

Permainan Ruang Kelas

Sebuah sekolah menengah terkemuka di Tashkent merayakan hari jadinya dengan menyelenggarakan suatu permainan antara murid dan guru. Terdapat N murid yang duduk di ruang kelas, dinomori dari 0 hingga $N - 1$. Setiap murid memegang selebar kertas yang berisi *array* bilangan bulat. Awalnya, setiap kertas masih kosong, yang berarti setiap kertas berisi *array* kosong.

Para murid bermain dengan M guru, yang dinomori dari 0 hingga $M - 1$, dan kepala sekolah. Permainan ini berlangsung selama M tahap, yang juga dinomori dari 0 hingga $M - 1$. Pada tahap j , guru j memasuki ruang kelas dan hal-hal berikut terjadi:

- Beberapa (mungkin saja nol) murid mengangkat tangan. Dijamin bahwa pada setiap tahap, setidaknya satu murid **tidak** mengangkat tangan, dan setiap murid hanya boleh mengangkat tangan **paling banyak satu kali** sepanjang permainan.
- Sang guru melihat kertas-kertas yang sedang dipegang para murid, dan dapat **mengubah** isi kertas yang dipegang oleh murid-murid yang **tidak** mengangkat tangan pada tahap ini. Untuk setiap murid yang demikian, sang guru dapat mengganti isi kertas mereka dengan *array* bilangan bulat baru (yang mungkin saja kosong). Setiap bilangan bulat harus bernilai antara 0 dan 63 (inklusif), dan *array* tersebut dapat memiliki paling banyak 63 elemen.
- Setelah pengubahan ini, guru j pergi dan para murid saling menukar kertas mereka berdasarkan suatu permutasi rahasia P . Artinya, setiap murid i ($0 \leq i < N$) memberikan kertasnya kepada murid $P[i]$, dengan $P[0], P[1], \dots, P[N - 1]$ merupakan N bilangan bulat **berbeda** antara 0 dan $N - 1$ (inklusif). Nilai P **tetap** sepanjang seluruh tahap permainan, tetapi para guru dan sang kepala sekolah tidak mengetahuinya.

Setelah M tahap tersebut selesai dan pertukaran terakhir berdasarkan P telah dilakukan, sang kepala sekolah masuk. Hanya dengan melihat kertas yang dipegang para murid, sang kepala sekolah harus menentukan, untuk setiap murid i ($0 \leq i < N$), nomor tahap j ketika murid i mengangkat tangan, atau menyimpulkan bahwa murid i tidak pernah mengangkat tangan.

Para guru tidak dapat berkomunikasi dengan guru lain maupun kepala sekolah, kecuali melalui *array* bilangan bulat yang tertulis pada kertas-kertas. Setiap guru mengetahui nomor tahap ketika guru tersebut memasuki ruang kelas.

Tugas Anda adalah merancang dan menerapkan strategi bagi para guru dan kepala sekolah yang menentukan apakah dan kapan setiap murid mengangkat tangan. Nilai Anda bergantung pada panjang maksimum dari setiap *array* yang ditulis di atas kertas: solusi dengan panjang maksimum yang lebih kecil memperoleh nilai yang sama atau lebih tinggi.

Detail Implementasi

Anda harus mengimplementasikan dua prosedur, satu untuk para guru dan satu untuk sang kepala sekolah.

Prosedur yang harus Anda implementasikan untuk **para guru** adalah:

```
std::vector<std::vector<int>> process_step(  
    int N, int M, int R,  
    std::vector<int> T,  
    std::vector<std::vector<int>> A)
```

- N : banyaknya murid.
- M : banyaknya tahap, yang juga merupakan banyaknya guru.
- R : nomor tahap saat ini (dari 0 hingga $M - 1$).
- T : sebuah *array* (yang mungkin saja kosong) yang berisi nomor murid-murid yang mengangkat tangan pada tahap ini, terurut secara menaik.
- A : sebuah *array* sepanjang N yang menyatakan isi kertas-kertas, dengan $A[i]$ ($0 \leq i < N$) adalah *array* bilangan bulat pada kertas yang dipegang murid i pada awal tahap.
- Prosedur ini dipanggil tepat M kali per permainan, dengan urutan $R = 0, 1, \dots, M - 1$.

