缩本质上是让每个点都成为与并查集的根直接相连的子节点，那么此时 val_u 也可以表示 u 到并查集的根的边的边权。

算法一

考虑使用并查集求解，令 f_u 为点 u 在**并查集中**的父节点，val_u 为点 u 到其**并查集中**的父节点的路径边权异或和。为了简化问题，我们直接考虑路径压缩后的情形。

路径压缩本质上是让每个点都成为与并查集的根直接相连的子节点，那么此时 val_u 也可以表示 u 到并查集的根的边的边权。

参考代码：

```
int ff(int x) {
    if (x == f[x]) return x;
    int fff = ff(f[x]);
    val[x] ^= val[f[x]];
    return f[x] = fff;
}
```

算法一

在这道题目中，如果我们知道了两个点之间的路径的边权异或和，那么就将这两个点归入同一个并查集。

算法一

在这道题目中，如果我们知道了两个点之间的路径的边权异或和，那么就将这两个点归入同一个并查集。

初始时，每个点都是独立的一个并查集，因为与它相关的连边情况都是未知的。

算法一

在这道题目中，如果我们知道了两个点之间的路径的边权异或和，那么就将这两个点归入同一个并查集。

初始时，每个点都是独立的一个并查集，因为与它相关的连边情况都是未知的。

加入一条信息 (u, v, d) 时，如果此时 u, v 已经在同一个并查集中了，那么根据技巧 3，路径 (u, v) 上边权的异或和为 $val_u \hat{} val_v$，如果这与 d 不同则这条信息与之前的信息矛盾（直接判定为无解的情况）。

算法一

在这道题目中，如果我们知道了两个点之间的路径的边权异或和，那么就将这两个点归入同一个并查集。

初始时，每个点都是独立的一个并查集，因为与它相关的连边情况都是未知的。

加入一条信息 (u, v, d) 时，如果此时 u, v 已经在同一个并查集中了，那么根据技巧 3，路径 (u, v) 上边权的异或和为 $val_u \hat{} val_v$，如果这与 d 不同则这条信息与之前的信息矛盾（直接判定为无解的情况）。

若 u, v 位于不同的并查集中，那么令 $f_{f_u} = f_v$，且这条边的边权 $val_{f_u} = val_u \hat{} val_v \hat{} d$。

算法一

如何判断多解?

算法一

如何判断多解？如果最后所有的点不在同一个并查集中，那么一定存在一些边的边权是推不出来的，即存在多解。

算法一

如何判断多解？如果最后所有的点不在同一个并查集中，那么一定存在一些边的边权是推不出来的，即存在多解。

如何求出每条边的边权？

算法一

如何判断多解？如果最后所有的点不在同一个并查集中，那么一定存在一些边的边权是推不出来的，即存在多解。

如何求出每条边的边权？找到唯一解之后将每条边两端点的 val 值异或一下即可。

算法二

如果你会 2-SAT……

算法二

如果你会 2-SAT……
根据技巧 4，我们接下来直接考虑边权以及询问的答案全部为 0 或 1 的情况。

算法二

如果你会 2-SAT……

根据技巧 4，我们接下来直接考虑边权以及询问的答案全部为 0 或 1 的情况。

根据技巧 2，如果询问路径 (u, v) 得到的答案为 d，那么可以得出的结论是 $x_u \hat{} x_v = d$。此时如果对 x_i 的两种取值分别建一个点，则可以根据这 m 条信息连边跑 2-SAT。

算法二

如果你会 2-SAT……

根据技巧 4，我们接下来直接考虑边权以及询问的答案全部为 0 或 1 的情况。

根据技巧 2，如果询问路径 (u, v) 得到的答案为 d，那么可以得出的结论是 $x_u \hat{} x_v = d$。此时如果对 x_i 的两种取值分别建一个点，则可以根据这 m 条信息连边跑 2-SAT。

如何判断无解?

算法二

如果你会 2-SAT……

根据技巧 4，我们接下来直接考虑边权以及询问的答案全部为 0 或 1 的情况。

根据技巧 2，如果询问路径 (u, v) 得到的答案为 d，那么可以得出的结论是 $x_u \hat{} x_v = d$。此时如果对 x_i 的两种取值分别建一个点，则可以根据这 m 条信息连边跑 2-SAT。

如何判断无解？2-SAT 跑出来无解等价于原图无解。

算法二

如果你会 2-SAT……

根据技巧 4，我们接下来直接考虑边权以及询问的答案全部为 0 或 1 的情况。

根据技巧 2，如果询问路径 (u, v) 得到的答案为 d，那么可以得出的结论是 $x_u \hat{} x_v = d$。此时如果对 x_i 的两种取值分别建一个点，则可以根据这 m 条信息连边跑 2-SAT。

如何判断无解？2-SAT 跑出来无解等价于原图无解。

如何判断多解？

算法二

如果你会 2-SAT……

根据技巧 4，我们接下来直接考虑边权以及询问的答案全部为 0 或 1 的情况。

根据技巧 2，如果询问路径 (u, v) 得到的答案为 d，那么可以得出的结论是 $x_u \hat{} x_v = d$。此时如果对 x_i 的两种取值分别建一个点，则可以根据这 m 条信息连边跑 2-SAT。

如何判断无解？2-SAT 跑出来无解等价于原图无解。

如何判断多解？先跑出一组可行解，然后枚举每一个变量反选的情况，如果存在另外一组可行解，那么存在多解。

算法二

如果你会 2-SAT……

根据技巧 4，我们接下来直接考虑边权以及询问的答案全部为 0 或 1 的情况。

根据技巧 2，如果询问路径 (u, v) 得到的答案为 d，那么可以得出的结论是 $x_u \hat{} x_v = d$。此时如果对 x_i 的两种取值分别建一个点，则可以根据这 m 条信息连边跑 2-SAT。

如何判断无解？2-SAT 跑出来无解等价于原图无解。

如何判断多解？先跑出一组可行解，然后枚举每一个变量反选的情况，如果存在另外一组可行解，那么存在多解。

如何求出每条边的边权？

算法二

如果你会 2-SAT……

根据技巧 4，我们接下来直接考虑边权以及询问的答案全部为 0 或 1 的情况。

根据技巧 2，如果询问路径 (u, v) 得到的答案为 d，那么可以得出的结论是 $x_u \hat{} x_v = d$。此时如果对 x_i 的两种取值分别建一个点，则可以根据这 m 条信息连边跑 2-SAT。

如何判断无解？2-SAT 跑出来无解等价于原图无解。

如何判断多解？先跑出一组可行解，然后枚举每一个变量反选的情况，如果存在另外一组可行解，那么存在多解。

如何求出每条边的边权？找到唯一解之后将每条边两端点的 x 值异或一下即可。