Πρόγραμμα όρασης

Υλοποιείτε ένα πρόγραμμα όρασης για ένα ρομπότ. Κάθε φορά που η κάμερα του ρομπότ βγάζει μία φωτογραφία, αυτή αποθηκεύεται στη μνήμη του ρομπότ ως μία ασπρόμαυρη εικόνα. Κάθε εικόνα είναι ένα πλέγμα από $H \times W$ pixels, στο οποίο οι γραμμές είναι αριθμημένες από 0 έως $H - 1$ και οι στήλες από 0 έως $W - 1$. Υπάρχουν **ακριβώς δύο** μαύρα pixels σε κάθε εικόνα, όλα τα υπόλοιπα pixels είναι άσπρα.

Το ρομπότ επεξεργάζεται κάθε εικόνα με ένα πρόγραμμα αποτελούμενο από απλές εντολές. Δίνονται οι τιμές H, W και ένας θετικός ακέραιος K. Ο στόχος σας είναι να γράψετε μία συνάρτηση που να παράγει ένα πρόγραμμα όρασης για το ρόμποτ, το οποίο για κάθε εικόνα να αποφασίζει αν η **απόσταση** ανάμεσα στα δύο μαύρα pixels είναι ακριβώς K. Η απόσταση ανάμεσα σε ένα pixel στη γραμμή r_1 και στήλη c_1 και ένα pixel στη γραμμή r_2 και στήλη c_2 ορίζεται ως $|r_1 - r_2| + |c_1 - c_2|$, όπου με $|x|$ συμβολίζεται η απόλυτη τιμή του x, που ισούται με x αν $x \geq 0$ και με $-x$ αν $x < 0$.

Στη συνέχεια θα περιγράψουμε πώς δουλεύει το ρομπότ.

Η μνήμη του ρομπότ είναι ένας επαρκώς μεγάλος πίνακας από θέσεις μνήμης, στον οποίο η αρίθμηση των θέσεων ξεκινάει από το 0. Σε κάθε θέση μνήμης μπορεί να αποθηκευτεί είτε η τιμή 0 είτε η τιμή 1 και η τιμή που θα αποθηκευτεί δεν μπορεί να αλλάξει. Η εικόνα αποθηκεύεται γραμμή-γραμμή στις θέσεις μνήμης από 0 έως $H \cdot W - 1$. Η πρώτη γραμμή αποθηκεύεται στις θέσεις μνήμης 0 έως $W - 1$ και η τελευταία στις θέσεις μνήμης $(H - 1) \cdot W$ έως $H \cdot W - 1$. Συγκεκριμένα, αν το pixel στη γραμμή i και στήλη j είναι μαύρο, η τιμή της θέσης μνήμης $i \cdot W + j$ είναι 1, διαφορετικά είναι 0.

Το πρόγραμμα του ρομπότ είναι μία ακολουθία από **εντολές**, αριθμημένες με διαδοχικούς ακεραίους αριθμούς ξεκινώντας από το 0. Όταν το πρόγραμμα τρέχει, οι εντολές εκτελούνται μία-μία. Κάθε εντολή διαβάζει τις τιμές μίας ή περισσότερων θέσεων μνήμης (ονομάζουμε αυτές τις τιμές **εισόδους** της εντολής) και παράγει μία ακριβώς τιμή ίση με 0 ή 1 (ονομάζουμε αυτήν την τιμή **έξοδο** της εντολής). Η έξοδος της εντολής i αποθηκεύεται στη θέση μνήμης $H \cdot W + i$. Οι είσοδοι της εντολής i μπορούν να είναι θέσεις μνήμης στις οποίες είναι αποθηκευμένα είτε τα pixels της εικόνας είτε οι εξόδοι των προηγούμενων εντολών, δηλαδή όλες οι θέσεις μνήμης από 0 έως $H \cdot W + i - 1$.

Υπάρχουν τέσσερα είδη εντολών:

- `NOT`: έχει ακριβώς μία είσοδο. Η έξοδος είναι 1 αν η είσοδος είναι 0, διαφορετικά η έξοδος είναι 0.

- `AND`: έχει μία ή περισσότερες εισόδους. Η έξοδος είναι 1 αν και μόνο αν **όλες** οι είσοδοι είναι 1.
- `OR`: έχει μία ή περισσότερες εισόδους. Η έξοδος είναι 1 αν και μόνο αν **τουλάχιστον μία** από τις εισόδους είναι 1.
- `XOR`: έχει μία ή περισσότερες εισόδους. Η έξοδος είναι 1 αν και μόνο αν **περιττό πλήθος** των εισόδων είναι 1.