Prosedur ini harus mengembalikan *array* B sepanjang N yang menyatakan isi kertas-kertas setelah modifikasi guru, dengan format yang sama dengan A .

- Untuk setiap murid i yang mengangkat tangan (dengan kata lain, i muncul di T), isi kertas tidak boleh diubah, sehingga $B[i] = A[i]$.
- Untuk setiap i sehingga $0 \leq i < N$, panjang $B[i]$ tidak boleh melebihi 63, dan setiap bilangan bulat dalam $B[i]$ harus bernilai antara 0 dan 63 (inklusif).

Prosedur yang harus Anda implementasikan untuk sang **kepala sekolah** adalah:

```
std::vector<int> determine_steps(  
    int N, int M, std::vector<std::vector<int>> A)
```

- N, M : sama seperti di atas.
- A : *array* sepanjang N . $A[i]$ adalah *array* bilangan bulat pada kertas yang dipegang oleh murid i setelah ke- M tahap.
- Prosedur ini dipanggil tepat satu kali per permainan, setelah pemanggilan terakhir ke `process_step`.

Prosedur ini harus mengembalikan *array* D sepanjang N . Untuk setiap i sehingga $0 \leq i < N$, harus berlaku bahwa

- $D[i] = j$ jika murid i mengangkat tangannya pada tahap j , atau

- $D[i] = -1$ jika murid i tidak pernah mengangkat tangan sepanjang permainan.

Program Anda tidak diperbolehkan menyimpan atau mengirim informasi apa pun dari suatu pemanggilan `process_step` ke pemanggilan lainnya, kecuali melalui nilai kembalian dari `process_step`. Jika program Anda mencoba melakukan hal ini, nilai Anda di CMS bisa saja salah dan dapat dikurangi setelah kontes berakhir. Perhatikan bahwa *grader* dapat menjalankan beberapa salinan proses (*instance*) dari program Anda secara bersamaan. Ini berarti pemanggilan `process_step` dan `determine_steps` dari permainan yang sama dapat dijalankan dalam salinan proses (*instance*) yang berbeda, dan pemanggilan `process_step` dan `determine_steps` dari permainan yang berbeda dapat dijalankan dalam salinan proses (*instance*) yang sama. Setiap kasus uji terdiri atas paling banyak 5 permainan.

Batasan

- $2 \leq N \leq 63$
- $1 \leq M \leq 63$
- Setiap murid mengangkat tangan **paling banyak satu kali** selama permainan berlangsung. Dengan kata lain, untuk setiap murid i , terdapat paling banyak satu tahap j sehingga i mengangkat tangan pada tahap tersebut.
- Pada setiap tahap, setidaknya satu murid **tidak** mengangkat tangan.

Subsoal dan Penilaian

Subsoal	Nilai	Batasan Tambahan
1	4	$M = 1$
2	6	$N = 2$
3	9	$P[i] = i$ untuk setiap $0 \leq i \leq N - 1$.
4	25	Terdapat paling banyak satu murid yang mengangkat tangan pada setiap tahap.
5	56	Tidak ada batasan tambahan.

Untuk setiap kasus uji, nilai Anda adalah 0 (dilaporkan sebagai `Output isn't correct` pada CMS) jika nilai kembalian dari suatu pemanggilan `process_step` tidak valid atau jika nilai kembalian dari suatu pemanggilan `determine_steps` salah.

