

GBA2026 Tutorial

CUHK, Shenzhen

2026.4.12

- 直接特判每种即可。

- 直接特判每种即可。
- 较好的方法是先判第一位是否为 /，再判缩写第 2 位，描述第 8 位。

B — Bugcat's Counting Game — 猫猫虫报数

- 枚举每一个数是否符合即可，因为倍数，答案 $\leq kx$ 。

- 维护一个值域数组 vis_i 表示该数是否在队列中。

- 维护一个值域数组 vis_i 表示该数是否在队列中。
- 维护数组 a , 左右端点 l, r , 操作一 $r \leftarrow r + 1$, 操作二 $l \leftarrow l + 1$, 操作三查询 a_{l+x-1} 。

- 如果我们将所有的上锁时间都减去下标（显然这并不影响上锁结果），那么上锁过程变成在 $t_s - l$ 时刻上 a_{l+k} 的锁。因此我们将上锁操作转换为在一个时刻同时操作一个区间内的锁。解锁同理。

- 如果我们将所有的上锁时间都减去下标（显然这并不影响上锁结果），那么上锁过程变成在 $t_s - l$ 时刻上 a_{l+k} 的锁。因此我们将上锁操作转换为在一个时刻同时操作一个区间内的锁。解锁同理。
- 我们先将操作拆成上锁操作和解锁操作，上锁时间为 $t_s - l$ ，解锁时间为 $t_e - l$ 。显然时间先的不会影响时间后的。因此我们按照时间排序，直接按照题意模拟即可。

- 令 $dis_{0,i,j}$ 表示从 1 出发到 i , 花了 j 元钱的最小消耗体力。

- 令 $dis_{0,i,j}$ 表示从 1 出发到 i , 花了 j 元钱的最小消耗体力。
- 令 $dis_{1,i,j}$ 表示从 n 出发到 i , 花了 j 元钱的最小消耗体力。

- 令 $dis_{0,i,j}$ 表示从 1 出发到 i , 花了 j 元钱的最小消耗体力。
- 令 $dis_{1,i,j}$ 表示从 n 出发到 i , 花了 j 元钱的最小消耗体力。
- 建一张 $n \times w$ 的图, 把打车连成 $(x, a) \rightarrow (y, a + 1)$ 边权为 0 的边。

- 令 $dis_{0,i,j}$ 表示从 1 出发到 i , 花了 j 元钱的最小消耗体力。
- 令 $dis_{1,i,j}$ 表示从 n 出发到 i , 花了 j 元钱的最小消耗体力。
- 建一张 $n \times w$ 的图, 把打车连成 $(x, a) \rightarrow (y, a + 1)$ 边权为 0 的边。
- 从 1, n 开始分别跑最短路。

E — Bugcat' s Gathering — 猫猫虫聚会

- 令 $dis_{0,i,j}$ 表示从 1 出发到 i , 花了 j 元钱的最小消耗体力。
- 令 $dis_{1,i,j}$ 表示从 n 出发到 i , 花了 j 元钱的最小消耗体力。
- 建一张 $n \times w$ 的图, 把打车连成 $(x, a) \rightarrow (y, a + 1)$ 边权为 0 的边。
- 从 1, n 开始分别跑最短路。
- 取后缀 min, 枚举相遇点, 在相遇点上枚举其中一只猫猫虫花费, 答案 $\min \{dis_{0,i,x} + dis_{1,i,w-x}\}$

- 若当前序列左右不同，放较小的肯定更优。

F — Card Game — 卡牌游戏

- 若当前序列左右不同，放较小的肯定更优。
- 若相同，从两端不断搜索至不同为止，取更小那端第一个即可。

F — Card Game — 卡牌游戏

- 若当前序列左右不同，放较小的肯定更优。
- 若相同，从两端不断搜索至不同为止，取更小那端第一个即可。
- 二分哈希找第一个不同位置。

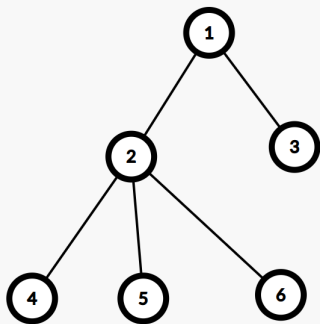
- 障碍只能放在当前无人机上一行或两行的位置，所以只有当前无人机上方一行的位置可能被提前放障碍，且最多放一个。