Η έξοδος της τελευταίας εντολής του προγράμματος πρέπει να είναι 1 αν η απόσταση ανάμεσα στα δύο μαύρα pixels είναι ακριβώς K, διαφορετικά 0.

Λεπτομέρειες υλοποίησης

Πρέπει να υλοποιήσετε την ακόλουθη συνάρτηση:

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

- H, W: διαστάσεις της κάθε εικόνας που παίρνει η κάμερα του ρομπότ
- K: ένας θετικός ακέραιος
- Η συνάρτηση πρέπει να παράγει ένα πρόγραμμα όρασης για το ρομπότ. Για κάθε εικόνα που παίρνει η κάμερα του ρομπότ, αυτό το πρόγραμμα όρασης πρέπει να αποφασίζει αν η απόσταση ανάμεσα στα δύο μαύρα pixels της εικόνας είναι ακριβώς K.

Η συνάρτηση σας πρέπει να καλεί μία ή περισσότερες από τις παρακάτω συναρτήσεις για να προσθέτει εντολές στο τέλος του προγράμματος όρασης του ρομπότ (το οποίο αρχικά είναι κενό):

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

- Πρόσθεσε μία εντολή `NOT`, `AND`, `OR` ή `XOR` στο τέλος του προγράμματος όρασης, αντίστοιχα.
- N (για την `add_not`): ο αριθμός της θέσης μνήμης από την οποία η εντολή `NOT` διαβάζει την είσοδό της
- Ns (για τις `add_and`, `add_or`, `add_xor`): ένας πίνακας που περιέχει τους αριθμούς των θέσεων μνήμης από τις οποίες η εντολή `AND`, `OR` ή `XOR` διαβάζει τις εισόδους της
- Κάθε συνάρτηση επιστρέφει τον αριθμό της θέσης μνήμης όπου αποθηκεύεται η έξοδος της εντολής. Διαδοχικές κλήσεις αυτών των συναρτήσεων θα επιστρέφουν διαδοχικούς ακεραίους ξεκινώντας από το $H \cdot W$.

Το πρόγραμμα του ρομπότ μπορεί να αποτελείται το πολύ από $10\,000$ εντολές. Οι

εντολές μπορούν να διαβάζουν το πολύ $1\,000\,000$ τιμές συνολικά. Με άλλα λόγια, το συνολικό μέγεθος των πινάκων Ns όλων των κλήσεων στις `add_and`, `add_or` και `add_xor` συν το πλήθος των κλήσεων στην `add_not` δεν πρέπει να ξεπερνάει το $1\,000\,000$.

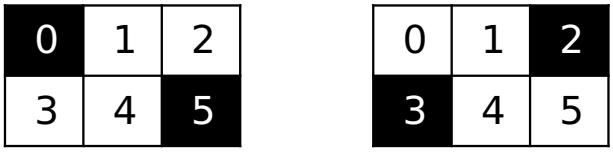
Αφού προσθέσετε την τελευταία σας εντολή, η συνάρτηση `construct_network` πρέπει να επιστρέφει (return). Το πρόγραμμα του ρομπότ θα αξιολογηθεί με ένα πλήθος εικόνων. Η λύση σας περνάει ένα συγκεκριμένο test case, αν ισχύει το εξής: για κάθε μία από τις εικόνες η έξοδος της τελευταίας εντολής είναι 1 αν και μόνο αν η απόσταση μεταξύ των δύο μαύρων pixels της εικόνας είναι ίση με K.

Η βαθμολόγηση της λύσης μπορεί να έχει ως αποτέλεσμα κάποιο από τα παρακάτω μηνύματα σφάλματος στα αγγλικά:

- `Instruction with no inputs`: δόθηκε κενός πίνακας ως είσοδος σε κάποια από τις `add_and`, `add_or` ή `add_xor`.
- `Invalid index`: δόθηκε λανθασμένος (πιθανώς αρνητικός) αριθμός θέσης μνήμης ως είσοδος σε κάποια από τις `add_and`, `add_or`, `add_xor` ή `add_not`.
- `Too many instructions`: η συνάρτησή σας προσπάθησε να προσθέσει περισσότερες από $10\,000$ εντολές.
- `Too many inputs`: οι εντολές του προγράμματος του ρομπότ διαβάζουν συνολικά περισσότερες από $1\,000\,000$ τιμές.

