

Elevator

Χρονικό όριο: 10 δευτερόλεπτα

Όριο μνήμης: 64 MiB (ανά διεργασία)

Αυτό είναι ένα πρόβλημα επικοινωνίας.

Υπάρχει ένα κτίριο με ορόφους αριθμημένους από 0 έως N . Ο όροφος 0 είναι το ισόγειο και πάνω από τον όροφο N είναι η ταράτσα. Έτσι, συμπεριλαμβανομένης της ταράτσας, το κτίριο έχει $N + 2$ ορόφους.

Κάθε όροφος f ($0 \leq f \leq N$) κατοικείται από ακριβώς έναν κάτοικο και έχει έναν μυστικό ακέραιο αριθμό v_f ($0 \leq v_f \leq 3$), γνωστό μόνο στον κάτοικο του συγκεκριμένου ορόφου. Επίσης υπάρχει ο κύριος Karlsson που ζει στην ταράτσα του κτιρίου. Ο στόχος είναι ο Karlsson να βρει όσες περισσότερες από τις τιμές $v_0, v_1, \dots, v_N$ μπορεί, ενώ κάθε κάτοικος **συνεργάζεται** για να μεγιστοποιηθεί ο αριθμός των τιμών που μπορεί να βρει.

Για να επικοινωνούν οι κάτοικοι χρησιμοποιούν ένα ασανσέρ του κτιρίου με τον ακόλουθο τρόπο.

Το ασανσέρ ξεκινά από τον όροφο 0 και ταξιδεύει μόνο προς τα πάνω. Μέσα στο ασανσέρ υπάρχουν αριθμημένα κουμπιά για τους ορόφους, από το 1 έως και το N . Αρχικά δεν υπάρχουν πατημένα κουμπιά. Όταν πατηθεί ένα κουμπί, παραμένει πατημένο για πάντα. Το ασανσέρ σταματά και ανοίγει στον όροφο f αν $f = 0$ ή αν το κουμπί που αντιστοιχεί στον όροφο f έχει πατηθεί από προηγούμενο σταμάτημα. Αφού φτάσει στον ψηλότερο όροφο του οποίου το κουμπί έχει πατηθεί (ή τον όροφο 0 εάν δεν έχει πατηθεί ποτέ κάποιο κουμπί), το ασανσέρ συνεχίζει απευθείας προς την ταράτσα χωρίς να σταματά σε κάποιον από τους υπόλοιπους ορόφους.

Όταν το ασανσέρ σταματήσει στον όροφο f συμβαίνουν τα επόμενα:

1. Ο κάτοικος βλέπει το πλήρες σύνολο των κουμπιών που είναι πατημένα εκείνη τη στιγμή.
2. Βασισμένος μόνο σε αυτές τις πληροφορίες και στην τιμή v_f , ο κάτοικος μπορεί να επιλέξει οποιοδήποτε υποσύνολο των κουμπιών που δεν είναι πατημένα αυτή τη στιγμή που αντιστοιχούν σε ορόφους **αυστηρά πάνω** από τον δικό του όροφο. Ο κάτοικος πατά αυτά τα κουμπιά.
3. Το ασανσέρ ανεβαίνει στον επόμενο χαμηλότερο όροφο για τον οποίο το αντίστοιχο κουμπί ήταν πατημένο (ή στην ταράτσα αν το ασανσέρ έχει επισκεφτεί όλους τους ορόφους που αντιστοιχούν σε πατημένα κουμπιά).

Αν το ασανσέρ δεν σταματήσει σε κάποιον όροφο, ο αντίστοιχος κάτοικος δεν μπορεί να πατήσει κάποιο κουμπί.

Όταν το ασανσέρ φτάνει στην ταράτσα, ο Karlsson βλέπει μόνο το τελικό σύνολο των πατημένων κουμπιών. Χρησιμοποιώντας αυτή την πληροφορία, προσπαθεί να βρει όσες περισσότερες τιμές $v_0, v_1, \dots, v_N$ γίνεται.

Οι τιμές του v είναι καθορισμένες από πριν ξεκινήσει το πρόγραμμά σου και δεν αλλάζουν κατά τη διάρκεια της διαδικασίας.

Λεπτομέρειες υλοποίησης

Θα υπάρχουν T περιπτώσεις ελέγχου (test cases).

Υποβάλετε ένα αρχείο που υλοποιεί αυτές τις δύο συναρτήσεις.

