

Elevator

Vremensko ograničenje: 10 sekundi

Memorijsko ograničenje: 64 MiB (po procesu)

Ovo je komunikacijski zadatak.

Ima jedna zgrada s katovima označima brojevima od 0 do N . Kat 0 je prizemlje, a iznad kata N nalazi se krov zgrade. Dakle, uključujući krov, zgrada ima $N + 2$ razine.

Na svakom katu f ($0 \leq f \leq N$) živi točno jedan stanar, koji ima tajni cijeli broj v_f ($0 \leq v_f \leq 3$), poznat samo njemu. Također, na krovu ove zgrade živi Krešo. Cilj je da Krešo odredi što je moguće više vrijednosti $v_0, v_1, \dots, v_N$, dok svaki stanar **surađuje** kako bi se maksimizirao broj vrijednosti koje on može otkriti.

Za komunikaciju, stanari koriste dizalo zgrade na sljedeći način.

Dizalo počinje na katu 0 i kreće se samo prema gore. Unutar dizala nalaze se gumbi označeni brojevima od 1 do N . Na početku nijedan gumb nije pritisnut. Kada je gumb jednom pritisnut, ostaje pritisnut zauvijek. Dizalo se zaustavlja i otvara na katu f ako je ili $f = 0$ ili je gumb koji odgovara katu f pritisnut na nekom ranijem zaustavljanju. Nakon posjeta najvišem katu čiji je gumb pritisnut (ili katu 0 ako nijedan gumb nikada nije pritisnut), dizalo nastavlja izravno prema krovu, ne zaustavljajući se na preostalim katovima.

Kada se dizalo zaustavi na katu f , događa se sljedeće:

1. Stanar vidi cjelokupan skup gumba koji su trenutno pritisnuti.
2. Na temelju samo te informacije i vrijednosti v_f , on može odabrati bilo koji podskup trenutno nepritisnutih gumba koji odgovaraju katovima **strogo iznad** njegovog vlastitog kata. Stanar pritišće te gumbe.
3. Dizalo se penje do sljedećeg najnižeg kata za koji je odgovarajući gumb pritisnut (ili do krova ako su posječeni svi katovi koji odgovaraju pritisnutim gumbima).

Ako se dizalo ne zaustavi na nekom katu, odgovarajući stanar ne može pritisnuti nijedan gumb.

Kada dizalo stigne na krov, Krešo vidi samo konačan skup pritisnutih gumba. Koristeći tu informaciju, on pokušava otkriti što je moguće više vrijednosti $v_0, v_1, \dots, v_N$.

Vrijednosti niza v fiksne su prije nego što vaš program počne s radom i ne mijenjaju se tijekom postupka.

Detalji implementacije

Bit će T testnih primjera.

Predajte jednu datoteku koja implementira ove dvije funkcije.

Prva funkcija koju trebate implementirati je sljedeća:

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

- subtask : podzadatak za koji se ova funkcija poziva, gdje je $0 \leq \text{subtask} \leq 4$;
- N : broj posljednjeg kata;
- f : trenutni kat, gdje je $0 \leq f \leq N$;
- v : vrijednost v_f zapisana na katu f ;
- p : trenutno pritisnuti gumbi u rastućem poretku.

Ova se funkcija poziva kada se dizalo otvori na katu f (ako je ili $f = 0$ ili je gumb koji odgovara katu f pritisnut na nekom ranijem zaustavljanju). Ova funkcija neće biti pozvana za određeni kat ako odgovarajući gumb nije prethodno pritisnut.

Povratna vrijednost je popis novih gumba koje treba pritisnuti. Redoslijed vrijednosti u vraćenom nizu nije bitan. Svaki pritisnuti gumb x mora zadovoljavati sljedeće:

- $f < x \leq N$;
- x se već ne nalazi u p i pojavljuje se točno jednom u vraćenom nizu.

