Vzdialenosť pixelov (Vision Program)

Vašou úlohou bude vyrobiť postupnosť inštrukcií pre robota spracúvajúceho obrázky.

Robot má foťák, ktorým občas odfotí chladničku a občas mrazničku. Každá fotka je čiernobiela bitmapa, ktorá má výšku h a šírku w. Riadky sú očíslované od 0 po $h-1$ a stĺpce sú očíslované od 0 po $w-1$. Na každej fotke sú skoro všetky pixely biele, až na **presne dva**, ktoré sú čierne.

Robot potrebuje odlíšiť chladničky od mrazničiek. Vie sa, že na fotke chladničky sú dva čierne pixely vždy vo vzdialenosti presne k, zatiaľ čo pre fotku mrazničky to nikdy nie je presne k. **Vzdialenosť** dvoch pixelov, z ktorých jeden leží v riadku r_1 a stĺpci c_1 a druhý leží v riadku r_2 a stĺpci c_2, je číslo $|r_1 - r_2| + |c_1 - c_2|$.

Váš program dostane na vstupe čísla h, w a k. Z nich má vyrobiť program pre robota, ktorý bude zisťovať, či je daná bitmapa fotkou chladničky. Tento program musí byť tvorený postupnosťou jednoduchých inštrukcií, ktoré si teraz popíšeme.

Pamäť robota je dostatočne veľké pole buniek, indexované od nuly. Do každej bunky si robot môže uložiť jeden bit - presnejšie, buď hodnotu 0 alebo hodnotu 1. Akonáhle už je v niektorej bunke uložená hodnota, už robot obsah tejto bunky nemôže zmeniť. Na začiatku behu robotovho programu sú už uložené hodnoty v prvých hw bunkách poľa: je do nich riadok po riadku uložená fotka, ktorú má robotov program na vstupe.

Presnejšie, riadok 0 fotky je v bunkách 0 až $w-1$, riadok 1 je v bunkách w až $2w-1$, atď., až posledný riadok nájdete v bunkách s číslami $(h-1)w$ až $hw-1$. Ešte presnejšie, ak je na políčku v riadku r a stĺpci c čierny pixel, je v bunke číslo $rw+c$ uložená hodnota 1, inak je tam uložená hodnota 0. Zvyšok pamäte robota je zatiaľ prázdny.

Robotov program je postupnosť **inštrukcií**. Inštrukcie postupne číslujeme, začínajúc od nuly. Keď sa program vykonáva, vykonávame postupne jeho inštrukcie jednu po druhej vo vami danom poradí. Každá inštrukcia najskôr načíta hodnoty z jednej alebo viacerých buniek pamäte (tieto hodnoty nazývame **vstupmi** tejto inštrukcie) a potom z nich vyrobí jednu hodnotu rovnú 0 alebo 1 (ktorú nazývame **výstup**). Výstup inštrukcie číslo i robot uloží do bunky pamäte s číslom $hw+i$.

Vstupmi inštrukcie i môžu byť len tie bunky, ktoré v danej chvíli obsahujú hodnoty - čiže bunky obsahujúce vstup a bunky obsahujúce výstupy predchádzajúcich inštrukcií. Ide teda o bunky s číslami od 0 po $hw+i-1$ vrátane.

Sú štyri typy inštrukcií:

- `NOT`: má presne jeden vstup. Výstupom je 1 ak na vstupe bola 0 a naopak.
- `AND`: má aspoň jeden vstup. Výstupom je 1 práve vtedy, ak **všetky** vstupy mali hodnotu 1.
- `OR`: má aspoň jeden vstup. Výstupom je 1 práve vtedy, ak **aspoň jeden** vstup mal hodnotu 1.
- `XOR`: má aspoň jeden vstup. Výstupom je 1 práve vtedy, ak **nepárne veľa** vstupov malo hodnotu 1.

Pre ľubovoľnú bitmapu s vlastnosťami zo zadania musí platiť, že posledná inštrukcia robotovho programu vráti hodnotu 1 ak bitmapa bola fotkou chladničky (t.j. vzdialenosť čiernych pixelov bola presne k) a hodnotu 0 inak.

Detaily implementácie

Implementujte nasledujúcu procedúru:

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

- H, W: rozmery bitmapy
- K: kladné celé číslo predstavujúce želanú vzdialenosť čiernych pixelov
- Táto procedúra má vyrobiť a graderu postupne oznámiť jeden možný správny program pre robota.

Na to, aby vaša procedúra oznámila graderu robotov program, môže volať nižšie popísané funkcie gradera. Každá z týchto funkcií pridá na koniec robotovho programu jednu novú inštrukciu. (Na začiatku behu vašej procedúry je robotov program prázdny.)

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

- Ako isto chápete, tieto funkcie pridávajú inštrukcie typu `NOT`, `AND`, `OR`, a `XOR`.
- N (pre `add_not`): index bunky ktorej hodnotu berie `NOT` ako vstup.
- Ns (pre `add_and`, `add_or`, `add_xor`): pole obsahujúce indexy buniek, ktoré príslušná inštrukcia berie ako vstupy.
- Každá z týchto funkcií vám ako návratovú hodnotu oznámi číslo bunky, do ktorej si robot uloží výstup práve pridanej inštrukcie. Postupné volania týchto funkcií z vašej procedúry vám teda vždy budú vracať postupne hodnoty $hw, hw + 1, hw + 2$, atď.

