Programme de vision (Vision Program)

Vous implémentez un programme de vision pour un robot. Chaque fois que la caméra du robot enregistre une image, elle est stockée en noir et blanc dans la mémoire du robot. Chaque image est une grille de pixels de taille $H \times W$, dont les lignes sont numérotées de 0 à $H-1$, et dont les colonnes sont numérotées de 0 à $W-1$. Il y a **exactement deux** pixels noirs dans chaque image, et tous les autres pixels sont blancs.

Le robot traite chaque image avec un programme composé d'instructions élémentaires. On vous donne les valeurs de H, W, et un entier positif K. Votre but est d'écrire une fonction qui construit un programme pour le robot qui, pour chaque image, détermine si la **distance** entre les deux pixels noirs est exactement K. Ici, la distance entre un pixel à la ligne r_1, colonne c_1 et un pixel à la ligne r_2, colonne c_2 est $|r_1 - r_2| + |c_1 - c_2|$. Dans cette formule, $|x|$ est la valeur absolue de x, égale à x si $x \geq 0$ et à $-x$ si $x < 0$.

Nous décrivons maintenant le fonctionnement du robot.

La mémoire du robot est un tableau suffisamment grand de cases mémoire, indexé à partir de 0. Chaque case mémoire peut contenir soit 0, soit 1, et sa valeur ne peut plus être changée après avoir été écrite. L'image est stockée ligne par ligne dans les cases mémoire d'indices 0 à $H \cdot W - 1$. La première ligne est stockée dans les cases mémoire de 0 à $W-1$, et la dernière ligne est stockée dans les cases mémoire de $(H-1) \cdot W$ à $H \cdot W - 1$. En particulier, si le pixel à la ligne i, colonne j est noir, la valeur de la case mémoire $i \cdot W + j$ est 1, sinon cette valeur est 0.

Un programme du robot est une séquence d'**instructions**, qui sont numérotées à partir de 0. Lors de l'exécution du programme, les instructions sont exécutées les unes après les autres. Chaque instruction lit la valeur d'une ou plusieurs cases mémoire (ces valeurs sont appelées les **entrées** de l'instruction), et produit une unique valeur égale à 0 ou 1 (cette valeur est appelée la **sortie** de l'instruction). La sortie de l'instruction i est stockée dans la case mémoire $H \cdot W + i$. Les entrées de l'instruction i ne peuvent être que des pixels de l'image, ou des sorties des instructions précédentes, soit les cases mémoires 0 à $H \cdot W + i - 1$.

Il y a quatre types d'instructions :

- `NOT` : prend exactement une entrée. Sa sortie est 1 si l'entrée est 0, sinon sa sortie est 0.

- `AND` : prend une ou plusieurs entrées. Sa sortie est 1 si est seulement si **toutes** les entrées sont égales à 1.
- `OR` : prend une ou plusieurs entrées. Sa sortie est 1 si est seulement si **au moins une** des entrées est égale à 1.
- `XOR` : prend une ou plusieurs entrées. Sa sortie est 1 si est seulement si **un nombre impair** d'entrées sont égales à 1.

La sortie de la dernière instruction du programme doit être 1 si la distance entre les deux pixels noirs est exactement K, et 0 sinon.

Détails d'implémentation

Vous devez implémenter la fonction suivante :

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

- H, W : les dimensions de chaque image capturée par la caméra du robot
- K : un entier positif
- Cette fonction doit produire un programme pour le robot. Pour chaque image capturée par la caméra du robot, ce programe doit déterminer si la distance entre deux pixels noirs de l'image est exactement K.

Cette fonction doit appeler une ou plusieurs des fonctions suivantes pour ajouter des instructions au programme du robot (qui est initialement vide) :

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

- Ajouter une instruction `NOT`, `AND`, `OR` ou `XOR`, respectivement.
- N (pour `add_not`) : l'index de la case mémoire où l'instruction `NOT` ajoutée lit son entrée.
- Ns (pour `add_and`, `add_or`, `add_xor`) : un tableau contenant les indices des cases mémoire où l'instruction `AND`, `OR` ou `XOR` lit son entrée.
- Chaque instruction renvoie l'indice de la case mémoire où est stockée la sortie de l'instruction. Les appels consécutifs de ces fonctions renvoient des entiers consécutifs commençant à $H \cdot W$.