- 障碍只能放在当前无人机上一行或两行的位置，所以只有当前无人机上方一行的位置可能被提前放障碍，且最多放一个。
- 于是设 $dp_{i,j,0/1/2/3} = true/false$ 表示无人机在 (i, j) ，上方一行没被放/放在 $(i-1, j+t)$ ($t \in -1, 0, 1$)，是否能突围。

- 障碍只能放在当前无人机上一行或两行的位置，所以只有当前无人机上方一行的位置可能被提前放障碍，且最多放一个。
- 于是设 $dp_{i,j,0/1/2/3} = true/false$ 表示无人机在 (i, j) ，上方一行没被放/放在 $(i-1, j+t)$ ($t \in -1, 0, 1$)，是否能突围。
- 转移时分别枚举放在上两行的 5 种位置，上一行的 3 种位置。

- 障碍只能放在当前无人机上一行或两行的位置，所以只有当前无人机上方一行的位置可能被提前放障碍，且最多放一个。
- 于是设 $dp_{i,j,0/1/2/3} = true/false$ 表示无人机在 (i, j) ，上方一行没被放/放在 $(i-1, j+t)$ ($t \in -1, 0, 1$)，是否能突围。
- 转移时分别枚举放在上两行的 5 种位置，上一行的 3 种位置。
- 输出方案只需要在转移时记录放置位置即可。

- 因为宽度充足，一种基础的想法是类似线段树构造，即将一个点的高度设为 1，宽度设为子树大小 -1 （或者 dp 出一个宽度）。

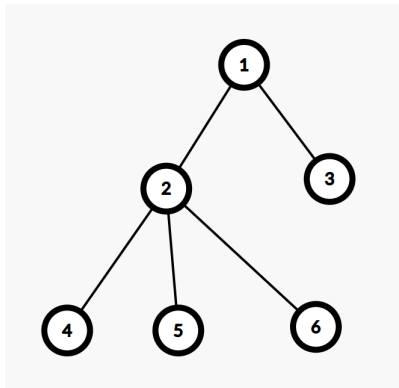


11111

22223

45600

- 因为宽度充足，一种基础的想法是类似线段树构造，即将一个点的高度设为 1，宽度设为子树大小 -1 （或者 dp 出一个宽度）。



11111
22223
45600

- 只有最长链会突出，于是先找到最长链。

- 提供两种处理。

- 提供两种处理。
 - ① 可以发现如果最长链 $> n$, 那么最长链一定有若干度为 2 的点。这些点可以缩起来。

- 提供两种处理。

- ① 可以发现如果最长链 $> n$ ，那么最长链一定有若干度为 2 的点。这些点可以缩起来。

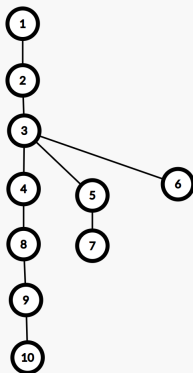
- ② 可以按以下方式构造最长链，然后把点延伸出来。



- 提供两种处理。

- ① 可以发现如果最长链 $> n$ ，那么最长链一定有若干度为 2 的点。这些点可以缩起来。

- ② 可以按以下方式构造最长链，然后把点延伸出来。



```

1 2 2 0 0 0 0 0 0 0
2 2 3 3 3 4 4 8 8 0
0 0 5 6 4 4 8 8 9 0
0 0 7 0 0 0 0 0 10 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
    
```

```

1 2 2 0 0 0 0 0 0 0
2 2 3 3 3 3 4 4 0 0
0 0 5 5 6 4 4 8 9 9
0 0 7 0 0 0 0 9 9 10
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
    
```

I — Bugcat's Adventure — 猫猫虫冒险

- 先考虑如果没有随从，也就是没有修改的话怎么做。

I — Bugcat's Adventure — 猫猫虫冒险

- 先考虑如果没有随从，也就是没有修改的话怎么做。
- 我们发现一个性质，如果 $x \geq 2^i, y \geq 2^i$ 并且 x 能将 y 吃掉，那么 $x \geq 2^{i+1}$ 。

