Vision Program

Bạn đang cài đặt một chương trình thị giác cho một con robot. Mỗi lần camera của robot chụp một bức ảnh, bức ảnh sẽ được lưu dưới dạng ảnh đen trắng trong bộ nhớ của robot. Mỗi ảnh là một lưới điểm ảnh kích thước $H \times W$, với các hàng được đánh số từ 0 đến $H-1$ và các cột được đánh số từ 0 đến $W-1$. Có **chính xác hai** điểm ảnh đen trong mỗi bức ảnh và tất cả các điểm ảnh khác đều là màu trắng.

Robot có thể xử lý mỗi bức ảnh với một chương trình bao gồm các lệnh đơn giản. Bạn được cung cấp các giá trị H, W và một số nguyên dương K. Nhiệm vụ của bạn là viết một thủ tục để tạo ra một chương trình cho robot mà với bất kỳ bức ảnh nào robot sẽ xác định xem **khoảng cách** giữa hai điểm ảnh đen có chính xác là K hay không. Ở đây, khoảng cách giữa một điểm ảnh trên hàng r_1 và cột c_1 và một điểm ảnh trên hàng r_2 và cột c_2 là $|r_1 - r_2| + |c_1 - c_2|$. Trong công thức này $|x|$ biểu thị giá trị tuyệt đối của x, nó bằng x nếu $x \geq 0$ và bằng $-x$ nếu $x < 0$.

Bây giờ chúng ta mô tả cách thức hoạt động của robot.

Bộ nhớ của robot là một mảng đủ lớn các ô nhớ, được đánh số bắt đầu từ 0. Mỗi ô có thể lưu trữ 0 hoặc 1 và giá trị của nó một khi đã được thiết lập sẽ không thay đổi nữa. Bức ảnh được lưu trữ lần lượt theo từng hàng trong các ô nhớ được đánh số từ 0 đến $H \cdot W - 1$. Hàng đầu tiên được lưu trữ trong các ô từ 0 đến $W-1$ và hàng cuối cùng được lưu trữ trong các ô từ $(H-1) \cdot W$ đến $H \cdot W - 1$. Cụ thể, nếu điểm ảnh trên hàng i và cột j là màu đen, giá trị của ô $i \cdot W + j$ là 1, ngược lại thì giá trị là 0.

Chương trình của robot là một dãy các **lệnh**, được đánh số bởi các số nguyên liên tiếp bắt đầu từ 0. Khi chương trình chạy, các lệnh được thực hiện lần lượt từng lệnh một. Mỗi lệnh đọc các giá trị của một hoặc nhiều ô (chúng tôi gọi các giá trị này là các **dữ liệu đầu vào** của lệnh) và tạo ra một giá trị đơn bằng 0 hoặc 1 (chúng tôi gọi giá trị này là **dữ liệu đầu ra** của lệnh). Dữ liệu đầu ra của lệnh i được lưu trữ trong ô $H \cdot W + i$. Dữ liệu đầu vào của lệnh i chỉ có thể là các ô lưu trữ các điểm ảnh hoặc các dữ liệu đầu ra của các lệnh trước đó, tức là các ô từ 0 đến $H \cdot W + i - 1$.

Có bốn loại lệnh:

- `NOT`: có chính xác một dữ liệu đầu vào. Dữ liệu đầu ra của nó là 1 nếu đầu vào là 0, trái lại đầu ra của nó là 0.
- `AND`: có một hoặc nhiều dữ liệu đầu vào. Dữ liệu đầu ra là 1 khi và chỉ khi **tất cả** dữ liệu đầu vào là 1.
- `OR`: có một hoặc nhiều dữ liệu đầu vào. Dữ liệu đầu ra của nó là 1 khi và chỉ khi **tối thiểu một** trong các dữ liệu đầu vào là 1.

- `XOR`: có một hoặc nhiều dữ liệu đầu vào. Dữ liệu đầu ra là 1 khi và chỉ khi một **số lẻ** các dữ liệu đầu vào là 1.

Dữ liệu đầu ra của lệnh cuối cùng của chương trình phải là 1 nếu khoảng cách giữa hai điểm ảnh đen chính xác là K và là 0 nếu ngược lại.

Chi tiết cài đặt

Bạn cần cài đặt thủ tục sau:

```
void construct_network(int H, int W, int K)
```

- H, W: các chiều của mỗi bức ảnh được chụp bởi camera của robot
- K: một số nguyên dương
- Thủ tục này cần tạo ra một chương trình của robot. Đối với bất kỳ bức ảnh nào được chụp bởi camera của robot, chương trình này cần xác định xem khoảng cách giữa hai điểm đen trong ảnh có chính xác là K hay không.

Thủ tục này sẽ gọi một hoặc nhiều thủ tục sau để thêm các lệnh vào chương trình của robot (ban đầu chương trình là rỗng):

```
int add_not(int N)
int add_and(int[] Ns)
int add_or(int[] Ns)
int add_xor(int[] Ns)
```

- Thêm tương ứng một lệnh `NOT`, `AND`, `OR`, hoặc `XOR`.
- N (đối với `add_not`): chỉ số của ô nhớ mà lệnh `NOT` được thêm vào sẽ đọc.
- Ns (đối với `add_and`, `add_or`, `add_xor`): mảng chứa các chỉ số của các ô nhớ mà các lệnh `AND`, `OR`, hoặc `XOR` được thêm vào sẽ đọc.
- Mỗi thủ tục trả về chỉ số của ô lưu trữ dữ liệu đầu ra của lệnh. Các lời gọi liên tiếp đến các thủ tục này trả về các số nguyên liên tiếp bắt đầu từ $H \cdot W$.