Le programme du robot peut contenir au plus $10\,000$ instructions. Les instructions peuvent lire au plus $1\,000\,000$ valeurs au total. Autrement dit, la longueur totale des tableaux Ns dans les appels à `add_and`, `add_or` et `add_xor`, additionnée au nombre d'appels à `add_not`, ne doit pas dépasser $1\,000\,000$.

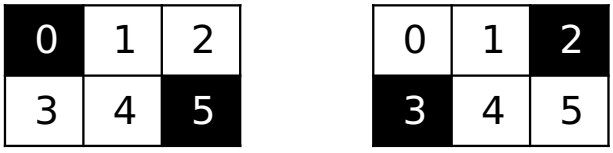
Après l'ajout de la dernière instruction, la fonction `construct_network` doit terminer. Le programme du robot sera ensuite évalué sur un certain nombre d'images. Votre solution passe le cas de test si pour chacune de ces images, la sortie de la dernière instruction est 1 si et seulement si la distance entre les deux pixels noirs de l'image est égale à K.

L'évaluation de votre solution peut produire des messages en anglais, expliqués ci-dessous:

- `Instruction with no inputs` : un tableau vide a été donné en entréé à `add_and`, `add_or` ou `add_xor`.
- `Invalid index` : un indice de case mémoire invalide (possiblement négatif) a été fourni en entrée à `add_and`, `add_or`, `add_xor` ou `add_not`.
- `Too many instructions` : votre fonction a essayé d'ajouter plus de $10\,000$ instructions.
- `Too many inputs` : les instructions de votre programme lisent plus de $1\,000\,000$ valeur au total.

Example

Prenons le cas $H = 2$, $W = 3$, $K = 3$. Il n'y a que deux images possibles pour lesquelles la distance entre les deux pixels noirs est de 3.



- Cas 1: les pixels noirs sont 0 et 5
- Cas 2: les pixels noirs sont 2 et 3

Une solution possible est de construire un programme pour le robot en faisant les appels de fonction suivants :

1. `add_and([0, 5])`, qui ajoute une instruction qui a pour sortie 1 si et seulement si on se trouve dans le cas 1. La sortie est stockée dans la case 6.
2. `add_and([2, 3])`, qui ajoute une instruction qui a pour sortie 1 si et seulement si on se trouve dans le cas 2. La sortie est stockée dans la case 7.
3. `add_or([6, 7])`, qui ajoute une instruction qui a pour sortie 1 si et seulement si on se trouve dans un des deux cas ci-dessus.

Contraintes

- $1 \leq H \leq 200$

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

Sous-tâches

1. (10 points) $\max(H, W) \leq 3$
2. (11 points) $\max(H, W) \leq 10$
3. (11 points) $\max(H, W) \leq 30$
4. (15 points) $\max(H, W) \leq 100$
5. (12 points) $\min(H, W) = 1$
6. (8 points) Le pixel à la ligne 0, colonne 0 est noir dans toutes les images.
7. (14 points) $K = 1$
8. (19 points) Aucune contrainte supplémentaire.

Évaluateur d'exemple

L'évaluateur d'exemple lit l'entrée dans le format suivant :

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

Chaque ligne excepté la première et la dernière représente une image avec deux pixels noirs. L'image décrite à la ligne $2 + i$ est appelée image i. L'un des pixels noirs se trouve à la ligne $r_1[i]$, colonne $c_1[i]$ et l'autre se trouve à la ligne $r_2[i]$, colonne $c_2[i]$.

L'évaluateur d'exemple commence par appeler `construct_network(H, W, K)`. Si `construct_network` enfreint une des contraintes listées dans le sujet, l'évaluateur d'exemple affiche un des messages d'erreur listés à la fin de la section Détails d'implémentation et s'arrête.

Sinon, l'évaluateur d'exemple produit deux sorties.

Premièrement, l'évaluateur d'exemple affiche la sortie du programme du robot dans le format suivant :

- ligne $1 + i$ $(0 \leq i)$: sortie de la dernière instruction du programme du robot sur l'image i (1 ou 0).

Deuxièmement, l'évaluateur d'exemple écrit un ficher `log.txt` dans le dossier courant, dans le format suivant :

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

La séquence de la ligne $1 + i$ décrit les valeurs stockées dans les cases mémoire du

robot après l'éxecution du programme du robot avec l'image i comme entrée. Précisément, $m[i][j]$ est la valeur de la case mémoire j. Notez que la valeur de c (la longueur de la séquence) est égale à $H \cdot W$, plus le nombre d'instructions du programme du robot.