

Lifts

Laika ierobežojums: 10 sekundes

Atmiņas ierobežojums: 64 MiB (katram procesam)

Šis ir komunikācijas uzdevums.

Dota ēka, kuras stāvi ir apzīmēti ar skaitļiem no 0 līdz N . 0-tais stāvs ir zemākais stāvs, un virs N -tā stāva ir jumts. Līdz ar to, ieskaitot jumtu, ēkai ir $N + 2$ līmeņi.

Katrā stāvā f ($0 \leq f \leq N$) ir tieši viens iedzīvotājs un slepens vesels skaitlis v_f ($0 \leq v_f \leq 3$), ko zina tikai tā stāva iedzīvotājs. Ēkas jumta līmenī dzīvo Karlsons. Karlsona mērķis ir noteikt pēc iespējas vairāk vērtību $v_0, v_1, \dots, v_N$, turklāt visi ēkas iedzīvotāji **sadarbojas**, lai Karlsons varētu atgūt pēc iespējas vairāk no vērtībām.

Lai savā starpā komunikētu, iedzīvotāji izmanto ēkas liftu šādā veidā.

Lifts sākumā atrodas 0-tajā stāvā un virzās tikai augšup. Lifta iekšpusē ir pogas, kas atbilst stāviem no 1 līdz N . Sākumā visas pogas ir nenospiestā stāvoklī. Kad poga tiek nospiesta, tā neatgriezeniski paliek nospiesta. Lifts apstājas un atver durvis f -tajā stāvā, ja vai nu $f = 0$, vai arī poga, kas atbilst f -tajam stāvam, tika nospiesta kādā no iepriekšējām pieturām. Pēc apstāšanās augstākajā stāvā, kura poga ir nospiesta (vai pēc apstāšanās 0-tajā stāvā, ja neviena poga nekad netiek nospiesta), lifts turpina virzīties līdz jumta līmenim, neapstājoties nevienā no pārējiem stāviem.

Kad lifts apstājas f -tajā stāvā, notiek šādas darbības:

1. Stāva iedzīvotājs redz visas lifta pogas, kas šobrīd ir nospiestas.
2. Balstoties tikai uz šo informāciju un uz vērtību v_f , stāva iedzīvotājs var izvēlēties jebkuru apakškopu no šobrīd nenospiestajām pogām, kas atbilst stāviem, kuri atrodas **stingri virs** f -tā stāva. Stāva iedzīvotājs nospiež šīs pogas.
3. Lifts pārvietojas uz augšu uz nākamo zemāko stāvu, kuram ir nospiesta attiecīgā poga (vai arī uz jumta līmeni, ja visi stāvi, kam atbilst nospiestās pogas, ir tikuši apmeklēti).

Ja lifts kādā stāvā neapstājas, tā stāva iedzīvotājs nevar nospiest nevienu pogu.

Kad lifts sasniedz jumta līmeni, Karlsons redz tikai galīgo nospiesto pogu kopu. Izmantojot šo informāciju, viņš mēģina atgūt pēc iespējas vairāk no vērtībām $v_0, v_1, \dots, v_N$.

Vērtības v ir fiksētas pirms jūsu programmas palaišanas un šī procesa laikā nemainās.

Implementācijas detaļas

Būs T testa gadījumi.

Iesniedziet vienu failu, kurā tiek implementētas šīs divas funkcijas.

Pirmā funkcija, kas jums jāimplementē, ir šāda:

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

- $subtask$: apakšuzdevums, kuram šī funkcija tiek izsaukta, kur $0 \leq subtask \leq 4$;
- N : skaitlis, kas apzīmē augstāko stāvu;
- f : pašreizējais stāvs, kur $0 \leq f \leq N$;
- v : vērtība v_f , kas ierakstīta f -tajā stāvā;
- p : šobrīd nospiestās pogas augošā secībā.

Šī funkcija tiek izsaukta, kad lifts atver durvis f -tajā stāvā (ja vai nu $f = 0$, vai arī poga, kas atbilst f -tajam stāvam, tika nospiesta kādā no iepriekšējām pieturām). Konkrētam stāvam šī funkcija netiks izsaukta, ja tam atbilstošā poga iepriekš nav tikusi nospiesta.

Šai funkcijai ir jāatgriež saraksts ar jaunajām pogām, kuras jānospiež. Vērtību secība atgrieztajā masīvā nav svarīga. Katrai nospiestajai pogai x ir jāizpildās šādiem nosacījumiem:

- $f < x \leq N$;
- x nav viena no p vērtībām un x atgrieztajā masīvā parādās tieši vienu reizi.

