Καταχωρητές (registers)

Ο μηχανικός Χριστόφορος πειραματίζεται με ένα νέο τύπο υπολογιστικού επεξεργαστή.

Ο επεξεργαστής έχει πρόσβαση σε m διαφορετικές θέσεις μνήμης των b bits (όπου $m = 100$ και $b = 2000$), οι οποίες ονομάζονται **καταχωρητές**, και είναι αριθμημένες από 0 μέχρι $m - 1$. Συμβολίζουμε τους καταχωρητές με $r[0], r[1], \ldots, r[m-1]$. Κάθε καταχωρητής είναι ένας πίνακας αποτελούμενος από b bits, αριθμημένα από 0 (το δεξιότερο bit) μέχρι $b - 1$ (το αριστερότερο bit). Για κάθε i $(0 \leq i \leq m-1)$ και κάθε j $(0 \leq j \leq b-1)$, συμβολίζουμε το j-οστό bit του i-οστού καταχωρητή με $r[i][j]$.

Για κάθε ακολουθία από bits $d_0, d_1, \ldots, d_{l-1}$ (αυθαίρετου μήκους l), η **ακέραια τιμή** της ακολουθίας ισούται με $2^0 \cdot d_0 + 2^1 \cdot d_1 + \ldots + 2^{l-1} \cdot d_{l-1}$. Λέμε ότι η **ακέραια τιμή που είναι αποθηκευμένη στον καταχωρητή** i είναι η ακέραια τιμή της ακολουθίας των bits του καταχωρητή, δηλαδή $2^0 \cdot r[i][0] + 2^1 \cdot r[i][1] + \ldots + 2^{b-1} \cdot r[i][b-1]$.

Ο επεξεργαστής έχει 9 είδη **εντολών** που τροποποιούν τις τιμές των καταχωρητών. Κάθε εντολή ενεργεί πάνω σε έναν ή περισσότερους καταχωρητές και αποθηκεύει το αποτέλεσμα της σε έναν καταχωρητή. Στη συνέχεια, θα συμβολίζουμε με $x := y$ την αλλαγή της τιμής του x έτσι ώστε να γίνει ίσο με y. Η λειτουργία των εντολών περιγράφεται παρακάτω.

- $move(t, y)$: Αντιγράφει τον πίνακα των bits του καταχωρητή y στον καταχωρητή t. Για κάθε j $(0 \leq j \leq b-1)$, κάνει $r[t][j] := r[y][j]$.
- $store(t, v)$: Κάνει την τιμή του καταχωρητή t ίση με v, όπου v είναι ένας πίνακας από b bits. Για κάθε j $(0 \leq j \leq b-1)$, κάνει $r[t][j] := v[j]$.
- $and(t, x, y)$: Υπολογίζει το bitwise-AND των καταχωρητών x και y, και αποθηκεύει το αποτέλεσμα στον καταχωρητή t. Για κάθε j $(0 \leq j \leq b-1)$, κάνει $r[t][j] := 1$ αν **και τα δύο** $r[x][j]$ και $r[y][j]$ είναι ίσα με 1, διαφορετικά κάνει $r[t][j] := 0$.
- $or(t, x, y)$: Υπολογίζει το bitwise-OR των καταχωρητών x και y, και αποθηκεύει το αποτέλεσμα στον καταχωρητή t. Για κάθε j $(0 \leq j \leq b-1)$, κάνει $r[t][j] := 1$ αν **τουλάχιστον ένα** από τα $r[x][j]$ και $r[y][j]$ είναι ίσο με 1, διαφορετικά κάνει $r[t][j] := 0$.
- $xor(t, x, y)$: Υπολογίζει το bitwise-XOR των καταχωρητών x και y, και αποθηκεύει το αποτέλεσμα στον καταχωρητή t. Για κάθε j $(0 \leq j \leq b-1)$, κάνει $r[t][j] := 1$ αν **ακριβώς ένα** από τα $r[x][j]$ και $r[y][j]$ είναι ίσο με 1, διαφορετικά κάνει $r[t][j] := 0$.
- $not(t, x)$: Υπολογίζει το bitwise-NOT του καταχωρητή x, και αποθηκεύει το αποτέλεσμα στον καταχωρητή t. Για κάθε j $(0 \leq j \leq b-1)$, κάνει $r[t][j] := 1 - r[x][j]$.
- $left(t, x, p)$: Ολισθαίνει όλα τα bits του καταχωρητή x προς τα αριστερά κατά p θέσεις, και αποθηκεύει το αποτέλεσμα στον καταχωρητή t. Το αποτέλεσμα της ολίσθησης των bits του