Selebihnya, misalkan C adalah panjang maksimum dari antara **semua** *array* dalam **setiap** B yang dikembalikan oleh `process_step`. Nilai suatu kasus uji dalam subsoal dengan nilai S adalah $S \cdot X$, dengan X bergantung pada C berdasarkan tabel berikut:

Syarat	X
$C \leq 2$	1.00
$C = 3$	0.75
$C = 4$	0.55
$5 \leq C \leq 13$	$0.50 - 0.03 \cdot (C - 5)$
$14 \leq C \leq 63$	$0.19 \cdot \frac{64-C}{64-14} + 0.04$

Contoh

Perhatikan suatu skenario dengan $N = 4$ murid, $M = 2$ guru, dan permutasi $P = [0, 3, 1, 2]$. Awalnya, semua kertas kosong.

Grader pertama-tama memanggil:

```
process_step(4, 2, 0, [1], [[], [], [], []])
```

Murid 1 mengangkat tangan, sehingga isi kertasnya tidak dapat diubah. Guru 0 dapat memutuskan untuk menulis [10] pada kertas murid 0, menulis [1, 63, 4] pada kertas murid 3, dan membiarkan kertas murid 2 kosong. Untuk melakukan ini, prosedur tersebut harus mengembalikan $B = [[10], [], [], [1, 63, 4]]$.

Setelah guru 0 pergi, para murid saling menukar kertas mereka berdasarkan P . Setelah pertukaran, isi kertas-kertas tersebut adalah $A = [[10], [], [1, 63, 4], []]$.

Setelah itu, *grader* memanggil:

```
process_step(4, 2, 1, [0, 3], [[10], [], [1, 63, 4], []])
```

Murid 0 dan 3 mengangkat tangan, sehingga isi kertas mereka tidak dapat diubah. Guru 1 dapat memutuskan untuk menulis [0, 40] pada kertas murid 1 dan menulis [1, 50] pada kertas murid 2. Untuk melakukan ini, prosedur tersebut harus mengembalikan $B = [[10], [0, 40], [1, 50], []]$.

Setelah guru 1 pergi, kertas-kertas tersebut ditukar lagi berdasarkan P . Setelah pertukaran, isi kertas-kertas tersebut adalah $A = [[10], [1, 50], [], [0, 40]]$.

Akhirnya, *grader* memanggil:

```
determine_steps(4, 2, [[10], [1, 50], [], [0, 40]])
```

Prosedur ini harus mengembalikan $D = [1, 0, -1, 1]$, karena murid 0 dan 3 mengangkat tangan pada tahap 1, murid 1 mengangkat tangan pada tahap 0, dan murid 2 tidak pernah mengangkat tangan. Pada contoh ini, $C = 3$.

Contoh Grader

Format masukan:

```
N M
Q[0] Q[1] ... Q[N-1]
P[0] P[1] ... P[N-1]
```

Di sini, Q adalah *array* sepanjang N , dengan $Q[i] = j$ jika murid i mengangkat tangan pada tahap j , atau $Q[i] = -1$ jika murid i tidak pernah mengangkat tangan sepanjang permainan.

Format keluaran:

Setelah setiap pemanggilan `process_step`, contoh *grader* akan mengeluarkan isi kertas-kertas. Misalkan K adalah panjang B dan $L[i]$ adalah panjang $B[i]$ (untuk setiap $0 \leq i < K$). Contoh *grader* mengeluarkan B dengan format berikut, **diikuti sebuah baris kosong**:

```
K
L[0] B[0][0] B[0][1] ... B[0][L[0]-1]
L[1] B[1][0] B[1][1] ... B[1][L[1]-1]
:
L[K-1] B[K-1][0] B[K-1][1] ... B[K-1][L[K-1]-1]
```

Contoh *grader* kemudian menukar kertas-kertas tersebut berdasarkan permutasi P . Jika $K \neq N$, contoh *grader* akan mencetak pesan galat dan berhenti.

Setelah memanggil `determine_steps`, contoh *grader* akan mengeluarkan:

```
C H
D[0] D[1] ... D[H-1]
```

Di sini, H adalah panjang *array* D yang dikembalikan oleh `determine_steps`.