Robota redze

Jūs implementējat robota redzes programmu. Katru reizi, kad robota kamera uzņem fotogrāfiju, tā kā melnbalts attēls tiek saglabāta robota atmiņā. Katrs attēls ir $H \times W$ pikseļu režģis, kur rindas ir numurētas no 0 līdz $H-1$ un kolonnas ir numurētas no 0 līdz $W-1$. Katrā attēlā ir **tieši divi** melni pikseļi, bet visi pārējie pikseļi ir balti.

Robots var apstrādāt katru attēlu ar programmu, kas sastāv no vienkāršām instrukcijām. Jums ir dotas H un W vērtības, kā arī naturāls skaitlis K. Jūsu uzdevums ir uzrakstīt funkciju, kas ģenerē tādu robota programmu, kas katram no dotajiem attēliem nosaka, vai **attālums** starp diviem melnajiem pikseļiem ir tieši K. Attālums starp pikseli rindā r_1 un kolonnā c_1 un pikseli rindā r_2 un kolonnā c_2 ir $|r_1 - r_2| + |c_1 - c_2|$. Šajā formulā $|x|$ apzīmē x absolūto vērtību, kas ir vienāda ar x, ja $x \geq 0$, vai ar $-x$, ja $x < 0$.

Tagad aprakstīsim, kā robots strādā.

Robota atmiņa ir pietiekami liels šūnu masīvs, kas numurēts no 0. Katras šūnas vērtība var būt vai nu 0, vai 1, turklāt šūnas vērtība pēc uzstādīšanas nevar tikt mainīta. Attēls tiek saglabāts rindu pēc rindas šūnās, kas numurētas no 0 līdz $H \cdot W - 1$. Pirmā rinda tiek saglabāta šūnās, kas numurētas no 0 līdz $W-1$, bet pēdējā rinda — šūnās no $(H-1) \cdot W$ līdz $H \cdot W - 1$. Ja pikselis i-tās rindas j-tajā kolonnā ir melns, tad šūnas $i \cdot W + j$ vērtība ir 1. Pretējā gadījumā šūnas vērtība ir 0.

Robota programma ir **instrukciju** virkne, kas ir numurēta ar veseliem skaitļiem pēc kārtas, sākot no 0. Kad programma tiek palaista, instrukcijas tiek izpildītas secīgi pa vienai. Katra instrukcija nolasa vienas vai vairāku šūnu vērtības (sauksim tās par instrukcijas **ievadiem**) un ģenerē vienu vērtību, kas vienāda vai nu ar 0, vai ar 1 (sauksim to par instrukcijas **izvadu**). Instrukcijas i izvads tiek saglabāts šūnā ar numuru $H \cdot W + i$. Instrukcijas i ievadi var būt vērtības tikai no tām šūnām, kurās glabājas vai nu attēla pikseļu vērtības, vai arī iepriekšējo instrukciju izvadi, t.i., no šūnām ar numuriem no 0 līdz $H \cdot W + i - 1$.

Ir četru tipu instrukcijas:

- `NOT`: ir tieši viens ievads. Tās izvads ir 1 tad un tikai tad, ja ievads ir 0.
- `AND`: ir viens vai vairāki ievadi. Tās izvads ir 1 tad un tikai tad, ja **visi** ievadi ir 1.
- `OR`: ir viens vai vairāki ievadi. Tās izvads ir 1 tad un tikai tad, ja **vismaz viens** ievads ir 1.
- `XOR`: ir viens vai vairāki ievadi. Tās izvads ir 1 tad un tikai tad, ja **nepāra** skaits

ievadu ir 1.

Programmas pēdējās instrukcijas izvadam ir jābūt 1, ja attālums starp diviem melnajiem pikseļiem ir K, un 0 pretējā gadījumā.

Implementēšanas detaļas

Jums ir jāuzraksta sekojoša funkcija:

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

- H, W: attēla, kas iegūts ar robota kameru, izmēri
- K: naturāls skaitlis
- Šai funkcijai ir jāuzģenerē robota programma. Katram attēlam, kas iegūts ar robota kameru, programmai ir jānoskaidro, vai attālums starp diviem melnajiem pikseļiem ir precīzi K.

Šai funkcijai ir jāizsauc viena vai vairākas no sekojošajām funkcijām, lai robota programmai (kas sākotnēji ir tukša) tiktu pievienotas instrukcijas:

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

- Katra funkcija pievieno, attiecīgi, `NOT`, `AND`, `OR` vai `XOR` instrukciju.
- N (funkcijai `add_not`): šūnas, no kuras pievienota `NOT` instrukcija nolasa ievadu, pozīcija.
- Ns (funkcijai `add_and`, `add_or`, `add_xor`): šūnu, no kurām pievienota `AND`, `OR` vai `XOR` instrukcija nolasa ievadus, pozīciju masīvs.
- Katra funkcija atgriež šūnas, kura saturēs instrukcijas izvadu, pozīciju. Secīgi funkcijas izsaukumi atgriež secīgus veselus skaitļus, sākot no $H \cdot W$.

Robota programma var sastāvēt no ne vairāk kā $10\,000$ instrukcijām. Visas instrukcijas kopumā var nolasīt ne vairāk kā $1\,000\,000$ vērtības. Citiem vārdiem, summārais Ns masīvu garums visos funkciju `add_and`, `add_or` un `add_xor` izsaukumos plus funkcijas `add_not` izsaukumu skaits nepārsniedz $1\,000\,000$.