καταχωρητή x προς τα αριστερά κατά p θέσεις είναι ένας πίνακας v αποτελούμενος από b bits. Για κάθε j $(0 \leq j \leq b-1)$, είναι $v[j] = r[x][j-p]$ αν $j \geq p$, διαφορετικά $v[j] = 0$. Για κάθε j $(0 \leq j \leq b-1)$, κάνει $r[t][j] := v[j]$.

- $right(t, x, p)$: Ολισθαίνει όλα τα bits του καταχωρητή x προς τα δεξιά κατά p θέσεις, και αποθηκεύει το αποτέλεσμα στον καταχωρητή t. Το αποτέλεσμα της ολίσθησης των bits του καταχωρητή x προς τα δεξιά κατά p θέσεις είναι ένας πίνακας v αποτελούμενος από b bits. Για κάθε j $(0 \leq j \leq b-1)$, είναι $v[j] = r[x][j+p]$ αν $j \leq b-1-p$, διαφορετικά $v[j] = 0$. Για κάθε j $(0 \leq j \leq b-1)$, κάνει $r[t][j] := v[j]$.

- $add(t, x, y)$: Προσθέτει τις ακέραιες τιμές των καταχωρητών x και y, και αποθηκεύει το αποτέλεσμα στον καταχωρητή t. Η πρόσθεση γίνεται σε modulo 2^b. Πιο τυπικά, έστω X η ακέραια τιμή που είναι αποθηκευμένη στον καταχωρητή x, και Y η ακέραια τιμή που είναι αποθηκευμένη στον καταχωρητή y πριν την εκτέλεση της εντολής. Έστω T η ακέραια τιμή που είναι αποθηκευμένη στον καταχωρητή t μετά την εκτέλεση της εντολής. Αν $X + Y < 2^b$, κάνει τα bits του t έτσι ώστε να είναι $T = X + Y$. Διαφορετικά, κάνει τα bits του t έτσι ώστε να είναι $T = X + Y - 2^b$.

Ο Χριστόφορος θέλει να λύσετε δύο είδη προβλημάτων χρησιμοποιώντας τον καινούργιο επεξεργαστή. Το είδος του προβλήματος συμβολίζεται με έναν ακέραιο s. Και για τα δύο είδη προβλημάτων, πρέπει να κατασκευάσετε ένα **program**, δηλαδή μία ακολουθία εντολών όπως αυτές ορίζονται παραπάνω.

Η **είσοδος** του προγράμματος αποτελείται από n ακέραιους $a[0], a[1], \ldots, a[n-1]$, καθένας από τους οποίους έχει k bits, δηλαδή $a[i] < 2^k$ ($0 \leq i \leq n-1$). Πριν την εκτέλεση του προγράμματος, όλοι οι αριθμοί της εισόδου αποθηκεύονται κατά σειρά στον καταχωρητή 0, έτσι ώστε για κάθε i $(0 \leq i \leq n-1)$ η ακέραια τιμή της ακολουθίας των k bits $r[0][i \cdot k], r[0][i \cdot k + 1], \ldots, r[0][(i+1) \cdot k - 1]$ να είναι ίση με $a[i]$. Προσέξτε ότι $n \cdot k \leq b$. Όλα τα υπόλοιπα bits του καταχωρητή 0 (δηλαδή αυτά με δείκτες μεταξύ $n \cdot k$ και $b-1$, συμπεριλαμβανομένων) και όλα τα bits όλων των άλλων καταχωρητών είναι αρχικά ίσα με 0.

