Програма за гледање

Вие имплементирате роботска програма за гледање. Секогаш кога камерата на роботот ќе направи слика, таа се зачувува како црно-бела слика во меморијата на роботот. Секоја слика е матрица со $H \times W$ пиксели, со редици нумерирани со целите броеви од 0 до $H - 1$ и колони нумерирани со целите броеви од 0 до $W - 1$. Постојат **точно два** црни пиксела во секоја слика, а сите останати пиксели се бели.

Роботот може да ја процесира секоја слика со помош на програма што се состои од едноставни инструкции. Дадени се вредностите H, W, како и позитивен цел број K. Ваша цел е да напишете процедура што ќе продуцира програма за роботот која што, за која било слика, ќе определува дали **растојанието** помеѓу двата црни пиксела е точно K. Овде, растојанието помеѓу пикселот во редицата r_1 и колоната c_1 и пикселот во редицата r_2 и колоната c_2 е $|r_1 - r_2| + |c_1 - c_2|$. Во оваа формула $|x|$ ја означува апсолутната вредност на x, која што е еднаква на x ако $x \geq 0$ и е еднаква на $-x$ ако $x < 0$.

Во продолжение ќе опишеме како функционира роботот.

Меморијата на роботот е доволно голема низа од клетки, чии што редни броеви (индекси) започнуваат од 0. Секоја клетка може да зачува или 0 или 1, и кога еднаш ќе се постави нејзината вредност, истата повеќе нема да биде променета. Сликата се зачувува редица по редица во клетки чии што редни броеви (индекси) се од 0 до $H \cdot W - 1$. Првата редица се зачувува во клетките од 0 до $W - 1$, а последната редица се зачувува во клетките од $(H - 1) \cdot W$ до $H \cdot W - 1$. Конкретно, ако пикселот во редицата i и колоната j е црн, тогаш вредноста на клетката $i \cdot W + j$ е 1, во спротивно таа е 0.

Програма за роботот е секвенца од **инструкции**, кои што се нумерирани со последователни цели броеви почнувајќи од 0. Кога ќе се стартува програмата, инструкциите се извршуваат една по една. Секоја инструкција ги чита вредностите на една или повеќе клетки (овие вредности ги нарекуваме **влезови** на инструкцијата) и продуцира една вредност еднаква на 0 или 1 (оваа вредност ја нарекуваме **излез** од инструкцијата). Излезот од инструкцијата i се чува во клетката $H \cdot W + i$. Влезови на инструкцијата i може да бидат само клетки кои што чуваат или пиксели или излези од претходни инструкции т.е. клетките од 0 до $H \cdot W + i - 1$.

Постојат четири типа на инструкции:

- `NOT`: има точно еден влез. Излезот од оваа инструкција е 1 ако влезот е 0, во спротивно излезот е 0.
- `AND`: има еден или повеќе влезови. Нејзиниот излез е 1 ако и само ако **сите** влезови се 1.
- `OR`: има еден или повеќе влезови. Нејзиниот излез е 1 ако и само ако **барем еден** од влезовите е 1.
- `XOR`: има еден или повеќе влезови. Нејзиниот излез е 1 ако и само ако **непарен број** од влезовите се 1.

Излезот од последната инструкција од програмата треба да биде 1 ако растојанието помеѓу двата црни пиксела е точно K, а во спротивно треба да биде 0.

Имплементациски детали

Треба да ја имплементирате следнава процедура:

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

- H, W: димензии на секоја од сликите направена од камерата на роботот
- K: позитивен цел број
- Оваа процедура треба да продуцира роботска програма. За секоја слика направена од страна на камерата на роботот, оваа програма треба да определи дали растојанието помеѓу двата црни пиксела во сликата е точно K.

Оваа процедура треба да повикува една или повеќе од следниве процедури за да додава инструкции во роботската програма (која на почетокот е празна):

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

- Додади `NOT`, `AND`, `OR` или `XOR` инструкција, соодветно.
- N (за `add_not`): реден број (индекс) на клетката од која што додадената `NOT` инструкција го чита својот влез
- Ns (за `add_and`, `add_or`, `add_xor`): низа што ги содржи редните броеви (индексите) на клетките од кои што додадената `AND`, `OR` или `XOR` инструкција ги чита своите влезови
- Секоја од процедурите го враќа редниот број (индексот) на клетката која што го чува излезот од инструкцијата. Последователните повици до овие процедури враќаат последователни цели броеви почнувајќи од $H \cdot W$.