Druga funkcija koju trebate implementirati je sljedeća:

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

- subtask : podzadatak za koji se ova funkcija poziva, gdje je $0 \leq \text{subtask} \leq 4$;
- N : broj posljednjeg kata;
- p : konačan skup pritisnutih gumba u rastućem poretku.

Ova se funkcija poziva točno jednom po testnom primjeru, kada dizalo stigne na krov.

Vraćeni niz mora imati duljinu $N + 1$, pri čemu i -ti element mora biti jednak v_i , dok vrijednosti koje Krešo **ne može otkriti** trebaju biti zamijenjene s -1 .

Važno: Ljudi u zgradi ne mogu komunicirati ni na koji drugi način osim putem pritisnutih gumba. Kako bi se to osiguralo, vaš program pokretat će se kao $N + 2$ zasebna procesa, jedan za svaki kat i jedan za krov. U svakom procesu za katove bit će najviše T poziva funkcije `press_buttons`, a u procesu za krov – točno T poziva funkcije `answer`. Stoga vaš program ne smije pretpostavljati

nikakvo dijeljeno stanje (na primjer globalne varijable). Svaki proces imat će na raspolaganju 64 MiB memorije, odvojene od memorije ostalih procesa, a svi ti pozivi (do $(N + 1) \times T$ poziva funkcije `press_buttons` i točno T poziva funkcije `answer`) moraju se izvršiti unutar 10 sekundi.

Ograničenja

- $N = 60$
- $T \leq 10\,000$
- $0 \leq v_i \leq 3$ za svaki $0 \leq i \leq N$

Primjer

Primjer testa ima 1 testni primjer sa sljedećim vrijednostima katova:

```
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
```

Evo primjera interakcije:

Program natjecatelja	Program ocjenjivača
	<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>	

Ova interakcija ispravno otkriva 0-tu i 2-gu vrijednost. Sve ostale vraćene vrijednosti jednake su -1 jer ih Krešo nije otkrio.

Podzadaci

Podzadatak	Bodovi	Dodatna ograničenja	Broj katova koje treba otkriti za pune bodove
0	0	Primjer.	–
1	15	$0 \leq v_i \leq 1$ $v_0 = v_N = 1$ Ako $v_i = 0$ onda je $v_{i+1} = 1$ za svaki $0 \leq i < N$	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

Bodovanje

Ako vaš program vrati netočno otkrivenu vrijednost ili izvrši nevaljanu operaciju (na primjer vrati `vector` pogrešne duljine) u bilo kojem od testnih primjera za neki podzadatak, vaše će rješenje dobiti nula bodova za taj podzadatak.

Za testni primjer, *broj otkrivenih vrijednosti* vašeg rješenja jednak je broju pozicija čija vraćena vrijednost nije -1 . Neka je K najmanji broj otkrivenih vrijednosti među svim testnim primjerima nekog podzadatka (svaki podzadatak ima samo 1 test koji sadrži do 10 000 testnih primjera). Tada se bodovi za svaki podzadatak izračunavaju prema tablici ispod.

Podzadatak	Raspon K	Bodovi
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

Primjer programa ocjenjivača

Primjer ocjenjivača izvršava sve pozive funkcija za sve testne primjere unutar jednog izvođenja.

Format ulaza je sljedeći:

- redak 1: tri cijela broja – broj testnih primjera T , broj posljednjeg kata N i broj podzadatka S ;
- redak $1 + i$: $N + 1$ cijelih brojeva $v_0, v_1, \dots, v_N$, gdje je v_f vrijednost zapisana na katu f za taj testni primjer.

Format izlaza je sljedeći:

- redak i : jedan cijeli broj – broj ispravno otkrivenih vrijednosti niza v u testnom primjeru i ;
- redak $T + 1$: konačni broj bodova.

Za detaljnije povratne informacije, promijenite makro `DETAILED` iz `false` u `true` u prvom retku ocjenjivača.

Ocjenjivaču možete prepustiti generiranje testnih primjera (vrijednosti v_f) postavljanjem makroa `AUTO_GENERATE` iz `false` u `true` u drugom retku ocjenjivača.