Chương trình của robot có thể bao gồm tối đa 10 000 lệnh. Các lệnh có thể đọc tổng cộng tối đa 1 000 000 giá trị. Nói cách khác, tổng chiều dài của các mảng Ns trong tất cả các lời gọi tới `add_and`, `add_or` và `add_xor` cộng với số lượng các lời gọi tới `add_not` không được vượt quá 1 000 000.

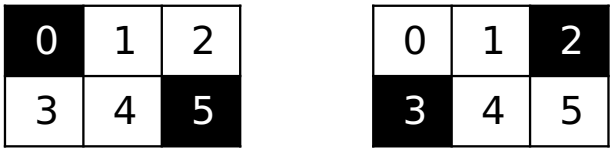
Sau khi thêm lệnh cuối cùng, thủ tục `construct_network` cần kết thúc. Chương trình của robot sau đó sẽ được đánh giá trên một số bức ảnh. Bài nộp của bạn sẽ vượt qua một trường hợp kiểm thử cho trước nếu với mỗi bức ảnh trong số những bức ảnh này, dữ liệu đầu ra của lệnh cuối cùng là 1 khi và chỉ khi khoảng cách giữa hai điểm ảnh đen trong ảnh bằng K.

Việc chấm điểm cho bài nộp của bạn có thể dẫn đến một trong các thông báo lỗi sau:

- `Instruction with no inputs`: một mảng rỗng được cung cấp như dữ liệu đầu vào cho `add_and`, `add_or`, hoặc `add_xor`.
- `Invalid index`: một chỉ số không chính xác (có thể là âm) được cung cấp như dữ liệu đầu vào cho `add_and`, `add_or`, `add_xor`, hoặc `add_not`.
- `Too many instructions`: thủ tục của bạn đã thêm nhiều hơn $10\,000$ lệnh.
- `Too many inputs`: các lệnh đọc tổng cộng nhiều hơn $1\,000\,000$ giá trị.

Ví dụ

Giả sử $H = 2$, $W = 3$, $K = 3$. Chỉ có hai bức ảnh có thể có khoảng cách giữa các điểm ảnh đen là 3.



- Trường hợp 1: các điểm ảnh đen là 0 và 5
- Trường hợp 2: các điểm ảnh đen là 2 và 3

Một lời giải khả thi là xây dựng một chương trình của robot bằng cách thực hiện các lời gọi sau:

1. `add_and([0, 5])`, thêm một lệnh mà dữ liệu đầu ra là 1 khi và chỉ khi trường hợp thứ nhất thỏa mãn. Dữ liệu đầu ra được lưu trữ trong ô 6.
2. `add_and([2, 3])`, thêm một lệnh mà dữ liệu đầu ra là 1 khi và chỉ khi trường hợp thứ hai thỏa mãn. Dữ liệu đầu ra được lưu trữ trong ô 7.
3. `add_or([6, 7])`, thêm một lệnh mà dữ liệu đầu ra là 1 khi và chỉ khi một trong các trường hợp trên thỏa mãn.

Các ràng buộc

- $1 \le H \le 200$
- $1 \le W \le 200$
- $2 \le H \cdot W$
- $1 \le K \le H + W - 2$

Subtasks

1. (10 điểm) $\max(H, W) \le 3$
2. (11 điểm) $\max(H, W) \le 10$
3. (11 điểm) $\max(H, W) \le 30$
4. (15 điểm) $\max(H, W) \le 100$
5. (12 điểm) $\min(H, W) = 1$

6. (8 điểm) Điểm ảnh tại hàng 0 và cột 0 là màu đen trên mỗi bức ảnh.
7. (14 điểm) $K = 1$
8. (19 điểm) Không có thêm ràng buộc nào.

Trình chấm mẫu

Trình chấm mẫu đọc dữ liệu đầu vào theo định dạng sau:

- dòng 1: H W K
- dòng $2 + i$ ($i \geq 0$): $r_1[i]$ $c_1[i]$ $r_2[i]$ $c_2[i]$
- dòng cuối: -1

Mỗi dòng ngoại trừ dòng đầu tiên và dòng cuối cùng biểu diễn một bức ảnh có hai điểm ảnh đen. Bức ảnh i được mô tả trên dòng $2 + i$. Một điểm ảnh đen trên dòng $r_1[i]$ và cột $c_1[i]$ và điểm ảnh đen còn lại trên dòng $r_2[i]$ và cột $c_2[i]$.

Trình chấm mẫu đầu tiên gọi `construct_network (H, W, K)`. Nếu `construct_network` vi phạm ràng buộc nào đó được nói trong phần mô tả bài toán, trình chấm mẫu sẽ in một trong các thông báo lỗi được liệt kê ở cuối phần Chi tiết cài đặt và thoát.

Nếu không, trình chấm mẫu sẽ sinh ra hai dữ liệu đầu ra.

Đầu tiên, trình chấm mẫu in ra dữ liệu đầu ra của chương trình của robot theo định dạng sau:

- dòng $1 + i$ ($0 \leq i$): dữ liệu đầu ra của lệnh cuối cùng trong chương trình của robot đối với bức ảnh i (1 hoặc 0).

Thứ hai, trình chấm mẫu ghi ra một tệp `log.txt` trong thư mục hiện tại theo định dạng sau:

- dòng $1 + i$ ($0 \leq i$): $m[i][0]$ $m[i][1]$ $\ldots$ $m[i][c-1]$

Dãy các giá trị trên dòng $1 + i$ mô tả các giá trị được lưu trữ trong các ô nhớ của robot sau khi chương trình của robot chạy, với bức ảnh i là dữ liệu đầu vào. Cụ thể, $m[i][j]$ đưa ra giá trị của ô j. Lưu ý rằng giá trị của c (độ dài của dãy các giá trị) bằng $H \cdot W$ cộng với số lượng lệnh trong chương trình của robot.