Роботската програма може да се состои од најмногу $10\,000$ инструкции. Инструкциите може да читаат најмногу вкупно $1\,000\,000$ вредности. Со други зборови, вкупната должина на Ns низите во сите повици до `add_and`, `add_or` и `add_xor` плус бројот на повици до `add_not` не смее да надмине $1\,000\,000$.

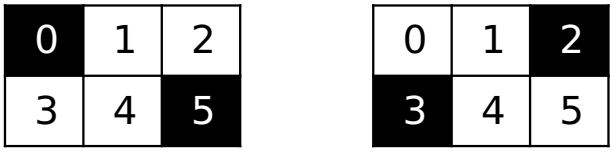
По додавањето на последната инструкција, процедурата `construct_network` треба да заврши. Роботската програма потоа ќе биде евалуирана на одреден број на слики. Вашето решение поминува на даден тест случај ако за секоја од овие слики, излезот од последната инструкција е 1 ако и само ако растојанието помеѓу двата црни пиксела во сликата е еднакво на K.

Оценувањето на вашето решение може да резултира со една од следниве пораки за грешка:

- `Instruction with no inputs`: како влез на `add_and`, `add_or` или `add_xor` била зададена празна низа.
- `Invalid index`: неточен (можеби негативен) индекс на клетка бил зададен како влез на `add_and`, `add_or`, `add_xor` или `add_not`.
- `Too many instructions`: вашата процедура се обидела да додаде повеќе од $10\,000$ инструкции.
- `Too many inputs`: инструкциите читаат вкупно повеќе од $1\,000\,000$ вредности.

Пример

Да претпоставиме дека $H = 2$, $W = 3$, $K = 3$. Постојат само две можни слики каде што растојанието помеѓу црните пиксели е 3.



- Случај 1: црни пиксели се 0 и 5
- Случај 2: црни пиксели се 2 и 3

Можно решение е да се изгради роботската програма со правење на следните повици:

1. `add_and([0, 5])`, кој што додава инструкција што на излез дава 1 ако и само ако важи првиот случај. Излезот се зачувува во клетката 6.
2. `add_and([2, 3])`, кој што додава инструкција што на излез дава 1 ако и само ако важи вториот случај. Излезот се зачувува во клетката 7.
3. `add_or([6, 7])`, кој што додава инструкција што на излез дава 1 ако и само ако важи еден од погоре наведените случаи.

Ограничувања

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

Подзадачи

1. (10 поени) $\max(H, W) \leq 3$
2. (11 поени) $\max(H, W) \leq 10$
3. (11 поени) $\max(H, W) \leq 30$
4. (15 поени) $\max(H, W) \leq 100$
5. (12 поени) $\min(H, W) = 1$
6. (8 поени) Пикселот во редица 0 и колона 0 е црн во секоја слика.
7. (14 поени) $K = 1$
8. (19 поени) Нема други ограничувања.

Пример-оценувач

Пример-оценувачот ги чита влезните податоци во следниот формат:

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

Секоја линија, освен првата и последната, претставува една слика со два црни пиксела. Сликата опишана во линијата $2 + i$ ќе ја означуваме со слика i. Едниот црн пиксел е во редицата $r_1[i]$ и колоната $c_1[i]$, а другиот е во редицата $r_2[i]$ и колоната $c_2[i]$.

Пример-оценувачот прво ја повикува `construct_network(H, W, K)`. Ако `construct_network` прекрши некое од ограничувањата опишани во текстот на задачата, пример-оценувачот ќе отпечати една од пораките за грешка наведени во делот Имплементациски детали, и ќе заврши со работа.

Во спротивно, пример-оценувачот ќе продуцира два излеза.

Прво, пример-оценувачот го печати излезот од роботската програма во следниот формат:

- линии $1 + i$ $(0 \leq i)$: излез од последната инструкција во роботската програма за слика i (1 или 0).

Второ, пример-оценувачот создава датотека `log.txt` во тековната папка, во

следниот формат:

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

Секвенцата во линијата $1 + i$ ги опишува вредностите зачувани во мемориските клетки на роботот по извршувањето на роботската програма, ако на влез е зададена сликата i. Конкретно, $m[i][j]$ ја дава вредноста на клетката j. Забележете дека вредноста на c (должината на секвенцата) е еднаква на $H \cdot W$ плус бројот на инструкции во роботската програма.