Rozpoznawanie obrazu

Implementujesz oprogramowanie do rozpoznawania obrazu dla robota. Za każdym razem, gdy robot robi zdjęcie kamerą, jest ono zapisywane jako czarno-biały obraz w pamięci robota. Każdy obraz jest siatką $H \times W$ piksli, której wiersze są numerowane od 0 do $H-1$, a kolumny od 0 do $W-1$. Na każdym obrazie są **dokładnie dwa** czarne piksle, a wszystkie pozostałe piksle są białe.

Robot przetwarza każdy obraz za pomocą programu złożonego z prostych instrukcji. Dane są wartości H, W oraz dodatnia liczba całkowita K. Twoim celem jest napisanie funkcji generującej program dla robota, który dla dowolnego obrazu stwierdza, czy **odległość** między dwoma czarnymi pikslami jest równa dokładnie K. Odległość między pikslem w wierszu r_1 i kolumnie c_1 oraz pikslem w wierszu r_2 i kolumnie c_2 jest zdefiniowana jako $|r_1 - r_2| + |c_1 - c_2|$. W powyższym wzorze $|x|$ oznacza wartość bezwzględną x, czyli x jeśli $x \geq 0$ oraz $-x$ gdy $x < 0$.

Teraz opiszemy jak działa robot.

Pamięć robota to dostatecznie duża tablica komórek indeksowanych od 0. Każda komórka może przechowywać 0 lub 1, a jej zawartość po ustawieniu nie jest już nigdy zmieniana. Obraz jest przechowywany wiersz po wierszu w komórkach o indeksach od 0 do $H \cdot W - 1$. Pierwszy wiersz jest przechowywany w komórkach o indeksach od 0 do $W-1$, a ostatni wiersz w komórkach o numerach od $(H-1) \cdot W$ do $H \cdot W - 1$. Konkretnie, jeśli piksel w wierszu i oraz kolumnie j jest czarny, to w komórce o indeksie $i \cdot W + j$ jest przechowywane 1, w przeciwnym wypadku znajduje się tam 0.

Program robota jest ciągiem **instrukcji**, które są numerowane kolejnymi liczbami całkowitymi, zaczynając od 0. Po uruchomieniu programu, instrukcje są wykonywane jedna po drugiej. Każda instrukcja wczytuje zawartość jednej lub wielu komórek (wczytane wartości nazywamy **wejściami** instrukcji) i produkuje jedną wartość równą 0 lub 1 (którą nazywamy **wyjściem** instrukcji). Wyjście instrukcji i jest zapisywane w komórce $H \cdot W + i$. Wejście instrukcji i może pochodzić tylko z komórek, które opisują piksle lub przechowują wyjścia wcześniejszych instrukcji, to znaczy komórki od 0 do $H \cdot W + i - 1$.

Są cztery typy instrukcji:

- `NOT`: ma tylko jedno wejście. Jej wyjście to 1 dla wejścia równego 0, a w przeciwnym przypadku 0.
- `AND`: ma jedno lub więcej wejść. Jej wyjście to 1 wtedy i tylko wtedy, gdy

wszystkie wejścia to 1.
- `OR`: ma jedno lub więcej wejść. Jej wyjście to 1 wtedy i tylko wtedy, gdy **przynajmniej jedno** wejście to 1.
- `XOR`: ma jedno lub więcej wejść. Jej wyjście to 1 wtedy i tylko wtedy, gdy **nieparzyście wiele** wejść to 1.

Wyjściem ostatniej instrukcji programu powinno być 1, gdy odległość między dwoma czarnymi pikslami to dokładnie K, a w przeciwnym przypadku 0.

Szczegóły implementacyjne

Twoim zadaniem jest zaimplementowanie następującej procedury:

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

- H, W: wymiary każdego obrazu pochodzącego z kamery robota
- K: dodatnia liczba całkowita
- Procedura powinna wygenerować program robota. Dla każdego obrazu pochodzącego z kamery robota, program powinien określić, czy odległość między dwoma czarnymi pikslami na tym obrazie to dokładnie K.

Ta procedura powinna wywoływać jedną lub więcej z poniższych funkcji, aby dopisywać instrukcje do programu robota (który początkowo jest pusty):

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

- Dopisuje (odpowiednio) instrukcję `NOT`, `AND`, `OR` lub `XOR`.
- N (dla funkcji `add_not`): indeks komórki, z której dopisywana instrukcja `NOT` wczytuje wejście
- Ns (dla funkcji `add_and`, `add_or`, `add_xor`): tablica zawierająca indeksy komórek, z których dopisywana instrukcja `AND`, `OR` lub `XOR` wczytuje swoje wejścia
- Wynikiem działania każdej funkcji jest indeks komórki, która przechowuje wyjście danej instrukcji. Wynikami kolejnych wywołań tych funkcji są kolejne liczby całkowite, zaczynając od $H \cdot W$.

