Puzzle: X-Sums Sudoku

An $n \times m$ sudoku puzzle is a grid consisting of $m \times n$ regions, and each region contains $n \times m$ cells. Hence an $n \times m$ sudoku puzzle contains $nm \times nm$ cells. Every integer from 1 to nm occurs exactly once in each row, each column, and each region of an $n \times m$ sudoku puzzle.

Listing the integers in a row or a column starting from some direction as a sequence of length nm, X is the first integer of the sequence, and X-sum is the sum of the first X integers of the sequence.

	1	6	11	20	22	32	34	36	
1	1	2	3	4	5	6	7	8	36
8	3	4	1	2	7	8	5	6	29
27	5	6	7	8	1	2	3	4	10
34	7	8	5	6	3	4	1	2	3
3	2	1	4	3	6	5	8	7	34
10	4	3	2	1	8	7	6	5	27
29	6	5	8	7	2	1	4	3	8
36	8	7	6	5	4	3	2	1	1
	36	34	32	22	20	11	6	1	

The above figure is a 4×2 sudoku puzzle with X-sums. The 7-th row listed from right to left is $[3, 4, 1, 2, 7, 8, 5, 6]$ and the first integer X is 3, so the X-sum of the 7-th row from the direction right is $8 = 3 + 4 + 1$.

Given two positive integers n and m, a direction d, and an index x, you need to find the X-sum of the x-th row or x-th column from the direction d in **the lexicographically smallest** $2^n \times 2^m$ sudoku.

Denoting $a_{i,j}$ as the i-th row and the j-th column of a sudoku puzzle a, a sudoku puzzle a is lexicographically smaller than a sudoku puzzle b of the same size if there exists i and j satisfying that $a_{i,j} < b_{i,j}$, that $a_{x,y} = b_{x,y}$ for all $x < i$, and that $a_{x,y} = b_{x,y}$ for all $x = i$ and $y < j$. You can find that the above is the lexicographically smallest 4×2 sudoku puzzle.

直接找规律：

$$A_{x=2^n a+b, y=2^m c+d} = (a \oplus d) + 2^m \cdot (b \oplus c) \tag{1}$$

显然 $+$ 的两边是独立的，行和列的求和是对称的。而对于一列的前缀求和，c, d 固定。

1. $a \oplus d$ 的部分：
    1. 对于 $A_{x=2^m e+f, y}(e < a)$ 的部分：每个的 f 都是 2^n 种，直接算就好 $\sum_{i=0}^{a}(i \oplus d)$ 就好，这个可以直接枚举 i 的自由位来算。
    2. 否则，单独算一下就好。
2. $b \oplus c$ 的部分：
    1. 对于 $A_{x=2^m e+f, y}(e < a)$ 的部分，每个 f 对应的 e 都是有 a 种，相当于把 c 的后 n 位变成任意的，求和。
    2. 否则，贡献系数都是 1，依然是转成算 $\sum_{i=0}^{a}(i \oplus d)$。

复杂度 $O(n+m)$。

puzzle: Patrick’ s Parabox

Patrick's Parabox is a *Sokoban*-like game. A Sokoban puzzle is a grid, and each cell is a wall or a floor. There are several boxes and a player in some distinct floor cells, and they can not move to the wall cells or coincide. You can control the player to move in one of four directions, left, right, up, and down. When the player touches a box, it can push the box. The target is to move all boxes to some target cells.

Please read the following rules carefully. They may be different from the usual rules.

In this problem, there is **only one** box, and the box is the grid itself. That means if the player moves out of the grid, they may be "teleported" to a cell adjacent to the box; if the player moves to the box, they may be "teleported" to a cell on the boundary of the grid. Besides, there is also a target cell for the player. The player needs to move to the target cell at the end too.

Given a puzzle, you need to find the **minimum number of times to push** the box, such that the box and the player can arrive to their respective target cells.

The following are the detailed and formal rules.

Consider an $n \times m$ grid. Denote (i, j) as the cell in i-th row and j-th column. The rows are numbered $1, 2, \ldots, n$ from top to bottom, and the columns are numbered $1, 2, \ldots, m$ from left to right.

Denote W, S, A, and D as the control commands, which mean to move up, down, left, and right respectively.

Define that $v_{\mathrm{W}} = (-1, 0)$, $v_{\mathrm{S}} = (1, 0)$, $v_{\mathrm{A}} = (0, -1)$, $v_{\mathrm{D}} = (0, 1)$.

Define that $w_{\mathrm{W}} = (n, \left\lceil \frac{m}{2} \right\rceil)$, $w_{\mathrm{S}} = (1, \left\lceil \frac{m}{2} \right\rceil)$, $w_{\mathrm{A}} = (\left\lceil \frac{n}{2} \right\rceil, m)$, $w_{\mathrm{D}} = (\left\lceil \frac{n}{2} \right\rceil, 1)$.