Otrā funkcija, kas jums jāimplementē, ir šāda:

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

- $subtask$: apakšuzdevums, kuram šī funkcija tiek izsaukta, kur $0 \leq subtask \leq 4$;
- N : skaitlis, kas apzīmē augstāko stāvu;
- p : galīgā nospiesto pogu kopa augošā secībā.

Šī funkcija tiek izsaukta tieši vienu reizi katrā testa gadījumā, kad lifts sasniedz jumta līmeni.

Šai funkcijai ir jāatgriež masīvs ar garumu $N + 1$, kur i -tajam elementam jābūt vienādam ar v_i . Vērtības v_i , kuras Karlsons **nevar atgūt**, ir jāaizstāj ar -1 .

Svarīgi: Cilvēki ēkā nevar savstarpēji sazināties nekādā citā veidā kā vien ar nospiestajām pogām. Lai to nodrošinātu, jūsu programma tiks palaista kā $N + 2$ atsevišķi procesi — pa vienam katram stāvam un viens jumtam. Katrā stāva procesā funkcija `press_buttons` tiks izsaukta ne vairāk kā T reizes, bet jumta procesā funkcija `answer` — tieši T reizes. Tādēļ jūsu programma nedrīkst

paļauties uz koplietotu stāvokli, piemēram, globāliem mainīgajiem. Katram procesam būs pieejami 64 MiB atmiņas, kas ir nošķirta no pārējo procesu atmiņas, un visiem šiem izsaukumiem (līdz $(N + 1) \times T_{\text{press_buttons}}$ izsaukumiem un tieši T_{answer} izsaukumiem) ir jāizpildās 10 sekunžu laikā.

Ierobežojumi

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

Piemērs

Piemēra testā ir 1 testa gadījums ar šādām stāvu vērtībām:

```
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
```

Šeit ir parādīts mijiedarbības piemērs:

Dalībnieka programma	Žūrijas programma
	<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>	

Šajā mijiedarbībā tiek pareizi atgūtas 0. un 2. vērtība. Visas pārējās atgrieztās vērtības ir vienādas ar -1 , jo Karlsons tās neatguva.

Apakšuzdevumi

Apakšuzdevums	Punkti	Papildu ierobežojumi	Atgūstamo stāvu skaits maksimālajam punktu skaitam
0	0	Piemērs.	–
1	15	$0 \leq v_i \leq 1, v_0 = v_N = 1$ Ja $v_i = 0$, tad $v_{i+1} = 1$ katram $0 \leq i < N$	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

Vērtēšana

Ja jūsu programma kādā no apakšuzdevuma testa gadījumiem atgriež nepareizi atgūtu vērtību vai veic nederīgu darbību (piemēram, atgriež nepareiza garuma `vector`), jūsu iesūtījums par šo apakšuzdevumu saņems 0 punktu.

Testa gadījumā jūsu iesūtījuma *atgūto vērtību skaits* ir to pozīciju skaits, kurās atgrieztā vērtība nav -1 . Ar K apzīmēsim mazāko atgūto vērtību skaitu starp visiem apakšuzdevuma testa gadījumiem (katrā apakšuzdevumā ir tikai 1 tests, kas satur līdz 10 000 testa gadījumiem). Tad punkti par katru apakšuzdevumu tiek aprēķināti saskaņā ar zemāk redzamo tabulu.

Apakšuzdevums	K vērtību diapazons	Punkti
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

Parauga vērtētājprogramma

Parauga vērtētājprogramma veic visus funkciju izsaukumus visiem testa gadījumiem vienā izpildē.

Ievaddatu formāts ir šāds:

- 1. rindā: trīs veseli skaitļi – testa gadījumu skaits T , skaitlis, kas apzīmē augstāko stāvu, N un apakšuzdevuma numurs S ;
- $(1 + i)$. rindā: $N + 1$ veseli skaitļi $v_0, v_1, \dots, v_N$, kur v_f ir vērtība, kas ierakstīta f -tajā stāvā attiecīgajā testa gadījumā.

Izvaddatu formāts ir šāds:

- i . rindā: viens vesels skaitlis – pareizi atgūto v vērtību skaits i -tajā testa gadījumā;
- $(T + 1)$. rindā: galīgais punktu skaits.

Detalizētākai atgriezeniskajai saitei nomainiet makro `DETAILED` vērtību no `false` uz `true` vērtētājprogrammas pirmajā rindā.

Jūs varat likt vērtētājprogrammai pašai ģenerēt testa gadījumus (vērtības v_f), nomainot makro `AUTO_GENERATE` vērtību no `false` uz `true` vērtētājprogrammas otrajā rindā.