



Bildverarbeitung

Ein Roboter kann mit seiner Kamera Schwarz-Weiß-Bilder aufnehmen. Ein Bild ist ein $H \times W$ Pixelraster; dessen Zeilen sind mit 0 bis $H - 1$, die Spalten mit 0 bis $W - 1$ nummeriert. In jedem Bild sind **genau zwei** Pixel schwarz; alle anderen Pixel sind weiß.

Der Roboter kann mit einfachen Instruktionen programmiert werden. Du sollst eine Funktion schreiben, die für den Roboter ein Programm zur Verarbeitung der aufgenommenen Bilder generiert. Dieses Programm soll für ein gegebenes Bild bestimmen, ob die **Entfernung** zwischen den beiden schwarzen Pixeln genau K ist. Die Integerzahlen K sowie H und W sind gegeben. Die Entfernung zwischen einem Pixel (r_1, c_1) (in Zeile r_1 und Spalte c_1) und einem anderen Pixel (r_2, c_2) ist definiert als $|r_1 - r_2| + |c_1 - c_2|$.

Der Roboter funktioniert wie folgt:

Der Roboter verfügt über einen Speicher, der ausreichend groß ist. Der Speicher ist ein Array aus Zellen, mit Index 0 für die erste Zelle. Jede Zelle kann nur einmal beschrieben werden, und zwar mit den Werten 0 oder 1. Das Kamerabild ist vorne im Speicher abgelegt. Es wird zeilenweise gespeichert, und zwar in den Zellen 0 bis $H \cdot W - 1$. Die erste Zeile wird also in den Zellen 0 bis $W - 1$ gespeichert, und die letzte Zeile wird in den Zellen $(H - 1) \cdot W$ bis $H \cdot W - 1$ gespeichert. Falls das Pixel in Zeile i und Spalte j schwarz ist, enthält Zelle $i \cdot W + j$ den Wert 1 und sonst den Wert 0.

Ein Programm für den Roboter ist eine mit $0, 1, 2, \dots$ nummerierte Folge von **Instruktionen**. Bei der Ausführung eines Programms werden seine Instruktionen nach und nach abgearbeitet. Jede Instruktion liest die Werte einer oder mehrerer Zellen als **Eingabewerte** und berechnet einen einzelnen Wert 0 oder 1 als **Ausgabewert**. Die Ausgabewerte der Instruktionen werden hinter dem Kamerabild im Speicher abgelegt. Der Ausgabewert von Instruktion i wird in Zelle $H \cdot W + i$ gespeichert. Die Eingabewerte von Instruktion i können nur Bildpixel oder Ergebnisse vorhergehender Instruktionen sein, also Werte der Zellen 0 bis $H \cdot W + i - 1$.

Es gibt vier Arten von Instruktionen:

- NOT: hat genau einen Eingabewert. Der Ausgabewert ist 1, falls der Eingabewert 0 ist, und sonst 0.
- AND: hat einen oder mehrere Eingabewerte. Der Ausgabewert ist 1 genau dann, wenn **alle** Eingabewerte 1 sind.

- OR: hat einen oder mehrere Eingabewerte. Der Ausgabewert ist 1 genau dann, wenn **mindestens ein** Eingabewert 1 ist.
- XOR: hat einen oder mehrere Eingabewerte. Der Ausgabewert ist 1 genau dann, wenn **eine ungerade Anzahl** der Eingabewerte 1 ist.

Du sollst ein Programm für den Roboter generieren. Dessen letzte Instruktion soll den Wert 1 ausgeben, falls die Entfernung zwischen den beiden schwarzen Pixeln genau K ist, und sonst 0.

Implementierung

Implementiere die folgende Funktion:

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

- H, W : die Höhe und Breite des Bildes, das die Kamera des Roboters aufnimmt.
- K : ein positiver Integer
- Die Funktion soll das Programm des Roboters generieren. Für jedes Bild, das die Kamera des Roboters aufnimmt, muss dieses Programm bestimmen, ob die Entfernung zwischen den beiden schwarzen Pixeln genau K ist.

Die Funktion soll eine oder mehrere der folgenden Funktionen aufrufen, um dem Programm des Roboters (welches zu Beginn leer ist) Instruktionen anzufügen.

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

- Fügt jeweils eine NOT, AND, OR oder XOR Instruktion an.
- N (für `add_not`): der Index der Zelle, von der die NOT Instruktion den Eingabewert liest.
- Ns (für `add_and`, `add_or`, `add_xor`): Array, welches die Indizes der Zellen enthält, von welchen die AND, OR oder XOR Instruktion ihre Eingabewerte lesen.
- Jede Funktion gibt den Index der Zelle zurück, welche den Ausgabewert der Instruktion speichert. Aufeinanderfolgende Aufrufe geben aufeinanderfolgende Indizes zurück, startend mit $H \cdot W$.

