Vision Program

Você está implementando um programa de visão para um robô. Cada vez que a câmera do robô tira uma foto, ela é armazenada na memória do robô como uma imagem em branco e preto. Cada imagem é um reticulado de $H \cdot W$ pixels, com linhas numeradas de 0 até $H-1$ e colunas numeradas de 0 até $W-1$. Há **exatamente dois** pixels pretos em cada imagem, e todos os outros pixels são brancos.

O robô pode processar cada imagem com um programa que consiste de instruções simples. Serão dados os valores de H, W e um número positivo K. Sua tarefa é escrever um procedimento para produzir um programa para o robô que, para qualquer imagem, determine se a **distância** entre os dois pixels pretos é exatamente K. A distância entre um pixel na linha r_1 e coluna c_1 e um pixel na linha r_2 e coluna c_2 é $|r_1 - r_2| + |c_1 - c_2|$. Nesta fórmula, $|x|$ denota o valor absoluto de x, que é igual a x se $x \geq 0$ e igual a $-x$ se $x < 0$.

Vamos descrever agora como o robô funciona.

A memória do robô é um *array* suficientemente grande de células, indexadas a partir de 0. Cada célula pode armazenar ou 0 ou 1, e seu valor, uma vez escrito, não é alterado. A imagem é armazenada linha por linha em 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 coluna j é preto, o valor da célula $i \cdot W + j$ é 1, caso contrário é 0.

Um programa para o robô é uma sequência de **instruções**, que são numeradas por números inteiros consecutivos a partir de 0. Quando um programa é executado, suas instruções são executadas uma por uma. Cada instrução lê os valores de uma ou mais células (estes valores são denominados **entrada** da instrução) e produz um único valor igual a 0 ou 1 (este valor é denominado **saída** da instrução). A saída da instrução i é armazenada na célula $H \cdot W + i$. A entrada da instrução i podem ser somente células que armazenam pixels ou que armazenam saídas de instruções anteriores, isto é células 0 até $H \cdot W + i - 1$.

Há quatro tipos de instruções:

- `NOT`: tem exatamente uma entrada. Sua saída é 1 se a entrada é 0, caso contrário sua saída é 0.
- `AND`: tem uma ou mais entradas. Sua saída é 1 se e somente se **todas** as entradas são 1.
- `OR`: tem uma ou mais entradas. Sua saída é 1 se e somente se **pelo menos uma**

das entradas é 1.
- `XOR`: tem uma ou mais entradas. Sua saída é 1 se e somente se **um número ímpar** de entradas são 1.

A saída da última instrução do programa deve ser 1 se a distância entre os dois pixels preto é exatamente K, e 0 caso contrário.

Detalhes de implementação

Você deve implementar o seguinte procedimento:

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

- H, W: dimensões de cada imagem tirada pela câmera do robô
- K: um número positivo
- Este procedimento deve produzir um programa para o robô. Para qualquer imagem tirada pela câmera do robô, o programa gerado deve determinar se a distância entre os dois pixels pretos na imagem é exatamente K.

O procedimento deve chamar um ou mais dos seguintes procedimentos para acrescentar instruções ao programa para o robô (que inicialmente está vazio):

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

- Acrescenta uma instrução `NOT`, `AND`, `OR`, or `XOR` , respectivamente.
- N (para `add_not`): o índice da célula a ser lida pela instrução `NOT` acrescentada
- Ns (para `add_and`, `add_or`, `add_xor`): *array* contendo os índices das células que serão lidas pelas instruções `AND`, `OR`, or `XOR`
- Cada procedimento retorna o índice da célula que armazena a saída da instrução. As chamadas consecutivas a esses procedimentos retornam inteiros a partir de $H \cdot W$.

O programa para o robô pode conter no máximo $10\,000$ instruções. As instruções podem ler no máximo um total de $1\,000\,000$ valores. Em outras palavras, a soma dos comprimentos dos *arrays* Ns em todas as chamadas `add_and`, `add_or` e `add_xor` mais o número de chamadas para `add_not` não pode exceder $1\,000\,000$.

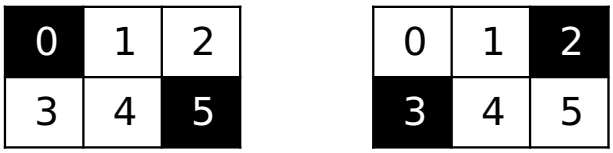
Após acrescentar a última instrução, o procedimento `construct_network` deve retornar. O programa do robô será então avaliado usando um número de imagens. Sua solução *passa* um dado caso de teste se para cada imagem do caso de teste, a saída da última instrução é 1 se e somente se a distância entre os dois pixels pretos na imagem é igual a K.

A correção de sua solução pode resultar em uma das seguintes mensagens de erro (em inglês):

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

Exemplo

Considere $H = 2$, $W = 3$, $K = 3$. Há apenas duas imagens possíves em que a distância entre os pixels pretos é 3.



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

Uma solução possível é construir um programa para o robô efetuando as seguintes chamadas:

1. `add_and([0, 5])`, que acrescenta uma instrução que produz saída 1 se e somente se o caso 1 é verdadeiro. A saída é armazenada na célula 6.
2. `add_and([2, 3])`, que acrescenta uma instrução que produz saída 1 se e somente se o caso 2 é verdadeiro. A saída é armazenada na célula 7.
3. `add_or([6, 7])`, que acrescenta uma instrução que produz saída 1 se e somente se algum dos dois casos é verdadeiro.

Restrições

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

Subtarefas

1. (10 pontos) $\max(H, W) \le 3$

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) Para cada imagem, o pixel na linha 0 e coluna 0 é preto.
7. (14 pontos) $K = 1$
8. (19 pontos) Sem restrições adicionais.

Corretor exemplo

O corretor exemplo lê a entrada no seguinte formato:

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

Cada linha exceto a primeira e a última representa uma imagem com dois pixels pretos. Denotamos 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 coluna $c_2[i]$.

O corretor exemplo primeiro executa uma chamada `construct_network(H, W, K)`. Se `construct_network` viola alguma restrição descrita no enunciado do problem, o corretor exemplo imprime uma das mensagens de erro listadas ao final da seção "Detalhes de Implementação" e termina sua execução.

Caso contrário, o corretor exemplo produz duas saídas.

Primeiro, o corretor de exemplo imprime a saída do programa para o robô no seguinte formato:

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

Segundo, o corretor exemplo escreve um arquivo `log.txt` no diretório corrente com o seguinte formato:

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

A sequência na linha $1 + i$ descreve os valores armazenados nas células da memória do robô após a execução do programa que você fez para o robô, dada a imagem i como entrada. Especificamente, $m[i][j]$ é o valor da célula j. Note que o valor de c (o comprimento da sequência) é igual a $H \cdot W$ mais o número de instruções no programa que você fez para o robô.