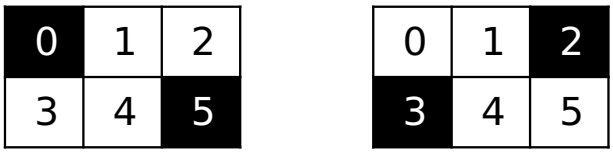
Pēc pēdējās instrukcijas pievienošanas, procedūrai `construct_network` ir jābeidz darbība. Tad jūsu robota programma tiks vērtēta, izmantojot vairākus attēlus. Jūsu risinājums tiks pieņemts, ja katram no šiem attēliem pēdējās instrukcijas izvads ir 1 tad un tikai tad, ja attālums starp diviem melnajiem pikseļiem ir vienāds ar K.

Jūsu risinājuma vērtēšana var beigties ar vienu no sekojošiem kļūdas paziņojumiem:

- `Instruction with no inputs`: funkcijai `add_and`, `add_or` vai `add_xor` kā ievads tika padots tukšs masīvs.
- `Invalid index`: funkcijai `add_and`, `add_or`, `add_xor` vai `add_not` kā ievads padota nekorekta (iespējams, negatīva) šūnas pozīcija.
- `Too many instructions`: jūsu funkcija ir mēģinājusi pievienot vairāk nekā $10\,000$ instrukcijas.
- `Too many inputs`: instrukcijas kopumā nolasa vairāk nekā $1\,000\,000$ vērtības.

Piemērs

Aplūkosim $H = 2$, $W = 3$, $K = 3$. Ir iespējami tikai divi attēli, kuros attālums starp melnajiem pikseļiem ir 3.



- 1. variants: melnie pikseļi ir 0 un 5
- 2. variants: melnie pikseļi ir 2 un 3

Viens iespējamais atrisinājums ir izveidot robota programmu, izpildot sekojošus funkciju izsaukumus:

1. `add_and([0, 5])`, kas pievieno instrukciju, kura izvada 1 tad un tikai tad, ja attēls atbilst 1. variantam. Izvads tiks saglabāts šūnā ar pozīciju 6.
2. `add_and([2, 3])`, kas pievieno instrukciju, kura izvada 1 tad un tikai tad, ja attēls atbilst 2. variantam. Izvads tiks saglabāts šūnā ar pozīciju 7.
3. `add_or([6, 7])`, kas pievieno instrukciju, kura izvada 1 tad un tikai tad, ja attēls atbilst vienam no diviem variantiem.

Ierobežojumi

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

Apakšuzdevumi

1. (10 punkti) $\max(H, W) \leq 3$
2. (11 punkti) $\max(H, W) \leq 10$
3. (11 punkti) $\max(H, W) \leq 30$
4. (15 punkti) $\max(H, W) \leq 100$

5. (12 punkti) $\min(H, W) = 1$
6. (8 punkti) Katrā attēlā pikselis 0-tās rindas 0-tajā kolonnā ir melns.
7. (14 punkti) $K = 1$
8. (19 punkti) Bez papildu ierobežojumiem.

Paraugvērtētājs

Paraugvērtētājs ielasa datus sekojošā formātā:

- 1. rinda: $H \;\; W \;\; K$
- $(2+i)$-tā rinda (visiem $i \geq 0$): $r_1[i] \;\; c_1[i] \;\; r_2[i] \;\; c_2[i]$
- pēdējā rinda (datu beigu pazīme): -1

Katra rinda, izņemot pirmo un pēdējo, atbilst attēlam ar diviem melniem pikseļiem. i-tā attēla apraksts ir dots $(2+i)$-tajā rindā: viens melnais pikselis atrodas rindā $r_1[i]$ un kolonnā $c_1[i]$, bet otrs — rindā $r_2[i]$ un kolonnā $c_2[i]$.

Paraugvērtētājs sākumā izsauc `construct_network(H, W, K)`. Ja `construct_network` pārkāpj kādu no uzdevuma nosacījumā aprakstītajiem ierobežojumiem, paraugvērtētājs izvada vienu no kļūdas paziņojumiem, kas ir aprakstīti «Implementēšanas detaļas» nodaļas beigās, un beidz darbu.

Citādi paraugvērtētājs izdod divus rezultātus.

Pirmkārt, paraugvērtētājs izvada robota programmas izvaddatus sekojošā formātā:

- $(1+i)$-tā rinda (visiem $0 \leq i$): robotu programmas pēdējās instrukcijas izvads i-tajam attēlam (1 vai 0).

Otrkārt, paraugvērtētājs izveido failu `log.txt` tekošajā darba katalogā sekojošā formātā:

- $(1+i)$-tā rinda (visiem $0 \leq i$): $m[i][0] \;\; m[i][1] \;\; \ldots \;\; m[i][c-1]$

$(1+i)$-tajā rindā izvadītā virkne apraksta vērtības, kas ir saglabātas robota atmiņas šūnās pēc tam, kad robota programma ir palaista i-tajam attēlam. Precīzāk, $m[i][j]$ satur šūnas j vērtību. Ievērojiet, ka c vērtība (virknes garums) ir vienāda ar $H \cdot W$ plus robota programmas instrukciju skaits.