Vision Program

Estás a implementar um programa de visão para um robot. De cada vez que a câmara do robot tira uma foto, esta é armazenada como uma imagem a preto e branco na memória do robot. Cada imagem é uma matriz de $H \times W$ pixeis, com as linhas numeradas de 0 a $H-1$ e as colunas numeradas de 0 a $W-1$. Existem **exatamente dois** pixeis pretos em cada imagem, e todos os outros pixeis são brancos.

O robot pode processar cada imagem com um programa que consiste em instruções simples. São-te dados os valores de H, W, e um inteiro positivo K. O teu objetivo é escrever um procedimento para produzir um programa para o robot que, para qualquer imagem, determina se a **distância** entre os dois pixeis pretos é exatamente K. Aqui, a distância entre um pixel na linha r_1 e a coluna c_1 e um pixel na linha r_2 e a coluna c_2 é $|r_1 - r_2| + |c_1 - c_2|$. Nesta fórmula, $|x|$ indica o valor absoluto de x, que é igual a x se $x \geq 0$ ou $-x$ se $x < 0$.

Vamos agora descrever o funcionamento do robot.

A memória do robot é um array de células suficientemente grande, indexado a partir de 0. Cada célula pode armazenar um valor 0 ou 1 e o seu valor, uma vez definido, nunca será modificado. A imagem é armazenada linha por linha nas células indexadas de 0 até $H \cdot W - 1$. A primeira linha é armazenada nas células 0 até $W-1$, e a última linha é armazenada nas células $(H-1) \cdot W$ até $H \cdot W - 1$. Em particular, se o pixel na linha i e na coluna j é preto, o valor da célula $i \cdot W + j$ é 1, caso contrário é 0.

O programa do robot é uma sequência de **instruções**, que são numeradas com inteiros consecutivos começando a partir do 0. Quando o programa é executado, as instruções são executadas uma por uma. Cada instrução lê os valores de uma ou mais células (chamamos a estes valores os **inputs** da instrução) e produz um único valor igual a 0 ou 1 (chamamos a este valor o **output** da instrução). O output da instrução i é armazenado na célula $H \cdot W + i$. Os inputs da instrução i só podem ser células que armazenam pixeis ou outputs de instruções anteriores, isto é, células de 0 até $H \cdot W + i - 1$.

Existem quatro tipos de instruções:

- `NOT`: tem exatamente um input. O seu output é 1 se o input for 0, ou 0 caso contrário.
- `AND`: tem um ou mais inputs. O seu output é 1 se e só se **todos** os inputs forem 1.
- `OR`: tem um ou mais inputs. O seu output é 1 se e só se **pelo menos um** dos inputs for 1.

- `XOR`: tem um ou mais inputs. O seu output é 1 se e só se **um número impar** dos inputs for 1.

O output da última instrução do programa deve ser 1 se a distância entre os pixeis pretos for exatamente K. Caso contrário, deve ser 0.

Detalhes de implementação

Deves implementar o seguinte procedimento:

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

- H, W: dimensões de cada imagem obtida através da câmara do robot.
- K: um inteiro positivo.
- Este procedimento deve produzir um programa para o robot. Para qualquer imagem obtida pela câmara do robot, este programa deve determinar se a distância entre os dois pixeis pretos da imagem é exactamente K.

Este procedimento deve chamar uma ou mais vezes as seguintes funções para adicionar uma instrução no final do programa do robot (que inicialmente está vazio):

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

- Adicionar a instrução `NOT`, `AND`, `OR`, ou `XOR`, respetivamente.
- N (para `add_not`): o índice da célula na qual a instrução `NOT` adicionada deve ler o seu input
- Ns (para `add_and`, `add_or`, `add_xor`): um array contendo os índices das células nais quais as instruções `AND`, `OR`, or `XOR` devem ler os seus inputs.
- Cada uma destas funções retorna o índice da célula que armazena o output dessa instrução. As chamadas consecutivas a estas funções devem devolver inteiros consecutivos começando a partir de $H \cdot W$.