I — Bugcat's Adventure — 猫猫虫冒险

- 先考虑如果没有随从，也就是没有修改的话怎么做。
- 我们发现一个性质，如果 $x \geq 2^i, y \geq 2^i$ 并且 x 能将 y 吃掉，那么 $x \geq 2^{i+1}$ 。
- 这启发我们按照二进制最高位进行分层。形式化的说，对于 i ，我们考虑所有满足 $h_u < 2^i$ 的点 u 构成的子图，我们假设这张图为 G_i 。然后我们考虑一层一层来做这个题。

I — Bugcat's Adventure — 猫猫虫冒险

- 先考虑如果没有随从，也就是没有修改的话怎么做。
- 我们发现一个性质，如果 $x \geq 2^i, y \geq 2^i$ 并且 x 能将 y 吃掉，那么 $x \geq 2^{i+1}$ 。
- 这启发我们按照二进制最高位进行分层。形式化的说，对于 i ，我们考虑所有满足 $h_u < 2^i$ 的点 u 构成的子图，我们假设这张图为 G_i 。然后我们考虑一层一层来做这个题。
- 这样，如果我们当前在点 u ，血量为 y 。假设 $2^i \leq y \leq 2^{i+1} - 1$ ，那么 u 可以走到在图 G_i 中和它相邻的所有点。

- 先考虑如果没有随从，也就是没有修改的话怎么做。
- 我们发现一个性质，如果 $x \geq 2^i, y \geq 2^i$ 并且 x 能将 y 吃掉，那么 $x \geq 2^{i+1}$ 。
- 这启发我们按照二进制最高位进行分层。形式化的说，对于 i ，我们考虑所有满足 $h_u < 2^i$ 的点 u 构成的子图，我们假设这张图为 G_i 。然后我们考虑一层一层来做这个题。
- 这样，如果我们当前在点 u ，血量为 y 。假设 $2^i \leq y \leq 2^{i+1} - 1$ ，那么 u 可以走到在图 G_i 中和它相邻的所有点。
- 在走完相邻的所有点后，我们只需要判断图 G_i 中当前连通块能走到的 h_u 最小值。也就是说，找到满足 $2^i \leq h_u \leq 2^{i+1} - 1$ 并且能够走到的点 u 中，最小的 h_u 。

I — Bugcat's Adventure — 猫猫虫冒险

- 先考虑如果没有随从，也就是没有修改的话怎么做。
- 我们发现一个性质，如果 $x \geq 2^i, y \geq 2^i$ 并且 x 能将 y 吃掉，那么 $x \geq 2^{i+1}$ 。
- 这启发我们按照二进制最高位进行分层。形式化的说，对于 i ，我们考虑所有满足 $h_u < 2^i$ 的点 u 构成的子图，我们假设这张图为 G_i 。然后我们考虑一层一层来做这个题。
- 这样，如果我们当前在点 u ，血量为 y 。假设 $2^i \leq y \leq 2^{i+1} - 1$ ，那么 u 可以走到在图 G_i 中和它相邻的所有点。
- 在走完相邻的所有点后，我们只需要判断图 G_i 中当前连通块能走到的 h_u 最小值。也就是说，找到满足 $2^i \leq h_u \leq 2^{i+1} - 1$ 并且能够走到的点 u 中，最小的 h_u 。
- 如果吃不掉 h_u ，那么必然走不到下一层。如果能吃掉 h_u ，则必然能进入下一层。我们只需维护图 G_i 中每个连通块的和，和每个连通块能走到的最小 h_u ，这个可以直接初始化。因此我们每次走到下一层都是 $O(1)$ 的。

- 接下来考虑带修怎么做。对于一个随从的修改，我们分成两部分考虑。假设该随从的血量为 y 。

- 接下来考虑带修怎么做。对于一个随从的修改，我们分成两部分考虑。假设该随从的血量为 y 。
- 首先对于连通块的和，我们对于所有满足 $2^i > y$ 的 G_i 都在点 x 的连通块加上 y 即可。

- 接下来考虑带修怎么做。对于一个随从的修改，我们分成两部分考虑。假设该随从的血量为 y 。
- 首先对于连通块的和，我们对于所有满足 $2^i > y$ 的 G_i 都在点 x 的连通块加上 y 即可。
- 其次对于连通块的最小值，我们直接遍历点 u 所有的边，然后修改最小值即可。因为每个点最多只会有一个随从，所以均摊是 $O(m)$ 的。