Η εκτέλεση ενός προγράμματος γίνεται εκτελώντας τις εντολές του κατά σειρά. Μετά την εκτέλεση της τελευταίας εντολής, η **έξοδος** του προγράμματος υπολογίζεται βάσει της τελικής τιμής των bits του καταχωρητή 0. Συγκεκριμένα, η έξοδος είναι μία ακολουθία n ακεραίων $c[0], c[1], \ldots, c[n-1]$, όπου για κάθε i ($0 \leq i \leq n-1$), το $c[i]$ είναι η ακέραια τιμή της ακολουθίας των k bits από $i \cdot k$ μέχρι $(i+1) \cdot k - 1$ του καταχωρητή 0. Προσέξτε ότι μετά την εκτέλεση του προγράμματος, τα υπόλοιπα bits του καταχωρητή 0 (με δείκτες μεγαλύτερους ή ίσους του $n \cdot k$) και όλα τα bits όλων των άλλων καταχωρητών μπορούν να περιέχουν αυθαίρετες τιμές.

- Το πρώτο είδος προβλήματος $(s = 0)$ είναι να βρείτε τον ελάχιστο ακέραιο μεταξύ των ακεραίων της εισόδου $a[0], a[1], \ldots, a[n-1]$. Συγκεκριμένα, το $c[0]$ πρέπει να είναι ίσο με την ελάχιστη τιμή των of $a[0], a[1], \ldots, a[n-1]$. Οι τιμές των $c[1], c[2], \ldots, c[n-1]$ μπορούν να είναι αυθαίρετες.

- Το δεύτερο είδος προβλήματος $(s = 1)$ είναι να ταξινομήσετε τους ακέραιους της εισόδου $a[0], a[1], \ldots, a[n-1]$ σε μη φθίνουσα σειρά. Συγκεκριμένα, για κάθε i ($0 \leq i \leq n-1$), το $c[i]$ πρέπει να είναι ίσο με τον $1+i$-οστό μικρότερο ακέραιο μεταξύ των

$a[0], a[1], \ldots, a[n-1]$ (δηλαδή το $c[0]$ θα έχει τη μικρότερη τιμή μεταξύ των ακεραίων της εισόδου).

Δώστε στον Χριστόφορο προγράμματα, αποτελούμενα από το πολύ q εντολές το καθένα, που να λύνουν τα παραπάνω προβλήματα.

Λεπτομέρειες υλοποίησης

Πρέπει να υλοποιήσετε την παρακάτω συνάρτηση:

```
void construct_instructions(int s, int n, int k, int q)
```

- s: το είδος του προβλήματος.
- n: το πλήθος των αριθμών της εισόδου.
- k: το πλήθος των bits κάθε αριθμού της εισόδου.
- q: το μέγιστο πλήθος εντολών που επιτρέπονται.
- Η συνάρτηση αυτή καλείται ακριβώς μία φορά και πρέπει να κατασκευάζει μία ακολουθία εντολών για τη λύση του προβλήματος.

