Programa de Visión

Estás implementando un programa de visión para un robot. Cada vez que la cámara del robot toma una fotografía, se almacena en la memoria del robot como una imagen blanco y negro. Cada imagen es una cuadrícula de $H \times W$ pixeles, con filas numeradas desde 0 hasta $H - 1$ y columnas numeradas desde 0 hasta $W - 1$. Cada imagen posee **exactamente dos** pixeles negros. El resto de pixeles son blancos.

El robot puede procesar cada imagen con un programa que consiste de instrucciones simples. Sabes los valores de H, W, y un entero positivo K. Tu objetivo es escribir un procedimiento para producir un programa para el robot tal que, dada cualquier imagen, determine si la **distancia** entre los dos pixeles negros es exactamente K. Para este problema, la distancia entre un pixel en la fila r_1 y columna c_1 y un pixel en la fila r_2 y columna c_2 es $|r_1 - r_2| + |c_1 - c_2|$. En esta fórmula, $|x|$ denota el valor absoluto de x, igual a x si $x \geq 0$ e igual a $-x$ si $x < 0$.

Ahora se describe cómo funciona el robot.

La memoria del robot es un arreglo suficientemente grande de celdas numeradas desde 0. Cada celda puede almacenar ya sea 0 o 1 y su valor, una vez establecido, no puede ser cambiado. La imagen es almacenada fila por fila en celdas numeradas desde 0 hasta $H \cdot W - 1$. La primera fila es almacenada en las celdas 0 a la $W - 1$ y la última fila es almacenada en las celdas $(H - 1) \cdot W$ a la $H \cdot W - 1$. En particular, si el pixel en la fila i y columna j es negro, el valor de la celda $i \cdot W + j$ es 1; de lo contrario, su valor es 0.

Un programa del robot es una secuencia de **instrucciones**, que se numeran con enteros consecutivos iniciando de 0. Cuando cada programa es ejecutado, las instrucciones se ejecutarán una a una. Cada instrucción lee los valores de una o más celdas (estos valores son llamados las **entradas** de la instrucción) y produce un solo valor igual a 0 ó 1 (este valor es llamado la **salida** de la instrucción). La salida de la instrucción i se almacena en la celda $H \cdot W + i$. Las entradas de la instrucción i pueden ser únicamente celdas que almacenen ya sea pixeles o salidas de instrucciones anteriores (es decir, las celdas entre 0 y $H \cdot W + i - 1$).

Existen cuatro tipos de instrucciones:

- `NOT`: posee exactamente una entrada. Su salida es 1 si la entrada es 0, y es 0 si la entrada es 1.
- `AND`: posee una o más entradas. Su salida es 1 si y solo si **todas** las entradas son 1.
- `OR`: posee una o más entradas. Su salida es 1 si y solo si **al menos una** de las

entradas es 1.
- `XOR`: posee una o más entradas. Su salida es 1 si y solo si un **número impar** de las entradas es 1.

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

Detalles de implementación

Debes implementar la siguiente función:

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

- H, W: las dimensiones de cada imagen tomada por la cámara del robot.
- K: un entero positivo.
- Esta función debe producir un programa del robot. Para cualquier imagen tomada por la cámara del robot, este programa debe determinar si la distancia entre los dos pixeles negros en la imagen es exactamente K.

Esta función debe llamar a uno o más de los siguientes procedimientos para agregar instrucciones al programa del robot (que inicialmente se encuentra vacío):

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

- Agregar una instrucción `NOT`, `AND`, `OR` o `XOR`, respectivamente.
- N (para `add_not`): El índice de la celda desde la cual la instrucción `NOT` a agregar lee su entrada.
- Ns (para `add_and`, `add_or`, `add_xor`): arreglo que contiene los índices de las celdas desde las cuales la instrucción `AND`, `OR` o `XOR` a agregar lee sus entradas.
- Cada procedimiento retorna el índice de la celda que almacena la salida de la instrucción. Llamadas consecutivas a estos procedimientos retornan enteros consecutivos iniciando con $H \cdot W$.

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. En otras palabras, el largo total de los arreglos Ns en 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 retornar. El programa del robot será entonces evaluado con algunas imágenes. Tu solución pasará un caso de prueba dado si para cada una de estas imágenes, la salida de la última instrucción es 1 si y solo si la distancia entre los dos pixeles negros en la

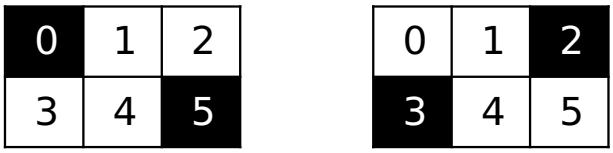
imagen es igual a K.

La calificación de tu solución puede resultar en uno de los siguientes mensajes de error:

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

Ejemplo

Asume que $H = 2$, $W = 3$, $K = 3$. Existen únicamente dos posibles imágenes donde la distancia entre los pixeles negros es 3.



- Caso 1: los pixeles blancos están en 0 y 5
- Caso 2: los pixeles negros están en 2 y 3

Una solución posible es construir el programa del robot realizando las siguientes llamadas:

1. `add_and([0, 5])`, que agrega una instrucción que tiene como salida 1 si y solo si el primer caso es verdadero. La salida es almacenada en la celda 6.
2. `add_and([2, 3])`, que agrega una instrucción que tiene como salida 1 si y solo si el segundo caso es verdadero. La salida es almacenada en la celda 7.
3. `add_or([6, 7])`, que agrega una instrucción que tiene como salida 1 si y solo si uno de los casos de arriba es verdadero.

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) El pixel en la fila 0 y columna 0 es negro en cada imagen.
7. (14 puntos) $K = 1$
8. (19 puntos) Sin restricciones adicionales.

Evaluador de ejemplo

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

- 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, a excepción de la primera y la última, representa una imagen con dos pixeles negros. Denotamos a la imagen descrita en la línea $2 + i$ como la imagen i. Un pixel negro está en la fila $r_1[i]$ y columna $c_1[i]$ y el otro se encuentra en la fila $r_2[i]$ y columna $c_2[i]$.

El evaluador de ejemplo llama primero a `construct_network(H, W, K)`. Si `construct_network` no cumple con alguna de las restricciones descritas en el enunciado, el evaluador de ejemplo imprime uno de los mensajes de error listados al final de la sección Detalles de Implementación y termina.

Si no se detecta un error, el evaluador de ejemplo produce dos salidas.

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

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

Segundo, el calificador de ejemplo escribe a 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 $1 + j$ describe los valores almacenados en las celdas de la memoria del robot después que el programa del robot es ejecutado, dada la imagen i como entrada. Específicamente, $m[i][j]$ devuelve el valor de la celda j. Note que los valores de c (el largo de la secuencia) es igual a $H \cdot W$ más el número de instrucciones en el programa del robot.