Platia nasledovné obmedzenia:

- Robotov program môže mať nanajvýš $10\,000$ inštrukcií.
- Všetky inštrukcie dokopy môžu mať nanajvýš $1\,000\,000$ vstupov. Inými slovami, keď sčítame dĺžky všetkých polí Ns pre všetky volania funkcií `add_and`, `add_or` a `add_xor` a pridáme k tomu počet volaní funkcie `add_not`, výsledok nesmie presiahnuť $1\,000\,000$.

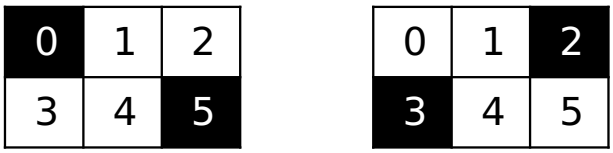
Po tom, ako graderu oznámi poslednú inštrukciu, má vaša procedúra `construct_network` skončiť (return) a nechať grader, nech vyhodnotí oznámený robotov program. Grader následne robotov program otestuje na sade rôznych fotiek. Vaša procedúra konkrétny testovací vstup úspešne vyrieši len vtedy, ak ňou vyrobený program robota dá pre všetky testovacie fotky správne výstupy.

Od gradera (ukážkového aj reálneho) môžete dostať jednu z nasledujúcich chybových správ:

- `Instruction with no inputs`: vaša procedúra zavolala funkciu `add_and`, `add_or`, alebo `add_xor` a dala jej na vstupe prázdne pole.
- `Invalid index`: vaša procedúra zavolala jednu z funkcií gradera a dala jej neplatný index vstupu (priveľký alebo záporný).
- `Too many instructions`: vaša procedúra sa pokúsila do robotovho programu pridať viac ako $10\,000$ inštrukcií.
- `Too many inputs`: vaša procedúra sa pokúsila do robotovho programu pridať inštrukcie, ktoré dokopy majú viac ako $1\,000\,000$ vstupov.

Príklad

Predpokladajme, že $h = 2$, $w = 3$ a $k = 3$. Existujú len dve bitmapy, ktoré predstavujú fotky chladničky (teda majú čierne pixely vo vzdialenosti 3):



- Bitmapa 1: čierne pixely zodpovedajú bunkám 0 a 5
- Bitmapa 2: čierne pixely zodpovedajú bunkám 2 a 3

Jedným správnym riešením je graderu oznámiť program robota pomocou nasledujúcich volaní jeho funkcií:

1. `add_and([0, 5])`. Táto inštrukcia do bunky 6 uloží hodnotu 1 práve vtedy, keď je na vstupe bitmapa 1.
2. `add_and([2, 3])`. Táto inštrukcia do bunky 7 uloží hodnotu 1 práve vtedy, keď je na vstupe bitmapa 2.
3. `add_or([6, 7])`. Táto inštrukcia do bunky 8 (a keďže je posledná, aj na výstup

robotovho programu) uloží hodnotu 1 ak nastal niektorý z dvoch vyššie otestovaných prípadov.

Obmedzenia

- $1 \leq h \leq 200$
- $1 \leq w \leq 200$
- $2 \leq hw$
- $1 \leq k \leq h + w - 2$

Podúlohy

1. (10 bodov) $\max(h, w) \leq 3$
2. (11 bodov) $\max(h, w) \leq 10$
3. (11 bodov) $\max(h, w) \leq 30$
4. (15 bodov) $\max(h, w) \leq 100$
5. (12 bodov) $\min(h, w) = 1$
6. (8 bodov) V každej testovacej fotke je pixel v riadku 0 a stĺpci 0 čierny.
7. (14 bodov) $k = 1$
8. (19 bodov) Bez ďalších obmedzení.

Ukážkový grader

Ukážkový grader očakáva vstup v nasledovnom tvare:

- riadok 1: h w k
- riadok $2 + i$ ($i \geq 0$): $r_1[i]$ $c_1[i]$ $r_2[i]$ $c_2[i]$
- posledný riadok: -1

Každý riadok okrem prvého a posledného popisuje jednu fotku. Fotka popísaná riadkom $2 + i$ má číslo i a obsahuje presne dva čierne pixely: jeden v riadku $r_1[i]$ a stĺpci $c_1[i]$, druhý v riadku $r_2[i]$ a stĺpci $c_2[i]$.

Ukážkový grader začne tým, že spustí vašu procedúru pre h, w a k a uloží si ňou oznámený program. Ak počas toho nastala chyba, oznámi vám jednu z vyššie popísaných chybových hlášok a skončí.

Ak vaša procedúra úspešne dobehne, ukážkový grader postupne spustí uložený robotov program na fotkách z vášho vstupu a vyrobí dva výstupy:

Pre každú fotku vypíše ukážkový grader na štandardný výstup jeden riadok a v ňom číslo, ktoré je výstupom poslednej inštrukcie robotovho programu (1 alebo 0).

Okrem toho ukážkový grader vyrobí v aktuálnom adresári súbor `log.txt`. Postupne pre každú fotku doň vypíše jeden riadok nasledovného tvaru:

- $m[i][0]$ $m[i][1]$ $\ldots$ $m[i][c-1]$

Hodnoty $m[i][j]$ predstavujú obsah nastavených buniek pamäte robota v okamihu, keď skončil výpočet programu pre fotku i. Hodnota c (t.j. počet bitov v každom riadku logu) je rovná $hw + x$, kde x je počet inštrukcií v programe robota.