Зрителна програма

Вие разработвате програма за зрението на робот. Всеки път, когато камерата на робота направи снимка, тя се запазва като черно бяло изображение в паметта на робота. Всяко изображение е таблица от $H \times W$ пиксела, като редовете са номерирани от 0 до $H-1$, а колоните са номерирани от 0 до $W-1$. На всяко изображение има **точно два** черни пиксела, а всички останали са бели.

Роботът може да обработва всяко изображение с програма, която се състои от елементарни инструкции. Зададени са стойностите на H, W и положително цяло число K. Вашата цел е да напишете процедура, която "пише" програма за робота. Произведената програма трябва за произволно изображение, да определя дали **разстоянието** между двата черни пиксела е точно K. Разстоянието между пиксел, намиращ се на ред r_1 и колона c_1 и пиксел, намиращ се на ред r_2 и колона c_2 се пресмята по формулата $|r_1 - r_2| + |c_1 - c_2|$. В тази формула $|x|$ е абсолютната стойност на x.

Следва описание на начина, по който роботът интерпретира получената програма.

Паметта на робота е достатъчно голям масив от клетки и е индексирана от 0. Всяка клетка на паметта може да съдържа стойност 0 или 1 и след като веднъж тази стойност е зададена, тя не може да се променя. Изображението е записано ред по ред в клетките на паметта с индекси от 0 до $H \cdot W - 1$. Първият ред е записан в клетки от 0 до $W-1$, а последният ред е записан в клетки от $(H-1) \cdot W$ до $H \cdot W - 1$. Ако пикселът, намиращ се на ред i и колона j е черен, то стойността на клетката от паметта с индекс $i \cdot W + j$ е 1, в противен случай стойността е 0.

Програмата на робота е последователност от **инструкции**, които са номерирани с последователни цели числа, започващи от 0. Когато програмата се стартира, инструкциите се изпълняват една след друга. Всяка инструкция чете стойностите на една или повече клетки от паметта на робота (наричаме тези клетки **входове** на инструкцията) и произвежда една стойност, която е равна на 0 или 1 (наричаме тази стойност **изход** на инструкцията). Изходът на инструкция с номер i се запазва в клетка с индекс $H \cdot W + i$. Входовете на инструкция с номер i могат да са само клетки, които съдържат стойности на пиксели или изходи на изпълнени вече инструкции, т.е. клетки с номера от 0 до $H \cdot W + i - 1$.

Има четири вида инструкции:

- `NOT`: има точно един вход. Ако входът е равен на 0 -- изходът е 1, а ако входът е 1 -- изходът е равен на 0.
- `AND`: има един или повече входа. Изходът от инструкцията е равен на 1 тогава и само тогава, когато **всички** входове са равни на 1.
- `OR`: има един или повече входа. Изходът от инструкцията е равен на 1 тогава и само тогава, когато **поне един** от входовете е равен на 1.
- `XOR`: има един или повече входа. Изходът от инструкцията е равен на 1 тогава и само тогава, когато **нечетен брой** от входовете са равни на 1.

Изходът на последната инструкция от програмата трябва да е 1, ако разстоянието между двата черни пиксела е точно K, и 0 в противен случай.

Детайли по реализацията

Трябва да реализирате следната процедура,:

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

- H, W: размери на всяко изображение, заснимано от камерата на робота
- K: положително цяло число
- Процедурата трябва да създава програмата на робота. Създадената програма трябва да може, за всякакво изображение, записано от камерата на робота, да определя дали разстоянието между двата черни пиксела на него е точно K.

Вашата процедура може да прави извиквания към следните процедури (произволен брой пъти всяка), за да добавя инструкции към програмата на робота (която в началото е празна):

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

- всяка от тези процедури добавя инстукция от тип `NOT`, `AND`, `OR` или `XOR` съответно.
- N (за `add_not`): индексът на клетката, от която добавената инструкция от тип `NOT` чете входа си
- Ns (за `add_and`, `add_or`, `add_xor`): масив, съдържащ индекси на клетките, от които добавената инстукция от тип `AND`, `OR` или `XOR` чете входовете си
- Всяка от горните процедури връща индекс на клетката от паметта, в която ще запише изхода от изпълнението си. Извикванията на тези процедури връщат последователни цели числа, започващи от $H \cdot W$.

