

# Lift

Vaxt limiti: 10 saniyə

Yaddaş limiti: 64 MiB (hər proses üçün)

*Bu kommunikasiya məsələsidir.*

0-dan  $N$ -ə qədər nömrələnmiş mərtəbələri olan bir bina var. 0-cı mərtəbə zəmin mərtəbəsidir,  $N$ -ci mərtəbədən yuxarıda isə damdır. Beləliklə, damla birlikdə binada  $N + 2$  səviyyə var.

Hər bir  $f$  mərtəbəsində ( $0 \leq f \leq N$ ) tam olaraq bir sakin yaşayır və onun yalnız o sakinə məlum olan gizli  $v_f$  tam ədədi ( $0 \leq v_f \leq 3$ ) var. Bundan başqa, bu binanın damında Karlsson yaşayır. Məqsəd Karlssonun  $v_0, v_1, \dots, v_N$  qiymətlərindən mümkün qədər çoxunu müəyyən etməsidir, bu zaman hər bir sakin Karlssonun bərpa edə biləcəyi qiymətlərin sayını maksimuma çatdırmaq üçün **əməkdaşlıq edir**.

Rabitə üçün sakinlər binanın liftindən aşağıdakı şəkildə istifadə edirlər.

Lift 0-cı mərtəbədən başlayır və yalnız yuxarıya doğru hərəkət edir. Liftin içərisində 1-dən  $N$ -ə qədər mərtəbələrlə işarələnmiş düymələr var. Əvvəlcə heç bir düyməyə basılmayıb. Bir düyməyə basıldıqdan sonra o, daimi olaraq basılmış vəziyyətdə qalır. Əgər  $f = 0$  olarsa və ya  $f$  mərtəbəsinə uyğun düymə daha əvvəlki bir dayanacaqda basılıbsa, lift  $f$  mərtəbəsində dayanır və qapıları açılır. Basılmış düyməsi olan ən yüksək mərtəbəyə baş çəkdikdən sonra (əgər heç bir düyməyə basılmayıbsa, 0-cı mərtəbədən sonra) lift qalan heç bir mərtəbədə dayanmadan birbaşa dama qalxır.

Lift  $f$  mərtəbəsində dayandıqda aşağıdakılar baş verir:

1. Sakin hazırda basılmış olan düymələrin tam dəstini görür.
2. Yalnız bu məlumata və öz  $v_f$  qiymətinə əsaslanaraq, o, öz mərtəbəsindən **ciddi şəkildə yuxarıda** olan mərtəbələrə uyğun, hazırda basılmamış olan düymələrin istənilən alt dəstini seçə bilər. Sakin bu düymələrə basır.
3. Lift, uyğun düyməsi basılmış olan növbəti ən aşağıdakı mərtəbəyə qalxır (əgər basılmış düymələrə uyğun bütün mərtəbələrə baş çəkilibsə, dama qalxır).

Əgər lift hər hansı bir mərtəbədə dayanmırsa, həmin mərtəbənin sakini heç bir düyməyə basa bilməz.

Lift dama çatdıqda, Karlsson yalnız düymələrin son basılmış dəstini görür. Yalnız bu məlumatdan istifadə edərək, o,  $v_0, v_1, \dots, v_N$  qiymətlərindən mümkün qədər çoxunu bərpa etməyə çalışır.

$v$  qiymətləri proqramınız işə düşməzdən əvvəl müəyyən edilir və proses ərzində dəyişmir.

## Tətbiq detalları

$T$  ədəd test halı olacaq.

Aşağıdakı iki funksiyanı tətbiq edən bir fayl təqdim edin.

Tətbiq etməli olduğunuz birinci funksiya aşağıdakıdır:

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

- $subtask$ : bu funksiyanın çağırıldığı alt-tapşırıq, burada  $0 \leq subtask \leq 4$ ;
- $N$ : son mərtəbənin nömrəsi;
- $f$ : cari mərtəbə, burada  $0 \leq f \leq N$ ;
- $v$ :  $f$  mərtəbəsinin  $v_f$  qiyməti;
- $p$ : artan sırada hazırda basılmış düymələr.

Bu funksiya lift  $f$  mərtəbəsində açıldıqda çağırılır (yəni ya  $f = 0$  olduqda, ya da  $f$  mərtəbəsinə uyğun düymə daha əvvəlki bir dayanacaqda basıldıqda). Əgər müvafiq düyməyə əvvəldən basılmayıbsa, bu funksiya həmin konkret mərtəbə üçün çağırılmayacaq.

Qaytarılan qiymət basılacaq yeni düymələrin siyahısıdır. Qaytarılan massivdəki qiymətlərin sırası önəmli deyil. Hər bir basılmış  $x$  düyməsi aşağıdakı şərtləri ödəməlidir:

- $f < x \leq N$ ;
- $x$  hələ  $p$ -də yoxdur və qaytarılan massivdə tam olaraq bir dəfə görünür.

Tətbiq etməli olduğunuz ikinci funksiya aşağıdakıdır:

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

- $subtask$ : bu funksiyanın çağırıldığı alt-tapşırıq, burada  $0 \leq subtask \leq 4$ ;
- $N$ : son mərtəbənin nömrəsi;
- $p$ : artan sırada basılmış düymələrin son dəsti.