Η συνάρτηση αυτή μπορεί να καλεί μία ή περισσότερες από τις ακόλουθες συναρτήσεις για να κατασκευάσει την ακολουθία των εντολών:

```
void append_move(int t, int y)
void append_store(int t, bool[] v)
void append_and(int t, int x, int y)
void append_or(int t, int x, int y)
void append_xor(int t, int x, int y)
void append_not(int t, int x)
void append_left(int t, int x, int p)
void append_right(int t, int x, int p)
void append_add(int t, int x, int y)
```

- Κάθε συνάρτηση προσθέτει μία εντολή $move(t, y)$ $store(t, v)$, $and(t, x, y)$, $or(t, x, y)$, $xor(t, x, y)$, $not(t, x)$, $left(t, x, p)$, $right(t, x, p)$ or $add(t, x, y)$ στο πρόγραμμα, αντίστοιχα.
- Για όλες τις σχετικές εντολές, τα t, x, y πρέπει να είναι μεταξύ 0 και $m-1$.
- Για όλες τις σχετικές εντολές, τα t, x, y δεν είναι κατ' ανάγκη διαφορετικά ανά δύο.
- Για τις εντολές $left$ και $right$, το p πρέπει να είναι μεταξύ 0 και b.
- Για την εντολή $store$, το μήκος του v πρέπει να είναι ίσο με b.

Μπορείτε επίσης να καλείτε την παρακάτω συνάρτηση για να ελέγξετε τη λύση σας:

```
void append_print(int t)
```

- Οι κλήσεις σε αυτή τη συνάρτηση θα αγνοούνται κατά τη βαθμολόγηση της λύσης σας.

- Στον υποδειγματικό βαθμολογητή, αυτή η συνάρτηση προσθέτει μία εντολή $print(t)$ στο πρόγραμμά σας.
- Όταν ο υποδειγματικός βαθμολογητής συναντήσει μία εντολή $print(t)$ κατά την εκτέλεση του προγράμματος, εκτυπώνει n ακέραιους των k bit που σχηματίζονται από τα πρώτα $n \cdot k$ bits του καταχωρητή t (βλ. την ενότητα "Υποδειγματικός βαθμολογητής" για λεπτομέρειες).
- Η τιμή του t πρέπει να ικανοποιεί τη σχέση $0 \leq t \leq m - 1$.
- Η κλήση αυτής της συνάρτησης δεν αυξάνει τον αριθμό των εντολών του προγράμματος.

Μετά την προσθήκη της τελευταίας εντολής, η συνάρτηση `construct_instructions` πρέπει να επιστρέφει. Το πρόγραμμα στη συνέχεια ελέγχεται με μια σειρά περιπτώσεων ελέγχου (test cases), κάθε μία από τις οποίες ορίζει μία είσοδο αποτελούμενη από n ακέραιους των k bit $a[0], a[1], \ldots, a[n-1]$. Η λύση σας περνάει την περίπτωση ελέγχου αν η έξοδος του προγράμματός σας $c[0], c[1], \ldots, c[n-1]$ για την αντίστοιχη είσοδο ικανοποιεί τις παρακάτω συνθήκες:

- Αν $s = 0$, το $c[0]$ πρέπει να είναι η ελάχιστη τιμή μεταξύ των $a[0], a[1], \ldots, a[n-1]$.
- Αν $s = 1$, για κάθε i ($0 \leq i \leq n-1$), το $c[i]$ πρέπει να είναι ο $1+i$-οστός μικρότερος αριθμός μεταξύ των $a[0], a[1], \ldots, a[n-1]$.

Η βαθμολόγηση της λύσης σας μπορεί να οδηγήσει σε κάποιο από τα παρακάτω μηνύματα σφάλματος:

- `Invalid index`: ένας εσφαλμένος (πιθανώς αρνητικός) αριθμός καταχωρητή δόθηκε στην παράμετρο t, x ή y κάποια κλήσης μίας εκ των συναρτήσεων.
- `Value to store is not b bits long`: το μήκος του v που δόθηκε σε κάποια κλήση της `append_store` δεν είναι ίσο με b.
- `Invalid shift value`: η τιμή του p που δόθηκε σε κάποια κλήση της `append_left` ή της `append_right` δεν είναι μεταξύ 0 και b, συμπεριλαμβανομένων.
- `Too many instructions`: η συνάρτησή σας προσπάθησε να προσθέσει περισσότερες από q εντολές στο πρόγραμμα.

