International Olympiad in Informatics 2016
12-19th August 2016
Kazan, Russia
day2_2

messy
Country: GRC

Διορθώνοντας το σφάλμα

Ο Παντελής (Ilshat) είναι προγραμματιστής και έχει μόλις ανακαλύψει μια νέα, αποδοτική δομή δεδομένων. Η συγκεκριμένη δομή μπορεί να αποθηκεύει ένα σύνολο από *μη-αρνητικούς* ακέραιους, μεγέθους n-δυαδικών ψηφίων (bits), όπου το n είναι ακέραιος υψωμένος σε δύναμη του δύο. Με άλλα λόγια, $n = 2^b$ για τυχαίο, μη-αρνητικό ακέραιο b.

Η δομή είναι αρχικά άδεια. Ένα πρόγραμμα, που χρησιμοποιεί τη δομή πρέπει να ακολουθεί τους εξής κανόνες:

- Το πρόγραμμα μπορεί να καταχωρεί στοιχεία στη δομή, τα οποία να είναι ακέραιοι αριθμοί μεγέθους n-δυαδικών ψηφίων (bits), ένα μόνο στοιχείο κάθε φορά, χρησιμοποιώντας τη συνάρτηση `add_element(x)`. Αν το πρόγραμμα προσπαθήσει να καταχωρίσει ένα στοιχείο, το οποίο υπάρχει ήδη στη δομή, τίποτα δεν θα συμβεί.
- Μετά που θα καταχωρίσει και το τελευταίο στοιχείο, το πρόγραμμα πρέπει να καλέσει τη συνάρτηση `compile_set()` ακριβώς μία φορά.
- Τέλος, το πρόγραμμα μπορεί να καλέσει τη συνάρτηση `check_element(x)` για να ελέγξει αν το στοιχείο x υπάρχει ήδη στη δομή. Αυτή η συνάρτηση μπορεί να κληθεί πολλές φορές.

Όταν ο Παντελής υλοποίησε αρχικά τη δομή, έκανε ένα σφάλμα (bug) στη συνάρτηση `compile_set()`. Το σφάλμα αλλάζει τη σειρά των δυαδικών ψηφίων κάθε στοιχείου της δομής, με τον ίδιο ακριβώς τρόπο. Ο Παντελής θέλει να γνωρίζει τον ακριβή τρόπο με τον οποίο αλλάζει τη σειρά των στοιχείων αυτό το σφάλμα.

Έστω ότι έχουμε την ακολουθία $p = [p_0, \ldots, p_{n-1}]$, όπου κάθε αριθμός από το 0 μέχρι το $n-1$ εμφανίζεται μόνο μία φορά. Καλούμε αυτή την ακολουθία *αντιμετάθεση (permutation)*. Έστω ότι έχουμε ένα στοιχείο του συνόλου (set), του οποίου τα δυαδικά ψηφία είναι $a_0, \ldots, a_{n-1}$ (όπου το a_0 είναι το ψηφίο που βρίσκεται στα αριστερά (significant bit)). Όταν κληθεί η συνάρτηση `compile_set()`, αυτό το στοιχείο θα αντικατασταθεί από το στοιχείο $a_{p_0}, a_{p_1}, \ldots, a_{p_{n-1}}$.

Η ίδια αντιμετάθεση p, χρησιμοποιείται για να αλλάξει τη σειρά των ψηφίων κάθε στοιχείου. Όλες οι αντιμεταθέσεις είναι πιθανές, με την πιθανότητα να έχουμε $p_i = i$ για κάθε $0 \le i \le n-1$.

Για παράδειγμα, αν έχουμε $n = 4$, $p = [2, 1, 3, 0]$ και έχετε καταχωρίσει μέσα στο σύνολο κάποιους ακέραιους, των οποίων οι δυαδικές αναπαραστάσεις είναι `0000`, `1100` και `0111`. Όταν κληθεί η συνάρτηση `compile_set`, αυτή τις αλλάζει σε `0000`, `0101` και `1110`, αντίστοιχα.

Να γράψετε πρόγραμμα, το οποίο να βρίσκει την αντιμετάθεση (permutation) p, αλληλεπιδρώντας με τη δομή δεδομένων. Το πρόγραμμα πρέπει (με την εξής σειρά) να:

1. επιλέγει ένα σύνολο από ακέραιους μεγέθους n-δυαδικών ψηφίων (bits),
2. εισάγει αυτούς τους ακέραιους μέσα στη δομή,
3. καλεί τη συνάρτηση `compile_set` για να προκαλέσει το σφάλμα,
4. ελέγξει αν κάποια στοιχεία εμφανίζονται στο τροποποιημένο σύνολο,
5. χρησιμοποιεί αυτές τις πληροφορίες για να εξακριβώσει και να επιστρέψει την αντιμετάθεση p.

Να σημειωθεί ότι το πρόγραμμά σας μπορεί να καλέσει τη συνάρτηση `compile_set`μόνο μία φορά.

Επιπρόσθετα, υπάρχει περιορισμός στο πλήθος των κλήσεων των συναρτήσεων βιβλιοθήκης (library functions). Συγκεκριμένα, μπορεί να:

- καλέσει την `add_element` το πολύ w φορές (w είναι για εγγραφή - "writes"),
- καλέσει την `check_element` το πολύ r φορές (r είναι για ανάγνωση - "reads").