Bu funksiya lift dama çatdıqda hər bir test halı üçün tam olaraq bir dəfə çağırılır.

Qaytarılan massivin uzunluğu  $N + 1$  olmalıdır, onun  $i$ -ci elementi  $v_i$ -ə bərabər olmalıdır, Karlssonun **bərpa edə bilmədiyi** qiymətlər isə  $-1$  ilə əvəz olunmalıdır.

**Vacib qeyd:** Binadakı insanlar basılmış düymələr vasitəsilə olduğundan başqa heç bir üsulla ünsiyyət qura bilməzlər. Bunu təmin etmək üçün proqramınız  $N + 2$  ayrı proses kimi işə salınacaq — hər mərtəbə üçün bir proses və dam üçün bir proses. Mərtəbələr üçün olan hər bir prosesdə

`press_buttons` funksiyası ən çox  $T$  dəfə, dam üçün olan prosesdə isə `answer` funksiyası tam olaraq  $T$  dəfə çağırılacaq. Buna görə də, proqramınız heç bir ümumi vəziyyəti (məsələn, qlobal dəyişənləri) fərz etməməlidir. Hər prosesə digər proseslərin yaddaşından ayrı olan 64 MiB yaddaş ayrılacaq və bütün bu çağırışlar (ən çox  $(N + 1) \times T_{\text{press\_buttons}}$  və tam olaraq  $T_{\text{answer}}$ ) 10 saniyə ərzində tamamlanmalıdır.

## Məhdudiyyətlər

- $N = 60$
- $T \leq 10\,000$
- Hər bir  $0 \leq i \leq N$  üçün  $0 \leq v_i \leq 3$

## Nümunə

Nümunə testdə aşağıdakı mərtəbə qiymətləri ilə 1 test halı var:

```
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
```

Nümunə qarşılıqlı əlaqə aşağıdakı kimidir:

| İştirakçı proqram                               | Münsiflər proqramı                                    |
|-------------------------------------------------|-------------------------------------------------------|
|                                                 | <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> |                                                       |

Bu qarşılıqlı əlaqə 0-cı və 2-ci qiymətləri düzgün bərpa edir. Qaytarılan digər bütün qiymətlər —1-ə bərabərdir, çünki Karlsson onları bərpa edə bilməyib.

## Alt-tapşırıqlar

| Alt-tapşırıq | Xal | Əlavə məhdudiyyətlər                                                                                | Tam xal üçün bərpa edilməli olan mərtəbələrin sayı |
|--------------|-----|-----------------------------------------------------------------------------------------------------|----------------------------------------------------|
| 0            | 0   | Nümunə.                                                                                             | –                                                  |
| 1            | 15  | $0 \leq v_i \leq 1$<br>$v_0 = v_N = 1$<br>Hər bir $0 \leq i < N$ üçün $v_i = 1$ və ya $v_{i+1} = 1$ | 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                                                 |

## Qiymətləndirmə

Əgər proqramınız hər hansı bir alt-tapşırığın test hallarından hər hansı birində düzgün bərpa edilməmiş qiymət qaytarırsa və ya yanlış əməliyyat həyata keçirirsə (məsələn, yanlış uzunluqda `vector` qaytarırsa), göndərişiniz həmin alt-tapşırıq üçün sıfır xal alacaq.

Bir test halı üçün göndərişinizin *bərpa edilmiş sayı* qaytarılan qiyməti —1 olmayan mövqelərin sayıdır. *K* bir alt-tapşırığın bütün test halları arasında minimum bərpa edilmiş say olsun (hər alt-tapşırığın yalnız 1 testi var, o da 10 000-ə qədər test halı ehtiva edir). Onda hər alt-tapşırıq üçün xallar aşağıdakı cədvələ əsasən hesablanır.

| Alt-tapşırıq | $K$ -nın diapazonu | Xal       |
|--------------|--------------------|-----------|
| 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        |

## Nümunə qiymətləndirici (grader)

Nümunə qiymətləndirici bütün test halları üçün bütün funksiya çağırışlarını bir icra daxilində edir.

Giriş formatı aşağıdakı kimidir:

- 1-ci sətir: üç tam ədəd – test hallarının sayı  $T$ , son mərtəbənin nömrəsi  $N$  və alt-tapşırığın nömrəsi  $S$ ;
- $1 + i$ -ci sətir:  $N + 1$  tam ədəd  $v_0, v_1, \dots, v_N$ , burada  $v_f$  test halı  $i$  üçün  $f$  mərtəbəsində yazılmış qiymətdir.

Çıxış formatı aşağıdakı kimidir:

- $i$ -ci sətir: bir tam ədəd –  $i$ -ci test halında düzgün bərpa edilmiş  $v$  qiymətlərinin sayı;
- $T + 1$ -ci sətir: yekun xal.

Daha ətraflı geribildirim üçün, qiymətləndiricidəki `DETAILED` makrosunu `false`-dan `true`-ya dəyişin.

`AUTO_GENERATE` makrosunu `false`-dan `true`-ya dəyişməklə qiymətləndiricinin sizin üçün test halları (yəni  $v_f$  qiymətlərini) yaratmasına icazə verə bilərsiniz.