Riconoscimento visivo

Devi generare un circuito di riconoscimento visivo per un robot in grado di scattare fotografie in bianco e nero. Le fotografie vengono memorizzate come una griglia $H \times W$ di pixel (righe da 0 a $H-1$, colonne da 0 a $W-1$), e contengono sempre **esattamente due** pixel neri (tutti gli altri sono bianchi). Lo scopo del robot è capire se i due pixel neri sono *esattamente* a una certa distanza K data (la distanza tra il pixel (r_1, c_1) e il pixel (r_2, c_2) è pari a $|r_1 - r_2| + |c_1 - c_2|$).

Il circuito del robot deve consistere di una sequenza di semplici **componenti**, numerati a partire da 0, con cui deve poter processare ogni immagine. La memoria del robot è un array m di bit, che all'inizio del procedimento è impostata per gli indici da 0 a $H \cdot W - 1$, di modo che $m[r \cdot W + c] = 1$ se e solo se il pixel (r, c) è nero. I componenti agiscono in sequenza, per cui il componente i-esimo scrive il suo output in $m[H \cdot W + i]$, e riceve come input alcuni dei valori tra $m[0]$ e $m[H \cdot W + i - 1]$ secondo il disegno del circuito. Ci sono quattro tipi di componenti:

- `NOT`: ha esattamente un input. Il suo output è 1 se l'input è 0, altrimenti il suo output è 0.
- `AND`: ha uno o più input. Il suo output è 1 se **tutti** gli input sono 1.
- `OR`: ha uno o più input. Il suo output è 1 se **almeno un** input è 1.
- `XOR`: ha uno o più input. Il suo output è 1 se è presente un **numero dispari** di 1 negli input.

L'output dell'ultima istruzione dovrebbe essere 1 se la distanza tra i due pixel neri è esattamente K, oppure 0 altrimenti.

Dettagli di implementazione

Devi implementare la seguente funzione:

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

- H, W: le dimensioni delle fotografie scattate dal robot
- K: la distanza richiesta tra i due pixel neri
- Questa funzione dovrebbe generare il circuito del robot, di modo che sia in grado di determinare se la distanza tra i due pixel neri è esattamente K per ogni possibile immagine in input.

Per aggiungere in sequenza componenti al circuito (inizialmente vuoto), la tua funzione deve chiamare le seguenti funzioni fornite nel grader:

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

- Aggiunge un componente `NOT`, `AND`, `OR`, o `XOR` rispettivamente.
- N (per `add_not`): l'indice dell'unico bit che deve essere usato come input dal componente;
- Ns (per `add_and`, `add_or`, `add_xor`): un array che contiene gli indici dei bit che devono essere usati come input dal componente;
- Le funzioni restituiscono l'indice del bit memorizzato come output dal componente (e quindi interi consecutivi a partire da $H \cdot W$).

Il circuito del robot deve consistere di al massimo $10\,000$ componenti. La quantità totale di input di tutti i suoi componenti deve essere al massimo $1\,000\,000$: in altre parole, la somma delle lunghezze degli array Ns più il numero di chiamate ad `add_not` non deve superare $1\,000\,000$.

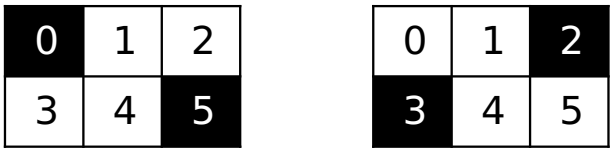
Dopo aver aggiunto l'ultimo componente, la funzione `construct_network` deve terminare la sua esecuzione. A quel punto, il tuo circuito verrà valutato su un certo numero di immagini. La tua soluzione risolverà un certo test case solo se per ognuna di queste immagini, l'output dell'ultima istruzione è 1 se e solo se la distanza tra i due pixel neri è uguale a K.

La valutazione della tua soluzione potrebbe riportare uno dei seguenti messaggi di errore:

- `Instruction with no inputs`: è stato passato un array vuoto come input ad un istruzione `add_and`, `add_or`, o `add_xor`.
- `Invalid index`: un indice scorretto (negativo o troppo grande) è stato passato tra gli input di `add_and`, `add_or`, `add_xor`, o `add_not`.
- `Too many instructions`: la tua funzione ha provato ad aggiungere più di $10\,000$ componenti.
- `Too many inputs`: i componenti inseriti leggono più di $1\,000\,000$ input in totale.

Esempio

Assumiamo che $H = 2$, $W = 3$, $K = 3$. Ci sono solo due possibili immagini in cui la distanza tra i due pixel neri è 3.



- Caso 1: i pixel neri sono 0 e 5.
- Caso 2: i pixel neri sono 2 e 3.

Per generare un circuito di riconoscimento, si possono quindi effettuare le seguenti chiamate:

1. `add_and([0, 5])`, per aggiungere un componente che restituisce 1 se siamo nel primo caso (0 altrimenti), memorizzandolo il risultato nell'indice 6.
2. `add_and([2, 3])`, per aggiungere un componente che restituisce 1 se siamo nel secondo caso (0 altrimenti), memorizzando il risultato nell'indice 7.
3. `add_or([6, 7])`, per aggiungere un componente che restituisce 1 se siamo in uno qualunque dei due casi.

Assunzioni

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

Subtask

1. (10 punti) $\max(H, W) \leq 3$.
2. (11 punti) $\max(H, W) \leq 10$.
3. (11 punti) $\max(H, W) \leq 30$.
4. (15 punti) $\max(H, W) \leq 100$.
5. (12 punti) $\min(H, W) = 1$.
6. (8 punti) Il pixel $(0, 0)$ è nero in ogni immagine.
7. (14 punti) $K = 1$.
8. (19 punti) Nessuna restrizione aggiuntiva.

Grader di esempio

Il grader di esempio legge l'input nel seguente formato:

- riga 1: H W K
- righe $2 + i$ $(i \geq 0)$: r_1 c_1 r_2 c_2
- ultima riga: -1

Ogni riga tranne la prima e l'ultima rappresenta un immagine con due pixel neri, per cui l'immagine i-esima è descritta nella riga $2 + i$ e contiene i pixel neri (r_1, c_1) e (r_2, c_2).

Il grader di esempio chiamerà la funzione `construct_network(H, W, K)`. Se questa viola le richieste, il grader stamperà uno dei messaggi di errore descritti precedentemente e terminerà. Altrimenti, il grader produrrà due output.

Innanzitutto, stamperà il risultato del programma del robot nel seguente formato:

- righe $1 + i$ $(0 \leq i)$: l'output dell'ultimo componente del circuito per l'immagine i-esima (1 or 0).

Inoltre, scriverà sul file `log.txt` (nella cartella corrente) nel seguente formato:

- righe $1 + i$ $(0 \leq i)$: $m[0]$ $m[1]$ $\ldots$ $m[L-1]$

La sequenza i-esima in riga $1 + i$ elenca i valori nella memoria del robot dopo l'esecuzione del circuito, data l'immagine i come input. Il numero di valori L (lunghezza della sequenza) è uguale ad $H \cdot W$ più il numero di componenti nel circuito del robot.