Η πρώτη συνάρτηση που πρέπει να υλοποιήσεις είναι η ακόλουθη:

```
std::vector<int> press_buttons(int subtask, int N,  
                               int f, int v, std::vector<int> p)
```

- $subtask$: το υποπρόβλημα για το οποίο γίνεται κλήση αυτής της συνάρτησης, όπου $0 \leq subtask \leq 4$,
- N : ο αριθμός του τελευταίου ορόφου,
- f : ο όροφος που βρίσκεται το ασανσέρ εκείνη τη στιγμή, όπου $0 \leq f \leq N$,
- v : η τιμή v_f που είναι γραμμένη στον όροφο f ,
- p : τα κουμπιά που πατούνται αυτή τη στιγμή σε αύξουσα σειρά.

Αυτή η συνάρτηση καλείται όταν το ασανσέρ ανοίγει στον όροφο f (αν $f = 0$ ή το κουμπί που αντιστοιχεί στον όροφο f έχει πατηθεί σε κάποια προηγούμενη στάση). Αυτή η συνάρτηση δεν θα κληθεί για συγκεκριμένο όροφο αν το αντίστοιχο κουμπί δεν είχε πατηθεί πριν.

Η τιμή που επιστρέφει είναι η λίστα των νέων κουμπιών που θα πατηθούν. Η σειρά των τιμών στον πίνακα που επιστρέφεται δεν έχει σημασία. Κάθε κουμπί x που θα πατηθεί πρέπει να ικανοποιεί τα ακόλουθα:

- $f < x \leq N$,
- το x δεν είναι ήδη στο p και το x εμφανίζεται ακριβώς μία φορά στον πίνακα που θα επιστραφεί.

Η δεύτερη συνάρτηση που πρέπει να υλοποιήσεις είναι η ακόλουθη:

```
std::vector<int> answer(int subtask, int N, std::vector<int> p)
```

- `subtask`: το υποπρόβλημα για το οποίο γίνεται κλήση αυτής της συνάρτησης, όπου $0 \leq \text{subtask} \leq 4$,
- N : ο αριθμός του τελευταίου ορόφου,
- p : το τελικό σύνολο των πατημένων κουμπιών σε αύξουσα σειρά.

Αυτή η συνάρτηση καλείται ακριβώς μία φορά ανά περίπτωση ελέγχου (test case) όταν το ασανσέρ φτάσει στην ταράτσα.

Ο πίνακας επιστροφής πρέπει να έχει μήκος $N + 1$, το i -οστό στοιχείο του θα πρέπει να είναι ίσο με v_i , ενώ οι τιμές που ο Karlsson **δεν μπορεί να βρει** θα πρέπει να αντικατασταθούν με -1 .

Σημαντικό: Οι άνθρωποι στο κτίριο δεν μπορούν να επικοινωνήσουν με κανέναν άλλο τρόπο παρά μόνο μέσω των πατημένων κουμπιών του ασανσέρ. Για να εξασφαλιστεί αυτό, το πρόγραμμά σου θα εκτελεστεί ως $N + 2$ ξεχωριστές διεργασίες (separate processes), μία για κάθε όροφο και μία για την ταράτσα. Σε κάθε διεργασία (process) για τους ορόφους, θα υπάρχουν το πολύ T κλήσεις της συνάρτησης `press_buttons` και στη διεργασία για την ταράτσα – ακριβώς T κλήσεις της συνάρτησης `answer`.

Επομένως, το πρόγραμμά σου δεν θα πρέπει να υποθέτει κάποια κοινή κατάσταση (shared state - για παράδειγμα καθολικές μεταβλητές global variables). Κάθε διεργασία θα έχει 64 MiB μνήμης διαθέσιμη, η οποία είναι ξεχωριστή από τη μνήμη άλλων διεργασιών, και όλες αυτές οι κλήσεις (έως $(N + 1) \times T$ `press_buttons` και ακριβώς T `answer`) πρέπει να ολοκληρωθούν σε 10 δευτερόλεπτα.

Περιορισμοί

- $N = 60$
- $T \leq 10\,000$
- $0 \leq v_i \leq 3$ για κάθε $0 \leq i \leq N$

Παράδειγμα

Το παράδειγμα δοκιμής έχει 1 περίπτωση ελέγχου με τις ακόλουθες τιμές ορόφου:

```
1 1 0 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 0 1 1 0 1 0 1 1 1
1 0 1 1 0 1 1 1 0 1 0 1 0 1 1 0 1 1 1 1 0 1 1 1 0 1 1
```

Ακολουθεί ένα παράδειγμα αλληλεπίδρασης (example interaction):