Λεπτομέρειες Υλοποίησης

Να υπολοποιήσετε τη συνάρτηση (method):

- `int[] restore_permutation(int n, int w, int r)`
  - `n`: το πλήθος των ψηφίων (bits) στην δυαδική αναπαράσταση για κάθε στοιχείο του συνόλου (και επίσης το μέγεθος της p).
  - `w`: το μέγιστο πλήθος κλήσεων της συνάρτησης `add_element` που μπορεί να κάνει το πρόγραμμά σας.
  - `r`: ο μέγιστο πλήθος κλήσεων της συνάρτησης `check_element` που μπορεί να κάνει το πρόγραμμά σας.
  - η συνάρτηση πρέπει να επιστρέφει την ανακτημένη αντιμετάθεση p.

Στην γλώσσα C, το πρότυπο της συνάρτησης είναι κάπως διαφορετικό:

- `void restore_permutation(int n, int w, int r, int* result)`
  - `n`, `w` και `r` έχουν την ίδια έννοια όπως παραπάνω.
  - Η συνάρτηση πρέπει να επιστρέφει την ανακτημένη αντιμετάθεση p αποθηκεύοντάς την στον πίνακα `result`: για κάθε i, πρέπει να αποθηκεύει την τιμή p_i στο `result[i]`.

Συναρτήσεις βιβλιοθήκης

Το πρόγραμμά σας για να αλληλεπιδράσει με τη δομή δεδομένων, πρέπει να χρησιμοποιήσει τις τρεις παρακάτω συναρτήσεις (methods):

- `void add_element(string x)`
  Η συνάρτηση προσθέτει στο σύνολο το στοιχείο που περιγράφεται από το `x`.
  - `x`: μια συμβολοσειρά αποτελούμενη από τους χαρακτήρες `'0'` και `'1'`, που δίνει την δυαδική αναπαράσταση ενός ακέραιου, ο οποίος πρέπει να προστεθεί στο σύνολο. Το μέγεθος της `x` πρέπει να είναι n.

- `void compile_set()`
  Η συνάρτηση πρέπει να κληθεί μόνο μία φορά. Το πρόγραμμά σας δεν μπορεί να

καλέσει την `add_element()` μετά από αυτή την κλήση. Το πρόγραμμά σας δεν μπορεί να καλέσει την `check_element()` πριν από αυτή την κλήση.
- `boolean check_element(string x)`
  Η συνάρτηση αυτή ελέγχει εάν το στοιχείο `x` βρίσκεται στο τροποποιημένο σύνολο.
  - `x`: μια συμβολοσειρά αποτελούμενη από τους χαρακτήρες `'0'` και `'1'`, που δίνει την δυαδική αναπαράσταση του ακέραιου που θα ελέγξει. Το μέγεθος της `x` πρέπει να είναι n .
  - επιστρέφει `true` εάν το στοιχείο `x` βρίσκεται στο τροποποιημένο σύνολο, και `false` σε αντίθετη περίπτωση.

Να σημειωθεί ότι αν το πρόγραμμά σας παραβιάζει οποιουσδήποτε από τους πιο πάνω περιορισμούς, το αποτέλεσμα θα είναι "Wrong Answer".

Σε όλες τις συμβολοσειρές, ο πρώτος χαρακτήρας δίνει το σημαντικότερο ψηφίο του αντίστοιχου ακέραιου.

Ο βαθμολογητής διορθώνει την αντιμετάθεση p πριν την κλήση της συνάρτησης `restore_permutation`.

Για λεπτομέρειες υλοποίησης στη γλώσσα προγραμματισμού σας χρησιμοποιήστε τα δοθέντα πρότυπα αρχείων.

Παράδειγμα

Ο βαθμολογητής (grader) θα κάνει την παρακάτω κλήση συνάρτησης:
- `restore_permutation(4, 16, 16)`. Έχουμε $n = 4$ και το πρόγραμμα μπορεί να κάνει το πολύ 16 "εγγραφές (writes)" και 16 "ανάγνώσεις (reads)".

Το πρόγραμμα κάνει τις ακόλουθες κλήσεις συναρτήσεων:
- `add_element("0001")`
- `add_element("0011")`
- `add_element("0100")`
- `compile_set()`
- `check_element("0001")` επιστρέφει `false`
- `check_element("0010")` επιστρέφει `true`
- `check_element("0100")` επιστρέφει `true`
- `check_element("1000")` επιστρέφει `false`
- `check_element("0011")` επιστρέφει `false`
- `check_element("0101")` επιστρέφει `false`
- `check_element("1001")` επιστρέφει `false`
- `check_element("0110")` επιστρέφει `false`
- `check_element("1010")` επιστρέφει `true`
- `check_element("1100")` επιστρέφει `false`

Μόνο μια αντιμετάθεση μπορεί να ταυτιστεί με τις τιμές που επιστρέφει η `check_element()`: η αντιμετάθεση $p = [2, 1, 3, 0]$. Έτσι, η `restore_permutation` πρέπει να επιστρέψει `[2, 1, 3, 0]`.

Υποπροβλήματα

1. (20 βαθμοί) $n = 8$, $w = 256$, $r = 256$, $p_i \neq i$ για το πολύ δύο δείκτες i ($0 \leq i \leq n - 1$),
2. (18 βαθμοί) $n = 32$, $w = 320$, $r = 1024$,
3. (11 βαθμοί) $n = 32$, $w = 1024$, $r = 320$,
4. (21 βαθμοί) $n = 128$, $w = 1792$, $r = 1792$,
5. (30 βαθμοί) $n = 128$, $w = 896$, $r = 896$.

Υπόδειγμα βαθμολογητή

Το υπόδειγμα βαθμολογητή που σας δίνεται, διαβάζει την είσοδο του με την παρακάτω μορφή:

- γραμμή 1: ακέραιοι n, w, r,
- γραμμή 2: n ακέραιοι που δείχνουν τα στοιχεία της p.