Παραδείγματα

Παράδειγμα 1

Έστω $s = 0$, $n = 2$, $k = 1$, $q = 1000$. Υπάρχουν δύο ακέραιοι στη είσοδο, $a[0]$ και $a[1]$, και καθένας έχει $k = 1$ bit. Πριν εκτελεστεί το πρόγραμμα, $r[0][0] = a[0]$ και $r[0][1] = a[1]$. Όλα τα άλλα bits στον επεξεργαστή είναι αρχικά ίσα με 0. Μετά την εκτέλεση των εντολών του προγράμματος, πρέπει να είναι $c[0] = r[0][0] = \min(a[0], a[1])$, δηλαδή η ελάχιστη τιμή μεταξύ των $a[0]$ και $a[1]$.

Υπάρχουν μόνο 4 δυνατές είσοδοι για αυτό το πρόγραμμα:

- Περίπτωση 1: $a[0] = 0, a[1] = 0$
- Περίπτωση 2: $a[0] = 0, a[1] = 1$
- Περίπτωση 3: $a[0] = 1, a[1] = 0$
- Περίπτωση 4: $a[0] = 1, a[1] = 1$

Παρατηρούμε ότι και για τις 4 περιπτώσεις, το $\min(a[0], a[1])$ είναι ίσο με το bitwise-AND των $a[0]$ και $a[1]$. Επομένως, μία δυνατή λύση είναι να κατασκευάσουμε ένα πρόγραμμα που να κάνει τις ακόλουθες κλήσεις:

1. `append_move(1, 0)`, προσθέτει μία εντολή αντιγραφής του $r[0]$ στον $r[1]$.
2. `append_right(1, 1, 1)`, προσθέτει μία εντολή που παίρνει όλα τα bits του $r[1]$, τα ολισθαίνει δεξιά κατά 1 bit, και αποθηκεύει το αποτέλεσμα ξανά στο $r[1]$. Αφού κάθε ακέραιος έχει μήκος 1 bit, αυτό έχει ως αποτέλεσμα το $r[1][0]$ να είναι ίσο με $a[1]$.
3. `append_and(0, 0, 1)`, προσθέτει μία εντολή που υπολογίζει το bitwise-AND των $r[0]$ και $r[1]$, και αποθηκεύει το αποτέλεσμα ξανά στο $r[0]$. Μετά την εκτέλεσή της, το $r[0][0]$ ισούται με το bitwise-AND των $r[0][0]$ και $r[1][0]$, που ισούται με το bitwise-AND των $a[0]$ και $a[1]$, όπως θέλαμε.

Παράδειγμα 2

Έστω $s = 1,\ n = 2,\ k = 1,\ q = 1000$. Όπως και στο προηγούμενο παράδειγμα, υπάρχουν μόνο 4 δυνατές είσοδοι στο πρόγραμμα. Και για τις 4, το $\min(a[0], a[1])$ ισούται με το bitwise-AND των $a[0]$ και $a[1]$, και το $\max(a[0], a[1])$ ισούται με το bitwise-OR των $a[0]$ και $a[1]$. Μία δυνατή λύση είναι να κάνουμε τις παρακάτω κλήσεις:

1. `append_move(1,0)`
2. `append_right(1,1,1)`
3. `append_and(2,0,1)`
4. `append_or(3,0,1)`
5. `append_left(3,3,1)`
6. `append_or(0,2,3)`

Μετά την εκτέλεση αυτών των εντολών, το $c[0] = r[0][0]$ περιέχει το $\min(a[0], a[1])$, και το $c[1] = r[0][1]$ περιέχει το $\max(a[0], a[1])$, επομένως η είσοδος έχει ταξινομηθεί.

