Vision Program

Estás implementando un nuevo programa de visión para un robot. Cada vez que la cámara del robot toma una foto, es guardada como una imagen en blanco y negro dentro de la memoria del robot. Cada imagen es una cuadrícula de pixeles de tamaño $H \times W$, con renglones numerados de 0 a $H-1$ y columnas numeradas de 0 a $W-1$. Hay **exactamente dos** pixeles negros en cada imagen, todos los demas pixeles son blancos.

El robot puede procesar cada imagen con un programa que consiste en instrucciones simples. Se te da los valores de H, W y un entero positivo K. Tu objetivo es escribir un procedimiento que crea un programa que ejecutará el robot. Este programa determina para cualquier imagen, si la **distancia** entre los dos pixeles negros es exactamente K. Definimos la distancia entre un pixel en el renglón r_1 y la columna c_1 y el pixel en el renglón r_2 y columna c_2 como $|r_1 - r_2| + |c_1 - c_2|$. En esta fórmula, $|x|$ denota el valor absoluto de x, que es igual a x si $x \geq 0$ e igual a $-x$ si $x < 0$.

Ahora describimos cómo el robot funciona:

La memoria del robot es un arreglo de celdas lo suficientemente largo, indexado a partir de 0. Cada celda puede guardar valor 0 o 1. Una vez asignado el valor de una celda, no puede ser cambiado. La imagen es guardada renglón por renglón en las celdas numeradas de 0 a $H \cdot W - 1$. El primer renglón es almacenado de la celda 0 a la celda $W-1$ y el último renglón es almacenado de la celda $(H-1) \cdot W$ a la celda $H \cdot W - 1$. En particular, si el pixel en el renglón i y columna j es negro, el valor de la celda $i \cdot W + j$ es 1,de lo contrario es 0.

El programa del robot es una secuencia de **instrucciones** que están numeradas con enteros consecutivos iniciando en 0. Cuando un programa es ejecutado, las instrucciones son ejecutadas una por una. Cada intrucción lee los valores de una o más celdas (llamamos a estos valores como las **entradas** de la instrucción) y produce un valor igual a 0 o 1 (llamamos a este valor como la **salida** de la instrucción). La salida de la instrucción i es guardada en la celda $H \cdot W + i$. Las entradas de la instrucción i sólo pueden ser celdas que guardan pixeles de la imagen o celdas que son salidas de instricciones anteriores, es decir las celdas 0 a $H \cdot W + i - 1$.

Existen cuatro tipo de instrucciones:

- `NOT`: tiene exactamente una entrada. Su salida es 1 si la entrada es 0, de lo contrario la salida es 0.
- `AND`: tiene una o más entradas. Su salida es 1 si y solo si **todas** las entradas son 1.

- `OR`: tiene una o más entradas. Su salida es 1 si y solo si **al menos una** de las entradas es 1.
- `XOR`: tiene una o más entradas. Su salida es 1 si y solo si es **número impar** la cantidad de 1s en la entrada.

La salida de la última instrucción del programa debe ser 1 si la distancia entre los dos pixeles negros es exactamente K, de lo contrario la salida debe ser 0.

Detalles de implementación

Debes implementar el siguiente procedimiento:

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

- H, W: dimenciones de cada foto tomada por la cámara del robot
- K: un entero positivo
- Este procedimiento debe crear un programa del robot. Para cualquier imagen tomada por la cámara del robot, el programa debe determinar si la distancia entre los dos pixeles negros es exactamente K.

Este procedimiento debe llamar uno o más de los siguientes procedimiento para añadir nuevas instrucciones al programa del robot (que inicialmente está vacío):

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

- Añade una instrucción `NOT`, `AND`, `OR`, o `XOR` , respectivamente.
- N (para `add_not`): el índice de la celda que la nueva instrucción `NOT` lee como entrada
- Ns (para `add_and`, `add_or`, `add_xor`): arreglo que contiene los índices de las celdas que la nueva instrucción añadida `AND`, `OR`, o `XOR` lee como su entrada.
- Cada procedimiento regresa el índice de la celda donde guarda la salida de la instrucción. La ejecuciones consecutivas de estos procedimientos regresan enteros consecutivos iniciando en $H \cdot W$.

