Zichtprogramma

Je bouwt een zichtprogramma voor een robot.

Elke keer dat de robot een foto neemt, wordt deze opgeslagen als een zwart/wit plaatje in het geheugen van de robot. Elk plaatje is een grid van $H \times W$ pixels, waarbij de rijen zijn genummerd van 0 tot en met $H - 1$ en de kolommen van 0 tot en met $W - 1$. Er zijn **precies twee** zwarte pixels in elk plaatje en alle andere pixels zijn wit.

De robot kan elk plaatje verwerken met een programma dat uit simpele instructies bestaat. Je krijgt de waarden H, W en een positief geheel getal K. Schrijf een procedure die een programma voor de robot oplevert dat, voor welk plaatje dan ook, bepaalt of de **afstand** tussen de twee zwarte pixels precies K is. De afstand tussen een pixel in rij r_1 en kolom c_1, en een pixel in rij r_2 en kolom c_2 is $|r_1 - r_2| + |c_1 - c_2|$. In deze formule is $|x|$ de absolute waarde van x, welke gelijk is aan x wanneer $x \geq 0$ en gelijk is aan $-x$ als $x < 0$.

We beschrijven nu hoe de robot werkt.

Het geheugen van de robot is een voldoende groot array van cellen, met een index vanaf 0. Elke cel kan een 0 of een 1 bevatten. De waarde kan één keer worden ingesteld en kan daarna niet meer wijzigen. Het plaatje wordt rij voor rij opgeslagen in de cellen 0 tot en met $H \cdot W - 1$. De eerste rij wordt opgeslagen in de cellen 0 tot en met $W - 1$, en de laatste rij wordt opgeslagen in de cellen $(H - 1) \cdot W$ tot en met $H \cdot W - 1$. Als de pixel in rij i en kolom j zwart is, dan is de waarde van cel $i \cdot W + j$ 1, anders is deze 0

Het programma van de robot is een rij van **instructies** die met opvolgende gehele getallen zijn genummerd, beginnend met 0. Wanneer het programma wordt uitgevoerd, dan worden de instructies één voor één uitgevoerd. Elke instructie leest de waardes van een of meer cellen (dit noemen we de **input**) en levert een enkele waarde op die 0 of 1 is (dit noemen we de **output**). De output van instructie i wordt opgeslagen in cel $H \cdot W + i$. De input van instructie i kan alleen cellen zijn waarin ofwel pixels, ofwel de output van een eerdere instructie is opgeslagen, dus de cellen 0 tot en met $H \cdot W + i - 1$.

Er zijn vier types instructie:

- `NOT`: heeft precies één cel als input. De output is 1 als de input 0 is, anders is de output 0.
- `AND`: heeft één of meer cellen als input. De output is 1 dan en slechts dan als **alle**

input 1 is.
- `OR`: heeft één of meer cellen als input. De output is 1 dan en slechts dan als **tenminste één** van inputcellen 1 is.
- `XOR`: heeft één of meer cellen als input. De output is 1 dan en slechts dan als een **oneven aantal** van de inputcellen 1 is.

De output van de laatste instructie van je programma moet 1 zijn als de afstand tussen de twee zwarte pixels precies K is, en anders 0.

Implementatiedetails

Schrijf de volgende procedure:

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

- H, W: dimensies van elke foto die de robot neemt.
- K: een positief geheel getal.
- Deze procedure moet het programma voor de robot opleveren. Voor elke foto die de robot maakt moet het programma bepalen of de afstand tussen de twee zwarte pixels in het plaatje precies K is.

De procedure moet één of meer van de volgende procedures aanroepen om instructies toe te voegen aan het programma van de robot. Het programma is aanvankelijk leeg.

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

- Deze voegen een `NOT`, `AND`, `OR`, of `XOR` instructie toe.
- N (in geval van `add_not`): de index van de cell waaruit deze instructie de input moet lezen.
- Ns (voor `add_and`, `add_or`, `add_xor`): een array met de indices van de cellen waaruit de toe te voegen `AND`, `OR`, of `XOR` instructie de input leest.
- Elke procedure levert als resultaat de index op van de cel waarin de output van de instructie wordt opgeslagen. Opvolgende aanroepen van de procedures leveren opvolgende gehele getallen op vanaf $H \cdot W$.

