视觉程序

你在给机器人编写一个视觉程序。每次机器人的摄像头拍下一张图片时，图片将以黑白图像的形式存储在机器人的内存中。每张图像是由像素构成的 $H \times W$ 网格。图像的行从 0 到 $H-1$ 编号，列从 0 到 $W-1$ 编号。每张图像含有恰好两个黑色像素，其他像素均为白色。

机器人可以用由简单指令构成的程序来处理图像。给出 H、W 和一个正整数 K 的值，你的目标是要编写一个函数，用来为机器人生成程序。该程序需要判定图像中两个黑色像素之间的距离是否正好为 K。 这里，在第 r_1 行及第 c_1 列上的像素与在第 r_2 行及第 c_2 列上的像素之间的距离定义为 $|r_1 - r_2| + |c_1 - c_2|$。在这个式子中，$|x|$ 表示 x 的绝对值，即当 $x \geq 0$ 时，其值为 x，而当 $x < 0$ 时，其值为 $-x$。

下面描述机器人是如何运作的。

机器人的内存有足够多的存储单元，从 0 开始编号。每个存储单元可以保存 0 或 1，且它的内容一旦设置之后就不可更改。图像一行接一行地保存在存储单元里，这些存储单元的编号为从 0 到 $H \cdot W - 1$。第一行保存在存储单元 0 到 $W-1$ 里，最后一行保存在存储单元 $(H-1)W$ 到 $H \cdot W - 1$ 里。特别地，如果位于第 i 行及第 j 列上的那个像素是黑色的，则保存在存储单元 $i \cdot W + j$ 里的值为 1，否则为 0。

机器人的程序是一个指令的序列，这些指令用从 0 开始的连续整数进行编号。在程序运行时，指令将一条接一条地被执行。每条指令读取一个或多个存储单元的值（我们将这些值称为指令的输入），同时产生一个为 0 或 1 的值（我们称之为指令的输出）。指令 i 的输出将会保存在存储单元 $H \cdot W + i$ 中。指令 i 的输入只能是保存图像像素的存储单元，或者是保存之前指令输出的存储单元，也就是编号为从 0 到 $H \cdot W + i - 1$ 的存储单元。

机器人共有四种指令：

- NOT：有唯一一个输入。若输入为 0 时，其输出为 1，否则为 0。
- AND：有一个或多个输入。其输出为 1 当且仅当输入全部为 1。
- OR：有一个或多个输入。其输出为 1 当且仅当输入中至少有一个 1。
- XOR：有一个或多个输入。其输出为 1 当且仅当输入中 1 的个数是奇数。

如果两个黑色像素之间的距离正好为 K，则程序最后一条指令的输出应为 1，否则输出应为 0。

实现细节

你需要实现以下函数：

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

- H, W：机器人摄像头所拍到的图像的尺寸
- K：一个正整数
- 这个函数需要生成一个机器人的程序。对于机器人摄像头所拍到的每幅图像，该程序应判定图像中两个黑色像素之间的距离是否正好为 K。

该函数应当通过调用以下函数将指令追加到机器人的程序中（最初机器人的程序是空的）：

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

- 分别追加一条 NOT、AND、OR 或 XOR 指令。
- N（对于 add_not 而言）：要追加的 NOT 指令的输入存储单元编号
- Ns（对于 add_and、add_or、add_xor 而言）：要追加的 AND、OR 或 XOR 指令的输入存储单元的编号的数组
- 每次函数调用都会返回追加指令的输出存储单元的编号。对这些函数的连续调用将会返回从 $H \cdot W$ 开始的连续整数。

机器人的程序最多可以包含 $10\,000$ 条指令。这些指令一共最多只能读入 $1\,000\,000$ 个值。换句话说，所有 add_and、add_or 及 add_xor 调用中的 Ns 数组的长度总和再加上 add_not 调用的次数不得超过 $1\,000\,000$。

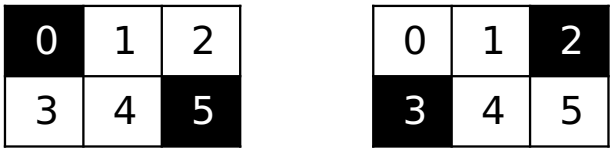
当追加完最后一条指令后，函数 construct_network 必须返回。所产生的机器人程序会在一些图像上进行评测。对于一幅图像，程序最后一条指令的输出是 1 当且仅当两个黑色像素之间的距离正好为 K。如果对测试点中的每幅图像，你的解答所产生的程序都可以正确地输出结果，那就通过了该测试点。

评测程序在评测你的程序时可能会出现以下错误信息：

- Instruction with no inputs：一个空数组被作为 add_and、add_or 或 add_xor 的输入。
- Invalid index：给 add_and、add_or、add_xor 或 add_not 提供了不正确（可能是负数）的存储单元编号作为输入。
- Too many instructions：你的函数尝试添加多于 $10\,000$ 条的指令。
- Too many inputs：程序中的指令一共读取了多于 $1\,000\,000$ 个值。

例子

假设 $H = 2$，$W = 3$，$K = 3$。在此情况下，两个黑色像素之间的距离为 3 的图像只有两种。



- 情况 1：黑色像素是 0 和 5
- 情况 2：黑色像素是 2 和 3

一种可行的方案是通过以下调用来构造机器人程序：

1. `add_and([0, 5])`，将加入一条指令，当且仅当图像符合情况 1时其输出为 1。输出结果将保存在存储单元 6 里。
2. `add_and([2, 3])`，将加入一条指令，当且仅当图像符合情况 2时其输出为 1。输出结果将保存在存储单元 7 里。
3. `add_or([6, 7])`，将加入一条指令，当且仅当上述两种情况之一成立时其输出为 1。

限制条件

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

子任务

1. （10分）$\max(H, W) \le 3$
2. （11分）$\max(H, W) \le 10$
3. （11分）$\max(H, W) \le 30$
4. （15分）$\max(H, W) \le 100$
5. （12分）$\min(H, W) = 1$
6. （8分）每幅图像上位于第 0 行且位于第 0 列的那个像素是黑色的。
7. （14分）$K = 1$
8. （19分）没有任何附加限制。

评测程序示例

评测程序示例读取下述格式的输入：

- 第 1 行：H W K
- 第 $2 + i$ 行 $(i \ge 0)$：$r_1[i]$ $c_1[i]$ $r_2[i]$ $c_2[i]$
- 最后一行：-1

除第一行及最后一行外，每一行都表示了一幅含有两个黑色像素的图像。记第 $2 + i$ 行上的图像为图像 i。该图像中，一个黑色像素位于第 $r_1[i]$ 行及第 $c_1[i]$ 列上，另一个黑色像素位于第 $r_2[i]$ 行及第 $c_2[i]$列上。

评测程序示例首先调用 `construct_network(H, W, K)`。若 `construct_network` 违反了题目描述中的限制条件，评测程序示例将会输出在实现细节一节末尾部分所列举的某条错误信息并退出。

否则，评测程序示例将输出两部分内容。

首先，评测程序示例会以下列格式输出机器人程序所产生的输出：

- 第 $1+i$ 行 $(0 \leq i)$：对于图像 i，机器人程序最后一条指令的输出（1 或 0）。

其次，评测程序示例会以下列格式输出到当前目录下一个名为 `log.txt` 的文件中：

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

在第 $1+i$ 行上的序列描述以图像 i 作为输入时，在机器人程序运行结束后放在内存中的数据。具体来说，$m[i][j]$ 是保存在存储单元 j 里的值。注意，c 的值（序列长度）等于 $H \cdot W$ 再加上机器人程序的指令数。