- 接下来考虑带修怎么做。对于一个随从的修改，我们分成两部分考虑。假设该随从的血量为 y 。
- 首先对于连通块的和，我们对于所有满足 $2^i > y$ 的 G_i 都在点 x 的连通块加上 y 即可。
- 其次对于连通块的最小值，我们直接遍历点 u 所有的边，然后修改最小值即可。因为每个点最多只会有一个随从，所以均摊是 $O(m)$ 的。
- 最终时间复杂度 $O(m \log w\alpha(n) + q \log w\alpha(n))$ 。

- 考虑求摸到第 i 张牌前已经合法的方案数 f_i , 答案就是 $\frac{1}{(4n)!} \sum i \times (f_i - f_{i-1})$ 。

- 考虑求摸到第 i 张牌前已经合法的方案数 f_i , 答案就是 $\frac{1}{(4n)!} \sum i \times (f_i - f_{i-1})$ 。
- 考虑求不合法方案数, 考虑 n 种不同的牌。

- 考虑求摸到第 i 张牌前已经合法的方案数 f_i ，答案就是 $\frac{1}{(4n)!} \sum i \times (f_i - f_{i-1})$ 。
- 考虑求不合法方案数，考虑 n 种不同的牌。
 - 摸了大于等于 3 张的生成函数 $4x^3 + x^4$ 。

- 考虑求摸到第 i 张牌前已经合法的方案数 f_i ，答案就是 $\frac{1}{(4n)!} \sum i \times (f_i - f_{i-1})$ 。
- 考虑求不合法方案数，考虑 n 种不同的牌。
 - 摸了大于等于 3 张的生成函数 $4x^3 + x^4$ 。
 - 摸了小于 2 张的生成函数 $1 + 4x$ 。

- 考虑求摸到第 i 张牌前已经合法的方案数 f_i , 答案就是 $\frac{1}{(4n)!} \sum i \times (f_i - f_{i-1})$ 。
- 考虑求不合法方案数, 考虑 n 种不同的牌。
 - 摸了大于等于 3 张的生成函数 $4x^3 + x^4$ 。
 - 摸了小于 2 张的生成函数 $1 + 4x$ 。
 - 摸了小于 2 张的生成函数 $1 + 4x + 6x^2$ 。

- 考虑求摸到第 i 张牌前已经合法的方案数 f_i , 答案就是 $\frac{1}{(4n)!} \sum i \times (f_i - f_{i-1})$ 。
- 考虑求不合法方案数, 考虑 n 种不同的牌。
 - 摸了大于等于 3 张的生成函数 $4x^3 + x^4$ 。
 - 摸了小于 2 张的生成函数 $1 + 4x$ 。
 - 摸了小于 2 张的生成函数 $1 + 4x + 6x^2$ 。
- 讨论每种情况已经有 0/1/2/3/4 种牌摸了大于等于三张。

- 考虑求摸到第 i 张牌前已经合法的方案数 f_i , 答案就是 $\frac{1}{(4n)!} \sum i \times (f_i - f_{i-1})$ 。
- 考虑求不合法方案数, 考虑 n 种不同的牌。
 - 摸了大于等于 3 张的生成函数 $4x^3 + x^4$ 。
 - 摸了小于 2 张的生成函数 $1 + 4x$ 。
 - 摸了小于 2 张的生成函数 $1 + 4x + 6x^2$ 。
- 讨论每种情况已经有 0/1/2/3/4 种牌摸了大于等于三张。
 - $i = 0 \sim 3$: $(1 + 4x + 6x^2)^{n-i} (4x^3 + x^4)^i$

- 考虑求摸到第 i 张牌前已经合法的方案数 f_i , 答案就是 $\frac{1}{(4n)!} \sum i \times (f_i - f_{i-1})$ 。
- 考虑求不合法方案数, 考虑 n 种不同的牌。
 - 摸了大于等于 3 张的生成函数 $4x^3 + x^4$ 。
 - 摸了小于 2 张的生成函数 $1 + 4x$ 。
 - 摸了小于 2 张的生成函数 $1 + 4x + 6x^2$ 。
- 讨论每种情况已经有 0/1/2/3/4 种牌摸了大于等于三张。
 - $i = 0 \sim 3$: $(1 + 4x + 6x^2)^{n-i} (4x^3 + x^4)^i$
 - $i = 4$: $(1 + 4x)^{n-4} (4x^3 + x^4)^4$