Het programma van de robot mag uit maximaal $10\,000$ instructies bestaan. De instructies kunnen samen maximaal $1\,000\,000$ waardes uitlezen. Met andere woorden, de totale lengte van de arrays in alle aanroepen van `add_and`, `add_or` en `add_xor`, plus het aantal aanroepen van `add_not` mag niet groter zijn dan $1\,000\,000$.

Nadat je de laatste instructie hebt toegevoegd moet de procedure `construct_network` eindigen. Je programma wordt dan geëvalueerd op een aantal plaatjes. Je oplossing

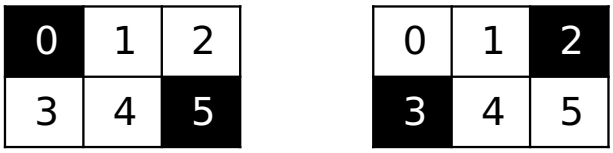
krijgt punten voor een testcase als voor alle plaatjes in de testcase de output van de laatste instructie 1 is dan en slechts dan als de afstand tussen de beide zwarte pixels in het plaatje gelijk is aan K.

Het graden van je oplossing kan resulteren in een van de volgende berichten:

- `Instruction with no inputs`: een leeg array is gegeven als input voor `add_and`, `add_or`, of `add_xor`.
- `Invalid index`: een incorrecte (mogelijk negatieve) index is gegeven als input voor `add_and`, `add_or`, `add_xor`, of `add_not`.
- `Too many instructions`: je programma probeert meer dan $10\,000$ instructies aan te maken.
- `Too many inputs`: alle instructies samen lezen meer dan $1\,000\,000$ cellen uit.

Voorbeeld

Stel $H = 2$, $W = 3$, $K = 3$. Er zijn maar twee mogelijke plaatjes waarbij de afstand tussen de pixels 3 is.



- Optie 1: de zwarte pixels zijn 0 en 5
- Optie 2: de zwarte pixels zijn 2 en 3

Een mogelijke oplossing is om een programma voor de robot te bouwen door de volgende aanroepen te doen:

1. `add_and([0, 5])`, dit voegt een instructie toe die 1 output dan en slechts dan als optie 1 van toepassing is. Het resultaat wordt opgeslagen in cel 6.
2. `add_and([2, 3])`, dit voegt een instructie toe die 1 output dan en slechts dan als optie 2 van toepassing is. Het resultaat wordt opgeslagen in cel 7.
3. `add_or([6, 7])`, dit voegt een instructie toe die 1 output dan en slechts dan als een van de beide opties van toepassing is.

Randvoorwaarden

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

Subtasks

1. (10 punten) $\max(H, W) \leq 3$
2. (11 punten) $\max(H, W) \leq 10$
3. (11 punten) $\max(H, W) \leq 30$
4. (15 punten) $\max(H, W) \leq 100$
5. (12 punten) $\min(H, W) = 1$
6. (8 punten) Pixel in rij 0 and kolom 0 is zwart in elk plaatje.
7. (14 punten) $K = 1$
8. (19 punten) Geen aanvullende voorwaarden.

Voorbeeld grader

De voorbeeld grader leest de input in het volgende formaat:

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

Elke regel, behalve de eerste en laatste regel, beschrijft een plaatje met twee zwarte pixels. Plaatje i wordt beschreven op regel $2 + i$. De ene zwarte pixel staat in rij $r_1[i]$ en kolom $c_1[i]$ en de andere in rij $r_2[i]$ en kolom $c_2[i]$.

De sample grader print `Invalid user input` als er een fout staat in de input (bijvoorbeeld als je refereert aan een niet bestaande rij of kolom).

De sample grader roept eerst `construct_network(H, W, K)` aan. Als `construct_network` een voorwaarde uit de taakbeschrijving breekt, dan print de grader één van de foutmeldingen die aan het einde van de 'Implementatiedetails'-sectie staan. Vervolgens stopt de grader.

Anders produceert de grader twee outputs:

Ten eerste print de grader de output van het programma van de robot in het volgende formaat:

- regel $1 + i$ ($0 \leq i$): de output van de laatste instructie in het programma voor plaatje i (1 or 0).

Ten tweede slaat de grader ook een bestand `log.txt` op in de huidige folder, in het volgende formaat:

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

De rij op regel $1 + i$ beschrijft de waarden van de cellen in het geheugen van de robot nadat je programma is gerund, gegeven plaatje i als input. Om precies te zijn geeft

$m[i][j]$ de waarde van cel j. De waarde van c (de lengte van de rij) is gelijk aan $H \cdot W$ plus het aantal instructies in het programma van de robot.