Vision Program 視覺程序

你打算給一個機械人編寫一個視覺程序。 每次當你的機械人用攝影機拍一下張影像時，該影像將以黑白影像的型式存於機械人的記憶體內。 每張影像的含 $H \times W$ 個像素，影像的橫行編號為 0 至 $H-1$ 而直列編號為 0 至 $W-1$。 每張影像含有 **恰好兩個** 黑色的像素，而其他像素均為白色的。

你的機械人可以用一個含有一些簡單指令的程序來處理這些影像。 你將會給予 H，W 的數值及一個正整數 K。 你的目標是要編寫一子程序用以生一個給予機械人使用的程序，該程序將會被機械人用來決定對於任何給定的影像，其上的黑色像素的距離是否正好是 K。 這裡，在 r_1 行及 c_1 列上的像素與在 r_2 行及 c_2 列上的像素的距離為 $|r_1 - r_2| + |c_1 - c_2|$。 在這個式子中，$|x|$ 代表 x 的絕對值，即當 $x \geq 0$ 時，其值為 x，而當 $x < 0$ 時，其值為 $-x$。

現在我們會描述一下機械人是如何運作的。

機械人有足夠大的記憶格，編號由 0 開始。 每個記憶格可以存放 0 或 1，而且它的存放內容一旦設定後是不可更改的。 影像是一行行地存放在這些記憶格內，其編號是由 0 至 $H \cdot W - 1$。 影像的第一行將存放在記憶格 0 至 $W-1$，而影像的最後一行將存放在記憶格 $(H-1)W$ 至 $H \cdot W - 1$。 若一個位於 i 行及 j 列上的像素是黑色的，則存在記憶格 $i \cdot W + j$ 為 1，否則為 0。

機械人的程序是由一些**指令**的序列所組成，這些指令是用連續的整數並由 0 開始加以編號之。 當據行一個程式時，其內的指令是一個一個地據行的。 每個指令讀入一個或多個記憶格內的值 (我們可以稱這些值為指令的**輸入**) 同時亦產生一個為 0 或 1 的信號 (我們稱此為指令的**輸出**)。 指令 i 的輸出將會被存放在記憶格 $H \cdot W + i$ 內。 指令 i 的輸入只能是影像的像素或是之前的指令的輸出，即記憶格 0 至 $H \cdot W + i - 1$ 的內容。

我們共有四種指令:

- NOT: 有唯一一個輸入。若輸入為 0 時，其輸出為 1，否則為 0。
- AND: 有一個至多個的輸入。其輸出為 1 若及只若所有的輸入都為 1，則否輸出為 0。
- OR: 有一個至多個的輸入。其輸出為 1 若及只若所有的輸入中**最少有一個** 1。
- XOR: 有一個至多個的輸入。其輸出為 1 若及只若所有的輸入中為 1 的輸入的數目是**奇數**。

程序中最後一個指令的輸出應為 1 若兩個黑色像素的距離是剛好為 K，否則輸出應為 0。

實現詳情

你需要實現以下的子程序:

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

- H, W: 機械人攝影機所拍到的影像的大小
- K: 一個正整數
- 這個子程式需要生成一個機械人的程序。該機械人程序將對於所有由攝影機拍得的影像中決定兩個黑色像素的距離是正好是 K 與否。

這個子程序應呼叫一個或多個以下的子程序以加入指令到機械人的程序中 (原本機械人程序是空的)。

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

- 分別加一個 NOT, AND, OR, 或 XOR 指令到機械人程序中。
- N (對於 add_not 而言): 是要加入的 NOT 指令的輸入記憶格的編號
- Ns (對於 add_and, add_or, add_xor 而言): 是一個含有記憶格編號的數組，它們是 AND, OR, 或 XOR 指令將會讀入的輸入
- 每次子程序呼叫都會返回用以存放剛新加入的指令的輸出的記憶格編號。連續的子程序呼叫將會返回由 $H \cdot W$ 開始的連續整數。