- 现在的问题是快速求 $(1 + 4x + 6x^2)$ 的 k 次方 $f(k)$ 。

- 现在的问题是快速求 $(1 + 4x + 6x^2)$ 的 k 次方 $f(k)$ 。
- 联立 $((1 + 4x + 6x^2)f(k))' = (12x + 4)(k + 1)f(k)$

- 现在的问题是快速求 $(1 + 4x + 6x^2)$ 的 k 次方 $f(k)$ 。
- 联立 $((1 + 4x + 6x^2)f(k))' = (12x + 4)(k + 1)f(k)$
- 考虑第 i 位可得
$$(i + 1)(f(k)_{i+1} + 4f(k)_i + 6f(k)_{i-1}) = (k + 1)(12f(k)_{i-1} + 4f(k)_i)$$

- 现在的问题是快速求 $(1 + 4x + 6x^2)$ 的 k 次方 $f(k)$ 。
- 联立 $((1 + 4x + 6x^2)f(k))' = (12x + 4)(k + 1)f(k)$
- 考虑第 i 位可得
$$(i + 1)(f(k)_{i+1} + 4f(k)_i + 6f(k)_{i-1}) = (k + 1)(12f(k)_{i-1} + 4f(k)_i)$$
- $f(k)_i = \frac{1}{i}((4k + 4 - 4i)f(k)_{i-1} + (12k + 12 - 6i)f(k)_{i-2})$ 。

K — Moonlit Trees — 落月摇情满江树

- 先考虑 $n = 0$ 。

K — Moonlit Trees — 落月摇情满江树

- 先考虑 $n = 0$ 。
- 容易发现 $type = 2$ 没有用。

K — Moonlit Trees — 落月摇情满江树

- 先考虑 $n = 0$ 。
- 容易发现 $type = 2$ 没有用。
- 考虑生成的排列有哪些可以通过 $type = 1$ 构造。

- 先考虑 $n = 0$ 。
- 容易发现 $type = 2$ 没有用。
- 考虑生成的排列有哪些可以通过 $type = 1$ 构造。
 - ① 第一位必须为 1，第二位必须为 2。

- 先考虑 $n = 0$ 。
- 容易发现 $type = 2$ 没有用。
- 考虑生成的排列有哪些可以通过 $type = 1$ 构造。
 - ① 第一位必须为 1，第二位必须为 2。
 - ② 从前往后维护单调栈，贪心考虑。栈尾是 u ，当前加入 v 。pop 到 $u < v$ 为止，并把 u 和 v 设为同层儿子。而若 u 没有儿子，将 v 设为 u 的儿子。

- 先考虑 $n = 0$ 。
- 容易发现 $type = 2$ 没有用。
- 考虑生成的排列有哪些可以通过 $type = 1$ 构造。
 - ① 第一位必须为 1，第二位必须为 2。
 - ② 从前往后维护单调栈，贪心考虑。栈尾是 u ，当前加入 v 。pop 到 $u < v$ 为止，并把 u 和 v 设为同层儿子。而若 u 没有儿子，将 v 设为 u 的儿子。
- 设计一个 dp 状态其中有一维表示当前的单调栈长度，但是会出现向上消耗一些栈的元素，向下新增一些栈的元素的情况。

- 先考虑 $n = 0$ 。
- 容易发现 $type = 2$ 没有用。
- 考虑生成的排列有哪些可以通过 $type = 1$ 构造。
 - ① 第一位必须为 1，第二位必须为 2。
 - ② 从前往后维护单调栈，贪心考虑。栈尾是 u ，当前加入 v 。pop 到 $u < v$ 为止，并把 u 和 v 设为同层儿子。而若 u 没有儿子，将 v 设为 u 的儿子。
- 设计一个 dp 状态其中有一维表示当前的单调栈长度，但是会出现向上消耗一些栈的元素，向下新增一些栈的元素的情况。
- 虑一个序列的所有“前缀 min 位置”，下一个前缀 min 位置会把所有之前向下新增的东西全部弹栈，只需要考虑向上消耗。

- 于是有一个 $O(m^4)$ 的 dp: 令 $dp_{i,j}$ 表示共 i 位, 会向上消耗 j 个栈的位置, 的好排列方案数.