Program robota może się składać z co najwyżej $10\,000$ instrukcji. Sumaryczna liczba wartości odczytanych przez wszystkie instrukcje nie może przekroczyć $1\,000\,000$. Mówiąc inaczej, sumaryczna długość tablic Ns we wszystkich wywołaniach funkcji `add_and`, `add_or` oraz `add_xor` plus liczba wywołań funkcji `add_not` nie może przekroczyć $1\,000\,000$.

Procedura `construct_network` powinna zakończyć swoje działanie po wypisaniu

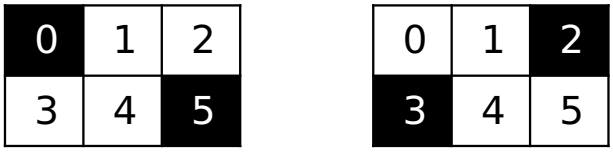
ostatniej instrukcji. Następnie program robota będzie sprawdzony na pewnej liczbie obrazów. Zestaw testowy zostanie uznany za prawidłowo rozwiązany, jeśli dla każdego z tych obrazów wyjście ostatniej instrukcji to 1 wtedy i tylko wtedy, gdy odległość między dwoma czarnymi pikslami jest równa K.

Efektem oceniania Twojego rozwiązania może być jeden z poniższych komunikatów w języku angielskim, których znaczenie jest wyjaśnione poniżej:

- `Instruction with no inputs`: argumentem funkcji `add_and`, `add_or` lub `add_xor` była pusta tablica.
- `Invalid index`: nieprawidłowy (być może ujemny) indeks komórki został podany w argumencie funkcji `add_and`, `add_or`, `add_xor` lub `add_not`.
- `Too many instructions`: Twoja procedura dopisała więcej niż 10 000 instrukcji.
- `Too many inputs`: sumaryczna liczba wartości odczytanych przez instrukcje przekracza 1 000 000.

Przykład

Załóżmy, że $H = 2$, $W = 3$, $K = 3$. Mamy tylko dwa możliwe obrazy, dla których odległość między czarnymi pikslami jest równa 3.



- Przypadek 1: czarne piksle to 0 oraz 5
- Przypadek 2: czarne piksle to 2 oraz 3

Jednym z możliwych rozwiązań jest zbudowanie programu robota za pomocą następujących wywołań:

1. `add_and([0, 5])` dodające instrukcję, której wyjściem jest 1 wtedy i tylko wtedy, gdy mamy do czynienia z Przypadkiem 1. Wyjście jest zapisywane w komórce 6.
2. `add_and([2, 3])` dodające instrukcję, której wyjściem jest 1 wtedy i tylko wtedy, gdy mamy do czynienia z Przypadkiem 2. Wyjście jest zapisywane w komórce 7.
3. `add_or([6, 7])` dodające instrukcję, której wyjściem jest 1 wtedy i tylko wtedy, gdy mamy do czynienia z Przypadkiem 1 lub 2.

Ograniczenia

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

Podzadania

1. (10 punktów) $\max(H, W) \leq 3$
2. (11 punktów) $\max(H, W) \leq 10$
3. (11 punktów) $\max(H, W) \leq 30$
4. (15 punktów) $\max(H, W) \leq 100$
5. (12 punktów) $\min(H, W) = 1$
6. (8 punktów) Na każdym obrazie piksel w wierszu 0 i kolumnie 0 jest czarny.
7. (14 punktów) $K = 1$
8. (19 punktów) Brak dodatkowych założeń.

Przykładowa sprawdzaczka

Przykładowa sprawdzaczka wczytuje wejście w następującym formacie:

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

Każdy wiersz za wyjątkiem pierwszego i ostatniego opisuje obraz zawierający dwa czarne piksle. Obraz opisany przez wiersz $2 + i$ nazywamy obrazem i. Jeden czarny piksel jest w wierszu $r_1[i]$ i kolumnie $c_1[i]$, podczas gdy drugi jest w wierszu $r_2[i]$ i kolumnie $c_2[i]$.

Przykładowa sprawdzaczka najpierw wywołuje procedurę `construct_network(H, W, K)`. Jeśli `construct_network(H, W, K)` narusza ograniczenia z treści zadania, przykładowa sprawdzaczka wypisuje jeden z komunikatów błędów wypisanych na końcu fragmentu poświęconego szczegółom implementacyjnym, a następnie kończy działanie.

W przeciwnym przypadku, efekt działania przykładowej sprawdzaczki jest dwojaki.

Po pierwsze, przykładowa sprawdzaczka wypisuje wyjście ostatniej instrukcji programu robota w następującym formacie:

- wiersz $1 + i$ $(0 \leq i)$: wyjście ostatniej instrukcji w programie robota uruchomionym dla obrazu i (1 lub 0).

Po drugie, przykładowa sprawdzaczka tworzy w bieżącym katalogu plik `log.txt`, którego format jest następujący:

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

Sekwencja podana w wierszu $1 + i$ opisuje wartości przechowywane w komórkach pamięci robota po wykonaniu programu robota na obrazie i. Konkretnie, $m[i][j]$ to wartość przechowywana w komórce j. Długość sekwencji c to $H \cdot W$ plus liczba

instrukcji w programie robota.