Програмата на робота може да се състои най-много от $10\,000$ инструкции. Инструкциите могат да прочетат общо най-много $1\,000\,000$ стойности. С други думи, сумата от дължините на масивите Ns от всички извиквания на `add_and`, `add_or` и `add_xor` плюс броя на извикванията на `add_not` не може да надхвърля $1\,000\,000$.

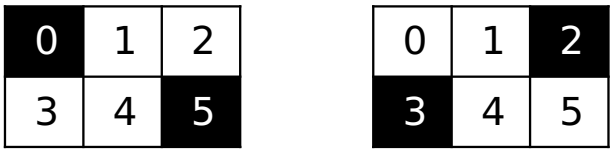
След добаване на последната инструкция процедурата `construct_network` трябва да извика return. Получената програма за робота ще бъде тествана на различни изображения. Вашето решение ще вземе точките, предвидени за конкретен тест, ако за всяко изображение, включено в групата за теста, последната инструкция от програмата върне изход 1 тогава и само тогава, когато разстоянието между двата черни пиксела на изображението е равно на K.

Оценяващата система може да върне едно от следните съобщения :

- `Instruction with no inputs`: подаден е празен масив към `add_and`, `add_or` или `add_xor`.
- `Invalid index`: некоректен индекс на клетка от паметта (възможна отрицателна стойност) е подаден като вход на `add_and`, `add_or`, `add_xor` или `add_not`.
- `Too many instructions`: вашата процедура се опитва да добави повече от $10\,000$ инструкции.
- `Too many inputs`: инструкциите, които генерирате четат общо повече от $1\,000\,000$ стойности.

Пример

Нека $H = 2$, $W = 3$, $K = 3$. Съществуват само две различни изображения, в които разстоянието между черните пиксели е равно на 3.



- Случай 1: черните пиксели са 0 и 5
- Случай 2: черните пиксели са 2 и 3

Възможно решение е да се конструира програма за робота, правейки следните извиквания:

1. `add_and([0, 5])` - добавя се инструкция, която дава изход 1 тогава и само тогава, когато е на лице първия случай. Стойността се запазва в клетка 6.
2. `add_and([2, 3])` - добавя се инстукция, която дава изход 1, тогава и само тогава, когато е на лице втория случай. Стойността се запазва в клетка 7.

3. `add_or([6, 7])` - добавя се инструкция, която дава изход 1 тогава и само тогава, когато е на лице един от двата случая.

Ограничения

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

Подзадачи

1. (10 точки) $\max(H, W) \leq 3$
2. (11 точки) $\max(H, W) \leq 10$
3. (11 точки) $\max(H, W) \leq 30$
4. (15 точки) $\max(H, W) \leq 100$
5. (12 точки) $\min(H, W) = 1$
6. (8 точки) Пикселът на ред 0 и колона 0 е черен за всяко изображение.
7. (14 точки) $K = 1$
8. (19 точки) Без допълнителни ограничения.

Примерен грейдър

Предоставеният примерен грейдър чете входните данни в следния формат:

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

Всеки ред, освен първия и последния, представлява едно изображение с два черни пиксела. Означаваме изображението, описано на ред $2 + i$ с "изображение i". За всяко изображение единият черен пиксел е на ред $r_1[i]$ и колона $c_1[i]$, а другият - на ред $r_2[i]$ и колона $c_2[i]$.

Примерният грейдър първо извиква `construct_network(H, W, K)`. Ако `construct_network` наруши някои от ограниченията, описани в условието на задачата, грейдърът извежда едно от съобщенията за грешка, описани в края на секция "Детайли за реализацията" и приключва работа. В противен случай примерният грейдър извежда две неща:

Първо на стандартния изход грейдърът извежда изхода от изпълнение на програмата за робота в следния формат:

- ред $1 + i$ $(0 \leq i)$: стойност на изхода на последната инструкция за изображение i (1 или 0).

Освен това примерният грейдър попълва файл `log.txt` в текущата директория, който има следния формат:

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

Редицата на ред $1 + i$ описва стойностите на клетките от паметта на робота след приключване на изпълнение на програмата върху изображение i. $m[i][j]$ показва стойността на клетка j. Стойността c (дължината на редицата) е равна на $H \cdot W$ плюс броя инструкции в програмата на робота.