Το πρόγραμμά σου	Πρόγραμμα του κριτή
	<code>press_buttons(0, 60, 0, 1, {})</code>
<code>return {2}</code>	
	<code>press_buttons(0, 60, 2, 0, {2})</code>
<code>return {13, 42}</code>	
	<code>press_buttons(0, 60, 13, 0, {2, 13, 42})</code>
<code>return {}</code>	
	<code>press_buttons(0, 60, 42, 1, {2, 13, 42})</code>
<code>return {}</code>	
	<code>answer(0, 60, {2, 13, 42})</code>
<code>return {1, -1, 0, -1, -1, ..., -1}</code>	

Αυτή η αλληλεπίδραση βρίσκει την 0-κή και τη 2-η τιμή σωστά. Όλες οι άλλες τιμές που επιστρέφει είναι ίσες με -1 γιατί ο Karlsson δεν τις βρήκε.

Υποπροβλήματα

Υποπρόβλημα	Πόντοι	Πρόσθετοι περιορισμοί	Αριθμός ορόφων που πρέπει να βρει για να πάρει όλους τους πόντους (full points)
0	0	Το παράδειγμα.	–
1	15	$0 \leq v_i \leq 1$ $v_0 = v_N = 1$ αν $v_i = 0$, τότε $v_{i+1} = 1$ για κάθε $0 \leq i < N$	60
2	35	$0 \leq v_i \leq 1$	40
3	15	$0 \leq v_i \leq 2$	30
4	35	$0 \leq v_i \leq 3$	25

Βαθμολόγηση

Εάν το πρόγραμμά σου επιστρέψει μια λάθος ανακτημένη τιμή (recovered value) ή εκτελέσει μια μη έγκυρη λειτουργία (για παράδειγμα, επιστρέψει ένα `vector` λανθασμένου μήκους) σε οποιαδήποτε από τις περιπτώσεις ελέγχου ενός υποπροβλήματος, η υποβολή σου θα λάβει μηδέν πόντους για αυτό το υποπρόβλημα.

Για μια περίπτωση ελέγχου το ανακτημένο πλήθος της υποβολής σου είναι ο αριθμός των θέσεων των οποίων η επιστρεφόμενη τιμή δεν είναι -1 . Έστω K το ελάχιστο ανακτημένο πλήθος μεταξύ όλων των περιπτώσεων ελέγχου ενός υποπροβλήματος (κάθε υποπρόβλημα έχει μόνο 1 δοκιμή που περιέχει έως 10 000 περιπτώσεις ελέγχου). Τότε, οι πόντοι για κάθε υποπρόβλημα υπολογίζονται σύμφωνα με τον παρακάτω πίνακα.

Υποπρόβλημα	Εύρος K	Πόντοι
0	-	0
1	$K < 60$	$0.25K$
	$K \geq 60$	15
2	$K < 30$	$0.5K$
	$30 \leq K < 40$	$2K - 45$
	$K \geq 40$	35
3	$K < 30$	$0.5K$
	$K \geq 30$	15
4	$K < 20$	$0.7K$
	$20 \leq K < 25$	$4K - 66$
	$K \geq 25$	35

Ενδεικτικός Βαθμολογητής

Ο ενδεικτικός βαθμολογητής κάνει όλες τις κλήσεις συναρτήσεων για όλες τις περιπτώσεις ελέγχου σε μία εκτέλεση.

Η μορφή εισόδου είναι η ακόλουθη:

- γραμμή 1: τρεις ακέραιοι – ο αριθμός των περιπτώσεων ελέγχου T , ο αριθμός του τελευταίου ορόφου N και ο αριθμός του υποπροβλήματος S ;
- γραμμή $1 + i$: $N + 1$ ακέραιοι $v_0, v_1, \dots, v_N$, όπου v_f είναι η τιμή που είναι γραμμένη στον όροφο f για την περίπτωση ελέγχου.

Η μορφή εξόδου είναι η ακόλουθη:

- γραμμή i : ένας ακέραιος – ο αριθμός των σωστών τιμών που βρέθηκαν v στην περίπτωση ελέγχου i ;
- γραμμή $T + 1$: τελικοί πόντοι.

Για πιο λεπτομερή πληροφόρηση, άλλαξε το macro `DETAILED` από `false` σε `true` στην πρώτη γραμμή του βαθμολογητή.

Μπορείς να αφήσεις τον βαθμολογητή να δημιουργήσει περιπτώσεις ελέγχου (τιμές του v_f) για εσένα αλλάζοντας το macro `AUTO_GENERATE` από `false` σε `true` στη δεύτερη γραμμή του βαθμολογητή.