Programa de Visión

Tú estás implementando un programa de visión para un robot. Cada vez que la cámara del robot toma una foto, esta es almacenada como una imagen en blanco y negro en la memoria del robot. Cada imagen es una malla de $H \times W$ pixeles, con filas numeradas de 0 a $H-1$ y columnas numeradas de 0 a $W-1$. Hay **exactamente dos** pixeles negros en cada imagen, y todos los demás pixeles son blancos.

El robot puede procesar cada imagen con un programa que consiste de instrucciones simples. Se te da los valores de H, W, y un entero positivo K. Tu tarea es escribir un procedimiento para producir un programa para el robot que, para cada imagen, determine si la **distancia** entre los dos pixeles negros es exactamente K. Aquí, 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, el cual es x si $x \geq 0$ y es $-x$ si $x < 0$.

Ahora describiremos como funciona el robot.

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

El programa de un robot es una secuencia de **instrucciones**, la cual está numerada con enteros consecutivos empezando desde 0. Cuando el programa está corriendo, las instrucciones son ejecutadas una por una. Cada instrucción lee los valores de una o más celdas (nosotros llamamos a estos valores las instrucciones de **entrada**) y producen un valor simple igual a 0 o 1 (nosotros llamamos a este valor instrucciones de **salida**). La salida de la instrucción i es almacenada en la celda $H \cdot W + i$. Las entradas de la instrucción i pueden solamente ser celdas que almacenan ya sean pixeles o salidas de instrucciones previas, por ejemplo celdas 0 a $H \cdot W + i - 1$.

Hay cuatro tipo de instrucciones:

- `NOT`: tiene exactamente una entrada. Su salida es 1 si su entrada es 0, de otra manera es 0.
- `AND`: tiene una o más entradas. Su salida es 1 sí y sólo sí **todas** sus entradas son 1.
- `OR`: tiene una o más entradas. Su salida es 1 sí y sólo sí **al menos una** de sus

entradas es 1.
- `XOR`: tiene una o más entradas Su salida es 1 sí y sólo sí un **número impar** de entradas son 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 de otra manera.

Detalles de implementación

Debes implementar el siguiente procedimiento:

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

- H, W: dimensiones de cada imagen tomadas por la cámara del robot
- K: un entero positivo
- Este procedimiento debe producir un programa para el robot. Por cada 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.

Este procedimiento debe llamar una o más veces a los siguientes procedimientos para añadir instrucciones al programa del robot (es cual inicialmente es 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 de donde la instrucción añadida `NOT` lee su entrada
- Ns (para `add_and`, `add_or`, `add_xor`): arreglo que contiene los índices de de las celdas de donde las instrucciones añadidas `AND`, `OR`, or `XOR` leen sus entradas
- Cada procedimiento retorna el índice de la celda que almacena la salida de la instrucción. Las llamadas consecutivas a esos procedimientos retornan enteros consecutivos empezando de $H \cdot W$.

El programa del robot puede consistir de a lo mucho $10\,000$ instrucciones. Las instrucciones pueden leer a lo mucho $1\,000\,000$ valores en total. En otras palabras, el tamaño total de Ns arreglos en todas las llamadas a `add_and`, `add_or` y `add_xor` mas el número de llamadas a `add_not` no puede exceder $1\,000\,000$.

Después de añadir la última instrucción, el procedimiento `construct_network` debe ser retornado. El programa del robot será luego evaluado con algún número de imágenes. Tu solución pasa un caso de prueba dado si para cada una de esas imágenes, la salida de la última instrucción es 1 sí y sólo sí la distancia entre los dos

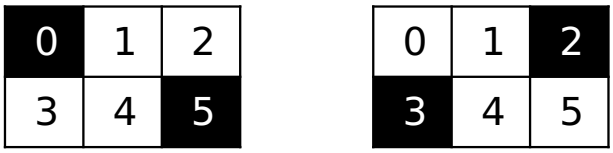
puntos negros en la imagen es igual a K.

La calificación de tu solución podría resultar en uno los siguientes mensajes de error.

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

Ejemplo

Asuma $H = 2$, $W = 3$, $K = 3$. Hay solamente dos posibles imágenes 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 solución posible es construir el programa del robot haciendo las siguientes llamadas:

1. `add_and([0, 5])`, lo cual añade una instrucción que devuelve 1 sí y sólo sí el primer caso es válido. La salida es almacenada en la celda 6.
2. `add_and([2, 3])`, lo cual añade una instrucción que devuelve 1 sí y sólo sí el segundo caso es válido.La salida es almacenada en la celda 7.
3. `add_or([6, 7])`, lo cual añade una instrucción que devuelve 1 sí y sólo sí uno 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 points) $\max(H, W) \leq 3$

2. (11 points) $\max(H, W) \leq 10$
3. (11 points) $\max(H, W) \leq 30$
4. (15 points) $\max(H, W) \leq 100$
5. (12 points) $\min(H, W) = 1$
6. (8 points) Pixel en la fila 0 y columna 0 es negro en cada imagen.
7. (14 points) $K = 1$
8. (19 points) 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 línea representan una imagen con dos pixeles negros. Denotamos la imagen descrita en la línea $2 + i$ como imagen i. Un pixel negro está en la fila $r_1[i]$ y columna $c_1[i]$ y el otro en la fila $r_2[i]$ y columna $c_2[i]$.

El grader de ejemplo podría imprimir `Invalid user input` si este detecta cualquier error en la entrada (por ejemplo, la entrada referencia una fila o columna que no existe).

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

De otra manera, el grader de ejemplo produce dos salidas.

Primero, el grader de 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 con 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 la memoria del robot después que el programa del robot corra, dada la imagen i como entrada. Especificamente, $m[i][j]$ devuelve el valor de la celda j. Note que el valor de c (el tamaño de la secuencia) es igual a $H \cdot W$ mas el número de instrucciones en el programa del robot.