- 于是有一个 $O(m^4)$ 的 dp: 令 $dp_{i,j}$ 表示共 i 位, 会向上消耗 j 个栈的位置, 的好排列方案数.
- 最后一个区间 $f_{i,j} = \sum_{t \leq i} \sum_{p \leq j} \binom{i-1}{t-1} f_{t-1,p} f_{i-t,j-p}$

- 于是有一个 $O(m^4)$ 的 dp: 令 $dp_{i,j}$ 表示共 i 位, 会向上消耗 j 个栈的位置, 的好排列方案数.
- 最后一个区间 $f_{i,j} = \sum_{t \leq i} \sum_{p \leq j} \binom{i-1}{t-1} f_{t-1,p} f_{i-t,j-p}$
- $n \neq 0$ 只需按前缀 min 区间递归即可, 从区间最小分半也可。

- 于是有一个 $O(m^4)$ 的 dp: 令 $dp_{i,j}$ 表示共 i 位, 会向上消耗 j 个栈的位置, 的好排列方案数.
- 最后一个区间 $f_{i,j} = \sum_{t \leq i} \sum_{p \leq j} \binom{i-1}{t-1} f_{t-1,p} f_{i-t,j-p}$
- $n \neq 0$ 只需按前缀 min 区间递归即可, 从区间最小分半也可。
- 合并 dp 只有 $O(m)$ 次, 复杂度 $O((\max m)^4 + \sum (n+m)m)$ 。

- 特判 $n = 1/2$ 。

- 特判 $n = 1/2$ 。
- 首先 $(4i) \text{ xor } (4i + 1) \text{ xor } (4i + 2) \text{ xor } (4i + 3) = 0$ ，而所猫猫虫属性值 xor 和 $S = 0$ 。

- 特判 $n = 1/2$ 。
- 首先 $(4i) \text{ xor } (4i + 1) \text{ xor } (4i + 2) \text{ xor } (4i + 3) = 0$ ，而所猫猫虫属性值 xor 和 $S = 0$ 。
- 桌号总 xor 和 $(2^{n-1}) \text{ xor } (2^{n-1} + 1) \text{ xor } x = x \text{ xor } 1$ ，只有 $x = 1$ 时可能有解。

- 特判 $n = 1/2$ 。
- 首先 $(4i) \text{ xor } (4i + 1) \text{ xor } (4i + 2) \text{ xor } (4i + 3) = 0$ ，而所猫猫虫属性值 xor 和 $S = 0$ 。
- 桌号总 xor 和 $(2^{n-1}) \text{ xor } (2^{n-1} + 1) \text{ xor } x = x \text{ xor } 1$ ，只有 $x = 1$ 时可能有解。
- 即桌号为 $2 \sim 2^{n-1} + 1$ 。从 $n - 1$ 推到 n ，保留 $2 \sim 2^{n-2} - 1$ 。

- 特判 $n = 1/2$ 。
- 首先 $(4i) \text{ xor } (4i + 1) \text{ xor } (4i + 2) \text{ xor } (4i + 3) = 0$ ，而所猫猫虫属性值 xor 和 $S = 0$ 。
- 桌号总 xor 和 $(2^{n-1}) \text{ xor } (2^{n-1} + 1) \text{ xor } x = x \text{ xor } 1$ ，只有 $x = 1$ 时可能有解。
- 即桌号为 $2 \sim 2^{n-1} + 1$ 。从 $n - 1$ 推到 n ，保留 $2 \sim 2^{n-2} - 1$ 。
- 考虑剩下一张桌子放一只 $2^{n-1} | 2^{n-2} | s_i$ 和 $2^{n-1} | t_i$ ，这样数量合适。

- 特判 $n = 1/2$ 。
- 首先 $(4i) \text{ xor } (4i + 1) \text{ xor } (4i + 2) \text{ xor } (4i + 3) = 0$ ，而所猫猫虫属性值 xor 和 $S = 0$ 。
- 桌号总 xor 和 $(2^{n-1}) \text{ xor } (2^{n-1} + 1) \text{ xor } x = x \text{ xor } 1$ ，只有 $x = 1$ 时可能有解。
- 即桌号为 $2 \sim 2^{n-1} + 1$ 。从 $n - 1$ 推到 n ，保留 $2 \sim 2^{n-2} - 1$ 。
- 考虑剩下一张桌子放一只 $2^{n-1} | 2^{n-2} | s_i$ 和 $2^{n-1} | t_i$ ，这样数量合适。
- 令 $S = 2^{n-2} - 1$ ，那么 $(4i) \text{ xor } (2i) \neq (4i + 2) \text{ xor } (2i + 1)$ ，
 $((4i) \text{ xor } (2i)) + 1 \neq (4i + 2) \text{ xor } (2i + 1)$

