Компьютерное зрение

Вам необходимо реализовать программу компьютерного зрения для робота. Каждый раз, когда камера робота делает снимок, он сохраняется в памяти робота как чёрно-белое изображение. Каждое изображение представляет собой прямоугольную таблицу пикселей размером $H \times W$, строки пронумерованы от 0 до $H - 1$, а столбцы пронумерованы от 0 до $W - 1$. В каждом изображении **ровно два** чёрных пикселя, остальные пиксели белые.

Робот может обрабатывать каждое изображение, используя программу, составленную из простых инструкций. Вам даны значения H, W и положительное целое число K. Ваша цель — написать функцию, которая составит программу для робота, определяющую для любого изображения, верно ли, что **расстояние** между двумя чёрными пикселями равно в точности K. Здесь расстояние между пикселем в строке r_1 в столбце c_1 и пикселем в строке r_2 в столбце c_2 равно $|r_1 - r_2| + |c_1 - c_2|$. В этой формуле $|x|$ обозначает абсолютную величину числа x, которая равна x, если $x \geq 0$, либо $-x$, если $x < 0$.

Рассмотрим, как работает робот.

Память робота представляет собой достаточно большой массив ячеек, проиндексированных, начиная с 0. Каждая ячейка может содержать 0 или 1, и значения ячеек после того, как они присваиваются, не могут быть изменены. Изображение сохранено в памяти строка за строкой в ячейках с 0 по $H \cdot W - 1$. Первая строка сохраняется в ячейках с 0 по $W - 1$, а последняя строка сохраняется в ячейках с $(H - 1) \cdot W$ по $H \cdot W - 1$. Таким образом, если пиксель в строке i в столбце j чёрный, то значение ячейки $i \cdot W + j$ равно 1, иначе оно равно 0.

Программа для робота представляет собой последовательность **инструкций**, пронумерованных последовательными целыми числами, начиная с 0. Когда программа запускается, инструкции исполняются одна за другой. Каждая инструкция читает значения одной или более ячеек (будем называть эти значения **входами** инструкции) и результат её выполнения представляет собой одно число, равное 0 или 1 (будем называть это значение **результатом** инструкции). Результат инструкции i сохраняется в ячейке $H \cdot W + i$. Входами для инструкции i могут быть только значения ячеек, в которых содержатся пиксели изображения или результаты предыдущих инструкций, то есть ячейки с номерами от 0 до $H \cdot W + i - 1$.

Есть четыре типа инструкций:

- `NOT`: имеет ровно один вход. Её результат равен 1, если вход равен 0, иначе результат равен 0.
- `AND`: имеет один или более входов. Её результат равен 1, если и только если **все** её входы равны 1.
- `OR`: имеет один или более входов. Её результат равен 1, если и только если **хотя бы один** её вход равен 1.
- `XOR`: имеет один или более входов. Её результат равен 1, если и только если **нечётное количество** её входов равны 1.

Результат последней инструкции в программе должен быть равен 1, если расстояние между чёрными пикселями равно в точности K, либо 0 в противном случае.

Детали реализации

Вы должны реализовать следующую функцию:

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

- H, W: размеры изображения, полученного камерой робота
- K: положительное число
- Функция должна создать программу для робота. Для любого изображения, сделанного камерой робота, созданная программа должна определять, верно ли, что расстояние между двумя черными пикселями на изображении равно в точности K.

Эта функция должна вызвать одну или более из следующих функций, каждая из которых добавляет инструкцию в конец программы робота (которая исходно пуста):

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

- Функции добавляют инструкцию `NOT`, `AND`, `OR` и `XOR`, соответственно.
- N (для функции `add_not`): номер ячейки, в которой находится вход для инструкции `NOT`
- Ns (для функций `add_and`, `add_or`, `add_xor`): массив, содержащий номера ячеек, которые содержат входы для инструкции `AND`, `OR` или `XOR`
- Каждая функция возвращает номер ячейки, в которую помещается результат выполнения инструкции. Последовательные вызовы функций возвращают

последовательные целые числа, начиная с $H \cdot W$.