Παράδειγμα

Έστω $H = 2,\ W = 3,\ K = 3$. Υπάρχουν μόνο δύο δυνατές εικόνες στις οποίες η απόσταση μεταξύ των μαύρων pixels είναι 3.



- Περίπτωση 1: τα μαύρα pixels είναι τα 0 και 5
- Περίπτωση 2: τα μαύρα pixels είναι τα 2 και 3

Μία δυνατή λύση είναι να κατασκευαστεί ένα πρόγραμμα για το ρομπότ κάνοντας τις παρακάτω κλήσεις:

1. `add_and([0, 5])`, προσθέτει μία εντολή με έξοδο 1 αν και μόνο αν βρισκόμαστε στην πρώτη περίπτωση. Η έξοδος αποθηκεύεται στη θέση 6.
2. `add_and([2, 3])`, προσθέτει μία εντολή με έξοδο 1 αν και μόνο αν βρισκόμαστε στη δεύτερη περίπτωση. Η έξοδος αποθηκεύεται στη θέση 7.
3. `add_or([6, 7])`, προσθέτει μία εντολή με έξοδο 1 αν και μόνο αν βρισκόμαστε σε οποιαδήποτε από αυτές τις δύο περιπτώσεις.

Περιορισμοί

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

Υποπροβλήματα

1. (10 βαθμοί) $\max(H, W) \leq 3$
2. (11 βαθμοί) $\max(H, W) \leq 10$
3. (11 βαθμοί) $\max(H, W) \leq 30$
4. (15 βαθμοί) $\max(H, W) \leq 100$
5. (12 βαθμοί) $\min(H, W) = 1$
6. (8 βαθμοί) Το pixel στη γραμμή 0 και τη στήλη 0 είναι μαύρο σε κάθε εικόνα.
7. (14 βαθμοί) $K = 1$
8. (19 βαθμοί) Κανένας επιπλέον περιορισμός.

Υποδειγματικός βαθμολογητής

Ο υποδειγματικός βαθμολογητής διαβάζει την είσοδο ως εξής:

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

Κάθε γραμμή εκτός από την πρώτη και την τελευταία αναπαριστούν μία εικόνα με δύο μαύρα pixels. Θα συμβολίζουμε την εικόνα που βρίσκεται στη γραμμή $2 + i$ ως εικόνα i. Το ένα μαύρο pixel θα βρίσκεται στη γραμμή $r_1[i]$ και τη στήλη $c_1[i]$ και το δεύτερο μαύρο pixel θα βρίσκεται στη γραμμή $r_2[i]$ και τη στήλη $c_2[i]$.

Ο υποδειγματικός βαθμολογητής πρώτα κάνει την κλήση `construct_network(H, W, K)`. Αν η `construct_network` παραβιάζει κάποιους από τους περιορισμούς που αναφέρονται στην εκφώνηση του προβλήματος, τότε ο υποδειγματικός βαθμολογητής τυπώνει ένα από τα μηνύματα σφάλματος που αναφέρονται στο τέλος των λεπτομερειών υλοποίησης και τερματίζει.

Διαφορετικά, ο υποδειγματικός βαθμολογητής παράγει δύο αποτελέσματα.

Πρώτον, τυπώνει την έξοδο του προγράμματος του ρομπότ ως εξής:

- γραμμή $1 + i$ ($0 \leq i$): η έξοδος (1 ή 0) της τελευταίας εντολής του προγράμματος του ρομπότ για την εικόνα i.

Δεύτερον, γράφει ένα αρχείο `log.txt` στο τρέχον directory που θα περιέχει τα εξής:

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

Η ακολουθία αριθμών (1 ή 0) στη γραμμή $1 + i$ θα περιγράφει τις τιμές που είναι αποθηκευμένες στις θέσεις μνήμης του ρομπότ μετά την εκτέλεση του προγράμματος, αν η είσοδος είναι η εικόνα i. Ειδικότερα, το $m[i][j]$ θα είναι η τιμή της θέσης j. Προσέξτε ότι η τιμή του c (το μήκος της ακολουθίας αριθμών) θα είναι ίση με $H \cdot W$ συν το πλήθος των εντολών του προγράμματος του ρομπότ.