- 特判 $n = 1/2$ 。
- 首先 $(4i) \text{ xor } (4i + 1) \text{ xor } (4i + 2) \text{ xor } (4i + 3) = 0$ ，而所猫猫虫属性值 xor 和 $S = 0$ 。
- 桌号总 xor 和 $(2^{n-1}) \text{ xor } (2^{n-1} + 1) \text{ xor } x = x \text{ xor } 1$ ，只有 $x = 1$ 时可能有解。
- 即桌号为 $2 \sim 2^{n-1} + 1$ 。从 $n - 1$ 推到 n ，保留 $2 \sim 2^{n-2} - 1$ 。
- 考虑剩下一张桌子放一只 $2^{n-1} | 2^{n-2} | s_i$ 和 $2^{n-1} | t_i$ ，这样数量合适。
- 令 $S = 2^{n-2} - 1$ ，那么 $(4i) \text{ xor } (2i) \neq (4i + 2) \text{ xor } (2i + 1)$ ，
 $((4i) \text{ xor } (2i)) + 1 \neq (4i + 2) \text{ xor } (2i + 1)$
- 所以 $((2i) \text{ xor } (i)) \text{ xor } 2$ 之后能变成另一半。

- 特判 $n = 1/2$ 。
- 首先 $(4i) \text{ xor } (4i + 1) \text{ xor } (4i + 2) \text{ xor } (4i + 3) = 0$ ，而所猫猫虫属性值 xor 和 $S = 0$ 。
- 桌号总 xor 和 $(2^{n-1}) \text{ xor } (2^{n-1} + 1) \text{ xor } x = x \text{ xor } 1$ ，只有 $x = 1$ 时可能有解。
- 即桌号为 $2 \sim 2^{n-1} + 1$ 。从 $n - 1$ 推到 n ，保留 $2 \sim 2^{n-2} - 1$ 。
- 考虑剩下一张桌子放一只 $2^{n-1} | 2^{n-2} | s_i$ 和 $2^{n-1} | t_i$ ，这样数量合适。
- 令 $S = 2^{n-2} - 1$ ，那么 $(4i) \text{ xor } (2i) \neq (4i + 2) \text{ xor } (2i + 1)$ ，
 $((4i) \text{ xor } (2i)) + 1 \neq (4i + 2) \text{ xor } (2i + 1)$
- 所以 $((2i) \text{ xor } (i)) \text{ xor } 2$ 之后能变成另一半。
- 用这种方式生成前半段，后半段将其中一个数全部 $\text{xor}(S \text{ xor } 2)$ 再倒序即可。

- 特判 $n = 1/2$ 。
- 首先 $(4i) \text{ xor } (4i + 1) \text{ xor } (4i + 2) \text{ xor } (4i + 3) = 0$ ，而所猫猫虫属性值 xor 和 $S = 0$ 。
- 桌号总 xor 和 $(2^{n-1}) \text{ xor } (2^{n-1} + 1) \text{ xor } x = x \text{ xor } 1$ ，只有 $x = 1$ 时可能有解。
- 即桌号为 $2 \sim 2^{n-1} + 1$ 。从 $n - 1$ 推到 n ，保留 $2 \sim 2^{n-2} - 1$ 。
- 考虑剩下一张桌子放一只 $2^{n-1} | 2^{n-2} | s_i$ 和 $2^{n-1} | t_i$ ，这样数量合适。
- 令 $S = 2^{n-2} - 1$ ，那么 $(4i) \text{ xor } (2i) \neq (4i + 2) \text{ xor } (2i + 1)$ ，
 $((4i) \text{ xor } (2i)) + 1 \neq (4i + 2) \text{ xor } (2i + 1)$
- 所以 $((2i) \text{ xor } (i)) \text{ xor } 2$ 之后能变成另一半。
- 用这种方式生成前半段，后半段将其中一个数全部 $\text{xor}(S \text{ xor } 2)$ 再倒序即可。
- 小调一下边界。