Программа робота может содержать не более $10\,000$ инструкций. Инструкции могут суммарно использовать не более $1\,000\,000$ входных значений. Другими словами, суммарная длина массивов Ns во всех вызовах `add_and`, `add_or` и `add_xor` плюс количество вызовов `add_not` не должно превышать $1\,000\,000$.

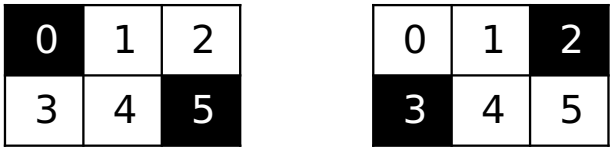
После добавления последней инструкции функция `construct_network` должна завершить работу. Программа для робота будет протестирована на некотором множестве изображений. Ваше решение проходит тест, если для каждого из этих изображений результат последней инструкции равен 1 тогда и только тогда, когда расстояние между чёрными пикселями на изображении в точности равно K.

По итогам тестирования вашего решения вы можете получить следующие сообщения об ошибках на английском языке:

- `Instruction with no inputs`: функция `add_and`, `add_or` или `add_xor` получила в качестве аргумента пустой массив.
- `Invalid index`: некорректный (возможно, отрицательный) номер ячейки указан как вход для `add_and`, `add_or`, `add_xor` или `add_not`.
- `Too many instructions`: ваша функция попыталась добавить в программу более $10\,000$ инструкций.
- `Too many inputs`: инструкции суммарно используют более чем $1\,000\,000$ входных значений.

Пример

Пусть $H = 2$, $W = 3$, $K = 3$. Существуют два изображения, на которых расстояние между двумя чёрными пикселями равно 3.



- Вариант 1: чёрные пиксели в ячейках 0 и 5
- Вариант 2: чёрные пиксели в ячейках 2 и 3

Возможное решение — создать программу для робота, вызвав следующие функции:

1. `add_and([0, 5])`, добавить инструкцию, результат которой равен 1, если и только если имеет место вариант 1. Результат этой инструкции сохраняется в ячейке 6.
2. `add_and([2, 3])`, добавить инструкцию, результат которой равен 1, если и только если имеет место вариант 2. Результат этой инструкции сохраняется в

ячейке 7.
3. `add_or([6, 7])`, добавить инструкцию, результат которой равен 1, если и только если один из двух этих вариантов имеет место.

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 баллов) $\max(H, W) \leq 3$
2. (11 баллов) $\max(H, W) \leq 10$
3. (11 баллов) $\max(H, W) \leq 30$
4. (15 баллов) $\max(H, W) \leq 100$
5. (12 баллов) $\min(H, W) = 1$
6. (8 баллов) На всех изображениях пиксель в строке 0 в столбце 0 чёрный.
7. (14 баллов) $K = 1$
8. (19 баллов) Нет дополнительных ограничений.

Пример проверяющего модуля

Пример проверяющего модуля читает данные в следующем формате:

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

Каждая строка, кроме первой и последней, описывает изображение с двумя чёрными пикселями. Изображение, описанное в строке $2 + i$, будем называть изображением i. Один чёрный пиксель находится в строке $r_1[i]$ и столбце $c_1[i]$, другой в строке $r_2[i]$ и столбце $c_2[i]$.

Пример проверяющего модуля сначала вызывает `construct_network(H, W, K)`. Если `construct_network` нарушает какие-либо ограничения, описнные в условии задачи, то выводится одно из сообщений об ошибке, указанных в разделе Детали реализации, и программа завершается.

Иначе пример проверяющего модуля выводит информацию двумя способами.

Во-первых, он выводит результат работы программы для робота в следующем формате:

- строка $1 + i$ $(0 \leq i)$: результат последней инструкции для программы робота при запуске на изображении i (1 или 0).

Во-вторых, пример проверяющего модуля создаёт файл `log.txt` в текущем каталоге в следующем формате:

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

Массив в строке $1 + i$ содержит значения, сохраненные в памяти робота после запуска программы на изображении i. А именно, значение $m[i][j]$ содержит значение ячейки j. Обратите внимание, что число c (длина массива) равно $H \cdot W$ плюс число инструкций в программе для робота.