Περιορισμοί

- $m = 100$
- $b = 2000$
- $0 \leq s \leq 1$
- $2 \leq n \leq 100$
- $1 \leq k \leq 10$
- $q \leq 4000$
- $0 \leq a[i] \leq 2^k - 1$ (για κάθε $0 \leq i \leq n - 1$)

Υποπροβλήματα

1. (10 βαθμοί) $s = 0, n = 2, k \leq 2, q = 1000$
2. (11 βαθμοί) $s = 0, n = 2, k \leq 2, q = 20$
3. (12 βαθμοί) $s = 0, q = 4000$
4. (25 βαθμοί) $s = 0, q = 150$

5. (13 βαθμοί) $s = 1, n \leq 10, q = 4000$
6. (29 βαθμοί) $s = 1, q = 4000$

Υποδειγματικός βαθμολογητής

Ο υποδειγματικός βαθμολογητής διαβάζει την είσοδο ως εξής:

- γραμμή 1 : $s\ n\ k\ q$

Αυτή ακολουθείται από κάποιο πλήθος γραμμών, κάθε μία από τις οποίες περιγράφει μία περίπτωση ελέγχου. Κάθε περίπτωση ελέγχου δίνεται στην εξής μορφή:

- $a[0]\ a[1]\ \ldots\ a[n-1]$

Αυτό περιγράφει μία περίπτωση ελέγχου αποτελούμενη από n ακέραιους $a[0], a[1], ..., a[n-1]$. Η περιγραφή όλων των περιπτώσεων ελέγχου ακολουθείται από μία γραμμή που περιέχει μόνο τον αριθμό -1.

Ο υποδειγματικός βαθμολογητής καλεί πρώτα την `construct_instructions(s, n, k, q)`. Αν αυτή η κλήση παραβιάζει κάποια από τις συνθήκες που περιγράφονται παραπάνω, ο υποδειγματικός βαθμολογητής τυπώνει ένα από τα μηνύματα σφάλματος που περιγράφηκαν παραπάνω και τερματίζει. Διαφορετικά, εκτυπώνει κατά σειρά τις εντολές που προστέθηκαν στο πρόγραμμα. Για τις εντολές $store$, η τιμή του v τυπώνεται από το δείκτη 0 προς το δείκτη $b-1$.

Στη συνέχεια, ο υποδειγματικός βαθμολογητής επεξεργάζεται τις περιπτώσεις ελέγχου κατά σειρά. Για κάθε μία, εκτελεί το πρόγραμμα που έχει κατασκευαστεί με την αντίστοιχη είσοδο.

Για κάθε εντολή $print(t)$, έστω $d[0], d[1], \ldots, d[n-1]$ μία ακολουθία ακεραίων, τέτοιων ώστε για κάθε i ($0 \leq i \leq n-1$), το $d[i]$ να είναι η ακέραια τιμή της ακολουθίας των bits από $i \cdot k$ μέχρι $(i+1) \cdot k - 1$ του καταχωρητή t (όταν η εντολή εκτελείται). Ο υποδειγματικός βαθμολογητής τυπώνει αυτή την ακολουθία ως εξής: `register` t: $d[0]\ d[1]\ \ldots\ d[n-1]$.

Όταν εκτελεστούν όλες οι εντολές, ο υποδειγματικός βαθμολογητής τυπώνει την έξοδο του προγράμματος.

Αν $s = 0$, η έξοδος του υποδειγματικού βαθμολογητή για κάθε περίπτωση ελέγχου έχει την εξής μορφή:

- $c[0]$.

Αν $s = 1$, η έξοδος του υποδειγματικού βαθμολογητή για κάθε περίπτωση ελέγχου έχει την εξής μορφή:

- $c[0]\ c[1]\ \ldots\ c[n-1]$.

Αφού εκτελεστούν όλες οι περιπτώσεις ελέγχου, ο υποδειγματικός βαθμολογητής τυπώνει `number of instructions:` X όπου X είναι το πλήθος των εντολών του προγράμματός σας.