

Elevator

Limita de timp: 10 secunde

Limita de memorie: 64 MiB (per proces)

Aceasta este o problemă de comunicare.

Există o clădire cu etaje numerotate de la 0 la N . Etajul 0 este parter, iar deasupra etajului N se află acoperișul. Astfel, incluzând acoperișul, clădirea are $N + 2$ niveluri.

Fiecare etaj f ($0 \leq f \leq N$) este ocupat de exact o persoană și are un număr secret v_f , cunoscut doar de acea persoană. De asemenea, pe acoperișul clădirii locuiește Karlsson. Scopul este ca Karlsson să determine cât mai multe dintre valorile $v_0, v_1, \dots, v_N$ posibile, în timp ce fiecare locuitor cooperează pentru a maximiza numărul de valori pe care Karlsson le poate recupera.

Pentru a comunica, locuitorii folosesc ascensorul clădirii în felul următor.

Ascensorul pornește de la etajul 0 și se deplasează doar în sus. În interiorul ascensorului se află butoane etichetate cu etajele 1 până la N . Odată ce un buton este apăsat, rămâne apăsat până când ascensorul ajunge la acoperiș. Ascensorul se oprește și se deschide la etajul f dacă $f = 0$, sau dacă butonul corespunzător etajului f a fost apăsat la o oprire anterioară. După ce vizitează etajul cel mai înalt al cărui buton a fost apăsat (sau etajul 0 dacă nu sunt apăstate niciodată butoane), ascensorul continuă direct la acoperiș fără să se oprească la niciunul dintre etajele rămase.

Când ascensorul se oprește la etajul f , locuitorul vede setul complet de butoane care sunt apăstate în prezent. Pe baza doar a acestei informații și a valorii v_f , poate apăsa orice subset din butoanele neapăsate în prezent, corespunzând etajelor **strict deasupra** propriului etaj.

Când ascensorul ajunge la acoperiș, Karlsson vede doar setul final de butoane apăstate. Folosind această informație, încearcă să recupereze cât mai multe dintre valorile $v_0, v_1, \dots, v_N$ posibile.

Valorile v sunt fixate înainte ca programul să pornească și nu se modifică în timpul procesului.

Detalii de implementare

Vor fi T cazuri de testare.

Trimiteți un fișier care implementează aceste două funcții.

Prima funcție pe care ar trebui să o implementați este următoarea:

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

- *subtask*: subtask-ul pentru care este apelată această funcție, unde $0 \leq \text{subtask} \leq 4$;
- *N*: numărul ultimului etaj;
- *f*: etajul curent, unde $0 \leq f \leq N$;
- *v*: valoarea v_f scrisă pe etajul *f*;
- *p*: butoanele apăstate în prezent în ordine crescătoare.

Această funcție este apelată atunci când ascensorul se deschide la etajul *f*.

Valoarea returnată este lista butoanelor noi de apăsat. Ordinea valorilor din tabloul returnat nu este importantă. Fiecare buton apăsat *x* trebuie să satisfacă următoarele:

- $f < x \leq N$;
- *x* nu se află deja în *p* și *x* apare cel mult o dată în vectorul returnat.

A doua funcție pe care ar trebui să o implementați este următoarea:

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

- *subtask*: subtask-ul pentru care este apelată această funcție, unde $0 \leq \text{subtask} \leq 4$;
- *N*: numărul ultimului etaj;
- *p*: setul final de butoane apăstate în ordine crescătoare.

Această funcție este apelată exact o dată pe caz de testare atunci când ascensorul ajunge la acoperiș.

Tabloul returnat trebuie să aibă lungimea $N + 1$, elementul *i* trebuie să fie egal cu v_i , în timp ce valorile pe care Karlsson **nu le poate recupera** trebuie înlocuite cu -1 .

Important: Oamenii din clădire nu pot comunica în niciun alt fel decât prin butoanele apăstate. Pentru a asigura acest lucru, programul dvs. va fi executat ca $N + 2$ procese separate, una pentru fiecare etaj și una pentru acoperiș. În fiecare proces pentru etaje, vor fi cel mult *T* apeluri ale funcției `press_buttons` și în procesul pentru acoperiș – exact *T* apeluri ale funcției `answer`. Prin urmare, programul dvs. nu trebuie să presupună nicio stare partajată (de exemplu, variabile globale). Fiecare proces va avea 64 MiB de memorie disponibilă, separată de memoria altor procese, iar toate aceste apeluri (până la $(N + 1) \times T$ apeluri `press_buttons` și exact *T* apeluri `answer`) trebuie să se completeze în 10 secunde.

Restricții

- $N = 60$
- $T \leq 10\,000$
- $0 \leq v_i \leq 3$ pentru fiecare $0 \leq i \leq N$

Exemplu

Testul exemplu are 1 caz de testare cu următoarele valori pe etaje:

```
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 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 1 0 1 1
```

Aici este un exemplu de interacțiune:

Programul participantului	Programul juriului
	<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>	

Această interacțiune recuperează valorile de pe etajele 0 și 2 corect. Toate celelalte valori returnate sunt egale cu -1 deoarece Karlsson nu le-a recuperat.

Subtask-uri

Subtask	Puncte	Constrângeri suplimentare	Numărul de etaje pentru punctaj complet
0	0	Exemplul.	–
1	15	$0 \leq v_i \leq 1$ $v_0 = v_N = 1$ $v_i = 1$ sau $v_{i+1} = 1$ pentru fiecare $0 \leq i < N$	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

Scoring

Dacă programul dvs. returnează o valoare recuperată incorect sau efectuează o operație nevalidă (de exemplu, returnează un `vector` de lungime greșită) în oricare dintre cazurile de testare ale unui subtask, trimiterea dvs. va primi zero puncte pentru acel subtask.

Pentru un caz de testare, *numărul recuperat* al trimiterii dvs. este numărul de poziții a căror valoare returnată nu este -1 . Fie K cel mai mic număr recuperat dintre toate cazurile de testare ale unui subtask (fiecare subtask are doar 1 test cu până la 10 000 cazuri de testare). Apoi, punctele pentru fiecare subtask sunt calculate conform tabelului de mai jos.

Subtask	Domeniu de K	Puncte
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

Grader local

Graderul local efectuează toate apelurile funcției pentru toate cazurile de testare într-o singură execuție.

Formatul de intrare este următorul:

- linia 1: trei numere întregi – numărul cazurilor de testare T , numărul ultimului etaj N și numărul subtask-ului S ;
- linia $1 + i$: $N + 1$ numere întregi $v_0, v_1, \dots, v_N$, unde v_f este valoarea scrisă pe etajul f pentru cazul de testare.

Formatul de ieșire este următorul:

- linia i : un număr întreg – numărul de valori v recuperate corect în cazul de testare i ;
- linia $T + 1$: punctaj final.

Pentru o reacție mai detaliată, schimbați macro-ul `DETAILED` din `false` în `true` în prima linie a evaluatorului.

Puteți lăsa evaluatorul să genereze cazuri de testare pentru dvs. (valori ale v_f) schimbând macro-ul `AUTO_GENERATE` din `false` în `true` în a doua linie a evaluatorului.