El programa del robot puede tener a lo más $10\,000$ instrucciones. Las instrucciones pueden leer a lo más $1\,000\,000$ valores en total. Es decir, la longitud total de los arreglos Ns de todas las llamadas a `add_and`, `add_or` y `add_xor` más el número de llamadas a `add_not` no puede exceder $1\,000\,000$.

Después de agregar la última instrucción, el procedimiento `construct_network` debe regresar. El programa del robot será evaluado con varias imágenes. Tu solución para el caso de ejemplo es correcta si para cada una de las imágenes, la salida de la

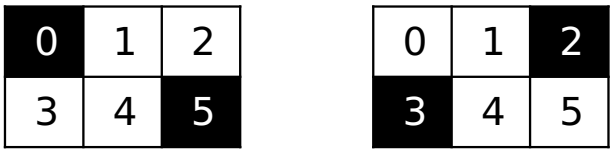
instrucción es 1 si y solo si la distancia entre los dos pixeles negros es igual a K.

El veredicto de tu solución puede ser alguno de los siguientes mensajes en inglés:

- `Instruction with no inputs`: un arreglo vacío fue dado como entrada a `add_and`, `add_or`, o `add_xor`.
- `Invalid index`: un índice incorrecto de una celda (posiblemente negativo) fue dado como parte de la entrada a `add_and`, `add_or`, `add_xor`, o `add_not`.
- `Too many instructions`: tu procedimiento intentó añadir más de $10\,000$ instrucciones.
- `Too many inputs`: las instrucciones leen más de $1\,000\,000$ valores en total.

Ejemplo

Asume $H = 2$, $W = 3$, $K = 3$. Sólo hay dos posibles imagenes donde la distancia entre los pixeles negros es 3.



- Caso 1: los pixeles negros son 0 y 5
- Caso 2: los pixeles negros son 2 y 3

Una posible solución es hacer un programa para el robot con las siguientes llamadas:

1. `add_and([0, 5])`, que agrega una intrucción que da como salida 1 si y solo sí el primer caso se mantiene. La salida es almacenada en la celda 6.
2. `add_and([2, 3])`, que agrega una intrucción que da como salida 1 si y solo si el segundo caso se cumple. La salida es almacenada en la casilla 7.
3. `add_or([6, 7])`, que agrega una intrucción que da como salida 1 si y solo si alguno de los casos de arriba se mantiene.

Restricciones

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

Subtareas

1. (10 puntos) $\max(H, W) \leq 3$

2. (11 puntos) $\max(H, W) \leq 10$
3. (11 puntos) $\max(H, W) \leq 30$
4. (15 puntos) $\max(H, W) \leq 100$
5. (12 puntos) $\min(H, W) = 1$
6. (8 puntos) Pixel en renglón 0 y columna 0 es negro en cada imagen.
7. (14 puntos) $K = 1$
8. (19 puntos) Sin restricciones adicionales.

Grader de ejemplo

El grader de ejemplo lee la entrada en el siguiente formato:

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

Cada línea, exceptuando la primera y la última, representan una imagen con dos píxeles negros. Denotamos la imagen descrita en la línea $2 + i$ por la imagen i. Un pixel negro se encuentra en la fila $r_1[i]$ y la columna $c_1[i]$ y la otra en la fila $r_2[i]$ y columna $c_2[i]$.

El grader de ejemplo primero llama `construct_network(H, W, K)`. Si `construct_network` viola alguna restriccion descrita en el problema, el grader de ejemplo imprime uno de los mensajes de error listados al final de la sección de detalles de implementación y termina.

De lo contrario, el grader de ejemplo produce dos salidas:

Primero, el grader de ejemplo imprime la salida del programa del robot en el siguiente formato:

- línea $1 + i$ $(0 \leq i)$: salida de la última instrucción en el programa del robot para la imagen i (1 o 0).

Segundo, el grader de ejemplo escribe un archivo `log.txt` en el directorio actual en el siguiente formato:

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

La secuencia en la línea $1 + i$ describe los valores almacenados en las celdas de memoria del robot luego de que el programa del robot es ejecutado, dado i como entrada. Específicamente, $m[i][j]$ da el valor de la celda j.

Fíjate que el valor de c (la longitud de la secuencia) es igual a $H \cdot W$ más el número de instrucciones en el programa del robot.