O programa do robot pode consistir num máximo de 10 000 instruções. As instruções podem ler no máximo 1 000 000 valores no total. Por outras palavras, o tamanho total dos arrays Ns nas chamadas a `add_and`, `add_or` e `add_xor`, mais o número de chamadas a `add_not`, não pode exceder 1 000 000.

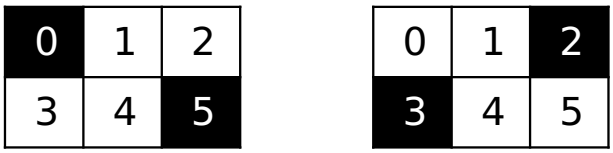
Depois de adicionar a última instrução ao final do programa, o procedimento `construct_network` deve terminar. O programa do robot será então avaliado numa certa quantidade de imagens. A tua solução passa um dado caso de teste se para cada uma destas imagens o output da última instrução for 1 se e só se a distância entre os dois pixeis pretos for igual a K.

A avaliação da tua solução pode resultar numa das seguintes mensagens de erro:

- `Instruction with no inputs`: um array vazio foi dado como input para `add_and`, `add_or`, ou `add_xor`.
- `Invalid index`: um incorreto (possivelmente negativo) índice de célula foi dado como input para `add_and`, `add_or`, `add_xor`, ou `add_not`.
- `Too many instructions`: o teu procedimento tentou adicionar mais do que $10\,000$ instrucões.
- `Too many inputs`: as instruções leram mais do que $1\,000\,000$ de valores no total.

Exemplo

Assume que $H = 2$, $W = 3$, $K = 3$. Existem duas possíveis imagens onde a distância entre os pixeis pretos é 3:



- Caso 1: os pixeis pretos são 0 e 5
- Caso 2: os pixeis pretos são 2 e 3

Uma possível solução é construir um programa de robot fazendo as seguintes chamadas:

1. `add_and([0, 5])`, que adiciona uma instrução que produz um output de 1 se e só se o primeiro caso se aplicar. O output é armazenado na célula 6.
2. `add_and([2, 3])`, que adiciona uma instrução que produz um output de 1 se e só se o segundo caso se aplicar. O output é armazenado na célula 7.
3. `add_or([6, 7])`, que adiciona uma instrução que produz um output de 1 se e só se um dos casos anteriores se aplicar.

Restrições

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

Subtarefas

1. (10 pontos) $\max(H, W) \leq 3$
2. (11 pontos) $\max(H, W) \leq 10$

3. (11 pontos) $\max(H, W) \leq 30$
4. (15 pontos) $\max(H, W) \leq 100$
5. (12 pontos) $\min(H, W) = 1$
6. (8 pontos) O pixel na linha 0 e na coluna 0 é preto em cada imagem.
7. (14 pontos) $K = 1$
8. (19 pontos) Nenhuma restrição adicional.

Avaliador exemplo

O avaliador exemplo lê o input no seguinte formato:

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

Cada linha excepto a primeira e a última representa uma imagem com dois pixeis pretos. Indicamos a imagem descrita na linha $2 + i$ por imagem i. Um pixel preto está na linha $r_1[i]$ e coluna $c_1[i]$ e o outro pixel preto está na linha $r_2[i]$ e na coluna $c_2[i]$.

O avaliador exemplo primeiro chama `construct_network(H, W, K)`. Se `construct_network` violar alguma das restrições descritas no enunciado, o avaliador exemplo imprime uma das mensanges de erro listas no final da secção de detalhes de implementação e termina a sua execução.

Caso contrário, o avaliador exemplo produz dois outputs.

Primeiro, o avaliador exemplo imprime o output do programa do robot no seguinte formato:

- linha $1 + i$ $(0 \leq i)$: o output da última instrução do programa do robot para a imagem i (1 or 0).

Depois, o avaliador exemplo escreve um ficheiro `log.txt` na pasta corrente no seguinte formato:

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

A sequência na linha $1 + i$ descreve os valores armazenados nas células de memória do robot depois do programa ser executado, dando a imagem i como input. Especificamente, $m[i][j]$ dá o valor da célula j. Nota que o valor de c (o tamanho da sequência) é igual a $H \cdot W$ mais o número de instruções no programa do robot.