Das Programm des Roboters darf aus höchstens 10 000 Instruktionen bestehen. Die Instruktionen dürfen insgesamt höchstens 1 000 000 Eingabewerte lesen. In anderen Worten: Die Gesamtlänge der Ns Arrays in allen Aufrufen von `add_and`, `add_or` und `add_xor` plus die Anzahl der Aufrufe der Funktion `add_not` darf höchstens 1 000 000 sein.

Nachdem dein Programm die letzte Instruktion angefügt hat, muss die Funktion

construct_network terminieren. Das Programm des Roboters wird dann auf einer gewissen Anzahl Bildern ausgeführt. Deine Lösung besteht einen Testfall, falls für jedes dieser Bilder die letzte Instruktion des Roboterprogramms genau dann 1 ausgibt, wenn die Entfernung zwischen den beiden schwarzen Pixeln K ist.

Beim Bewerten deiner Lösung können folgende Fehlermeldungen auftreten:

- `Instruction with no inputs`: Ein leeres Array wurde als Eingabe zu `add_and`, `add_or` oder `add_xor` übergeben.
- `Invalid index`: Ein ungültiger Index (möglicherweise ein negativer) wurde als Eingabe für `add_and`, `add_or`, `add_xor` oder `add_not` übergeben.
- `Too many instructions`: Deine Funktion hat versucht, mehr als 10 000 Instruktionen anzufügen.
- `Too many inputs`: Die Instruktionen lesen insgesamt mehr als 1 000 000 Eingabewerte.

Beispiel

Angenommen $H = 2$, $W = 3$, $K = 3$. Es gibt nur zwei Möglichkeiten, bei denen die Entfernung zwischen den beiden schwarzen Zellen 3 ist.

0	1	2
3	4	5

0	1	2
3	4	5

- Fall 1: Die schwarzen Pixel sind 0 und 5.
- Fall 2: Die schwarzen Pixel sind 2 und 3.

Eine mögliche Lösung ist ein Programm für den Roboter zu bauen, indem man folgende Funktionsaufrufe macht:

1. `add_and([0, 5])`, dies fügt eine Instruktion hinzu, die genau dann 1 ausgibt, falls der erste Fall zutrifft. Der Ausgabewert wird in Zelle 6 gespeichert.
2. `add_and([2, 3])`, dies fügt eine Instruktion hinzu, die genau dann 1 ausgibt, falls der zweite Fall zutrifft. Der Ausgabewert wird in Zelle 7 gespeichert.
3. `add_or([6, 7])`, dies fügt eine Instruktion hinzu, die genau dann 1 ausgibt, falls einer der obigen Fälle zutrifft.

Beschränkungen

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

Subtasks

1. (10 Punkte) $\max(H, W) \leq 3$
2. (11 Punkte) $\max(H, W) \leq 10$
3. (11 Punkte) $\max(H, W) \leq 30$
4. (15 Punkte) $\max(H, W) \leq 100$
5. (12 Punkte) $\min(H, W) = 1$
6. (8 Punkte) Das Pixel in Zeile 0 and Spalte 0 ist in jedem Bild schwarz.
7. (14 Punkte) $K = 1$
8. (19 Punkte) Keine weiteren Beschränkungen.

Beispiel-Grader

Der Beispiel-Grader liest die Eingaben im folgenden Format:

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

Jede Zeile ausser der ersten und letzten beschreibt ein Bild mit zwei schwarzen Pixeln. Wir bezeichnen das Bild, welches in Zeile $2 + i$ beschrieben wird, als Bild i . Ein schwarzer Pixel ist in Zeile $r_1[i]$ und Spalte $c_1[i]$ und der andere in Zeile $r_2[i]$ und Spalte $c_2[i]$.

Der Beispiel-Grader ruft zuerst `construct_network(H, W, K)` auf. Falls `construct_network` irgendeine der oben genannten Beschränkungen verletzt, gibt der Beispiel-Grader eine der Fehlermeldungen aus, die oben am Ende des Abschnitts "Implementierung" aufgeführt sind. Ansonsten passiert Folgendes:

Zum einen gibt der Beispiel-Grader den Ausgabewert des Programms des Roboters im folgenden Format aus:

- Zeile $1 + i$ ($0 \leq i$): Ausgabewert der letzten Instruktion im Programm des Roboters für Bild i (1 or 0).

Zum anderen schreibt der Beispiel-Grader eine Datei `log.txt` im momentanen Ordner im folgenden Format:

- Zeile $1 + i$ ($0 \leq i$): $m[i][0] \ m[i][1] \ \dots \ m[i][c - 1]$

Die Folge in Zeile $1 + i$ beschreibt die Werte, die im Speicher des Roboters stehen, nachdem das Programm des Roboters auf Bild i ausgeführt wurde. Genauer gesagt ist $m[i][j]$ der Wert der Zelle j . Beachte, dass c (die Länge der Folge) gleich $H \cdot W$ plus die Anzahl Instruktionen im Programm des Roboters ist.