In each operation, you can choose one of the control commands c, one of W, S, A, and D. Denote p as the cell which contains the player and b as the cell which contains the box before the operation:

- If $p + v_c = b$ and $b + v_c$ is a floor cell, the player moves to $p + v_c$ and the box moves to $b + v_c$. **Only this case** counts towards the answer, and so has to happen the minimum possible number of times.
- If $p + v_c$ is a wall cell, nothing happens.
- If $p + v_c$ is a floor cell and $p + v_c \neq b$, the player moves to $p + v_c$.
- If $p + v_c = b$ and $b + v_c$ is outside the grid, nothing happens.
- If $p + v_c = b$, $b + v_c$ is a wall cell and w_c is a wall cell, nothing happens.
- If $p + v_c = b$, $b + v_c$ is a wall cell and w_c is a floor cell, the player moves to w_c.
- if $p + v_c$ is outside the grid and $b + v_c$ is a wall cell, nothing happens.
- if $p + v_c$ is outside the grid and $b + v_c$ is outside the grid, nothing happens.
- if $p + v_c$ is outside the grid and $b + v_c$ is a floor cell, the player moves to $b + v_c$.

Note that the above are listed for covering all possibilities, but the operations are valid in only four of them (in other cases, nothing happens).

Input

我们直接记 $(u, 0/1/2/3)$，表示箱子在 u，人在 u 四个方向的哪个连通块里。考虑怎么连边，一种是推箱子的边，一种是绕到了另一个连通块里的边。第二种边怎么连？用圆方树判断一下就好了。对于飞走的边，可以用树形 dp 判一下。

Puzzle: Hearthstone

Hearthstone is one of the popular video games. **Please read the following rules carefully. They are different from the usual rules.**

There are n kinds of secret cards numbered 1, 2, ..., n. There are two types of events about secrets:

- `add`: Add a secret with an unknown number into the hero zone. No two secrets with the same number can be in the hero zone simultaneously.
- `test` x y: Test whether secret x exists. If secret x exists, then $y = 1$ and secret x is removed from the hero zone; otherwise, $y = 0$. Note that whatever y is, secret x does not exist in the hero zone after `testing` x.

An event sequence $E = [e_1, \ldots, e_m]$ is valid if and only if it is possible to assign a number from 1 to n for each `add` event and perform the events $e_1, e_2, \ldots, e_m$ in order such that:

- no secrets are in the hero zone at the beginning;
- secret x does not exist right before an event which `add`s a secret x;
- secret x exists right before an event `test` x 1;
- secret x does not exist right before an event `test` x 0.

Given q events $e_1, e_2, \ldots, e_q$, you need to maintain an event sequence E. Initially, E is empty. For each $i = 1, 2, \ldots, q$ in order, try to append e_i to the end of E. If E is invalid, remove e_i and report a bug. Otherwise, find the number of secrets that must exist in the hero zone and the number of secrets that must not exist in the hero zone after performing the events of E in order.

Note that the number of *secrets that must (not) exist* is not just the *number of* (non-)existing secrets. For example, if $n = 2$, initially, secret 1 is missing and secret 2 is missing, so the answers would be 0 and 2. After a single `add`, secret 1 is unknown (can be or not be in hero zone) and secret 2 is unknown, so the answers are 0 and 0. After `test` 2 0, secret 2 is missing, so we know the added one was certainly secret 1, so secret 1 is present, and the answers are 1 and 1. See examples for better understanding.

我们记下每个 add 可能对应的集合，不难发现，时间越晚集合越大，构成一个杨图的结构。必选就等价于它上方的 add 数量等于其候选集合大小；不可选就意味着它在最新的 add 中都不可能。

用一个线段树维护候选集合大小减上方 add 数量，维护其最靠后的 = 0 的位置，就是最严格的必选的限制。

用一个数组维护每个牌的最大的禁止时间 $time$（也就是杨图这一列的高度）。

用一个树状数组维护时间上哪些位置有 add，用于快速求上方的 add 的数量；用一个并查集维护一个时间后最新的 add，能够帮助我们定位 add。

`add` 操作时：新增一行就好。如果当前所有牌都必选，那么就 bug。

`test x 1` 时：找到 $time_x$ 下方第一个 add，然后删这个 add，显然是影响最小的。

`test x 0` 时：若这个 x 是否是必选的，bug；否则，更新这一列的 $time$，并且更新线段树。

复杂度 $O(q \log n)$。

「ICPC World Finals 2021」蛛网漫游

蜘蛛 Charlotte 位于她的蜘蛛网的中心，蜘蛛网由一系列丝质直线组成，这些直线都是从中心延伸到网的外部边界。Charlotte 的网也有桥，每个桥都连接着相邻的两根线。桥的两个端点到蜘蛛网中心的距离总是一样的。

> 当 Charlotte 在网的中心吃完了深夜的大餐，想要退到某个角落时，她会自动走到边缘。要做到这一点，她会选择一根起始线并沿着它走，直到她遇到该线上的第一座桥。她会通过这座桥，走到另一条线上，然后一直向外走，直到她遇到另一座桥。然后她会通过那座桥，重复这个过程，直到当前的线上不再有桥，然后她会走到当前线的尽头。请注意，Charlotte 必须通过她所遇到的所有桥。图 I.1 展示了 Charlotte 可能经过的一条路径。
>
> Charlotte 白天最喜欢睡觉的角落是在线 s 的末端。对于每条可能的起始线，她想知道为了在 s 处结束，需要在原网上增加的桥的最少数量。Charlotte 可以在线上的任何一点增加一个桥，只要增加的桥不接触任何其他的桥。任何增加的桥的两个端点到蜘蛛网中心的距离必须相同，而且该桥必须连接相邻的两根线。$3 \leq n \leq 200\,000$。

我们从上到下 dp，相当于两两相差不能超过 1，一条必须的边就是交换两个 dp 值。数据结构维护一下就好。