機械人的程序可以容納最多 $10\,000$ 個指令。這些指令將只可以讀入總數最多為 $1\,000\,000$ 的輸入數值。換言之，在所有呼叫 add_and, add_or 及 add_xor 時所用的 Ns 數組的總長度再加上呼叫 add_not 的次數是不能超過 $1\,000\,000$ 的。

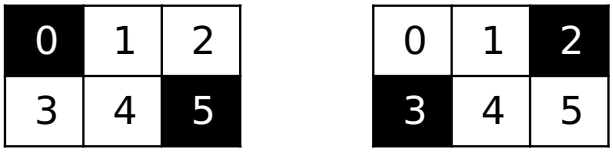
當最後一個指令都加了入機械入的程序後，子程序 construct_network 必須結束並返回。 之後機械人的程序就會被評分程式以一些影像來加以測試。 你的解可以通過一個測試點若對於每個影像，程式的最後一個指令輸出為 1 若及只若該影像中兩個黑色像素的距離正好是 K。

評分程序在評測你的程序時可能會出現以下錯誤信息，以下是這些信息的說明:

- Instruction with no inputs: 一個空的數組被作為 add_and, add_or, 或 add_xor 的輸入
- Invalid index: 不正確 (可解是負數) 記憶格編號用了作為 add_and, add_or, add_xor, 或 add_not 的輸入
- Too many instructions: 你的程序嘗加多於 $10\,000$ 個指令到機械人程序
- Too many inputs: 程序讀入總數多於 $1\,000\,000$ 的輸入數值

樣例

假設 $H = 2, W = 3, K = 3$。 在這個情況下，黑色像素的距離均為 3 的只有兩個可能的影像情況。



- 情況 1: 黑色像素在 0 及 5
- 情況 2: 黑色像素在 2 及 3

其中一個可能的機械人程序建立方法是通過以下的程序呼叫:

1. `add_and([0, 5])` 加入一個指令其輸出為 1 若影像合乎情況 1。這指令的輸出將會存放在記憶格 6 內
2. `add_and([2, 3])`, 加入一個指令其輸出為 1 若影像合乎情況 2。這指令的輸出將會存放在記憶格 7 內
3. `add_or([6, 7])`, 加入一個指令其輸出為 1 若及只若影像合乎上述的兩種情況的其中一種

限制條件

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

子任務

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) 每張影像上在 0 行及 0 列上的像素均是黑色的
7. (14 points) $K = 1$
8. (19 points) 沒有其他的限制

樣例評分程序

樣例評分程序將以以下的格式讀入數據:

- 行 1: H W K
- 行 $2 + i$ $(i \geq 0)$: $r_1[i]$ $c_1[i]$ $r_2[i]$ $c_2[i]$
- 最後一行: -1

除第一行及最後一行外，每一行都代表著一個含有兩個黑色像素的影像。 讓我們標示在行 $2 + i$ 上的影像為影像 i。 該影像有一個黑色像素 $r_1[i]$ 行及 $c_1[i]$ 列上，另一個黑色像素則 $r_2[i]$ 行及 $c_2[i]$列上。

樣例評分程序首先呼叫 `construct_network(H, W, K)`。若 `construct_network` 違反了題目敍述中的限制條件的話，樣例評分程序將會輸出在實現細則結尾部份所列出的一些錯誤信息。

否則樣例評分程序則會輸出兩筆資料:

首先樣列評分程序會以以下的格式輸出機械人程序所產生的輸出:

- 行 $1 + i$ $(0 \leq i)$: 對於影像 i，機械人程的最後一個指令的輸出 (1 或 0).

另外，樣列評分程序亦會以以下格式輸出一些資料到在當前目錄內的一個名為`log.txt` 檔案中:

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

在 $1 + i$ 行上的數列代表對於以影像 i 作為輸入時，在機械人程序運行結束後放在記憶格內的數據。詳細地說， $m[j]$ 給出存放在記憶格 j 內的數值。 注: c 的值 (即數列的長度) 是等於 $H \cdot W$ 再加上機械人程序的指令數目。